SMNE UFPR
Simpósio de Métodos Numéricos em Engenharia
25 a 27 de outubro, 2017

Identificação de precondições e efeitos em operadores STRIPS a partir de problemas solucionados

Romulo de Oliveira Leite
Volmir Eugenio Wilhelm
Programa de Pós-Graduação em Métodos Numéricos em Engenharia
Universidade Federal do Paraná
Curitiba, Brasil
romulool@ufpr.br, volmirw@gmail.com

***Resumo*—Este trabalho tem por objetivo apresentar um método para identificação de precondições e efeitos em operadores STRIPS empregando um conjunto de problemas corretamente solucionados como informação a priori, no intuito de desenvolver uma ferramenta para auxílio à modelagem de domínios. Os resultados evidenciam a eficácia do método proposto, apesar da verificação de alguns erros, os quais podem ser reparados por meio do emprego de outras técnicas de apoio à modelagem disponíveis na literatura.**

***Palavras-chave*—domínios STRIPS; modelagem; planejamento automatizado.**

I. Introdução

A modelagem de dominios STRIPS é uma tarefa difícil, mesmo nos casos mais simples. Essa dificuldade chamou a atenção de parte da comunidade de pesquisadores em planejamento para o processo de modelagem, especialmente porque muitas das técnicas aplicadas à solução de problemas dependem de modelos bem formulados. Tal dependência representa um gargalo para o deselvolvimento de novas tecnologias na área de planejamento [4].

A construção de modelos a partir de dados é o tema central de diversos trabalhos encontrados na literatura, como [6] e [7]. Nesses trabalhos, um conjunto de observações de problemas corretamente solucionados é usado como base de conhecimento.

Outra fonte de informação útil no processo de modelagem é o conjunto de axiomas do domínio. O conhecimento desses axiomas proporciona uma formulação mais natural e compacta de modelos [3].

Formular modelos de domínios STRIPS equivale a determinar os conjuntos de precondições e efeitos dos operadores que compõem esse domínio. Sendo assim, no intuito de prover uma ferramenta de suporte ao processo de modelagem, o presente trabalho tem por objetivo apresentar um método para a identificação de precondições e efeitos de operadores STRIPS a partir de observações de problemas solucionados e fazendo uso do conhecimento de axiomas do domínio.

As seções seguintes estão assim organizadas: a seção II apresenta as definições de domínios STRIPS e de seus elementos, além da formulação precisa do problema abordado; a seção III descreve em detalhes o método proposto; a seção IV apresenta os resultados de um experimento em que a técnica é aplicada a três domínios *benchmark*, além da discussão concernente a esses resultados; e a seção V apresenta as conclusões do presente trabalho.

II. Definições

Primeiramente, a presente seção apresenta uma série de definições necessárias para a compreensão da técnica proposta e, em seguida, a formulação precisa do problema de identificação de precondições e efeitos em operadores STRIPS.

A. Domínios Strips

Seja L uma linguagem de primeira ordem sem funções definidas e com um número finito de símbolos de predicados e de constantes; um domínio STRIPS sobre L é um sistema estado-transição $\Sigma = (S, A, \gamma)$ em que S é um conjunto finito de estados STRIPS, A é um conjunto finito de instâncias de operadores STRIPS pertencentes a um conjunto O e γ é a função de transição [1].

O conjunto $O = \{o_1, ..., o_k\}$ é tal que cada operador o_i representa um padrão de transição único. Um operador STRIPS $o \in O$ é definido como uma tripla $(nome(o), pre(o), eff(o))$, em que $nome(o)$ é o nome do operador, $pre(o)$ é o conjunto de precondições e $eff(o)$ é o conjunto de efeitos [5]. O nome do operador, $nome(o)$ é uma expressão sintática da forma $\nabla(?z_1, ..., ?z_l)$, em que ∇ é um símbolo único e $(?z_1, ..., ?z_l)$ é uma l-tupla que contém os argumentos ordenados do operador. Ainda, $pre(o)$ e $eff(o)$ são ambos conjuntos de literais não-instanciados.

Para o propósito do presente trabalho, assume-se que modelos de domínios são tipados. Assim, cada símbolo de constante representa um objeto ao qual é designado um único tipo t_i de um conjunto de tipos $T = \{t_1, ..., t_n\}$. Consequentemente, cada variável é associada a um tipo, de modo que uma variável associada ao tipo t_i pode assumir apenas valores correspondentes a objetos do tipo t_i.

Seja $P = \{p_1, ..., p_m\}$ o conjunto de símbolos de predicados; cada elemento $p \in P$ é da forma $p(?w_1, ..., ?w_r)$, em que $(?w_1, ..., ?w_r)$ é uma r-tupla com as variáveis ordenadas que são argumentos de p. Eventualmente, r pode ser igual a zero.

Seja $s \in S$ um estado STRIPS; s é representado como uma conjunção de átomos $\alpha_1 \wedge ... \wedge \alpha_h$. Cada átomo é um literal da forma $\alpha_i = p_{\alpha_i}(w_1, ..., w_r)$, em que p_{α_i} é um símbolo de predicado e $(w_1, ..., w_r)$ é uma r-tupla instanciada com objetos relacionados por p_{α_i}.

Um problema de planejamento é uma tripla $X = (\Sigma, s_0, g)$, em que Σ é um domínio STRIPS, $s_0 \in S$ é um estado inicial e $g = \{\beta_1, ..., \beta_n\}$ é um conjunto de metas tal que cada meta β_i é um literal instanciado. A solução para um problema de planejamento é um plano sequencial de ações $\psi =< a_1, ..., a_q >$, em que $a_i \in A$, para todo i.

Um axioma $\pi \rightarrow \xi$ é uma regra inerente a Σ, em que π e ξ são literais não-instanciados [3]. No contexto do domínio Σ, um axioma é equivalente à afirmação de que se alguma instância de π é válida em um determinado estado, então uma instância de ξ também deve ser válida nesse estado.

B. Delimitação do Problema

Seja Σ um domínio STRIPS e $O = \{o_1, ..., o_k\}$ o conjunto de operadores STRIPS tal que as ações em Σ são instâncias de operadores em O. Sejam os operadores $o_1, ..., o_k$ tais que as composições de seus respectivos conjuntos de precondições e efeitos não são conhecidos. Assim, para um dado operador o_i e um dado literal não-instanciado $?\alpha$, o problema de identificação de precondições e efeitos de operadores STRIPS consiste em determinar se $?\alpha$ pertence ao conjunto de precondições ou se pertence ao conjunto de efeitos de o_i.

Ressalta-se que, respeitando a correta representação de cada operador [6], um literal pode pertencer a no máximo um desses conjuntos, ou seja, deve valer $pre(o_i) \cap eff(o_i) = \emptyset$, para todo i.

Uma possível abordagem para a solução deste problema é fazer uso de observações sobre a dinâmica das transições em Σ. Seja $C = \{X_1, ..., X_u\}$ um conjunto de problemas de planejamento em Σ cujas respectivas soluções $\psi_1, ..., \psi_u$ são conhecidas e supostamente corretas; assumindo que cada ação afeta apenas os objetos em seu conjunto de argumentos [2], conhecidos o estado inicial e o conjunto de metas de cada problema, é possível fazer suposições sobre a pertinência de um literal a um determinado conjunto em um operador. A seção seguinte fornece detalhes sobre esta abordagem.

III. Identificando Precondições e Efeitos

Esta seção tem por objetivo apresentar estratégias para identificação de precondições e efeitos de operadores STRIPS a partir de um conjunto de problemas previamente solucionados. Sumariamente, a proposta consiste em tomar um literal não instanciado como candidato a integrar um conjunto de precondições ou efeitos de um operador e testá-lo através de um conjunto de restrições; caso o literal candidato não seja adequado às restrições, então deve ser descartado.

Há dois conjuntos de informações fundamentais para a composição das restrições: o caráter dinâmico dos predicados e os axiomas do domínio. O primeiro diz respeito à alterabilidade de literais escritos com cada predicado, enquanto que o segundo trata de implicações entre literais, conforme descrito na seção anterior.

Quanto ao caráter dinâmico, um predicado pode ser classificado como dinâmico ou estático [2]. Se for dinâmico, então qualquer literal que faça referência a tal predicado pode ser removido por alguma ação. Por outro lado, um literal escrito com predicado estático jamais pode ser alterado por qualquer ação. Tomando como exemplo o domínio dos Robôs das Docas (DWR) [1], o predicado $top(?c1?p)$ é classificado como dinâmico, uma vez que um contêiner que esteja no topo de uma pilha pode ser removido desta condição, enquanto que o predicado $belong(?k?l1)$ é classificado como estático, pois descreve a relação de pertinência de um guindaste a uma localidade, a qual não é afetada por qualquer ação nesse domínio, visto que assume-se não ser possível a remoção de guindastes de suas respectivas localidades.

Por sua vez, o conjunto de axiomas do domínio contém as regras que são assumidas como válidas e que relacionam literais entre si. É de particular interesse a classe de axiomas $\pi \rightarrow \xi$ em que ξ é a negação de algum átomo. A partir desse tipo de axiomas, é possível identificar pares de literais mutuamente exclusivos, ou seja, que não podem figurar simultaneamente em um determinado estado. Por exemplo, no domínio do Mundo dos Blocos [7], se em um determinado estado o literal $holding(A)$ estiver presente, então o literal $ontable(A)$ jamais poderá ser verdadeiro nesse mesmo estado, visto que o axioma $holding(?x) \rightarrow \neg(ontable(?x))$ é válido para esse domínio; tal axioma refere-se à regra de que se um

bloco está sendo manipulado pelo guindaste, então esse bloco não está sobre a mesa.

Tanto o caráter dinâmico dos predicados quanto o conjunto de axiomas podem ser ou previamente conhecidos, ou pesquisado nos dados. Caso não sejam conhecidos, alguma técnica de mineração de dados deve ser aplicada de forma a extrair essas informações do conjunto de problemas resolvidos. As técnicas utilizadas neste trabalho são descritas em seção posterior.

A. Precondições

Seja um literal não instanciado $?\alpha = p_\alpha(?w_1, ..., ?w_r)$ em que os tipos dos argumentos formam um conjunto que é subconjunto dos tipos de argumentos de um operador $o \in O$. Determina-se que $?\alpha$ é candidato para compor o conjunto de precondições de o se, para algum problema $X = (\Sigma, s_0, g) \in C$ cuja solução é o plano ψ:

- Alguma instância de α de $?\alpha$ é válida em s_0; e
- Alguma instância a de o é a primeira ação em ψ tal que o conjunto de argumentos de α é subconjunto do conjunto de argumentos de a.

Assim, se α cumpre essas condições, então possivelmente $?\alpha$ seja uma precondição para a aplicação de o. No entanto, há condições adicionais que devem ser verificadas para que $?\alpha$ seja aceito como precondição. Para verificar essas condições é necessária uma pesquisa de outros casos em C em que alguma instância de o foi aplicada e analisar a ocorrência de instâncias de $?\alpha$ similares àquela que deu origem à suposição de que $?\alpha$ é candidato ao conjunto de precondições.

Caso em C haja um problema em que uma instância de o seja aplicada, mas não houver átomo equivalente a $?\alpha$ no estado inicial, caracteriza-se um indício de que $?\alpha$ não seja precondição de o. No entanto, apenas isso não é suficiente para o descarte de $?\alpha$, visto que alguma ação anterior à referida instância de o pode vir a excluir átomo, de modo que ele não esteja presente quando da aplicação da instância de o.

Entretanto, essa possibilidade de exclusão depende do caráter dinâmico de p_α. Se p_α for estático, não é necessário observar se alguma ação anterior à instância de o possa ter excluído ou adicionado a instância de $?\alpha$, pois nenhuma ação excluirá ou adicionará um literal que descreva uma relação estática [2].

Por outro lado, se p_α for dinâmico e alguma ação anterior à instância de o afetar algum objeto que esteja em seu conjunto de argumentos, então nada pode ser afirmarmado a respeito da pertinência de $?\alpha$ ao conjunto de precondições de o. No entanto, se não houver ação anterior à instância de o com essas características, então o valor lógico da instância de $?\alpha$ será o mesmo no estado inicial e no estado imediatamente anterior à aplicação da instância de o.

Assim, sejam as proposições a seguir:

- $pre1$ O átomo equivalente à instância de $?\alpha$ não está presente em s_0.
- $pre2$ O predicado p_α é dinâmico.
- $pre3$ O predicado p_α é estático.
- $pre4$ Nenhuma ação anterior à instância de o afeta algum objeto que esteja em seu conjunto de argumentos.

Então, $?\alpha$ deve ser descartado como precondição de o caso, para algum problema $X = (\Sigma, s_0, g) \in C$, a seguinte fórmula proposicional tenha valor lógico verdadeiro.

$$pre1 \ \wedge \ ((pre2 \ \wedge \ pre4) \ \vee \ pre3) \tag{1}$$

Caso contrário, admite-se que $?\alpha$ seja precondição de o.

B. Efeitos Positivos

Seja um literal não instanciado $?\alpha = p_\alpha(?w_1, ..., ?w_r)$ em que os tipos dos argumentos formam um conjunto que é subconjunto dos tipos de argumentos de um operador $o \in O$. Determina-se que $?\alpha$ é candidato para compor o conjunto de efeitos de o se, para algum problema $X = (\Sigma, s_0, g) \in C$ cuja solução é o plano ψ:

- Alguma instância α de $?\alpha$ é válida em g; e
- Alguma instância a de o é a última ação em ψ tal que o conjunto de argumentos de α é subconjunto do conjunto de argumentos de a.

Assim como ocorre na identificação de precondições, o fato de α atender às condições supracitadas não qualifica $?\alpha$ para o conjunto de efeitos de o. Faz-se necessário verificar condições adicionais em outros casos de C.

Se α for um efeito positivo de a e nenhuma ação após a afetar algum objeto de seus argumentos, então pode-se afirmar que α será um componente do estado resultante da aplicação de ψ, ainda que não pertença a g. Porém, para que o estado resultante seja válido, não deve existir um literal β tal que $\alpha \rightarrow \neg(\beta)$ seja instância de algum axioma do domínio. Dessa forma, se $\beta \in g$ e nenhuma ação após a afeta todos os argumentos de β, então o estado imediatamente após a aplicação de a contém β e, portanto haveria uma contradição se a adicionasse α a esse estado. Se houver alguma ocorrência dessa contradição em qualquer problema de C, então $?\alpha$ não pode ser um efeito positivo de o.

Assim, sejam as proposições a seguir:

- $add1$ Existe um axioma $?\alpha \rightarrow \neg(?\beta)$ tal que para alguma de suas instâncias é verdade que $\beta \in g$.
- $add2$ Nenhuma ação após a instância de o afeta algum objeto de seus argumentos.
- $add3$ Nenhuma ação após a instância de o afeta todos os objetos que são argumentos de β.

Então, $?\alpha$ deve ser descartado como efeito positivo de o caso, para algum problema $X = (\Sigma, s_0, g) \in C$, a seguinte fórmula proposicional tenha valor lógico verdadeiro.

$$add1 \ \wedge \ add2 \ \wedge \ add3 \tag{2}$$

Caso contrário, admite-se que $?\alpha$ seja efeito positivo de o.

C. Efeitos Negativos

Seja um literal não instanciado $?\alpha = p_\alpha(?w_1, ..., ?w_r)$ em que os tipos dos argumentos formam um conjunto que é subconjunto dos tipos de argumentos de um operador $o \in O$. Determina-se que $\neg(?\alpha)$ é candidato para compor o conjunto

de efeitos de o se, para algum problema $X = (\Sigma, s_0, g) \in C$ cuja solução é o plano ψ, $?\alpha$ for admitido como precondição de o.

O fato de $?\alpha$ ser uma precondição para a aplicação de o gera a garantia de que alguma instância α de $?\alpha$ será componente do estado imediatamente anterior à aplicação de uma instância a de o. No entanto, isso não é suficiente para afirmar que a remove α do estado corrente.

Se $\alpha \in g$ e se a for a última ação de ψ a afetar algum objeto que pertença aos argumentos de α, então, se nenhuma ação após a adicionar α ao estado corrente, é certo que a não exclui α. Portanto, se houver algum problema em C em que isso ocorra para algum átomo α e alguma ação a, deve-se desconsiderar $\neg(?\alpha)$ como efeito negativo de o.

Assim, sejam as proposições a seguir:

- $del1$ Alguma instância α de $?\alpha$ pertence a g.
- $del2$ Alguma instância a de o é a última açao de um plano $\psi \in C$ tal que o conjunto de argumentos de α é subconjunto do conjunto de argumentos de a.
- $del3$ Nenhuma ação após a afeta todos os objetos que são argumentos de α.

Então, $?\neg(\alpha)$ deve ser descartado como efeito negativo de o caso, para algum problema $X = (\Sigma, s_0, g) \in C$, a seguinte fórmula proposicional tenha valor lógico verdadeiro.

$$del1 \ \wedge \ del2 \ \wedge \ del3 \tag{3}$$

Caso contrário, admite-se que $\neg(?\alpha)$ seja efeito negativo do operador o.

IV. Experimento

Para verificar a eficácia do método para identificação de precondições e efeitos em modelos de operadores STRIPS, realizou-se o teste com três domínios *benchmark*: Mundo dos Blocos (versão com 4 operadores), Logística e DWR. Para cada domínio, um conjunto de 100 problemas gerados aleatoriamente e corretamente solucionados é usado como banco de casos. Em cada problema, literais candidatos são identificados e testados de acordo com os critérios descritos na seção anterior.

O critério adotado para determinar o caráter dinâmico de um predicado p consiste em verificar se em algum problema do banco de casos há algum literal no conjunto de metas que faz referência a p. Isso porque, para evitar a ocorrência de planos vazios no banco de casos, optou-se pela construção de problemas em que o conjunto de metas é composto por literais que não estão presentes no estado inicial, ou seja, $s_0 \cap g = \emptyset$. Assim, nenhum literal no conjunto de metas poderá ser escrito com predicado estático, visto que, por definição, nenhuma ação poderia adicionar tal literal.

O conjunto de axiomas é obtido através dos próprios dados do banco de casos, antes da aplicação da técnica de identificação de precondições e efeitos. Os axiomas são descobertos por indução [8]. Basicamente, a indução de um axioma consiste na generalização de uma regra a partir de uma observação em particular; uma vez que uma regra é assumida como válida, esta será descartada apenas se outra observação introduzir um conflito tal que a regra se torne insustentável. O conjunto de regras que permanecem válidas após todas as observações compõem o conjunto de axiomas para um determinado domínio.

Tabela I
PRECONDIÇÕES E EFEITOS IDENTIFICADOS PARA OS OPERADORES DO DOMÍNIO MUNDO DOS BLOCOS

Operadores	Precondições	Efeitos	
		Positivos	Negativos
$pickup(?x)$	$clear(?x)$ $ontable(?x)$	$holding(?x)$	$not(clear(?x))$ $not(ontable(?x))$
$putdown(?x)$	$holding(?x)$	$clear(?x)$ $ontable(?x)$	$not(holding(?x))$
$stack(?x?y)$	$clear(?y)$ $holding(?x)$	$clear(?x)$ $on(?x?y)$	$not(clear(?y))$ $not(holding(?x))$
$unstack(?x?y)$	$clear(?x)$ $on(?x?y)$	$clear(?y)$ $holding(?x)$	$not(clear(?x))$ $not(on(?x?y))$

O método foi implementado em linguagem Python, versão 3.5.2. Os resultados são descritos a seguir.

A. Mundo dos Blocos

O Mundo dos Blocos é um domínio proposto na competição de planejamento AIPS 2000 [9]. O ambiente é composto por uma mesa sobre a qual blocos são empilhados. Há um guindaste usado para formar as pilhas de blocos; essas pilhas não tem limite de altura. Ainda, não há limite para o número de blocos e assume-se que a mesa seja grande o suficiente para comportar todos os blocos. Sem embargo, assume-se que o número de blocos seja constante. Todos os blocos são identificados com um rótulo e têm as mesmas dimensões.

O modelo considerado para o Mundo dos Blocos tem quatro operadores. As operações disponíveis consistem em pegar um bloco ($pickup(?x)$), pôr um bloco sobre a mesa ($putdown(?x)$), empilhar um bloco sobre outro ($stack(?x?y)$) e desempilhar um bloco de cima de outro ($unstack(?x?y)$). Os predicados utilizados são: $clear(?x)$ (não há algum bloco sobre o bloco ?x), $ontable(?x)$ (o bloco ?x está sobre a mesa), $on(?x?y)$ (o bloco ?x está sobre o bloco ?y), $holding(?x)$ (o guindaste está segurando o bloco ?x) e $handempty()$ (o guindaste está vazio).

A tabela I apresenta os elementos identificados como precondições e efeitos positivos e negativos para cada operador do Mundo dos Blocos.

A partir desses resultados, verifica-se que todos os elementos identificados como precondições e efeitos para todos os operadores do Mundo dos Blocos estão corretos. No entanto, nenhum literal referente ao predicado $handempty()$ foi encontrado. Isso evidencia a dificuldade de identificação de precondições e efeitos que tenham um conjunto vazio de argumentos; o fato de que ações só afetam objetos que estejam em seu conjunto de argumentos não ajuda na investigação de qual ação

Tabela II
PRECONDIÇÕES E EFEITOS IDENTIFICADOS PARA OS OPERADORES DO DOMÍNIO LOGÍSTICA

Operadores	Precondições	Efeitos	
		Positivos	Negativos
$loadtruck(?p?t?l1)$	$stored(?p?l1)$ $at(?t?l1)$	$in(?p?t)$ $at(?t?l1)$	$not(stored(?p?l1))$ $not(at(?t?l1))$
$unloadtruck(?p?t?l1)$	$in(?p?t)$ $at(?t?l1)$	$stored(?p?l1)$ $at(?t?l1)$	$not(in(?p?t))$
$loadplane(?p?a?l1)$	$airport(?l1)$ $stored(?p?l1)$ $landed(?a?l1)$	$aboard(?p?a)$ $landed(?a?l1)$	$not(stored(?p?l1))$ $not(landed(?a?l1))$
$unloadplane(?x?y)$	$aboard(?p?a)$ $airport(?l1)$ $landed(?a?l1)$	$stored(?p?l1)$ $landed(?a?l1)$	$not(aboard(?p?a))$ $not(landed(?a?l1))$
$drive(?t?l1?l2?c)$	$belong(?l1?c)$ $belong(?l2?c)$ $at(?l1?c)$	$at(?t?l2)$	$not(at(?t?l1))$
$fly(?a?l1?l2)$	$airport(?l1)$ $airport(?l2)$ $landed(?a?l1)$	$landed(?a?l2)$	$not(landed(?a?l1))$

de determinado plano tenha sido aplicada condicionalmente à presença desse literal ou que o tenha adicionado.

Apesar dessa limitação, a técnica proposta pode ser combinada com o método apresentado em [8] para auxiliar na construção do modelo a partir dos dados. O modelo completo pode ser encontrado utilizando-se os axiomas do domínio.

Supondo que os literais identificados sejam corretos, os axiomas nos quais esse literais são antecedentes tornam-se fundamentos para propostas de inserção de outros literais, os quais devem ser testados antes de serem incorporados ao modelo. Por exemplo, o axioma $clear(?x) \rightarrow \neg(handempty())$ pode ser usado para propor que $not(handempty())$ deve integrar o conjunto de efeitos de $putdown(?x)$, visto que $clear(?x)$ é um efeito desse operador. A combinação dessas técnicas será objeto de estudo em trabalhos futuros.

B. Logística

O domínio Logística é um modelo simplificado das operações de uma companhia logística. Assim como o Mundo dos Blocos, é um domínio proposto na competição de planejamento AIPS 2000 [9]. Há um certo número de cidades e um certo número de localidades em cada cidade. Cada localidade contém um armazém no qual a companhia estoca pacotes de encomendas que deverão ser entregues. A empresa tem um caminhão em cada cidade; cada caminhão é usado para transportar encomendas entre localidades dentro dos limites de cada cidade, ou seja, caminhões não fazem entregas entre cidades. Ao menos uma localidade por cidade possui um aeroporto onde os aviões da empresa podem pousar. Os aviões são usados para transportar encomendas entre cidades e não há limites para o tamanho da frota de aviões.

Há seis operadores no modelo, que descrevem as ações de carregar e descarregar um pacote $?p$ em um caminhão $?t$ que esteja em uma localidade $?l1$ ($loadtruck(?p?t?l1)$ e $unloadtruck(?p?t?l1)$), carregar e descarregar um pacote $?p$ em um avião $?a$ em uma localidade $?l1$ ($loadplane(?p?a?l1)$ e $unloadplane(?p?a?l1)$), dirigir um caminhão $?t$ da localidade $?l1$ para a localidade $?l2$ dentro de uma cidade $?c$ ($drive(?t?l1?l2?c)$) e voar com um avião $?a$ de uma localidade $?l1$ para uma localidade $?l2$ ($fly(?a?l1?l2)$). Os predicados são: $stored(?p?l1)$ (o pacote $?p$ está armazenado em $?l1$), $in(?p?t)$ (o pacote $?p$ está no caminhão $?t$), $aboard(?p?a)$ (o pacote $?p$ está abordo do avião a), $at(?t?l1)$ (o caminhão $?t$ está em $?l1$), $landed(?a?l1)$ (o avião $?a$ está em $?l1$), $belong(?l1?c)$ (a localidade $?l1$ se situa na cidade $?c$) e $airport(?l1)$ (há um aeroporto em $?l1$).

A tabela II apresenta os elementos identificados como precondições e efeitos positivos e negativos para cada operador do domínio Logística.

Primeiramente, verifica-se que todos os elementos necessários para compor o modelo foram identificados. Entretanto, há alguns elementos que foram incorretamente identificados como precondições e efeitos dos operadores deste domínio.

Há duas modalidades de erros nos resultados apresentados. O primeiro consiste na determinação de efeitos contraditórios em três operadores; no operador $loadtruck(?p?t?l1)$, tanto $at(?t?l1)$ quanto $not(at(?t?l1))$ são identificados como efeitos, assim como nos operadores $loadplane(?p?a?l1)$ e $unloadplane(?p?a?l1)$, nos quais $landed(?a?l1)$ e $not(landed(?a?l1))$ também foram designados como efeitos. A designação do efeito negativo é feita porque o literal positivo é precondição para o operador e não há um problema no banco de casos em que a presunção de que o operador elimine esse literal seja contrariada. Por outro lado, a designação do efeito positivo deve-se ao fato de que o literal sempre está presente após a aplicação do operador, visto que este não elimina o referido literal. Este tipo de erro pode ser

Tabela III
PRECONDIÇÕES E EFEITOS IDENTIFICADOS PARA OS OPERADORES DO DOMÍNIO DWR

Operadores	Precondições	Efeitos	
		Positivos	Negativos
$move(?r?l1?l2)$	$adjacent(?l1?l2)$ $adjacent(?l2?l1)$ $at(?r?l1)$ $free(?l2)$	$at(?r?l2)$ $free(?l1)$	$not(at(?r?l1))$ $not(free(?l2))$
$load(?k?l1?c1?r)$	$belong(?k?l1)$ $at(?r?l1)$ $unloaded(?r)$ $holding(?k?c1)$	$loaded(?r?c1)$ $empty(?k)$ $at(?r?l1)$	$not(at(?r?l1))$ $not(unloaded(?r))$ $not(holding(?k?c1))$
$unload(?k?l1?c1?r)$	$belong(?k?l1)$ $empty(?k)$ $loaded(?r?c1)$ $at(?r?l1)$	$unloaded(?r)$ $holding(?k?c1)$ $at(?r?l1)$	$not(loaded(?r?c1))$ $not(empty(?k))$ $not(at(?r?l1))$
$take(?k?l1?c1?c2?p)$	$attached(?p?l1)$ $belong(?k?l1)$ $empty(?k)$ $in(?c1?p)$ $top(?c1?p)$ $on(?c1?c2)$	$holding(?k?c1)$ $top(?c2?p)$	$not(empty(?k))$ $not(in(?c1?p))$ $not(top(?c1?p))$ $not(on(?c1?c2))$
$put(?k?l1?c1?c2?p)$	$attached(?p?l1)$ $belong(?k?l1)$ $holding(?k?c1)$ $top(?c2?p)$	$top(?c1?p)$ $empty(?k)$ $on(?c1?c2)$ $in(?c2?p)$ $in(?c1?p)$	$not(holding(?k?c1))$ $not(top(?c2?p)$

facilmente reconhecido e eliminado utilizando-se os filtros prescritos em [5].

A outra modalidade de erro acontece no operador $unloadtruck(?p?t?l1)$, no qual o efeito positivo $at(?t?l1)$ é erroneamente adicionado. Apesar de ser um tipo de erro bastante similar à modalidade citada anteriormente, esta se difere por não acarretar efeitos contraditórios. A identificação deste erro também é relativamente simples, visto que a intersecção entre o conjunto de precondições e o conjunto de efeitos positivos deve resultar num conjunto vazio [6]. Além disso, tanto este tipo de erro quanto o anterior não ocasionam problemas para o emprego do modelo, visto que os efeitos negativos são aplicados antes dos efeitos positivos e que adição de um literal que já esteja presente em um estado é uma operação inerte. Sem embargo, os filtros proposto por [5] também são capazes de remover este tipo de erro.

C. DWR

O domínio do Robô das Docas é porposto por [1], com propósitos didáticos e ilustrativos. Trata-se de uma simplificação das operações em um porto, no qual contêineres são transportados por robôs. Esses robôs se movem entre localidades adjacentes e cada robô tem a capacidade de carregar apenas um contêiner por vez. Uma localidade tem a capacidade de um robô por vez. Cada localidade dispõe de um guindaste que é usado na manupulação de contêineres e de um certo número de *pallets* sobre os quais os contêineres são empilhados. Não há limite de contêineres por pilha e nem de pilhas por localidade.

O modelo do domínio DWR contém cinco operadores, os quais descrevem as ações de mover um robô de uma localidade para outra ($move(?r?l1?l2)$), carregar e descarregar um robô com um contêiner ($load(?k?l1?c1?r)$ e $unload(?k?l1?c1?r)$) e empilhar e desempilhar contêineres ($put(?k?l1?c1?c2?p)$ e $take(?k?l1?c1?c2?p)$). Os predicados usados e suas respectivas interpretações são: $adjacent(?l1?l2)$ (a localidade $?l1$ é adjacente à localidade $?l2$), $attached(?p?l1)$ (a pilha $?p$ está situada à localidade $?l1$), $belong(?k?l1)$ (o guindaste $?k$ pertence à localidade $?l1$), $at(?r?l1)$ (o robô r ocupa a localidade $?l1$), $free(?l1)$ (nenhum robô ocupa a localidade $?l1$), $loaded(?r?c1)$ (o contêiner $?c1$ está carregado no robô $?r$), $unloaded(?r)$ (o robô $?r$ não carrega contêiner algum), $holding(?k?c1)$ (o guindaste $?k$ segura o contêiner $?c1$), $empty(?k)$ (o guindaste $?k$ não segura contêiner algum), $in(?c1?p)$ (o contêiner $?c1$ integra a pilha $?p$), $top(?c1?p)$ (o contêiner $?c1$ está no topo da pilha $?p$) e $on(?c1?c2)$ (o contêiner $?c1$ está empilhado imediatamente sobre o contêiner $?c2$).

A tabela III apresenta os elementos identificados como precondições e efeitos positivos e negativos para cada operador do domínio DWR.

Assim como no caso do domínio Logística, todos os elementos necessários para a composição do modelo foram corretamente identificados. No entanto, há erros de identificação

similares ao caso anterior.

Nos operadores $load(?k?l1?c1?r)$ e $unload(?k?l1?c1?r)$, há efeitos contraditórios, visto que $at(?r?l1)$ e $not(at(?r?l1))$ foram ambos designados como efeitos desses operadores. Essa designação ocorre pelos mesmos motivos que levam ao erro semelhante no domínio Logística e podem ser corrigidos da mesma forma.

Outro erro verificado é a designação de $in(?c2?p)$ como efeito do operador $put(?k?l1?c1?c2?p)$. De fato, sempre que o referido operador for acionado para a retirada de um contêiner $?c1$ de cima de um contêiner $?c2$ será constatado que $?c2$ permanecerá na pilha $?p$; por conta disso, o algoritmo de identificação partiu do pressuposto de que o operador acrescenta esse literal e não encontrou no banco de casos alguma ocorrência que violasse essa suposição. Por um lado, esse erro é difícil de ser identificado e tratado, pois não leva a efeitos contraditórios e nem faz com que a intersecção entre precondições e efeitos positivos deixe de ser vazia. Por outro lado, trata-se de um efeito inerte, pois adiciona um literal que necessariamente está presente no estado imediatamente anterior à aplicação do operador e que, portanto, não acarreta problemas ao emprego do modelo.

V. Considerações Finais

No presente trabalho, um método para identificação de precondições e efeitos de operadores STRIPS é apresentado e testado em três domínios *benchmark*. Os resultados indicam que o método proposto mostrou-se eficaz, apesar da ocorrência de alguns erros que podem ser removidos com a aplicação de um conjunto de técnicas apresentadas em outros trabalhos disponíveis na literatura.

A combinação dessas técnicas pode ser usada para a formulação de uma ferramenta para a modelagem de domínios STRIPS a partir de um banco de casos. Trabalhos futuros serão dedicados ao desenvolvimento e teste de métodos com essa finalidade.

Referências

[1] M. Ghallab, D. Nau and P. Traverso, *Automated Planning:* Theory and Practice, Morgan Kaufmann, 2004.

[2] P. Gregory and S. Cresswell, Domain Model Acquisition in the Presence of Static Relations in the LOP System, *Proceedings of the XXV International Conference on Automated Planning and Scheduling*, p. 97–105, AAAI, 2015.

[3] F. Ivankovic and P. Haslum, Optimal Planning with Axioms, *Proceedings of the XXIV International Joint Conference on Artificial Intelligence*, p. 1580–1586, 2015.

[4] S. Kambhampati, Model-Lite Planning for the Web Age Masses: The Challanges of Planning with Incomplete and Evolving Domain Models, *Proceedings of the XXII AAAI Conference on Artificial Intelligence*, p. 1601–1605, AAAI, 2007.

[5] R. O. Leite and V. E. Wilhelm, An Action Model Learning Method for Planning in Incomplete STRIPS Domains, *Anais do XII Encontro Nacional de Inteligência Artificial e Computacional*, p. 14–19, 2015.

[6] Q. Yang, K. Wu and Y. Jiang, Learning Action Models from Plan Examples Using Weighted MAX-SAT, *Artificial Intelligence Journal*, vol. 171, p. 107–143, AAAI, 2007.

[7] H. H. Zhuo, T. Nguyen and S. Kambhampati, Refining Incomplete Planning Domain Models Through Plan Traces, *Proceedings of the XXIII International Joint Conference on Artificial Intelligence*, p. 2451–2457, 2013.

[8] R. O. Leite and V. E. Wilhelm, Axiom Acquisition for STRIPS Planning Domains, *Anais do XIV Encontro Nacional de Inteligência Artificial e Computacional*, 2017.

[9] F. Bacchus, AIPS'00 Planning Competition: The Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems, *AI Magazine*, vol. 22, n. 3, p. 47–56, 2001.