
**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS QUIXADÁ**

**CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**RÔMULO HERBERT FERREIRA FRANCO ALVES**

# MUDANÇA DE AÇÕES EM PROBLEMAS DE PLANEJAMENTO SEM SOLUÇÃO

**QUIXADÁ**

**2018**

RÔMULO HERBERT FERREIRA FRANCO ALVES

# MUDANÇA DE AÇÕES EM PROBLEMAS DE PLANEJAMENTO SEM SOLUÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Prof. Dra. Maria Viviane de Menezes

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

A482m Alves, Rômulo Herbert Ferreira Franco.
Mudança de ações em problemas de planejamento sem solução / Rômulo Herbert Ferreira Franco Alves. – 2018.
39 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2018.
Orientação: Profa. Dra. Maria Viviane de Menezes.

1. Inteligência Artificial. 2. Planejamento Automatizado. I. Título.

CDD 004.6

---

RÔMULO HERBERT FERREIRA FRANCO ALVES

# MUDANÇA DE AÇÕES EM PROBLEMAS DE PLANEJAMENTO SEM SOLUÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: ___/___/___

BANCA EXAMINADORA

---

Prof. Dra. Maria Viviane de Menezes (Orientadora)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Paulo de Tarso Guerra Oliveira
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

## RESUMO

Planejamento Automatizado é uma área importante da Inteligência Artificial que se preocupa com o processo de escolha de ações para que um agente inteligente alcance suas metas. Essas ações são criadas por um planejador, um algoritmo que recebe como entrada um problema de planejamento e, se esse problema possui solução, retorna uma sequência de ações (plano) que leva o agente do estado inicial do problema para um estado que satisfaz a meta (estado meta). Nem sempre o problema de planejamento possui solução. Nesses casos existem três possíveis razões: especificação errada do estado inicial, meta de planejamento é muito restritiva, *conjunto de ações especificado incorretamente ou incompleto*. Neste trabalho lidamos com a terceira possibilidade, propondo um método que, através de mudanças no conjunto de ações, torna o problema solucionável.



**Palavras-chave:** Inteligência Artificial. Planejamento Automatizado. Problemas sem solução.

**ABSTRACT**

Automated Planning is an important area of Artificial Intelligence that deals with the process of choosing actions for an intelligent agent to achieve its goals. These actions are created by a planner, an algorithm that receives as input a planning problem, and if this problem has a solution, returns a sequence of actions (plan) that takes the agent from the initial state of the problem to a state that satisfies the goal (goal state). The planning problem does not always have a solution. In these cases there are three possible reasons: incorrect specification of the initial state, planning goal is too restrictive, actions set incorrectly or incompletely specified. In this work we deal with the third possibility, proposing a method that, through changes in the set of actions, makes the problem solvable.

**Keywords:** Automated Planning. Artificial Intelligence. Problems without solution.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| GIPO | *Graphical Interface for Planning with Objects* |
| HTN | *Hierarchical Task Network* |
| ICAPS | *International Conference on Automated Scheduling and Planning* |
| ICKEPS | *International Competition on Knowledge Engineering for Planning and Scheduling* |
| IPC | *International Planning Competion* |
| KEPS | *Knowledge Engineering for Planning and Scheduling* |
| PDCK | *Procedural Domain Control Knowledge* |
| PDDL | *Planning Domain Definition Language* |
| STRIPS | *Stanford Research Institute Planning System* |
| UIPC | *Unsolvability International Planning Competition* |
| UML | *Unified Modeling Language* |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $A$ | Conjunto de ações |
| $d$ | Distância de Hamming |
| $D$ | Domínio de planejamento |
| $G$ | Conjunto dos estados que satisfazem a meta |
| $L$ | Função de rotulação de estados |
| $P$ | Conjunto de proposições |
| $R$ | Conjunto dos estados alcançáveis a partir do estado inicial |
| $R_a$ | Conjunto de estados de $R$ mais próximos de $U$ |
| $s_0$ | Estado inicial de um problema de planejamento |
| $s_g$ | Estado que satisfaz a meta de planejamento |
| $S$ | Conjunto de estados |
| $T$ | Função de transição de estados |
| $U$ | Conjunto dos estados que alcançam algum estado meta |
| $U_a$ | Conjunto de estados de $U$ mais próximos de $R_a$ |
| $\mu$ | Fato |
| $\pi$ | Plano |
| $\Pi$ | Problema de planejamento |
| $\psi$ | Base de conhecimento |
| $\varphi$ | Meta de planejamento |

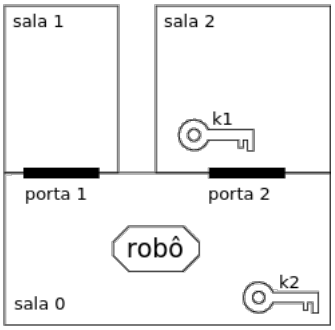
**SUMÁRIO**

# 1 INTRODUÇÃO

Planejamento é o ato de criar uma sequência de ações que quando executadas permitem alcançar um objetivo. *Planejamento Automatizado* (GHALLAB *et al.*, 2004) é uma área essencial da Inteligência Artificial que se preocupa com o processo de escolha de ações para que um agente inteligente alcance suas metas.

Considere um robô que tem a tarefa de sair de uma sala e chegar em outra, usando chaves para abrir as portas das salas. Esta tarefa pode ser modelada como um *domínio de planejamento* denominado *Domínio das Chaves* (GÖBELBECKER *et al.*, 2010). Um *domínio de planejamento* é composto pelo ambiente em que o agente se encontra para executar uma tarefa e pelas ações que o agente pode executar nesse ambiente. No *Domínio das Chaves*, por exemplo, o ambiente é constituído de salas, portas e chaves e as ações que o agente pode executar são: *pegar uma chave*, *abrir uma porta* e *entrar em uma sala*.

Figura 1 – Estado inicial $s_0$ de um problema de planejamento solucionável no Domínio das Chaves, cuja meta é que o robô alcance a sala 1.




Fonte: Adaptado de (GÖBELBECKER *et al.*, 2010).

As ações de um domínio de planejamento são dadas por precondições e efeitos. *Precondições* são proposições que precisam ser verdade para que a ação possa ser executada. Por exemplo, para o robô executar a ação *pegar uma chave* em uma sala é preciso existir tal chave nesta sala e o robô também precisa estar nesta sala. Os *efeitos*, por suas vez, modificam os valores das proposições. Por exemplo, quando o robô executa a ação *abrir uma porta*, o fato de que a porta está aberta passa de falso a verdadeiro.

A linguagem *Planning Domain Definition Language* (PDDL) (Linguagem de Definição de Domínios e Problemas de Planejamento, do inglês *Planning Domain Definition Language*) é a linguagem padrão para descrição de domínios e problemas de planejamento. Nela, o domínio é descrito através de predicados e ações. Os predicados são usados para descrever propriedades

do agente e do ambiente, por exemplo, a localização de objetos nesse ambiente. As ações representam o que pode ser feito para modificar o ambiente e, como já dito anteriormente, são descritas por precondições e efeitos.

Um *problema de planejamento* é composto por um domínio de planejamento, estado inicial e meta. No Domínio das Chaves podemos definir um problema com 3 salas (sala 0, sala 1, sala 2), duas portas (porta 1 e porta 2) e duas chaves ($k1$ e $k2$). A Figura 1 descreve o *estado inicial* deste problema, em que o robô está na sala 0, as portas 1 e 2 estão fechadas, a chave $k1$ está na sala 2, a chave $k2$ está na sala 0. Neste problema, a meta é que o robô alcance a sala 1.

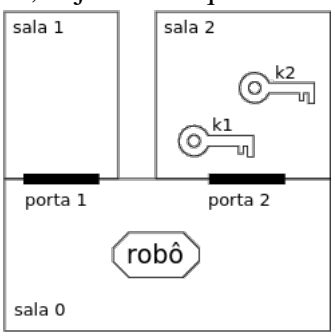
Figura 2 – Esquema de um planejador.


Fonte: Adaptado de (MENEZES, 2014).

Um planejador é um algoritmo que recebe um problema de planejamento e retorna um plano ou falha (Figura 2). Um plano é uma sequência de ações que leva o robô do estado inicial para um estado que satisfaz a meta. No problema definido anteriormente, um plano é a sequência de ações: *pegar a chave* $k2$ na sala 0, *abrir a porta* 2, *entrar na sala* 2, *pegar a chave* $k1$ na sala 2, *entrar na sala* 0, *abrir a porta* 1, *entrar na sala* 1.

No entanto, nem sempre é possível encontrar um plano para um problema de planejamento. Nestes casos, dizemos que o *problema de planejamento não possui solução*. Considere um problema no domínio das chaves em que o estado inicial é o estado apresentado na Figura 3 e a meta é que o robô alcance a sala 1. Este problema é um problema de planejamento sem solução, uma vez que não é possível que o robô tenha acesso à sala 2 nas quais as chaves estão localizadas.

Figura 3 – Estado inicial $s_0$ de um problema de planejamento sem solução no Domínio das Chaves, cuja meta é que o robô alcance a sala 1.




Fonte: Adaptado de (GÖBELBECKER *et al.*, 2010).

Existem três razões possíveis para que um problema de planejamento não possua solução: (i) especificação errada do estado inicial, assim não sendo possível alcançar um estado meta a partir dele; (ii) a meta de planejamento é muito restritiva, fazendo com que nenhum estado meta possa ser alcançado do estado inicial; (iii) *o conjunto de ações foi especificado incorretamente ou o conjunto de ações está incompleto*. Assim, para tonar um problema de planejamento sem solução solucionável, podemos realizar:

- mudança do estado inicial;
- mudanças na meta;
- mudanças no conjunto de ações.

As mudanças no estado inicial e na meta são caracterizadas como um problema de revisão de crenças (HERZIG *et al.*, 2014; MENEZES, 2014). Tais mudanças são realizadas modificando-se minimamente a definição original do problema de planejamento sem solução e não causa alterações em outros problemas definidos para o mesmo domínio, sendo denominadas *mudanças locais*. A mudança do conjunto de ações também é caracterizada como um problema de revisão de crenças (ALCHOURRÓN *et al.*, 1985), no entanto, é uma *mudança global*, uma vez que ao modificar um domínio de planejamento, outros problemas definidos para este mesmo domínio podem ser afetados.

## 1.1 Motivação

Historicamente, a maioria dos trabalhos na área de Planejamento Automatizado são focados na criação e melhoria de algoritmos para tornar a busca por um plano a mais eficiente possível. O desempenho destes planejadores é testado em uma competição bianual, que ocorre desde 1998, denominada *Competição Internacional de Planejamento - International Planning Competion* (IPC) (*International Planning Competion*) (PLANNING; SCHEDULING, 2018). Nesta competição, domínios e problemas *benchmarks* são elaborados e os planejadores são avaliados em relação ao desempenho na obtenção de planos.

Assim, nos primeiros anos da IPC, foi possível verificar: (i) de um lado, um esforço grande para elaboração de heurísticas e métodos que deixassem os planejadores cada vez mais eficientes; (ii) e de outro lado, pouca atenção na elaboração de ferramentas automáticas para construção, verificação e validação dos domínios *benchmarks*.

(LONG; FOX, 2003) apresenta erros contidos em domínios da IPC, o que mostra a pouca atenção dada a uma área de pesquisa, ainda hoje pouco explorada: a *verificação e*

*validação de domínios e problemas de planejamento*.

Um exemplo de erro de modelagem de domínio de planejamento ocorreu na IPC3 (PLANNING; SCHEDULING, 2002). Nesta competição, foi proposto o *Domínio dos Colonos* (*Settlers*). Neste domínio, o agente é um colono em uma ilha e deve explorar os recursos naturais desta ilha para construir estradas, aldeias e cidades. Algumas ações deste domínio estavam especificadas de forma errada e tais erros só foram encontrados seis anos após a criação do domínio (LONG; FOX, 2003). Esse domínio com erros de modelagem foi utilizado durante três edições da IPC. Outro exemplo de erro de modelagem ocorreu no domínio *MPrime* (BACCHUS, 2001), originalmente projetado para a IPC1. Este domínio modelava a tarefa de transporte de objetos por veículos que consumiam uma certa quantidade de combustível por cada ação. Um erro na modelagem do consumo de combustível foi constatado posteriormente.

É importante perceber que o trabalho de (LONG *et al.*, 2009), ao apontar os erros em domínios *benchmarks* da IPC, não tem o objetivo de desvalorizar a competição, mas chamar a atenção da comunidade científica para o fato de que o processo de construção, modelagem, verificação e validação de domínios de planejamento estava sendo feito de forma manual, sem nenhum suporte de processo ou ferramenta automática.

Outra competição que visa incentivar a pesquisa na área de Engenharia do Conhecimento é a *International Competition on Knowledge Engineering for Planning and Scheduling* (ICKEPS) (*International Competition on Knowledge Engineering for Planning and Scheduling*) (PLANNING; SCHEDULING, 2012). Até agora ela teve quatro edições (2005, 2007, 2009, 2012), com cada edição tendo diferentes focos na área de Engenharia do Conhecimento.

Em 2016, ocorreu pela primeira vez a competição internacional para problemas de planejamento sem solução, *Unsolvability International Planning Competition* (UIPC) (*Unsolvability International Planning Competition*)
(PLANNING; SCHEDULING, 2016). Nesta competição o objetivo é avaliar a capacidade dos planejadores em detectar se um problema de planejamento possui ou não solução. Essa competição foi mais um esforço de reunir algoritmos para tratar dos casos em que falhas na definição de um problema de planejamento pode comprometer o desempenho até mesmo dos planejadores estado da arte.

A motivação deste trabalho é contribuir para a área de *verificação e validação de modelos de domínios de planejamento*, especificamente na tarefa de modificar domínios de planejamento para que problemas sem solução tornem-se solucionáveis.

## 1.2 Trabalhos Relacionados

Para incentivar trabalhos e discussões na área de engenharia do conhecimento para planejamento, a comunidade científica promove um *workshop* denominado *Knowledge Engineering for Planning and Scheduling* (KEPS) (*Knowledge Engineering for Planning and Scheduling*). Tal *workshop* foi realizado no *International Conference on Automated Scheduling and Planning* (ICAPS) (*International Conference on Automated Scheduling and Planning*) em 2008, 2010, 2011, 2013, 2014 e 2017. Dentre os trabalhos publicados do KEPS, os relacionados com este projeto são: (VAQUERO *et al.*, 2011), (SHAH *et al.*, 2013), (WICKLER, 2013), (BARTÁK; MAILLARD, 2017), (GÖBELBECKER *et al.*, 2010).

O trabalho de Vaquero, Silva & Beck (VAQUERO *et al.*, 2011) lista as seis fases do processo de projeto de modelos de domínios de planejamento, a saber:

- *Especificação de Requerimentos*,
- *Modelagem do Conhecimento*,
- *Análise do Modelo*,
- *Implantação do Modelo no Planejador*,
- *Síntese do Plano*,
- *Análise do Plano e Pós-Design*.

A fase de *Especificação de Requerimentos* trata do levantamento, análise e validação dos requisitos para a elaboração do domínio de planejamento. A fase de *Modelagem do Conhecimento* trata da criação do modelo de domínio de planejamento baseado nos requisitos levantados, analisados e validados. A fase de *Análise do Modelo* trata da verificação e validação do modelo de domínio de planejamento e do problema de planejamento. A fase de *Implantação do Modelo no Planejador* trata da tradução das especificações do problema em algum tipo de linguagem entendida por um planejador automatizado, geralmente PDDL. A fase de *Síntese do Plano* trata da criação de um plano solução para o problema de planejamento pelo planejador automatizado escolhido. A fase de *Análise do Plano e Pós-Design* trata da análise do plano criado usando alguma métrica. Finalmente, (VAQUERO *et al.*, 2011) lista ferramentas de suporte existentes para cada uma dessas fases.

O trabalho de Shah et. al., (SHAH *et al.*, 2013) avalia três abordagens utilizadas na fase de *Modelagem do Conhecimento*: a abordagem tradicional em que um especialista em planejamento constrói manualmente o modelo do domínio de planejamento, a partir de uma situação ou sistema observado; a abordagem que utiliza a ferramenta itSIMPLE (VAQUERO *et*

*al.*, 2007) de modelagem de domínios usando métodos baseados em *Unified Modeling Language* (UML) (*Unified Modeling Language*); a abordagem que utiliza a notação hierárquica $OCL_h$ baseada em métodos formais, com a ajuda da ferramenta *Graphical Interface for Planning with Objects* (GIPO) (SIMPSON *et al.*, 2007). Essa avaliação aponta vantagens e desvantagens de cada uma das abordagens e mostra deficiências que futuras ferramentas de modelagem baseada em PDDL precisam atender.

O trabalho de Wickler (WICKLER, 2013) propõe um método para a fase de *Análise do Modelo* de domínios de planejamento com a identificação de grafos estáticos implícitos no conjunto de ações desses domínios. Esses grafos são criados baseados em sistema de tipagem e em predicados estáticos do domínio de planejamento (aqueles que não são alterados pelas ações). Tais grafos são utilizados para fazer análises sobre o domínio de planejamento, podendo auxiliar a encontrar erros na especificação desse domínio.

O trabalho de Barták e Maillard (BARTÁK; MAILLARD, 2017) explora técnicas de linguagens formais para verificar domínios de planejamento e validar planos soluções. Bartak & Maillard provê um método para converter gramáticas de atributos para linguagens de especificação de domínios como: *Stanford Research Institute Planning System* (STRIPS) (EROL *et al.*, 1995), *Hierarchical Task Network* (HTN) (*Hierarchical Task Network*) (EROL *et al.*, 1996) e *Procedural Domain Control Knowledge* (PDCK) (*Procedural Domain Control Knowledge*) (BAIER *et al.*, 2007).

A abordagem da literatura mais relacionada com este trabalho é a de Göbelbecker (GÖBELBECKER *et al.*, 2010) que propõe a inserção de novas ações, denominadas ações fictícias, para tornar um problema de planejamento sem solução solucionável. Estas ações modificam o valor de apenas uma variável do domínio de planejamento. No entanto, o objetivo do trabalho de Göbelbecker é o uso destas ações como guia para a mudança do estado inicial.

## 1.3 Objetivos

O objetivo deste trabalho é propôr um método de mudança no conjunto de ações de um domínio de planejamento para que problemas não solucionáveis tornem-se solucionáveis.

## 1.4 Organização

Este texto está organizado de forma a apresentar: no Capítulo 2, Planejamento Automatizado, incluindo os conceitos mais aprofundado sobre domínios, problemas e planos, formas de encontrar um plano, problemas sem solução e mudanças de problemas sem solução; no Capítulo 3, Revisão de Crenças, são descritos os conceitos básicos de revisão de crenças; no Capítulo 4, Mudança de Ações com Revisão de Crenças, descrevemos a mudança do conjunto de ações de um domínio de planejamento usando a revisão de crenças; no Capítulo 5, Método de Inclusão de Novas Ações com Mudanças Minimais, é proposto o método de mudança de ações para tornar um problema sem solução solucionável; no Capítulo 6, Conclusão, temos uma discussão final deste trabalho e de possíveis trabalhos futuros.

# 2 PLANEJAMENTO AUTOMATIZADO

## 2.1 Domínios, Problemas e Planos

Seja $P$ o conjunto de proposições descrevendo as propriedades do agente e $A$ o conjunto de ações que o agente é capaz de executar, um domínio de planejamento (GHALLAB *et al.*, 2004) é um sistema de transição de estados $D = <S, L, T>$ onde:

- $S = \{s_0, ..., s_n\}$ é um conjunto finito e enumerado de estados;
- $L : S \rightarrow 2^P$ é a função de rotulação de estados, a qual descreve que proposições atômicas são verdadeiras em um estado;
- $T : SxA \rightarrow S$ é uma função de transição de estados em que dado um estado $s \in S$ e uma ação $a_i \in A$, $T$ devolve o estado sucessor de $s$ após a execução da ação $a_i$.

Dizemos que o par $(P,A)$ é a assinatura do domínio de planejamento.

Domínios de tamanho realista, geram sistemas de transição de estados com um número muito grandes de estados e, assim, torna-se inviável tal representação. Desta forma, em planejamento utiliza-se uma representação implícita do domínio dada por meio das *ações* que o agente pode executar. Assim, no lugar de ter o grafo completo de todo o espaço de estados, é possível gerar a partir do estado inicial ou da meta, apenas os estados que possivelmente levam à obtenção de um plano solução.

Usualmente, é utilizada uma linguagem padrão para descrição do domínio de planejamento, a linguagem PDDL (*Planning Domain Definition Language*) (MCDERMOTT *et al.*, 1998). A linguagem PDDL utiliza uma notação baseada em lógica de predicados com variaveis que posteriormente serão instanciadas com objetos do problema.

A Figura 4 ilustra a descrição em PDDL do domínio do Mundo das Chaves dado pelo conjunto de predicados e pelo conjunto de ações.

Os predicados que descrevem as características do ambiente são indicados em (:predicates), são eles:

- `(porta ?p)`, predicado unário que indica que um objeto ?$p$ é uma porta;
- `(sala ?s)`, predicado unário que indica que um objeto ?$s$ é uma sala;
- `(chave ?c)`, predicado unário que indica que um objeto ?$c$ é uma chave;
- `(robô-na ?s)`, predicado unário que indica que o robô está em ?$s$;
- `(chave-na ?c ?s)`, predicado binário que indica que a chave ?$c$ está em ?$s$;
- `(fechada ?p)`, predicado unário que indica que a porta ?$p$ está fechada;

Figura 4 – Descrição do Domínio das Chaves em PDDL.

```
(define (domain Chaves)

  (:predicates (porta ?p) (sala ?s) (chave ?c)
               (robô-na ?s) (chave-na ?c ?s)
               (fechada ?p) (chave-abre ?c ?p)
               (conecta ?p ?s ?s) )

  (:action abrir-porta
      :parameters (?c ?p ?s1 ?s2)
      :precondition (and (chave ?c) (porta ?p) (sala ?s1)
                         (sala ?s2) (conecta ?p ?s1 ?s2)
                         (robô-na ?s1) (chave-na ?c ?s1)
                         (fechada ?p) (chave-abre ?c ?p))
      :effect (and (not (fechada ?p))))

  (:action fechar-porta
      :parameters (?p ?s1 ?s2)
      :precondition (and (conecta ?p ?s1 ?s2)
                         (robô-na ?s1) (not (fechada ?p))
                         (porta ?p) (sala ?s1) (sala ?s2))
      :effect (and (fechada ?p)))

  (:action mover
      :parameters (?p ?s1 ?s2)
      :precondition (and (conecta ?p ?s1 ?s2)
                         (not (fechada ?p)) (robô-na ?s1)
                         (porta ?p) (sala ?s1) (sala ?s2))
      :effect (and (not (robô-na ?s1)) (robô-na ?s2)))

  (:action mover-com-chave
      :parameters (?p ?c ?s1 ?s2)
      :precondition (and (conecta ?p ?s1 ?s2)
                         (not (fechada ?p)) (robô-na ?s1)
                         (chave-na ?s1) (porta ?p)
                         (chave ?c) (sala ?s1) (sala ?s2))
      :effect (and (not (robô-na ?s1)) (robô-na ?s2)
                   (not (chave-na ?s1))
                   (chave-na ?s2))))
```

Fonte: Elaborado pelo autor.

- `(chave-abre ?c ?p)`, predicado binário que indica que a chave ?$c$ é capaz de abrir a porta ?$p$;
- `(conecta ?p ?s1 ?s2)`, predicado ternário que indica que uma porta ?$p$ conecta a sala ?$s1$ a sala ?$s2$.

Em PDDL uma variável seguida do símbolo ? indica que ela pode ser substituída por

objeto concreto em uma versão instanciada do domínio (*grounded version*). Por exemplo, a variável ?$p$ em `(porta ?p)` pode ser substituída por um objeto $p1$, gerando o átomo proposicional `(porta-p1)`.

Cada ação do domínio é dada por meio de parâmetros, precondições e efeitos. No domínio das chaves, temos as seguintes ações:

- `abrir-porta`: na qual o robô abre a porta de uma determinada sala, utilizando uma determinada chave;
- `fechar-porta`: na qual o robô fecha a porta de uma determinada sala, não sendo necessário utilizar chave;
- `mover`: o robô move-se de uma sala para outra e;
- `mover-com-chave`: o robô move-se de uma sala para outra levando consigo uma chave.

Os *parâmetros* de cada ação, localizados na lista `(:parameters)`, indicam que objetos estão envolvidos na descrição da ação. A ação mover, por exemplo, tem 3 parâmetros ?$p$ ?$s1$ e ?$s2$.

As *precondições* de uma ação são o conjunto de proposições que precisam ser verdade em um estado para que a ação possa ser executada nesse estado. Se uma ação pode ser executada em um determinado estado, diz-se que essa ação é *aplicável* a esse estado. Por exemplo, as precondições da ação mover (indicadas na lista `:precondition`) são: que ?$s1$ e ?$s2$ sejam salas (sala ?$s1$) e (sala ?$s2$); que ?$p$ seja uma porta (porta ?$p$); que o robô esteja na sala $s1$; que a porta ?$p$ conecte a sala ?$s1$ à sala ?$s2$ e; que a porta ?$p$ esteja aberta (not (fechada ?$p$)). A utilização da palavra reservada `and`, indica que todas as proposições devem ser verdadeiras.

Os *efeitos* de uma ação são as proposições que serão modificadas após a aplicação da ação. Efeitos podem ser positivos (a proposição será verdadeira no estado posterior à aplicação da ação) ou negativos (a proposição será falsa no estado posterior à aplicação da ação). Por exemplo, os efeitos da ação `mover` (indicados na lista `:effects`) são: que o robô esteja na sala ?$s2$ (loc-robô $s2$) e que o robô não esteja na sala ?$s1$, `(not (loc-robô s1))`.

Formalmente, uma ação $a \in A$ é especificada pelo par <$precond(a), efeitos(a)$>, sendo precond($a$) um conjunto de proposições de $P$ que devem ser verdadeiras no estado em que a ação a é executada, e efeitos($a$) um par <$efeitos^{+}(a), efeitos^{-}(a)$> de efeitos positivos e negativos da ação $a$ em que $efeitos^{+}(a)$ é o conjunto de proposições de $P$ que se tornam verdadeiras após a execução da ação e $efeitos^{-}(a)$ é o conjunto de proposições de $P$ que se tornam falsas após a ser executada.

Um problema de planejamento (GHALLAB *et al.*, 2004) é uma tupla $\Pi = <D, s_0, \varphi>$ onde $D$ é o domínio de planejamento, $s_0 \in S$ é o estado inicial, e $\varphi$ é a meta de planejamento. A meta de planejamento $\varphi$ é um conjunto de proposições $p$. Um estado $s$ satisfaz a meta $\varphi$ se $\varphi \subseteq s$.

A Figura 5 descreve um problema de planejamento do Domínio das Chaves em PDDL.

Figura 5 – Descrição em PDDL de um problema no Domínio das Chaves.

```
(define (problem chaves-prob)
  (:domain Chaves)
  (:objects k1 k2 sala0 sala1 sala2 porta1 porta2)

  (:init (chave k1) (chave k2)
         (sala sala0) (sala sala1) (sala sala2)
         (porta porta1) (porta porta2)
         (chave-abre k1 porta1) (chave-abre k2 porta2)
         (robô-na sala0)
         (conecta porta1 sala0 sala1)
         (conecta porta1 sala1 sala0)
         (conecta porta2 sala0 sala2)
         (conecta porta2 sala2 sala0)
         (chave-na k1 sala2) (chave-na k2 sala0)
         (fechada ?porta1) (fechada ?porta2))

  (:goal (and (loc-robô sala1))))
```

Fonte: Elaborado pelo autor.

Utilizamos o termo (:objects) para definir os objetos que são instanciados no problema, o termo (:init) para definir as proposições que são verdadeiras no estado inicial e os tipos dos objetos, e o termo (:goal) para definir a meta de planejamento.

Um plano (GHALLAB *et al.*, 2004) é uma sequência de ações $\pi = (a_1,..., a_j)$, onde j $\geq 0$. Um plano $\pi = (a_1, a_2,..., a_k)$ é solução de um problema $\Pi = <D, s_0, \varphi>$ se $T(s_0, a_1) = s_1$, $T(s_1, a_2) = s_2,..., T(s_{k-1}, a_k) = s_k$, onde $s_k$ é um estado que satisfaz a meta $\varphi$.

No problema de planejamento da Figura 5, um plano solução é a sequência de ações: *abrir-porta(k2, porta2, sala0, sala2)*, *mover(porta2, sala0, sala2)*, *mover-com-chave(porta2, k1, sala2, sala0)*, *abrir-porta(k1, porta1, sala0, sala1)*, *mover(porta1, sala0, sala1)*.

## 2.2 Em Busca de uma Solução

Um plano solução para um problema de planejamento pode ser encontrado realizando uma busca progressiva a partir do estado inicial, tentando alcançar um estado que satisfaz a meta, ou com uma busca regressiva a partir de todos os estados que satisfazem a meta, tentando encontrar um estado inicial.

### *2.2.1 Busca Progressiva*

A busca progressiva começa do estado inicial $s_0$ e progride para seus estados sucessores, depois ela progride para os sucessores dos sucessores, repetindo essa progressão para cada estado encontrado, excluindo estados que já foram encontrados antes, até achar um estado que satisfaz a meta $\varphi$ ou não encontrar nenhum estado novo. Quando a busca chega em um estado que satisfaz a meta $\varphi$, ela retorna a sequência de ações que foram executadas para chegar nesse estado. Se a busca não encontrar estado que satisfaz a meta, o problema $\Pi$ não possui solução.

Uma ação $a$ é aplicável em um estado $s$ se precond($a$) $\subseteq s$. Por exemplo, no Domínio das Chaves a ação de abrir uma porta só é aplicável se a portar que o robô vai abrir esta conectada a sala que ele está, a chave que ele vai usar também esta na sala em que ele está e essa chave é da porta que ele vai abrir.

Dado um estado $s$ e uma ação aplicável $a$, o estado sucessor de $s$ é obtido por meio das precondições e efeitos das ações, segundo a Equação 2.1 (GHALLAB *et al.*, 2004).

$$prog(s,a) = (s - efeitos^{-}(a)) \cup efeitos^{+}(a) \tag{2.1}$$

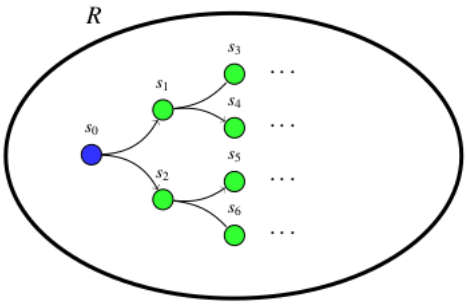
Na Equação 2.1, $efeitos^{-}(a)$ são os efeitos negativos da ação $a$, ou seja, as proposições do estado $s$ que vão se tornar falsas com a execução dessa ação, e $efeitos^{+}(a)$ são os efeitos positivos, ou seja, o que vai se tornar verdade com a execução dessa ação.

A busca progressiva realizada continuamente até que nenhum novo estado seja gerado encontra o conjunto de todos os estados alcançáveis a partir do estado inicial, denominado **R** (MENEZES, 2014).

### *2.2.2 Busca Regressiva*

O algoritmo de busca regressiva é outra forma de encontrar um plano solução para um problema (GHALLAB *et al.*, 2004). Ele recebe um problema de planejamento $\Pi$ = <$D$, $s_0$,

Figura 6 – Conjunto **R** dos estados alcançáveis a partir do estado inicial $s_0$ para um problema de planejamento $\Pi$ = <$D$, $s_0$, $\varphi$>.




Fonte: (MENEZES, 2014).

$\varphi$> como entrada e, se possui solução, retorna um plano $\pi$.

Para entender a busca regressiva é preciso primeiro entender o conceito de ação relevante para um estado. Uma ação $a$ é dita relevante para um estado $s$ (GHALLAB *et al.*, 2004) se e somente se

$$L(s) \cap efeitos^{+}(a) \neq \emptyset \text{ e } L(s) \cap efeitos^{-}(a) = \emptyset \tag{2.2}$$

L($s$) é o conjunto das proposições verdadeiras em $s$.

A busca regressiva começa na meta $\varphi$, ações relevantes a meta são escolhidas e o cálculo da Equação 2.3 (GHALLAB *et al.*, 2004) é efetuado.

$$regr(s,a) = (s - efeitos^{+}(a)) \cup precond(a) \tag{2.3}$$

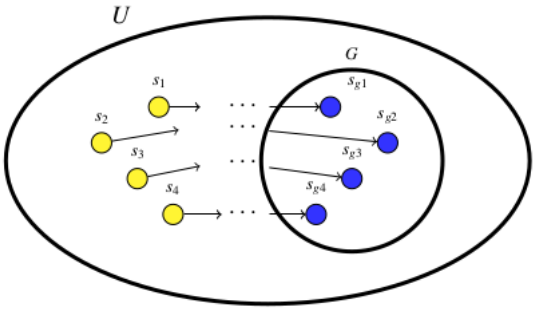
No primeiro cálculo da Equação 2.3 $s = \varphi$. O cálculo se repete até que $s_0$ seja alcançado ou não existam mais ações relevantes. Se $s_0$ for encontrado, a sequência de ações para chegar em $s_0$ em reverso é retornada, se não, o problema $\Pi$ não possui solução.

A busca regressiva, se aplicada continuamente até que nenhum novo estado seja gerado, encontra o conjunto de todos os estados que alcançam algum estado meta, denominado **U** (MENEZES, 2014). Todo estado em que ações relevantes a meta são aplicáveis são estados que alcançam algum estado meta.

### *2.2.3 Condições Para a Existência de um Plano Solução*

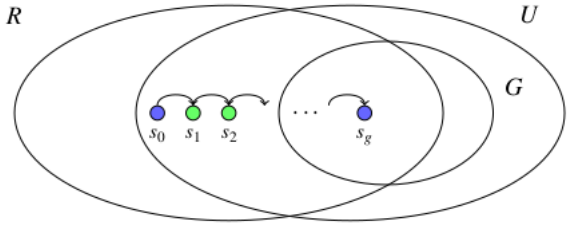
Dado um problema de planejamento $\Pi$ = <$D$, $s_0$, $\varphi$>, o conjunto **R** de estados alcançáveis a partir de $s_0$ e o conjunto **U** de estados que alcançam estados meta, o problema $\Pi$ possui solução se e somente se $\mathbf{R} \cap \mathbf{U} \neq \emptyset$ (MENEZES, 2014).

Figura 7 – Conjunto **U** dos estados que alcançam algum estado meta para um problema de planejamento $\Pi = <D, s_0, \varphi>$, sendo **G** o conjunto dos estados que satisfazem a meta $\varphi$.




Fonte: (MENEZES, 2014).

Figura 8 – Conjuntos de estados **R**, **U** e **G** para um problema de planejamento solucionável: $\mathbf{R} \cap \mathbf{U} \neq \emptyset$.

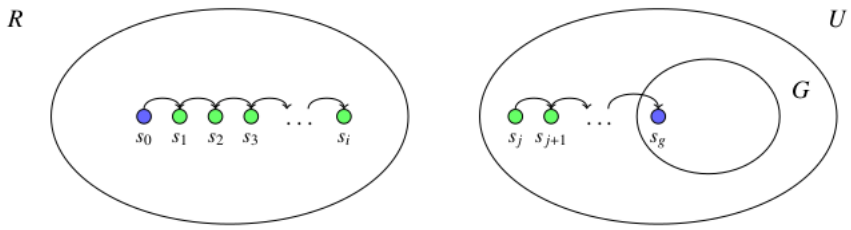



Fonte: (MENEZES, 2014).

## 2.3 Problemas de Planejamento Sem Solução

Dado um problema de planejamento $\Pi = <D, s_0, \varphi>$, o conjunto **R** de estados alcançáveis a partir de $s_0$ e o conjunto **U** de estados que alcançam estados meta, o problema $\Pi$ não possui solução se e somente se $\mathbf{R} \cap \mathbf{U} = \emptyset$ (MENEZES, 2014).

Figura 9 – Conjuntos de estados **R**, **U** e **G** quando um problema de planejamento não possui solução ($\mathbf{R} \cap \mathbf{U} = \emptyset$).




Fonte: (MENEZES, 2014).

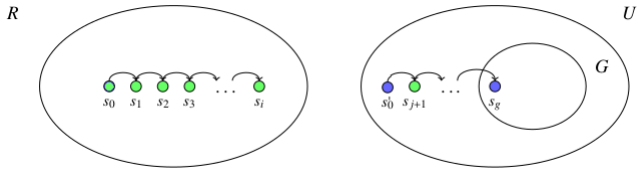
Outras condições para que um problema de planejamento não possua solução são $s_0 \notin \mathbf{U}$ e $s_g \notin \mathbf{R}$. Isso é ilustrado na Figura 9.

## 2.4 Mudanças em Problemas Sem Solução

Dado um problema de planejamento $\Pi = <D, s_0, \varphi>$ sem solução, o conjunto **R** de estados alcançáveis a partir de $s_0$ e o conjunto **U** de estados que alcançam estados meta. Existem três tipos de mudança para tornar o problema $\Pi$ solucionável (MENEZES, 2014):

- **Mudança do estado inicial** $s_0$: Um novo estado inicial $s'_0 \in$ **U** é selecionado;

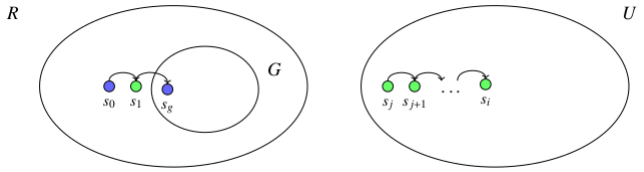
Figura 10 – Novo estado inicial dentro do conjunto **U**.




Fonte: Adaptado de (MENEZES, 2014).

- **Mudanças na meta** $\varphi$: Uma nova meta de planejamento $\varphi'$ tal que **R** $\cap$ **U'** $\neq \emptyset$ (**U'** é o conjunto de estados que alcançam estados que satisfazem a meta $\varphi'$) é definida;

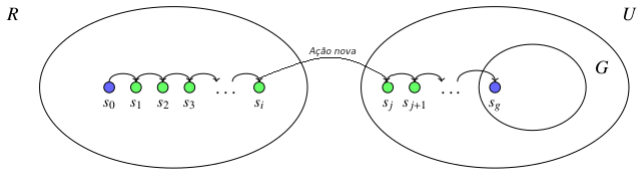
Figura 11 – Novos estados meta dentro do conjunto **R**.




Fonte: Adaptado de (MENEZES, 2014).

- **Mudanças no conjunto de ações** $A$: Inserir ações novas que conectem os conjuntos **R** e **U**. O conjunto de ações novo $A' = A \cup A_{NOVA}$ ($A_{NOVA}$ é o conjunto das ações novas) criará um domínio de planejamento novo $D' = <S, A', \psi>$ em que o problema de planejamento $\Pi' = <D', s_0, \varphi>$ é solucionável.

Figura 12 – Nova ação conectando **R** a **U**.




Fonte: Adaptado de (MENEZES, 2014).

# 3 REVISÃO DE CRENÇAS

A mudança de um problema de planejamento sem solução pode ser vista como um problema de mudança de crenças de uma base de conhecimento na lógica proposicional (MENEZES, 2014). Neste capitulo apresentamos os principais conceitos de revisão de crenças, um tipo de mudança em uma base de conhecimento.

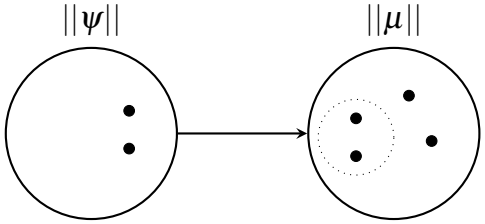
*Mudança de crenças* (GÄRDENFORS, 1988; WINSLETT, 2005) é a área que estuda o processo de mudanças em uma base de conhecimento quando novos fatos são descobertos sobre o mundo. Considere que a base de conhecimento é representada por uma teoria $\psi$ de uma lógica proposicional e que o novo fato $\mu$ a ser incorporado é inconsistente com $\psi$. Para que $\mu$ possa ser incorporada à base de conhecimento, devemos modificá-la a fim de eliminar a inconsistência. É desejável que a mudança em uma base de conhecimento siga algum *princípio da mudança minimal*, i.e., a base de conhecimento deve ser minimamente modificada para incorporar o novo fato.

A *revisão* de uma base de conhecimento $\psi$ por um fato $\mu$ é denotada por $\psi * \mu$, sendo fundamentada nos postulados AGM propostos por (ALCHOURRÓN *et al.*, 1985) que são regras de racionalidade que todo operador de revisão deve satisfazer. O operador de revisão escolhe os modelos de $\mu$ *mais próximos* ao conjunto de modelos de $\psi$, minimizando a distância global com relação a todos os estados do conjunto. Ou seja, a revisão $\psi * \mu$ é o conjunto de modelos do novo fato $\mu$ que estão *mais próximos* aos modelos da base.

Visualizando $\psi$ e $\mu$ como conjuntos de estados, o processo de revisão de $\psi * \mu$ devolve os estados de $\mu$ que são mais proximos de $\psi$, utilizando alguma métrica. A Figura 13 ilustra como ocorre o processo de revisão de $\psi$ por $\mu$. No lado direito, temos o conjunto de estados em que a fórmula $\mu$ é verdade, i.e., os modelos de $\mu$, denotado por $||\mu||$. No lado esquerdo da figura temos o conjunto de estados da base de conhecimento que são os modelos de $||\psi||$, i.e., o conjunto de estados em que $\psi$ é verdade. Na figura, os estados de $\mu$ mais próximos de $\psi$ estão representados dentro do círculo pontuado. Uma possível métrica para ordenação de estados é a distância de Hamming. Uma definição formal da distância de Hamming pode ser vista na Definição 3.0.1.

**Definição 3.0.1 (Distância de Hamming entre estados)** *Sejam $s_1$ e $s_2$ dois estados de um domínio de planejamento D = <S, L, T>, cada um definido pelo conjunto das proposições que são verdadeiras e que rotulam estado ($L(s_1)$ e $L(s_2)$), a distância de Hamming entre $s_1$ e $s_2$ é dada*

Figura 13 – Revisão de $||\psi||$ por $||\mu||$.




Fonte: Elaborado pelo autor.

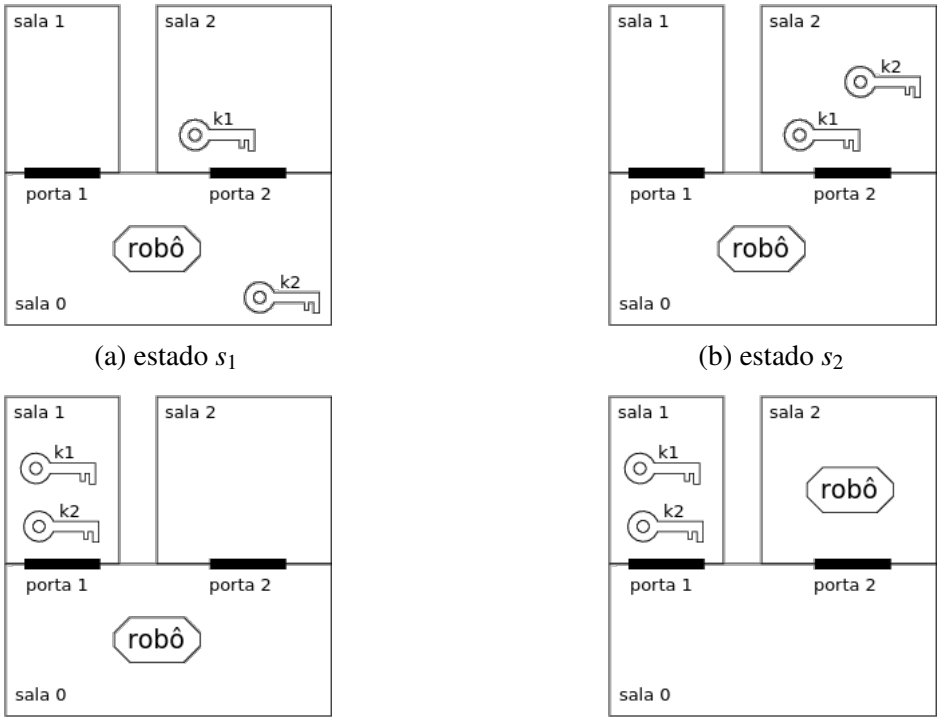
*por:*

$$h(s_1, s_2) = |L(s_1) - L(s_2)|$$

*Em que* $L(s_1) - L(s_2) = \{L(s_1) \backslash L(s_2)\} \cup \{L(s_2) \backslash L(s_1)\}$ *é a diferença simétrica entre os conjuntos de proposições* $L(s_1)$ *e* $L(s_2)$ *e* $|(L(s_1) - L(s_2)|$ *é a cardinalidade do conjunto* $L(s_1) - L(s_2)$.

**Exemplo 3.0.1** *(Distância de Hamming) Para exemplificar a distância de Hamming, vamos calcular a distância entre o estado* $s_1$ *e os estados* $s_2$, $s_3$, $s_4$ *da Figura 14.*

Figura 14 – Estados do Exemplo 3.0.1.




(a) estado $s_1$

(b) estado $s_2$

(c) estado $s_3$

(d) estado $s_4$

Fonte: Adaptado de (GÖBELBECKER *et al.*, 2010).

*Como a distância de Hamming usa a diferença simétrica entre os conjuntos de proposições verdadeiras dos estados, as proposições verdadeiras dos estados* $s_1$, $s_2$, $s_3$ *e* $s_4$ *são:*

- $s_1$ = {*porta-1-fechada, porta-2-fechada, robo-na-sala-0, chave-k1-na-sala-2, chave-k2-na-sala-0*}
- $s_2$ = {*porta-1-fechada, porta-2-fechada, robo-na-sala-0, chave-k1-na-sala-2, chave-k2-na-sala-2*}
- $s_3$ = {*porta-1-fechada, porta-2-fechada, robo-na-sala-0, chave-k1-na-sala-1, chave-k2-na-sala-1*}
- $s_4$ = {*porta-1-fechada, porta-2-fechada, robo-na-sala-2, chave-k1-na-sala-1, chave-k2-na-sala-1*}

*Baseado nesses conjuntos de proposições o cálculo da distância entre o estado $s_1$ e os estados $s_2$, $s_3$ e $s_4$ é:*

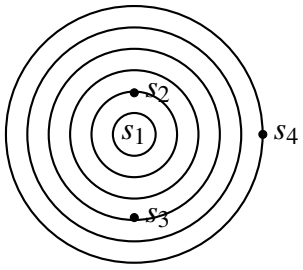
$h(s_1, s_2) = 2$

$h(s_1, s_3) = 4$

$h(s_1, s_4) = 6$

A revisão de crenças realiza uma ordenação dos mundos possíveis de acordo com a métrica adotada. Uma possível representação é dada por um sistema de esferas em que o centro representa os modelos da base e os modelos da fórmula são colocados no sistema de esferas de acordo com a distância para os modelos da base. Por exemplo, considere o Exemplo 3.0.1 com o estado $s_1$ como modelo da base, teremos o sistema de esferas da Figura 15.

Figura 15 – Sistema de esferas do Exemplo 3.0.1.




Fonte: Elaborado pelo autor.

# 4 MUDANÇA DE AÇÕES COM REVISÃO DE CRENÇAS

O trabalho de (MENEZES, 2014) propõe um método de mudança de ação para tornar problemas de planejamento sem solução solucionáveis usando a abordagem formal de revisão de crenças (ALCHOURRÓN *et al.*, 1985). Este método funciona como descrito a seguir. A partir de um domínio de planejamento, descrito em PDDL, e de um problema sem solução neste domínio, é realizada uma busca no espaço de estados de forma a obter os conjuntos **R** (por meio de uma busca progressiva) e **U** (por meio de uma busca regressiva). Como o problema não tem solução, temos que: $\mathbf{R} \cap \mathbf{U} = \emptyset$ conforme mostrado na seção 2.3. Desta forma o método propõe conectar estados do conjunto **R** com estados do conjunto **U** adicionando ações no domínio de planejamento. Estes estados não são escolhidos ao acaso, mas usando os princípios da abordagem formal de revisão de crenças a qual obedece o princípio da mudança minimal.

Intuitivamente, queremos adicionar ações que realizem pequenas modificações nos estados em que são aplicáveis. O processo é feito em duas fases: i) primeiro, a obtenção do conjunto $R_a$ dos estados de **R** que são mais próximos de **U**; ii) segundo, a obtenção do conjunto $U_a$ dos estados de **U** mais próximos de $R_a$; iii) finalmente, a ação que conecta **R** e **U** $a_{nova} = <R_a, U_a>$ é criada. O processo é descrito formalmente na Definição 4.0.1.

**Definição 4.0.1 (Mudança de Ações com Revisão de Crenças)** *Sejam* $\Pi = <D, s_0, \varphi>$ *um problema de planejamento sem solução com assinatura (P,A);* ***R*** *o conjunto de estados alcançáveis a partir do estado inicial; e* ***U*** *o conjunto dos estados que alcançam algum estado meta, os estados* $R_a \subseteq \boldsymbol{R}$ *e* $U_a \subseteq \boldsymbol{U}$ *são obtidos como a seguir (MENEZES, 2014):*

$$R_a = U * R = \{s_r \in R : \exists s_u \in U \text{ tal que } h(s_u, s_r) \leq h(s'_u, s'_r) \forall s'_r \in R, \forall s'_u \in U\}.$$

$$U_a = R_a * U = \{s_u \in U : \exists s_r \in R_a \text{ tal que } h(s_r, s_u) \leq h(s'_r, s'_u) \forall s'_r \in R_a, \forall s'_u \in U\}.$$

*Em que h(s,s′) é a distância de Hamming entre os estados s e s′. A ação* $a_{nova} = <R_a, U_a>$ *é inserida no domínio de planejamento, em que* $R_a$ *são as precondições de* $a_{nova}$ *e* $U_a$ *seus efeitos.*

A Definição 4.0.1 mostra que a partir dos conjuntos **R** e **U** obtém-se os conjuntos $R_a$ e $U_a$, com operações de revisão de crenças.

Para ilustrar o passo a passo deste método de mudança de ações, considere o domínio de planejamento do Exemplo 4.0.1 e o problema sem solução neste domínio no Exemplo 4.0.2.

**Exemplo 4.0.1** *(Domínio de Planejamento) Seja P um conjunto de proposições* $\{p, q, r\}$ *e seja A o conjunto de ações* $\{a_1, a_2, a_3, a_4, a_5\}$ *descritos em PDDL como na Figura 16.*

Figura 16 – Domínio de Planejamento em PDDL.

```
(:predicates (p) (q) (r))

(:action a1
    :precondition (and (not (p)) (not (q)) (not (r)))
    :effect (and (p)))

(:action a2
    :precondition (and (not (p)) (not (q)) (not (r)))
    :effect (and (r)))

(:action a3
    :precondition (and (not (p)) (not (q)) (r))
    :effect (and (not (r)) (q)))

(:action a4
    :precondition (and (not (p)) (q) (r))
    :effect (and (p)))

(:action a5
    :precondition (and (p) (not (q)) (r))
    :effect (and (q)))
```

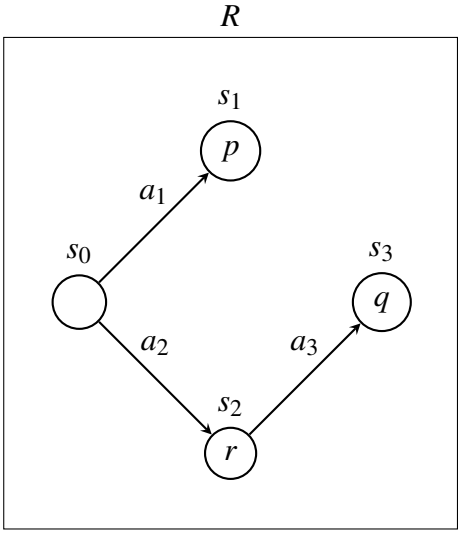
Fonte: Elaborado pelo autor.

**Exemplo 4.0.2** *(Problema de Planejamento) Dado o domínio de planejamento do Exemplo 4.0.1, podemos definir o problema de planejamento* $\Pi$ = <*D*, $s_0$, $p \wedge q \wedge r$> *em que* $s_0$ *é o estado inicial e a meta de planejamento é alcançar o estado* $s_7$*, onde a fórmula* $p \wedge q \wedge r$ *é verdadeira.*

Realizando-se uma busca progressiva a partir do estado inicial $s_0$, obtemos o conjunto **R** de todos os estados alcançáveis a partir do estado inicial, como pode ser observado na Figura 17. Podemos observar que este problema não possui solução, uma vez que $s_7 \notin \mathbf{R}$. Além disso, também podemos executar uma busca regressiva a partir do estado $s_7$ que satisfaz a meta e obter todos os estados que alcançam o estado meta, como pode ser observado na Figura 18. Da mesma forma observamos que o problema não possui solução, pois $s_0 \in \mathbf{U}$.

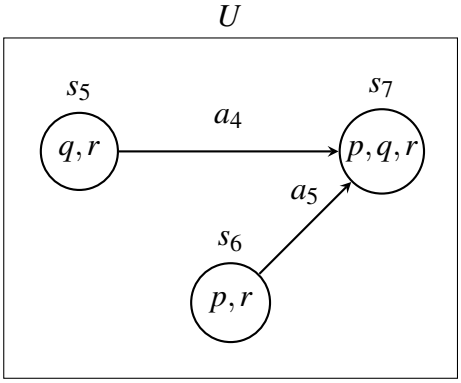
A Figura 19 ilustra o espaço de estados com os conjuntos **R** e **U**. Ademais, há estados que não são alcançados pelas ações do domínio a partir do estado inicial ou meta, como é o caso do estado $s_4$ que não pertence a **R** e nem a **U**.

Figura 17 – Conjunto **R** do problema do Exemplo 4.0.2.




Fonte: Elaborado pelo autor.

Figura 18 – Conjunto **U** do problema do Exemplo 4.0.2.




Fonte: Elaborado pelo autor.

Para encontrar os estados minimais, calculamos as distâncias entre os estados de **R** e **U**, usando a distância de Hamming, conforme mostrado na Tabela 1. Por exemplo, a distância entre os estados $s_1$ e $s_7$ é 2 pois duas das proposições desses estados possuem valores diferentes.

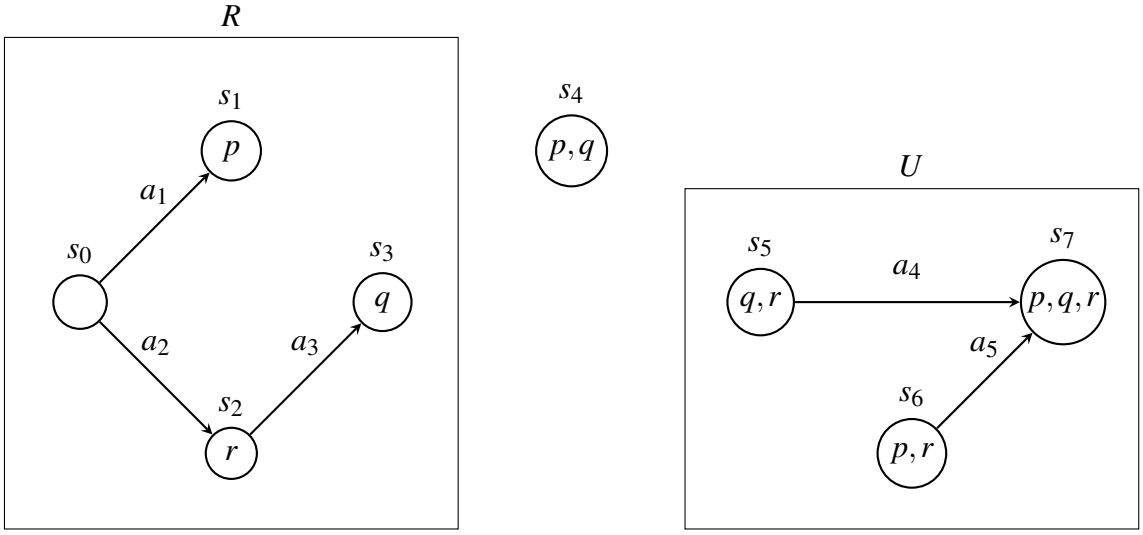
Os estados de **R** que vão fazer parte de $R_a$ são aqueles que possuem distância $d$ minimal para algum estado de **U**. Observando a Tabela 1, os estados de **R** que possuem distância minimal para estados de **U** são: $s_1$, $s_2$ e $s_3$. Assim $R_a = \{s_1, s_2, s_3\}$.

Após encontrar $R_a$ é feita a revisão de $R_a$ por **U** para encontrar o conjunto $U_a$ de estados de **U** mais próximos de $R_a$. As distâncias entre os estados de **U** e $R_a$ podem ser vistas na Tabela 2.

Observando esta tabela, os estados de **U** que possuem distância minimal para estados de $R_a$ são $s_5$ e $s_6$. Assim $U_a = \{s_5, s_6\}$.

Após os conjuntos $R_a$ e $U_a$ serem encontrados a ação nova $a_{nova}$ é criada. Essa ação

Figura 19 – Conjuntos **R** e **U** do problema de planejamento do Exemplo 4.0.2.




Fonte: Elaborado pelo autor.

Tabela 1 – Tabela com a distância de Hamming entre os estados dos conjuntos **R** e **U**.

| $R$ | $U$ | $d = h(r,u)$ |
|---|---|---|
| $s_0 = \{\}$ | $s_5 = \{q,r\}$ | 2 |
| | $s_6 = \{p,r\}$ | 2 |
| | $s_7 = \{p,q,r\}$ | 3 |
| $s_1 = \{p\}$ | $s_5 = \{q,r\}$ | 3 |
| | $s_6 = \{p,r\}$ | 1 |
| | $s_7 = \{p,q,r\}$ | 2 |
| $s_2 = \{r\}$ | $s_5 = \{q,r\}$ | 1 |
| | $s_6 = \{p,r\}$ | 1 |
| | $s_7 = \{p,q,r\}$ | 2 |
| $s_3 = \{q\}$ | $s_5 = \{q,r\}$ | 1 |
| | $s_6 = \{p,r\}$ | 3 |
| | $s_7 = \{p,q,r\}$ | 2 |

Fonte: Elaborado pelo autor.

possui precond($a_{nova}$) = $\{R_a\}$ e efeitos($a_{nova}$) = $\{U_a\}$. Com isso ela conecta os conjuntos **R** e **U**, tornando assim o problema solucionável.

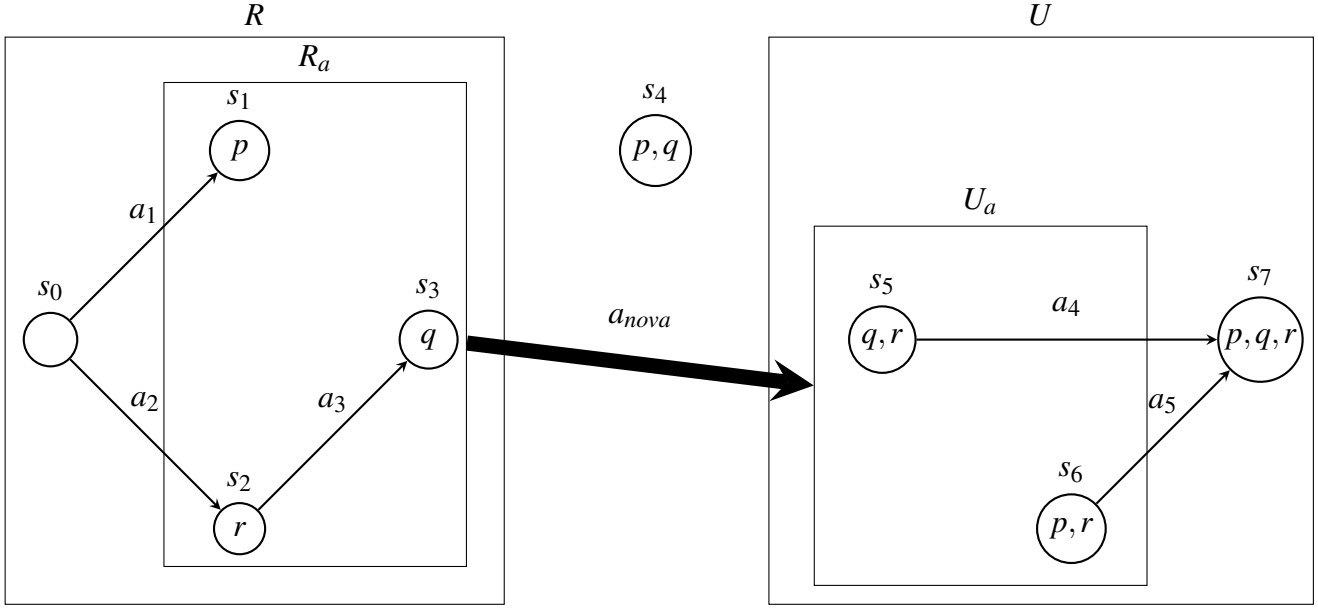
Observe que a ação $a_{nova}$ criada por este método possui suas precondições e seus efeitos dados por conjuntos de estados, temos que: precond($a_{nova}$) = $\{s_1, s_2, s_3\}$ e efeitos($a_{nova}$) = $\{s_5, s_6\}$. É importante ressaltar que as precondições e os efeitos da ação $a_{nova}$ não são dados por conjuntos de proposições, como devem ser as ações PDDL. Neste trabalho iremos propor um método que a partir dos conjuntos $R_a$ e $U_a$, extrai um conjunto de ações PDDL para tornar o problema de planejamento sem solução solucionável.

Tabela 2 – Tabela com a distância de Hamming entre os estados dos conjuntos **U** e $R_a$.

| $U$ | $R_a$ | $d = h(u, r_a)$ |
|---|---|---|
| $s_5 = \{q, r\}$ | $s_1 = \{p\}$ | 3 |
| | $s_2 = \{r\}$ | 1 |
| | $s_3 = \{q\}$ | 1 |
| $s_6 = \{p, r\}$ | $s_1 = \{p\}$ | 1 |
| | $s_2 = \{r\}$ | 1 |
| | $s_3 = \{q\}$ | 3 |
| $s_7 = \{p, q, r\}$ | $s_1 = \{p\}$ | 2 |
| | $s_2 = \{r\}$ | 2 |
| | $s_3 = \{q\}$ | 2 |

Fonte: Elaborado pelo autor.

Figura 20 – Problema de planejamento com a ação $a_{nova}$ conectando $R_a$ e $U_a$.




Fonte: Elaborado pelo autor.

# 5 MÉTODO DE INCLUSÃO DE NOVAS AÇÕES COM MUDANÇAS MINIMAIS

O método proposto neste trabalho cria novas ações a partir dos conjuntos $R_a$ e $U_a$. Primeiro, são escolhidos todos os pares de estados <$s_r$, $s_u$>, onde $s_r \in R_a$ e $s_u \in U_a$, que possuem distância $d = \text{h}(s_r, s_u)$ minimal. Depois, baseado em cada um desses pares é criada uma ação nova. As precondições de cada ação nova serão as proposições do estado $s_r$ do par e as proposições que pertencem ao estado $s_u$ mas que não pertencem ao estado $s_r$ do par vão compor os efeitos das ações novas. Essas ações serão inseridas no domínio de planejamento do problema sem solução, tornando ele solucionável.

Vamos ilustrar esse método usando o problema sem solução do Exemplo 4.0.2. Primeiro, vamos enumerar os possíveis pares de estados de $R_a$ e $U_a$, obtendo:

- <$s_1$, $s_6$>
- <$s_2$, $s_5$>
- <$s_2$, $s_6$>
- <$s_3$, $s_5$>

Baseado em cada um desses pares é criada uma ação nova. As proposições do estado que pertence a $R_a$ vão compor as precondições da ação nova e as proposições que pertencem ao estado de $U_a$, mas não ao estado de $R_a$, vão compor os efeitos da ação nova. Com isso, as ações criadas são:

- $a^1_{nova}$, onde precond($a^1_{nova}$) = $\{p, \neg q, \neg r\}$ e efeitos($a^1_{nova}$) = $\{r\}$;
- $a^2_{nova}$, onde precond($a^2_{nova}$) = $\{\neg p, \neg q, r\}$ e efeitos($a^2_{nova}$) = $\{q\}$;
- $a^3_{nova}$, onde precond($a^3_{nova}$) = $\{\neg p, \neg q, r\}$ e efeitos($a^3_{nova}$) = $\{p\}$;
- $a^4_{nova}$, onde precond($a^4_{nova}$) = $\{\neg p, q, \neg r\}$ e efeitos($a^4_{nova}$) = $\{r\}$;

Cada uma dessas ações é uma ação PDDL e conecta os conjuntos **R** e **U**, tornando o problema solucionável, como pode ser visto na Figura 22.

Esse método propõe ações que realizam "pequenas" modificações nas proposições que descrevem o mundo. Note que preferimos ações do tipo

$$\text{precond}(a_1) = \{p, \neg q, \neg r\} \text{ e efeitos}(a_1) = \{r\}$$

A ações do tipo

$$\text{precond}(a_2) = \{p, \neg q, \neg r\} \text{ e efeitos}(a_2) = \{\neg p, q, r\}$$

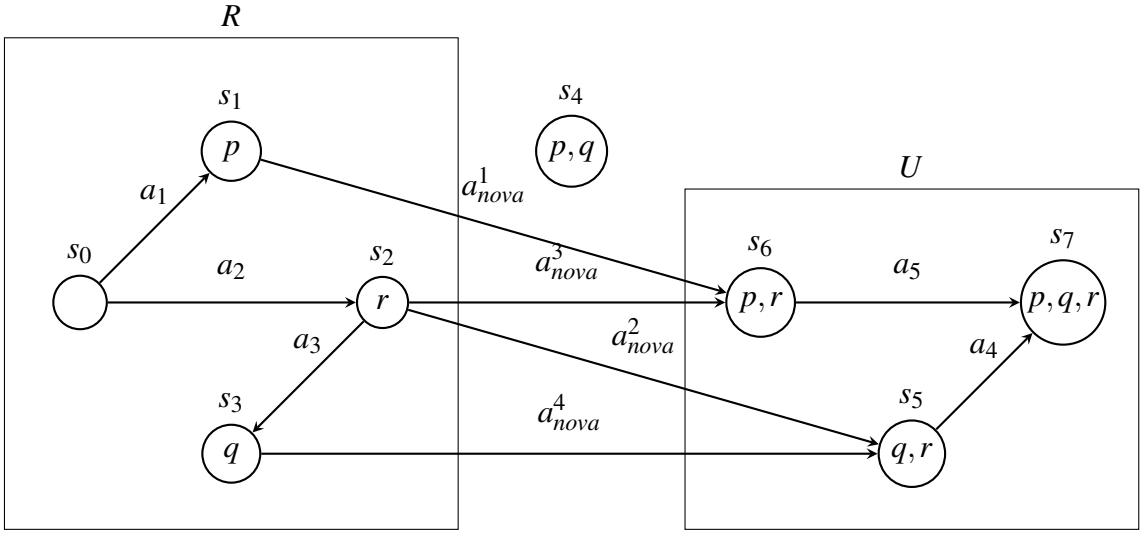
Ou seja, ações que modificam minimamente valores das proposições do problema de planejamento.

Figura 21 – Ações novas em PDDL.

```
(:predicates (p) (q) (r))

(:action a^1_nova
    :precondition (and (p) (not (q)) (not (r)))
    :effect (and (r)))

(:action a^2_nova
    :precondition (and (not (p)) (not (q)) (r))
    :effect (and (q)))

(:action a^3_nova
    :precondition (and (not (p)) (not (q)) (r))
    :effect (and (p)))

(:action a^4_nova
    :precondition (and (not (p)) (q) (not (r)))
    :effect (and (r)))
```

Fonte: Elaborado pelo autor.

Figura 22 – Conjuntos **R** e **U** do problema de planejamento conectados pelas ações novas.




Fonte: Elaborado pelo autor.

Para um projetista de um domínio que se deparou com uma anomalia, representada pelo problema sem solução, este método pode auxiliá-lo no processo de modificação do conjunto de ações do domínio.

No decorrer do processo de elaboração deste método observamos que outros tipos de mudanças minimais poderiam ser interessantes para o projetista de um domínio. Por exemplo, mudanças que adicionam ações que sejam mais próximas de ações já existentes no domínio. Estas novas ações podem ser construídas a partir da mudança das precondições e/ou efeitos

das ações existentes. Acreditamos que essa mudança também possa ser caracterizada como um problema de revisão de crenças.

# 6 CONCLUSÃO

Problemas de planejamento sem solução são aqueles em que não é possível sair do estado inicial e chegar em um estado que satisfaz a meta. Neste trabalho partimos da hipótese de que um problema não possui solução devido ao fato de seu conjunto de ações ter sido projetado de forma errada. Com isso, propomos um método de mudança do conjunto de ações que é uma extensão do método proposto por (MENEZES, 2014), o qual obtém uma nova ação a ser inserida no domínio com a abordagem formal de revisão de crenças (ALCHOURRÓN *et al.*, 1985). Esta nova ação tornara o problema solucionável por conectar os conjuntos **R** e **U**. No entanto, o método de (MENEZES, 2014) não gera ações PDDL. Este trabalho propõe a geração de ações PDDL a partir da ação $a_{nova}$ dada pelo método de (MENEZES, 2014).

As dificuldades de usar a mudança de ações para tornar um problema de planejamento solucionável são garantir que as ações novas criadas são adequadas ao domínio do problema e que essas ações realmente tornam o problema solucionável. Uma ação é adequada ao domínio de um problema quando é possível executar essa ação no "mundo real" no contexto desse domínio. Também não basta a ação nova ser adequada, ela precisa tornar o problema solucionável, ou seja, ela precisa assegurar que com sua inclusão no conjunto de ações do domínio é possível sair do estado inicial do problema e chegar em um estado meta.

O método proposto neste trabalho resolve essas dificuldades das seguintes formas: possibilidade de criar múltiplas ações novas, ações novas com efeitos minimais e ações novas que sempre irão tornar o problema solucionável. A desvantagem desse método é que não há como garantir que as ações novas são adequadas ao domínio.

No decorrer do desenvolvimento deste trabalho, outras soluções de modificação de ações foram pensadas. Modificações estas que tornam o problema solucionável a partir da mudança das precondições e/ou efeitos de ações existentes. Além disso, outras métricas podem ser utilizadas para escolher os estados $R_a$ e $U_a$ do método de (MENEZES, 2014).

Como trabalhos futuros, temos:

- a implementação do método de mudança de ações no arcabouço de mudanças de problemas sem solução;
- a avaliação da qualidade das ações propostas pelo método deste trabalho;
- explorar métodos de mudança de ações diferentes que tornam problemas sem solução solucionáveis.

Na literatura, há apenas o método de Göbelbecker (GÖBELBECKER *et al.*, 2010)

que insere ações no domínio a fim de tornar um problema sem solução solucionável. Este método insere ações que modificam o valor de apenas uma proposição. As ações, chamadas de ações fictícias, são do tipo: precond($a^i_{fic}$) = $\{p\}$, efeitos($a^i_{fic}$) = $\{\neg p\}$.

O método de Göbelbecker não foi proposto com o sentido de modificar ações mas de mudar o estado inicial de um problema sem solução. Após o estado inicial que torna o problema solucionável é encontrado, as ações fictícias que foram inseridas no domínio são descartadas.

# REFERÊNCIAS

ALCHOURRÓN, C. E.; GÄRDENFORS, P.; MAKINSON, D. On the logic of theory change: Partial meet contraction and revision functions. **The journal of symbolic logic**, Cambridge University Press, v. 50, n. 2, p. 510–530, 1985.

BACCHUS, F. Aips 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems. **Ai magazine**, v. 22, n. 3, p. 47, 2001.

BAIER, J. A.; FRITZ, C.; MCILRAITH, S. A. Exploiting procedural domain control knowledge in state-of-the-art planners. In: **ICAPS**. [S.l.: s.n.], 2007. p. 26–33.

BARTÁK, R.; MAILLARD, A. Attribute grammars with set attributes and global constraints as a unifying framework for planning domain models. In: ACM. **Proceedings of the 19th International Symposium on Principles and Practice of Declarative Programming**. [S.l.], 2017. p. 39–48.

EROL, K.; HENDLER, J.; NAU, D. S. Complexity results for htn planning. **Annals of Mathematics and Artificial Intelligence**, Springer, v. 18, n. 1, p. 69–93, 1996.

EROL, K.; HENDLER, J. A.; NAU, D. S. **Semantics for hierarchical task-network planning**. [S.l.], 1995.

GÄRDENFORS, P. **Knowledge in flux:** Modeling the dynamics of epistemic states. [S.l.]: The MIT press, 1988.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning:** theory and practice. [S.l.]: Elsevier, 2004.

GÖBELBECKER, M.; KELLER, T.; EYERICH, P.; BRENNER, M.; NEBEL, B. Coming up with good excuses: What to do when no plan can be found. **Cognitive Robotics**, v. 10081, 2010.

HERZIG, A.; MENEZES, M. V. de; BARROS, L. N. de; WASSERMANN, R. On the revision of planning tasks. In: **ECAI**. [S.l.: s.n.], 2014. p. 435–440.

LONG, D.; FOX, M. The 3rd international planning competition: Results and analysis. **Journal of Artificial Intelligence Research**, v. 20, p. 1–59, 2003.

LONG, D.; FOX, M.; HOWEY, R. Planning domains and plans: validation, verification and analysis. In: **Proc. Workshop on V&V of Planning and Scheduling Systems**. [S.l.: s.n.], 2009.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. Pddl-the planning domain definition language. [S.l.], 1998.

MENEZES, M. V. de. **Mudanças em Problemas de Planejamento sem Solução**. Tese (Doutorado) — Universidade de São Paulo, 2014.

PLANNING, I. C. on A.; SCHEDULING. **International Planning Competition**. 2002. Disponível em: http://ipc02.icaps-conference.org/. Acessado em: 08 maio 2018.

PLANNING, I. C. on A.; SCHEDULING. **International Competition on Knowledge Engineering for Planning and Scheduling**. 2012. Disponível em: http://icaps12.icaps-conference.org/ickeps.html. Acessado em: 16 jun. 2018.

PLANNING, I. C. on A.; SCHEDULING. **Unsolvability International Planning Competition**. 2016. Disponível em: https://unsolve-ipc.eng.unimelb.edu.au/. Acessado em: 08 maio 2018.

PLANNING, I. C. on A.; SCHEDULING. **International Planning Competition**. 2018. Disponível em: https://ipc2018.bitbucket.io/. Acessado em: 08 maio 2018.

SHAH, M.; CHRPA, L.; JIMOH, F.; KITCHIN, D.; MCCLUSKEY, T.; PARKINSON, S.; VALLATI, M. Knowledge engineering tools in planning: State-of-the-art and future challenges. **Knowledge Engineering for Planning and Scheduling**, v. 53, 2013.

SIMPSON, R. M.; KITCHIN, D. E.; MCCLUSKEY, T. L. Planning domain definition using gipo. **The Knowledge Engineering Review**, Cambridge University Press, v. 22, n. 2, p. 117–134, 2007.

VAQUERO, T. S.; ROMERO, V.; TONIDANDEL, F.; SILVA, J. R. itsimple 2.0: An integrated tool for designing planning domains. In: **ICAPS**. [S.l.: s.n.], 2007. p. 336–343.

VAQUERO, T. S.; SILVA, J. R.; BECK, J. C. A brief review of tools and methods for knowledge engineering for planning & scheduling. **KEPS 2011**, p. 7, 2011.

WICKLER, G. Using static graphs in planning domains to understand domain dynamics. **Knowledge Engineering for Planning and Scheduling**, Citeseer, v. 69, 2013.

WINSLETT, M. **Updating logical databases**. [S.l.]: Cambridge University Press, 2005. v. 9.