Mudanças em Problemas de Planejamento sem Solução

Maria Viviane de Menezes

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
DOUTOR EM CIÊNCIAS

Programa: Ciência da Computação

Orientadora: Prof.ª Dr.ª Leliane Nunes de Barros

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da FAPESP

— São Paulo, agosto de 2014 —

À minha mãe, Ana Cleydes;

ao meu noivo, Paulo de Tarso;

aos meus sogros, Tânia e José Wilson.

Resumo

Planejar é a capacidade que um agente inteligente possui de escolher que ações levam ao alcance de uma meta. Um problema de planejamento é descrito em termos das ações que o agente pode executar, um estado inicial e a meta a ser alcançada. A solução para um problema de planejamento é um plano, i.e., uma sequência de ações que leve o agente do estado inicial a um estado satisfazendo a meta. Em alguns casos, não existe um plano possível para o problema: o problema não possui solução. As possíveis causas para que um problema de planejamento não possua solução são: (i) erros na definição do estado inicial tal que é impossível a partir dele alcançar a meta; (ii) super especificação da meta de planejamento, i.e., imposição de muitos objetivos ao agente que torna impossível o alcance de todos eles ou; (iii) erros na especificação das ações. A análise de problemas não solucionáveis é utilizada por projetistas de domínios de planejamento no processo de definição de um novo domínio. No entanto, quando um problema não possui solução muito mais útil do que apenas informar que não foi possível obter um plano seria mostrar ao projetista as possíveis modificações que deveriam ser feitas para que o problema torne-se solucionável. Nesta tese, formalizamos a tarefa de modificar um problema de planejamento sem solução bem como propomos métodos formais que sejam capazes de sugerir os três tipos possíveis de modificações: modificação do estado inicial, modificação da meta de planejamento e modificação das ações. Propomos e implementamos um método baseado em verificação de modelos capaz de sugerir modificações para o estado inicial e meta de planejamento e ações para problemas sem solução determinísticos e, também, não determinísticos. Além disso, propomos e implementamos um método baseado em lógica booleana quantificada que analisa fórmulas representado as ações do domínio de planejamento e sugere modificações do estado inicial para um problema sem solução determinístico. Os métodos propostos foram implementados de forma eficiente utilizando Diagramas de Decisão Binária.

Abstract

Planning is the capability of an agent choosing its action in order to reach a given goal. A classical planning problem is described terms of a set of actions, an initial state and a goal that have to be reached. A solution to a classical planning problem is a sequence of actions, i.e., a plan, that leads the agent from the initial state to a state satisfying the goal. Sometimes there is no such plan: the problem is *unsolvable*. Possible explanations are: (i) the initial state is wrongly specified; (ii) the goal is over-constrained ("over-subscribed"); (iii) the action specifications are not sound. In this case, a small change in the problem specification would make it solvable. The analysis of unsolvable tasks can help a knowledge engineer when modeling a new planning application. In this thesis, we propose the use of formal methods to find a minimal modifications in an unsolvable planning problem that become it solvable. We propose an approach based on model checking that is able to suggest modifications in the: initial state, goal specification and actions. This approach can be used for deterministic and non-deterministic planning problemas. We also propose another method based on quantified boolean formulas that analysis the actions of a deterministic unsovable problem and suggest modifications for the initial state definition to become the problem solvable. The proposed methods were efficiently implemented using Binary Decision Diagrams.

Sumário

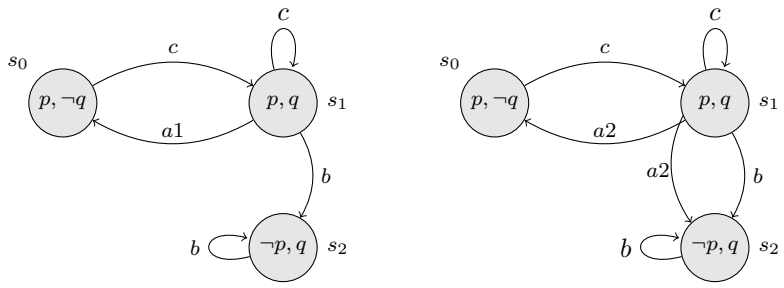
Lista de Figuras

Capítulo 1

Introdução

A habilidade de planejar tarefas é um aspecto fundamental do comportamento inteligente e sua automatização tem sido um dos principais objetivos da pesquisa realizada em Inteligência Artificial [Russell et al., 1995]. Neste contexto, *planejamento automatizado* [Ghallab et al., 2004] preocupa-se com a automatização do processo de escolha de ações de um agente para atingir uma dada meta. Um *problema de planejamento* é dado em termos: (i) da representação do ambiente, o qual é chamado *domínio de planejamento*; (ii) da situação inicial em que o agente se encontra, denominado *estado inicial* e; (iii) de uma propriedade que deve ser satisfeita, denominada *meta de planejamento*.

Visando simplificar a tarefa de encontrar soluções para problemas de planejamento, a abordagem clássica supõe que o ambiente evolui de forma *determinística*, isto é, que não há incerteza sobre os efeitos das ações do agente. A Figura 1.1(a) mostra um exemplo abstrato de domínio de planejamento em termos de um sistema de transição de estados em que os nós representam estados (definidos pelos possíveis valores das proposições p e q) e os arcos representam ações determinísticas. Uma ação determinística quando aplicada a um estado específico leva um agente a um único estado sucessor, por exemplo: a execução da ação a_1 no estado s_1 leva ao estado sucessor s_0; a execução da ação b no estado s_1 (s_2) leva ao estado s_2 (s_2); e a execução da ação c no estado s_0 (s_1) leva o agente ao estado s_1 (s_1). Porém, no mundo real há diversas situações em que as ações possuem efeitos incertos, i.e., *não determinísticos*. Por exemplo, na Figura 1.1(b), a ação a_2 é não determinística, pois quando executada em s_1 pode levar um agente ao estado s_0 ou ao estado s_2. Um domínio de planejamento com pelo menos uma ação com efeito incerto é chamado de *domínio de planejamento não determinístico*.



(a) Espaço de transição de estados de um domínio determinístico.

(b) Espaço de transição de estados de um domínio não determinístico.

Figura 1.1: Representação (explícita) do espaço de transição de estados de dois domínios de planejamento.

A solução para um problema de planejamento determinístico é um *plano*, i.e., uma sequência de ações que leva o agente do estado inicial para um estado que satisfaz a meta. Devido à incerteza no resultado da execução das ações, a solução para um problema de planejamento não determinístico é uma *política*, i.e., uma função que informa para cada estado visitado a ação que deve ser executada para que a meta seja alcançada. Em alguns casos, um *problema de planejamento pode não ter uma solução*, i.e., não é possível obter um plano (ou política) que leve o agente do estado inicial para um estado satisfazendo a meta. Um *problema de planejamento* também pode ser representado implicitamente por um conjunto de ações, um estado inicial e uma meta, como mostrado na Definição 1.1.

Definição 1.1. *(Problema de Planejamento) Um problema de planejamento* Π *sobre um conjunto de proposições* $\mathbb{P}$ *é definido por uma tupla* $\langle \mathbb{A}, s_0, \varphi \rangle$ *em que:*

- $\mathbb{A}$ *é o conjunto de ações;*
- s_0 *é a definição do estado inicial do problema, i.e., uma valoração completa para os elementos de* $\mathbb{P}$*;*
- φ *é a meta de planejamento, ou seja, uma fórmula proposicional sobre* $\mathbb{P}$*.*

Como exemplo, podemos definir implicitamente um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ para o domínio da Figura 1.1(a) em que $\mathbb{A} = \{a_1, b, c\}$, o estado inicial é o estado s_0 e a meta φ é dada pela fórmula proposicional $\neg p \wedge q$. Um possível plano solução para Π é a sequência de ações: *c*, *b*.

Definição 1.2. *(Verificação da existência do plano) Dado um problema de planejamento* $\Pi =$

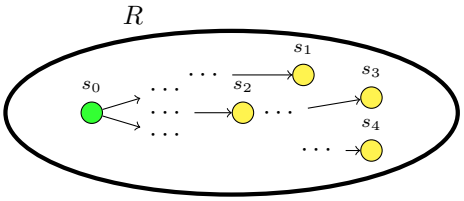


Figura 1.2: Conjunto R dos estados alcançáveis a partir do estado inicial s_0 para um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$.

$\langle \mathbb{A}, s_0, \varphi \rangle$ decidir se existe ou não um plano para Π é um problema PSPACE *[Ghallab et al., 2004].*

O problema de verificar a existência de um plano é denominado PLANEX (do inglês, "*Plan Existence*"). Esta verificação, no entanto, pode ser feita analisando-se o conjunto de estados alcançáveis a partir do estado inicial (Definição 1.3) e o conjunto de estados que alcançam os estados satisfazendo a meta (Definição 1.4). A Figura 1.2 ilustra o conjunto R dos estados alcançáveis a partir do estado inicial s_0 para um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$. Já a Figura 1.3 ilustra o conjunto dos estados U que finalmente alcançam um estado meta φ.

Definição 1.3. *(Estados alcançáveis a partir do estado inicial) Dado um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, dizemos que R é o conjunto de todos os estados sucessores obtidos a partir do estado inicial, i.e., os estados alcançáveis a partir de s_0.*

Definição 1.4. *(Estados que alcançam um estado meta) Dado um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, dizemos que U é o conjunto de todos os estados que alcançam um estado $s_g \models \varphi$.*

Definição 1.5. *(Problemas de Planejamento Solucionáveis) Seja $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ um problema de planejamento em que $\mathbb{A}$ é o conjunto de ações, s_0 é o estado inicial e φ é a meta. Sejam G o conjunto de estados que satisfazem a meta φ, R o conjunto dos estados alcançáveis a partir de s_0 e U o conjunto dos estados que alcançam um estado meta. Π é um problema solucionável se e somente se $G \cap R \neq \emptyset$ e $s_0 \in U$.*

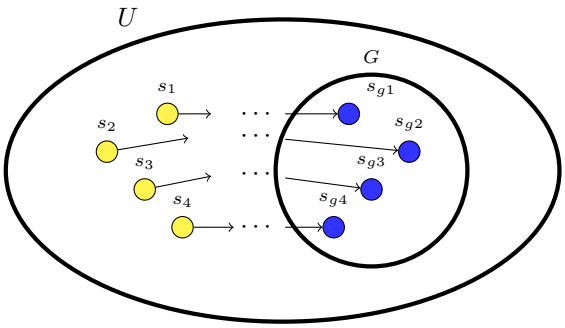


Figura 1.3: Conjunto U dos estados que alcançam algum estado meta para um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, sendo G o conjunto dos estados que satisfazem a fórmula φ.

	Grafo de Transição de Estados	*Linguagem de Ações*
Estados Enumerados	Representação Explícita e Enumerativa Ex: verificação de modelos	Representação Implícita e Enumerativa Ex: FF, FD, LAO*
Formulas Lógicas	Representação Explícita e Símbolica Ex: verificação simbólica de modelos	Representação Implícita e Símbolica Ex: Propplan, EpcPlanner

Figura 1.4: Quatro categorias de representação de estados e domínios de planejamento definidos em termos de duas dimensões: explicíta versus ímplicita e enumerativa versus símbolica.

1.1 Representação de Domínios de Planejamento

A Figura 1.4 mostra quatro diferentes formas de se representar estados e domínios de planejamento organizadas na forma de uma tabela com 2 linhas e 2 colunas. As representações da coluna esquerda da tabela referem-se à **representação explícita** do domínio de planejamento (dada por um grafo de transição de estados), enquanto que as representações da coluna direita referem-se à **representação implícita**, dada por uma linguagem de ações. As representações da linha superior referem-se à **representação enumerativa** dos estados, enquanto que aquelas da linha inferior referem-se à **representação simbólica** dos estados, dada por fórmulas lógicas (lógica proposicional ou de primeira ordem). Cada região da Figura 1.4 corresponde a uma forma conhecida e estudada de representação de domínios de planejamento e são descritas em detalhes a seguir.

1.1.1 Representação Explícita e Enumerativa

Dados um conjunto de estados do mundo e um conjunto de ações $\mathbb{A}$, representando as habilidades do agente, um domínio de planejamento pode ser representado explicitamente por meio de um sistema de transição de estados como os da Figura 1.1. Nessa representação, os estados são rotulados por um conjunto de proposições $\mathbb{P}$ (que representam as propriedades do mundo) e as transições são rotuladas por elementos de $\mathbb{A}$. Tal sistema é denominado *sistema de transição de estados rotulado por ações*, sendo o par $(\mathbb{P}, \mathbb{A})$ sua *assinatura* [1]. Adota-se a *suposição do mundo fechado* na descrição dos estados, i.e., descreve-se apenas as proposições que são verdadeiras no estado, sendo que as proposições consideradas falsas não são descritas.

Definição 1.6. *(Representaçao Explícita para Domínios de Planejamento com Estados Enumerados) Dado um conjunto de proposições $\mathbb{P} = \{p_1, \cdots, p_n\}$ e um conjunto de ações $\mathbb{A}$, um domínio de planejamento pode ser representado explicitamente como um sistema de transição de estados $M = \langle S, L, T \rangle$ em que:*

- *$S = \{s_1, \cdots, s_n\}$ é um conjunto finito e enumerado de estados;*
- *$L : S \mapsto 2^{\mathbb{P}}$ é a função de rotulação de estados, a qual descreve que proposições são verdadeiras em um estado;*
- *$T : S \times \mathbb{A} \mapsto S$ é uma função de transição de estados em que dado um estado $s \in S$ e uma ação $a \in \mathbb{A}$, devolve o estado sucessor de s por meio da execução da ação a.*

Exemplo 1.1. *(Representaçao Explícita de um Domínio de Planejamento Proposicional) Suponha que queremos representar o domínio das chaves [Göbelbecker et al., 2010], um domínio de planejamento determinístico em que o robô deve navegar entre as salas do ambiente, e se necessário, deve abrir portas com chave específicas. Inicialmente são definidos o conjunto de objetos deste domínio que são: duas salas (sala 0 e sala 1); uma chave e; uma porta. Depois, definimos um conjunto $\mathbb{P}$ de proposições, sobre o conjunto de objetos, que descrevem a localização do robô, a localização da chave, o estado da porta (aberta ou fechada), qual chave abre uma determinada porta e as conexões entre as salas, sendo $\mathbb{P} = \{$loc-robô-sala0, loc-robô-sala1, loc-chave-sala0, loc-chave-sala1, aberta-porta0, abre-chave0-porta0, conecta-sala0-sala1- porta0, conecta-sala1 -sala0-porta0$\}$. O conjunto de ações $\mathbb{A}$ representam as habilidades do robô no ambiente, sendo $\mathbb{A} = \{$mover-0-1, mover-1-0, moverchave-0-1, moverchave*

[1] No Capítulo 5, veremos que estados também podem ser representados por variáveis multivaloradas, ao invés de proposicionais. Porém, essa representação pode ser facilmente traduzida para uma representação proposicional.

-1-0, abrir0, abrir1, fechar0, fechar1}. A ação mover-0-1 (mover-1-0) move o robô da sala 0 para a sala 1 (da sala 1 para a sala 0). As ações moverchave-0-1 e moverchave-1-0 movimentam o robô juntamente com a chave. As ações abrir0 (abrir1) e fechar0 (fechar1) fazem o robô, respectivamente, abrir e fechar a porta a partir da sala 0 (sala 1). Cada um dos estados ($s_0, s_1, s_2, s_3, s_4, s_5, s_6$ e s_7) é representado pelo conjunto de proposições que são verdadeiras no estado. No estado s_0, por exemplo, o robô está na sala 0, a chave está na sala 0 e a porta está fechada. Assim, $L(s_0) = \{$loc-robô-sala0, loc-chave-sala0, conecta-sala0-sala1-porta0, conecta-sala1-sala0- porta0$\}$. As transições são rotuladas pelas ações que causam a mudança de um estado para o outro. Por exemplo, a transição entre o estado s_0 e o estado s_1 é rotulada pela ação abrir0.

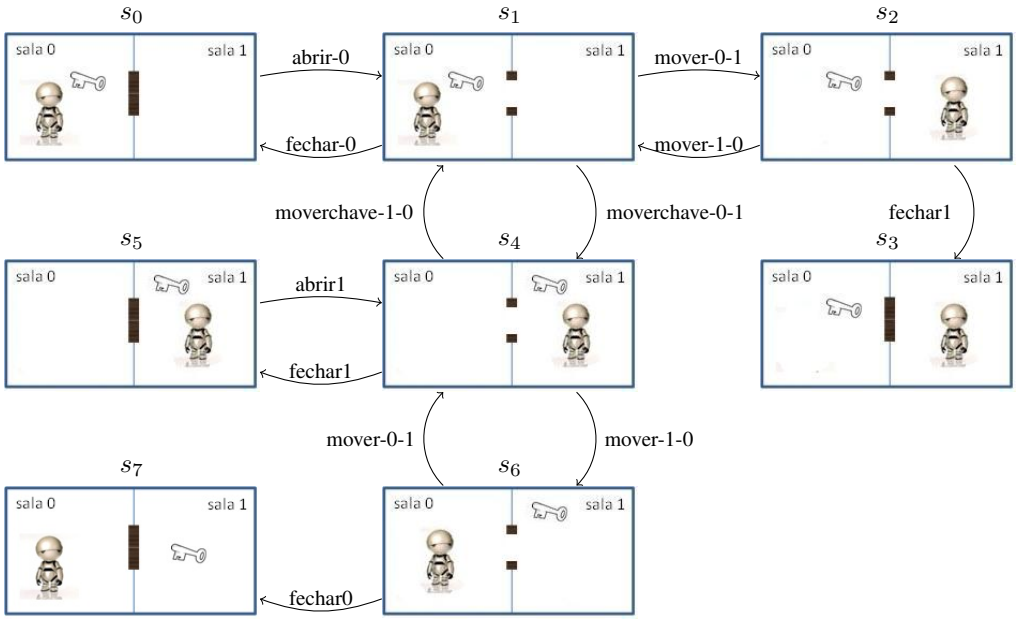


Figura 1.5: Representação explícita para o domínio das chaves com 8 estados e 8 ações.

Proposições cujos valores não são alterados pelas ações são denominadas *invariantes*. Note que dado o número de proposições do domínio da chaves, o número de estados possíveis seria $2^8 = 256$. No entanto, devido às invariantes (*abre-chave0-porta0*, *conecta-sala0-sala1-porta0* e *conecta-sala1-sala0-porta0*) e às restrições do domínio (por exemplo, o robô e a chave devem estar em pelo menos uma, e em apenas uma, sala) muitos estados são considerados inválidos. De fato, o domínio das chaves poderia ser descrito apenas pelo conjunto de proposições $\mathbb{P}' = \{$*loc-robô-sala0*, *loc-chave-sala0*, *aberta-porta0*$\}$, considerando as invariantes e assumindo que

se a proposição *loc-robô-sala0* (*loc-chave-sala0*) é verdadeira o robô (a chave) está na sala 0 e se é falsa então o robô (a chave) está na sala 1. O fato é que a representação explícita de um domínio de planejamento é exponencial no número de proposições, o que torna impraticável a representação de domínios com, por exemplo, 30 variáveis proposicionais ($2^{30} = 1073741824 \approx 10^9$ estados) [Ritanen, 2006]. Este problema é denominado problema da *explosão do espaço de estados*.

1.1.2 Representação Explícita e Simbólica

Dentre as abordagens propostas para contornar o problema da explosão do espaço de estados, a abordagem de *representação simbólica do espaço de estados* é uma que tem sido amplamente utilizada. Nesta abordagem, conjunto de estados (ao invés de estados individuais enumerados) são representados por meio de fórmulas da lógica proposicional que são codificadas em estruturas de dados eficientes denominadas *Diagramas de Decisão Binária* (BDDs - *Binary Decision Diagrams*)[2] [McMillan, 1992]. A representação explícita e simbólica é amplamente utilizada para representar domínios de planejamento pela abordagem de planejamento como verificação simbólica de modelos [Cimatti et al., 2003, Giunchiglia and Traverso, 2000, Edelkamp and Helmert, 2001, Pereira and Barros, 2008]. Essa abordagem será discutida em detalhes no Capítulo 4.

1.1.3 Representação Implícita e Enumerável

No lugar de representar explicitamente os estados do domínio de planejamento, é possível descrever apenas as mudanças que as ações causam no mundo. Esta é uma forma bastante compacta de representação e é conhecida como representação implícita de domínios de planejamento. Neste tipo de representação adota-se uma linguagem para descrever as ações em termos de suas pré-condições e efeitos, como por exemplo a linguagem STRIPS [Fikes and Nilsson, 1972]. Os problemas de planejamento usados na análise deste trabalho são representados dessa forma.

Definição 1.7. *(Representaçao Implícita para Domínios de Planejamento) Dado um conjunto de proposições $\mathbb{P}$, um domínio de planejamento pode ser representado implicitamente por um*

[2]Outras estruturas para representação de fórmulas booleanas tais como tabelas verdades e formulas normais conjuntivas são exponenciais

conjunto de ações $\mathbb{A}$ onde cada $a \in \mathbb{A}$ é especificada por:

$$\alpha = \langle \mathit{precond}(\alpha), \mathit{efeitos}(\alpha) \rangle,$$

em que precond(α) é um conjunto de precondições, representando as proposições que devem ser verdadeiras no estado corrente para que a ação α seja executada, e efeitos(α) representam como o estado é modificado após a execução da ação. Os efeitos são dados pelo par efeitos$(\alpha) = \langle add(\alpha), del(\alpha) \rangle$ em que: add(α) é o conjunto de proposições que se tornam verdadeiras após a execução da ação e del(α) é o conjunto de proposições que se tornam falsa após α ser executada.

Exemplo 1.2. *(Representaçao Implícita de um Domínio de Planejamento) A Figura 1.6 mostra a representação implícita do domínio das chaves proposicional por meio de ações* STRIPS. *Cada ação $a \in \mathbb{A}$ é dada por 3 conjuntos de proposições representando as pré-condições, efeitos positivos e efeitos negativos da ação.*

mover-0-1 :
$\langle$*precond* $= \{$*loc-robô-sala0*, *aberta-porta0*, *conecta-sala0-sala1-porta0*$\}$;
efeitos $= \langle \{$*loc-robô-sala1*$\}; \{$*loc-robô-sala0*$\} \rangle$
mover-1-0 :
$\langle$*precond* $= \{$*loc-robô-sala1*, *aberta-porta0*, *conecta-sala1-sala0-porta0*$\}$;
efeitos $= \langle \{$*loc-robô-sala0*$\}; \{$*loc-robô-sala1*$\} \rangle$
moverchave-0-1 :
$\langle$*precond* $= \langle \{$*loc-robô-sala0*, *loc-chave-sala0*, *aberta-porta0*, *conecta-sala0-sala1-porta0*$\}$;
efeitos $= \langle \{$*loc-robô-sala1*, *loc-chave-sala1*$\}; \{$*loc-robô-sala0*, *loc-chave-sala0*$\} \rangle$
moverchave-1-0 :
$\langle$*precond* $=$ *conecta-sala1-sala0-porta0*$\}$;
efeitos $= \langle \{$*loc-robô-sala0*, *loc-chave-sala0*$\}; \{$*loc-robô-sala1*, *loc-chave-sala1*$\} \rangle$
abrir0 :
$\langle$*precond* $= \langle \{$*loc-robô-sala0*, *loc-chave-sala0*, *conecta-sala0-sala1-porta0*$\}$;
efeitos $= \langle \{$*aberta-porta0*$\}; \{\} \rangle$
abrir1 :
$\langle$*precond* $= \langle \{$*loc-robô-sala1*, *loc-chave-sala1*, *conecta-sala1-sala0-porta0*$\}$;
efeitos $= \langle \{$*aberta-porta0*$\}; \{\} \rangle$
fechar0 :
$\langle$*precond* $= \langle \{$*loc-robô-sala0*, *conecta-sala0-sala1-porta0*, *aberta-porta0*$\}$;
efeitos $= \langle \{\}; \{$*aberta-porta0*$\} \rangle$
fechar1 :
$\langle$*precond* $= \langle \{$*loc-robô-sala1*, *conecta-sala0-sala1-porta0*$\}$;
efeitos $= \langle \{$*aberta-porta0*$\}; \{\} \rangle$

Figura 1.6: Exempplo de Representação implícita para o domínio das chaves com ações STRIPS.

É possível também generalizar a representação proposicional das ações utilizando uma notação derivada da lógica de primeira ordem com predicados, variáveis e símbolos constantes. No

domínio das chaves, podemos ter o seguinte conjunto de predicados $\mathcal{P} = \{$*robô-pos(sala x)*, *chave-pos(sala x)*, *aberta(porta d)*, *abre(chave k, porta d)*, *conecta(sala x, sala y, porta d)*$\}$. Utilizando os predicados para definir as ações, obtém-se *operadores de planejamento*.

Definição 1.8. *(Operadores de Planejamento) Dada uma linguagem de primeira ordem, um domínio de planejamento pode ser representado implicitamente por um conjunto de operadores $\mathcal{O}$ onde cada $o \in \mathcal{O}$ é especificado por:*

$$o = \langle \mathit{precond}(o), \mathit{efeitos}(o) \rangle,$$

em que precond(o) e efeitos(o) são, respectivamente, generalizações das pré-condições e efeitos das ações proposicionais. Desta forma, no lugar de conjunto de proposições usa-se conjuntos de literais (átomos e negações de átomos) para definir as pré-condições e efeitos dos operadores.

Exemplo 1.3. *(Representaçao Implícita de um Domínio de Planejamento utilizando Operadores) A Figura 1.7 mostra a representação implícita do domínio das chaves relacional. Cada operador $o \in \mathcal{O}$ é dado por dois conjuntos de literais representando suas pré-condições e efeitos.*

mover(sala x, sala y, porta z) :
  $\langle$*precond* $= \{$*loc-robô(x)*, *aberta(z)*, *conecta (x, y, z)*$\}$;
  efeitos $= \{$*robô-pos(y)*, $\neg$*robô-pos(x)*$\}\rangle$
mover-com-chave(sala x, sala y, porta z, chave k) :
  $\langle$*precond* $= \{$*robô-pos(x)*, *chave-pos(x)*, *aberta(z)*, *conecta(x,y,z)*$\}$;
  efeitos $= \{$*robô-pos(y)*, *chave-pos(y)*, $\neg$*robô-pos(x)*, $\neg$*chave-pos(x)*$\}\rangle$
abrir(porta z, chave k, sala x, sala y) :
  $\langle$*precond* $= \{$*robô-pos(x)*, *chave-pos(x)*, *conecta(x,y, z)*$\}$;
  efeitos $= \{$*aberta(z)*$\}\rangle$
fechar-porta0-sala0-sala1 :
  $\langle$*precond* $= \{$*robô-pos(x)*, *conecta(x, y,z)*, *aberta(z)*$\}$;
  efeitos $= \{\neg$*aberta(z)*$\}\rangle$

Figura 1.7: Exemplo de representação implícita para o domínio das chaves com operadores.

1.1.4 Representação Implícita e Simbólica

[Fourman, 2000, Rintanen, 2008] utilizam a representação implícita e simbólica do domínio de planejamento. Neste tipo de representação, as ações (dadas em termos de pré-condições e efeitos) são transformadas em fórmulas lógicas de forma que é possível raciocinar a partir da representação simbólica de um conjunto de estados e das ações por meio de manipulação de fórmulas.

1.2 Problemas de Planejamento sem Solução

Considere o problema no domínio das chaves em que no estado inicial o robô está na sala 0, a porta está fechada e a chave está na sala 1 (estado s_7 da Figura 1.5). Esse problema não tem solução é um problema sem solução, uma vez que não é possível que o robô abra a porta e alcance a sala 1. Nesse caso, informar os motivos pelo qual não foi possível solucionar o problema é mais importante do que simplesmente dizer que não foi possível obter um plano (ou política) é . Por exemplo, se mudassemos a localização da chave, ou ainda, se abrissemos a porta tornaria o problema solucionável.

Definição 1.9. *(Problemas de Planejamento sem Solução) Seja* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *um problema de planejamento em que* $\mathbb{A}$ *é o conjunto de ações,* s_0 *é o estado inicial e* φ *é a meta. Sejam* G *o conjunto de estados que satisfazem a meta* φ*,* R *o conjunto dos estados alcançáveis a partir de* s_0 *e* U *o conjunto dos estados que alcançam um estado meta.* Π *é um problema de planejamento sem solução se e somente se* $G \cap R = \emptyset$ *e* $s_0 \notin U$*.*

A análise de problemas de planejamento sem solução é uma abordagem importante na área de engenharia de conhecimento para planejamento [McCluskey, 2000], i.e., problemas sem solução podem ser usados por um projetista durante o processo de desenvolvimento de um novo domínio. Problemas sem solução podem ter três possíveis causas: (i) o estado inicial foi especificado incorretamente e, assim, não é possível a partir dele alcançar um estado meta; (ii) a meta de planejamento não pode ser sartisfeita pelos estados alcançáveis a partir do estado inicial ("*over-subscribed goals*" [Benton et al., 2007, Rabideau et al., 2008]) e; (iii) as ações não foram especificadas corretamente, i.e., pode existir erros na especificação das pré-condições ou efeitos das ações.

Dado um problema de planejamento sem solução sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ (Definição 1.9) é possível modificá-lo para torná-lo um problema com solução, fazendo-se uma das seguintes modificações (atualizações ou revisões):

1 *Modificação do estado inicial* s_0 [Göbelbecker et al., 2010, Menezes et al., 2012, Menezes and Barros, 2013, Herzig et al., 2014], nesse caso é preciso selecionar o melhor estado $s \in U$ (estados que alcançam estados que satisfazem φ) para ser o novo estado inicial do problema;

2 *Modificação da meta de planejamaneto* φ [Edelkamp and Kissmann, 2009, Benton et al.,

2007, Rabideau et al., 2008, Herzig et al., 2014], nesse caso é preciso definir uma nova meta φ' tal que $\exists s \in R$ e $s \models \varphi'$;

3 *Modificação do conjunto de ações* $\mathbb{A}$ [Menezes et al., 2010, Menezes and Barros, 2011b, Menezes et al., 2012, Herzig et al., 2014], nesse caso é preciso modificar uma ação $a \in \mathbb{A}$ obtendo um novo conjunto de ações $\mathbb{A}'$ que induz um conjunto R' de estados alcançáveis a partir de s_0 e um novo conjunto U' de estados que alcançam o conjunto G (i.e., o conjunto dos estados meta). Os novos conjuntos R' e U' são tais que $s_0 \in U'$ e $R \cap G' \neq \emptyset$.

1.3 Motivação

Engenharia de Conhecimento para Planejamento é o processo que trata da aquisição, validação e manutenção de modelos de domínios de planejamento. Aquisição de um modelo de domínio envolve uma análise prolongada do domínio da aplicação por engenheiros do conhecimento, por meio de entrevistas com as partes interessadas, manuais do sistema, modelos existentes, documentação de software, etc. Já a validação de um modelo é o *modelo do domínio de planejamento como dos problemas para esse domínio* [McCluskey, 2000].

Apesar dos esforços empregados na Competição Internacional de Engenharia de Conhecimento para Planejamento e Escalonamento (ICKEPS - *International Competition on Knowledge Engineering for Planning and Scheduling*) [Barták and McCluskey, 2007], ainda há poucas ferramentas que forneçam suporte a um projetista de novos domínios de planejamento, tanto durante o processo de aquisição do conhecimento como na validação de modelos de domínios. O resultado disto é que a maioria dos domínios amplamente utilizados por pesquisadores da área para teste e avaliação de planejadores, os chamados domínios *benchmarks*, foram inspecionados apenas por seus projetistas, muitas vezes sem uma ferramenta adequada que os auxiliasse neste processo, o que resultou em erros de modelagem [Long and Fox, 2003].

Um exemplo de erro na modelagem ocorreu no Domínio dos Colonos [Long and Fox, 2003] (*Settlers*, proposto para a terceira IPC). Este domínio modela tarefas de construção de sistemas de transporte (carros, trens, estradas e barcos). As ações permitem coletar matéria-prima, tais como madeira e pedra, e usá-las para desenvolver construções mais complexas, o que então permite acesso a materiais mais sofisticados, tais como ferro e carvão. Uma análise mais cautelosa neste domínio revelou que a ação responsável por construir um barco foi mal especificada: sua única

precondição é a existência de ferro em quantidade suficiente para a construção de barcos e o seu efeito é a criação de um barco em uma localização que pode não ser em um cais de porto, o que impede a navegação desses barcos a partir dessa localização. Este erro de modelagem só foi percebido após seis anos da criação desse domínio [Long et al., 2009].

Um outro exemplo de erro de modelagem ocorreu no domínio MPrime [Bacchus, 2001], proposto para a primeira IPC. Este domínio modela um problema de transporte que possui tanques de combustível em diferentes localizações e usa veículos capazes de transportar esse combustível de um tanque para outro. O domínio inclui uma ação que permite a transferência de combustível por um veículo que consome uma unidade de combustível para se locomover e gera uma unidade de combustível no tanque de destino. Porém, esta ação podia fazer a transferência de combustível pode ser instanciada para permitir a transferência de combustível de um tanque para ele próprio. Devido a este erro de modelagem era possível que um veículo consumisse todo o combustível de um tanque sem sair do lugar. Este é um exemplo de erro cujo modelo do domínio está sintaticamente correto, mas que não está de acordo com o desejado. Além disso, os veículos também consomem parte desse combustível para se locomoverem.

1.4 Objetivos

O objetivo dessa tese é formalizar a tarefa de modificação de problemas de planejamento sem solução, propondo um conjunto de métricas para a escolha da melhor modificação e métodos eficientes que sejam capazes de sugerir os três tipos possíveis de modificações para tornar um problema sem solução solucionável em termos de: modificação do estado inicial, modificação da meta de planejamento e modificação das ações. Utilizamos a representação implícita e simbólica dos domínios de planejamento nos métodos propostos.

Propomos um método baseado em verificação de modelos capaz de sugerir modificações para o estado inicial e meta de planejamento para problemas sem solução determinísticos e, também, não determinísticos. Além disso, propomos um método que analisa fórmulas representado as ações do domínio de planejamento e é capaz de sugerir modificações do estado inicial para um problema sem solução determinístico. Este método é inspirado em uma abordagem existente proposta por [Göbelbecker et al., 2010]. Finalmente, propomos um método para modificar as pré-condições ou efeitos das ações para que um problema sem solução torne-se solucionável. No entanto, a modificação de uma ação pode interferir na solubilidade de outros problemas que são

definidos sobre tal ação. Assim, um conjunto de problemas é utilizado como casos de teste para a modificação das ações e a métrica para escolha da melhor modificação deve maximar o número de problemas do conjunto que são solucionáveis.

1.5 Contribuições

As principais contribuições dessa tese são as seguintes:

- formalização da tarefa de encontrar modificações do estado inicial e meta de planejamento como um problema, respectivamente, de atualização e revisão de modelos [Herzig et al., 2014];
- formalização da tarefa de encontrar modificações para o conjunto de ações [Menezes and Barros, 2011a, Menezes et al., 2010, Menezes and Barros, 2011b, Menezes et al., 2011];
- elaboração de métricas para escolha da melhor modificação do estado inicial, meta e ações;
- formulação original da progressão e regressão simbólica de ações não determinísticas com o raciocínio sobre a representação implícita e simbólica do domínio de planejamento [Menezes et al., 2013, Menezes et al., 2014];
- implementação da versão simbólica do verificador de modelos α-CTL que pode receber como entrada tanto a representação explícita como a representação implícita do domínio de planejamento, estendendo assim a versão explícita e enumerativa proposta por [Pereira and Barros, 2008];
- comparativo das abordagens simbólicas explícitas e implícitas na fase de verificação da existência de um plano;
- elaboração e implementação do método de sugerir um novo estado inicial [Menezes and Barros, 2013], meta e conjunto de ações utilizando a verificação simbólica de modelos com domínios representados de forma implícita e simbólica;
- definição e implementação do método de sugerir modificações para o estado inicial de problemas de planejamento determinístico [Menezes et al., 2012], inspirado na abordagem proposta por [Göbelbecker et al., 2010];
- definição e implementação do método para modificação das ações baseado em um conjunto de problemas teste [Menezes and Barros, 2011b].

1.6 Organização

O restante dessa tese está organizado conforme segue. No Capítulo 2, mostramos as diferentes formas de raciocinar sobre ações determinísticas, dependendo dos diferentes tipos de representação do domínio de planejamento. No Capítulo 3, mostramos as diferentes formas de raciocinar sobre ações não determinísticas, incluindo uma das contribuições desta tese que é a progressão e regressão simbólica de ações não determísticas. No Capítulo 4, formalizamos a tarefa de encontrar modificações para problemas de planejamento sem solução com a lógica DL-PA [Herzig et al., 2014] e com verificação de modelos [Menezes and Barros, 2013]. No Capítulo 5, mostramos as abordagens existentes na literatura bem como as propostas nesta tese para modificação do estado inicial de um problema de planejamento sem solução. No Capítulo 6, mostramos as abordagens existentes na literatura bem como as abordagens propostas nessa tese para modificação da meta de planejamento. No Capítulo 7, mostramos nossa proposta para modificação das précondições e efeitos das ações do domínio de planejamento baseada em um conjunto de problemas. Finalmente, no Capítulo 8, apresentamos as principais contribuições desse trabalho e apontamos alguns possíveis trabalhos futuros.

Capítulo 2

Raciocínio sobre Ações Determinísticas

Neste capítulo apresentamos como raciocinar sobre ações determinísticas. Na Seção 2.1 mostramos como realizar este raciocínio de forma progressiva e regressiva sobre um modelo *implícito e enumerativo* do domínio de planejamento, dado pela representação STRIPS para ações proposicionais e determinísticas. Neste caso, os estados e ações são representados como conjuntos de proposições e o raciocínio sobre as ações determinísticas é feito por meio de *operações entre conjuntos*. Na Seção 2.2 mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *implícito e simbólico* em que estados e ações são representados como *fórmulas da lógica proposicional*. Neste caso, o raciocínio sobre ações é realizado por meio de operações entre fórmulas da lógica proposicional e operações de quantificação sobre os valores das proposições (*Quantified Boolean Formulas* [Buning et al., 1995]).

2.1 Raciocínio sobre Modelos Implícitos e Enumerativos

Nesta seção mostramos como raciocinar sobre um modelo *implícito e enumerativo* de um domínio de planejamento determinístico. Na Seção 2.1.1 mostramos como representar ações proposicionais e determinísticas utilizando-se a linguagem STRIPS. Na Seção 2.1.2 mostramos como realizar a progressão de ações determinísticas e na Seção 2.1.3 mostramos como realizar a regressão de ações determinísticas.

$a_1 : \langle \textit{precond}(a_1) = \{p, q\};\ \langle \textit{add}(a_1) = \{p\};\ \textit{del}(a_1) = \{q\}\rangle\rangle$
$b :\ \langle \textit{precond}(b) = \{q\};\ \langle \textit{add}(b) = \{\};\ \textit{del}(b) = \{p\}\rangle\rangle$
$c :\ \langle \textit{precond}(c) = \{p\};\ \langle \textit{add}(c) = \{q\};\ \textit{del}(c) = \{\}\rangle\rangle$

Figura 2.1: Representação (implícita) do domínio de planejamento determinístico por meio de ações STRIPS referente à representação explícita mostrada na Figura 1.1(a).

2.1.1 Representação de Ações Determinísticas

STRIPS [Fikes and Nilsson, 1972] é uma linguagem de primeira ordem amplamente utilizada para representar ações determinísticas. Em uma versão proposicional da linguagem STRIPS, o domínio de planejamento é definido por um conjunto de *proposições* atômicas $\mathbb{P}$ que representam propriedades do mundo e um conjunto de *ações* $\mathbb{A}$, representando as habilidades do agente. Os *estados* são conjuntos de propriedades (fazendo-se a *suposição do mundo fechado* na qual expressa-se apenas as proposições que são verdadeiras, sendo que as proposições não representadas são consideradas falsas). O *estado inicial* s_0 de um problema de planejamento é dado por um conjunto completo de propriedades que definem um único estado no mundo. A *meta* de planejamento é dada por um conjunto incompleto de propriedades (i.e., um subconjunto de $\mathbb{P}$) que define um conjunto de estados que as satisfazem. As ações STRIPS são funções parciais de estados para estados, sendo que cada ação $a \in \mathbb{A}$ é caracterizada por conjuntos de propriedades sobre $\mathbb{P}$.

Definição 2.1. *(Ações determinísticas - representação* STRIPS*) Uma ação determinística α sobre um conjunto de proposições $\mathbb{P}$ é especificada por:*

$$\alpha = \langle \textit{precond}(\alpha), \textit{efeitos}(\alpha)\rangle$$

, em que precond(α) é um conjunto de precondições, representando as proposições que devem ser verdadeiras no estado corrente para que a ação α seja executada, e efeitos(α) representam como o estado é modificado após a execução da ação. Os efeitos são dados pelo par efeitos$(\alpha) = \langle$add(α), del$(\alpha)\rangle$ em que: add(α) é o conjunto de proposições que se tornam verdadeiras após a execução da ação e del(α) é o conjunto de proposições que se tornam falsa após α ser executada.

A Figura 2.1 exibe a representação STRIPS para as ações determinísticas a_1, b e c. Cada ação é dada pelo conjunto de precondições, conjunto dos efeitos positivos e conjunto dos efeitos negativos. Esta representação implícita correspondente à representação explícita mostrada na Figura 1.1(a).

2.1.2 Progressão de Ações Determinísticas

A *progressão* de um estado inicial x, por uma ação determinística a, produz um único estado sucessor $progr^a(x)$. Por exemplo, na representação explícita do domínio de planejamento da Figura 1.1(a), $progr^c(s_0) = s_1$. No entanto, para computar $progr^a(x)$ utilizando-se a representação implícita, é necessário primeiramente verificar ser a ação a é *aplicável* a um estado x, i.e., se $precond(a) \subseteq x$. Nesse caso, o estado sucessor alcançado pela execução da ação a em x é obtido pela inclusão dos efeitos positivos de a e pela subsequente exclusão dos efeitos negativos de a no estado x (i.e., $(x \setminus del(a)) \cup add(a)$). Se $precond(a) \not\subseteq x$, então não existe estado sucessor. Note que, como $del(a)$ contém todos os efeitos negativos da ação a, quando a é executada no estado x, todas as propriedades que valem em x e não são afetadas pela execução dessa ação continuam valendo no estado resultante (axioma de frame [Shanahan, 1997]). Formalmente, a progressão de um estado x por uma ação determinística a é dada por:

$$progr^a(x) = \begin{cases} (x \setminus del(a)) \cup add(a) & \text{se } \textit{precond}(a) \subseteq x \\ \emptyset & \text{c.c..} \end{cases} \tag{2.1}$$

Exemplo 2.1. *(Progressão do estado s_0 pela ação c) Na Figura 1.1(a), o estado sucessor de s_0 pela ação c é obtido usando a Equação 2.1 como a seguir:*

$$progr^c(s_0) = progr^c(\{p\}) = (\{p\} \setminus \{\}) \cup \{q\} = \{p, q\} = s_1.$$

A operação de progressão também podem ser estendida para conjuntos de estados. Seja X um conjunto de estados. A Equação 2.2 define o conjunto $progr^a(X)$ de estados que são sucessores de um conjunto de estados X, pela aplicação de uma ação a. Um sucessor é gerado para cada estado $x \in X$ no qual a ação a é aplicável, i.e.,

$$progr^a(X) = \bigcup_{x \in X} \{progr^a(x)\}. \tag{2.2}$$

Exemplo 2.2. *(Progressão do conjunto de estados $X = \{s_0, s_1\}$ pela ação c) Sendo $progr^c(s_0) = progr^c(\{p\}) = (\{p\} \setminus \{\}) \cup \{q\} = \{p, q\}$ e $progr^c(s_1) = progr^c(\{p, q\}) = (\{p, q\} \setminus \{\}) \cup \{q\} = \{p, q\}$. Então, $progr^a(s_0, s_1) = \{p, q\} \cup \{p, q\} = \{p, q\}$.*

2.1.3 Regressão de Ações Determinísticas

A *regressão* de um estado x, por uma dada ação determinística a, produz um conjunto de estados predecessores *regr*$^a(x)$. Por exemplo, na representação explícita do domínio de planejamento da Figura 1.1(a), temos que *regr*$^c(s_1) = \{s_0, s_1\}$. A regressão sobre uma representação implícita, no entanto, gera um estado abstrato que pode representar um conjunto de estados do domínio. Isto se deve ao fato de que as proposições não incluídas na representação do estado abstrato são consideradas desconhecidas, i.e., elas podem ser verdadeiras ou falsas [Alcázar et al., 2013]. Note a diferença da representação de estado utilizada na regressão daquela utilizada na progressão: na progressão é feita a supossição do mundo fechado (i.e., as proposições não inlcuídas na representação do estado são consideradas falsas) enquanto que na regressão esta suposição não é permitida. Assim, pode-se concluir que a regressão raciocina sobre estados parcialmente especificados (i.e., espaço de estados de crenças) e a progressão raciocina sobre estados completamente especificados [Geffner and Bonet, 2013].

Um conceito muito importante na regressão é a relevância de uma ação para um estado. Para que uma ação seja *relevante* a um estado x ela deve contribuir com as propriedades satisfeitas em x, i.e., no mínimo um dos efeitos das ações devem unificar com um elemento de x [Russell, 2009]. Além disso, a ação deve não ter um efeito que negue uma propriedade de x. Formalmente, dado um estado x, uma ação quando aplicada em um estado pode alcançar x se $add(a) \subseteq x$ e $del(a) \cap x = \emptyset$, i.e., a é relevante a x. O conjunto de estados predecessores de x pela ação a é obtido pela exclusão dos efeitos positivos de a em x e pela subsequente inclusão das precondições de a, i.e., $(x \setminus add(a)) \cup precond(a)$ [Ghallab et al., 2004]. Note que, como $del(a) \subseteq precond(a)$, todas as propriedades que deixam de valer em x devem valer em $regr^a(x)$. Caso a condição de relevância não seja satisfeita, não existe estado predecessor. Formalmente, a regressão de um estado x por uma ação a é dada por:

$$regr^a(x) = \begin{cases} (x \setminus add(a)) \cup precond(a) & \text{se} \quad add(a) \subseteq x \text{ e } del(a) \cap x = \emptyset \\ \emptyset & \text{c.c..} \end{cases} \tag{2.3}$$

Exemplo 2.3. *(Regressão do estado s_1 pela ação c) Na Figura 1.1(a), a regressão do estado s_1 pela ação c é obtida usando a Equação 2.3 como a seguir: regr*$^c(s_1) =$ *regr*$^c(\{p, q\}) = (\{p, q\} \setminus \{q\}) \cup \{p\} = \{p\} = \{p, \neg q\}, \{p, q\} = \{s_0, s_1\}$.

A operação de regressão também podem ser estendida para conjuntos de estados. Seja X um conjunto de estados. A Equação 2.4 define o conjunto $regr^a(X)$ de estados que são predecessores de um conjunto de estados X, segundo a ação a. Um predecessor é gerado para cada estado $x \in X$ para o qual a ação a é relevante, i.e.,

$$regr^a(X) = \bigcup_{x \in X} \{regr^a(x)\}. \tag{2.4}$$

Exemplo 2.4. *(Regressão do conjunto de estados $X = \{s_1, s_2\}$ pela ação c) Sendo $regr^c(s_1) = regr^c(\{p, q\}) = (\{p, q\} \setminus \{q\}) \cup \{p\} = \{p\} = \{\{p, \neg q\}, \{p, q\}\} = \{s_0, s_1\}$ e $regr^c(s_2) = \{\}$ (uma vez que c não é relevante para s_2). Então, $regr^c(\{s_1, s_2\}) = \{s_0, s_1\} \cup \{\} = \{s_0, s_1\}$.*

2.2 Raciocínio sobre Modelos Implícitos e Simbólicos

Nessa seção mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *implícito e simbólico*. Na Seção mostramos como representar estados, conjunto de estados e ações determinísticas STRIPS como fórmulas da lógica proposicional. As operações de progressão (Seção 2.2.2) e regressão (Seção 2.2.3) simbólicas são realizadas por meio de manipulação de fórmulas da lógica proposicional e operações de quantificação sobre os valores das proposições.

2.2.1 Representação Simbólica de Estados e Ações Determinísticas

A seguir, mostramos como representar estados, conjunto de estados e ações determinísticas como fórmulas da lógica proposicional.

Definição 2.2. *(Representação proposicional para estados e conjunto de estados) Seja $M = \langle S, L, T \rangle$ um sistema de transição de estados sobre um conjunto de proposições $\mathbb{P}$. A codificação de um estado $s \in S$, denotada por $\xi(s)$ é a fórmula*

$$\xi(s) = \bigwedge_{p \in L(s)} p, \tag{2.5}$$

e a codificação proposicional para um conjunto de estados $X \subseteq S$ é a fórmula

$$\xi(X) = \bigvee_{s \in X} \xi(s). \tag{2.6}$$

Exemplo 2.5. *(Representação simbólica de um conjunto de estados) O conjunto de estados* $S = \{s_0, s_1, s_2\}$ *do sistema de transição de estados da Figura 1.1(a) pode ser representado pela fórmula:*

$$\xi(S) = (p \wedge \neg q) \vee (p \wedge q) \vee (\neg p \wedge q).$$

Definição 2.3. *(Representação proposicional para ações determinísticas) A representação proposicional para uma ação* STRIPS *determinística* $\alpha = \langle \mathit{precond}(\alpha); \mathit{efeitos}(\alpha) \rangle$ *é um par de fórmulas* $\langle \xi(precond(\alpha)); \xi(\mathit{effects}(\alpha)) \rangle$ *tal que:*

- $\xi(precond(\alpha))$ *é uma conjunção de literais representando as precondições da ação* α*,*

$$\xi(\mathit{precond}(\alpha)) = \bigwedge_{p \,\in\, \mathit{precond}(\alpha)} p \qquad e, \tag{2.7}$$

- $\xi(\mathit{efeitos}(\alpha))$ *é uma conjunção de literais representando os efeitos de* α*, i.e.,*

$$\xi(\mathit{efeitos}(\alpha)) = \bigwedge_{d \,\in\, add(\alpha)} d \;\wedge \bigwedge_{r \,\in\, del(\alpha)} \neg r. \tag{2.8}$$

Exemplo 2.6. *(Representação proposicional para as ações* $a1, b$ *e* c*) As ações determinísticas* $a1, b$ *e* c *especificadas na Figura 2.1 podem ser representadas pelas seguintes fórmulas proposicionais:*

$$a1 = \langle p \wedge q; p \wedge \neg q \rangle; b = \langle q; \neg p \rangle; c = \langle p; q \rangle.$$

Fórmulas Booleanas Quantificadas

Para realizar o raciocínio simbólico é necessário utilizar uma extensão da lógica proposicional que permite a quantificação sobre os valores das proposições (QBF - *Quantified Boolean Formulas* [Buning et al., 1995]). As operações mais utilizadas para realizar cálculos simbólicos entre conjuntos de estados e ações (ou transições) são baseadas na quantificação existencial e universal de QBF [Rintanen, 2008]. Dada uma fórmula proposicional φ e uma proposição p que ocorre em φ, a quantificação existencial é definida como:

$$\exists p.\ \varphi \equiv \varphi[\top/p] \vee \varphi[\bot/p] \tag{2.9}$$

, em que $\varphi[\top/p]$ é obtida trocando-se a proposição p pelo valor $\top$ na fórmula φ e $\varphi[\bot\ /p]$ é obtida trocando-se p por $\bot$ na fórmula φ.

Exemplo 2.7. *(Quantificação Existencial de QBF) Seja $\varphi = (p \wedge q) \vee (p \wedge q)$ uma fórmula proposicional sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$, a quantificação existencial da fórmula φ sobre os valores da proposição p é dada por:*

$$\begin{aligned}
\exists p.\varphi &= \varphi[\top/p] \vee \varphi[\bot/p] \\
&= ((\top \wedge q) \vee (\top \wedge q)) \vee ((\bot \wedge q) \vee (\bot \wedge q)) \\
&= (q \vee q) \vee (\bot \vee \bot) \\
&= (q \vee q) \vee \bot \\
&= q.
\end{aligned}$$

Dada uma fórmula proposicional φ e uma proposição p que ocorre em φ, a quantificação universal é definida como:

$$\forall p.\ \varphi \equiv \varphi[\top/p] \wedge \varphi[\bot/p]. \tag{2.10}$$

Exemplo 2.8. *(Quantificação Universal de QBF) Seja $\varphi = (p \wedge q) \vee (p \wedge q)$ uma fórmula proposicional sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$, a quantificação universal da fórmula φ sobre os valores da proposição p é dada por:*

$$\begin{aligned}
\forall p.\varphi &= \varphi[\top/p] \wedge \varphi[\bot/p] \\
&= ((\top \wedge q) \vee (\top \wedge q)) \wedge ((\bot \wedge q) \vee (\bot \wedge q)) \\
&= (q \vee q) \wedge (\bot \vee \bot) \\
&= (q \vee q) \wedge \bot \\
&= \bot .
\end{aligned}$$

As quantificações também podem ser definidas para um conjunto de variáveis. Seja $B = \{b_1, b_2, \cdots, b_n\}$ um subconjunto das variáveis proposicionais que ocorrem em φ. Assim, $\exists B.[\varphi] \equiv \exists b_1.(\cdots \exists b_n.\varphi)$ e $\forall B.[\varphi] \equiv \forall b_1.(\cdots \forall b_n.\varphi)$.

Diagramas de Decisão Binária

Diagramas de Decisão Binária (BDDs) são uma forma canônica de representação para funções booleanas. Um BDD é similar a uma árvore de decisão: um grafo acíclico em que nós não terminais são rotulados com variáveis booleanas e nós terminais são rotulados com 0 ou 1. Para permitir uma representação compacta, as seguintes otimizações são realizadas: (i) remoção de nós

terminais e não terminais duplicados e; (ii) remoção de testes redundantes. Além disso, é fixada uma ordenação das variáveis . Aplicando-se estas remoções e fixando-se a ordem das variáveis, obtém-se um BDD *reduzido e ordenado* (chamado *Reduced and Ordering BDD* (ROBDD)). Por simplicidade, quando a partir de então mencionarmos um BDD, na verdade, estamos nos referindo a um RODD. BDDs permitem a representação compacta de funções booleanas, as quais somente possuem representações exponenciais tais como tabelas verdades e formulas normais conjuntivas [Huth and Ryan, 2004].

A Figura 2.2 mostra o BDD que representa a função booleana $(x_1 \Leftrightarrow x_2) \vee x_3$, usando a seguinte ordenação sobre as variáveis $x_1 < x_2 < x_3$. Cada nó não terminal é uma variável booleana e possui duas arestas partindo dele: a aresta sólida representa o valor verdadeiro e; a aresta pontilhada reepresenta o valor falso. Dada uma valoração qualquer para as variáveis x_1, s_2 e x_3, é possível calcular o valor da função booleana atravessando o BDD, iniciando da raiz e analisando o valor de cada nó: se a variável tem valor falso, deve-se seguir a linha pontilhada; caso contrário deve-se seguir a linha sólida. Por exemplo, a valoração $x_1 = 0, x_2 = 1, x_3 = 1$ leva para o nó terminal 1, então a fórmula $(x_1 \Leftrightarrow x_2) \vee x_3$ é *verdadeira* para esta valoração.

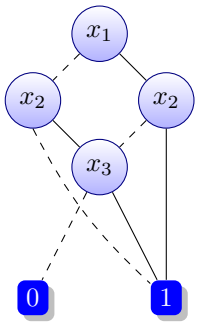


Figura 2.2: BDD B_f para a fórmula $f = (x_1 \Leftrightarrow x_2) \vee x_3$ com ordenação $x_1 < x_2 < x_3$ [Andersen, 1997].

2.2.2 Progressão Simbólica de Ações Determinísticas

Dados um conjunto de estados X e um conjunto de ações $\mathbb{A}$ representados como fórmulas proposicionais, [Fourman, 2000] propõe uma maneira de computar a progressão (e também a regressão) de forma simbólica. Além da representação proposicional dos estados e ações, é definido para cada ação $a = \langle \mathit{precond}(a); \mathit{efeitos}(a) \rangle \in \mathbb{A}$ um conjunto das proposições que ocorrem em $\mathit{efeitos}(a)$, denominado $\mathit{modifica}(\mathit{efeitos}(a))$. Por exemplo, $\mathit{modifica}(\mathit{efeitos}(a1)) =$

$\{p, q\}$, *modifica*(*efeitos*$(b)) = \{p\}$ e *modifica*(*efeitos*$(c)) = \{q\}$ são, respectivamente, o conjunto das proposições envolvidas nos efeitos das ações $a1, b$ e c (Figura 2.1).

Definição 2.4. *(Progressão simbólica de um conjunto de estados por uma ação determinística) Seja α uma ação determinística representada pelo par de fórmulas proposicionais $\langle \xi($*precond*$(a)); \xi($*efeitos*$(\alpha))\rangle$ e seja $\xi(X)$ a representação proposicional do conjunto de estados X, a progressão de X por α é [Fourman, 2000]*

$$sprogr^{\alpha}(X) = \exists \textit{modifica(efeitos}(\alpha)).(\xi(X) \wedge \xi(\textit{precond}(\alpha))) \wedge \xi(\textit{efeitos}(\alpha)).$$

Exemplo 2.9. *Dado o conjunto de ações determinísticas $\mathbb{A} = \{a_1, b, c\}$ (Figura 2.1), $sprogr^{a1}(X)$, $sprogr^{b}(X)$ e $sprogr^{c}(X)$ para $\xi(X) = p \wedge q$ é calculada como a seguir:*

$$\begin{aligned}
sprogr^{a_1}(X) &= \exists \textit{modifica(efeitos}(a1)).(\xi(X) \wedge \xi(precond(a1))) \wedge \xi(\textit{efeitos}(a1)) \\
&= \exists p.\exists q.(p \wedge q \wedge p \wedge q) \wedge (p \wedge \neg q) \\
&= \exists p.\exists q.(p \wedge q) \wedge (p \wedge \neg q) \\
&= \exists p.((p \wedge \top) \vee (p \wedge \bot)) \wedge (p \wedge \neg q) \\
&= \exists p.((p \wedge \top) \vee (p \wedge \bot)) \wedge (p \wedge \neg q) \\
&= \exists p.(p) \wedge (p \wedge \neg q) \\
&= (\top \vee \bot) \wedge (p \wedge \neg q) \\
&= (p \wedge \neg q);
\end{aligned}$$

$$\begin{aligned}
sprogr^{b}(X) &= \exists \textit{modifica(efeitos}(b)).(\xi(X) \wedge \xi(precond(b))) \wedge \xi(\textit{efeitos}(b)) \\
&= \exists p.(p \wedge q \wedge p) \wedge \neg p \\
&= q \wedge \neg p \\
&= q;
\end{aligned}$$

$$\begin{aligned}
sprogr^{c}(X) &= \exists \textit{modifica(efeitos}(c)).(\xi(X) \wedge \xi(precond(c))) \wedge \xi(\textit{efeitos}(c)) \\
&= \exists q.(p \wedge q \wedge p) \wedge q \\
&= p \wedge q.
\end{aligned}$$

2.2.3 Regressão Simbólica de Ações Determinísticas

Dada a representação proposicional de um conjunto de estados X e de uma ação determinística a, esta seção mostra como computar a regressão de forma simbólica [Fourman, 2000].

Definição 2.5. *(Regressão simbólica de um conjunto de estados por uma ação determinística) Seja α uma ação determinística representada pelo par de fórmulas proposicionais $\langle \xi(\textit{precond}(\alpha)); \xi(\textit{efeitos}(\alpha))\rangle$ e seja $\xi(X)$ a representação proposicional do conjunto de estados X, a regressão de X por α é [Fourman, 2000]:*

$$\textit{sregr}^{\alpha}(X) = \xi(precond(\alpha)) \wedge \exists \textit{modifica(efeitos}(\alpha)).(\xi(\textit{efeitos}(\alpha)) \wedge \xi(X)).$$

Exemplo 2.10. *Dado o conjunto de ações determinísticas $\mathbb{A} = \{a_1, b, c\}$ (Figura 2.1), $sregr^{a1}(X)$, $sregr^{b}(X)$ e $sregr^{c}(X)$ para $\xi(X) = \neg p \wedge q$ é calculada como a seguir:*

$$\begin{aligned} sregr^{a_1}(X) &= \xi(precond(a1)) \wedge \exists \textit{modifica(efeitos}(a1)).(\xi(\textit{efeitos}(a1)) \wedge \xi(X)) \\ &= (p \wedge q) \wedge \exists\{p,q\}.((p \wedge \neg q) \wedge (\neg p \wedge q)) \\ &= (p \wedge q) \wedge \exists\{p,q\}.(\bot) \\ &= \bot; \end{aligned}$$

$$\begin{aligned} sregr^{b}(X) &= \xi(precond(b)) \wedge \exists \textit{modifica(efeitos}(b)).(\xi(\textit{efeitos}(b)) \wedge \xi(X)) \\ &= q \wedge \exists p.(\neg p \wedge (\neg p \wedge q)) \\ &= q \wedge \exists p.(\neg p \wedge q) \\ &= q; \end{aligned}$$

$$\begin{aligned} sregr^{c}(X) &= \xi(precond(c)) \wedge \exists \textit{modifica(efeitos}(c)).(\xi(\textit{efeitos}(c)) \wedge \xi(X)) \\ &= p \wedge \exists q.(q \wedge (\neg p \wedge q)) \\ &= p \wedge \exists q.(\neg p \wedge q) \\ &= p \wedge \neg p \\ &= \bot\ . \end{aligned}$$

Capítulo 3

Raciocínio sobre Ações Não Determinísticas

Neste capítulo apresentamos abordagens que tratam do raciocínio sobre ações não-determinísticas. Na Seção 3.1 mostramos como realizar este raciocínio de forma progressiva e regressiva (com operações de *progressão* e *regressão* de ações) sobre um modelo *implícito e enumerativo*, dado pela representação STRIPS para ações não determinísticas. Neste caso, as ações e estados são representadas como *conjuntos de proposições* e o raciocínio progressivo e regressivo é feito por meio de *operações entre conjuntos*. Na Seção 3.2 mostramos como realizar o raciocínio progressivo e regressivo (com operações de *progressão e regressão simbólicas* de ações) sobre um modelo *implícito e simbólico* em que estados e ações são representados como *fórmulas da lógica proposicional*. Neste caso, a progressão e regressão são realizadas por meio de operações entre fórmulas da lógica proposicional. Além disso, é necessário realizar a quantificação sobre os valores das proposições usando-se QBF. Na Seção 3.3, mostramos como raciocinar progressivamente e regressivamente (com, respectivamente, operações de *imagem e pré-imagem de um conjunto de estados*) sobre um modelo *explícito e enumerativo*, dado por uma estrutura de Kripke [Kripke, 1963] ou por um sistema de transição de estados rotulado por ações (tais como os mo-delos da Figura 1.1). Neste caso, a imagem e pré-imagem de um conjunto de estados são computadas com operações entre conjuntos. Na Seção 3.4, mostramos como realizar o raciocínio progressivo e regressivo (com, respectivamente, operações de imagem e pré-imagem simbólicas de um conjunto de estados) sobre um modelo *explícito e simbólico* em que os estados e transições do modelo explícito são representados por fórmulas da lógica proposicional. Neste caso, as operações de

imagem e pré-imagem simbólicas de um conjunto de estados são realizadas com operações sobre fórmulas da lógica proposicional e quantificação sobre os valores das proposições (QBF).

3.1 Raciocínio Implícito e Enumerativo sobre Ações Não Determinísticas

Nesta seção mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *implícito e enumerativo*. Na Seção 3.1.1 mostramos como representar ações não determinísticas utilizando uma linguagem semelhante à linguagem STRIPS. Na Seção 3.1.2 definimos a operação de progressão para ações não-determinísticas. Na Seção 3.1.3 definimos a operação de regressão para ações não determinísticas. Ambas são definidas por meio de operações entre conjuntos.

3.1.1 Representação de Ações Não Determinísticas

No Capítulo 2 utilizamos a linguagem STRIPS para representar ações deterministicas. Neste capítulo, utilizamos uma extensão desta linguagem capaz de expressar os possíveis efeitos não determinísticos resultantes da execução de uma ação.

Definição 3.1. *(Ações não determinísticas - representação baseada em* STRIPS*) Com base nas ações determinísticas* STRIPS*, uma ação não determinística é representada por* $\alpha = \langle precond(\alpha); efeitos(\alpha)\rangle$ *em que* $precond(\alpha)$ *é o conjunto das precondições de* α *e* $efeitos(\alpha)$ *é o conjunto de efeitos não determinísticos de* α *tal que* $efeitos(\alpha) = \{e_1, e_2, \cdots, e_k\}$*. Cada* $e_i \in efeitos(\alpha)$ *tem seu próprio conjunto de efeitos positivos e negativos dado pelo par* $\langle add(\alpha, e_i), del(\alpha, e_i)\rangle$*.*

Na Figura 3.1, a ação a_2 é um exemplo de ação não determinística representada segundo a extensão da linguagem STRIPS dada pela Definição 3.1. Esta ação tem dois efeitos não determinísticos: e_1 e e_2. O efeito e_1 é dado pela sua lista de efeitos positivos e negativos, representados pelo par $\langle add(a_2, e_1) = \{p\};\ del(a_2, e_1) = \{q\}\rangle$. Já o efeito e_2 é dado pelo par $\langle add(a_2, e_2) = \{q\};\ del(a_2, e_2) = \{p\}\rangle$.

3.1.2 Progressão de Ações Não Determinísticas

A *progressão* de um estado inicial x, por uma ação não determinística a, produz um conjunto de possíveis estados sucessores $progr^a(x)$. Por exemplo, considerando a representação explícita

$a_2 : \langle \textit{precond}(a_2) = \{p, q\};\ \{\ \langle\ \textit{add}(a_2, e_1) = \{p\};\ \textit{del}(a_2, e_1) = \{q\}\rangle\ ,$
$\langle \textit{add}(a_2, e_2) = \{q\};\ \textit{del}(a_2, e_2) = \{p\}\rangle\ \}\rangle$
$b : \langle \textit{precond}(b) = \{q\};\ \langle \textit{add}(b) = \emptyset;\ \textit{del}(b) = \{p\}\rangle\rangle$
$c : \langle \textit{precond}(c) = \{p\};\ \langle \textit{add}(c) = \{q\};\ \textit{del}(c) = \emptyset\rangle\rangle$

Figura 3.1: Representação (implícita) do domínio de planejamento não determinístico por meio de ações STRIPS referente à representação explícita mostrada na Figura 1.1(b).

do domínio da Figura 1.1(b), $progr^{a2}(s_1) = \{s_0, s_2\}$. Na representação implícita (Figura 3.1), uma ação a é aplicada em um estado x se *precond*$(a) \subseteq x$ e o conjunto dos estados sucessores é gerado pela união dos estados sucessores calculados para cada um dos efeitos possíveis de a. Se *precond*$(a) \not\subseteq x$, então não existe estado sucessor. Formalmente, a progressão de um estado x por uma ação não determinística a é dada pela Equação 3.1:

$$progr^a(x) = \begin{cases} \bigcup_{e\ \in\ \textit{efeitos}(a)} \{(x \setminus \textit{del}(a, e)) \cup \textit{add}(a, e)\} & \text{se } \textit{precond}(a) \subseteq x, \\ \emptyset & \text{c.c..} \end{cases} \quad (3.1)$$

Exemplo 3.1. *(Progressão dos estados s_0, s_1 e s_2 segundo a ação $a2$) Na Figura 1.1(b), o conjunto de estados sucessores de s_0, s_1 e s_2 gerados pela progressão da ação não determinística $a2$ é obtido usando a Equação 3.1 como a seguir:*

$$\begin{aligned} progr^{a2}(s_0) &= \emptyset\ ,\textit{pois precond}(a2) \not\subseteq s_0; \\ progr^{a2}(s_1) &= \{(\{p, q\} \setminus \{q\}) \cup \{p\}\} \cup \{(\{p, q\} \setminus \{p\}) \cup \{q\}\} \\ &= \{p\} \cup \{q\} = \{s_0\} \cup \{s_2\} = \{s_0, s_2\}; \\ progr^{a2}(s_2) &= \emptyset\ ,\textit{pois precond}(a2) \not\subseteq s_2. \end{aligned}$$

A Equação 3.2 estende a operação de progressão não determinística para um conjunto de estados $X = \{x_1, x_2, \cdots, x_n\}$, i.e.,

$$progr^a(X) = \bigcup_{x \in X} progr^a(x), \quad (3.2)$$

que computa o conjunto de estados que são possíveis sucessores de cada estado $x \in X$ pela ação não determinística a.

Exemplo 3.2. *(Progressão do conjunto de estados $X = \{s_0, s_1\}$ pela ação $a2$) Como $progr^{a2}(s_0) = \emptyset$ e $progr^{a2}(s_1) = \{s_0, s_2\}$. Então, $progr^{a2}(\{s_0, s_1\}) = \emptyset \cup \{s_0, s_2\} = \{s_0, s_2\}$;*

A Equação 3.3 define a operação de progressão não determinística de um conjunto de estados $X = \{x_1, x_2, \cdots, x_n\}$ segundo um conjunto de ações $a \in \mathbb{A}$, i.e.,

$$progr(X) = \bigcup_{a \in \mathbb{A}} progr^a(X), \tag{3.3}$$

que computa o conjunto de estados que são possíveis sucessores de X por cada ação $a \in \mathbb{A}$.

Exemplo 3.3. *(Progressão de $X = \{s_0, s_1\}$ segundo o conjunto de ações $\mathbb{A} = \{a2, b, c\}$) Como $progr^{a2}(X) = \{s_0, s_2\}$, $progr^b(X) = \{s_2\}$ e $progr^c(X) = \{s_0, s_1\}$, então $progr(X) = \{s_0, s_2\} \cup \{s_2\} \cup \{s_0, s_1\} = \{s_0, s_1, s_2\}$.*

3.1.3 Regressão de Ações Não Determinísticas

A *regressão* de um único estado x, com relação a uma ação não determinística a, produz um conjunto de estados predecessores *regr*$^a(x)$. Na Figura 1.1(b) (representação explícita do domínio de planejamento), temos que *regr*$^{a2}(s_0) = \{s_1\}$. Utilizando uma representação implícita (como o conjunto de ações da Figura 3.1), no entanto, é possível calcular os estados predecessores de um estado x segundo uma ação não determinística a. Basicamente, se algum efeito da ação a é *relevante* para o estado x, então os estados predecessores são obtidos considerando-se este efeito. Formalmente, a regressão de um estado x por uma ação não determinística a é dada pela Equação 3.4:

$$\textit{regr}^a(x) = \begin{cases} (x \setminus \textit{add}(a,e)) \cup \textit{precond}(a) & \text{se} \quad \exists e \in \textit{efeitos(a)} \text{ tal que } \textit{add}(a,e) \subseteq x \\ & \qquad\qquad \text{e } del(a,e) \cap x = \emptyset; \\ \emptyset & , \quad \text{c.c..} \end{cases} \tag{3.4}$$

Exemplo 3.4. *(Regressão de um estado pela ação não determinística $a2$) Na Figura 1.1(a), o estado que precede s_0 pela ação $a2$ pode ser obtido por meio da Equação 3.4 como a seguir:*

$$\textit{regr}^{a2}(s_0) = \textit{regr}^c(\{p\}) = (\{p\} \setminus \{p\}) \cup \{p, q\} = \{p, q\}$$

, uma vez que somente o efeito $e1$ *da ação* $a2$ *é relevante para o estado* s_0*.*

Também é possível calcular a regressão de um conjunto de estados. Seja $X = \{x_1, \cdots, x_n\}$ um conjunto de estados e a uma ação não determinística relevante a pelo menos um dos estados em X. Ao ser executada em um estado, a ação não determinística a pode levar *necessariamente* a um estado em X ou, então, pode levar *possivelmente* a um estado em X (e possivelmente a um estado fora de X). Dada uma ação não determinística a, o conjunto de estados predecessores que levam possivelmente a estados em X é computado pela *regressão fraca* de X segundo a. Por outro lado, o conjunto de estados predecessores que levam necessariamente a estados em X é computado pela *regressão forte* de X segundo a ação a.

Definição 3.2. *(Regressão fraca de um conjunto de estados por uma ação não determinística) Sejam* S *o conjunto de estados do domínio de planejamento,* X *um subconjunto de* S *e* α *uma ação não determinística. A regressão fraca de* X *segundo a ação* α*, denotada por* $regr^{\alpha}_{fraca}(X)$*, é o conjunto dos estados a partir dos quais um estado sucessor em* X *é possivelmente alcançado como resultado da execução de* α*, isto é:*

$$regr^{a}_{fraca}(X) = \{y \in S \ : \ progr^{a}(y) \cap X \neq \emptyset\}. \tag{3.5}$$

Exemplo 3.5. *(Regressão fraca do conjunto de estados* $X = \{s_1, s_2\}$ *segundo* $a2$*) A regressão fraca de* X *segundo a ação não determinística* $a2$ *é* $regr^{a2}_{fraca}(\{s_1, s_2\}) = \{s_1\}$*, uma vez que* $progr^{a2}(s_0) \cap X = \emptyset$*,* $progr^{a2}(s_2) \cap X = \emptyset$ *e* $progr^{a2}(s_1) \cap X = \{s_2\} \neq \emptyset$*. Isto é, a ação* $a2$ *quando aplicada ao estado* s_1 *leva possivelmente ao estado* $s_2 \in X$*.*

Definição 3.3. *(Regressão forte de um conjunto de estados por uma ação não determinística) Sejam* S *o conjunto de estados do domínio de planejamento,* X *um subconjunto de* S *e* α *uma ação não determinística. A regressão forte de* X *segundo a ação* α*, denotada por* $regr^{\alpha}_{forte}(X)$*, é o conjunto dos estados a partir dos quais um estado sucessor em* X *é necessariamente alcançado como resultado da execução de* α*, isto é:*

$$regr^{\alpha}_{forte}(X) = \{y \in S \ : \ \emptyset \neq progr^{\alpha}(y) \subseteq X\}.$$

Exemplo 3.6. *(Regressão forte do conjunto de estados* $X = \{s_1, s_2\}$ *segundo* $a2$*) A regressão forte de* X *segundo a ação não determinística* $a2$ *é* $regr^{a2}_{forte}(\{s_1, s_2\}) = \emptyset$*, uma vez que* $progr^{a2}(s_0) = progr^{a2}(s_2) = \emptyset$ *e* $progr^{a2}(s_1) = \{s_0, s_2\} \nsubseteq X$*. Isto é, a ação* $a2$ *quando aplicada ao estado* s_1 *leva possivelmente ao estado* $s_0 \notin X$*.*

Também é possível calcular a regressão fraca e forte de um estados X segundo um conjunto de ações $\mathbb{A}$, como mostrado nas Equações 3.6 e 3.7. A regressão fraca de um conjunto de estados X segundo um conjunto de ações $\mathbb{A}$ pode ser obtida em termos da regressão fraca de X segundo cada $a \in \mathbb{A}$, como a seguir:

$$regr_{fraca}(X) = \bigcup_{a \in \mathbb{A}} regr^{a}_{fraca}(X). \tag{3.6}$$

Exemplo 3.7. *(Regressão fraca de $X = \{s_1, s_2\}$ segundo o conjunto de ações $\mathbb{A} = \{a2, b, c\}$) Como $regr^{a2}_{fraca}(X) = \{s_1\}$, $regr^{b}_{fraca}(X) = \{s_1, s_2\}$ e $regr^{c}_{fraca}(X) = \{s_1\}$, então $regr_{fraca}(X)$ $= \{s_1\} \cup \{s_1, s_2\} \cup \{s_1\} = \{s_1, s_2\}$.*

A regressão forte de um conjunto de estados X segundo um conjunto de ações $\mathbb{A}$ é calculada como a seguir:

$$regr_{forte}(X) = \bigcup_{a \in \mathbb{A}} regr^{a}_{forte}(X). \tag{3.7}$$

Exemplo 3.8. *(Regressão forte de $X = \{s_1, s_2\}$ segundo o conjunto de ações $\mathbb{A} = \{a2, b, c\}$) Como $regr^{a2}_{forte}(X) = \emptyset$, $regr^{b}_{forte}(X) = \{s_1, s_2\}$ e $regr^{c}_{forte}(X) = \{s_1\}$, então $regr_{forte}(X) =$ $\emptyset \cup \{s_1, s_2\} \cup \{s_1\} = \{s_1, s_2\}$.*

3.2 Raciocínio Implícito e Simbólico sobre Ações Não Determinísticas

Nesta seção mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *implícito e simbólico* do domínio de planejamento. Na Seção 3.2.1 mostramos como representar ações não determinísticas como fórmulas da lógica proposicional. Na Seção 3.2.2 definimos a operação de progressão simbólica de ações. Na Seção 3.2.3 definimos a operação de regressão simbólica de ações. Em ambas, é necessário realizar operações entre fórmulas da lógica proposicional e quantificação sobre os valores de proposições.

3.2.1 Representação Proposicional para Ações Não Determinísticas

As ações não determinísticas STRIPS (Definição 3.1) podem ser representadas como fórmulas da lógica proposicional. Essa representação, juntamente com a representação proposicional de um conjunto de estados, permite a definição das operações de progressão e regressão de forma simbólica.

Definição 3.4. *(Representação proposicional para ações não determinísticas) Uma ação não determinística* $\alpha = \langle \mathit{precond}(\alpha); \mathit{efeitos}(\alpha) \rangle$ *pode ser representada por um par de fórmulas proposicionais* $\alpha = \langle \xi(\mathit{precond}(\alpha); \xi(\mathit{efeitos}(\alpha)) \rangle$ *tal que:*

- $\xi(\mathit{precond}(\alpha))$ *é uma conjunção de literais representando as pré-condições de* α*, i.e.,*

$$\xi(\mathit{precond}(\alpha)) = \bigwedge_{p \,\in\, \mathit{precond}(\alpha)} p \quad e, \tag{3.8}$$

- $\xi(\mathit{efeitos}(\alpha))$ *é uma disjunção de todos os efeitos não determinísticos de* α*, i.e.,*

$$\bigvee_{e_i \,\in\, \mathit{efeitos}(\alpha)} \Big(\bigwedge_{d \,\in\, \mathit{add}(\alpha, e_i)} d \;\wedge \bigwedge_{r \,\in\, \mathit{del}(\alpha, e_i)} \neg r \Big). \tag{3.9}$$

Exemplo 3.9. *(Representação proposicional para a ação* $a2$*) A ação não determinística* $a2$ *da Figura 3.1 pode ser representada como:*

$$a2 = \langle p \wedge q; (p \wedge \neg q) \vee (\neg p \wedge q) \rangle.$$

Para realizar o cálculo simbólico da progressão e regressão de ações não determinísticas, cada um dos efeitos da ação deve possuir o seu conjunto *modifica* das proposições que ocorrem neste efeito. Assim, sendo $\mathit{efeitos}(\alpha) = \{e_1, \cdots, e_n\}$ o conjunto dos efeitos não determinísticos de uma ação $\alpha \in \mathbb{A}$, então para cada $e_i \in \mathit{efeitos}(\alpha)$ há um conjunto $\mathit{modifica}(\alpha, e_i)$ correspondente. O conjunto de todas as proposições envolvidas nos efeitos não determinísticos de uma ação α é obtido pela união de cada $\mathit{modifica}(\alpha, e_i)$, como mostrado a seguir:

$$\mathit{modifica}(\mathit{efeitos}(\alpha)) = \bigcup_{e_i \,\in\, \mathit{efeitos}(\alpha)} \mathit{modifica}(\alpha, e_i). \tag{3.10}$$

Por exemplo, o conjunto *modifica* da ação não determinística a_2 (Figura 3.1), que possui dois efeitos não determinísticos e_1 e e_2, é dado por: *modifica*$(a2) =$ *modifica*$(a2, e_1) \cup$ *modifica*$(a2, e_2) = \{p, q\} \cup \{p, q\} = \{p, q\}$.

3.2.2 Progressão Simbólica de Ações Não Determinísticas

A *progressão simbólica* de um conjunto de estados por uma ação não determinística pode ser computada pela mesma fórmula da progressão simbólica para ações determinísticas (Definição 2.4). No entanto, no caso não determinístico a fórmula que representa o efeito é uma disjunção das fórmulas que representam cada um dos efeitos da ação (como mostrado na Definição 3.4) e, além disso, o conjunto *modifica* das proposições envolvidas nos efeitos é a união das proposições envolvidas em cada um dos efeitos não determinísticos (como mostrado na Equação 3.10).

Definição 3.5. *(Progressão simbólica de um conjunto de estados por uma ação não determinística) Seja α uma ação não determinística representada por um par de fórmulas proposicionais $\langle \xi($*precond*$(a)); \xi($*efeitos*$(a))\rangle$ (de acordo com a Definição 3.4) e seja $\xi(X)$ a representação proposicional de um conjunto de estados X (de acordo com a Equação 2.6), a progressão de X segundo a ação α é dada por:*

$$\textit{sprogr}^a(X) = \exists \textit{modifica}(\textit{efeitos}(\alpha)).(\xi(\textit{precond}(a)) \wedge \xi(X)) \wedge \xi(\textit{efeitos}(\alpha)). \qquad (3.11)$$

A Equação 3.11, que define a progressão simbólica de um conjunto de estados X segundo uma ação não determinística α, pode ser analisada da seguinte forma. Primeiro, a conjunção $\xi($*precond*$(\alpha)) \wedge \xi(X)$ seleciona o subconjunto dos estados em X para os quais a ação α é aplicável. Se α não é aplicável em nenhum estado $x \in X$, então $\xi($*precond*$(\alpha)) \wedge \xi(X) = \perp$ e, consequentemente, *sprogr*$^{\alpha}(X) = \perp$. No entanto, se α é aplicável em algum estado $x \in X$, então $\xi($*precond*$(\alpha)) \wedge \xi(X) \neq \perp$ (note que uma análise similar é feita na Equação 3.1, quando é verificado se *precond*$(\alpha) \subseteq x$). Após a conjunção, a quantificação existencial sobre os valores das proposições do conjunto *modifica*(*efeitos*$(\alpha))$ elimima da fórmula resultante as proposições que ocorrem em qualquer efeito não determinístico de α. Note que uma eliminação similar é feita na Equação 3.1, i.e., os elementos do conjunto *del*(a, e) são eliminados de x. Finalmente, é feita a conjunção da fórmula resultante com a fórmula representando os efeitos não determinísticos da ação α (note que na Equação 3.1 isto é feito pela união do conjunto *add*(a, e)).

Exemplo 3.10. *Dados o conjunto de estados $X = \{s_1\}$ (Figura 1.1(b)), representado pela fórmula proposicional $\xi(X) = \{p \wedge q\}$, e a ação não determinística $a2$, representada pelo par*

de fórmulas $\langle \xi(\mathit{precond}(a2)); \xi(\mathit{efeitos}(a2))\rangle = \langle (p \wedge q); (p \wedge q) \vee (\neg p \wedge q)\rangle$*, a progressão de* X *pela ação* $a2$ *é calculada como a seguir:*

$$\begin{aligned} sprogr^{a2}(\xi(X)) &= \exists \mathit{modifica}(\mathit{efeitos}(\alpha)).(\xi(\mathit{precond}(a)) \wedge \xi(X)) \wedge \xi(\mathit{efeitos}(\alpha)) \\ &= \exists p.\exists q.(p \wedge q \wedge p \wedge q) \wedge ((p \wedge q) \vee (\neg p \wedge q)) \\ &= \exists p.\exists q(p \wedge q) \wedge ((p \wedge q) \vee (\neg p \wedge q)) \\ &= \top \wedge ((p \wedge q) \vee (\neg p \wedge q)) \\ &= (p \wedge q) \vee (\neg p \wedge q). \end{aligned}$$

Definição 3.6. *(Progressão simbólica de um conjunto de estados por um conjunto de ações) Seja* $\mathbb{A}$ *um conjunto de ações e seja* $\xi(X)$ *a representação proposicional de um conjunto de estados* X*, a progressão simbólica de* X *pelo conjunto de ações* $\mathbb{A}$ *é dada por:*

$$\mathit{sprogr}(X) = \bigvee_{a \in \mathbb{A}} \mathit{sprogr}^a(\xi(X)). \tag{3.12}$$

A Equação 3.3 é a versão enumerativa da Equação 3.12. Enquanto que na primeira equação é feita a união dos conjuntos obtidos pela progressão de cada ação $a \in \mathbb{A}$; na segunda, é feita a disjunção das fórmulas obtidas pela progressão de cada $a \in \mathbb{A}$.

3.2.3 Regressão Simbólica de Ações Não Determinísticas

Nessa seção, dada a representação proposicional de um conjunto de estados X e de uma ação não determinística a, definimos como computar de forma simbólica o conjunto de estados predecessores que levam *possivelmente* a estados em X (*regressão simbólica fraca* de X segundo a) e o conjunto de estados predecessores que levam *necessariamente* a estados em X (*regressão forte* de X segundo a).

Definição 3.7. *(Regressão simbólica fraca de um conjunto de estados por uma ação não determinística) Seja* α *uma ação não determinística representada por um par de fórmulas proposicionais* $\langle \xi(\mathit{precond}(a)); \xi(\mathit{efeitos}(a))\rangle$ *(de acordo com a Definição 3.4) e seja* $\xi(X)$ *a representação proposicional de um conjunto de estados* X *(de acordo com a Equação 2.6), a regressão simbólica fraca de* X *segundo a ação* α *é dada por:*

$$\mathit{sregr}^a_{\mathit{fraca}}(X) = \xi(\mathit{precond}(a)) \wedge \exists \mathit{modifica}(\mathit{efeitos}(\alpha)).(\xi(\mathit{efeitos}(\alpha)) \wedge \xi(X)). \tag{3.13}$$

A Equação 3.13, que define a regressão simbólica fraca de um conjunto de estados X segundo uma ação não determinística α, pode ser analisada da direita para esquerda. Primeiro, a conjunção $\xi(\textit{efeitos}(\alpha)) \wedge \xi(X))$ seleciona o subconjunto dos estados em X para os quais os efeitos da ação α são relevantes. Se nenhum efeito de α é relevante para X, então $\xi(e) \wedge \xi(X) = \bot$ e, consequentemente, $sregr^{a}_{\textit{fraca}}(X) = \bot$. No entanto, se algum efeito de α é relevante para algum estado em X, então $\xi(e) \wedge \xi(X) \neq \bot$ (note que uma análise similar é feita na Equação 3.4, quando é verificado se $\exists e \in \textit{efeitos}(\alpha)$ tal que e é relevante para x). Após a conjunção, a quantificação existencial sobre os valores das proposições do conjunto $\textit{modifica}(\textit{efeitos}(\alpha))$ elimima da fórmula que representa o conjunto de estados relevantes todas as variáveis que ocorrem nos efeitos de α. Note que uma eliminação similar é feita na Equação 3.4, no entanto, naquela equação a eliminação é feita pela operação de diferença de conjuntos. Finalmente, é feita a conjunção da fórmula resultante com a fórmula representando as pré-condições da ação (note que na Equação 3.4 isto é feito pela união do conjunto das pré-condições).

Exemplo 3.11. *Dados o conjunto de estados $X = \{s_2\}$ (Figura 1.1(b)), representado pela fórmula proposicional $\xi(X) = \{\neg p \wedge q\}$, e a ação não determinística $a2$, representada pelo par de fórmulas $\langle \xi(\textit{precond}(a2)); \xi(\textit{efeitos}(a2))\rangle = \langle (p \wedge q); (p \wedge q) \vee (\neg p \wedge q)\rangle$, a regressão simbólica fraca de X pela ação $a2$ é calculada como a seguir:*

$$\begin{aligned} sregr^{a2}_{\textit{fraca}}(\xi(X)) &= \xi(\textit{precond}(a2)) \wedge \exists \textit{modifica}(\textit{efeitos}(a2)).(\xi(\textit{efeitos}(a2)) \wedge \xi(X))) \\ &= (p \wedge q) \wedge (\exists\{p,q\}.(((p \wedge \neg q) \vee (\neg p \wedge q)) \wedge (\neg p \wedge q)) \\ &= (p \wedge q) \wedge (\exists p.\exists q.(\neg p \wedge q)) \\ &= (p \wedge q) \wedge \top \\ &= (p \wedge q). \end{aligned}$$

Definição 3.8. *(Regressão simbólica fraca de um conjunto de estados por um conjunto de ações) Seja $\mathbb{A}$ um conjunto de ações e seja $\xi(X)$ a representação proposicional de um conjunto de estados X, a regressão simbólica fraca de X pelo conjunto de ações $\mathbb{A}$ é dada por:*

$$sregr_{\textit{fraca}}(X) = \bigvee_{a \in \mathbb{A}} sregr^{a}_{\textit{fraca}}(\xi(X)). \tag{3.14}$$

A Equação 3.6 é a versão enumerativa da Equação 3.14. Enquanto que na primeira equação é feita a união dos conjuntos obtidos pela regressão fraca de cada ação $a \in \mathbb{A}$; na segunda, é feita a disjunção das fórmulas obtidas pela regressão fraca de cada $a \in \mathbb{A}$.

Definição 3.9. *(Regressão simbólica forte de um conjunto de estados por uma ação) Seja α uma ação não determinística representada por um par de fórmulas proposicionais $\langle \xi(\mathit{precond}(a)); \xi(\mathit{efeitos}(a))\rangle$ (de acordo com a Definição 3.4) e seja $\xi(X)$ a representação proposicional de um conjunto de estados X (de acordo com a Equação 2.6), a regressão simbólica forte de X segundo a ação α é dada por:*

$$\mathit{sregr}^{a}_{\mathit{forte}}(\xi(X)) = \xi(\mathit{precond}(a)) \wedge \forall \mathit{modifica(efeitos}(\alpha)).(\mathit{efeitos}(\alpha) \rightarrow \xi(X)). \quad (3.15)$$

Na Equação 3.15, que define a regressão simbólica forte de um conjunto de estados X segundo uma ação não determinística α, a implicação $\mathit{efeitos}(\alpha) \rightarrow \xi(X)$ determina se todos os efeitos da ação α levam apenas a estados em X (e não a estados fora de X). Em seguida, a quantificação universal elimina da fórmula resultante as variáveis pertencentes ao conjunto $\mathit{modifica(efeitos}(\alpha))$. Por fim, a conjunção da fórmula resultante com as pré-condições da ação gera a representação proposicional do conjunto de estados antecessores de X.

Exemplo 3.12. *Dados o conjunto de estados $X = \{s_2\}$ (Figura 1.1(b)), representado pela fórmula proposicional $\xi(X) = \{\neg p \wedge q\}$, e a ação não determinística $a2$, representada pelo par de fórmulas $\langle \xi(\mathit{precond}(a2)); \xi(\mathit{efeitos}(a2))\rangle = \langle (p \wedge q); (p \wedge q) \vee (\neg p \wedge q)\rangle$, a regressão simbólica fraca de X pela ação $a2$ é calculada como a seguir:*

$$\begin{aligned}
\mathit{sregr}^{a2}_{\mathit{forte}}(\xi(X)) &= \xi(\mathit{precond}(a2)) \wedge \forall \mathit{modifica(efeitos}((a2)).(\xi(\mathit{efeitos}(a2)) \wedge \xi(X))) \\
&= (p \wedge q) \wedge (\forall p.\forall q.(((p \wedge \neg q) \vee (\neg p \wedge q)) \rightarrow (\neg p \wedge q)) \\
&= (p \wedge q) \wedge (\forall p.\forall q.(\neg p \vee q)) \\
&= (p \wedge q) \wedge \forall p.(\neg p) \\
&= (p \wedge q) \wedge (\top \wedge \bot) \\
&= (p \wedge q) \wedge (\bot) \\
&= \bot .
\end{aligned}$$

Definição 3.10. *(Regressão simbólica forte de um conjunto de estados por um conjunto de ações) Seja $\mathbb{A}$ um conjunto de ações e seja $\xi(X)$ a representação proposicional de um conjunto de estados X, a regressão forte de X é dada por:*

$$\mathit{sregr}_{\mathit{forte}}(X) = \bigvee_{a \in \mathbb{A}} \mathit{sregr}^{a}_{\mathit{forte}}(X). \quad (3.16)$$

A Equação 3.7 é a versão enumerativa da Equação 3.16. Enquanto que na primeira equação

é feita a união dos conjuntos obtidos pela regressão forte de cada ação $a \in \mathbb{A}$; na segunda, é feita a disjunção das fórmulas obtidas pela regressão forte de cada $a \in \mathbb{A}$.

3.3 Raciocínio Explícito e Enumerativo sobre Ações não Determinísticas

Nesta seção mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *explícito e enumerativo* contendo ações não determinísticas. Na Seção 3.3.1 definimos dois tipos de modelos explícitos utilizados para representar domínios de planejamento: a *estrutura de Kripke* e os *sistemas de transição de estados rotulados por ações*. Na Seção 3.3.2 mostramos como realizar a operação de *imagem* de um conjunto de estados (o que equivale à operação de regressão no modelo implícito) tanto em uma estrutura de Kripke como em um sistema de transição de estados rotulado por ações. Na Seção 3.3.3 mostramos como realizar a operação de *pré-imagem* de um conjunto de estados (o que equivale à operação de regressão no modelo implícito) tanto em uma estrutura de Kripke como em um sistema de transição de estados rotulado por ações.

3.3.1 Estruturas de Kripke e Sistema de Transição de Estados Rotulados por Ações

Definimos nesta seção dois tipos de modelos explícitos utilizados para representar domínios de planejamento: a *estrutura de Kripke* (que não possui ações nos rótulos das transições entre os estados) e os *sistemas de transição de estados rotulados por ações* (os quais possuem ações rotulando as transições entre estados).

Definição 3.11. *(Estrutura de Kripke) Uma estrutura de Kripke [Kripke, 1963] sobre um conjunto finito de proposições $\mathbb{P} \neq \emptyset$ é uma tupla $M = \langle S, L, T \rangle$ em que:*

- *$S \neq \emptyset$ é um conjunto finito de estados;*
- *$L : \mathcal{S} \mapsto 2^{\mathbb{P}}$ é uma função de interpretação de estados;*
- *$T : \mathcal{S} \mapsto 2^{\mathcal{S}}$ é uma função de transição de estados.*

A Figura 3.2 mostra uma estrutura de Kripke $M = \langle S, L, T \rangle$ sobre o conjunto de proposições $\mathbb{P} = \{p, q\}$ cujo conjunto de estados é $S = \{s_0, s_1, s_2\}$. Note que essa estrutura de Kripke é

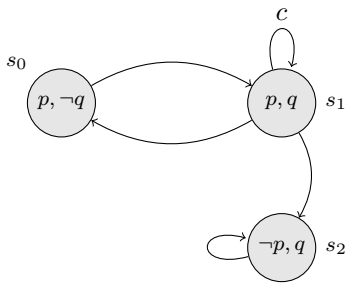


Figura 3.2: Estrutura de Kripke sobre um conjunto de proposições $\mathbb{P} = \{p, q\}$ referente ao sistema de transição de estados rotulado por ações da Figura 1.1(b).

semelhante ao sistema de transição de estados mostrado na Figura 1.1(b), sendo que a estrutura de Kripke não possui as transições rotuladas por ações.

No entanto, em planejamento, é fundamental disntinguir que ação é responśavel por uma dada transição. Por exemplo, na estrutura de Kripke da Figura 3.2 não é possível saber que ação está causando a transição entre os estados s_1 e s_2. Porém, na Figura 1.1(b) duas ações diferentes (a_2 e b) que podem ser aplicadas em s_1 e terem s_2 como estado resultante. Assim, é mais apropriado ter rótulos de ações nas transições entre os estados. Esta estrutura de Kripke que possui rótulos nas transições é chamada de *sistema de transição de estados rotulado por ações*.

Definição 3.12. *(Sistema de Transição de Estados Rotulado por Ações) Um sistema de transição de estados rotulado por ações sobre um conjunto finito de proposições $\mathbb{P} \neq \emptyset$ e um conjunto de ações $\mathbb{A} \neq \emptyset$ é uma tupla $M = \langle S, L, \mathcal{T} \rangle$ em que:*

- $S \neq \emptyset$ *é um conjunto finito de estados;*
- $L : \mathcal{S} \mapsto 2^{\mathbb{P}}$ *é uma função de interpretação de estados;*
- $T : \mathcal{S} \times \mathbb{A} \mapsto 2^{\mathcal{S}}$ *é uma função de transição de estados em que cada transição é rotulada por uma ação $a \in \mathbb{A}$.*

O par $(\mathbb{P}, \mathbb{A})$ é denominado *assinatura* do sistema de transição de estados rotulado por ações. Na Figura 1.1(a) a assinatura é dada pelo par $(\{p, q\}, \{a1, b, c\})$ e na Figura 1.1 assinatura é o par $(\{p, q\}, \{a2, b, c\})$.

3.3.2 Imagem de um Conjunto de Estados

A *imagem* de um conjunto de estados X computa o conjunto de estados que são sucessores de X em um modelo explícito e enumerativo. O conjunto de estados obtidos como resultado da imagem é o mesmo que o obtido com o cálculo da progressão. No entanto, no cálculo da imagem o raciocínio é feito sobre as transições do modelo explícito enquanto que na progressão o raciocínio é feito sobre as ações STRIPS. A Definição 3.13 mostra como obter a imagem de um conjunto de estados X em uma estrutura de Kripke enquanto que a Definição 3.14 mostra como computar a imagem de X em um sistema de transição de estados rotulado por ações.

Definição 3.13. *(Imagem de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$ e seja $X \subseteq S$ um conjunto de estados. A imagem de X, denotada por img(X), é o conjunto de estados alcançáveis a partir de cada estado $s \in X$ em um passo, i.e.,*

$$img(X) = \{s' \in X' \mid s \in X \text{ e } T(s) = X'\}. \tag{3.17}$$

Exemplo 3.13. *(Imagem de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2 e seja $X = \{s_1\}$ um subconjunto de estados de S. A imagem de X é:*

$$img(X) = \{s_0, s_1, s_2\}$$

, uma vez que a partir de s_1 é possível alcançar s_0, s_1 ou s_2 em um passo.

Definição 3.14. *(Imagem de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, T\rangle$ um sistema de transição de estados com assinatura $(\mathbb{P}, \mathbb{A})$ e seja $X \subseteq S$ um conjunto de estados. A imagem de X, denotada por a-img(X), é o conjunto de estados alcançáveis a partir de cada estado $s \in X$ em um passo por alguma ação $a \in \mathbb{A}$, i.e.,*

$$a\text{-}img(X) = \{s' \in Y \mid s \in X \text{ e } \exists a \in \mathbb{A} \text{ tal que } T(s, a) = Y\}. \tag{3.18}$$

Exemplo 3.14. *(Imagem de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, \mathcal{T}\rangle$ um sistema de transição de estados da Figura 1.1(b) e $X = \{s_1\}$ um subconjunto de S. A imagem de X é:*

$$a\text{-}img(X) = \{s_0, s_1, s_2\}$$

, uma vez que a partir de s_1 é possível alcançar os estados s_0, s_1 ou s_2 executando-se as ações a, b ou c.

Também é possível definir a imagem de um conjunto de estados segundo cada ação $a \in \mathbb{A}$, como mostrado pela Equação: 3.19.

$$\textit{a-img}^a(X) = \{s' \in Y \mid s \in X \text{ e } \mathcal{T}(s,a) = Y\}; \tag{3.19}$$

Exemplo 3.15. *(Imagem segundo uma ação) Seja $M = \langle S, L, T\rangle$ o sistema de transição de estados com assinatura $(\mathbb{P}, \mathbb{A})$ da Figura 1.1(b) e seja $X = \{s_1\}$ um subconjunto de S. A imagem de X segundo cada ação $a \in \mathbb{A}$ é:*

$$\textit{a-img}^{a2}(X) = \{s_0, s_2\}; \quad \textit{a-img}^b(X) = \{s_2\}; \quad \textit{a-img}^c(X) = \{s_1\}.$$

A união dos estados computados pela imagem segundo cada ação $a \in \mathbb{A}$ resulta na $\textit{a-img}(X)$, como pode ser visto a seguir:

$$\textit{a-img}(X) = \bigcup_{a \in \mathbb{A}} \textit{a-img}^a(X). \tag{3.20}$$

3.3.3 Pré-imagem de um Conjunto de Estados

A *pré-imagem* de um conjunto de estados X computa o conjunto de estados que antecedem X em um modelo explícito e enumerativo. O conjunto de estados obtidos como resultado da pré-imagem é o mesmo que o obtido com o cálculo da regressão. No entanto, no cálculo da pré-imagem o raciocínio é feito sobre as transições do modelo explícito enquanto que na regressão o raciocínio é feito sobre as ações STRIPS.

Definição 3.15. *(Pré-imagem fraca de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$ e seja $X \subseteq S$ um conjunto de estados. A pré-imagem fraca de X, denotada por $pre_{\textit{fraca}}(X)$, devolve o conjunto de estados a partir dos quais possivelmente alcança-se um estado $x \in X$ em um passo, i.e.,*

$$pre_{\textit{fraca}}(X) = \{s \in S \mid T(s) \cap X \neq \emptyset\}. \tag{3.21}$$

Exemplo 3.16. *(Pré-imagem fraca de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2 e seja $X = \{s_2\}$ um subconjunto de estados de S. A pré-imagem fraca de X é:*

$$\textit{pre}_{\textit{fraca}}(X) = \{s_1, s_2\}$$

, uma vez que a partir desses estados é possível alcançar s_2 em um passo.

Definição 3.16. *(Pré-imagem forte de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$ e seja $X \subseteq S$ um conjunto de estados. A pré-imagem forte de X, denotada por $pre_{forte}(X)$, devolve o conjunto de estados a partir dos quais necessariamente alcança-se estados em X em um passo, i.e.,*

$$pre_{forte}(X) = \{s \in S \mid T(s) \cap X \neq \emptyset\}. \tag{3.22}$$

Exemplo 3.17. *(Pré-imagem forte de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2 e seja $X = \{s_2\}$ um subconjunto de estados de S. A pré-imagem forte de X é:*

$$pre_{forte}(X) = \{s_2\}$$

, uma vez que todas as transições partindo de s_2 levam apenas a estados pertencentes ao conjunto X. Considerando o estado s_1, por exemplo, apesar de haver uma transição levando para $s_2 \in X$, há outra transição que leva a $s_0 \notin X$. Assim, $s_1 \notin pre_{forte}(X)$.

Definição 3.17. *(Pré-imagem fraca de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, \mathcal{T}\rangle$ um sistema de transição de estados com assinatura $(\mathbb{P}, \mathbb{A})$ e seja $X \subseteq S$ um conjunto de estados. A pré-imagem fraca de X, denotada por $pre_{fraca}(X)$, computa o conjunto de estados $Y \subseteq S$ tal que, para cada estado $s \in Y$, há uma ação $a \in \mathbb{A}$ cuja execução em s leva a um estado em X.*

$$a\text{-}pre_{fraca}(X) = \{s \in S \mid \exists a \in \mathbb{A} \text{ e } \mathcal{T}(s, a) \cap X \neq \emptyset\}. \tag{3.23}$$

Exemplo 3.18. *(Pré-imagem fraca de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, \mathcal{T}\rangle$ o sistema de transição de estados da Figura 1.1(b) e $X = \{s_2\}$ um subconjunto de S. A pré-imagem fraca de X é:*

$$a\text{-}pre_{fraca}(X) = \{s_1, s_2\}$$

, uma vez que a partir de s_1 é possível alcançar s_2 (seguindo um dos efeitos da ação $a2$ ou a ação b) e a partir de s_2 é possível alcançar s_2 (seguindo a ação b).

Definição 3.18. *(Pré-imagem forte de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, \mathcal{T} \rangle$ um sistema de transição de estados com assinatura $(\mathbb{P}, \mathbb{A})$ e seja $X \subseteq S$ um conjunto de estados. A pré-imagem forte de X, denotada por $pre_{forte}(X)$, computa o conjunto de estados $Y \subseteq S$ tal que, para cada estado $s \in Y$, há uma ação $a \in \mathbb{A}$ cuja execução em s leva a um estado em X.*

$$\textit{a-pre}_{\textit{forte}}(X) = \{s \in S \mid \exists a \in \mathbb{A} \textit{ e } \emptyset \neq \mathcal{T}(s, a) \subseteq X\}. \tag{3.24}$$

Exemplo 3.19. *(Pré-imagem forte de um conjunto de estados em um sistema de transição rotulado por ações) Seja $M = \langle S, L, \mathcal{T} \rangle$ um sistema de transição de estados da Figura 1.1(b) e $X = \{s_2\}$ um subconjunto de S. A pré-imagem forte de X é:*

$$\textit{a-pre}_{\textit{forte}}(X) = \{s_1, s_2\}$$

, uma vez que todas as transições causadas pela execução da ação b em s_1 (e s_2) levam a s_2. Note que é possível considerar apenas as transições que partem de s_1 e são rotuladas pela ação b. Isto não é possível em uma estrutura de Kripke uma vez que as transições não são rotuladas pelas ações.

Também é possível definir a pré-imagem fraca e forte de um conjunto de estados segundo cada ação $a \in \mathbb{A}$, como mostrado nas Equações 3.25 e 3.26.

$$\textit{a-pre}^{a}_{\textit{fraca}}(X) = \{s \in S \mid \mathcal{T}(s, a) \cap X \neq \emptyset\}; \tag{3.25}$$

$$\text{a-pre}^{a}_{\textit{forte}}(X) = \{s \in S \mid \emptyset \neq \mathcal{T}(s, a) \subseteq X\}. \tag{3.26}$$

Exemplo 3.20. *(Pré-imagem fraca e forte segundo uma ação) Seja $M = \langle S, L, T \rangle$ o sistema de transição de estados com assinatura $(\mathbb{P}, \mathbb{A})$ da Figura 1.1(b) e seja $X = \{s_2\}$ um subconjunto de S. A pré-imagem fraca de X segundo cada ação $a \in \mathbb{A}$ é:*

$$\textit{a-pre}^{a2}_{\textit{fraca}}(X) = \{s_1\}; \quad \textit{a-pre}^{b}_{\textit{fraca}}(X) = \{s_1, s_2\}; \quad \textit{a-pre}^{c}_{\textit{fraca}}(X) = \emptyset.$$

A pré-imagem forte de X de acordo com cada ação $a \in \mathbb{A}$ é:

$$\textit{a-pre}^{a2}_{\textit{forte}}(X) = \emptyset; \quad \textit{a-pre}^{b}_{\textit{forte}}(X) = \{s_1, s_2\}; \quad \textit{a-pre}^{c}_{\textit{forte}}(X) = \emptyset.$$

A união dos estados computados pela pré-imagem (fraca e forte), segundo cada ação $a \in \mathbb{A}$, resulta em *a-pre*$_{fraca}(X)$ e *a-pre*$_{forte}(X)$, repectivamente:

$$\textit{a-pre}_{fraca}(X) = \bigcup_{a \in \mathbb{A}} \textit{a-pre}^{a}_{fraca}(X); \tag{3.27}$$

$$\textit{a-pre}_{forte}(X) = \bigcup_{a \in \mathbb{A}} \textit{a-pre}^{a}_{forte}(X). \tag{3.28}$$

Seja X um conjunto de estados, o Teorema 3.1 prova que o conjunto de estados obtidos pela *regressão* de X é o mesmo obtido pela *pré-imagem* de X.

Teorema 3.1. *Se $a \in \mathbb{A}$ é uma ação e $X \subseteq S$ é um conjunto de estados, então $regr^{a}_{fraca}(X) = pre^{a}_{fraca}(X)$.*

Demonstração. Parte (I): Seja $s \in regr^{a}_{fraca}(X)$ um estado e suponha que $s \notin pre^{a}_{fraca}(X)$. Assim, de acordo com a Equação 3.5, temos que $progr^{a}(s) \cap X \neq \emptyset$; e, de acordo com a Equação 3.21, temos que $\mathcal{T}(s,a) \cap X = \emptyset$. Porém, isto é uma contradição (se $\mathcal{T}(s,a) \cap X = \emptyset$, então o estado s não tem um sucessor no conjunto X que pode ser alcançado por uma transição rotulada pela ação a e, consequentemente, a progressão de s pela ação a não leva a um estado em X). Assim, temos que $s \in regr^{a}_{fraca}(X) \rightarrow s \in pre^{a}_{fraca}(X)$, i.e., $regr^{a}_{fraca}(X) \subseteq pre^{a}_{fraca}(X)$. Part (II): Analogamente, podemos mostrar que $s \in pre^{a}_{fraca}(X) \rightarrow s \in regr^{a}_{fraca}(X)$, i.e., $pre^{a}_{fraca}(X) \subseteq regr^{a}_{fraca}(X)$. Então, conclui-se que $regr^{a}_{fraca}(X) = pre^{a}_{fraca}(X)$. □

3.4 Raciocínio Explícito e Simbólico sobre Ações não Determinísticas

Nesta seção mostramos como realizar o raciocínio progressivo e regressivo sobre um modelo *explícito e simbólico*. Na Seção 3.4.1 definimos como representar transições de um modelo explícito (estrutura de Kripke ou sistema de transição de estados rotulado por ações) como fórmulas da lógica proposicional. Na Seção 3.4.2 definimos a imagem simbólica de um conjunto de estados e na Seção 3.4.3 definimos a pré-imagem simbólica de um conjunto de estados.

3.4.1 Representação Proposicional para Transições

Em uma estrutura de Kripke, uma transição é um par $t = (s, s')$. Para representar os estados antes e depois da transição é necessário gerar uma cópia $\mathbb{P}'$ do conjunto de proposições $\mathbb{P}$, tal que cada $p' \in \mathbb{P}'$ tem um $p \in \mathbb{P}$ correspondente. Na Figura 3.3, por exemplo, $\xi(s_0) = p \wedge \neg q$ é a fórmula que representa o estado corrente na transição e $\xi(s'_1) = p' \wedge q'$ é a fórmula que representa o próximo estado na transição.

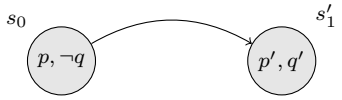


Figura 3.3: Representação dos estados antes e depois de uma transição em uma estrutura de Kripke.

Definição 3.19. *(Representação proposicional para transição e relação de transição em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$. A codificação proposicional para uma transição $(s, s') \in T$, denotada por $\xi(t)$ é dada pela fórmula*

$$\xi(t) = \xi(s) \wedge \xi(s') \tag{3.29}$$

e a codificação proposicional para a relação de transição T é a fórmula $\xi(T)$

$$\xi(T) = \bigvee_{t \in T} \xi(t). \tag{3.30}$$

Exemplo 3.21. *(Representação proposicional para a relação de transição em uma estrutura de Kripke) A relação de transição $T = \{(s_0, s_1), (s_1, s_1), (s_1, s_0), (s_1, s_2), (s_2, s_2)\}$ da estrutura de Kripke da Figura 3.2 pode ser dada pela fórmula:*

$$\xi(T) = (p \wedge \neg q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q') \vee (p \wedge q \wedge \neg p' \wedge q') \vee (\neg p \wedge q \wedge p' \wedge \neg q').$$

Já em um sistema de transição de estados rotulado por ações com assinatura $(\mathbb{P}, \mathbb{A})$, uma transição t é dada por (s, a, s'). Uma abordagem que realiza o cálculo simbólico da imagem e pré-imagem de um conjunto de estados [Cimatti et al., 2003] considera que as ações que rotulam as transições são proposições. Assim, cada $a \in \mathbb{A}$ tem uma proposição $\alpha \in \mathcal{A}$ correspondente. O conjunto $\mathcal{A}$ é chamado conjunto das *variáveis proposicionais de ações*. Intuitivamente, uma

variável proposicional de ação é verdadeira se e somente se a ação correspondente está sendo executada.

Definição 3.20. *(Representação proposicional para transição e relação de transição em um sistema de transição de estados rotulado por ações) Seja $M = \langle S, L, \mathcal{T} \rangle$ um sistema de transição de estados rotulado por ações com assinatura $(\mathbb{P}, \mathbb{A})$. A codificação proposicional para uma transição $(s, a, s') \in \mathcal{T}$, denotada por $\xi(t)$ é a seguinte fórmula:*

$$\xi(t) = \xi(s) \wedge \alpha \wedge \xi(s') \tag{3.31}$$

e a codificação proposicional para a relação de transição T é a fórmula $\xi(T)$

$$\xi(T) = \bigvee_{t \in T} \xi(t). \tag{3.32}$$

Exemplo 3.22. *(Representação proposicional para a relação de transição em um sistema de transição rotulado por ações) A relação de transição $T = \{(s_0, c, s_1), (s_1, c, s_1), (s_1, a_2, s_0), (s_1, a_2, s_2), (s_1, b, s_2), (s_2, b, s_2)\}$ do sistema da Figure 1.1(b) pode ser dada pela seguinte fórmula proposicional:*

$$\begin{aligned}\xi(T) = (p \wedge \neg q \wedge c \wedge p' \wedge q') \ \vee \ (p \wedge q \wedge c \wedge p' \wedge q') \ \vee \ (p \wedge q \wedge a_2 \wedge p' \wedge \neg q') \vee \\ (p \wedge q \wedge a_2 \wedge \neg p' \wedge q') \ \vee \ (p \wedge q \wedge b \wedge \neg p' \wedge q') \ \vee \ (\neg p \wedge q \wedge b \wedge p' \wedge \neg q').\end{aligned}$$

3.4.2 Imagem Simbólica de um Conjunto de Estados

Dada a representação proposicional da relação de transição em uma estrutura de Kripke (ou em um sistema de transição de estados rotulados por ações) e a representação proposicional de um conjunto de estados X, nesta seção mostramos como calcular a imagem de X de maneira simbólica.

Definição 3.21. *(Imagem simbólica de um conjunto de estados em uma estrutura de Kripke) Sejam $M = \langle S, L, T \rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$ e $\xi(X)$ a representação proposicional de um conjunto $X \subseteq S$. A imagem simbólica de X, denotada por $simg(X)$, é dada por:*

$$simg(X) = \exists \mathbb{P}.(\xi(T) \wedge \xi(X)). \tag{3.33}$$

Exemplo 3.23. *(Imagem simbólica de um conjunto de estados em uma estrutura de Kripke) Sejam $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2; $X = \{s_1\} \subseteq S$ um conjunto de estados, cuja representação proposicional é dada pela fórmula $\xi(X) = p \wedge q$ e; $\xi(T)$ a representação proposicional da relação de transição (Exemplo 3.21). A imagem simbólica de X é dada por:*

$$\begin{aligned}
\mathit{simg}(X) &= \exists\mathbb{P}.(\xi(T) \wedge \xi(X)) \quad (\textit{pela Equação } 3.34)\\
&= \exists\{p,q\}.(((p \wedge \neg q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q')\\
&\quad \vee (p \wedge q \wedge \neg p' \wedge q') \vee (\neg p \wedge q \wedge p' \wedge \neg q')) \wedge (p \wedge q))\\
&= \exists p.\exists q.((p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q') \vee (p \wedge q \wedge \neg p' \wedge q'))\\
&= (p' \wedge q') \vee (p' \wedge \neg q') \vee (\neg p' \wedge q')) \ (\textit{representando } \{s_0, s_1, s_2\})
\end{aligned}$$

Definição 3.22. *(Imagem simbólica de um conjunto de estados em um sistema de transição de estados rotulado por ações) Sejam $M = \langle S, L, T\rangle$ um sistema de transições rotulado por ações com assinatura $(\mathbb{P}, \mathbb{A})$; $\xi(X)$ a representação proposicional de um conjunto $X \subseteq S$; $\xi(T)$ a representação proposicional da relação de transição e; $\mathcal{A}$ o conjunto de variáveis proposicionais referentes às ações em $\mathbb{A}$. A imagem simbólica de X, denotada por a-simg(X), é dada por:*

$$\textit{a-simg}(X) = \exists\mathcal{A}.\exists\mathbb{P}.(\xi(\mathcal{T}) \wedge \xi(X)). \tag{3.34}$$

Exemplo 3.24. *(Imagem simbólica de um conjunto de estados em um sistema de transição de estados rotulado por ações) Sejam $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2; $X = \{s_1\} \subseteq S$ um conjunto de estados, cuja representação proposicional é dada pela fórmula $\xi(X) = p \wedge q$; $\xi(T)$ a representação proposicional da relação de transição (Exemplo 3.21) e; $\mathcal{A}$ o conjunto de variáveis proposicionais referentes às ações em $\mathbb{A}$. A imagem simbólica de X é dada por:*

$$\begin{aligned}
\textit{a-simg}(X) &= \exists\mathcal{A}.\exists\mathbb{P}.(\xi(T) \wedge \xi(X)) \quad (\textit{pela Equação } 3.34)\\
&= \exists\{p,q\}.(((p \wedge \neg q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q')\\
&\quad \vee (p \wedge q \wedge \neg p' \wedge q') \vee (\neg p \wedge q \wedge p' \wedge \neg q')) \wedge (p \wedge q))\\
&= \exists p.\exists q.((p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q') \vee (p \wedge q \wedge \neg p' \wedge q'))\\
&= (p' \wedge q') \vee (p' \wedge \neg q') \vee (\neg p' \wedge q')) \ (\textit{representando } \{s_0, s_1, s_2\})
\end{aligned}$$

3.4.3 Pré-Imagem Simbólica de um Conjunto de Estados

Nesta seção definimos como computar as operações de pré-imagem (fraca e forte) de um conjunto de estados de forma simbólica, tanto para a representação proposicional da estrutura de Kripke como para a representação proposicional de um sistema de transição de estados rotulados por ações. A Definição 3.23 mostra como computar a pré-imagem simbólica fraca de um conjunto de estados em uma estrutura de Kripke e a Definição 3.24 mostra como calcular a pré-imagem simbólica forte.

Definição 3.23. *(Pré-imagem simbólica fraca de um conjunto de estados em uma estrutura de Kripke) Sejam $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$; $\xi(X)$ a representação proposicional de um conjunto $X \subseteq S$ e; $\xi(T)$ a representação proposicional da relação de transição. A pré-imagem fraca X é calculada de forma simbólica como a seguir [Huth and Ryan, 2004]:*

$$spre_{fraca}(X) = \exists \mathbb{P}'.(\xi(T) \wedge \xi(X')). \tag{3.35}$$

Exemplo 3.25. *(Pré-imagem simbólica fraca de um conjunto de estados em uma estrutura de Kripke) Sejam $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2; $X = \{s_2\} \subseteq S$ um conjunto de estados, cuja representação usando "variáveis linha" é dada pela fórmula $\xi(X') = \neg p' \wedge q'$ e; $\xi(T)$ a representação proposicional da relação de transição (Exemplo 3.21). A pré-imagem simbólica fraca de X é dada por:*

$$\begin{aligned}
spre_{fraca}(X) &= \exists \mathbb{P}'.(\xi(T) \wedge \xi(X)') \quad (\textit{pela Equação } 3.35) \\
&= \exists\{p', q'\}.(((p \wedge \neg q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q') \\
&\quad \vee (p \wedge q \wedge \neg p' \wedge q') \vee (\neg p \wedge q \wedge p' \wedge \neg q')) \wedge (\neg p' \wedge q')) \\
&= \exists p'.\exists q'.(q \wedge \neg p' \wedge q') \\
&= \exists p'.((q \wedge \neg p' \wedge \bot) \vee (q \wedge \neg p' \wedge \top)) \\
&= \exists p'.(q \wedge \neg p') \\
&= q \quad (\textit{representando o conjunto de estados } \{s_1, s_2\}).
\end{aligned}$$

Definição 3.24. *(Pré-imagem simbólica forte de um conjunto de estados em uma estrutura de Kripke) Seja $M = \langle S, L, T\rangle$ uma estrutura de Kripke sobre um conjunto de proposições $\mathbb{P}$, a pré-imagem simbólica forte de um conjunto de estados $X \subseteq S$ é computada por [Cimatti et al., 2003]:*

$$spre_{forte}(X) = \forall \mathbb{P}'.(\xi(T) \rightarrow \xi(X')) \wedge \exists \mathbb{P}'.\xi(T). \tag{3.36}$$

Exemplo 3.26. *(Pré-imagem forte simbólica de um conjunto de estados em uma estrutura de Kripke) Sejam $M = \langle S, L, T\rangle$ a estrutura de Kripke da Figura 3.2, $\xi(X')$ a representação proposicional do conjunto de estados $X = \{s_2\} \subseteq S$ usando "variáveis linha" e; $\xi(T)$ a representação proposicional da relação de transição (Exemplo 3.21). A pré-imagem simbólica forte de X é dada por:*

$$\begin{aligned}
\mathit{spre}_{\mathit{forte}}(X) &= \forall\mathbb{P}'.(\xi(T) \rightarrow \xi(X')) \wedge \exists\mathbb{P}'.\xi(T) \quad (\textit{pela Equação 3.36})\\
&= \forall\mathbb{P}'.((p \wedge \neg q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge q') \vee (p \wedge q \wedge p' \wedge \neg q')\\
&\quad \vee (p \wedge q \wedge \neg p' \wedge q') \vee (\neg p \wedge q \wedge \neg p' \wedge q') \rightarrow (\neg p' \wedge q')) \wedge \exists\{p', q'\}.\xi(T)\\
&= \forall\mathbb{P}'((\neg p \vee \neg q \vee \neg p') \wedge (\neg p \vee \neg p' \vee \neg q')) \wedge ((p \wedge \neg q) \vee (p \wedge q) \vee (\neg p \wedge q))\\
&= (\neg p \vee \neg q) \wedge \neg p \wedge ((p \wedge \neg q) \vee (p \wedge q) \vee (\neg p \wedge q))\\
&= \neg p \wedge q \quad (\textit{representando o estado } s_2).
\end{aligned}$$

A pré-imagem de um sistema de transição rotulado por ações também pode ser calculada de forma simbólica. A Definição 3.25 mostra como computar a pré-imagem fraca de um conjunto de estados quando as transições são rotuladas por ações e a Definição 3.26 mostra como calcular a pré-imagem forte. Em ambas as definições, é necessário realizar a eliminação das variáveis proposicionais de ações usando QBF ($\exists\mathcal{A}$).

Definição 3.25. *(Pré-imagem simbólica fraca de um conjunto de estados em um sistema de transição rotulado por ações) Sejam $M = \langle S, L, \mathcal{T}\rangle$ um sistema de transição rotulado por ações com assinatura ($\mathbb{P}, \mathbb{A}$); $\xi(X')$ a representação proposicional de $X \subseteq S$ usando "variáveis linha"; $\xi(\mathcal{T})$ a representação proposicional da relação de transição $\mathcal{T}$ e; $\mathcal{A}$ o conjunto de variáveis propoicionais de ações. A pré-imagem simbólica fraca de X é:*

$$\mathit{a\text{-}spre}_{\mathit{fraca}}(X) = \exists\mathcal{A}.\exists\mathbb{P}'.(\xi(\mathcal{T}) \wedge \xi(X')). \tag{3.37}$$

Exemplo 3.27. *(Pré-imagem simbólica fraca de um conjunto de estados em um sistema de transição rotulado por ações) Sejam $M = \langle S, L, \mathcal{T}\rangle$ o sistema de transição de estados da Figura 1.1(b); $\xi(X') = \neg p' \wedge q'$ a representação proposicional com "variáveis linha" do conjunto $X = \{s_2\} \subseteq S$ e; $\xi(\mathcal{T})$ a representação proposicional da relação de transição (Exem-*

plo 3.22). A pré-imagem simbólica fraca de X é dada por:

$$\begin{aligned}
\textit{a-spre}_{\textit{fraca}}(X) &= \exists\mathcal{A}.\exists\mathbb{P}'.(\xi(T) \wedge \xi(X')) \quad (\textit{pela Equação 3.37}) \\
&= \exists\mathcal{A}.\exists\mathbb{P}'.(((p \wedge \neg q \wedge c \wedge p' \wedge q') \vee (p \wedge q \wedge c \wedge p' \wedge q') \vee (p \wedge q \wedge \\
&\quad a_2 \wedge p' \wedge \neg q') \vee (p \wedge q \wedge a_2 \wedge \neg p' \wedge q') \vee (p \wedge q \wedge b \wedge \neg p' \wedge q') \\
&\quad \vee (\neg p \wedge q \wedge b \wedge \neg p' \wedge q')) \wedge (\neg p' \wedge q')) \\
&= \exists a2.\exists b.\exists c.\exists p'.\exists q'.((p \wedge q \wedge a2 \wedge \neg p' \wedge q') \vee (q \wedge b \wedge \neg p' \wedge q')) \\
&= (p \wedge q) \vee q \\
&= q \quad \textit{(representando o conjunto de estados} \{s_1, s_2\}).
\end{aligned}$$

Definição 3.26. *(Pré-imagem simbólica forte de um conjunto de estados em um sistema de transição rotulado por ações) Sejam $M = \langle S, L, \mathcal{T}\rangle$ um sistema de transição rotulado por ações com assinatura $(\mathbb{P}, \mathbb{A})$; $\xi(X')$ a representação proposicional com "variáveis linhas" de um conjunto $X \subseteq S$; $\xi(\mathcal{T})$ a representação proposicional da relação de transição $\mathcal{T}$ e; $\mathcal{A}$ o conjunto de variáveis proposicionais de ações. A pré-imagem simbólica forte de X é:*

$$\textit{a-spre}_{\textit{forte}}(X) = \exists\mathcal{A}.(\forall\mathbb{P}'.(\xi(\mathcal{T}) \rightarrow \xi(X')) \wedge \exists\mathbb{P}'.\xi(\mathcal{T})). \tag{3.38}$$

Exemplo 3.28. *(Pré-imagem simbólica forte de um conjunto de estados em um sistema de transição rotulado por ações) Sejam $M = \langle S, L, \mathcal{T}\rangle$ o sistema de transições de estados da Figura 1.1(b); $\xi(X')$ a representação proposicional usando variáveis linhas do conjunto de estados $X = \{s_2\} \subseteq S$; e $\xi(\mathcal{T})$ a representação proposicional da relação de transição (Exemplo 3.22). A pré-imagem simbólica forte de X é dada por:*

$$\begin{aligned}
\textit{a-spre}_{\textit{forte}}(X) &= \exists\mathcal{A}.(\forall\mathbb{P}'.(\xi(\mathcal{T}) \rightarrow \xi(X')) \wedge \exists\mathbb{P}'.\xi(\mathcal{T})) \\
&= \exists\mathcal{A}.(\forall\mathbb{P}'.(((p \wedge \neg q \wedge c \wedge p' \wedge q') \vee (p \wedge q \wedge c \wedge p' \wedge q') \vee (p \wedge q \wedge \\
&\quad a2 \wedge p' \wedge \neg q') \vee (p \wedge q \wedge a2 \wedge \neg p' \wedge q') \vee (p \wedge q \wedge b \wedge \neg p' \wedge q') \\
&\quad \vee(\neg p \wedge q \wedge b \wedge \neg p' \wedge q')) \rightarrow (\neg p' \wedge q')) \wedge \exists\mathbb{P}'.\xi(\mathcal{T}) \\
&= \exists\mathcal{A}.((\forall\mathbb{P}'.((\neg p \vee \neg q \vee \neg a2 \vee \neg p' \vee q') \wedge (\neg p \vee \neg c \vee \neg p' \vee \neg q')) \\
&\quad \wedge((p \wedge q \wedge a2) \vee (q \wedge b) \vee (p \wedge c))) \\
&= \exists\mathcal{A}.((\neg p \vee \neg c) \wedge (\neg p \vee \neg q \vee \neg a2) \wedge ((p \wedge q \wedge a2) \vee (q \wedge b) \vee (p \wedge c))) \\
&= \exists\mathcal{A}.((q \wedge \neg a2 \wedge b \wedge \neg c) \vee (\neg p \wedge q \wedge b)) \\
&= q \vee (\neg p \wedge q) \\
&= q \quad \textit{(represenando o conjunto de estados} \{s_1, s_2\}).
\end{aligned}$$

Capítulo 4

Problemas de Planejamento sem Solução

Neste capítulo mostramos o que pode ser feito quando um problema de planejamento não possui solução. Na Seção 4.1 é feita uma discussão informal sobre as possíveis modificações de um problema sem solução a fim de torná-lo solucionável, i.e., a modificação do estado inicial, da meta e da especificação das ações. Na Seção 4.2 mostramos a relação existente entre modificar um problema de planejamento sem solução para torná-lo solucionável e modificar uma base de conhecimento para incorporar um novo fato. Na Seção 4.3 definimos métricas para a modificação de problemas de planejamento sem solução. Na Seção 4.4 formalizamos a modificação de um problema de planejamento com o uso da Lógica Dinâmica de Atribuições Proposicionais (DL-PA) e na Seção 4.5 formalizamos este problema com o arcabouço de verificação de modelos.

4.1 Possíveis Modificações para um Problema de Planejamento

Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ (Definição 1.9) é possível modificá-lo para que ele se torne solucionável. As possíveis modificações para um problema de planejamento são: *modificação do estado inicial* s_0; *modificação da meta de planejamaneto* φ e; *modificação do conjunto de ações* $\mathbb{A}$.

Na modificação do estado inicial s_0, um estado s'_0 é escolhido para ser o novo estado ini-

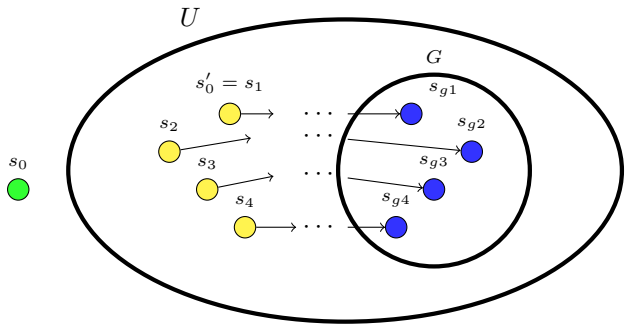


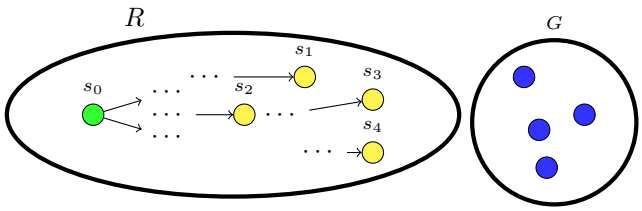
Figura 4.1: Modificação do estado inicial para o problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ em que $s_0 \notin U$. Qualquer estado $s \in U$ é um candidato a ser o novo estado inicial do problema.

cial do problema Π. Tal estado deve finalmente alcançar um estado satisfazendo a meta φ, i.e., s'_0 deve alcançar algum estado do conjunto de estados meta G. Para um problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o estado inicial $s_0 \notin U$ onde U é o conjunto dos estados que finalmente alcançam a meta φ (Definição 1.4). Desta forma, qualquer estado $s \in U$ é um candidato para ser o novo estado inicial do problema. Por exemplo, na Figura 4.1, o estado $s_1 \in U$ é uma possível modificação do estado inicial, uma vez que o problema $\Pi' = \langle \mathbb{A}, s_1, \varphi \rangle$ é um problema solucionável.

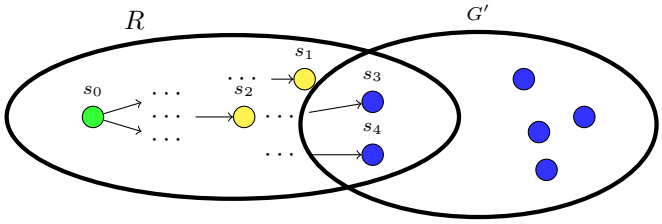
Na modificação da meta de planejamento φ, a meta é modificada para que o problema sem solução torne-se solucionável. A nova meta φ' deve ser satisfeita em algum estado $s \in R$ tal que R é o conjunto dos estados alcançáveis a partir do estado inicial (Definição 1.4), i.e., o conjunto R deve ter alguma intersecção com o novo conjunto dos estados meta G'. A Figura 6.1 ilustra o processo de modificação da meta de planejamento para um problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$. Na Figura 6.1(a), o problema $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ não possui solução, pois não é possível, a partir do estado inicial s_0, alcançar um estado que satisfaça a meta φ, i.e., $R \cap G = \emptyset$. Assim, na Figura 6.1(b), a meta φ é modificada, obtendo uma nova meta φ' de forma que algum estado s alcançável a partir de s_0 a satisfaça, i.e., $s \in R$ e $s \models \varphi'$. Desta forma, o problema com a meta atualizada $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$ é solucionável uma vez que $R \cap G' \neq \emptyset$.

É importante observar que as atualizações apresentadas até então, tanto a modificação do estado inicial como a modificação da meta de planejamento, são *atualizações locais*, isto é, tais atualizações somente afetam a definição do problema que está sendo atualizado.

Na modificação do conjunto de ações $\mathbb{A}$, dado que um problema de planejamento $\Pi =$



(a) Problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$.

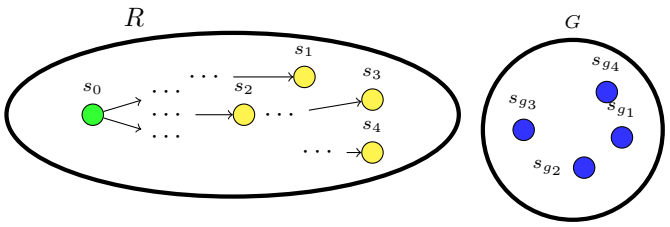


(b) Problema solucionável $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$.

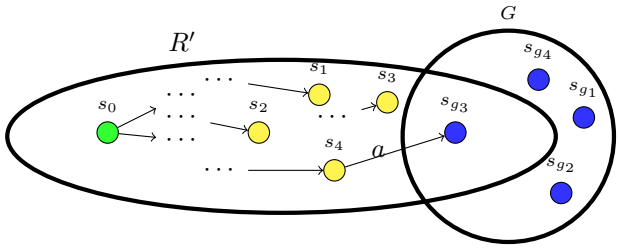
Figura 4.2: Modificação da meta de planejamento para o problema Π. Em (a) Π não é solucionável pois $R \cap G = \emptyset$. Em (b), a meta é modificada de forma que $R \cap G' \neq \emptyset$ obtendo o problema solucionável Π'.

$\langle \mathbb{A}, s_0, \varphi \rangle$ não possui solução, alguma ação $a \in \mathbb{A}$ tem suas precondições ou efeitos modificados para que o problema torne-se solucionável. Dado um problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ (i.e., um problema em que $R \cap G = \emptyset$ e $s_0 \notin U$), uma ação $a \in \mathbb{A}$ é modificada de forma que o problema $\Pi' = \langle \mathbb{A}', s_0, \varphi \rangle$ é solucionável. A modificação de ações altera os conjuntos R e U, obtendo R' e U' tal que $R' \cap G \neq \emptyset$ e $s_0 \in U'$. A Figura 4.3 ilustra o processo de modificação de uma ação de planejamento para um problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$. Na Figura 6.1(a), o problema $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ não possui solução,i.e., $R \cap G = \emptyset$. Assim, na Figura 6.1(b), a ação $a \in \mathbb{A}$ é modificada para ser aplicável ao estado s_4 e levar ao estado $s_g 1$, obtendo um novo conjunto de ações $\mathbb{A}'$ a partir do qual é induzido um novo conjunto R' (e também um novo conjunto U'). Desta forma, o problema com ação modificada $\Pi' = \langle \mathbb{A}', s_0, \varphi' \rangle$ é solucionável uma vez que $R' \cap G \neq \emptyset$.

No entanto, diferente da modificação do estado inicial e da meta, a modificação de ações é uma *modificação global*. Isto significa que ao modificar uma ação $a \in \mathbb{A}$ para que o problema Π passe a ter uma solução, outros problemas definidos para o mesmo conjunto de ações $\mathbb{A}$ também são modificados. Esse efeito global da modificação das ações para tornar um problema solucionável pode interferir na solubilidade de outros problemas que estão definidos para o mesmo



(a) Problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$.



(b) Problema solucionável $\Pi' = \langle \mathbb{A}', s_0, \varphi \rangle$.

Figura 4.3: Modificação de ações para o problema Π. Em (a) Π não é solucionável pois $R \cap G = \emptyset$. Em (b), a ação a é modificada obtendo um novo conjunto $\mathbb{A}'$ de forma que o novo conjunto R' induzido por $\mathbb{A}'$ é tal que $R' \cap G \neq \emptyset$.

conjunto de ações. Assim, dado um conjunto de problemas $\Upsilon = \{\Pi_1, \Pi_2, \cdots \Pi_n\}$ definidos segundo o mesmo conjunto de ações $\mathbb{A}$, a modificação de $\mathbb{A}$ para que Π_i $(1 \leq i \leq n)$ torne-se solucionável pode:

- (i) não alterar a solubilidade de Π_j, isto é, se Π_j era solucionável antes da modificação então ele continua sendo solucionável após a modificação (ou ainda, se Π_j não era solucionável antes da modificação, então ele continua não sendo soluçionável após a modificação);
- (ii) tornar Π_j, que era solucionável antes da modificação, não solucionável;
- (iii) tornar Π_j, que não tinha solução antes da modificação, solucionável.

O conjunto de problemas Υ é denominado *conjunto-teste* para o conjunto de ações $\mathbb{A}$. Do ponto de vista do projetista que está elaborando um novo domínio de planejamento, Υ pode conter problemas que são solucionáveis e problemas sem solução mas que o projetista acredita que devam ser solucionáveis.

4.2 Modificando uma Base de Conhecimento

A tarefa de *modificar um problema de planejamento* sem solução para que ele torne-se solucionável é um problema de *modificação de uma base de conhecimento*, o qual pode ocorrer pelo processo de revisão [Alchourron et al., 1985] ou de atualização [Katsuno and Mendelzon, 1991a]. Nesta seção, mostraremos: a diferença entre revisar e atualizar uma base de conhecimento; que o problema de modificação do estado inicial é um problema de atualização e; que o problema de modificação da meta de planejamento é um problema de revisão [Herzig et al., 2014].

Considere uma *base de conhecimento* representada por uma teoria ψ de uma lógica proposicional, para modificar ψ de forma a incorporar o fato μ deve-s considerar as duas situações. A primeira, é que o fato μ é *consistente* com a base ψ, então ele é adicionado a ψ. A segunda, é que há inconsistências entre μ e ψ, então alguns fatos de ψ devem ser abandonados para que μ possa ser incorporado. Neste processo de modificação, seja por meio de revisão ou de atualização, deve-se obdecer ao *princípio da mudança minimal*, i.e., a base de conhecimento deve incorporar o novo fato mas para isto deve ser sofrer o mínimo de alterações possíveis. Para obter o conjunto das modificações minimais é necessário elaborar uma forma de ordenar as possíveis modificações.

A tarefa de modificar um problema de planejamento sem solução é um problema de modificação de uma base de conhecimento (que pode ser o estado inicial, o conjunto de estados meta ou o conjunto de ações) com relação a um subconjunto dos estados do domínio de planejamento [Herzig et al., 2014].

Por exemplo, na tarefa de modificação de um problema de planejamento é fundamental ter uma forma de medir o quão distante um estado está de outro. A *distância de Hamming* é utilizada para quantificar a distância entre dois estados do modelo explícito representando o domínio de planejamento.

Definição 4.1. *(Distância de Hamming entre estados) Seja $M = \langle S, L, T\rangle$ um sistema de transição de estados, a distância de Hamming entre dois estados $s_1 \in S$ e $s_2 \in S$ é dada por:*

$$h(s_1, s_2) = card(L(s_1) \triangle L(s_2))$$

em que $card(X)$ é a cardinalidade do conjunto X.

Exemplo 4.1. *(Distância de Hamming entre s_1 e s_7) Sejam s_0 o estado da Figura 1.1 tal que*

$L(s_0) = \{$*loc-robô-sala-0*, *loc-chave-sala-0*$\}$ *e* s_7 *o estado tal que* $L(s_7) = \{$*loc-robô-sala-0*, *loc-chave-sala-1*$\}$. *Então, a distância de Hamming entre* s_1 *e* s_7 *é:*

$$h(s_1, s_7) = card(\{\textit{loc-chave-sala-0}\} \cup \{\textit{loc-chave-sala-1}\}) = 2.$$

A atualização de uma base de conhecimento ψ por um fato μ é denotada por $\psi \diamond \mu$, sendo fundamentada nos postulados KM propostos por [Katsuno and Mendelzon, 1991a]. O resultado da atualização $\psi \diamond \mu$ é o conjunto de modelos do novo fato μ que estão *mais próximos* aos modelos da base ψ. Em princípio, qualquer operador que obedeça aos postulados KM pode ser utilizado como operador de atualização. Neste trabalho, consideramos o operador de atualização de Forbus que realiza a atualização de um estado em relação a um conjunto de estados, tendo como base a minimização da distância de Hamming entre estados [Forbus, 1988].

Definição 4.2. *(Atualização de Forbus de um estado) Sejam* $M = \langle S, L, T\rangle$ *um sistema de transição de estados e* s_0 *um estado pertencente a* S. *A atualização de Forbus de* s_0 *pelo conjunto de estados* S *é o conjunto de estados* $s \in S$ *que é mais próximo possível de* s_0 *com relação a distância de Hamming, i.e.,*

$$s_0 \diamond^{\textit{forbus}} S = \{s \in S : \textit{não há } s' \in S \textit{ tal que } h(s_0, s') \leq h(s_0, s)\}.$$

Já a *revisão* de uma base de conhecimento ψ por um fato μ é denotada por $\psi * \mu$, sendo fundamentada nos postulados propostos por [Alchourron et al., 1985]. O resultado da revisão $\psi * \mu$ é o conjunto de modelos do novo fato μ que estão *mais próximos* aos modelos da base ψ. Neste trabalho, consideramos o operador de revisão de Dalai [Dalal, 1988] que realiza a revisão de um conjunto de estados por um conjunto de estados, tendo como base a minimização da distância de Hamming entre estados. No entanto, o operador de revisão minimiza a distância global com relação a todos os estados do conjunto (e não estado por estado como ocorre na atualização).

Definição 4.3. *(Revisão de Dalai de um conjunto de estados) Sejam* $M = \langle S, L, T\rangle$ *um sistema de transição de estados e* S_0 *um subconjunto de* S. *A revisão de Dalai de* S_0 *por* S *é o conjunto de estados* $s \in S$ *que é mais próximo possível de* S_0 *com relação a distância de Hamming, i.e.,*

$$S_0 *^{\textit{dalai}} S = \{s \in S : \textit{ existe } s_0 \in S_0 \textit{ tal que } h(s_0, s) \leq h(s'_0, s') \textit{ para todo } s' \in S, s'_0 \in S_0\}.$$

A modificação do estado inicial, para que um problema $\Pi = \langle \mathbb{A}, C, s_0, \varphi\rangle$ sem solução torne-se solucionável, é um problema de atualização do estado inicial s_0 em relação ao conjunto de

estados do domínio de planejamento que alcançam um estado meta (i.e., os estados pertencentes a R na Figura 4.1), uma vez que deseja-se obter estados de R com distância minimal em relação a s_0. Já a modificação da meta de planejamento, para que um problema sem solução torne-se solucionável, é um problema de revisão do conjunto de estados meta G (i.e., o conjunto de estados que satisfaz a fórmula meta φ) em relação ao conjunto de estados do domínio de planejamento que são alcançáveis a partir do estado inicial s_0 (i.e., os estados pertencentes a R na Figura 6.1).

4.3 Métricas para Modificação de Problemas de Planejamento

Diversas são as modificações possíveis para que um problema de planejamento sem solução torne-se solucionável. O *princípio da mudança minimal* [Katsuno and Mendelzon, 1991b] exige que se obtenha uma modificação que altere o mínimo possível a definição do problema original, segundo uma determinada métrica. Nesta seção, definimos métricas para escolha da modificação do estado inicial, da meta de planejamento e das ações de acordo com: (1) as métricas de revisão de crenças e atualização de modelos (baseada na distância de Hamming); o tamanho do plano; (3) função custo e (4) preferência das submetas .

4.3.1 Métricas para Modificação do Estado Inicial

As métricas propostas neste trabalho para obtenção das modificações do estado inicial com distância minimal consideram diferentes critérios como: distância minimal de Hamming (métrica M1); mínima distância de Hamming com máxima distância da meta (métrica M21); distância minimal de Hamming com mínima distância da meta (métrica M22); menor custo (métrica M3); menor custo com máxima distância da meta (métrica M41) e; menor custo com mínima distância da meta (métrica M42).

Modificações minimais de acordo com a distância de Hamming Esta métrica é baseada na teoria de atualização de modelos que usa a distância de Hamming [Forbus, 1988] com diferença minimal entre s_0 e s'_0. Tal métrica deve ser utilizada quando o interesse é obter um novo estado inicial s'_0 cujo conjunto das proposições definidas na sua função rótulo seja o mais semelhante possível ao conjunto das proposições definidas no rótulo do estado inicial original s_0.

Definição 4.4. *(Métrica M1) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e o conjunto* U *de estados que alcançam algum estado satisfazendo a meta* φ*, uma modificação* s'_0 *do estado inicial possui distância minimal em relação a* s_0 *se e somente se:*

- $\Pi' = \langle \mathbb{A}, s'_0, \varphi \rangle$ *é um problema solucionável e;*
- $s'_0 \in s_0 \diamond^{forbus} U$.

Modificações minimais de acordo com a distância para a meta As métricas que analisam a distância do estado para a meta selecionam dentre os estados escolhidos pela métrica M1 aqueles que possuem distância maximal em relação a meta (M21) ou distância minimal em relação a meta (M22). A distância de um estado s para a meta é definida em relação ao tamanho do menor caminho possível entre o estado s e um estado meta s_g.

Definição 4.5. *(Distância para a meta) Dado um problema de planejamento* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *sobre um conjunto de proposições* $\mathbb{P}$*, a distância de um estado* s *tal que* $L(s) \subseteq 2^{\mathbb{P}}$ *para a meta* φ*, denotada por* $D_{meta}(s, \varphi)$*, é o menor inteiro tal que existe um caminho de tamanho* i *de* s *para um estado* $s_g \models \varphi$*.*

É desejável obter um estado com maior distância possível da meta nos casos em que um robô é o executor do plano e o objetivo é deixá-lo fazer a maior parte do trabalho. Por exemplo, no domínio das chaves, quando o robô está trancado em uma sala e não possui a chave para abrir a porta, é preferível fornecer a chave para que o robô abra a porta do que modificar estado inicial abrindo a porta para o robô. Uma vez que o robô foi projetado para realizar esse trabalho é desejável que se faça a menor interferência possível. No entanto, há situações em que é desejável escolher um novo estado inicial que minimize a distância para a meta. Por exemplo, no domínio de logística quando o problema não possui solução pela falta de um caminhão para entregar um determinado pacote, é interessante que o estado inicial seja modificado introduzindo um caminhão em um lugar mais próximo ao pacote para que o plano resultante seja o menor possível.

Definição 4.6. *(Métrica M21) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* s'_0 *um estado que é uma modificação minimal de* s_0 *de acordo com a métrica M1. Dizemos que* s'_0 *é um modificação minimal de* s_0 *considerando a maximização da distância para a meta* φ *se e somente se não existe* s''_0 *que é uma modificação minimal segundo a métrica M1 e que* $D_{meta}(s''_0, \varphi) > D_{meta}(s'_0, \varphi)$*.*

Definição 4.7. *(Métrica M22) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* s'_0 *um estado que é uma modificação minimal de* s_0 *de acordo com a métrica M1. Dizemos que* s'_0 *possui distância minimal de* s_0 *considerando a minimização da distância para a meta* φ *se e somente se não existe* s''_0 *que é minimal de acordo com a métrica M1 e que* $D_{meta}(s''_0, \varphi) < D_{meta}(s'_0, \varphi)$.

Modificações de menor custo Esta métrica considera a modificação do estado inicial de menor custo, não levando em consideração a quantidade de proposições que diferenciam o estado inicial original e o novo estado inicial. Considere que, por exemplo, uma possível modificação do estado inicial no domínio de logística seja introduzir conexões entre cidades com a construção de estradas ou aeroportos. No entanto, estas operações posssuem um alto custo e dificilmente serão selecionadas por esta métrica.

Definição 4.8. *(Métrica M3) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *sobre um conjunto de proposições* $\mathbb{P}$ *e uma função custo* $c(s_0, s)$ *que define o custo de transformar o estado inicial* s_0 *em um estado* s *cuja função rótulo é um subconjunto de* $2^{\mathbb{P}}$*, uma modificação* s'_0 *do estado inicial possui distância minimal em relação a* s_0 *se e somente se:*

- $\Pi' = \langle \mathbb{A}, s'_0, \varphi \rangle$ *é um problema solucionável e;*
- *Não existe* s''_0 *tal que* $\Pi'' = \langle \mathbb{A}, s''_0, \varphi \rangle$ *é solucionável e* $c(s_0, s'') \leq c(s_0, s')$.

A função custo pode ser qualquer função cujo domínio é um subconjutno de proposições e o contra-domínio é os números reais. Por simplicidade, iremos neste trabalho associar um custo $c(p)$ a cada uma das proposições $p \in \mathbb{P}$ e o custo de um subconjunto das proposições será a soma dos custos de cada $c(p)$. Assim, dados dois estados s_0 e s'_0 o custo de transformar s_0 em s'_0 é dado por $c(s_0, s'_0) = c(L(s_0) \Delta L(s'_0))$ em que Δ denota a diferença simétrica de conjuntos.

Modificações de menor custo considerando a distância para a meta É possível também propor uma métrica que considere dentre todas as modificações de menor custo aquela que maximize (métrica M41) ou minimize (métrica M42) a distância para a meta.

Definição 4.9. *(Métrica M41) (Modificação de menor custo considerando a distância para a meta) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* s'_0 *um estado que é uma modificação de menor custo. Dizemos que* s'_0 *possui distância minimal de* s_0 *considerando*

a maximização da distância para a meta φ se e somente se não existe s_0'' que também é uma modificação de menor custo e que $D_{meta}(s_0'', \varphi) > D_{meta}(s_0', \varphi)$.

Definição 4.10. *(Métrica M42) (Modificação de menor custo considerando a distância para a meta) Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ e seja s_0' um estado que é uma modificação de menor custo. s_0' possui distância minimal de s_0 considerando a minimização da distância para a meta φ se e somente se não existe s_0'' que também é uma modificação de menor custo e que $D_{meta}(s_0'', \varphi) < D_{meta}(s_0', \varphi)$.*

4.3.2 Métricas para a modificação da meta

As métricas para obtenção das modificações da meta consideram diferentes critérios como: mínima distância de Hamming (métrica M5); mínima distância de Hamming com máxima distância do estado inicial (métrica M61); mínima distância de Hamming com mínima distância do estado inicial (métrica M62); preferências (métrica M7); preferências com máxima distância da meta (métrica M81) e; preferências com mínima distância da meta (métrica M82).

Modificações minimais de acordo com a distância de Hamming A métrica M5 considera como modificação de distância minimal do conjunto de estados meta S_g, i.e. o conjunto de estados que satisfaz a meta φ, um conjunto de estados S_g' que possui diferença minimal de S_g em relação a distância de Hamming.

Definição 4.11. *(Métrica M5) Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ tal que $S_g = \{s_g \mid s_g \models \varphi\}$, o conjunto de estados $S_g' = \{s_g' \mid s_g' \models \varphi'\}$ possui modificação minimal em relação a S_g se e somente se:*

- *$\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$ é um problema solucionável e;*
- *S_g' é o resultado da revisão $S_g *^{dalai} S$.*

Modificações minimais de acordo com a distância para o estado inicial As métricas M61 e M62 selecionam dentre os estados obtidos pela métrica M1 aqueles que possuem distância maximal em relação ao estado inicial (M61) ou distância minimal em relação ao estado inicial (M62). A distância do estado inicial s_0 para um estado s é definida em relação ao tamanho do menor caminho possível que conecta esses dois estados.

Definição 4.12. *(Distância para o estado inicial) Dado um problema de planejamento* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$*, a distância do estado inicial* s_0 *para um estado* s*, denotada por* $D_{inicial}(s_0, s)$*, é o menor inteiro tal que há um caminho de tamanho* i *de* s_0 *para* s*.*

Em certas situações, é desejável que os novos estados meta estejam a uma menor distância possível do estado inicial, uma vez que não sendo possível alcançar a meta por inteiro é necessário atingir a nova meta no menor número de passos. Esta situação ocorre, por exemplo, no domínio de logística. Sendo impossível a entrega de dois pacotes, é necessário que se consiga entregar um deles em um menor número de passos. No entanto, em outras situações, é interessante que os novos estados meta estejam o mais distante possível do estado inicial. Esta situação ocorre, por exemplo, no domínio do robô de marte. Imagine que o robô tenha que comunicar à terra dados de análise de rocha e solo. No entanto, o canal de comunicação do robô só suporta uma única comunicação. Neste caso, mesmo sem conseguir comunicar os dados à terra é necessário que o robô colete a rocha ou o solo, uma vez que o ambiente pode sofrer alterações e não seja mais possível fazer a análise destes. No futuro, se o canal de comunicação voltar a funcionar, é possível comunicar os dados coletados e analisados anteriormente.

Definição 4.13. *(Métrica M61) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* S'_g *o conjunto de estados com distância minimal obtido pela métrica M5. O conjunto* S''_g *é o subconjunto dos estados de* S'_g *cuja distância para o estado inicial é maximal, i.e.,* $s''_g \in S''_g$ *se e somente se não existe* $s_g \in S'_g$ *tal que* $D_{inicial}(s_0, s_g) > D_{inicial}(s_0, s''_g)$*.*

Definição 4.14. *(Métrica M62) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* S'_g *o conjunto de estados com distância minimal obtido pela métrica M5. O conjunto* S''_g *é o subconjunto dos estados de* S'_g *cuja distância para o estado inicial é minimal, i.e.,* $s''_g \in S''_g$ *se e somente se não existe* $s_g \in S'_g$ *tal que* $D_{inicial}(s_0, s_g) < D_{inicial}(s_0, s''_g)$*.*

Modificações minimais baseada em preferências Uma vez que não é possível alcançar a meta por inteiro, é possível estabelecer preferências sobre quais proposições e seus valores devem ser alcançados. Considere, por exemplo, o domínio de logística em que há restrições de tempo tornando impossível a entrega dos cinco pacotes: p_1, p_2, p_3, p_4, p_5. Neste caso, é possível estabelecer preferências na entrega dos pacotes. Para representar preferências utiliza-se um grafo denominado rede de preferências [Brafman and Chernyavsky, 2005].

Definição 4.15. *(Rede de Preferências) Uma rede de preferências é um grafo em que os vértices são proposições e as arestas podem ser trê tipos: arestas de preferência condicional (*$A \rightarrow B$*),*

i.e., arestas direcionadas e não rotuladas indicando que a preferência sobre os valores de B depende do valor de A; arestas de importância ($A \Rightarrow B$), i.e., arestas direcionados e não-rotuladas denotando que é mais importante alcançar A do que B e; arestas de importância condicinal ($A - B$), i.e., arestas não direcionadas e rotuladas indicando que, dependendo do valor das proposições do rótulo, é mais importante alcançar A do que B ou é mais importante alcançar B do que A. Cada nó do grafo pode conter uma tabela condicional de preferências associada.

Exemplo 4.2. *(Expressando preferências no domínio de logística) Na impossibilidade de entregar os cinco pacotes p_1, p_2, p_3, p_4 e p_5 pode-se estabelecer preferências na entrega como: é preferível entregar o pacote p_1 do que não entregá-lo; se p_1 for entregue, é preferível que p_2 seja entregue mas se p_1 não for entregue não importa se p_2 será entregue ou não; se p_1 e p_2 forem entregues, é preferível que p_3 seja entregue mas se p_1 e p_2 não forem entregues não importa se p_3 será entregue; se p_1 não for entregue, não há necessidade de entregar p_2 nem p_3; é preferível entregar p_1 do que p_4; se p_4 for entregue, é preferível que p_5 seja entregue, mas se p_4 não for entregue não há necessidade de entregar p_5. Estas preferências podem ser visualizadas na rede de preferências da Figura 4.4.*

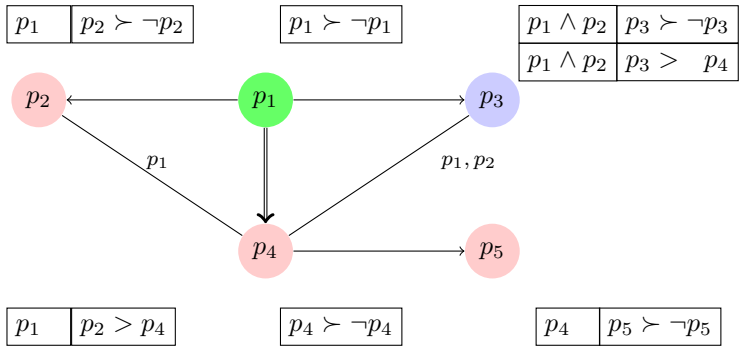


Figura 4.4: Rede de preferências para a entrega dos pacotes p_1, p_2, p_3, p_4 e p_5 no domínio de Logística.

Uma rede de preferências define uma ordem parcial sobre o conjunto de possíveis valores para as variáveis. Cada tabela de preferência, transição de importância e transição de importância condicional, adiciona alguma informação a esta ordem parcial.

Definição 4.16. *(Métrica M7) Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ sobre um conjunto de proposições $\mathbb{P}$ e uma rede de preferências sobre as proposições da meta, um estado s'_g é o mais preferível possível se e somente se:*

- $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$ *tal que* $s'_g \models \varphi'$ *é um problema solucionável e;*
- *Não existe* s''_g *tal que* $\Pi'' = \langle \mathbb{A}, s_0, \varphi'' \rangle$ *é solucionável e* $s''_g <_{pref} s'_g$.

Modificações minimais baseadas em preferências considerando a distância para o estado inicial É possível também propor uma métrica que considere dentre todas as modificações de menor custo aquela que maximize (métrica M81) ou minimize (métrica M82) a distância para o estado inicial.

Definição 4.17. *(Métrica M81) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* S'_g *o conjunto que contém todos os estados mais preferíveis em relação a meta* φ. $S''_g \subseteq S'_g$ *é o subconjunto dos estados de* S'_g *cuja distância para o estado inicial é maximal, i.e.,* $s''_g \in S''_g$ *se e somente se não existe* $s_g \in S'_g$ *tal que* $D(s_0, s_g) > D(s_0, s''_g)$.

Definição 4.18. *(Métrica M82) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e seja* S'_g *o conjunto que contém todos os estados mais preferíveis em relação a meta* φ. $S''_g \subseteq S'_g$ *é o subconjunto dos estados de* S'_g *cuja distância para o estado inicial é minimal, i.e.,* $s''_g \in S''_g$ *se e somente se não existe* $s_g \in S'_g$ *tal que* $D(s_0, s_g) < D(s_0, s''_g)$.

4.4 Formalização Utilizando a Lógica DL-PA

4.4.1 A Lógica DL-PA

Lógicas Dinâmicas [Harel et al., 2000] são lógicas modais utilizadas para raciocinar sobre programas. A lógica dinâmica mais básica é a *Lógica Dinâmica Proposicional* (PDL, do inglês *Propositional Dynamic Logic*). Para formalizar a tarefa de encontrar modificações para problemas de planejamento sem solução será utilizada a *Lógica Dinâmica de Atribuições Proposicionais* (DL-PA, do inglês *Dynamic Logic of Propositional Assignments*) [Balbiani et al., 2013] que é uma instanciação de PDL em que todo programa π é uma *atribuição* de valor verdadeiro ou falso a uma proposição , i.e., todo programa π é da forma $p \leftarrow \top$ ou $p \leftarrow \bot$ em que p é uma proposição.

Definição 4.19. *(Sintaxe da* DL-PA*) Seja* $\mathbb{P}$ *um conjunto de proposições, a sintaxe da* DL-PA *é definida de acordo com a gramática:*

$$\varphi \doteq p \in \mathbb{P} \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi?$$

$$\pi \doteq p \leftarrow \top \mid p \leftarrow \bot \mid \pi; \pi | \pi \cup \pi \mid \pi^* \mid \pi^- \mid \langle \pi \rangle \varphi.$$

Na Definição 4.19, a fórmula φ é uma fórmula proposicional e π é um *programa atômico*. As operações possíveis em DL-PA são: composição sequêncial (;); composição não determinística ($\cup$); iteração finita($(.)^\star$); teste ((?)) e; conversão ($(.)^-$). Finalmente, a fórmula modal $\langle \pi \rangle \varphi$ é verdadeira em um estado s se há uma execução de π a partir de s que leva a um estado satisfazendo φ.

Definição 4.20. *(Semântica da* DL-PA*) Define-se a função interpretação para fórmulas e programas. A função interpretação para fórmulas é dada como a seguir:*

- $||p|| = \{s : p \in s\}$;
- $||\top|| = 2^{\mathbb{P}}$;
- $|| \bot || = \emptyset$;
- $||\neg\varphi|| = 2^{\mathbb{P}} \setminus ||\varphi||$;
- $||\varphi \vee \psi|| = ||\varphi|| \vee ||\psi||$;
- $||\langle \pi \rangle \varphi|| = \{s :$ *existe* s_1 *tal que* $(s, s_1) \in ||\pi||$ *e* $s_1 \in ||\varphi||\}$.

A função de interpretação para programas é dada como a seguir:

- $||p \leftarrow \top|| = \{(s_1, s_2) : s_2 = s_1 \cup \{p\}\}$;
- $||p \leftarrow \bot || = \{(s_1, s_2) : s_2 = s_1 \setminus \{p\}\}$;
- $||\pi; \pi'|| = ||\pi|| \circ ||\pi'||$;
- $||\pi \cup \pi'|| = ||\pi|| \cup ||\pi'||$;
- $||\pi^*|| = \bigcup_{0 \leq k \leq n} ||(\pi)^k||$;
- $||\pi^-|| = (||\pi||)^{-1}$;
- $||\varphi?|| = \{(s, s) : s \in ||\varphi||\}$.

Dizemos que uma fórmula é DL-PA válida se é uma fórmula equivalente a $\bot$, i.e., se $||\varphi|| = 2^{\mathbb{P}}$. Dizemos também que uma fórmula é DL-PA satisfatível se não é uma fórmula equivalente a $\bot$, i.e., se $||\varphi|| \neq \varphi$.

Foi mostrado em [Herzig, 2014] que as operações de revisão e atualização podem ser escritas como programas DL-PA: a atualização de B por A tem os mesmos modelos que a fórmula DL-PA $\langle upd(A)^{-}\rangle B$ enquanto que a revisão de B por A pode ser capturada pelo programa $rev(A, B)$.

4.4.2 Problemas de Planejamento Expressos em DL-PA

O estado inicial s_0 de um problema de planejamento pode ser especificado pela fórmula DL-PA $\mathit{fml}(s_0) = \bigwedge_{p\in s_0} p \wedge \bigwedge_{p\in\mathbb{P}\setminus s_0} \neg p$. O conjunto de estados meta S_g pode ser dado pela fórmula DL-PA $\mathit{fml}(S_g) = \bigvee_{s\in S_g} \mathit{fml}(s)$.

Definição 4.21. *(Ação como programa DL-PA) Uma ação* STRIPS $a = \langle precond(a); add(a); del(a)\rangle$ *pode ser expressa como um programa DL-PA* $pmg(a)$*, como definido a seguir:*

$$pmg(a) = precond(a)?; p_1 \leftarrow \top, \cdots p_k \leftarrow \top; q_1 \leftarrow\bot, \cdots q_l \leftarrow\bot$$

em que $\{p_1, \cdots p_k\}$ *é o conjunto das proposições que ocorre em* $add(a)$ *e* $\{q_1 \leftarrow\bot, \cdots q_l\}$ *é o conjunto das proposições que ocorre em* $del(a)$*.*

Exemplo 4.3. *(Ações do domínio das chaves expressas em DL-PA) A ação mover-0-1 do domínio das chaves* $\langle\{$*loc-robô-sala0*, *aberta-porta0*, *conecta-sala0-sala1-porta0*$\}; \langle\{$*loc-robô-sala0*$\}; \{$ *loc-robô-sala1*$\}\rangle\rangle$ *pode ser capturada pelo programa DL-PA:*

$$pgm(\textit{mover-0-1}) = \textit{loc-robô-sala0} \ \wedge\ \textit{aberta-porta0} \ \wedge\ \textit{conecta-sala0-sala1-porta0}?$$
$$\textit{loc-robô-sala0} \leftarrow \top; \textit{loc-robô-sala1} \leftarrow\bot\ .$$

Do mesmo modo, a ação abrir0 = $\langle\{$*loc-robô-sala0, loc-chave-sala0*, *conecta-sala0-sala1-porta0*$\}$ $; \langle\{$*aberta-porta0*$\}; \{\}\rangle\rangle$ *é capturada pelo programa:*

$$pgm(\textit{abrir0}) = \textit{loc-robô-sala0} \wedge \textit{loc-chave-sala0} \wedge \textit{conecta-sala0-sala1-porta0}?$$
$$\textit{aberta-porta0} \leftarrow \top.$$

Dado um conjunto de ações $\mathbb{A} = \{a_1, \cdots, a_n\}$, o programa DL-PA $(pmg(a_1) \cup \cdots \cup pmg(a_n))^*$ descreve todas as possíveis sequências finitas de execução das ações em $\mathbb{A}$. Por

exemplo, para $\mathbb{A} = \{$*mover-0-1*, *abrir0*$\}$, a fórmula DL-PA $(pmg($*mover-0-1*$) \cup pmg($*abrir-0*$))^*$ expressa todas as possíveis combinações de execução das ações *mover-0-1* e *abrir0*. Similarmente, o programa $(pmg(a_1)^- \cup \cdots \cup pmg(a_n)^-)^*$ descreve todas as possíveis sequências finitas das ações executadas em ordem reversa.

Definição 4.22. *(Conjunto dos estados que alcançam um estado meta) Dado um problema de planejamento $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$, o conjunto dos estados que finalmente alcançam um estado satisfazendo a meta φ, por meio da execução das ações $\mathbb{A}$, pode ser representado pela seguinte fórmula* DL-PA*:*

$$\textit{Alcançam}(S_g, \mathbb{A}) = \langle (pmg(a_1) \cup \cdots \cup pmg(a_n))^* \rangle \varphi,$$

em que S_g é o conjunto de estados que satisfaz a meta φ.

Exemplo 4.4. *(Conjunto dos estados que alcançam um estado meta no domínio das chaves) No domínio das chaves, o conjunto dos estados que finalmente alcançam um estado satisfazendo a meta loc-robô-sala1, por meio das ações mover-0-1 e abrir0, pode ser representado pela fórmula* DL-PA*:*

$$\textit{Alcançam}(S_g, \{\textit{mover-0-1,abrir0}\}) = \langle (\textit{pmg(mover-0-1)} \cup \textit{pmg(abrir0)})^* \rangle \textit{loc-robô-sala1}.$$

Definição 4.23. *(Conjunto dos estados que alcançam o estado inicial) Dado um problema de planejamento $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$, o conjunto dos estados que finalmente alcançam o estado inicial s_0, considerando a execução na ordem reversa das ações $\mathbb{A}$, pode ser representado pela seguinte fórmula* DL-PA*:*

$$\textit{Alcançam}(s_0, \mathbb{A}^-) = \langle (pmg(a_1)^- \cup \cdots \cup pmg(a_n)^-)^* \rangle \textit{fml}(s_0).$$

Definição 4.24. *(Problemas sem solução em* DL-PA*) Um problema de planejamento $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$ possui solução quando a fórmula* DL-PA *$\textit{fml}(s_0) \rightarrow \textit{Alcançam}(S_g, \mathbb{A})$ é válida.*

4.4.3 Modificando Problemas de Planejamento sem Solução com DL-PA

A lógica DL-PA pode ser utilizada para formalizar a tarefa de modificar um problema de planejamento sem solução para que ele torne-se solucionável. Como mostrado em [Herzig et al., 2014], dado um problema de planejamento sem solução Π, a modificação do estado inicial para que Π torne-se solucionável é uma operação de atualização enquanto que a modificação da meta para que Π torne-se solucionável é uma operação de revisão.

Definição 4.25. *(Atualização do estado inicial em* DL-PA*) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$*, o conjunto dos estados mais próximos a* s_0 *que tornam* Π *solucionável pode ser obtido a partir da atualização de fml*(s_0) *por Alcançam*$(S_g, \mathbb{A})$*, conforme descrito a seguir:*

$$||s_0 \diamond^{forbus} Alcançam(S_g, \mathbb{A})|| = ||\langle upd^{forbus}(Alcançam(S_g, \mathbb{A}))^{-}\rangle fml(s_0)||.$$

Definição 4.26. *(Revisão do conjunto de estados meta em* DL-PA*) Dado um problema de planejamento sem solução* $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$*, o conjunto dos estados mais próximos ao conjunto dos estados meta* S_g *(i.e., o conjunto dos estados que satisfazem* φ*) que tornam* Π *solucionável, pode ser obtido pela revisão de Alcançam*$(s_0, \mathbb{A}^{-})$ *por* φ*, conforme descrito a seguir:*

$$||S_g *^{dalai} Alcançam(s_0, \mathbb{A}^{-})|| = ||\langle (rev^{dalai}(Alcançam(s_0, \mathbb{A}^{-}), \varphi)^{-}\rangle \varphi||.$$

4.5 Formalização Utilizando Verificação de Modelos

4.5.1 O Arcabouço de Verificação de Modelos

O problema da verificação de modelos consiste em determinar se um modelo formal $\mathcal{M}$, descrevendo a dinâmica de um sistema, satisfaz uma dada propriedade φ. A Figura 4.5 ilustra um esquema básico do verificador de modelos. O algoritmo de verificação recebe como entrada o par ($\mathcal{M}$, φ) e verifica a validade da propriedade φ em todos os estados do modelo $\mathcal{M}$. Se todos os estados de $\mathcal{M}$ satisfazem a propriedade φ, o algoritmo devolve *sucesso*. Caso contrário um *contraexemplo* é devolvido (por exemplo, um conjunto de estados que não satisfaz φ).

Figura 4.5: *Verificador de Modelos [Pereira, 2007].*

A lógica temporal CTL

CTL (COMPUTATION TREE LOGIC) [Clarke and Emerson, 1982] é uma lógica temporal de tempo ramificado que permite especificar propriedades quantificadas sobre caminhos em uma árvore de computação. Esta lógica é dita de tempo ramificado pois considera que para cada instante de tempo pode haver vários futuros possíveis. Há quatro operadores temporais nesta lógica: $\circ$

(*próximo*), $\Diamond$ (*finalmente*), $\Box$ (*globalmente*) e $\sqcup$ (*até que*). Todo operador temporal é precedido de um quantificador, existencial ($\exists$) ou universal ($\forall$). A Definição 4.27 apresenta a sintaxe da lógica CTL.

Definição 4.27. *(Sintaxe da lógica CTL) O conjunto de fórmulas CTL é definido indutivamente como:* $\varphi \doteq p \in \mathbb{P} \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists\circ\varphi_1 \mid \forall\circ\varphi_1 \mid \exists\Box\varphi_1 \mid \forall\Box\varphi_1 \mid \exists\Diamond\varphi_1 \mid \forall\Diamond\varphi_1 \mid \exists(\varphi_1 \sqcup \varphi_2) \mid \forall(\varphi_1 \sqcup \varphi_2)$

Intuitivamente: $\forall$ significa para todos os caminhos; $\exists$ significa que existe um caminho; $\circ$ refere-se ao próximo estado; $\Box$ refere-se a todos os estados globalmente; $\Diamond$ refere-se a finalmente em um estado; e $\sqcup$ refere-se à obrigatoriedade da validade de uma fórmula até que outra seja válida. A Figura 4.6 ilustra o significado dos operadores temporais da lógica CTL. Sendo s um estado inicial, pode-se constatar que: em (a) existe um próximo estado em que p é verdadeira; em (b) existe um caminho a partir de s em que p é verdadeira para todos os estados; em (c) existe um caminho em que p é finalmente verdadeira; em (e) para todos os próximos estados p é verdadeira; em (f) para todos os caminhos, partindo de s, p é verdadeira; em (g) p é finalmente verdadeira para todos os caminhos; e em (h), para todos os caminhos p é verdadeira até que q seja verdadeira. Formalmente, a semântica da lógica CTL é dada como na Definição 4.28.

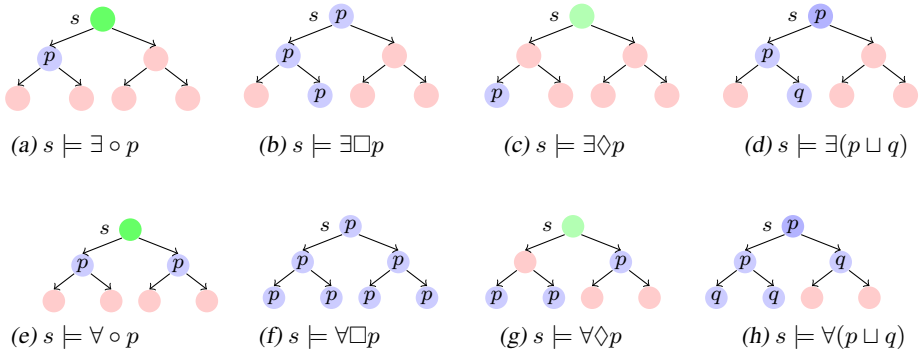


Figura 4.6: *Semântica dos operadores temporais da lógica* CTL.

Definição 4.28. *(Semântica da Lógica CTL) Sejam $\mathcal{M}$ uma estrutura de Kripke, s um estado dessa estrutura e φ uma fórmula* CTL. *A semântica das fórmulas* CTL *é definida como segue:*

- $s \models p$ *se e somente se* $p \in \mathcal{L}(s)$;
- $s \models \neg\varphi_1$ *se e somente se* $s \not\models \varphi_1$;
- $s \models \varphi_1 \wedge \varphi_2$ *se e somente se* $s \models \varphi_1$ *e* $s \models \varphi_2$;

- $s \models \varphi_1 \vee \varphi_2$ *se e somente se* $s \models \varphi_1$ *ou* $s \models \varphi_2$*;*
- $s \models \exists \circ \varphi_1$ *se e somente se, para algum caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, $\rho_1 \models \varphi_1$*;*
- $s \models \forall \circ \varphi_1$ *se e somente se, para todo caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, $\rho_1 \models \varphi_1$*;*
- $s \models \exists\Box\varphi_1$ *se e somente se, para algum caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$ *e todo* $i \geq 1$, $\rho_i \models \varphi_1$*;*
- $s \models \forall\Box\varphi_1$ *se e somente se, para todo caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$ *e todo* $i \geq 1$, $\rho_i \models \varphi_1$*;*
- $s \models \exists\Diamond\varphi_1$ *se e somente se para algum caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, *existe* $i \geq 0$ *tal que* $\rho_i \models \varphi_1$*;*
- $s \models \forall\Diamond\varphi_1$ *se e somente se para todo caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, *existe* $i \geq 0$ *tal que* $\rho_i \models \varphi_1$*;*
- $s \models \exists(\varphi_1 \sqcup \varphi_2)$ *se e somente se para algum caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, *existe* $j \geq 0$ *tal que* $\rho_j \models \varphi_2$ *e, para todo* $i < j$, $\rho_i \models \varphi_1$*;*
- $s \models \forall(\varphi_1 \sqcup \varphi_2)$ *se e somente se para todo caminho* $\rho \in \Upsilon^s_{\mathcal{M}}$, *existe* $j \geq 0$ *tal que* $\rho_j \models \varphi_2$ *e, para todo* $i < j$, $\rho_i \models \varphi_1$.

Caracterização de ponto fixo da CTL

Nesta subseção, apresentamos a caracterização de ponto-fixo da CTL: a fundamentação matemática do algoritmo de verificação de modelos. Inicialmente, mostramos o conceito de ponto fixo na Definição 4.29. Em seguida, apresentamos como os operadores temporais CTL podem ser escritos em termos de ponto fixos.

Um *ponto fixo* de uma função Γ é qualquer A tal que $\Gamma[A] = A$. O teorema a seguir garante que uma função monótona, sempre tem pontos-fixos mínimos e máximos.

Definição 4.29. *(Pontos fixos) Se* $\Gamma[Y]$ *é uma função monótona, então* $\Gamma[Y]$ ***tem*** *um ponto fixo mínimo denotado por* $\mu Y.\Gamma[Y]$ *e um ponto fixo máximo denotado por* $\nu Y.\Gamma[Y]$ *[Tarski, 1955].*

Teorema 4.1. *Se* $\mathcal{S}$ *é finito, os operadores temporais globais de* CTL *podem ser caracterizados em termos de pontos fixos mínimos e máximos, conforme a seguir: [Huth and Ryan, 2004]*

- $\exists\Box\varphi = \nu Y.(\varphi \wedge \exists \circ Y)$
- $\forall\Box\varphi = \nu Y.(\varphi \wedge \forall \circ Y)$
- $\exists\Diamond\varphi = \mu Y.(\varphi \vee \exists \circ Y)$
- $\forall\Diamond\varphi = \mu Y.(\varphi \vee \forall \circ Y)$
- $\exists(\varphi_1 \sqcup \varphi_2) = \mu Y.(\varphi_2 \vee (\varphi_1 \wedge \exists \circ Y))$
- $\forall(\varphi_1 \sqcup \varphi_2) = \mu Y.(\varphi_2 \vee (\varphi_1 \wedge \forall \circ Y))$

Limitações da lógica CTL

Nesta seção mostramos, por meio de um exemplo [Pereira, 2007], que a lógica CTL é inadequada para expressar determinadas propriedades dos sistemas. Considere um *agente aspirador de pó* que, além de realizar a limpeza do ambiente, seja capaz de dirigir-se a uma estação de recarga sempre que sua bateria estiver fraca. Considere ainda o sistema de transições rotuladas deste agente e sua estrutura de *Kripke* correspondente (Figura 4.7). A proposição r representa os estados onde há uma estação de recarga de bateria e a proposição g, uma meta que o agente deve alcançar.

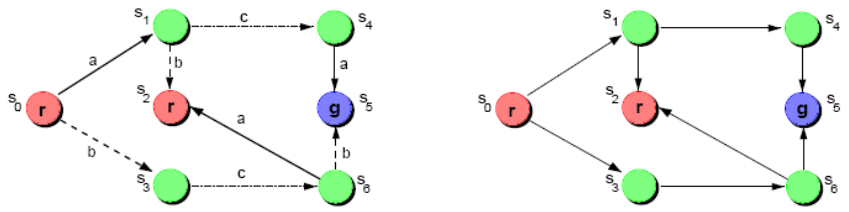


Figura 4.7: *(a) Sistema de transições rotuladas para o agente aspirador de pó e (b) estrutura de Kripke correspondente [Pereira, 2007].*

Uma propriedade desejável para o *agente aspirador de pó* seria: *"necessariamente alcance um estado que satisfaça g, passando apenas por estados a partir dos quais seja possível alcançar uma estação de recarga em no máximo dois passos."*.

A fórmula CTL mais adequada para expressar esta propriedade seria: $\forall((r \vee \forall \circ r \vee \forall \circ \forall \circ r) \sqcup g)$. No entanto, há alguns problemas nesta especificação [Pereira, 2007]:

- Não queremos expressar que o agente tenha *realmente* que alcançar um estado onde haja uma estação de recarga de bateria. Desejamos apenas que ele alcance este estado, quando sua bateria estiver fraca. A lógica CTL considera que cada "seta" que parte de um estado é uma transição que será executada e, por isso, não permite raciocinar sobre as ações que não serão efetivamente executadas. Assim, é claro que a semântica da fórmula $\forall((r \vee \forall \circ r \vee \forall \circ \forall \circ r) \sqcup g)$ não especifica exatamente a propriedade que desejamos verificar no sistema.

- Mesmo assumindo que a fórmula $\forall((r \vee \forall \circ r \vee \forall \circ \forall \circ r) \sqcup g)$ possa ser usada para especificar a propriedade, podemos ver que $(\mathcal{M}, s_0) \not\models \forall((r \vee \forall \circ r \vee \forall \circ \forall \circ r) \sqcup g)$

pois, na estrutura de *Kripke* $\mathcal{M}$, ilustrada na Figura 4.7(b), não há garantia de que uma estação de recarga possa ser alcançada em no máximo dois passos, em todos os estados nos caminhos que partem do estado inicial s_0 (*e.g.*, a partir de s_4 não há como alcançar uma estação de recarga). Note, porém, que:

- há uma estação de recarga no estado s_0;
- a partir do estado s_3, a estação de recarga em s_2 pode ser alcançada em dois passos;
- a partir do estado s_6, a estação de recarga em s_2 pode ser alcançada em um passo.

Portanto há um caminho no modelo, partindo de s_0, no qual o agente alcança g e garantidamente está (em todos os estados deste caminho) há no máximo dois passos da estação de recarga. Porém, a estrutura de *Kripke* $\mathcal{M}$ não satisfaz a propriedade especificada em CTL. Desta forma, temos que a semântica da CTL não é adequada para expressar este tipo de propriedade.

A lógica α-CTL

A lógica α-CTL é uma extensão da lógica CTL que é capaz de distinguir as ações que produzem as transições entre estados. Os símbolos dos operadores temporais α-CTL são diferenciados dos operadores CTL por serem "pontuados". Assim sendo: $\odot$ (sucessor imediato), $\boxdot$ (invariantemente), $\Diamond$ (finalmente) e $\sqcup$ (até que). De acordo com a sintaxe de α-CTL, todos os operadores temporais devem ser prefixados por um quantificador de caminhos ($\exists$ ou $\forall$).

Definição 4.30. *(Sintaxe de α-CTL) Seja $p \in \mathbb{P}$ uma proposição atômica, a sintaxe de α-CTL é definida indutivamente como:* $\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists \odot \varphi_1 \mid \forall \odot \varphi_1 \mid \exists \boxdot \varphi_1 \mid \forall \boxdot \varphi_1 \mid \exists \diamond \varphi_1 \mid \forall \diamond \varphi_1 \mid \exists(\varphi_1 \sqcup \varphi_2) \mid \forall(\varphi_1 \sqcup \varphi_2)$

Intuitivamente, um estado s em um sistema de transições rotuladas $\mathcal{M}$ satisfaz uma fórmula $\forall \odot \varphi$ se *existe* uma ação $\alpha \in \mathbb{A}$ que, quando executada em s, *necessariamente* alcança um estado sucessor imediato de s que satisfaz a fórmula φ. Já um estado s em um sistema de transições rotuladas $\mathcal{M}$ satisfaz uma fórmula $\exists \odot \varphi$ se *existe* uma ação $\alpha \in \mathbb{A}$ que, quando executada em s, *possivelmente* alcança um estado sucessor imediato de s que satisfaz a fórmula φ. Isto é, a modalidade $\odot$ representa o conjunto de α-sucessores de s, *para alguma ação particular* α; o quantificador $\forall$ requer que *todos* esses α-sucessores satisfaçam φ; e o quantificador $\exists$ requer que *algum* desses α-sucessores satisfaça φ.

Na Figura 4.8 é possível visualizar a semântica dos operadores temporais α-CTL. Observe que agora as transições de estados são rotuladas pelas ações a, b, c e d. Sejam $\mathcal{M} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$ um sistema de transições rotuladas com assinatura $(\mathbb{P}, \mathbb{A})$ e $s \in \mathcal{S}$ o estado inicial, temos que:

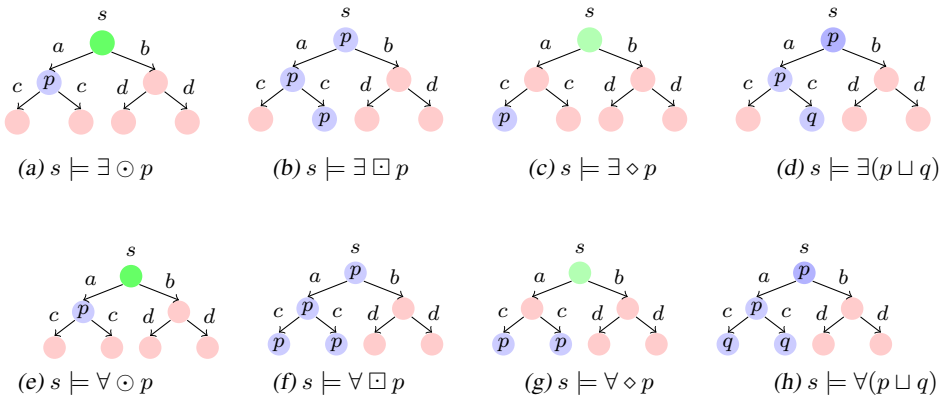


Figura 4.8: *Semântica dos operadores temporais da lógica* α-CTL.

- Em (a), $s \models \exists \odot p$ expressa que *existe um próximo estado*, *para alguma ação*, em que p é verdade (neste caso, executando-se a ação a);
- Em (e), $s \models \forall \odot p$ expressa que em *todos os próximos estados*, *para alguma ação*, p é verdade (neste caso, executando-se a ação a);
- Em (b), $s \models \exists \boxdot p$ significa que existe um caminho na árvore no qual em todos os estados, *para alguma ação*, é válida a proposição p (neste caso, executando-se a ação a);
- Em (f), $s \models \forall \boxdot p$ expressa que para todos os caminhos da árvore, partindo de s, *para alguma ação*, é válida a proposição p (neste caso, executando-se a ação a);
- Em (c), $s \models \exists \diamond p$ significa que em algum estado futuro para alguma ação é válida a proposição p (neste caso, executando-se a ação a);
- Em (g), $s \models \forall \diamond p$ expressa que para todos os estados futuros, *para alguma ação*, é válida a proposição p (neste caso, executando-se a ação a);
- Em (d), $s \models \exists(p \sqcup q)$ exprime que algum estado futuro, *para alguma ação*, a proposição p é válida até que a proposição q seja verdade (neste caso, executando-se a ação a);
- Em (h), $s \models \forall(p \sqcup q)$ expressa que para todos os estados do caminho a partir de s, *executando-se alguma ação*, a proposição p é verdade até que q seja verdade (neste caso, executando-se a ação a).

Na lógica α-CTL , os conceitos de pré-imagem de um conjunto de estados foram redefinidos para englobar as ações que rotulam as transições entre estados, de acordo com a Definição **??**.

A seguir, temos a definição formal da semântica de α-CTL. Note que os operadores temporais globais (*i.e.*, $\exists\boxdot$, $\forall\boxdot$, $\exists\sqcup$, $\forall\sqcup$) são definidos em termos das operações de ponto fixo mínimo e máximo.

Definição 4.31. *(Semântica de α-CTL) Sejam $\mathcal{M} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$ um modelo temporal com assinatura $(\mathbb{P}, \mathbb{A})$ e $p \in \mathbb{P}$ uma proposição atômica. A intensão de uma fórmula α-CTL φ em $\mathcal{M}$ (i.e., o conjunto de estados de $\mathcal{M}$ que satisfazem φ), denotado por $\| \varphi \|_{\mathcal{M}}$, é definida indutivamente como:*

- $\| p \|_{\mathcal{M}} = \{s \in \mathcal{S} : p \in \mathcal{L}(s)\}$
- $\| \neg p \|_{\mathcal{M}} = \mathcal{S} \setminus \| p \|_{\mathcal{M}}$
- $\| \varphi_1 \wedge \varphi_2 \|_{\mathcal{M}} = \| \varphi_1 \|_{\mathcal{M}} \cap \| \varphi_2 \|_{\mathcal{M}}$
- $\| \varphi_1 \vee \varphi_2 \|_{\mathcal{M}} = \| \varphi_1 \|_{\mathcal{M}} \cup \| \varphi_2 \|_{\mathcal{M}}$
- $\| \exists \odot \varphi_1 \|_{\mathcal{M}} = pre_{\exists}(\| \varphi_1 \|_{\mathcal{M}})$
- $\| \forall \odot \varphi_1 \|_{\mathcal{M}} = pre_{\forall}(\| \varphi_1 \|_{\mathcal{M}})$
- $\| \exists \boxdot \varphi_1 \|_{\mathcal{M}} = \nu Y.(\| \varphi_1 \|_{\mathcal{M}} \cap pre_{\exists}(Y))$
- $\| \forall \boxdot \varphi_1 \|_{\mathcal{M}} = \nu Y.(\| \varphi_1 \|_{\mathcal{M}} \cap pre_{\forall}(Y))$
- $\| \exists \diamond \varphi_1 \|_{\mathcal{M}} = \| \exists(\top \sqcup \varphi_1) \|_{\mathcal{M}}$
- $\| \forall \diamond \varphi_1 \|_{\mathcal{M}} = \| \forall(\top \sqcup \varphi_1) \|_{\mathcal{M}}$
- $\| \exists(\varphi_1 \sqcup \varphi_2) \|_{\mathcal{M}} = \mu Y.(\| \varphi_2 \|_{\mathcal{M}} \cup (\| \varphi_1 \|_{\mathcal{M}} \cap pre_{\exists}(Y)))$
- $\| \forall(\varphi_1 \sqcup \varphi_2) \|_{\mathcal{M}} = \mu Y.(\| \varphi_2 \|_{\mathcal{M}} \cup (\| \varphi_1 \|_{\mathcal{M}} \cap pre_{\forall}(Y)))$

4.5.2 Modificando Problemas de Planejamento sem Solução com Verificação de Modelos

O arcabouço formal de verificação de modelos pode ser utilizado para obter modificações aceitáveis e boas modificações do estado inicial. Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, a ideia central é que o conjunto R de estados alcançados na busca regressiva, durante o processo de verificação da satisfação de φ, contém o conjunto das modificações aceitáveis e das boas modificações para Π. Um estado s é uma modificação aceitável para Π se $s \in R$ (i.e,

a partir de s é possível finalmente alcançar φ) e s possui distância minimal em relação a s_0. Já um estado s é dito ser uma boa modificação para Π se s é aceitável e não existe s' tal que s' é aceitável e s' está no caminho que inicia em s e termina em um estado satisfazendo φ.

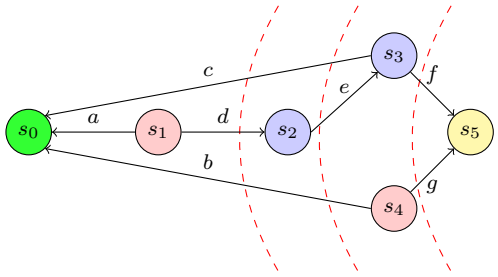


Figura 4.9: Iniciando do estado meta s_5 não é possível alcançar o estado inicial s_0.

A Figura 4.9 ilustra como o processo de verificação de modelos pode ser utilizado para gerar uma modificação do estado inicial para o problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ em que $\mathbb{A} = \{a, b, c, d, e, f, g\}$ e $s_5 \models \varphi$. Iniciando a busca regressiva no estado meta s_5, obtém-se, na primeira iteração, o conjunto dos estados que antecedem s_5 os quais são $X = \{s_3, s_4\}$. Assim, $R = \{s_3, s_4, s_5\}$. Na segunda iteração obtém-se os antecessores de R, os quais são $X = \{s_2\}$. Assim, $R = \{s_2, s_3, s_4, s_5\}$. Na terceira iteração, obtém-se os antecessores de R os quais são $X = \{s_1\}$. Assim, $R = \{s_1, s_2, s_3, s_4, s_5\}$. Na quarta iteração nenhum novo estado antecessor é encontrado e um ponto-fixo é atingido. No entanto, note que todos os estados pertencentes ao conjunto R finalmente alcançam o estado meta s_5. Desta forma, as modificações aceitáveis e também as boas modificações para o estado inicial s_0 são subconjuntos do conjunto R.

A seguir, dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ mostraremos a formalização do problema de encontrar modificações aceitáveis e boas modificações do estado inicial para que Π torne-se solucionável, utilizando o arcabouço formal de verificação de modelos.

Definição 4.32. *(Encontrando modificações aceitáveis do estado inicial com verificação de modelos) Dado um problema de planejamento não-determinístico sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *as modificações aceitáveis do estado inicial* s_0 *para tornar* Π *solucionável podem ser obtidas como a seguir.*

$$\textit{aceitáves-}s_0 = \{s \mid s \in s_0 \diamond_R^{\textit{forbus}}\}$$

tal que $R = \{(M, s_0) \models \exists \diamond \varphi\}$ *para uma solução fraca;* $R = \{(M, s_0) \models \forall \diamond \varphi\}$ *para uma solução forte e;* $R = \{(M, s_0) \models \forall \diamond \varphi\}$ *para uma solução forte-cíclica.*

Definição 4.33. *(Encontrando boas modificações do estado inicial com verificação de modelos) Dados um problema de planejamento não-determinístico sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *e o conjunto* Y *das modificações aceitáveis do estado inicial para este problema, o conjunto de boas modificações de* s_0 *para tornar* Π *solucionável pode ser obtido como a seguir:*

$$\textit{boas-}s_0 = \{s \in Y \mid \neg\exists s' \in Y \textit{ e } s' \in \{(M, s) \models \exists \circ \Psi\}\}.$$

tal que $\Psi = \exists \diamond \varphi$ *para uma solução fraca;* $\Psi = \forall \diamond \varphi$ *para uma solução forte e;* $\Psi = \forall \diamond \varphi$ *para uma solução forte-cíclica.*

Definição 4.34. *(Encontrando modificações aceitáveis da meta com verificação de modelos) Dado um problema de planejamento não-determinístico sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *as modificações aceitáveis da meta* φ *para tornar* Π *solucionável podem ser obtidas como a seguir.*

$$\textit{aceitáves-}g = \{s \mid s \in s_0 *_R^{\textit{dalai}}\}$$

tal que $R = \{(M, s_0) \models \exists \diamond \varphi\}$ *para uma solução fraca;* $R = \{(M, s_0) \models \forall \diamond \varphi\}$ *para uma solução forte e;* $R = \{(M, s_0) \models \forall \diamond \varphi\}$ *para uma solução forte-cíclica.*

Capítulo 5

Modificação do Estado Inicial

5.1 Explicações para Problemas sem Solução com Variáveis Multivaloradas

Göbelbecker et. al., (2010) propõe um método capaz de sugerir modificações no estado inicial de um problema sem solução para que ele torne-se solucionável. A modificação proposta pode ser apenas um nova configuração dos objetos existentes no problema, ou ainda, o acréscimo de novos objetos ao problema. O novo estado inicial , o qual pode ser definido sobre o conjunto original de objetos ou sobre este conjunto adicionado de novos objetos, é chamado *estado explicação*. O método de Göbelbecker et. al., (2010) utiliza a representação implícita do domínio de planejamento com variáveis multivaloradas.

5.1.1 Representação de Domínios de Planejamento com Variáveis Multivaloradas

Considere o átomo *robô-pos(sala0)* na representação do domínio das chaves relacional. Este átomo representa um elemento de uma relação entre o robô e a sala *sala0*. O valor verdade desta relação varia de um estado para outro. No entanto, uma vez que o robô não pode estar em mais de uma sala ao mesmo tempo, há somente uma sala x tal que *robô-pos(sala x)* é verdadeira em um estado s. Na representação baseada em variáveis multivaloradas, a relação *robô-pos(sala0)* é transformada em uma *variável multivalorada robô-pos* que pode assumir o conjunto de possíveis

salas em que o robô pode estar, i.e., *sala0* ou *sala1*. O conjunto de valores que uma variável multivalorada v pode assumir é denominado domínio da variável v, denotado por *dom*(v). Assim, uma variável multivalorada com aridade k é uma expressão da forma $v(x_1, \cdots, x_k)$ que pode assumir algum valor $y \in$ *dom*(v).

O domínio das chaves pode ser descrito pelo seguinte conjunto de variáveis multivaloradas: $\mathcal{V} = \{$*robô-pos*, *chave-pos*, *aberta(porta d)*, *conecta(sala x, sala y, porta d)*, *abre(chave k)*$\}$. Os domínios de cada uma das variáveis $v \in \mathcal{V}$ é dado por: dom(*robô-pos)* $= \{\text{sala0}, \text{sala1}\}$; dom(*chave-pos)* $= \{\text{sala0}, \text{sala1}\}$; *dom(aberta(porta d))* $= \{\top, \bot\}$; *dom(aberta(porta d))* $= \{\top, \bot\}$ e *dom(abre(chave k))* $= \{$*porta0*$\}$. Uma variável multivalorada $v(x_1, \cdots, x_k)$ é dita instanciada se cada $x_i (1 \leq i \leq k)$ é um símbolo constante. Nesta representação, um estado é um conjunto de variáveis multivaloradas que podem receber um dos valores do domínio da variável e as ações são responsáveis por modificar os valores das variáveis de acordo com seus domínios.

Para representar implicitamente domínios de planejamento com variáveis multivaloradas, é utilizado o formalismo SAS$^+$ [Bäckström and Nebel, 1995] em que cada operador é definido por pré-condições e efeitos sobre as variáveis multivaloradas não instanciadas. A Definição 5.1 mostra como representar operadores no formalismo SAS$^+$.

Definição 5.1. *(Representaçao Implícita para Domínios de Planejamento com Variáveis Multivaloradas) Dado um conjunto de variáveis multivaloradas* $\mathcal{V} = \{v_1(x_1, \cdots, x_m), \cdots, v_n(x_1, \cdots, x_m)\}$*, um domínio de planejamento pode ser descrito implicitamente por um conjunto de operadores* $\mathcal{O}$ *em que cada* $o \in \mathcal{O}$ *é dado em termos de precondições e efeitos (*$o = \langle$*precond*(o); *efeitos*$(o)\rangle$*) definidos sobre as variáveis multivaloradas, i.e.:*

- *precond*$(o) = \{(v_i(x_i, \cdots, x_k) = x_{k+1})\}$ *e;*
- *efeitos*$(o) = \{(v_i(x_i, \cdots, x_k) = x_{k+1})\}$.

Exemplo 5.1. *(Representaçao Implícita de um Domínio de Planejamento com Variáveis Multivaloradas) A Figura 5.1 mostra a representação implícita do domínio das chaves com variáveis multivaloradas. Cada operador* $o \in \mathcal{O}$ *é dado por dois conjuntos representando as pré-condições e efeitos do operador.*

mover(sala x, sala y, porta z) :
⟨*precond* = {*robô-pos* = *x*, *aberta(z)* = ⊤, *conecta (x, y, z)* = ⊤};
efeitos = {*robô-pos* = *y*}⟩
mover-com-chave(sala x, sala y, porta z, chave k) :
⟨*precond* = {*robô-pos* = *x*, *chave-pos(k)* = *x*, *aberta(z)* = ⊤, *conecta(x, y, z)* = ⊤};
efeitos = {*robô-pos* = *y*, *chave-pos(k)* = *y*}⟩
abrir(porta z, chave k, sala x, sala y) :
⟨*precond* = {*robô-pos* = *x*, *chave-pos(k)* = *x*, *conecta(x, y, z)* = ⊤};
efeitos = {*aberta(z)* = ⊤}⟩
fechar-porta0-sala0-sala1 :
⟨*precond* = {*robô-pos* = *x*, *conecta(x,y,z)* = ⊤, *aberta(z)* = ⊤};
efeitos = {*aberta(z)* =⊥}⟩

Figura 5.1: Representação implícita do domínio das chaves relacional.

5.1.2 Métricas para Modificação do Estado Inicial

[Göbelbecker et al., 2010] classificou as modificações do estado inicial em três categorias: *aceitáveis*, *boas* e *perfeitas*. As *modificações aceitáveis* são aquelas que possuem diferença minimal em relação ao estado inicial original baseada na distância de Hamming. As *boas modificações* são modificações aceitáveis tais que o plano solução (que inicia no novo estado inicial e termina em um estado meta) não contém nenhum outro possível estado inicial que seja uma modificação aceitável. As *modificações perfeitas* são aquelas que possuem custo mínimo, considerando o custo para transformar o estado inicial original no novo estado inicial.

Definição 5.2. *(Modificações Aceitáveis do Estado Inicial) Dadas duas modificações* $\mathcal{X}_1 = \langle \mathcal{C}_{\mathcal{X}1}, s_{\mathcal{X}1} \rangle$ *e* $\mathcal{X}_2 = \langle \mathcal{C}_{\mathcal{X}2}, s_{\mathcal{X}2} \rangle$ *para o problema sem solução* $\Pi = \langle \mathbb{A}, \mathcal{C}_{\mathcal{X}}, s_0, \varphi \rangle$*, dizemos que* $\mathcal{X}_1$ *é tão aceitável quando* $\mathcal{X}_2$ *(denotado por* $\mathcal{X}_1 \preceq \mathcal{X}_2$*) se e somente se:*

- $\mathcal{C}_{\mathcal{X}1} \subseteq \mathcal{C}_{\mathcal{X}2}$
- $h(s_0, s_{\mathcal{X}1}) \leq h(s_0, s_{\mathcal{X}2})$

Um elemento mínimo na ordem $\preceq$ *é uma* ***modificação aceitável****.*

Definição 5.3. *(Boas Modificações do Estado Inicial) Seja* $\mathcal{X}_1 = \langle \mathcal{C}_{\mathcal{X}1}, s_{\mathcal{X}1} \rangle$ *uma modificação aceitável para o problema sem solução* $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ *com o plano* Φ *solucionando o problema modificado* $\Pi_1 = \langle \mathbb{A}, \mathcal{C}_{\mathcal{X}1}, s_{\mathcal{X}1}, \varphi \rangle$ *e seja* $\mathcal{X}_2 = \langle \mathcal{C}_{\mathcal{X}2}, s_{\mathcal{X}2} \rangle$ *uma outra modificação para* Π*. A modificação* $\mathcal{X}_2$ *é* ***tão boa quanto*** $\mathcal{X}_1$ *(denotado por* $\mathcal{X}_2 \sqsubseteq \mathcal{X}_1$*) se* $\mathcal{X}_2$ *é uma modificação aceitável também para o problema* $\Pi_{nec} = \langle \mathbb{A}, s_0, nec(s_{\mathcal{X}2}, \Psi, \varphi) \rangle$ *em que* $nec(s_{\mathcal{X}2}, \Psi, \varphi)$ *é uma fórmula que descreve a atribuição de valores necessária para a correta execução de* Ψ *iniciando em* s_2

e levando para um estado que satisfaz φ. Dizemos que $\mathcal{X}_2$ é melhor que $\mathcal{X}_1$ (denotada por $\mathcal{X}_2 \sqsubset \mathcal{X}_1$) se e somente se $\mathcal{X}_2 \sqsubseteq \mathcal{X}_1$ e $\mathcal{X}_1 \not\sqsubseteq \mathcal{X}_2$. Finalmente, $\mathcal{X}$ é dita ser uma ***boa modificação*** *se não há uma outra modificação $\mathcal{X}'$ tal que $\mathcal{X}' \sqsubset \mathcal{X}$ ou se existe uma modificação $\mathcal{X}''$ tal que $\mathcal{X} \sqsubseteq \mathcal{X}''$ e $\mathcal{X}'' \sqsubseteq \mathcal{X}$.*

Definição 5.4. *(Modificações Perfeitas do Estado Inicial) Sejam $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ um problema de planejamento sem solução, $\mathcal{X} = \langle C_{\mathcal{X}}, s_{\mathcal{X}} \rangle$ uma modificação para Π e $c(s_0, s_{\mathcal{X}})$ a função que determina o custo de transformar o estado inicial s_0 no estado inicial modificado $s_{\mathcal{X}}$, definida como a seguir:*

$$c(s_0, s_{\mathcal{X}}) = \sum_{p \in s_0 \setminus s_{\mathcal{X}}} c(p)$$

, em que cada proposição p possui um custo $c(p)$ associado. Boas modificações com custo mínimo são ***modificações perfeitas****. A função custo respeita a ordem de aceitabilidade $\preceq$, isto é, se $\mathcal{X}_1 \preceq \mathcal{X}_2$ então $c(s_0, \mathcal{X}_1) \preceq c(s_0, \mathcal{X}_2)$. .*

5.1.3 Gerando Explicações para Problemas de Planejamento sem Solução com Variáveis Multivaloradas

Formalmente, dado um problema de planejamento $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$, Göbelbecker et. al., (2010) define que uma *explicação* para o fato de que o problema não tem solução é uma tupla $\mathcal{X} = \langle C_{\mathcal{X}}, s_{\mathcal{X}} \rangle$ tal que o problema modificado $\Pi_{\mathcal{X}} = \langle \mathbb{A}, C_{\mathcal{X}}, s_{\mathcal{X}}, \varphi \rangle$ possua solução.

Um dos passos fundamentais deste método é a identificação dos valores das variáveis que são importantes para o alcance da meta e que não estão sendo atingidos pelas ações do domínio de planejamento. Assim, são criadas *ações fictícias* responsáveis por atribuir a estas variáveis os valores importantes (relevantes) para o alcance da meta. As ações fictícias são introduzidas no conjunto de ações que o planejador pode utilizar e, desta forma, é possível encontrar uma solução para o problema de planejamento que antes não era solucionável. As *ações fictícias* podem ser de dois tipos:

- ações que adicionam novos objetos ao problema (chamadas ações *adiciona-objeto*);
- ações que modificam o valor de certas variáveis para que possam assumir valores relevantes para o alcance da meta (chamadas ações *muda-valor*).

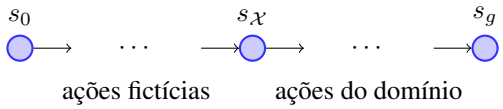


Figura 5.2: A introdução de ações fictícias no problema leva a um estado $s_{\mathcal{X}}$ a partir do qual o problema de planejamento possui solução.

A Figura 5.2 ilustra que, somente utilizando as ações reais do domínio, não é possível partir do estado inicial s_0 e alcançar o estado meta s_g. Porém, com o uso das ações fictícias é possível alcançar um estado $s_{\mathcal{X}}$, a partir do qual (apenas com as ações do reais do domínio) é possível chegar no estado meta s_g. Um detalhe importante é que as ações fictícias são utilizadas apenas no prefixo do plano de forma que o plano solução é formado por: ações fictícias + ações reais do domínio. Assim, o primeiro estado no plano solução em que uma ação real do domínio é usada é um estado explicação.

Como mostrado na Figura 5.2, para encontrar o estado explicação $s_{\mathcal{X}}$ deve-se considerar planos que contenham apenas ações fictícias em seu prefixo e, a partir do estado $s_{\mathcal{X}}$, apenas ações reais do domínio. Para obter planos dessa natureza, [Göbelbecker et al., 2010] utiliza uma variável denominada *iniciada*: $\neg$*iniciada* é introduzida no estado inicial e nas pré-condições de cada ação fictícia e *iniciada* é introduzida nos efeitos de cada ação do domínio. Assim, quando a primeira ação real do domínio for executada *started* passa a ser verdadeira, impedindo a aplicação de ações ficitícias em qualquer estado posterior a $s_{\mathcal{X}}$.

A adição de objetos é feita da seguinte maneira. Inicialmente, um número n de *objetos sobressalentes* $sp_1^t, \cdots, sp_n^t$ é adicionado ao conjunto de objetos do problema para cada novo objeto do tipo t que se deseja adicionar. Estes objetos, no entanto, encontram-se desabilitados no estado inicial e só serão ativados, caso necessário. Para controlar a habilitação e desabilitação dos objetos sobressalentes, é introduzida uma variável *não-usado*(sp_i^t) para cada objeto sobressalente sp_i^t. O estado inicial contém *não-usado*(sp_i^t) para cada um dos objetos sobressalentes (indicando que tais objetos estão inicialmente desabilitados) e, para evitar o uso de objetos desabilitados, $\neg$*não-usado*(sp_i^t) é introduzida nas pré-condições de cada ação do domínio que não envolve o objeto sp_i^t. A ação *adiciona-objeto*$^t(p)$, responsável pela real adição dos objetos sobressalentes, é definida da seguinte forma:

- *precond(adiciona-objeto*$^t(p)) = \{$*não-usado*$(p), \neg$*iniciado*$\}$;

- *efeitos(adiciona-objeto*$^t(p)) = \{\neg$*não-usado*$(p)\}$.

Para saber os valores relevantes das variáveis e identificar aqueles que não estão sendo alcançados pelas ações do domínio de planejamento é realizada uma análise em dois tipos de estruturas que descrevem o comportamento das variáveis em um domínio de planejamento: o *grafo causal* e os *grafos de transição de domínios* [Helmert, 2006].

O grafo causal (V, E) é uma estrutura que captura as dependências causais entre as variáveis do domínio de planejamento. Cada vértice $v \in V$ representa uma variável do domínio e cada aresta $(u, v) \in E$ representa uma dependência causal entre os valores das variáveis u e v. Uma aresta $(u, v) \in E$ se e somente se $u \neq v$ e existe $a \in \mathbb{A}$ tal que $u \in$ *precond*(a) e $v \in$ *efeitos*(a) ou, ambos, u e $v \in$ *efeitos*(a). Dizemos que u é causa de v, ou ainda, que v depende de u.

Definição 5.5. *(Dependência causal entre as variáveis) Dado o grafo causal* (V, E)*, o conjunto das variáveis* u *que são dependentes de* v *é dado por:*

$$\textit{dependentes}(v) = \{u \mid \exists (u, v) \in E\}.$$

emas de Planejamento sem Solução O grafo de transição de domínio (V, E) [Helmert, 2006] de uma variável do domínio de planejamento representa as possíveis maneiras que esta variável pode modificar seu valor e as condições necessárias para que esta modificação ocorra. Sejam v uma variável do domínio de planejamento e *dom*(v) o conjunto de valores que esta variável pode assumir. Cada vértice $x \in V$ representa um valor possível que v pode assumir, i.e., existe um vértice para cada elemento do conjunto dom(v). Uma aresta $(x, x') \in E$ se e somente se existe uma ação $a \in \mathbb{A}$ que modifica o valor de v de x para x', i.e., $v = x \in$ *precond*(a) e $v = x' \in$ *efeitos*(a).

No entanto, dentre todos os valores possíveis que uma variável v pode assumir, há aqueles que são relevantes para o agente alcançar sua meta. O conjunto desses valores é chamado *domínio relevante* de uma variável e pode ser calculado de acordo com a Definição 5.6. Nesta definição, considera-se que uma variável v *contribui para a meta* se uma atribuição de valor $v := x$ em um estado s, torna s um estado meta.

Definição 5.6. *(Cálculo do domínio relevante) O domínio relevante de uma variável* $v \in V$*, denotado por* $\textit{dom}_{rel}(v)$*, pode ser calculado como a seguir, até que um ponto-fixo seja encontrado [Göbelbecker et al., 2010]:*

- *Se* $v := x$ *contribui para a meta, então* $x \in$ *dom*$_{rel}(v)$*;*

- *Para cada variável* $u \in$ *dependentes*(v)*,* $dom_{rel}(u)$ *contém o subconjunto de* $dom(u)$ *que é (potencialmente) necessário para alcançar algum elemento de* $dom_{rel}(v)$*.*

Um ponto fixo é alcançado quando, em uma iteração do procedimento, o domínio relevante de cada variável é o mesmo que o computado na iteração anterior. Este procedimento atravessa o grafo causal a fim de descobrir *quais variáveis e seus valores são relevantes* para o alcance da meta. No entanto, para descobrir *se os valores relevantes estão sendo alcançados pelas ações* reais do domínio é necessário fazer uma análise no grafo de transição de domínios. Deve-se verificar então, para cada variável v, os elementos $x \in$ *dom*$_{rel}(v)$ que não são alcançados pelas ações do domínio. Estes são armazenados em um conjunto denominado *não-alcançados*(v). Assim, para cada variável v e $x \in$ *não-alcançados*(v), uma ação fictícia *muda-valor*$_x^v$ é introduzida.

Depois de gerar todas as ações ficitícias relevantes para resolver o problema sem solução $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$, é necessário utilizar um planejador (neste caso foi utilizado o planejador *fast downward* [Helmert, 2006]) que devolverá um plano Ψ (o qual pode ser dividido em Ψ_{s_0} contendo somente ações fictícias e Ψ_Π contendo apenas ações do domínio) para o problema de planejamento modificado $\Pi_\mathcal{X} = \langle \mathbb{A}, C_\mathcal{X}, s_\mathcal{X}, \varphi \rangle$. Assim, a explicação $\mathcal{X} = \langle C_\mathcal{X}, s_\mathcal{X} \rangle$ é então extraída do plano solução Ψ da seguinte forma:

- $C_\mathcal{X} = C_\Pi \cup \{\text{adiciona-objeto}^t(c) \in \Psi\}$

- $s_\mathcal{X}$ é o estado resultante da execução de Ψ_{s0} restrito às variáveis definidas no problema original Π.

As explicações encontradas pelo método de [Göbelbecker et al., 2010] são explicações aceitáveis, uma vez que o planejador ótimo, ao resolver o problema modificado, escolhe um plano com o menor número de ações possíveis para sair do estado inicial, chegar em um estado explicação e alcançar um estado meta. Tais explicações também são boas, pois o planejador ótimo garante que não existe outro plano possível contido no plano devolvido.

5.2 Explicações para Problemas sem Solução com Variáveis Proposicionais

Inspirado na abordagem de [Göbelbecker et al., 2010], propomos um método para encontrar boas explicações para problemas de planejamento sem solução [Menezes et al., 2012]. Este método, no entanto, pode ser aplicado a domínios de planejamento apenas com *variáveis proposicionais* (i.e., $dom(v) = \{\bot, \top\}$ para toda variável v do domínio de planejamento) diferente do método descrito na seção anterior que trata domínios com variáveis multivaloradas. O método proposto em [Menezes et al., 2012] transforma as ações do domínio de planejamento em fórmulas da lógica proposicional e através da manipulação dessas fórmulas (utilizando operações de restrição e quantificação sobre os valores das proposições) é possível extrair o domínio relevante das variáveis e determinar que ações ficitícias devem ser introduzidas para que o problema de planejamento sem solução passe a ter uma solução. Desta maneira, não é necessário a construção do grafo causal e nem dos grafos de transição de domínios no processo de criação das ações fictícias, como é feito no método proposto por [Göbelbecker et al., 2010].

Para que cada ação $a \in \mathbb{A}$ do domínio de planejamento seja transformada em uma fórmula da lógica proposicional, é necessário introduzir uma cópia do conjunto de proposições $\mathbb{P}$ para que seja possível representar os valores das proposições antes e depois da execução da ação. A cópia do conjunto $\mathbb{P}$ é o conjunto $\mathbb{P}'$ tal que para cada $p \in \mathbb{P}$ há um $p' \in \mathbb{P}'$ correspondente. As proposições em $\mathbb{P}$ representam o valor da variável *antes da execução de uma ação* e as proposições em $\mathbb{P}'$ representam o valor da variável *após a execução da ação*. Desta forma, nessa representação as variáveis p ocorrerem nas precondições das ações e as variáveis p', nos efeitos. A Definição 5.7 mostra como representar cada ação $a \in \mathbb{A}$ como uma fórmula da lógica proposicional.

Definição 5.7. *(Ação como fórmula proposicional) Uma ação $\alpha = \langle precond(\alpha), efeitos(\alpha)\rangle$ pode ser representada como uma fórmula da lógica proposicional da seguinte forma:*

$$\varphi_\alpha = \bigwedge_{p \in precond(\alpha)} p \quad \wedge \quad \bigwedge_{p' \in efeitos(\alpha)} p'.$$

O *domínio relevante* de cada proposição $p \in \mathbb{P}$ é, então, extraído a partir da representação proposicional das ações. A ideia fundamental para este cálculo é que para cada ação que contém uma atribuição $p := x$ nos efeitos relevantes ao alcance da meta, todas as outras atribuições das

variáveis envolvidas nas pré-condições e efeitos da ação também são relevantes para o alcance da meta (exceto a atribuição do valor de p na pré-condição da ação uma vez que o grafo causal não admite arestas de uma proposição para ela mesma). Assim, dada a fórmula φ_α representando uma ação $\alpha \in \mathbb{A}$ e uma valoração relevante $p := x$, a restrição $\varphi_\alpha[p'\backslash x]$ é capaz de nos informar se a ação α alçança tal submeta. Comparemos o tamanho da fórmula original da ação com a fórmula restrita:

- (i) se $|\varphi_\alpha[p'\backslash x]| = |\varphi_\alpha|$ então a restrição da proposição p' não modificou a fórmula φ_α, significando que a ação não possui a proposição p em seus efeitos;
- (ii) se $|\varphi_\alpha[p'\backslash x]| = 0$ então a restrição de p' modificou a fórmula φ_α colocando no lugar de p' um valor x inverso ao que estava presente no efeito da ação, resultando em $\varphi_\alpha[p'\backslash x] = \bot$;
- (iii) se $|\varphi_\alpha[p'\backslash x]| = |\varphi_\alpha[p'\backslash x]| - 1$ então a restrição modificou a fórmula colocando no lugar da proposição p' o valor x igual ao valor presente no efeito da ação, i.e., a ação possui o valor relevante em seus efeitos.

Além disso, como o grafo causal não possui arestas do tipo (p, p), então sempre que uma ação alcançar um valor relevante $p := x$, a variável p deve ser relaxada (com a quantificação existencial sobre os valores de p) na fórmula da ação φ_α. A Definição 5.8 mostra como computar o dominio relevante das variáveis $p \in \mathbb{P}$ segundo uma ação $\alpha \in \mathbb{A}$ (denotado por $\text{dom}_{rel}(p_i)_\alpha$) por meio de manipulação da fórmula φ_α.

Definição 5.8. *(Obtendo o domínio relevante segundo uma ação) Seja $p := x$ uma variável que ocorre nos efeitos de uma ação $\alpha = \langle precond(\alpha), efeitos(\alpha)\rangle$ e seja φ_α a representação proposicional de α. A fórmula*

$$\exists p.\varphi_\alpha[p'\backslash x] = \bigwedge(p_i := x_i)$$

resgata o valor das pré-condições e efeitos da ação α (excluindo o próprio valor de p), sendo:

$$dom_{rel}(p_i)_\alpha = \{x_i\}.$$

Para obter o conjunto completo (i.e., referente ao conjunto de ações $\mathbb{A}$) do domínio relevante de uma variável $p \in \mathbb{P}$ é necessário fazer a união dos domínios relevantes de p segundo cada ação $\alpha \in \mathbb{A}$.

Definição 5.9. *(Obtendo o domínio relevante de uma variável p) Seja p uma variável do domínio de planejamento, o domínio relevante de p é dado por:*

$$dom_{rel}(p) = \bigcup_{\alpha \in \mathbb{A}} dom_{rel}(p)_{\alpha}.$$

O Algoritmo 1 computa o domínio relevante das variáveis $p \in \mathbb{P}$ por meio da manipualação das fórmula φ_α de cada ação $\alpha \in \mathbb{A}$. Ele é inicialmente chamado para cada submeta $g_i := x$ do problema de planejamento e recebe como entrada o conjunto das formulas φ_α, uma variável p e um valor $x \in \{\top, \bot\}$. O laço das linhas 1-6 tem a função de determinar quais ações $\alpha \in \mathbb{A}$ possuem $p := x$ em seus efeitos. Esta constatação é feita analisando a restrição da variável p' com valor x na fórmula φ_α (linha 2). Se a ação α possui $p := x$ nos seus efeitos, a restrição $\varphi_\alpha[p \setminus x]$ (linha 3) resulta em uma fórmula $\bigwedge p_i := x_i$, onde todos os valores $p_i := x_i$ estão nas pré-condições ou efeitos de α e por isto são *relevantes* para que p torne-se x. O laço das linhas 4-6 iterará sobre os pares $\langle p_i, x_i \rangle$ e, então, o algoritmo será chamado recursivamente para cada um desses valores (linha 6).

Algorithm 1: *obterDomínioRelevante*

Input: Conjunto das fórmulas φ_α, variável $p \in \mathbb{P}$, valor $x \in \{\top, \bot\}$
Output: $dom_{rel}(p')$, onde $p' \in dependentes(p)$
for *cada* φ_α **do**
  if $(\varphi_\alpha[p' \setminus x]) \neq \bot$ *e* $\varphi_\alpha[p' \setminus x] \neq \varphi_\alpha$ **then**
    $\bigcup \langle p_i, x_i \rangle \leftarrow \exists p.\varphi_\alpha[p' \setminus x] = (\bigwedge p_i = x_i)$;
  for *cada par* $\langle p_i, x_i \rangle$ **do**
    $dom_{rel}(p_i) \leftarrow dom_{rel}(p_i) \cup \{x_i\}$;
    obterDomínioRelevante(p_i, x_i);

Sendo p uma variavel proposicional, a garantia de que o valor relevante x de p é sempre alcançado é que existe uma ação que modifica o valor p de $\bar{x}$ para x. Denotamos por $\bar{x}$ o inverso do valor x, isto é, se $x = \bot$ então $\bar{x} = \top$ e se $x = \top$ então $\bar{x} = \bot$. Assim, procuramos alguma ação que possui $p := \bar{x}$ em suas precondições e $p' := x$ em seus efeitos. Esta inspeção pode ser feita analisando o resultado da fórmula restrita $\varphi_\alpha[p \backslash \bar{x}][p' \backslash x]$. Se o tamanho da fórmula restrita contiver duas variáveis a menos que a fórmula original φ_α, então significa que a ação α possui $p := \bar{x}$ em suas precondições e $p' := x$ em seus efeitos. Assim, a restrição $\varphi_\alpha[p \backslash \bar{x}][p' \backslash x]$ é feita para todas as variáveis p, valor $x \in \text{dom}_{rel}(p)$ e ações $\alpha \in \mathbb{A}$. Se existir um valor do domínio relevante de p que não está sendo alcançado, então a ação fictícia *muda-valor*^p_x é criada tal que:

precond(*muda-valor*^p_x) $= \{\bar{x}\}$ e *efeitos*(*muda-valor*^p_x) $= \{x\}$.

O Algoritmo 2 gera o conjunto de ações *muda-valor*. Ele recebe o conjunto das fórmulas φ_α referente a cada ação $\alpha \in \mathbb{A}$; o conjunto de variáveis $\mathbb{P}$ e o domínio relevante de cada $p \in \mathbb{P}$. O laço das linha 1-7 itera sobre cada variável $p \in \mathbb{P}$ e o laço das linhas 2-7 itera sobre os valores x do domínio relevante de p. Em seguida, o laço das linhas 3-7 verifica para cada ação se ela é capaz de transformar o valor da variável p de $\bar{x}$ para x. Esta verificação é feita analisando se a fórmula $\varphi_\alpha[p \backslash \bar{x}][p' \backslash x]$ possui duas proposições a menos que a fórmula φ_α(linha 4). Se a ação α não transforma $p := \bar{x}$ em $p := x$, então é necessário continuar procurando uma ação que o faça. Se nenhuma ação realiza tal transformação (linha 6), é necessário criar a ação *muda-valor* (linha 6). Esta ação é então adicionada ao conjunto das ações fictícias (linha 7) que será devolvido ao final do algoritmo (linha 8).

Algorithm 2: *gerarAçõesFictícias*.

Input: Conjunto das fórmulas φ_α, $\mathbb{P}$ e $dom_{rel}(p)$ para cada $p \in \mathbb{P}$.
Output: Conjunto das ações *muda-valor*[].
for *cada* $p \in \mathbb{P}$ **do**
  for *cada* $x \in dom_{rel}(p)$ **do**
    for *cada* φ_α **do**
      if $|\varphi_\alpha[p \setminus x][p' \setminus \bar{x}]| = |\varphi_\alpha| - 2$ **then**
        break ;
    crie a ação *muda-valor*^p_x, em que *precond(muda-valor*^p_x*)* = $\{\bar{x}\}$ e *efeitos(muda-valor*^p_x*)* = $\{x\}$;
    muda-valor[] $\leftarrow$ *muda-valor*[] $\cup$ *muda-valor*^p_x;
return *muda-valor*[] ;

5.3 Verificação de Modelos para Encontrar Explicações

A abordagem de verificação de modelos pode ser utilizada para encontrar explicações para problemas de planejamento não-determinístico sem solução. Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o algoritmo de verificação de modelos realiza uma busca regressiva a partir do conjunto dos estados X que satisfazem φ e, em cada passo, computa o conjunto dos antecessores de X, armazenando-os em um conjunto dos estados alcançáveis R. O processo termina quando um ponto fixo é encontrado, i.e., quando em uma iteração $R \cup X = R$. Observando que todos os estados $s \in R$ *finalmente alcançam algum estado meta*, o conjunto de

modificações aceitáveis do estado inicial pode ser extraído a partir de R. Nesse caso, um estado s é uma modificação de estado inicial aceitável se $s \in R$ e s possui diferença minimal em relação ao estado inicial s_0. Os estados s que são boas modificações para o problema Π são os estados aceitáveis a partir dos quais não há outro estado aceitável s' no caminho que inicia em s e termina em um estado satisfazendo a meta.

Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o algoritmo 3 retorna o conjunto de estados que são boas explicações para o problema Π. Neste algoritmo, os conjuntos X e R armazenam os estados alcançados pela busca regressiva a partir do conjunto de estados que satisfaz a fórmula meta φ, os quais são obtidos na linha 2 pela função $\textsc{Sat}(\varphi)$. O laço das linhas 3-7 itera sobre o conjunto de estados alcançáveis pela regressão do conjuntos de estados X de acordo com o conjunto de ações $\mathbb{A}$. Estes são armazenados no conjunto de estados R até que nenhum novo estado seja encontrado pela busca regressiva, isto é, quando $X = R$. De posse do conjunto de estados que finalmente alcançam a meta, chama-se na linha 6 a função capaz de obter o conjunto de estados em R que possuem distância minimal em relação ao estado inicial s_0. Finalmente, de posse do conjuntos dos estados que são explicações aceitáveis, é possível obter o conjunto das boas explicações na linha 7, as quais são devolvidas pelo algoritmo.

Algorithm 3: VM-EXPLICAÇÕES($\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$)

$X \leftarrow \emptyset$;
$R \leftarrow \textsc{Sat}(\varphi)$;
while $X \neq R$ **do**
  $X \leftarrow R$;
  $R \leftarrow R \cup$ *regressão*$(X, \mathbb{A})$;
ExplicaçõesAceitáveis $\leftarrow$ *obterExplicaçõesAceitáveis*(Π, R);
BoasExplicações $\leftarrow$ *obterBoasExplicações*(Π, *ExplicaçõesAceitáveis*);
return *BoasExplicações*

O Algoritmo 4 recebe o problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ e o conjunto R dos estados que finalmente alcançam a meta φ, devolvendo o conjunto de estados que são explicações aceitáveis para Π. O algoritmo itera sobre o número n de proposições $\mathbb{P}$ do problema de planejamento e, em cada passo, gera o conjunto dos estados com diferença i em relação a s_0. Por exemplo, seja $\mathbb{P} = \{p, q\}$ e $s_0 = \{p, q\}$, os estados $s_1 = \{p\}$ e $s_2 = \{q\}$ são estados com diferença 1 em relação a s_0 e obtidos pela chamada a função *gerarEstadosComDiferençaI*$(s_0, 1)$. Assim, na linha 3 do algoritmo é verificado se há intersecção entre o conjunto de estados com diferença i em relação a s_0 e o conjunto de estados que finalmente alcançam a meta. Se há interesecção,

o algoritmo termina devolvendo o resultado desta intersecção que são as explicações aceitáveis para Π.

Algorithm 4: OBTEREXPLICAÇÕESACEITÁVEIS(Π, R)

```
for (i = 1; i ≤ n; i = i + 1) do
    Y ← gerarEstadosComDiferençaI(s_0, i);
    if Y ∩ R ≠ ∅ then
        ExplicaçõesAceitáveis ← Y ∩ R;
        return ExplicaçõesAceitáveis ;
```

O Algoritmo 5 recebe o problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ e o conjunto das explicações aceitáveis para Π, devolvendo o conjunto de estados que são boas explicações. O algoritmo itera sobre cada estado explicação aceitável s para verificar se existe outro estado s' que também é uma explicação aceitável para Π e está em algum caminho entre o estado s e um estado satisfazendo a meta. Isto é verificado obtendo-se o conjunto dos estados Y que satisfazem a fórmula temporal $\exists \circ \exists \diamond \varphi$ a partir do estado s. Se há intersecção entre o conjunto Y e o conjunto dos estados aceitáveis, então s não é uma boa explicação para Π. Se não há intersecção, então s é uma boa explicação para Π.

Algorithm 5: OBTERBOASEXPLICAÇÕES$(\Pi, \textit{ExplicaçõesAceitáveis})$

```
for cada s ∈ Aceitáveis do
    Y ← MC(A, s, ∃ ∘ ∃ ◇ φ);
    if Aceitáveis ∩ Y ≠ ∅ then
        BoasExplicações ← BoasExplicações ∪ s;
return BoasExplicações ;
```

Capítulo 6

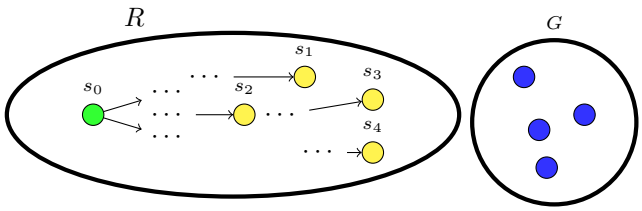
Modificação da Meta de Planejamento

Neste capítulo, dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, propomos uma abordagem para encontrar uma nova meta de planejamento φ' tal que o problema modificado $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$ seja solucionável. Denominamos G o conjunto dos estados que satisfazem φ e G' o conjunto dos estados que satisfazem G'. A ideia é utilizar a abordagem de verificação de modelos para obter o conjunto R de estados alcançáveis a partir do estado inicial s_0 e selecionar um subconjunto de R que satisfaça uma meta φ' que seja o mais próximo possível a meta original φ, segundo uma determinada métrica.

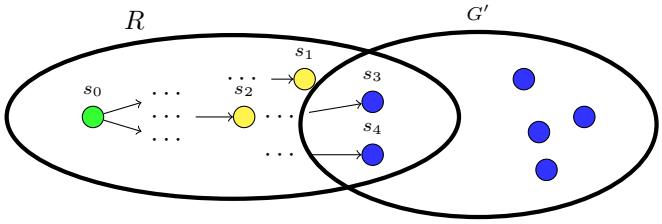
A Figura 6.1 ilustra o processo de modificação da meta de planejamento para um problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$. Na Figura 6.1(a), o problema Π não possui solução, pois $R \cap G = \emptyset$. Na Figura 6.1(b), a meta φ é modificada, obtendo uma nova meta φ' de forma que $R \cap G' \neq \emptyset$.

Para obter o conjunto R implementamos o verificador de modelos que raciocina progressivamente sobre o modelo implícito e simbólico do domínio de planejamento. Assim, dado um problema de planejamento $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o verificador de modelos realiza uma busca progressiva a partir do estado inicial s_0 e, computando o conjunto X de estados sucessores de s_0. Em cada passo, a progressão simbólica computa o conjunto dos sucessores de X e os adiciona ao conjunto R. A busca termina quando um ponto-fixo é encontrado, i.e., quando em uma certa iteração i temos que $R_i \cup X = R_{i-1}$.

Qualquer estado $s \in R$ é um candidato a pertencer ao novo conjunto de estado meta do



(a) Problema sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$.



(b) Problema solucionável $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$.

Figura 6.1: Modificação da meta de planejamento para o problema Π. Em (a) Π não é solucionável pois $R \cap G = \emptyset$. Em (b), a meta é modificada de forma que $R \cap G' \neq \emptyset$ obtendo o problema solucionável Π'.

problema, uma vez que s é alcançável a partir do estado inicial s_0. No entanto, a escolha da nova meta φ' pode ser baseada na satisfação do maior número possível de proposições ou ainda em preferências na satisfação das proposições. Considere, por exemplo, um problema sem solução no domínio de Logística no qual a meta é entregar os pacotes p_1, p_2, p_3, p_4 e p_5. Podemos estar interessados em uma nova meta que maximize o número de pacotes, i.e., que tenha uma distância minimal em relação a φ segundo o número de proposições (distância de Hamming). Também é possível desejar obter uma meta que atenda certas preferências estabelecidas como, por exemplo, é preferível entregar o pacote p_1 do que o pacote p_4. Desta forma, qualquer modificação da meta obtida que atenda a entrega do pacote p_1 é pereferível do que uma meta que não atenda a entrega deste pacote. Desta forma, as métricas $M5$, $M61$ e $M62$ de escolha da melhor modificação da meta são baseadas na maximização do número de proposições atendidas enquanto que as métricas $M7$, $M81$ e $M82$ são baseadas em preferências. Vale ressaltar também que o processo de modificação da meta de planejamento é um processo de revisão do conjunto G que satisfaz φ pelo conjunto R de estados alcançados a partir do estado inicial s_0.

Para obter o conjunto dos estados G'_{M5} de acordo com a métrica $M5$ é necessário realizar a *revisão* do conjunto de estados meta original G pelo conjunto de estados alcançáveis R. Para

obter o conjunto de estados G'_{M61} que são minimais em relação a distância de Hamming e possuem a maior distância possível do estado inicial é necessário selecionar os estados pertencentes a G'_{M5} que foram obtidos o mais tardiamente possível durante a progressão a partir do estado inicial. Similarmente, para obter o conjunto de estados G'_{M62} que são minimais em relação a distância de Hamming e possuem a menor distância possível do estado inicial é necessário selecionar os estados pertencentes a G'_{M5} que foram obtidos o mais precocemente possível durante a progressão a partir do estado inicial. Para obter o conjunto de estados G'_{M7} que são minimais em relação a métrica baseada em preferências é necessário selecionar os estados em R que são minimais em relação a ordem parcial de preferência estabelecida por uma rede de preferências dada. Para os obter os conjuntos G'_{M81} (G'_{M82}), que são minimais em relação a métrica que combina preferências e maximização (minimização) da distância para o estado inicial s_0, é necessário selecionar o subconjunto de G'_{M7} que foi obtido o mais tarde (precoce) posssível na progressão a partir de s_0.

Dado um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ e uma métrica μ de seleção de modificações minimais, o Algoritmo 6 retorna o conjunto G' de estados que satisfazem a fórmula φ' tal que: (i) G' é formado pelos estados mais próximos possíveis, segundo a métrica μ, ao conjunto de estados G que satisfaz a meta φ e; (ii) o problema modificado $\Pi' = \langle \mathbb{A}, s_0, \varphi' \rangle$ é solucionável.

Algorithm 6: OBTERMODIFICAÇÕESMETA($\Pi = \langle \mathbb{A}, s_0, \varphi \rangle, \mu$)

```
X ← ∅ ;
R ← s_0 ;
i ← 0 ;
L[i] ← R ;
while X ≠ R do
    X ← R ;
    R ← R ∪ progressão(X, 𝔸) ;
    L[i] ← R \ X ;
    i ← i + 1 ;
G' ← selecionarSegundoMétrica(Π, R, μ);
return G'
```

Neste algoritmo, os conjuntos X e R armazenam os estados alcançados pela busca progressiva a partir do estado inicial s_0. O vetor $L[i]$ armazena os novos estados obtidos em cada iteração de forma que $\bigcup_{0 \leq i \leq n} L[i] = R$. O laço das linhas 5-9 itera sobre o conjunto de estados alcançados pela progressão do conjuntos de estados X de acordo com o conjunto de ações

$\mathbb{A}$. Estes são armazenados no conjunto de estados R até que nenhum novo estado seja encontrado pela busca progressiva, isto é, quando $X = R$. Ao final do laço, o conjunto de todos os estados alcançados a partir de s_0 está armazenado em R. Desta forma, na linha 10, a função *selecionarSegundoMétrica* é responsável por selecionar os estados em R que são minimais em relação a φ segundo uma das possíveis métricas de modificação da meta. O conjunto G' dos estados obtidos é então devolvido pelo algoritmo.

O Algoritmo 7 recebe como entrada o problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o conjunto R de estados alcançáveis pela progressão a partir do estado inicial s_0, o vetor $L[n]$ que contém em cada posição i o conjunto dos estados alcançados na iteração i durante a progressão a partir de s_0 e uma das possíveis métricas μ ($M5, M61, M62, M7, M81, M82$) de modificação da meta de planejamento. *selecionarSegundoMétrica*($\Pi, R, L[n], \mu$) devolve, então, o conjunto de mínimas modificações de meta de planejamento de acordo com a métrica μ. O condicional da linha 1 verifica se a métrica μ é baseada na distância de Hamming (métricas $M5, M61$ e $M62$). Se este for o caso, a variável MODMINHAMMING recebe o conjunto de estados com diferença minimal em relação a φ (linha 2). Na linha 3, verifica-se se a métrica a ser utilizada é a métrica $M5$. Se este for o caso, a variável MODMINHAMMING é devolvida pelo algoritmo na linha 4. Na linha 5 é verificado se μ é a métrica $M61$. Em caso afirmativo, é devolvido na linha 6 o suconjunto dos estados em MODMINHAMMING cuja distância para o estado inicial é maximal. Na linha 7 é verificado se μ é a métrica $M62$. Em caso afirmativo, é devolvido na linha 8 o suconjunto dos estados em MODMINHAMMING cuja distância para o estado inicial é minimal. O condicional da linha 9 verifica se a métrica μ é baseada em preferências (métricas $M7, M81$ e $M82$). Se este for o caso, a rede de preferências do problema Π é obtida na linha 10 e utilizada na linha 11 para obter o conjunto dos estados MODMINPREFERÊNCIAS que satisfazem a fórmula φ' contendo as proposições mais preferíveis de acordo com a rede de preferência. Se μ é a métrica $M7$ então o conjunto de estados MODMINPREFERÊNCIAS é devolvido na linha 12. Na linha 13 é verificado se μ é a métrica $M81$. Em caso afirmativo, é devolvido na linha 14 o suconjunto dos estados em MODMINPREFERÊNCIAS cuja distância para o estado inicial é maximal. Na linha 15 é verificado se μ é a métrica $M82$. Em caso afirmativo, é devolvido na linha 16 o suconjunto dos estados em MODMINPREFERÊNCIAS cuja distância para o estado inicial é minimal.

Algorithm 7: SELECIONARSEGUNDOMÉTRICA($\Pi, R, L[n], \mu$)

```
if μ = M5 || μ = 61 || μ = 62 then
    MODMINHAMMING ← obterModMinHamming(Π, R) ;
    if μ = M5 then
        return MODMINHAMMING ;
    if μ = M61 then
        return obterModMinDistMax(Π, L[n], MODMINHAMMING) ;
    if μ = M62 then
        return obterModMinDistMin(Π, L[n], MODMINHAMMING) ;
if μ = M7 || μ = M81 || μM82 then
    REDEDEPREFERÊNCIAS ← obterRedeDePreferências(Π) ;
    MODMINPREFERÊNCIAS obterModMinPreferências(Π, R, REDEDEPREFERÊNCIAS) ;
    if μ = M7 then
        return MODMINPREFERÊNCIAS ;
    if μ = M81 then
        return obterModMinDistMax(Π, L[n], MODMINPREFERÊNCIAS) ;
    if μ = M82 then
        return obterModMinimaisM2(Π, L[n], MODMINPREFERÊNCIAS) ;
```

O Algoritmo 8 recebe como entrada um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$ e o conjunto Rde estados alcançáveis a partir de s_0. Ele devolve o conjunto $G' \subseteq R$ dos estados que satisfazem a meta φ', a qual possui distância minimal em relação a meta original φ de acordo com a distância de Hamming. Na linha 1, a variável m armazena o número de proposições da meta. Em cada iteração do laço das linhas 2-7, o conjunto de metas que possuem diferença i em relação a φ é gerado e armazenado na variável Y (linha 3). Na linha 4, a variável MODIFICAÇÕESM5 recebe a intersecção do conjunto dos estados R e do conjunto das metas Y. Se no teste da linha 5, este conjunto for diferente de vazio significa que algum estado alcançável a partir de s_0 satisfaz uma das metas com modificação i. Desta forma, a variável MODIFICAÇÕESM5 é devolvida na linha 6. Se no teste da linha 5, MODIFICAÇÕESM5 for vazio uma próxima iteração do laço é executada até que uma modificação minimal da meta seja encontrada.

Algorithm 8: obterModMinHamming(Π, R)

```
m ← |φ| ;
for i = 1; i ≥ m; i ← i + 1 do
    Y ← gerarMetaComDiferençaI(φ, i);
    MODIFICAÇÕESM5 ← Y ∩ R ;
    if MODIFICAÇÕESM5 ≠ ∅ then
        return MODIFICAÇÕESM5;
```

O Algoritmo 9 recebe como entrada um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, o conjunto Rde estados alcançáveis a partir de s_0 e uma rede de preferências sobre as proposições da meta. Ele devolve o conjunto $G' \subseteq R$ dos estados que satisfazem a meta φ', os quais são satisfazem as proposições da meta que são preferíveis de acordo com a rede de preferências. Em cada iteração do laço das linhas 1-5, o conjunto de metas que são preferíveis é gerado e armazenado na variável Y (linha 2). Na linha 3, a variável MODIFICAÇÕESM7 recebe a intersecção do conjunto dos estados R e do conjunto das metas preferíveis Y. Se no teste da linha 4, este conjunto for diferente de vazio significa que algum estado alcançável a partir de s_0 satisfaz uma das metas preferíveis. Desta forma, a variável MODIFICAÇÕESM7 é devolvida na linha 5. Se no teste da linha 4, MODIFICAÇÕESM7 for vazio uma próxima iteração do laço é executada até que uma modificação preferível da meta seja encontrada.

Algorithm 9: obterModMinPreferências(Π, R, REDEDEPREFERÊNCIA)

```
for i = 1; i ≥ k; i ← i + 1 do
    Y ← obterMetasPreferíveis(φ, i, REDEDEPREFERÊNCIA);
    MODIFICAÇÕESM7 ← Y ∩ R ;
    if MODIFICAÇÕESM7 ≠ ∅ then
        return MODIFICAÇÕESM7;
```

Os Algoritmos 10 e 11 recebem como entrada um problema de planejamento sem solução $\Pi = \langle \mathbb{A}, s_0, \varphi \rangle$, um conjunto Y de estados minimais em relação a alguma métrica e o vetor $L[n]$ de estados em que cada $L[i]$ $(1 \leq n)$ é o conjunto dos estados obtidos na iteração i da progressão a partir do estado inicial s_0. O Algoritmo 10 devolve o subconjunto dos estados em Y cuja distância para Y é maximal, realizando uma busca no vetor L, iniciando na posição n, e verificando se algum estado em Y foi alcançado. O Algoritmo 11 devolve o subconjunto dos estados em Y cuja distância para Y é minimal, realizando uma busca no vetor L, iniciando na posição 1, e verificando se algum estado em Y foi alcançado.

Algorithm 10: obterModMinDistMax($\Pi, Y, L[n]$)

```
for i = n; i ≤ 1; i ← i − 1 do
    MODIFICAÇÕESDISTMAX ← Y ∩ L[i] ;
    if MODIFICAÇÕESDISTMAX ≠ ∅ then
        return MODIFICAÇÕESDISTMAX ;
```

Algorithm 11: obterModMinDistMin($\Pi, Y, L[n]$)

```
for i = 1; i ≥ n; i ← i + 1 do
    MODIFICAÇÕESDISTMIN ← Y ∩ L[i] ;
    if MODIFICAÇÕESDISTMIN ≠ ∅ then
        return MODIFICAÇÕESDISTMIN ;
```

Capítulo 7

Experimentos

Para avaliar os métodos propostos dividimos os experimentos em duas fases: (i) verificação da existência do plano e (ii) verificação da não existência de um plano com geração das possíveis modificações do estado inicial, meta e ações para que o problema torne-se solucionável. Para a fase (i) utilizamos como entrada problemas solucionáveis e avaliamos o desempenho das seguintes abordagens:

- progressão e regressão simbólica realizando operações com fórmula booleanas quantificadas (QBF), uma abordagem que denominados QBFPLAN;
- regressão simbólica baseada no cálculo do EPC proposta por [Rintanen, 2008], abordagem que denominamos de EPCPLAN.
- imagem e pré-imagem de um conjunto de estados, implementada por planejadores tais como MIPS [Edelkamp and Helmert, 2001] (para domínios determinísticos) e MBP [Cimatti et al., 2003] (para domínios não determinísticos).
- imagem e pré-imagem de um conjunto de estados baseado no cálculo do EPC como proposto por [Rintanen, 2008]. Esta é a versão explícita e simbólica do planejador EPCPLAN, a qual chamaremos de EPCPLANEXPL.

Note que as abordagens QBFPLAN e EPCPLAN raciocinam sobre um modelo simbólico e explícito do domínio de planejamento. No entanto, o planejador QBFPLAN pode realizar a busca progressiva e regressiva enquanto o planejador EPCPLAN só realiza a busca regressiva. Já as

Grafo de Transição de Estados	*Linguagem de Ações*
Representação Explícita e Simbólica Mips, EpcPlanExpl	Representação Implícita e Simbólica QbfPlan, EpcPlan

Figura 7.1: Abordagens implementadas nesta tese: representaçao explicíta e símbolica versus implícita e simbólica.

abordagens MIPS, MBP e EPCPLANEXPL raciocionam sobre um modelo simbólico e explícito do domínio de planejamento. Verificar a existência de um plano é um problema PSPACE, denominado PLANEX [Ghallab et al., 2004]. A Figura 7.1 mostra as abordagens simbólicas implementadas dividas em representação explícita versus implementação implícita.

Todas as abordagens citadas acima foram implementadas em JAVA, utilizando-se a biblioteca JAVABDD [Whaley, 2010] para representação simbólica dos estados, transições e ações. Definimos como parâmetro para os BDDs um número máximo de nós de 9×10^6 e o tamanho da cache de 9×10^5. Os testes foram executados no servidor *brucutu* do IME-USP.

7.1 Domínios Utilizados na Análise Experimental

Com o intuito de testar as abordagens propostas nesta tese utilizaremos domínios e problemas provenientes da Competição Internacional de Planejamento (IPC, *do inglês International Planning Competition*): um evento bianual, realizado em paralelo com a Conferência Internacional de Planejamento Automatizado e Escalonamento (ICAPS, do inglês *International Conference on Automated Planning and Schedulling*), no qual diversos planejadores são avaliados com o intuito de se definir o estado da arte em planejamento.

Os domínios e problemas determinísticos da IPC utilizados nesta análise experimental são: domínio de logística e domínio do robô de marte [Dimopoulos et al., 2006]. Utilizamos também o domínio das chaves [Göbelbecker et al., 2010] descrito da introdução. A seguir, uma breve descrição destes três domínios:

- **Domínio de Logística** - No domínio de logística há um determinado número de pacotes que devem ser entregues em diversas cidades. Tais pacotes são transportados dentro da mesma cidade por caminhões e entre as cidades por aviões. As diversas instâncias desse

domínio variam o número de objetos a serem entregues, o número de cidades, número de caminhões e aviões.

- **Domínio do Robô de Marte** - No domínio do robô de marte, um determinado número de robôs deve navegar pela superfície de marte, obtendo fotografias e coletando análise de rochas e solo. As diversas instânciass deste domínio variam o número de robôs, o número de locais a serem visitados e o número de objetos a serem coletados e analisados.

- **Domínio das Chaves** - Como descrito na introdução, neste domínio um robô deve navegar entre salas abrindo portas com o uso de chaves. As diversas instâncias desse domínio variam o número de salas que o robô pode visitar, podendo possuir diferentes configurações de disposição das salas.

(a) Domínio do robô de marte. (b) Domínio de logística.

Figura 7.2: Domínios da IPC utilizados na análise experimental.

As ações dos domínios determinísticos utilizados na análise experimental foram modificadas para torná-las não-determinísticas. No domínio de logística modificamos o efeito da ação de carregar um pacote em um caminhão fazendo que após sua execução o pacote esteja dentro do caminhão ou permaneça no local.

A Tabela 7.1 mostra o número de proposições e o número de ações de cada um dos problemas dos domínios de Logística, Robô de Marte e Chaves, utilizados na análise experimental.

Problema	# Proposições	# Ações
Logistics-4-0	48	78
Logistics-6-0	48	78
Logistics-8-0	99	171
Logistics-10-0	168	308
Logistics-12-0	168	308
Logistics-14-0	276	650
Rovers-01	35	63
Rovers-03	44	76
Rovers-04	50	86
Rovers-05	59	144
Rovers-06	63	151
Rovers-07	68	178
Rovers-08	110	328
Keys-3	27	46
Keys-4	64	150
Keys-5	125	112
Keys-6	76	85
Keys-7	93	102
Keys-8	113	133

Tabela 7.1: Número de proposições e número de ações dos problemas utilizados na análise experimental.

7.2 Problemas Sem Solução - Modificação do Estado Inicial

Modificamos o estado inicial dos problemas de planejamento a fim de torná-los não solucionáveis. Desta forma, elaboramos três versões não solucionáveis de cada um dos problemas dos domínios analisados. Para os problemas do domínio de logística modificamos o estado inicial: eliminando todos os caminhões de uma cidade (versão 1), impedindo a entrega de pacotes na cidade, ou todos os aviões do problema, impedindo a entrega de pacotes entre as cidades (versão 2) e; eliminado um pacote que é necessário ser entregue (versão 3). Para os problemas do domínio do robô de marte modificamos o estado inicial: retirando um dos robôs (versão 1); ocupando o canal de comunicação (versão 2), impedindo que os resultados das análises sejam enviados e; retirando uma amostra de rocha ou solo que deve ser analisada (versão 3). Para tonar os problemas do domínio das chaves sem solução modificamos o estado inicial: tornando uma chave necessária inacessível ao robô (versão 1); retirando o robô (versão 2) e; fazendo com que uma porta não possua chave para abrí-la (versão 3).

A Tabela 7.2 mostra a quantidade de modificações minimais encontradas para cada versão

dos problemas sem solução dos domínio analisados, segundo cada métrica M1, M21, M22, M3, M3, M41 e M42. Na coluna das métricas M21, M22, M41 e M42 o número dentro do parenteses é referente a camada em que as modificações foram encontradas.

As Figuras 7.3, 7.4 e 7.5 mostram os gráficos do tempo de execução gasto pelas abordagens PROPLAN, EPCPLAN e MIPS de verificação da existência de um plano e geração de um conjunto de boas modificações do estado inicial, respectivamente, para os problemas não solucionáveis dos domínios de Logística, Robô de Marte e Chaves.

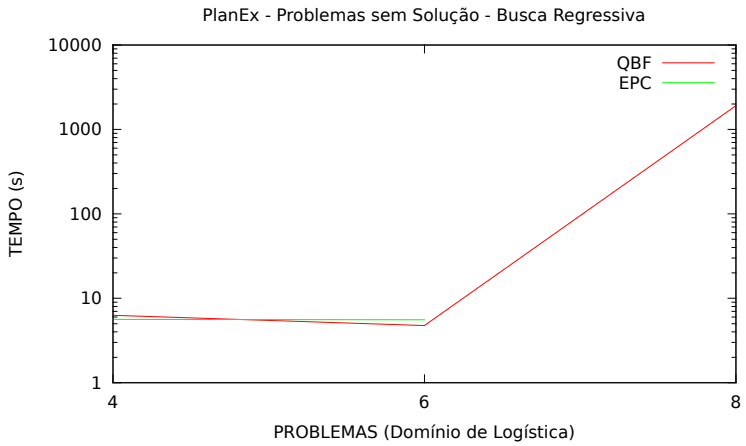


Figura 7.3: Tempo de execução gasto para verificar a não existência de um plano e sugestão de boas modificações do estado inicial para os problemas sem solução do domínio de logística.

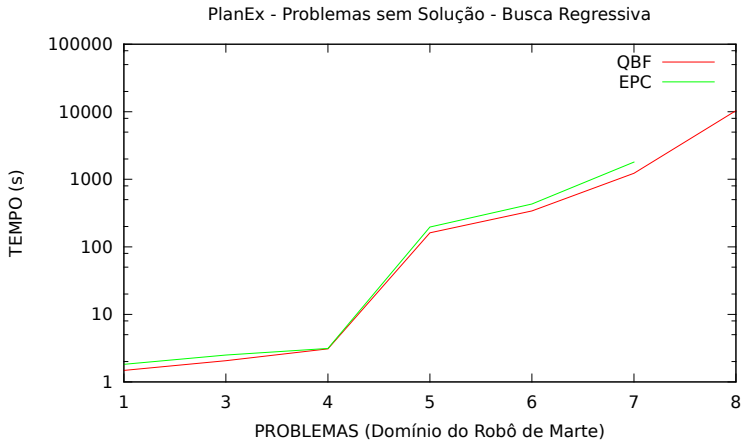


Figura 7.4: Tempo de execução gasto para verificar a não existência de um plano e sugestão de boas modificações do estado inicial para os problemas sem solução do domínio do robô de marte.

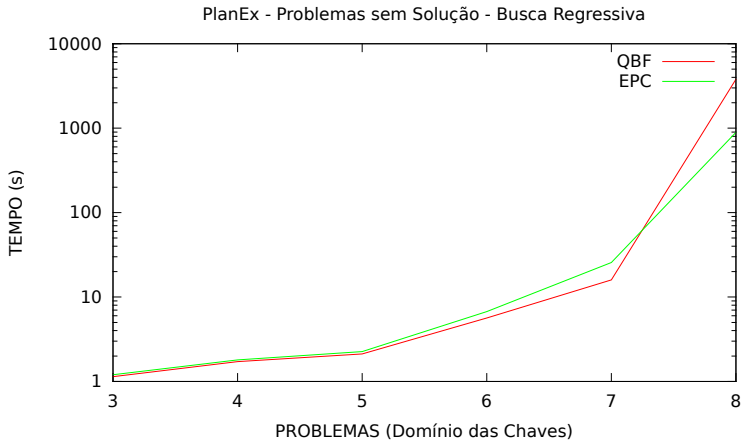


Figura 7.5: Tempo de execução gasto para verificar a não existência de um plano e sugestão de boas modificações do estado inicial para os problemas sem solução do domínio das chaves.

Problema	M1	M21	M22
keys3-v1	2	1(3)	1(3)
keys3-v2	2	1(3)	1(3)
keys3-v3	6	2(3)	1(1)
keys4-v1	2	1(4)	1(4)
keys4-v2	2	1(6)	1(6)
keys4-v3	3	1(6)	1(5)
keys5-v1	1	1(11)	1(11)
keys5-v2	1	1(11)	1(11)
keys5-v3	1	1(11)	1(11)
keys6-v1	1	1(16)	1(16)
keys6-v2	2	1(16)	1(16)
keys6-v3	2	1(16)	1(16)
keys7-v1	6	3(23)	1(21)
keys7-v2	2	1(18)	1(18)
keys7-v3	2	1(18)	1(18)
keys8-v1	2	1(11)	1(11)
keys8-v2	5	1(11)	1(0)
keys8-v3	2	1(11)	1(11)
rovers1-v1	4	1(13)	1(10)
rovers1-v2	1	1(9)	1(9)
rovers1-v3	1	1(8)	1(8)
rovers3-v1	3	1(11)	1(9)
rovers3-v2	1	1(11)	1(11)
rovers3-v3	2	1(11)	1(9)
rovers4-v1	2	1(8)	1(7)
rovers4-v2	1	1(7)	1(7)
rovers4-v3	1	1(7)	1(7)
rovers5-v1	3	1(22)	2(21)
rovers5-v2	1	1(22)	1(22)
rovers5-v3	2	1(22)	1(20)
rovers6-v1	6	1(33)	1(30)
rovers6-v2	1	1(32)	1(32)
rovers6-v3	2	1(32)	1(28)
rovers7-v1	6	3(23)	1(21)
rovers7-v2	1	1(22)	1(22)
rovers7-v3	2	1(22)	1(17)
logistics-4-0v1	2	1(22)	1(21)
logistics-4-0v2	2	1(22)	1(21)
logistics-4-0v3	7	1(22)	1(16)
logistics-6-0v1	2	1(27)	1(26)
logistics-6-0v2	2	1(26)	1(25)
logistics-6-0v3	7	1(26)	1(22)
logistics-8-0v1	2	1(25)	1(24)
logistics-8-0v2	3	2(26)	1(25)
logistics-8-0v3	10	10(25)	10(25)
logistics-10-0v1	2	1(46)	1(45)
logistics-10-0v2	4	2(46)	2(45)
logistics-10-0v3	13	3(45)	1(41)

Tabela 7.2: Quantidade de modificações minimais do estado inicial encontradas de acordo com as métricas M1, M21, M22, M3, M41 e M51 para cada versão dos problemas não solucionáveis dos domínios de logística, robô de marte e chaves.

Capítulo 8

Conclusões

A análise de problemas de planejamento sem solução é uma abordagem importante na área de engenharia de conhecimento para planejamento, i.e., problemas sem solução podem ser usados por um projetista durante o processo de desenvolvimento de um novo domínio. Problemas sem solução podem ter três possíveis causas: (i) o estado inicial foi especificado incorretamente e, assim, não é possível a partir dele alcançar um estado meta; (ii) a meta de planejamento foi super especificada, i.e., a meta envolve tantos objetivos a serem atingidos que é impossível atender a todos e; (iii) as ações não foram especificadas corretamente, i.e., pode ocorrer algum erro na especificação das pré-condições ou efeitos das ações.

Assim, dado um problema de planejamento $\Pi = \langle \mathbb{A}, C, s_0, \varphi \rangle$ sem solução é possível modificá-lo para que ele torne-se solucionável. As possíveis modificações para um problema de planejamento são: *modificação do estado inicial* s_0 [Göbelbecker et al., 2010, Menezes et al., 2012, Menezes and Barros, 2013, Herzig et al., 2014]; *modificação da meta de planejamaneto* φ [Edelkamp and Kissmann, 2009, Benton et al., 2007, Rabideau et al., 2008, Herzig et al., 2014] e; *modificação do conjunto de ações* $\mathbb{A}$ [Menezes et al., 2010, Menezes and Barros, 2011b, Menezes et al., 2012, Herzig et al., 2014].

Nesta tese, propomos e implementamos um método baseado em verificação de modelos capaz de receber um problema de planejamento sem solução e sugerir modificações minimais do estado inicial, meta e ações para que tal problema torne-se solucionável. O método proposto raciocina sobre uma representação implícita e simbólica de domínios de planejamento determinísticos e não determinísticos. Para ser possivel o raciocínio implícito e simbólico sobre

ações determinísticas, propomos uma forma de realizar a progressão e regressão de ações não-determinísticas como uma extensão do método de raciocínio implícito e simbólico de ações determinísticas [Fourman, 2000]. Além disso, propomos e implementamos um método que é capaz de analisar fórmulas proposicionais representando ações determinísticas de um domínio de planejamento e sugerir modificações do estado inicial para um problema sem solução. Este método é baseado no proposto por [Göbelbecker et al., 2010], no entanto, é capaz de sugerir um novo estado inicial analisando apenas as ações do domínios sem a necessidade de construção de estruturas auxiliares (como o grafos causal e grafos de transição de domínios). Para implementação dos algoritmos foram utilizados Diagramas de Decisão Binária, o que tornou possível a representação de todas as instâncias dos problemas avaliados. As principais contribuições dessa tese foram:

- formulação original da progressão e regressão simbólica de ações não determinísticas com o raciocínio sobre a representação implícita e simbólica do domínio de planejamento;
- implementação da versão simbólica do verificador de modelos α-CTL que pode receber como entrada tanto a representação explícita como a representação implícita do domínio de planejamento, estendendo assim a versão explícita e enumerativa proposta por [Pereira and Barros, 2008];
- comparativo das diferentes abordagens de progressão e regressão simbólicas versus as abordagens de imagem e pré-imagem de um conjunto de ações.
- definição e implementação do método de sugerir modificações para o estado inicial de problemas de planejamento determinístico inspirado na abordagem proposta por [Göbelbecker et al., 2010];
- definição e implementação do método de sugerir modificações para o estado inicial, meta e ações baseado em verificação simbólica de modelos.
- definição e implementação do método para modificação das ações baseado em um conjunto de problemas teste.

Referências Bibliográficas

[Alcázar et al., 2013] Alcázar, V., Borrajo, D., Fernández, S., and Fuentetaja, R. (2013). Revisiting regression in planning. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2254–2260. AAAI Press.

[Alchourron et al., 1985] Alchourron, C., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change. *Journal of symbolic logic*, 50:510–530.

[Andersen, 1997] Andersen, H. (1997). An introduction to binary decision diagrams. *Lecture notes, available online, IT University of Copenhagen*.

[Bacchus, 2001] Bacchus, F. (2001). AIPS 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems. *Ai magazine*, 22(3):47.

[Bäckström and Nebel, 1995] Bäckström, C. and Nebel, B. (1995). Complexity results for sas+ planning. *Computational Intelligence*, 11(4):625–655.

[Balbiani et al., 2013] Balbiani, P., Herzig, A., and Troquard, N. (2013). Dynamic logic of propositional assignments: a well-behaved variant of PDL. In Kupferman, O., editor, *Logic in Computer Science (LICS), New Orleans, June 25-28, 2013*, http://www.ieee.org/. IEEE.

[Barták and McCluskey, 2007] Barták, R. and McCluskey, L. (2007). Introduction to the special issue on knowledge engineering tools and techniques for automated planning and scheduling systems. *The Knowledge Engineering Review*, 22(02):115–116.

[Benton et al., 2007] Benton, J., van den Briel, M., and Kambhampati, S. (2007). Finding admissible bounds for over-subscription planning problems. In *Proceedings o f ICAPS'07 Workshop on Heuristics for Domain independent Planning: Progress, Ideas, Limitations, Challenges*.

[Brafman and Chernyavsky, 2005] Brafman, R. I. and Chernyavsky, Y. (2005). Planning with goal preferences and constraints. In *ICAPS*, pages 182–191.

[Buning et al., 1995] Buning, H. K., Karpinski, M., and Flogel, A. (1995). Resolution for quantified boolean formulas. *Information and computation*, 117(1):12–18.

[Cimatti et al., 2003] Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2003). Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1):35–84.

[Clarke and Emerson, 1982] Clarke, E. M. and Emerson, E. A. (1982). Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, London, UK. Springer-Verlag.

[Dalal, 1988] Dalal, M. (1988). Investigations into a theory of knowledge base revision: preliminary report. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, pages 475–479.

[Dimopoulos et al., 2006] Dimopoulos, Y., Gerevini, A., Haslum, P., and Saetti, A. (2006). The benchmark domains of the deterministic part of ipc-5. *Abstract Booklet of the competing planners of ICAPS-06*, pages 14–19.

[Edelkamp and Helmert, 2001] Edelkamp, S. and Helmert, M. (2001). Mips: The model-checking integrated planning system. *AI magazine*, 22(3):67.

[Edelkamp and Kissmann, 2009] Edelkamp, S. and Kissmann, P. (2009). Optimal symbolic planning with action costs and preferences. In *IJCAI*, pages 1690–1695.

[Fikes and Nilsson, 1972] Fikes, R. E. and Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208.

[Forbus, 1988] Forbus, K. D. (1988). Introducing actions into qualitative simulation. Technical report, DTIC Document.

[Fourman, 2000] Fourman, M. P. (2000). Propositional planning. In *Proceedings of AIPS-00 Workshop on Model-Theoretic Approaches to Planning*, pages 10–17.

[Geffner and Bonet, 2013] Geffner, H. and Bonet, B. (2013). A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: theory and practice*. Morgan Kaufmann.

[Giunchiglia and Traverso, 2000] Giunchiglia, F. and Traverso, P. (2000). Planning as model checking. *Recent Advances in AI Planning*, pages 1–20.

[Göbelbecker et al., 2010] Göbelbecker, M., Keller, T., Eyerich, P., Brenner, M., and Nebel, B. (2010). Coming up with good excuses: What to do when no plan can be found. In *Proc. of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 81–88.

[Harel et al., 2000] Harel, D., Kozen, D., and Tiuryn, J. (2000). *Dynamic Logic*. MIT Press.

[Helmert, 2006] Helmert, M. (2006). The fast downward planning system. *J. Artif. Intell. Res.(JAIR)*, 26:191–246.

[Herzig, 2014] Herzig, A. (2014). Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments. In Baral, C. and De Giacomo, G., editors, *In KR?14: Proceedings of the 14th International Conference on the Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann. Available from `http://www.irit.fr/~Andreas.Herzig/P/Kr14.html`.

[Herzig et al., 2014] Herzig, A., Menezes, M. V., Barros, L. N., and Wassermann, R. (2014). On the revision of planning tasks. In *(to appear) 21st European Conference on Artificial Intelligence*.

[Huth and Ryan, 2004] Huth, M. and Ryan, M. (2004). *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge Univ Pr.

[Katsuno and Mendelzon, 1991a] Katsuno, H. and Mendelzon, A. (1991a). On the difference between updating a knowledge base and revising it. In *KR'91: Proceedings of the 2nd International Conference on the Principles of Knowledge Representation and Reasoning*, pages 387–394, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Katsuno and Mendelzon, 1991b] Katsuno, H. and Mendelzon, A. O. (1991b). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294.

[Kripke, 1963] Kripke, S. (1963). Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94.

[Long and Fox, 2003] Long, D. and Fox, M. (2003). The 3rd international planning competition: Results and analysis. *J. Artif. Intell. Res. (JAIR)*, 20:1–59.

[Long et al., 2009] Long, D., Fox, M., and Howey, R. (2009). Planning domains and plans: Validation, verification and analysis. In *Proc. Workshop on V&V of Planning and Scheduling Systems*.

[McCluskey, 2000] McCluskey, T. (2000). Knowledge engineering for planning roadmap.

[McMillan, 1992] McMillan, K. L. (1992). Symbolic model checking: An approach to the state explosion problem. In *PhD thesis, CMU*.

[Menezes and Barros, 2011a] Menezes, M. V. and Barros, L. N. (2011a). Model update for automated planning. In *AAAI SIGART Doctoral Consortium*.

[Menezes and Barros, 2011b] Menezes, M. V. and Barros, L. N. (2011b). Planning domain model update. In *ICAPS Doctoral Consortium*.

[Menezes and Barros, 2013] Menezes, M. V. and Barros, L. N. (2013). Model checking for unsolvable planning problems. In *Proceedings of 16th Brazilian Symposium on Formal Methods*, pages 30–35.

[Menezes et al., 2012] Menezes, M. V., Barros, L. N., and Pereira, S. L. (2012). Planning task validation. *Scheduling and Planning Applications woRKshop (ICAPS)*.

[Menezes et al., 2013] Menezes, M. V., Barros, L. N., and Pereira, S. L. (2013). Regressao de acoes nao deterministicas: uma solucao simbolica. In *X Encontro Nacional de Inteligencia Artificial e Computacional*.

[Menezes et al., 2014] Menezes, M. V., Barros, L. N., and Pereira, S. L. (2014). Symbolic regression for non deterministic actions. *Learning and Nonlinear Models (to appear)*.

[Menezes et al., 2010] Menezes, M. V., Pereira, S. L., and Barros, L. N. (2010). Model updating in action. *Workshop on Knowledge Engineering for Planning and Scheduling (ICAPS)*.

[Menezes et al., 2011] Menezes, M. V., Pereira, S. L., and Barros, L. N. (2011). System design modification with actions. *LNAI–Advances in Artificial Intelligence–SBIA 2010*, pages 31–40.

[Pereira, 2007] Pereira, S. L. (2007). *Planejamento sob incerteza para metas de alcancabilidade estendidas*. PhD thesis, Instituto de Matematica e Estatistica da Universidade de Sao Paulo.

[Pereira and Barros, 2008] Pereira, S. L. and Barros, L. N. (2008). A logic-based agent that plans for extended reachability goals. *Autonomous Agents and Multi-Agent Systems*, 16(3):327–344.

[Rabideau et al., 2008] Rabideau, G., Chien, S., and McLaren, D. (2008). Tractable goal selection with oversubscribed resources.

[Rintanen, 2008] Rintanen, J. (2008). Regression for classical and nondeterministic planning. In *ECAI*, pages 568–571.

[Ritanen, 2006] Ritanen, J. (2006). *Introduction to Automated Planning*. ? (Draft).

[Russell, 2009] Russell, Stuart, P. N. (2009). *Artificial intelligence: A modern approach*. Prentice Hall.

[Russell et al., 1995] Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (1995). *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs.

[Shanahan, 1997] Shanahan, M. (1997). *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT press.

[Tarski, 1955] Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its applications. In *Pacific J. Math.*, volume 5, pages 285–309.

[Whaley, 2010] Whaley, J. (2010). JavaBDD-Java binary decision diagram library.