UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

WARLLES CARLOS COSTA MACHADO

ESPECIFICANDO PREFERÊNCIAS SOBRE POLÍTICAS EM PROBLEMAS DE PLANEJAMENTO NÃO DETERMINÍSTICOS USANDO A LÓGICA ALPHA-CTL.

QUIXADÁ

2022

WARLLES CARLOS COSTA MACHADO

ESPECIFICANDO PREFERÊNCIAS SOBRE POLÍTICAS EM PROBLEMAS DE PLANEJAMENTO NÃO DETERMINÍSTICOS USANDO A LÓGICA ALPHA-CTL.

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Computação. Área de Concentração: Ciência da Computação

Orientadora: Profa. Dra. Maria Viviane de Menezes

QUIXADÁ

2022

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

M135e Machado, Warlles Carlos Costa.
Especificando preferências sobre políticas em problemas de planejamento não determinísticos usando a lógica alpha-CTL / Warlles Carlos Costa Machado. – 2022.
75 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2022.
Orientação: Prof. Dra. Maria Viviane de Menezes.

1. Planejamento Automatizado. 2. Inteligência Artificial. 3. Benchmarks (Computação). 4. Lógica Temporal. 5. Algoritmos. I. Título.

CDD 005

WARLLES CARLOS COSTA MACHADO

ESPECIFICANDO PREFERÊNCIAS SOBRE POLÍTICAS EM PROBLEMAS DE PLANEJAMENTO NÃO DETERMINÍSTICOS USANDO A LÓGICA ALPHA-CTL.

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Computação. Área de Concentração: Ciência da Computação

Aprovada em: __/__/____

BANCA EXAMINADORA

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

Prof. Dr. Davi Romero de Vasconcelos
Universidade Federal do Ceará (UFC)

Prof. Dra. Leliane Nunes de Barros
Universidade de São Paulo (USP)

Á Deus, que sempre esteve ao meu lado. À minha minha família, que me inspira todos os dias.

AGRADECIMENTOS

À Deus, por ser meu refúgio e minha força durante o desenvolvimento desse trabalho.

À professora Viviane, por seus ensinamentos, pela dedicação e paciência na orientação durante a produção deste trabalho.

À minha esposa, Mônica, pela paciência, dedicação e carinho que foram fundamentais, principalmente, na conclusão final desse trabalho. Sua abdicação do emprego para ficar ao meu lado contribuiu significativamente para conclusão desse trabalho.

Aos professores Davi Romero e Paulo de Tarso pela ajuda inestimável em disciplinas do mestrado.

Finalmente, à minha amiga Axia, pelos momentos de descontração e apoio emocional que transformaram situações estressantes em momentos motivantes.

“A única forma de encontrar os limites do possí-
vel é indo além dele, para dentro do impossível.”
(Arthur C. Clark)

RESUMO

Sistemas computacionais capazes de planejar e executar ações, sem a necessidade de um operador humano, são uma realidade em nosso cotidiano. Planejamento automatizado é uma subárea da Inteligência Artificial que estuda o desenvolvimento de algoritmos capazes de planejar tarefas de forma autônoma. Esses algoritmos recebem como entrada uma descrição formal do ambiente e devolve um conjunto de ações que levam o agente de um estado inicial a um estado meta. A fim de simplificar a elaboração desses algoritmos, a abordagem clássica supõem que o ambiente evolui de forma determinística. Neste trabalho, abordamos duas evoluções da abordagem clássica que a aproxima de muitas situações reais: (i) a que considera incertezas nos efeitos das ações do agente e; (ii) a que permite que o usuário imponha preferências na trajetória para o alcance da meta. Propomos fórmulas da lógica temporal de ações α-CTL para expressar tanto as preferências qualitativas do usuário como a qualidade das políticas obtidas. Utilizamos o arcabouço de verificação de modelos da lógica α-CTL para obter soluções para o domínio *benchmark* Rovers, que modela a exploração de um robô em Marte.

Palavras-chave: Planejamento Automatizado. Inteligência Artificial. Benckmarks. Lógica Temporal. Algoritmos

ABSTRACT

Computer systems capable of planning and executing actions, without the need for a human operator, are a reality in our days. Automated Planning is a subarea of Artificial Intelligence that concerns about the development of algorithms capable of planning tasks in an autonomous way. These algorithms receive as input a formal description of the environment and return a set of actions that take the agent from an initial state to a goal state. In order to simplify the construction of these algorithms, the classical approach assumes that the environment evolves in a deterministic way. In this work, we approach two evolutions of the classic planning that bring it closer to many real situations: (i) the one that deals with non-deterministic effects of agent's actions and; (ii) the one that allows users impose preferences on the trajectory to reach the goal. We propose formulae on α-CTL logic that captures both user's preferences and quality of the solutions (weak, strong and strong-cyclic solutions). We use the the model checking framework based on α-CTL to solve problems on the benchmark domain Rovers, which models situation on .

Keywords: Automated Planning. Artificial Intelligence. Benckmarks. Temporal Logic, Algorithms

LISTA DE FIGURAS

LISTA DE TABELAS

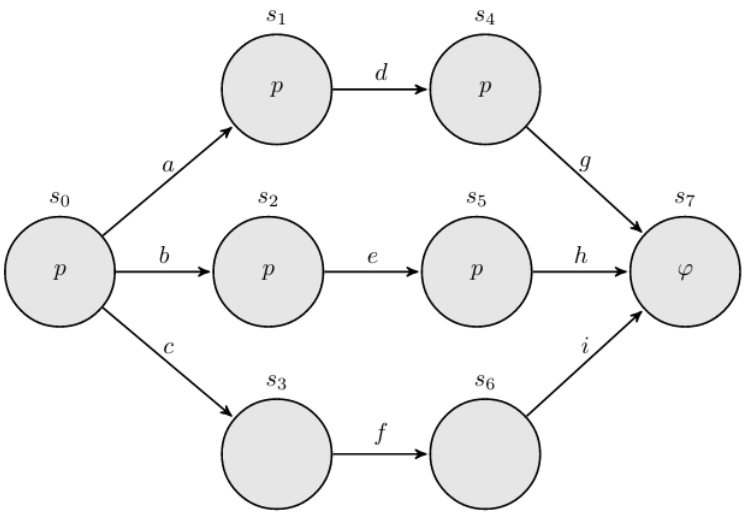
SUMÁRIO

1 INTRODUÇÃO

Sistemas computacionais, capazes de planejar e executar ações sem a necessidade de um operador humano, são uma realidade em nossas vidas. De fato, a criação de modelos computacionais que permitem automatizar o processo de *planejamento* é o objeto de estudo de uma subárea da *Inteligência Artificial* chamada de *Planejamento Automatizado* (GHALLAB *et al.*, 2004).

O avanço em pesquisas nessa área têm possibilitado o desenvolvimento de algoritmos eficientes capazes de planejar tarefas de forma autônoma. Nesse contexto, a tarefa a ser planejada pelo algoritmo é chamada de *problema de planejamento* e é definido por meio de um *domínio de planejamento*, um *estado inicial* e uma *meta* a ser alcançada. O *domínio de planejamento* modela o ambiente e sua evolução com base nas ações que o agente pode executar. Enquanto o *estado* corresponde a uma configuração particular do ambiente em um dado instante de tempo. Por sua vez, o *estado inicial* descreve uma configuração particular do ambiente e do agente no momento em que o agente inicia a execução da tarefa. Uma *ação* é uma manifestação do agente sobre o ambiente fazendo-o evoluir do estado atual para outro estado. Uma *ação* consiste na materialização da vontade do agente em modificar o ambiente a fim de alcançar seu objetivo.

Figura 1 – Domínio de planejamento representado por um sistema de transição de estados.



Fonte: Elaborado pelo autor.

Domínios de planejamento podem ser representados por sistemas de transição de estados, como o que pode ser observado na Figura 1. Nesse sistema, os vértices representam os estados possíveis que o ambiente poderá assumir e são rotulados por propriedades (*i.e., p, t* e *s*) que descrevem aspectos do mundo real. As arestas são rotuladas com ações (*i.e., a, b* e *c*).

As ações descrevem as transições de estados no ambiente e, consequentemente, a dinâmica do ambiente.

Ambiente determinístico. Encontrar um *plano* para um *problema de planejamento* é computacionalmente difícil (HOFFMANN, 2011). A fim de simplificar este processo, diversas abordagens de planejamento impõem restrições ou fazem suposições sobre o ambiente. A *abordagem determinística* supõe que o ambiente evolui de uma forma em que não há incertezas sobre os efeitos das ações do agente (GHALLAB *et al.*, 2004). Assim, na abordagem determinística, apenas um único estado é obtido a partir da execução de uma ação.

Exemplo 1 (Problema de planejamento na abordagem determinística) *No domínio na Figura 1, suponha que o agente esteja no estado inicial s_0 e que sua meta seja alcançar um estado em que a propriedade φ seja verdadeira. A sequência de ações $\langle a,d,g\rangle$ corresponde a um plano, pois quando executadas levam o agente do estado s_0 para o estado s_7. Outros possíveis planos são: $\langle c,f,i\rangle$ e $\langle b,e,h\rangle$.*

Os principais algoritmos capazes de obter um plano em ambientes determinísticos são baseados em busca heurística no espaço de estados (HOFFMANN, 2001; HELMERT, 2006; RICHTER *et al.*, 2011; XIE *et al.*, 2014) e em satisfazibilidade booleana (KAUTZ; SELMAN, 1999).

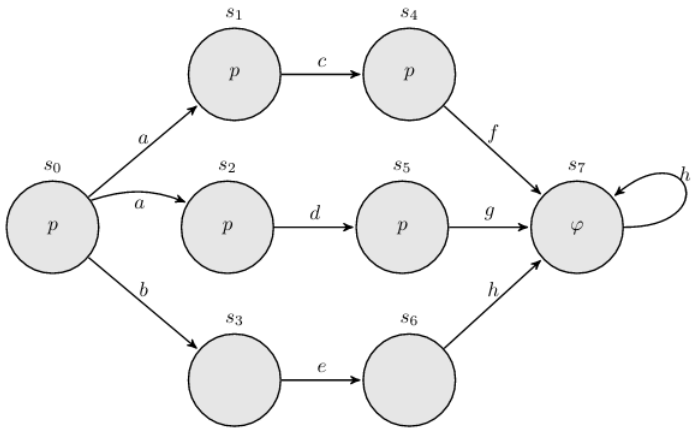
Ambiente não determinístico. Em situações práticas o ambiente não evolui de forma determinística, isto é, a suposição de que cada ação leva a um único estado sucessor, não se mostra adequada. De fato, devido à ocorrência de eventos da natureza sobre os quais o agente não tem controle, os chamados *eventos exógenos*, durante a execução de uma ação, não é possível o agente prever com certeza qual será seu próximo estado. Assim, na abordagem não-determinística devemos considerar que a execução de uma ação pode levar a um conjunto de possíveis estados futuros. Um domínio de planejamento não determinístico expressa as incertezas do ambiente com relação à atuação da natureza (PEREIRA; BARROS, 2008).

Uma solução para um problema de planejamento não determinístico não poder ser uma sequência de ações, uma vez que uma ação poderá levar a um conjunto de possíveis estados. Assim, em ambientes não determinístico, as soluções para os problemas são mapeamentos de estados em ações. Esse mapeamento é chamado *política* (GHALLAB *et al.*, 2004). Soluções para problemas de planejamento não determinístico podem ser categorizados em três tipos de classes de acordo com as várias possibilidades de execução da política: uma solução *fraca* corresponde

a planos que poderão alcançar a meta, mas que não há garantia devido ao não determinismo; uma solução *forte* corresponde ao plano que garante que a meta seja alcançada, apesar do não determinismo; e uma solução *forte-cíclica* garante que a meta seja alcançada porém sua execução pode conter ciclos infinitos (sob a suposição que o agente eventualmente consiga sair de todos os ciclos) (CIMATTI *et al.*, 2003). Assim, para cada possível estado que o agente pode alcançar, é definido que ação ele deve tomar para se chegar a um estado meta.

Exemplo 2 (Problema de planejamento na abordagem não-determinística) *Considere o domínio de planejamento não determinístico representado na Figura 2. Esse domínio de planejamento é bem semelhante ao domínio da Figura 1, exceto pelo fato de que a ação a é não determinística. Assim, quando esta ação é executada, pode levar o agente do estado* s_0 *ao estado* s_1 *ou ao estado* s_2*. Suponha que o agente esteja no estado* s_0 *e que sua meta seja alcançar o estado* s_7 *em que a propriedade* φ *é verdadeira. Para esse problema, uma possível política é:* $\{(s_0,a),(s_1,c),(s_4,f),(s_2,d),(s_5,g)\}$*. Essa política informa que partindo do estado* s_0 *e executando a ação não determinística a, o agente estará em* s_1 *ou* s_2*; executando a ação determinística c, a partir de* s_1*, o agente estará em* s_4 *e executando a ação* f*, a partir de* s_4 *o agente alcançará o estado* s_7 *com a propriedade* φ*. Uma outra política que pode ser obtida é:* $\{(s_0,b),(s_3,e),(s_6,h)\}$.

Figura 2 – Domínio de planejamento não determinístico.



Fonte: Elaborado pelo autor.

Os principais algoritmos para solucionar problemas em ambientes de planejamento não-determinístico são baseado em verificação de modelos com lógicas temporais (CIMATTI *et al.*, 2003) , (PEREIRA, 2007), (MENEZES, Maria Viviane de, 2014), (SANTOS *et al.*, 2019) e

no processo de *determinização* de ações para obtenção de políticas relevantes (MUISE *et al.*, 2012; MUISE *et al.*, 2014).

Planejamento com Preferências. Outro aspecto relevante em agentes inteligentes é a habilidade de decidir como atuar para alcançar seu objetivo. Encontrar uma solução que leve o agente ao estado desejado é fundamental. Entretanto, em muitas situações o usuário deseja especificar alguma propriedade que deverá ser preservada ou evitada ao longo do caminho para alcance da meta.

Exemplo 3 (Planos com preferências) *Considere o domínio de planejamento da Figura 1. Suponha que o agente esteja inicialmente no estado s_0 e que sua meta seja alcançar um estado em que a propriedade φ é satisfeita, mas com a* ***restrição de passar apenas por estados em que a propriedade*** *p* ***é verdadeira****. O plano $\langle a,d,g\rangle$ e $\langle b,e,h\rangle$ são soluções possíveis para este problema, uma vez que representam caminhos em que a meta φ é alcançada e todos os estados ao longo do caminho satisfazem a propriedade p.*

Algoritmos capazes de gerar soluções que atendam as preferencias do usuário são usualmente algoritmos para domínios de planejamento determinísticos (HSU *et al.*, 2007; BAIER *et al.*, 2007; KIM *et al.*, 2017; PERCASSI; GEREVINI, 2019). Agora, considere que queremos obter o mesmo tipo de restrição para o alcance da meta em um ambiente não-determinístico.

Exemplo 4 (Políticas com preferências) *Considere o domínio de planejamento na Figura 2. Suponha que o agente esteja inicialmente no estado s_0 e que sua meta seja alcançar o estado em que φ é verdade, mas com a restrição de passar apenas por estados em que a propriedade p verdadeira. A política $\pi = \{(s_0,a),(s_1,c),(s_4,f),(s_2,d),(s_5,g)\}$ é a única solução possível para este problema, uma vez que representa os caminhos em que a meta φ é alcançada e todos os estados ao longo do caminho satisfazem a propriedade p.*

Nesse trabalho, propomos soluções em domínios de planejamento que combinam duas situações presentes em ambientes práticos a saber:

(i) consideramos que *as ações do agente podem ter efeitos não determinísticos*, isto é, podem ter efeitos incertos;

(ii) consideramos que o usuário deseja impor *preferências na trajetória para o alcance da meta*, considerando situações que devem ser mantidas ou evitadas em uma política.

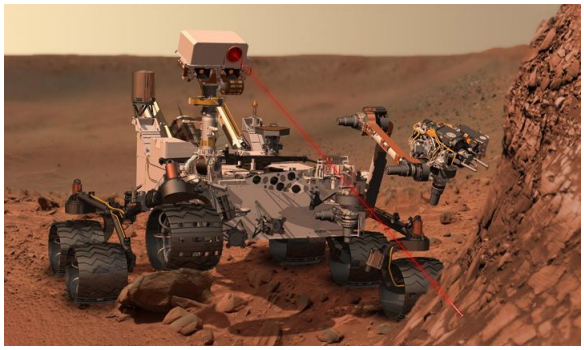
Recentemente, algoritmos de planejamento capazes de obter políticas com preferências foram propostos (CAMACHO *et al.*, 2017; BONET *et al.*, 2019). No entanto, diferente

das abordagens anteriores em ambientes não determinísticos com preferências, expressamos preferências em uma lógica temporal capaz de representar ações em sua semântica. Além disso, neste trabalho propomos como expressar também a qualidade da política obtida: se fraca, forte ou forte-cíclica.

1.1 Motivação

O domínio *Rovers* (Figura 3) é um domínio de exploração planetária inserido como domínio *benchmark* na terceira Competição Internacional de Planejamento IPC (*International Planning Competition*)(LONG; FOX, 2003) . Neste domínio, os robôs são capazes de executar ações tais como navegar entre áreas, obter amostras de solo, obter amostras de rocha, tirar fotografias, calibrar as câmeras fotográficas, analisar as amostras obtidas e transmitir os dados analisados para a estação espacial. Todas estas as ações do domínio original são determinísticas.

Figura 3 – Robô de Marte realizando a obtenção de amostras de rochas



Fonte: wiredscience.

Entretanto, há razões para considerar que um ambiente de exploração espacial evolui de forma *não determinística*. Por exemplo, ao tentar navegar entre áreas, o robô pode encontrar obstáculos que impedem sua locomoção como terrenos muito rochosos ou solo muito arenoso ou ainda, obrigando-o a retornar ao ponto de origem ou mudar de localização destino; ao tentar obter uma amostra de solo, pode ocorrer alguma falha nos dispositivos de coleta; ao tentar obter uma amostra de rocha, o robô pode deparar-se com uma rocha suficiente dura tal que seja necessária várias execuções da ação para que um pedaço da rocha seja obtido; ou ainda, podem ocorrer ruídos na tentativa de comunicação dos dados analisados à estação espacial. Nesse domínio, devido às incertezas do agente em relação à atuação do ambiente, um problema de planejamento pode ter muitas soluções.

O problema de encontrar uma política para um problema de planejamento não

determinístico torna-se mais desafiador e complexo quando **estendemos esse domínio de planejamento com preferências sobre caminhos**. Considere, por exemplo, o robô de Marte (Figura 3). Missões espaciais futuras pretendem trazer amostras de rocha do planeta para análise mais detalhada. Assim é importante que o robô Marte, durante o alcance da meta, também colete e armazene uma amostra de rocha,i.e, o robô deve *sempre* manter a amostra de rocha(vide Seção 2.3). Ademais, outras restrições sobre o caminho poderiam exigir que o robô visite *alguma região no caminho que possa recarregar as baterias*; ou ainda *obter amostras de solo de uma região antes que o robô esteja em outra região (vide Seção 2.3).* Assim, um aspecto fundamental e desafiador é a fim de encontrar políticas preferíveis. Dessa forma, uma questão que motivou o desenvolvimento desse trabalho é a seguinte:

É possível usar o arcabouço de planejamento como verificação de modelos baseado em uma lógica temporal capaz de representar ações em sua semântica para sintetizar políticas preferíveis sobre caminhos em ambientes não determinísticos?

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo desse trabalho é verificar se o arcabouço de planejamento como verificação de modelos na lógica α-CTL é adequado para estabelecer preferências dos usuários nas trajetórias de políticas.

1.2.2 Objetivos específicos

São objetivos específicos desse trabalho:

- Especificar as preferências sobre políticas com fórmulas da lógica α-CTL.
- Implementar algoritmos especializados para as fórmulas especificadas no arcabouço de verificação de modelos α-CTL.
- Realizar experimentos com domínios da IPC (*International Planning Competition*)[1].

[1] <https://www.icaps-conference.org/competitions/>

1.3 Organização do texto

Este trabalho apresenta a seguinte organização: O Capítulo 2 apresenta os conceitos básicos necessários para a compreensão da proposta. O Capítulo 3 descreve alguns trabalhos relevantes da literatura que se relacionam com pelo menos um dos aspectos desta proposta. O Capítulo 4 detalha como as preferências sobre politicas podem ser expressas utilizando a logica α-CTL. O Capítulo 5 apresenta a análise experimental. Finalmente, o Capítulo 6 apresenta as conclusões desse trabalho e direcionamentos para alguns possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentamos a fundamentação teórica do trabalho. Na seção 2.1 introduzimos a abordagem de planejamento determinística. Na Seção 2.2 apresentamos uma abordagem de planejamento para uma forma particular de incertezas na qual ambientes evoluem de forma *não determinística*. Na Seção 2.3, introduzimos a abordagem de planejamento com *preferências* que expressa a noção sobre as diferentes maneiras para se alcançar a meta. Na Seção 2.4, abordamos a técnica de *verificação de modelos*. Na seção 2.5, abordaremos aspectos sintáticos e semânticos de cada uma das seguintes lógicas temporais: Computation Tree Logic (CTL) (EMERSON; CLARKE, 1982), e CTL com ações (α-CTL) (PEREIRA, 2007). Na Seção 2.6 apresentamos uma forma de representação de um domínio de planejamento que minimiza o problema da *explosão de estados* e permite o raciocínio sobre ações não determinísticas. Finalmente, na seção 2.7 apresentamos os algoritmos de planejamento usados nos experimentos.

2.1 Planejamento Determinístico

O processo de automação de uma tarefa de planejamento é influenciado por diversos fatores associados ao ambiente e ao agente que refletem na forma como o agente interage e modifica o ambiente. A fim de simplificar a tarefa de planejamento, a abordagem clássica supõe que o ambiente evolui de forma *determinística*, isto é, não há incertezas sobre os efeitos das ações do agente (GHALLAB *et al.*, 2004).

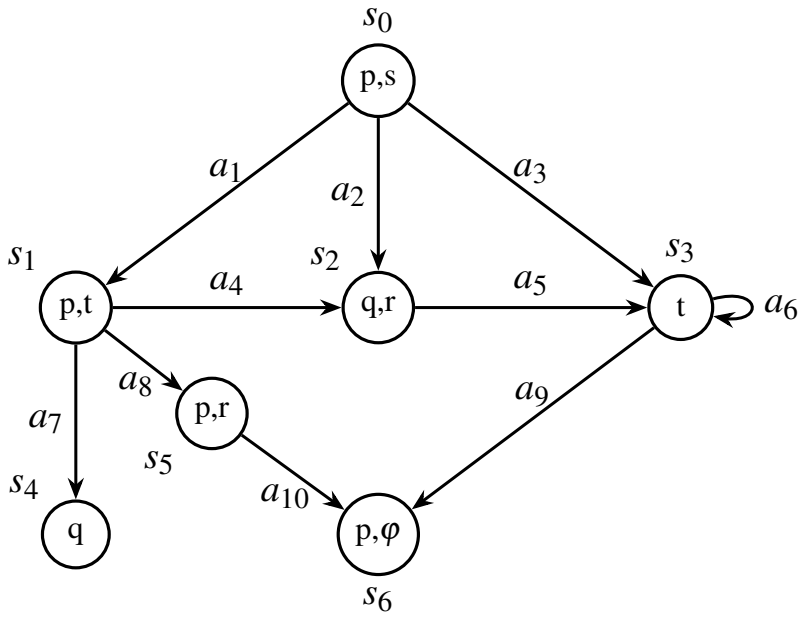
Dado um conjunto de proposições $\mathbb{P}$ que descrevem as características do agente e do ambiente e um conjunto de ações $\mathbb{A}$ que descrevem as habilidades do agente neste ambiente, um *domínio de planejamento* pode ser descrito por um *sistema de transição de estados* em que os estados correspondem às possíveis configurações do mundo e as transições entre os estados são ocasionadas pelas ações. O par $(\mathbb{P}, \mathbb{A})$ é denominado de *assinatura* do domínio de planejamento.

Definição 2.1.1 (Domínio de planejamento) *Dado um conjunto de átomos proposicionais $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, um domínio de planejamento determinístico (GHALLAB et al., 2004) pode ser definido como uma tupla $\mathcal{D} = \langle S, L, T \rangle$ em que os estados são rotulados por elementos de $\mathbb{P}$ e as ações são rotuladas por elementos de $\mathbb{A}$:*

- *$S \neq \varnothing$ é um conjunto finito de estados possíveis do ambiente;*
- *$L : S \mapsto 2^{\mathbb{P}}$ é uma função de rotulação de estados;*
- *$T : S \times \mathbb{A} \to S$ é uma função de transição de estados rotulada por ações.*

A Figura 4 apresenta um sistema de transição de estados sobre um conjunto de proposições $\mathbb{P} = \{p, q, r, s, t\}$ e um conjunto de ações $\mathbb{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$, representando um domínio de planejamento. Neste domínio temos o conjunto de estados $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, rotulados pelos átomos proposicionais em $\mathbb{P}$ e as transições rotuladas pelas ações do conjunto $\mathbb{A}$. Note que as transições são ocasionadas pelas ações, por exemplo, quando a ação a_1 é executada no estado s_0 leva o agente ao estado s_1.

Figura 4 – Um domínio de planejamento representado por um sistema de transição de estados.



Fonte: Elaborado pelo autor.

Em situações práticas, em que a quantidade de estados e transições é muito grande, a representação do domínio de planejamento como um sistema de transição de estados torna-se inviável, uma vez que o número de estados aumenta exponencialmente com o número de proposições, fato conhecido como *problema de explosão de estados*. A fim de contornar esse problema, utilizamos *linguagens de ações* para descrever implicitamente o domínio (por exemplo, PDDL (MCDERMOTT *et al.*, 1998), STRIPS (FIKES; NILSSON, 1971), RDDL (SANNER,)).

Neste trabalho adotamos a linguagem PDDL (*Planning Domain Description Language*) para descrever os domínios de planejamento. Em PDDL, o domínio é descrito por meio de predicados e esquema de ações. Os predicados descrevem de forma compacta as características do ambiente e do agente. Estes serão posteriormente instanciados com objetos do mundo para corresponder aos átomos proposicionais da Definição 2.1.1. As ações, do mesmo modo, são descritas por meio de esquemas que serão instanciados com objetos para corresponder às

transições entre os estados da Definição 2.1.1. A Figura 5 descreve um trecho de código contendo o domínio *Rovers* em PDDL. Inicialmente, são descritos os predicados (`:predicates`). Cada predicado é descrito entre parenteses com o nome do predicado seguido das variáveis associadas a este predicado. Cada variável é antecedida do símbolo `?`. Por exemplo, o predicado unário `(rover ?r)` descreve que a variável `r` é um rover. Já o predicado binário `(calibrated ?c ?r)` descreve que `c` foi calibrada por `r`.

Figura 5 – Um trecho do domínio Rovers em PDDL.

```
(define(domain Rover)
  (:predicates
      (rover ?r)
      (camera ?c)
      (waypoint ?x)
      (calibrated ?c ?r)
       (at_soil_sample ?x)
            ⋮
      (at_rock_sample ?x)
  )
  (:action navigate
    :parameters (?r ?x ?y)
    :precondition (and (at ?r ?x)
                       (visible ?x ?y)
                       (can_traverse ?r ?x ?y)
                       (available ?r))
  (:effect (and ( (at ?r ?y)
                  (not (at ?r ?x)) ))
  )
                              ⋮
```

Fonte: International Planning Competition

Após a descrição dos predicados, temos a descrição das ações (`:action`) por meio de parâmetros, pré-condições e efeitos. Os *parâmetros* são as variáveis do domínio envolvidas na ação. As *pré-condições* são as condições que precisam ser verdadeiras em um estado para que a ação seja executada neste estado. Os efeitos são os fatos que devem ser acrescentados ou removidos do estado atual para obtenção do estado sucessor.

Na Figura 5 temos a descrição da ação *navigate* que move um robô r de uma localização origem x para uma localização destino y. Para que a ação seja executada em um dado estado, as seguintes pré-condições devem ser atendidas: r dever ser um robô (`rover ?r`); x e y devem ser pontos de locomoção `(waypoint ?x)` e `(waypoint ?y)`; o robô deve estar em uma localização x `(at ?r ?x)`; a localização destino y precisa ser visível a partir de x (`visible` $?x$ $?y$); o robô precisa ser capaz de deslocar-se entre as localizações x e y (`can_traverse` $?r$ $?x$ $?y$) e; o robô precisa estar disponível (`available` $?r$). Os efeitos da

ação são: que o robô estará na localização destino *y* (`at ?r ?y`) e que o robô não mais estará na localização origem *x* (`not (at ?r ?y)`). Outras ações deste domínio são: *sample_soil* que coleta uma amostra de solo, *sample_rock* que coleta uma amostra de rocha, *drop* responsável por esvaziar o dispositivo de armazenamento de amostras do robô, *calibrate* responsável por calibrar a câmera do robô, *take_image* responsável por obter imagens em um determinado modo, *communicate_soil_data* realiza a comunicação dos dados obtidos a partir da análise do solo de uma região, *communicate_rock_data* realiza a comunicação dos dados obtidos a partir da análise de rochas de uma região, *communicate_image_data* realiza a comunicação de uma imagem obtida pelo robô a partir de uma região.

Uma vez definido o domínio de planejamento, precisamos formalizar o problema de planejamento descrevendo um estado inicial e a meta.

Definição 2.1.2 (Problema de Planejamento) *Um problema de planejamento determinístico (GHALLAB et al., 2004) é definido por uma tupla* $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ *onde:*

- $\mathcal{D}$ *é o domínio de planejamento;*
- $s_0 \in S$ *é o estado inicial;*
- φ *é uma fórmula da lógica proposicional que representa a meta de planejamento. O conjunto de estados que satisfaz a meta é denominado de conjunto meta* $\mathcal{G}$.

Em PDDL, o problema de planejamento é descrito pela especificação dos objetos que serão utilizados para instanciar os predicados do domínio, pelo estado inicial e pela meta. A Figura 6 descreve um problema de planejamento para o domínio *rovers* em PDDL. Inicialmente, temos o nome do problema *(rover01)* seguido do domínio ao qual o problema está associado *(Rover)*.Na seção *:object*, especificamos os objetos usados como valores dos parâmetros na especificação de predicados e ações no arquivo de domínio: *general* corresponde à estação espacial de suporte ao robô; *colour,high_res e low_res* correspondem, respectivamente, aos modos colorido, alta e baixa resolução para transmissão de uma imagem; *waypoint0,waypoint1,waypoint2,waypoint3* correspondem a regiões da superfície planetária; *camera0* corresponde a câmera do robô e *objetive0, objective1* que correspondem a objetivos do robô. Na seção *:init*, definimos o estado inicial do problema de planejamento por meio das proposições: *at-rover0-waypoint3* que indica que o robô está na região *waypoint3*; *at_soil_sample-waypoint2 e at_soil_sample-waypoint0* indicam respectivamente que o robô tem amostras de solo das regiões *waypoint2* e *waypoint0*; *channel_free-general* especifica que o canal

de comunicação entre o robô e a estação está livre; *empty-rover0store* indica que o local de armazenamento do robô está vazio; *at_rock_sample-waypoint1 e at_rock_sample-waypoint2* indicam, respectivamente, que o robô tem amostra de solo das regiões *waypoint1* e *waypoint2*; *store_of-rover0store-rover0* refere-se ao local de armazenamento do agente *rover0*. O estado inicial é totalmente especificado, ou seja, devemos indicar todos os fatos que são verdadeiros no ambiente. E, por último, temos a seção *:goal* que representa as condições que devem ser satisfeitas no final do plano: as proposições *communicated_rock_data-waypoint3,communicated_soil_data-waypoint2,communicated_image_data-objective1-high_res* indicam, respectivamente, que o agente deverá comunicar os dados de amostras de rochas da região 3, comunicar os dados de amostras de solo da região 2 e comunicar os dados de imagens do objetivo1 em alta resolução. Nesse contexto, a meta de um problema de panejamento é uma especificação parcial, pois precisamos definir apenas o que se deseja alcançar e não todos os fatos de um estado meta.

Figura 6 – Código PDDL para um problema de planejamento do robô de Marte.

```
(define (problem rover01)
 (:domain Rover)
 (:objects
     general,colour,high_res,low_res,rover0,waypoint0,waypoint1,waypoint2,waypoint3,
     camera0,objective0,objective1
 )
 (:init
    at-rover0-waypoint3,at_soil_sample-waypoint2,at_soil_sample-waypoint0,channel_fr-
    ee-general,empty-rover0store,at_rock_sample-waypoint1,at_rock_sample-waypoint3,
    ,at_rock_sample-waypoint2,at_rock_sample-waypoint1,at_soil_sample-waypoint3
 )
 (:goal
    communicated_soil_data waypoint2,communicated_rock_data waypoint3,communicated_
    image_data-objective1-high_res
 )
)
```

Fonte: International Planning Competition

A solução para um problema de planejamento é um plano, conforme mostrado na Definição 2.1.3.

Definição 2.1.3 (Plano) *Dado um problema de planejamento determinístico, um plano* $\pi = \langle a_1, a_2, \cdots, a_n \rangle$ *é uma sequência de ações que quando executada a partir do estado inicial leva o agente a um estado que satisfaça a meta.*

Considere o domínio na Figura 4. Suponha que o agente esteja no estado s_0 e que sua meta seja alcançar o estado em que a propriedade φ seja verdadeira, ou seja, $\mathcal{G} = \{s_6\}$. Uma

possível solução para esse problema de planejamento é a sequência de ações $\langle a_3, a_9 \rangle$, pois define um caminho que leva o agente do estado inicial s_0 até um estado onde a meta é satisfeita.

Um plano para o problema do domínio do robô de Marte (Figura 6) é a sequência de ações: *calibrate-rover0-camera0-objective1-waypoint3, take_image-rover0-waypoint3-objective1-camera0 high_res, communicate_image_data-rover0-general-objective1-high_res-waypoint3-waypoint0, sample_rock rover0-rover0store-waypoint3, communicate_rock_data-rover0-general-waypoint3-waypoint3-waypoint0, drop rover0-rover0store, navigate-rover0-waypoint3-waypoint1, navigate-rover0-waypoint1-waypoint2, sample_soil-rover0-rover0store-waypoint2, communicate_soil_data-rover0-general-waypoint2-waypoint2 waypoint0.*

Embora, na abordagem de clássica, há a suposição de que os efeitos de todas as ações são determinísticos, na prática, a maioria dos ambientes evoluem de forma *não determinística*, isto é, ambientes onde a natureza atua causando *"incertezas"* sobre os efeitos das ações do agente.

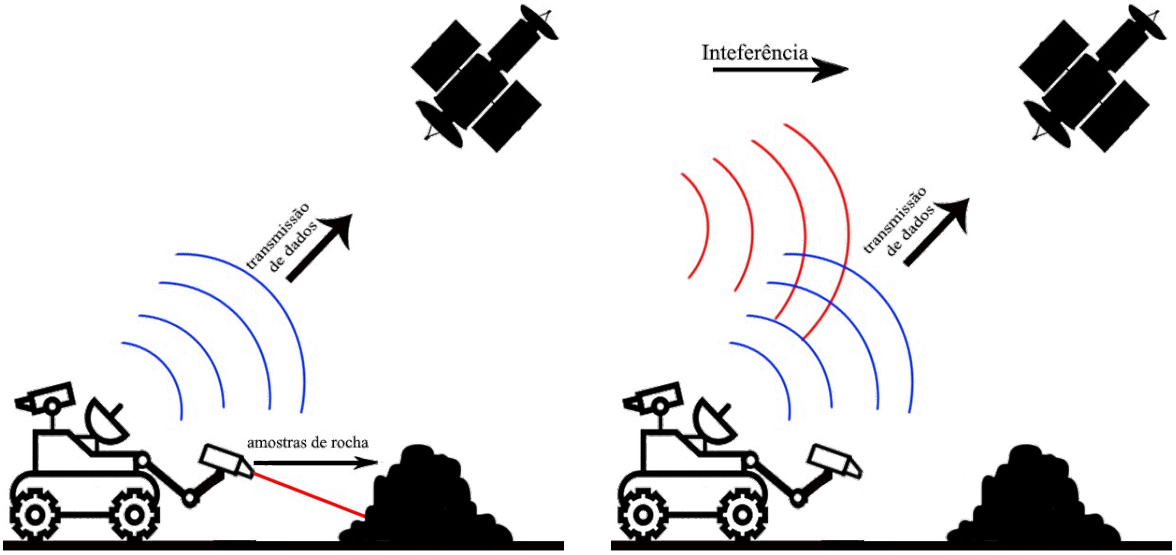
2.2 Planejamento não Determinístico

No mundo real, a natureza atua causando imprevisibilidade nos efeitos das ações do agente. Segundo (PEREIRA, 2007), tais incertezas podem ser modeladas como um jogo entre duas entidades: o *agente* e a *natureza*. O agente, planeja suas ações. A natureza, por sua vez, pode interferir nos efeitos das ações do agente. Assim, quando um agente em um estado s_i, seleciona uma ação a_i, sua intenção é alcançar um dado estado s_j. No entanto, a natureza, ao interferir nos efeitos da ação, pode levar o agente para um outro estado s_k.

Exemplo 5 (Incertezas no ambiente) *Considere o domínio do robô de marte em que o agente, após realizar análise de uma determinada rocha, tenta transmitir os dados para a estação espacial (Figura 7). Considere ainda que ruídos, provenientes da natureza, podem interferir na execução desta ação e, nesse caso, não há garantias de que, após executar a ação de transmissão, os dados serão efetivamente transmitidos para a estação espacial em um estado sucessor.*

A área de planejamento sob incerteza é divida em duas subáreas: *planejamento probabilístico* (SANNER; BOUTILIER, 2006; DELGADO *et al.*, 2011; KELLER; EYERICH, 2012) em que há uma distribuição de probabilidade sobre os efeitos das ações e *planejamento não determinístico* (CIMATTI *et al.*, 2003; PEREIRA; BARROS, 2008; MUISE *et al.*, 2012; MENEZES, Maria Viviane de, 2014; SANTOS, 2019) em que não se define tal distribuição de

Figura 7 – Incertezas no ambiente do domínio do Robô de Marte.



(a) ação transmitir dados. (b) a natureza pode interferir na ação.

Fonte: Elaborado pelo autor.

probabilidade. Neste trabalho, consideramos problemas em que as ações do agente podem ter efeitos não determinísticos.

Um domínio de planejamento não determinístico pode ser caracterizado como um sistema de transição de estados. A diferença, em relação aos domínios determinísticos, é que quando uma ação é executada em um estado pode levar para mais de um estado sucessor.

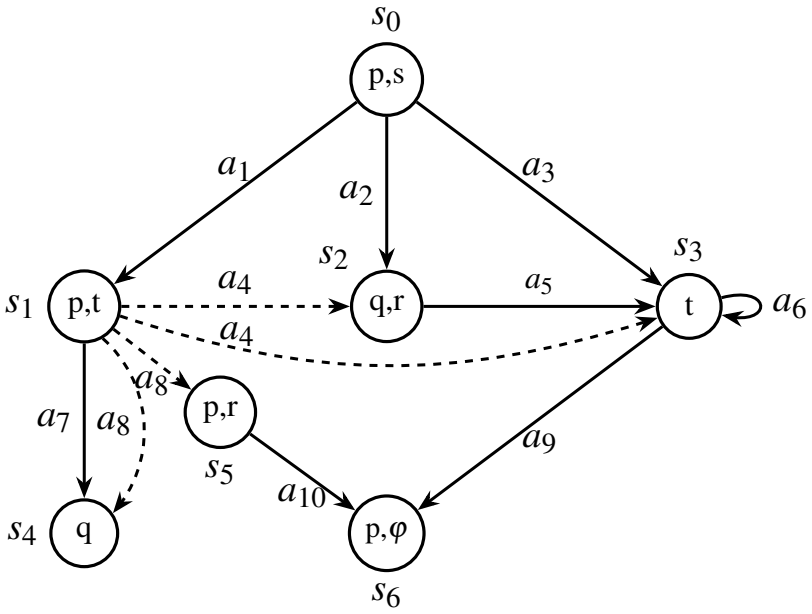
Definição 2.2.1 (Domínio de Planejamento não determinístico) *Dados um conjunto de átomos proposicionais $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, um domínio de planejamento não determinístico é definido por uma tupla $\mathcal{D} = \langle S, L, T \rangle$ em que (GHALLAB et al., 2004) os estados são rotulados por elementos de $\mathbb{P}$ as ações são rotuladas por elementos de $\mathbb{A}$:*

- *S é um conjunto finito de estados;*
- *$L : S \to 2^p$ é uma função de rotulação de estados; e*
- *$T : S \times A \to 2^S$ é uma função de transição de estados não determinística.*

A Figura 8 apresenta um sistema de transição de estados com ações não determinísticas sobre um conjunto de proposições $\mathbb{P} = \{p, q, r, t\}$ e um conjunto de ações $\mathbb{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}\}$, representando um domínio de planejamento não determinístico. Neste domínio temos o conjunto de estados $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$, rotulados pelos átomos proposicionais em $\mathbb{P}$ e as transições rotuladas pelas ações do conjunto $\mathbb{A}$. Entretanto, diferentemente do domínio determinístico, temos as seguintes ações não determinísticas: a ação

a_4, quando executada no estado s_1, pode levar o agente ao estado s_2 ou ao estado s_3 e; a ação a_8, que uma vez executada no estado s_1, pode levar o agente ao estado s_4 ou ao estado s_5.

Figura 8 – Um domínio de planejamento com ações não determinísticas representado por um sistema de transição de estados.



Fonte: Elaborado pelo autor.

Assim como no domínio de planejamento determinístico, a representação explicita de estados e transições de um domínio de planejamento não determinístico, Figura 8, torna-se inviável, pois a quantidade de estados aumenta de forma exponencial com a quantidade de componentes do sistema. Assim, a fim de contornar esse problema, também usamos a linguagem PDDL para descrever o domínio de planejamento não determinístico.

A representação de um domínio de planejamento não determinístico em PDDL é bem semelhante à representação determinística. No entanto, há o uso da palavra reservada **`oneof`** para indicar que um dos efeitos da ação podem ocorrer. Figura 9 apresenta a versão não determinística para ação *navigate*. Os efeitos dessa ação poderão levar o robô ao local y, caso não ocorra efeitos exógenos, ou fazer com que o robô retorne ao local de origem x, caso alguma ação do ambiente (por exemplo, terreno muito arenoso) ocorra.

Uma vez definido o domínio de planejamento não determinístico, precisamos formalizar *o problema de planejamento não determinístico*. Um problema de planejamento não determinístico $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ é definido da mesma forma da Definição 2.1.2, alterando-se apenas o domínio $\mathcal{D}$ para ser um domínio com ações não determinísticas.

Figura 9 – Um trecho do domínio não determinístico Rovers em PDDL.

```
(define(domain Rover)
 (:predicates
     (at ?r - rover ?y - waypoint)
     (at_lander ?x - lander ?y - waypoint)
     (can_traverse ?r - rover ?x - waypoint ?y - waypoint)
     (equipped_for_soil_analysis ?r - rover)
      (equipped_for_rock_analysis ?r - rover)
           ⋮
     (equipped_for_imaging ?r - rover)
 )
 (:action navigate
     :parameters (?r - rover ?x - waypoint ?y - waypoint)
     :precondition (and (at ?r ?x)
                        (visible ?x ?y)
                        (can_traverse ?r ?x ?y)
                        (available ?r)
                    )
      :effect (oneof (and ((at ?r ?y) (not (at ?r ?x)) ))
                     (and ((at ?r ?x) (not (at ?r ?y)) ))
              )
 )
                              ⋮
```

Fonte: Elaborado pelo autor.

Definição 2.2.2 (Problema de planejamento não determinístico) *Dado um domínio de planejamento não determinístico D, um problema de planejamento é definido por uma tupla* $P = (\mathcal{D}, s_0, \varphi)$ *onde:*

- $\mathcal{D}$ *é um domínio de planejamento não determinístico (com assinatura* $\mathbb{P}$ *e* $\mathbb{A}$*);*
- s_0 *é o estado inicial do problema, e;*
- φ *é uma fórmula da lógica proposicional que representa a meta de planejamento. O conjunto de estados que satisfaz a meta* φ *é denominado conjunto meta G.*

Representamos um problema de planejamento não determinístico em PDDL de forma similar ao que foi feito em planejamento determinístico. A Figura 10 representa o código PDDL para um problema de planejamento não determinístico.

Devido às incertezas dos efeitos das ações, uma solução para um problema de planejamento com ações não determinísticas não pode ser uma sequência de ações, uma vez que devemos explorar todos os caminhos possíveis que uma ação pode gerar. Assim, uma solução para uma problema de planejamento em ambiente não determinístico corresponde a um mapeamento entre estados e ações, o qual denominamos *política* (GHALLAB *et al.*, 2004).

Definição 2.2.3 (Política) *Seja* $P = \langle \mathcal{D}, s_0, \varphi \rangle$ *um problema planejamento em um domínio* $\mathcal{D} = \langle S, L, T \rangle$ *com ações não determinísticas. Uma política* π *é uma função parcial* $\pi : S \to A$ *que*

Figura 10 – Código PDDL para um problema de planejamento não determinístico do robô de Marte.

```
(define (problem rover12345)
 (:domain Rover)
 (:objects
     rover0 rover1 - rover
     waypoint0 waypoint1 waypoint2 waypoint3 waypoint4 waypoint5 - waypoint
     camera0 - camera
     objective0 objective1 - objective
     calibrated camera0 rover0
     colour high_res low_res - Mode
     rover0store - Store
 )
 (:init
    at-rover0-waypoint3,at_soil_sample-waypoint2,at_soil_sample-waypoint0,channel_fr-
    ee-general,empty-rover0store,at_rock_sample-waypoint1,at_rock_sample-waypoint3,
    ,at_rock_sample-waypoint2,at_rock_sample-waypoint1,at_soil_sample-waypoint3
 )
 (:goal
    communicated_soil_data waypoint3,communicated_rock_data waypoint2,communicated_
    image_data-objective1-high_res
 )
)
```

Fonte: Elaborado pelo autor

mapeia estados em ações; tal que, para todo estado $s \in S$*, se* π *está definida para s, então* $\pi(s) \in \{a \in \mathbb{A} : T(s,a) \neq \varnothing\}$ *(PEREIRA, 2007).*

De acordo com (CIMATTI *et al.*, 2003), a qualidade de uma política pode ser:

- *fraca*, são políticas que podem atingir a meta, mas que não há garantias disso, devido ao não determinismo das ações;
- *forte*, são políticas que têm a garantia de atingir a meta, a despeito do não determinismo das ações e;
- *forte-cíclica*, são políticas que têm a garantia de atingir a meta, sob a suposição de que o agente consiga sair dos ciclos decorrentes do não determinismo das ações.

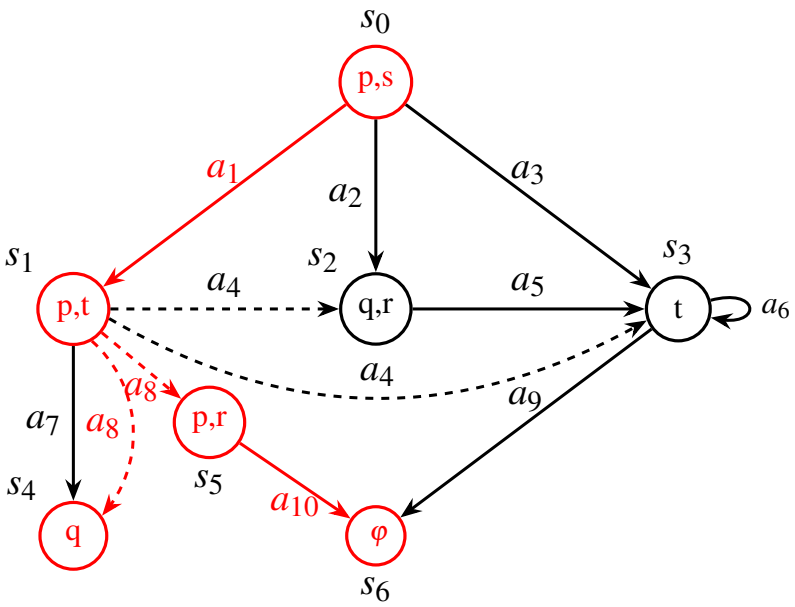
Exemplo 6 (Qualidade da solução) *Considere o problema de planejamento em que o domínio é o sistema de transição de estados da Figura 8, o estado inicial é* s_0 *e a meta é* $p \wedge \varphi \wedge \neg q \neg r \wedge \neg s \wedge \neg t$ *(satisfeita no estado* s_6*), temos que:*

- *A solução* $\pi_1 = \{(s_0,a_1),(s_1,a_8),(s_5,a_{10})\}$ *(Figura 11 é uma política fraca, uma vez que não há garantias de que a execução da ação* a_7 *no estado* s_1 *leve o agente ao estado* s_6*;*
- *A solução* $\pi_2 = \{(s_0,a_1),(s_1,a_4),(s_2,a_5),(s_3,a_9)\}$ *(Figura 12 é uma política forte, uma vez que, independentemente do não determinismo da ação* a_5*, sempre é possível alcançar*

o estado s_6;

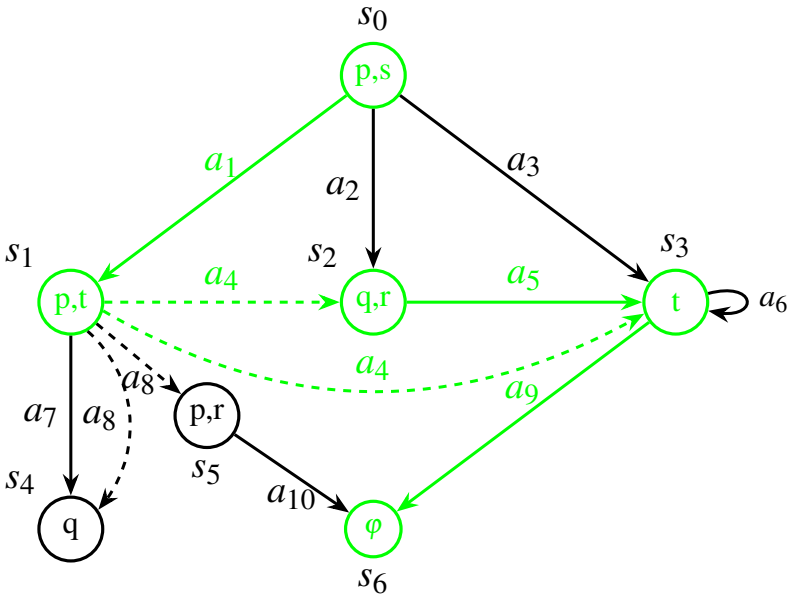
- *A solução $\pi_3 = \{(s_0, a_3), (s_3, a_6), (s_3, a_9)\}$ (Figura 13 é uma política forte cíclica, uma vez que o agente para alcançar o estado s_6 pode ficar preso em um um laço entre os estados s_0 e s_3.*

Figura 11 – política fraca



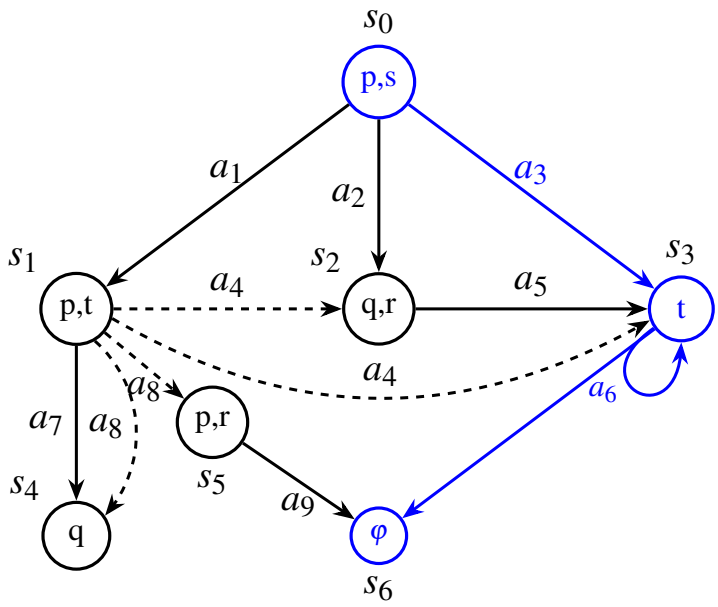
Fonte: Elaborado pelo autor.

Figura 12 – política forte



Fonte: Elaborado pelo autor.

Figura 13 – política forte cíclica



Fonte: Elaborado pelo autor.

Definição 2.2.4 (Conjunto de estados alcançáveis por uma política) *Seja* $P = \langle \mathcal{D}, s_0, \varphi \rangle$ *um problema de planejamento não determinístico,* π *uma política para um domínio de planejamento* $\mathcal{D}$*. O conjunto de estados alcançáveis por* π*, denotado por* $S_{alc[\pi]}$ *é definido como (PEREIRA, 2007).*

$$\{s : (s,a) \in \pi\} \cup \{s' : (s,a) \in \pi \, e \, s' \in \mathrm{T}(s,a)\}$$

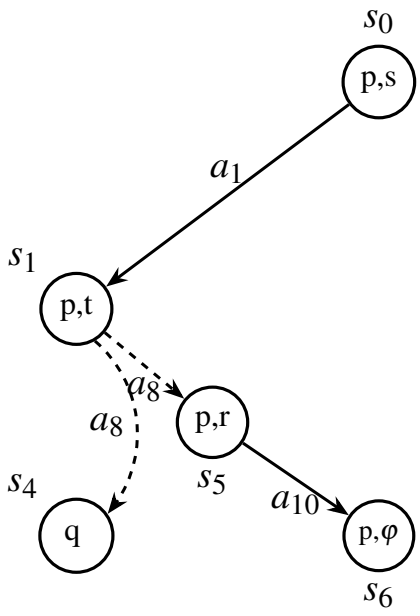
Definição 2.2.5 (Estrutura de execução de uma política) *Seja* $P = \langle \mathcal{D}, s_0, \varphi \rangle$ *um problema de planejamento não determinístico e* π *uma política para um domínio de planejamento* $\mathcal{D}$*. A estrutura de execução induzida pela política* π *corresponde ao grafo de transição de estados, indicado por* $\langle S_{alc[\pi]}, T \rangle$*, onde* $T \subseteq S \times S$ *corresponde a transição induzida por uma ação em* π*,* $S_{alc[\pi]}$ *corresponde aos estados alcançados pela política* π *e* $\langle S_{alc[\pi]}, T \rangle \subseteq \mathcal{D}$ *é uma tupla que contém todos os estados e transições que podem ser alcançados quando executamos uma política* π*.*

Exemplo 7 (Conjunto de estados alcançáveis) *Considere o Exemplo 6, temos que:*

- $S_{alc[\pi_1]} = \{s_0, s_1, s_4, s_5, s_6\}$;
- $S_{alc[\pi_2]} = \{s_0, s_1, s_2, s_3, s_6\}$;
- $S_{alc[\pi_3]} = \{s_0, s_3, s_6\}$.

A Figura 14 apresenta a estrutura de execução para a política fraca do Exemplo 6.

Figura 14 – Estrutura de execução para política fraca Figura 11



Fonte: Elaborado pelo autor.

Intuitivamente, a estrutura de execução de uma política a partir de um estado inicial determina uma sequência de estados que levam a meta. Formalmente:

Definição 2.2.6 (Caminho de execução de uma política) *Seja π uma política para um problema de planejamento não determinístico $P = \langle \mathcal{D}, s_0, \varphi \rangle$ e $S_{alc[\pi]}$ o conjunto de estados alcançados para a política π. O caminho de execução de uma política π, denotado por $\mathcal{P}_{(\pi)}$, a partir s_0 para o estado s_i em que φ é verdade, corresponde a sequência de estados $s_0, s_1, s_2, \ldots s_i$ em $S_{alc[\pi]}$ tal que, $\forall k = 0, 1, 2, \ldots i-1$ há uma relação binária $T(s_k, s_{k+1}) \subseteq S \times S$.*

O caminho de execução para a política fraca do Exemplo 6 corresponde a sequência de estados s_0, s_1, s_5, s_6 na Figura 14.

2.3 Planejamento com Preferências

Agentes devem ser capazes de raciocinar sobre as diferentes maneiras para alcançar estados metas, uma vez que, nas situações reais, um problema de planejamento pode apresentar um conjunto de soluções possíveis. Assim, tais agentes devem refletir a noção de *plano de alta qualidade* ou *mais preferível*. Nesse sentido, a abordagem de *planejamento baseado em preferências* desempenha um papel importante, pois flexibiliza a visão de metas em planejamento, permitindo ao agente decidir como atuar para alcançar a meta (JORGE *et al.*, 2008). Por exemplo, considere o robô de exploração planetária Figura 3. Poderíamos preferir que o agente, em algum

momento futuro, navegue por uma determinada região (e.g, uma rota através de uma região que permita a recarga de bateria) ou que obtenha a análise de rocha de uma região A antes de navegar para região B. Ademais, há um projeto futuro de enviar materiais coletados na superfície do planeta para Terra, assim, é fundamental que o agente, após obter a análise de uma rocha, *sempre* guarde uma amostra dela, durante todo o percurso que leve ao alcance da meta. Há duas abordagens principais em planejamento com preferências: *planejamento com preferências quantitativas* e *planejamento com preferências qualitativas* (SANTHANAM *et al.*, 2011).

Na abordagem de *preferências quantitativas*, as preferências representam subconjunto de metas desejáveis, caso não seja possível alcançar todas as metas. Por fim, apresentam uma função de utilidade usada para mapear um conjunto de alternativas em uma escala numérica o que permite a definição de uma ordem entre as soluções. (SANTHANAM *et al.*, 2016; MOHAMMED *et al.*, 2018).

Na abordagem de *preferências qualitativas*, metas podem ser vistas como condições que devem ser satisfeitas ao longo dos estados alcançados, estabelecendo restrições ou imposições sobre as diferentes trajetórias para o alcance da meta (*preferências qualitativas sobre planos*) (AKINTUNDE, ; KIM *et al.*, 2017; SANTOS, 2019). Nesse sentido, podemos usá-las para expressar as noções de *safety property* e *liveness property* (LAMPORT, 1977). Informalmente *safety property* afirma que "algo (ruim) não deverá acontecer" durante a execução, enquanto que *liveness property* expressa que eventualmente "algo (bom) deverá acontecer " durante a execução (KINDLER, 1994). *Safety properties* e *liveness properties* permitem que o sistema persiga a meta enquanto mantém-se livre de falhas durante a trajetória de alcance da meta (HSU *et al.*, 2021).

Neste trabalho, abordamos o problema de preferências sobre planos. O domínio de planejamento neste tipo de problema é o domínio clássico, com ações determinísticas, conforme mostrado na Seção 2.1.

Considere o domínio da Figura 4 e o problema de sair do estado s_0 e alcançar o estado s_6. Suponha ainda que o usuário deseja que a meta seja alcançada *evitando* estados em que q é verdade *(safety property)*. Desta forma, o algoritmo de planejamento deve devolver a solução $< a_3, a_9 >$, $< a_1, a_8, a_{10} >$, $< a_3, a_6, a_9 >$. Todos os outros caminhos violam a preferência estabelecida pelo usuário. O usuário pode também preferir que, na trajetória para alcance da meta, o agente *passe apenas por estados* em que p é verdade *(liveness property)*. Neste caso, a solução possível é: $< a_1, a_8, a_{10} >$. Outros caminhos que levam a s_6 passam pelos estados s_2 ou

pelo estado s_3 nos quais p não é verdade, violando a preferência estabelecida pelo usuário.

Preferências qualitativas podem ser expressas por operadores na linguagem PDDL (BAIER; MCILRAITH, 2009; GEREVINI *et al.*, 2009). A seguir abordamos alguns tipos deles.

- `sometime (p)` significa encontrar um plano em que a propriedade p seja verdadeira em algum estado no caminho para se alcançar a meta.
- `always (p)` significa encontrar um plano em que a propriedade p seja verdadeira em todos os estados no caminho para se alcançar a meta.
- `sometime-before ((p), (q))` significa encontrar um plano em que a propriedade q seja verdadeira antes que a propriedade p seja verdadeira e esta, por sua vez, deve ocorrer antes do alcance da meta.
- `at-most-once (p)` significa encontrar um plano em que a propriedade p seja verdadeira no máximo uma vez no caminho para se alcançar a meta.

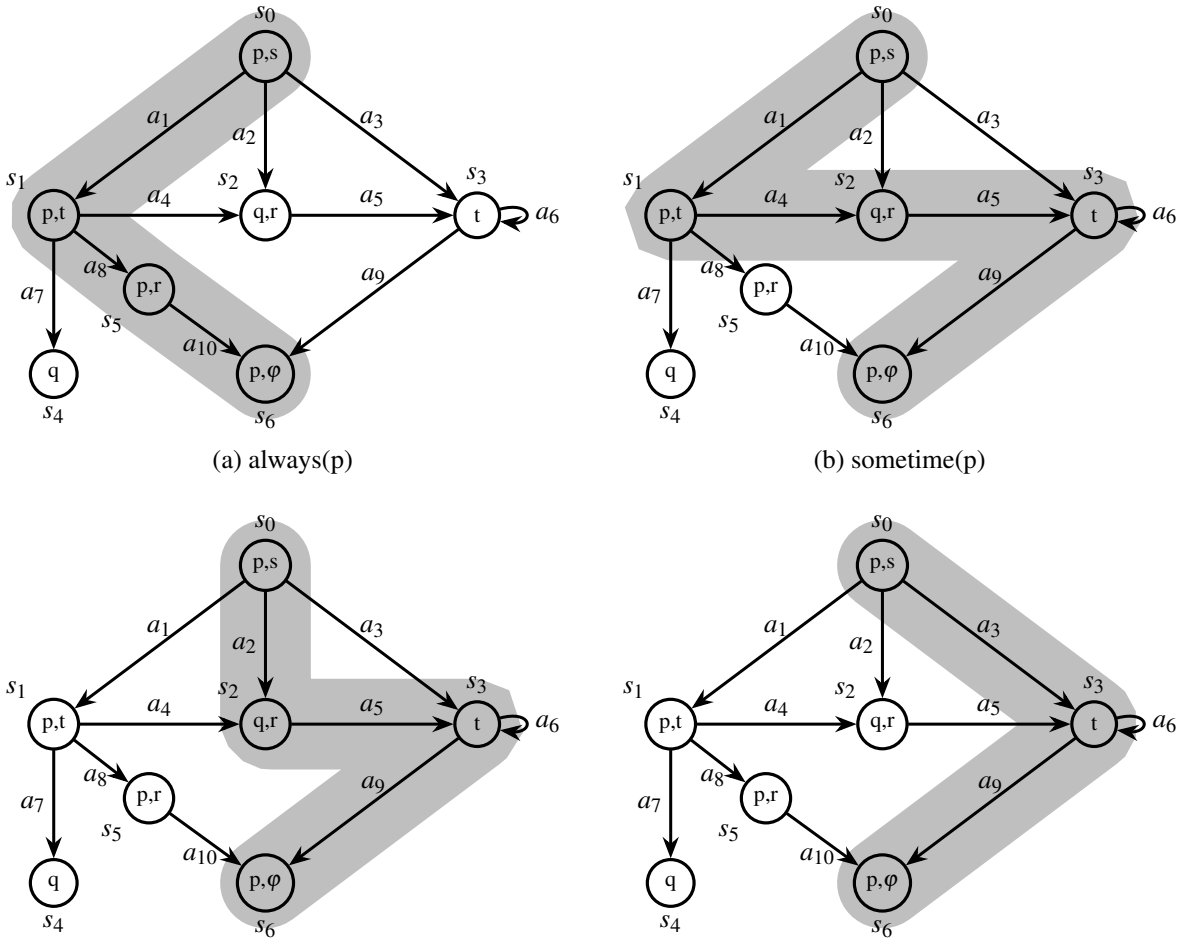
Considere novamente o domínio da Figura 4 e o problema de sair do estado s_0 e alcançar o estado s_6:

- a Figura 15a destaca um plano ($< a_1, a_8, a_{10} >$) que atende a preferência *always(p)*;
- a Figura 15b destaca um plano ($< a_1, a_4, a_5, a_9 >$) que atende a preferência *sometimes(t)*;
- a Figura 15c destaca um plano ($< a_2, a_5, a_9 >$) que atende a preferência *sometime-before(t)(q)*;
- a Figura 15d destaca um plano ($< a_3, a_9 >$) que atende a preferência *at-most-once (t)*.

Exemplos de propriedades desta natureza no domínio do robô de marte são: *sempre passar por regiões consideradas seguras (safety property)*; *em algum momento realizar recarga de bateria ou limpeza de algum sensor*; *realizar todas as análises antes de enviar algum dado para a estação espacial* e; *calibrar a câmera no máximo uma vez* com o intuito de economizar bateria.

A maior parte dos algoritmos de planejamento (BAIER *et al.*, 2009), (COLES; COLES, 2011), (CAMACHO *et al.*, 2017), abordam a classe de problemas de planejamento com preferências representando as preferências qualitativas como autômatos que são traduzidos nos estados e operadores gerados por um plano (PERCASSI; GEREVINI, 2019). Assim um estado aceito no autômato corresponde a satisfação da preferência (PERCASSI; GEREVINI, 2019). (SANTOS, 2019), por sua vez, não utilizam a abordagem baseada em autômatos. Preferências qualitativas são representadas por operadores temporais (GEREVINI; LONG, 2006) que são especificados na lógica α-CTL.

Figura 15 – Tipos de preferências sobre planos



(a) always(p)

(b) sometime(p)

(c) sometime_before(t)(p)

(d) at_most(t)

Fonte: Elaborado pelo autor

2.4 Planejamento como Verificação de Modelos

Verificação de modelos (EDMUND *et al.*, 1999; GRUMBERG *et al.*, 1999) é uma técnica de verificação formal que explora todos os possíveis estados de um sistema de transição de estados para verificar se uma determinada propriedade ocorre. Aplicar a técnica de verificação de modelos é um processo de três passos: *modelagem* do sistema, normalmente por meio de uma estrutura de Kripke (KRIPKE, 1959), *especificação* da propriedade a ser verificada por meio de uma fórmula lógica; e, por fim, a *verificação* que deverá ser automática.

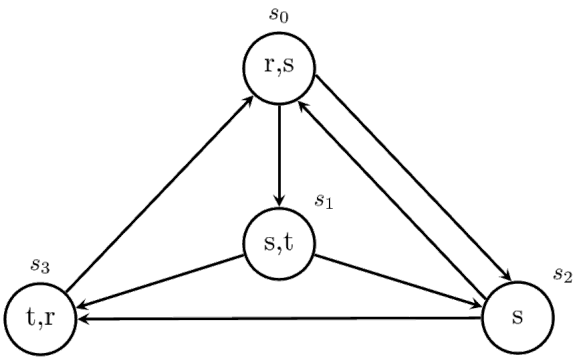
Definição 2.4.1 (Estrutura de Kripke) *Dado um conjunto $\mathbb{P}$ não vazio de proposições atômicas, uma estrutura de Kripke (KRIPKE, 1959) é uma tupla $\mathcal{M} = \langle S, L, T \rangle$ onde :*

- *S é o conjunto finito não vazio de estados.*
- *$L : S \to 2^{\mathbb{P}}$ e uma função de interpretação de estados.*

– $T : 2^S \to 2^{\mathbb{P}}$ *é uma função de transição de estados.*

Podemos visualizar uma estruturas de Kripke (KRIPKE, 1959) como um *sistemas de transição*, com rótulos de estados. Assim, podemos representá-lo por grafo direcionado em que os vértices representam os estados e as arestas as transições.

Figura 16 – Estrutura de Kripke



Fonte: Elaborado pelo autor

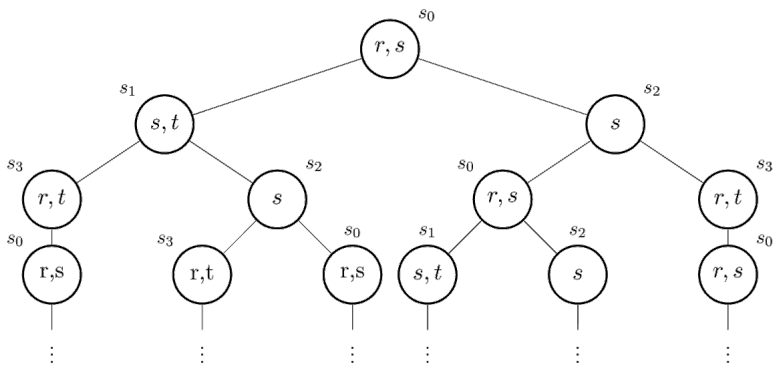
Definição 2.4.2 (Caminho em um estrutura de Kripke) *Um caminho* ρ*, em uma estrutura de Kripke* $\mathcal{M}$*, é uma sequência máxima de estados* $s_0, s_1, s_2, s_3, \ldots$*, tal que, para todo* $i \geq 0$ *temos,* (s_i, s_{i+1}).

Caminho em uma estrutura de Kripke corresponde a noção de futuro. Podemos representar todos os caminhos computacionais possíveis de uma estrutura de Kripke, a partir de um estado inicial s_0, "desdobrando" a estrutura de Kripke para obter uma *árvore de computação* infinita. No contexto da *verificação de modelos*, podemos adotar o termo *modelo* para referir-se à estrutura de Kripke.

Um *verificador* de modelos é um algoritmo que recebe como entrada uma estrutura de Kripke $\mathcal{M}$ e uma propriedade φ. Se a propriedade é satisfeita em $\mathcal{M}$, o algoritmo devolve *sucesso*. Caso contrário um *contraexemplo* é devolvido.

Podemos combinar as abordagens de verificação de modelos e planejamento para resolver *problemas de planejamento*. Nesse contexto, dado um *problema de planejamento*, o modelo a ser verificado é um domínio $\mathcal{D}$ e a propriedade a ser satisfeita, expressa por meio de uma fórmula em uma lógica temporal, é a meta de planejamento. Especificamente, a fórmula deve ser satisfeita a partir do estado inicial s_0 do problema.

Figura 17 – Árvore de computação

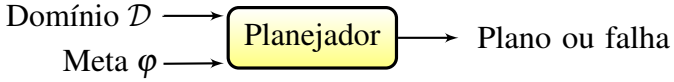


Fonte: Elaborado pelo autor

Figura 18 – Verificador de Modelos.

Fonte: (PEREIRA, 2007)

Figura 19 – Planejamento baseado na verificação de modelos



Fonte: (PEREIRA, 2007)

2.5 Lógicas Temporais

Lógicas Temporais são formalismos capazes de especificar comportamentos temporais de sistemas dinâmicos, modelados como *estruturas de Kripke* (KRIPKE, 1959). O valor verdade de uma fórmula em lógica temporal pode variar, conforme o momento em que a fórmula é avaliada. Nesta seção iremos apresentar as lógicas temporais CTL (EMERSON; CLARKE, 1982) e $\alpha - CTL$ (PEREIRA, 2007).

2.5.1 Lógica temporal CTL

CTL (Computation Tree Logic) (EMERSON; CLARKE, 1982) é uma lógica temporal de tempo ramificado. Tempo ramificado significa que, a cada momento no tempo, pode haver diferentes futuros possíveis. Em CTL, as fórmulas permitem especificar propriedades quantificadas sobre caminhos. Na lógica CTL os operadores temporais devem ser imediatamente precedidos por um quantificador de caminho ($\exists$, $\forall$).

Definição 2.5.1 (Sintaxe) *A sintaxe da lógica CTL é definida como a seguir (EMERSON; CLARKE, 1982)*

$$\varphi = p \mid \neg p \mid \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \forall \circ \varphi \mid \exists \circ \varphi \mid \forall \Box \varphi \mid \exists \Box \varphi \mid \forall \diamond \varphi \mid \exists \diamond \varphi \mid \forall (\varphi_1 \sqcup \varphi_2) \mid \exists (\varphi_1 \sqcup \varphi_2)$$

Intuitivamente:

* $\forall \circ \varphi$ significa que a fórmula φ ocorre em todos os estados e todos os caminhos da árvore de computação.
* $\exists \circ \varphi$ significa que existe um caminho na árvore de computação em que fórmula φ ocorre no próximo estado do caminho.
* $\forall \Box \varphi$ significa que a fórmula φ ocorre em todos os estados e todos os caminhos da árvore de computação.
* $\exists \Box \varphi$ significa que existe um caminho na árvore de computação em que fórmula φ ocorre em todos os estados e todos os caminhos da árvore de computação.
* $\forall \varphi_1 \sqcup \varphi_2$ significa que a fórmula φ_1 deve ser obrigatoriamente verdade até que φ_2 seja verdade em todos os estados e caminhos da árvore de computação.
* $\exists \varphi_1 \sqcup \varphi_2$ significa que existe um caminho na árvore de computação em que fórmula φ deve ser obrigatoriamente verdade até φ_2 seja verdade em todos os estados deste caminho.

Definição 2.5.2 *(Semântica da Lógica CTL) Sejam $\mathcal{M} =< S, L, T >$ uma estrutura de Kripke, $s \in S$ um estado e φ uma fórmula CTL. A relação $(\mathcal{M}, s) \vDash \varphi$ é definida por (HUTH; RYAN, 2008):*

- $(\mathcal{M}, s) \vDash p$ *sss* $p \in L(s)$.
- $(\mathcal{M}, s) \vDash \neg\varphi$ *sss* $(\mathcal{M}, s) \nvDash \varphi$.
- $(\mathcal{M}, s) \vDash \varphi_1 \wedge \varphi_2$ *sss* $(\mathcal{M}, s) \vDash \varphi_1$ *e* $(\mathcal{M}, s) \vDash \varphi_2$.
- $(\mathcal{M}, s) \vDash \varphi_1 \vee \varphi_2$ *sss* $(\mathcal{M}, s) \vDash \varphi_1$ *ou* $(\mathcal{M}, s) \vDash \varphi_2$.
- $(\mathcal{M}, s) \vDash \varphi_1 \rightarrow \varphi_2$ *sss* $(\mathcal{M}, s) \nvDash \varphi_1$ *ou* $(\mathcal{M}, s) \vDash \varphi_2$.
- $(\mathcal{M}, s) \vDash \forall \circ \varphi_1$ *sss para todo* s_1 *tal que* $(s, s_1) \in T$, *temos* $(\mathcal{M}, s_1) \vDash \varphi_1$.
- $(\mathcal{M}, s) \vDash \exists \circ \varphi_1$ *sss para algum* s_1 *tal que* $(s, s_1) \in T$, *temos* $(\mathcal{M}, s_1) \vDash \varphi_1$.
- $(\mathcal{M}, s) \vDash \forall \Box \varphi_1$ *é verdade sss para todos os caminhos* $s_1, s_2, s_3, \ldots$, *onde* $s_1 = s$, *e para todos os* s_i *ao longo deste caminho, temos* $(\mathcal{K}, s_i) \vDash \varphi$.
- $(\mathcal{M}, s) \vDash \exists \Box \varphi_1$ *é verdade sss existe um caminho* $s_1, s_2, s_3, \ldots$, *onde* $s_1 = s$, *e para todos os* s_i *ao longo deste caminho, temos* $(\mathcal{M}, s_i) \vDash \varphi$

- $(\mathcal{M},s) \vDash \forall \Diamond \varphi_1$ *é verdade sss para todos os caminhos* $s_1,s_2,s_3,\ldots$, *existe algum* s_i *ao longo deste caminho, tal que* $(\mathcal{M},s_i) \vDash \varphi$.
- $(\mathcal{M},s) \vDash \exists \Diamond \varphi_1$ *é verdade sss existe algum caminho* $s_1,s_2,s_3,\ldots$, *onde* $s_1 = s$, *e para algum* s_i *ao longo deste caminho, temos* $(\mathcal{M},s_i) \vDash \varphi$
- $(\mathcal{M},s) \vDash \forall(\varphi_1 \sqcup \varphi_2)$ *é verdade sss para todos os caminhos* $s_1,s_2,s_3,\ldots$, *este caminho satisfaz* $\varphi_1 \sqcup \varphi_2$, *isto é, existe algum* s_i *ao longo deste caminho tal que* $(\mathcal{K},s_i) \vDash \varphi_2$ *e, para cada* $j < i$, *temos* $(\mathcal{K},s_j) \vDash \varphi_1$
- $(\mathcal{M},s) \vDash \exists(\varphi_1 \sqcup \varphi_2)$ *é verdade sss existe algum caminho* $s_1,s_2,s_3,\ldots$, *onde* $s_1 = s$, *e este caminho satisfaz* $\varphi_1 \sqcup \varphi_2$ *como especificado no item anterior*

2.5.2 Lógica temporal α-CTL

A lógica temporal α-CTL (PEREIRA, 2007) é uma extensão da lógica CTL capaz de representar ações em sua semântica. Assim, o modelo $\mathcal{M}$ não é uma estrutura de Kripke, como na lógica *CTL*, mas, um sistema de transição de estados cujas transições são rotuladas por ações. Nesta lógica, operadores temporais são representados por símbolos pontuados.

Definição 2.5.3 (Sintaxe) *A sintaxe da lógica* α*-CTL (PEREIRA, 2007) é definida como a seguir:*

$$\varphi = p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \forall \odot \varphi \mid \exists \odot \varphi \mid \forall \boxdot \varphi \mid \exists \boxdot \varphi \mid \forall \diamondsuit \varphi \mid \exists \diamondsuit \varphi \mid \forall(\varphi_1 \sqcup \varphi_2) \mid \exists(\varphi_1 \sqcup \varphi_2)$$

Definição 2.5.4 (Modelo temporal na lógica α-CTL) *Dados um conjunto de átomos proposicionais não vazio* $\mathbb{P}$ *e um conjunto de ações não vazio* $\mathbb{A}$, *um modelo temporal* $\mathcal{M}$ *(PEREIRA, 2007) na lógica* α*-CTL é um sistema de transição de estados* $\mathcal{D} = \langle S,L,T \rangle$ *cujos estados são rotulados por subconjuntos de* $\mathbb{P}$ *e cujas transições são rotuladas por elementos de* $\mathbb{A}$*:*

- S*: é um conjunto finito não vazio de estados;*
- $L : S \rightarrow 2^{\mathbb{P}}$ *é uma função de rotulação de estados;*
- $T : S \times A \rightarrow 2^S$ *é uma função de transição de estados rotulada por ações.*

Definição 2.5.5 *(Semântica* α*-CTL) Seja* $\mathcal{M} =< S,L,T >$ *um sistema de transição em que os estados são rotulados com elementos de* $\mathbb{P}$ *e as transições são rotuladas com elementos de* $\mathbb{A}$. *A semântica da lógica* α*-CTL é definida como a seguir:*

- $(\mathcal{M},s) \vDash p$ (2.1)

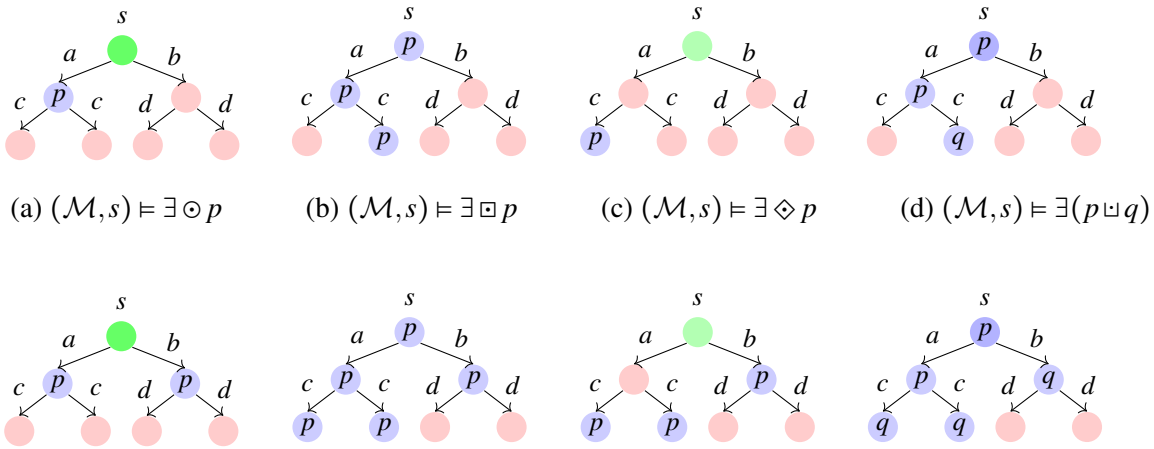
- $(\mathcal{M},s) \vDash \neg p \; sss \; (\mathcal{M},s) \nvDash p$ (2.2)
- $(\mathcal{M},s) \vDash \varphi_1 \wedge \varphi_2 \; sss \; (\mathcal{M},s) \vDash \varphi_1 \; e \; (\mathcal{M},s) \vDash \varphi_2$ (2.3)
- $(\mathcal{M},s) \vDash \varphi_1 \vee \varphi_2 \; sss \; (\mathcal{M},s) \vDash \varphi_1 \; ou \; (\mathcal{M},s) \vDash \varphi_2$ (2.4)
- $(\mathcal{M},s) \vDash \varphi_1 \rightarrow \varphi_2 \; sss \; (\mathcal{M},s) \nvDash \varphi_1 \; ou \; (\mathcal{M},s) \vDash \varphi_2$ (2.5)
- $(\mathcal{M},s) \vDash \forall \odot \varphi_1 sss \; existe \; a \in \mathbb{A} \; tal \; que \; para \; todos \; s_1 = T(a,s), temos (\mathcal{M},s_1) \vDash \varphi_1$ (2.6)
- $(\mathcal{M},s) \vDash \exists \odot \varphi_1 sss \; existe \; a \in \mathbb{A} \; tal \; que \; para \; algum \; s_1 = T(a,s), \; temos \; (\mathcal{M},s_1) \vDash \varphi_1$ (2.7)
- $(\mathcal{M},s) \vDash \forall \boxdot \varphi_1 é \; verdade \; sss \; existe \; ação \; a \; \in \mathbb{A} \; em \; que \; todos \; os \; estados \; s_i, (para \; i \geq 0), ao \; longo \; de \; todos \; caminhos \; iniciando \; em \; s, \; temos (\mathcal{M},s_i) \vDash \varphi$ (2.8)
- $(\mathcal{M},s) \vDash \exists \boxdot \varphi_1 é \; verdade \; sss \; existe \; ação \; a \; \in \mathbb{A} \; em \; que \; todos \; os \; estados \; s_i, (para \; i \geq 0), ao \; longo \; de \; algum \; caminho \; iniciando \; em \; s, \; temos (\mathcal{M},s_i) \vDash \varphi$ (2.9)
- $(\mathcal{M},s) \vDash \exists \diamondsuit \varphi_1$ *é verdade sss existe ação* $a \; \in \mathbb{A}$ *em que todos os estados* $s_i, (para \; i \geq 0), ao \; longo \; de \; algum \; caminho \; iniciando \; em \; s, \; temos (\mathcal{M},s_i) \vDash \varphi$ (2.10)
- $(\mathcal{M},s) \vDash \forall \diamondsuit \varphi_1$ *é verdade sss existe ação* $a \; \in \mathbb{A}$ *em que todos os estados* $s_i, (para \; i \geq 0), ao \; longo \; de \; algum \; caminho \; iniciando \; em \; s, \; temos (\mathcal{M},s_i) \vDash \varphi$ (2.11)
- $(\mathcal{M},s) \vDash \forall (\varphi_1 \sqcup \varphi_2)$ *sss existe ação* $a \; \in \mathbb{A}$ *tal que para algum caminho iniciando em s existe algum estado* s_i *(para* $i \geq 0$*) ao longo deste caminho tal que* $(\mathcal{M},s_i) \vDash \varphi_2$ *e, para cada* $0 \leq k < i$*,* $temos \; (\mathcal{M},s_j) \vDash \varphi_1$ (2.12)
- $(\mathcal{M},s) \vDash \exists (\varphi_1 \sqcup \varphi_2)$ *sss existe ação* $a \; \in \mathbb{A}$ *tal que para algum caminho iniciando em s existe algum estado* s_i *(para* $i \geq 0$*) ao longo deste caminho tal que* $(\mathcal{M},s_i) \vDash \varphi_2$ *e, para cada* $0 \leq k < i$*,* $temos \; (\mathcal{M},s_j) \vDash \varphi_1$ (2.13)

A Figura 20 ilustra a semântica dos operadores temporais da lógica α-CTL. As transições de estados são rotulados por elementos pertencentes a $\mathbb{A} = \{a,b,c,d\}$. Assim, sendo $\mathcal{M} = \langle S,L,T \rangle$ um sistema de transição rotulado por ações $\mathbb{A}$, $\mathbb{P}$ um conjunto não vazio de proposições e s um estado inicial($s \in S$), temos que em (MENEZES, Maria Viviane de, 2014):

1. (a) executando a ação a a partir de s existe um caminho em que no próximo estado p é verdade;
2. (b) existe um caminho em que todos os estados são p executando a ação a a partir de s.
3. (c) executando a ação a a partir de s, existe um caminho em que algum estado futuro p é verdade;

4. (d) executando-se uma ação a a partir de s, existe um caminho em que todos os estados futuros são p até encontrar um estado em que q seja verdade;
5. (e) para todos os caminhos a partir da execução de a em s no próximo estado p é verdade.
6. (f) em todos os caminhos futuros alcançados pela execução da ação a a partir de s, em cada estados p é verdade.
7. (g) para todos os caminhos futuros alcançados pela execução da ação a a partir de s, p é verdade em algum estado.
8. (h) para todos os caminhos futuros alcançados pela execução da ação a a partir de s, p é verdade até q ser verdade.

Figura 20 – Semântica dos operadores temporais da lógica α-CTL



Fonte: (MENEZES, Maria Viviane de, 2014)

2.6 Raciocínio sobre Ações não determinísticas

A representação de um domínio de planejamento por meio de uma *linguagem de ações*, e.g PDDL, permite uma representação compacta do espaço de estados. Assim, podemos partir de um estado inicial e avançar progressivamente no espaço de estados, determinando uma sequência de ações e estados que leve ao alcance do estado meta, *busca progressiva* (HOFFMANN, 2001) ou, partir do estado meta e regressivamente determinar a sequência de ações e estados que alcance o estado inicial (RINTANEN, 2008).

Na busca progressiva temos que o estado inicial é completamente definido enquanto na busca regressiva temos que o estado meta é parcialmente definido. Assim, em uma busca regressiva, lidamos com estados abstratos, isto é, estados em que as proposições não presentes

na descrição são consideradas com valoração desconhecida (podem ser verdadeiras ou falsas) (SANTOS; BARROS, 2017).

2.6.1 Representação de ações não determinísticas

Uma ação não determinística pode ser representada como na Definição 2.6.1 (MENEZES, Maria Viviane de, 2014).

Definição 2.6.1 *(Representação de Ação não determinística) Dado um domínio de planejamento não determinístico $\mathcal{D}$ com assinatura $(\mathbb{P}, \mathbb{A})$, uma ação $a_i \in \mathbb{A}$ é representada pelo par:*

$$\langle \boldsymbol{precond}(a_i), \boldsymbol{Efeitos}(a_i) \rangle,$$

onde, $\boldsymbol{precond}(a_i)$ corresponde ao conjunto de proposições $p_i \in \mathbb{P}$ que são verdadeiras no estado em que a ação a_i é executada e $\boldsymbol{Efeitos}(a_i) = \{e_1, e_2, \ldots, e_n\}$ corresponde ao conjunto de efeitos não determinísticos da ação a_i. Cada $e_i \in \boldsymbol{Efeitos}(a_i)$ corresponde a tupla $\langle \boldsymbol{ef^+}(e_i), \boldsymbol{ef^-}(e_i) \rangle$ onde:

- $\boldsymbol{ef^+}(e_i)$ *corresponde às proposições que se tornam verdadeiras após e_i ocorrer e*
- $\boldsymbol{ef^-}(e_i)$ *corresponde às proposições que se tornam falsas após e_i ocorrer.*

A Figura 21 apresenta uma representação da ação não-determinística *navigate* do domínio *Rover*. A ação *navigate(waypoint0,waypoint2,waypoint1)*, quando executada em um estado possui dois efeitos possíveis: levar o agente para a localização *waypoint1* ou para a localização *waypoint2*. A Figura 22 mostra o sistema de transição de estados em que a ação não-determinística `navigate` é utilizada.

Figura 21 – Descrição da ação não determinística *navigate*.

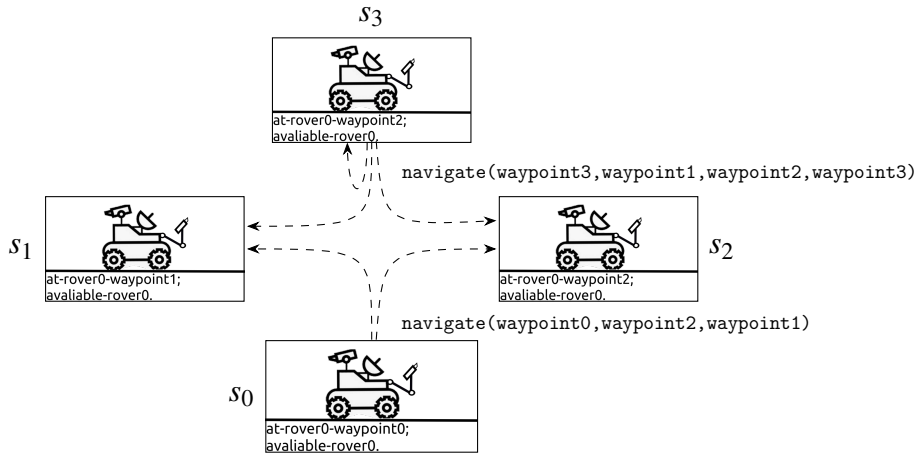
`navigate(waypoint0,waypoint2,waypoint1):`
$\langle precond = \{at-rover0-waypoint0\}$
$\langle ef^+(e_1) = \{at-rover0-waypoint2\}; ef^-(e_1) = \{at-rover0-waypoint0\}\rangle$
$\langle ef^+(e_2) = \{at-rover0-waypoint1\}; ef^-(e_2) = \{at-rover0-waypoint0\}\rangle$

Fonte: Elaborado pelo autor.

2.6.2 Regressão de estados por meio de ações não determinísticas

A *regressão* de um estado s_t por uma ação não determinística a_t consiste em determinar um *estado abstrato* s_x, representando os estados s_{i-1}, denominado *estados predecessores* de

Figura 22 – Descrição de um cenário não determinístico para domínio Rover.



Fonte: Elaborado pelo autor

s_i, de forma que, a execução da ação a_i a partir de qualquer s_{i-1} faz com que o ambiente evolua para o estado s_i.

Definição 2.6.2 *(Estado predecessor) Dado um problema de planejamento não determinístico $P = (\mathcal{D}, s_0, \varphi)$, denominamos de estado predecessor de $s_i \in S$ pela ação a_t, denotado por $pred(s_i, a_t)$ o estado $s_{i-1} \in S$ tal que a execução da ação a_t em s_{i-1} faz o ambiente evoluir de s_{i-1} para s_i.*

A computação do estado predecessor constitui um passo fundamental na operação de regressão. Entretanto, para computar o estado predecessor, é importante identificar as ações que são **relevantes** a um estado s_i, isto é, ações do domínio que podem contribuir com as propriedades de s_i.

Definição 2.6.3 *(Ação relevante) Dado um problema de planejamento não determinístico $P = (\mathcal{D}, s_0, \varphi)$ com assinatura $(\mathbb{P}, \mathbb{A})$, uma ação não determinística $a = \langle precond(a), Efeitos(a) \rangle$ é relevante para um estado $s \in S$, se existir algum $p \in \mathbb{P}$ tal que, se $p \in ef^+(e_i)$ então $p \in s$ e se não existir um $p_k \in \mathbb{P}$ tal que, se $p_k \in ef^-(e_j)$ então $p_k \in s$*

Uma ação a é relevante a um estado s se existir algum $ef^+(e_i)$ de a que faça parte do estado s, isto é, $(ef^+(e_i) \cap s \neq \varnothing \wedge ef^-(e_i) \cap s = \varnothing)$. No cenário não determinístico da Figura 22 a ação *navigate(waypoint0, waypoint2, waypoint1)* é relevante ao estado s_2, pois existe um $p = at-rover0-waypoint2$ tal que $p \in ef^+(e_1) = \{at-rover0-waypoint2\}$ e $p \in s_2$. Além disso, $ef^-(e_1) \cap s_2 = \varnothing$.

A regressão de um estado s_i abstrato (isto é, estado em que as proposições não declaradas podem ser consideradas tanto verdadeiras quanto falsas) por uma ação não determinística

relevante a_i, indicamos por $Regr^{ND(a_i)}(s_i)$, é determinada eliminando os efeitos positivos de a_i para s_i ($x \setminus ef^+(e_i)$, Equação 2.14) e, a seguir, unindo o resultado com as precondições de a_i. Formalmente, sendo a_i relevante(Definição 2.6.3), $Regr^{ND(a_i)}(s_i)$ é definida como (MENEZES, Maria Viviane de, 2014):

$$Regr^{ND(a_i)}(s_k) = \{(x \setminus ef^+(e_i)) \cup precond(a_i)\} \tag{2.14}$$

A regressão de um conjunto de estados X por uma ação a leva a um conjunto Y de estados antecessores. No entanto, devido ao não-determinismo das ações, ao aplicar uma ação a em Y, os estados em X podem ser **necessária** (regressão forte) ou **possivelmente** (regressão fraca) alcançados.

Definição 2.6.4 *(Regressão fraca de um conjunto de estados por uma ação não determinística) Sejam S o conjunto de estados do domínio, o conjunto de estados $X \subseteq S$ e a_i uma ação não determinística. A regressão fraca de X pela ação a_i ($RegFraca^{ND(a_i)}(X)$) corresponde ao conjunto de estados $pred(X, a_i)$, a partir dos quais os estados $x_i \in X$ são **possivelmente** alcançados como resultado da aplicação de a_i (MENEZES, Maria Viviane de, 2014).*

$$RegFraca^{ND(a_i)}(X) = \{y \in pred(X, a_i) : y^{a_i} \cap X \neq \varnothing\} \tag{2.15}$$

A expressão y^{a_i} indica os estados alcançados após a aplicação de a_i no estado y.

Exemplo 8 *(Regressão fraca) Considere o cenário não determinístico para o domínio Rover Figura 22. Seja $S = \{s_0, s_1, s_2, s_3\}$, o conjunto de estados do domínio, $A = \{navigate(waypoint3, waypoint1, waypoint2, waypoint3), navigate(waypoint0, waypoint2, waypoint1)\}$ o conjunto de ações, $a_i = navigate(waypoint3,\ waypoint1, waypoint2, waypoint3) \in A$ uma ação não determinística e que $X = \{s_1, s_2\} \in S$. A regressão fraca de X pela ação $a_i = navigate(waypoint3, waypoint1, waypoint2, waypoint3)$ corresponde ao estado s_3, $(RegFraca^{ND(a_i)}(\{s_1, s_2\}) = \{s_3\}$, pois ao aplicar a ação $a_i = navigate(waypoint3,\ waypoint1, waypoint2,\ waypoint3)$ ao estado s_3 ($s_3^{a_i}$), **possivelmente**, alcançamos $X = \{s_1, s_2\}$, uma vez que, também, podemos alcançar o estado $s_3 \notin X$.*

Definição 2.6.5 *(Regressão forte) Seja S o conjunto de estados do domínio, um conjunto $X \subseteq S$ de estados e a_i uma ação não determinística. A regressão forte de X pela ação a_i ($RegForte^{ND(a_i)}(X)$) corresponde ao conjunto de estados $pred(X, a_i)$, a partir dos quais os estados $x_i \in X$ são **necessariamente** alcançados como resultado da execução de a_i.*

$$RegForte^{ND(a_i)}(X) = \{y \in pred(X, a_i) : \varnothing \neq y^{a_i} \subseteq X\} \tag{2.16}$$

Exemplo 9 *(Regressão forte de um conjunto de estados por uma ação não determinística) Considere o cenário não determinístico para o domínio Rover Figura 22. Seja* $S = \{s_0, s_1, s_2, s_3\}$*, o conjunto de estados do domínio,* $A = \{navigate(waypoint3, waypoint1, waypoint2, waypoint3), navigate(waypoint0, waypoint2, waypoint1)\}$ *o conjunto de ações,* $a_i = navigate(waypoint0, waypoint2, waypoint1) \in A$ *uma ação não determinística e que* $X = \{s_1, s_2\} \in S$*. A regressão forte de X pela ação* $a_i = navigate(waypoint0, waypoint2, waypoint1)$ *corresponde ao estado* s_0 $(RegForte^{ND(a_i)}(\{x_1, x_2\}) = \{s_0\})$*, pois ao aplicar a ação* $a_i = navigate(waypoint0, waypoint2, waypoint1)$ *ao estado* $s_0(s_0^{a_i})$*,* ***necessariamente*** *alcançamos o conjunto de estados* $\{s_1, s_2\} \subseteq X$*.*

2.6.3 Regressão simbólica de estados através de ações não determinísticas

Nessa seção apresentamos um formalismo capaz de computar a regressão de estado por uma ação não determinística. Nessa abordagem, estados e ações são representados como fórmulas da lógica proposicional (MENEZES, Maria Viviane de, 2014).

Definição 2.6.6 *(Domínio de planejamento simbólico não determinístico) Dado um conjunto de átomos proposicionais* $\mathbb{P}$*, um domínio de planejamento simbólico não determinístico (*$\mathcal{D}_{simb}$*) é especificado por um conjunto de ações* $\langle \boldsymbol{precond(a_i), Efeitos(a_i)} \rangle$*. Cada ação é representada pelo par* $\langle \boldsymbol{\xi(precond(a_i)), \xi(Efeitos(a_i))} \rangle$*, onde:*

$$\xi(precond(a_i)) = \bigwedge_{p \in precond(a_i)} p;$$

$$\xi(Efeitos(a_i)) = \bigvee_{k=1}^{i} \left(\bigwedge_{p \in ef^+(e_k)} p \wedge \bigwedge_{q \in ef^-(e_k)} \neg q \right).$$

Cada estado $x_i \in S$ é representado pela fórmula da lógica proposicional como $\xi(x_i) = \bigwedge_{p \in \mathbb{P}}$. Um conjunto de estado $X = \{x_1, x_2, x_3, \ldots x_n\}$ é então representado como $\xi(X) = \bigvee_{i=1}^{n} (\xi(x_i))$.

A seguir, definimos como computar simbolicamente o conjunto de estados predecessores que levam ***possivelmente*** a um conjunto de estados X através de ações não determinísticas. A regressão simbólica fraca é definida como (MENEZES, Maria Viviane de, 2014):

Definição 2.6.7 *(Regressão simbólica fraca de um conjunto de estados por uma ação não determinística) Seja* a_i *uma ação não determinística representada pela tupla* $\langle \boldsymbol{precond(a_i)}, \langle \boldsymbol{ef^+(e_i), ef^-(e_i)} \rangle \rangle$*(Definição 2.6.1), a regressão simbólica fraca de um conjunto de estados X*

segundo a ação a_i é dada por:

$$Regr^{ND(a_i)}(X) = (\xi(precond(a_i)) \wedge \exists modifica(a_i) \cdot \overbrace{(\xi(ef^+(e_i)) \wedge \xi(X))}^{E_{rel}}) \tag{2.17}$$

A expressão $(\xi(ef^+(e_i)) \wedge \xi(X)$ computa o conjunto de estados relevantes(E_{rel}) para a_i. O conjunto $modifica(a_i)$ corresponde ao conjunto de proposições p_i que ocorrem nos efeitos de a_i. A quantificação existencial $\exists modifica(a_i)$ sobre E_{rel} elimina de E_{rel} as proposições que ocorrem nos efeitos de a_i. A seguir, uma conjunção da expressão resultante com a pré-condição de a_i é realizada para determinar o estado predecessor.

Definição 2.6.8 *(Regressão simbólica forte de um conjunto de estados por uma ação não determinística) Seja a_i uma ação não determinística STRIPS representada pela tupla $\langle \boldsymbol{precond(a_i)}, \langle \boldsymbol{ef^+(e_i)}, \boldsymbol{ef^-(e_i)} \rangle \rangle$(Definição 2.6.1) a regressão forte de um conjunto de estados X segundo a ação a_i é dada por (MENEZES, Maria Viviane de, 2014):*

$$RegrForte^{ND(a_i)}(X) = (\xi(precond(a_i)) \wedge \forall modifica(a_i) \cdot \overbrace{(\xi(ef^+(e_i)) \longrightarrow \xi(X))}^{E'_{rel}}) \tag{2.18}$$

A expressão $((\xi(ef^+(e_i)) \longrightarrow \xi(X)) \wedge \xi(X)$ computa o conjunto de estados relevantes que levam apenas a estados em X (E'_{rel}) para a_i. A quantificação universal $\forall modifica(a_i)$ sobre E'_{rel} elimina de E'_{rel} as proposições que ocorrem nos efeitos de . Por fim, uma conjunção da expressão resultante com as pré-condições de a_i é realizada para determinar a representação proposicional do conjunto de estados predecessor de X.

2.7 Algoritmos de planejamento para verificação de modelos usando a lógica α−CTL

A seguir, descrevemos os algoritmos de planejamento para verificação de modelos usando a lógica α−CTL que serão utilizados neste trabalho. Os algoritmos recebem como entrada o domínio de planejamento representado por meio de ações com pré-condições e efeitos e uma fórmula φ em α-CTL a ser verificada. Esta é uma diferença fundamental entre os algoritmos utilizados neste trabalho (SANTOS *et al.*,) e os algoritmos de planejamento como verificação de modelos da literatura (CIMATTI *et al.*, 2003), os quais racionam sobre transições do modelo formal e não sobre o esquema de ações do domínio de planejamento. Todos os algoritmos são implementados utilizando *diagramas de decisão binária* (BRYANT, 2018) para representar os estados e ações do domínio e para realizar o raciocínio sobre esta representação simbólica.

2.7.1 Algoritmo SAT-AG

O algoritmo SAT-AG computa o conjunto de estados satisfazendo a fórmula $\forall \boxdot p$. Recebe como entrada uma propriedade $p \in \mathcal{P}$ e realiza uma busca regressiva no espaço. Este algoritmo utiliza as funções auxiliares `SAT` (PEREIRA; BARROS, 2008) e `regressão` (MENEZES, Maria Viviane de, 2014) que pode ser fraca ou forte.

O algoritmo inicia na linha 2 atribuindo à variável X o conjunto de todos os estados S do domínio de planejamento. Na linha 3, a função $SAT(p)$ devolve o conjunto de estados do modelo que satisfazem a propriedade p. Em cada iteração, os estados em Y são armazenados no conjunto X (linha 5) e a variável Y acumulará os estados resultantes do passo da regressão forte de Y (linha 6). A regressão forte determina os estados que possuem ações cujos efeitos levam necessariamente a estados que estão em Y. Quando nenhum novo estado é acumulado em Y, um ponto-fixo foi alcançado e o laço é encerrado. Na linha 8, o conjunto de estados Y é retornado.

Algoritmo 1: SAT-AG(p)

Entrada: Propriedade $p \in \mathcal{P}$.
Saída: Conjunto de estados que satisfaz $\forall \boxdot p$.

```
1  begin
2      X := S;
3      Y := SAT(p);
4      while X != Y do
5          X := Y;
6          Y := Y ∩ regressao_forte(Y);
7          se s0 ∈ Y então
8              retorne Y;
9          fim
10     end
11     retorne Y;
12 end
```

Fonte: (SANTOS *et al.*,)

2.7.2 Algoritmo SAT-EU

O algoritmo $SATEU(\varphi_1, \varphi_2)$ devolve uma sequência de estados em que a propriedade φ_2 ocorre em um estado e, em todos os estados que precedem a φ_2 ocorre φ_1, ou seja, o conjunto de estados que satisfaz a fórmula $\exists(\varphi_1 \sqcup \varphi_2)$.

O algoritmo inicia recebendo as fórmulas φ_1 e φ_2. As instruções $SAT(\varphi_1)$ e $SAT(\varphi_2)$, linhas 2 e 4, devolvem um conjunto de estados em que as fórmulas φ_1 e φ_2 são

verdadeiras. Na linha 7, a intersecção de W com a regressão garante que φ_1 ocorra em todos os estados do caminho que levam a φ_2. A seguir uma operação de união é realizada para garantir que os estados que satisfazem φ_2, mas não φ_1 ainda possam ser considerados no conjunto final de estados devolvidos. Quando nenhum outro estado é obtido no passo regressivo o procedimento é finalizado e o conjunto de estados no passo regressivo é devolvido.

Algoritmo 2: SAT-EU(φ_1, φ_2)

Entrada: Fórmulas φ_1 e φ_2.
Saída: Conjunto de estados que satisfaz $\exists(\varphi_1 \sqcup \varphi_2)$

```
1  begin
2      W := SAT(φ1);
3      X:=S;
4      Y := SAT(φ2);
5      while X != Y do
6          X := Y;
7          Y := Y ∪ (W ∩ regressão_fraca(Y));
8          se s0 ∈ Y então
9              retorne Y;
10         fim
11     end
12     retorne falha;
13 end
```

Fonte: (SANTOS *et al.*,)

2.7.3 Algoritmo SAT-AU

O algoritmo $SATAU(\varphi_1, \varphi_2)$ é semlhante ao Algoritmo 2. Ele devolve uma sequência de estados em que a propriedade φ_2 ocorre em um estado e, em todos os estados que precedem a φ_2 ocorre φ_1, ou seja, o conjunto de estados que satisfaz a expressão $\forall(\varphi_1 \sqcup \varphi_2)$. A diferença fundamental encontra-se da função regressão. Enquanto o Algoritmos 2 utiliza a função regressão fraca o Algoritmo 3 usa a função regressão forte.

O algoritmo inicia recebendo as fórmulas φ_1 e φ_2. As instruções $SAT[\mathcal{D}]\varphi_1$ e $SAT[\mathcal{D}]\varphi_2$, linhas 2 e 4, devolvem um conjunto de estados em que as fórmulas φ_1 e φ_2 são verdadeiras no domínio $\mathcal{D}$. Na linha 7, a intersecção de W com a regressão garante que φ_1 ocorra em todos os estados do caminho que levam a φ_2. A seguir uma operação de união é realizada para garantir que os estados que satisfazem φ_2, mais não φ_1 ainda possam ser considerados no conjunto final de estados devolvidos. Quando nenhum outro estado é obtido no passo regressivo o procedimento é finalizado e o conjunto de estados no passo regressivo é devolvido.

Algoritmo 3: SAT-AU(φ_1, φ_2)

Entrada: Fórmulas φ_1 e φ_2.
Saída: Conjunto de estados que satisfaz $\forall(\varphi_1 \sqcup \varphi_2)$

```
1  begin
2      W := SAT(φ1);
3      X:=S;
4      Y := SAT(φ2);
5      while X != Y do
6          X := Y;
7          Y := Y ∪ (W ∩ regressão_forte(Y));
8          se s0 ∈ Y então
9              retorne Y;
10         fim
11     end
12     retorne falha;
13 end
```

Fonte: (SANTOS *et al.*,)

3 TRABALHOS RELACIONADOS

Neste capítulo, estabelecemos um paralelo entre alguns trabalhos relevantes da literatura que contextualizam pelo menos um dos aspectos desta proposta de mestrado: *planejamento com ações não determinísticas* e *planejamento com preferências*. Os trabalhos abordados aqui apresentam diferentes soluções para tratar representações mais próximas de ambientes reais em relação ao que os planejadores clássicos conseguem tratar.

3.1 Non-Deterministic Planning with Temporally Extended Goals: LTL over finite and infinite traces

O trabalho apresenta uma abordagem para **planejamento não determinístico** em que as **preferências qualitativas** podem ser expressas como metas estendidas na Lógica Temporal de Tempo Linear (LTL). De maneira geral, planejadores não determinísticos não são capazes de tratar metas estendidas, sendo possível tratar apenas metas de alcançabilidade simples, isto é, a satisfação de uma dada propriedade em um estado final. Então, para obter uma solução para um problema não-determinístico P com meta estendida a abordagem realiza os seguintes passos: (i) constrói um autômato de Buchi para a meta estendida expressa como uma fórmula LTL; (ii) realiza a *determinização* das ações não determinísticas, isto é, cada ação não determinística é substituída por um conjunto de ações determinísticas; (iii) constrói um problema P' sem metas estendidas; (iv) usa um planejador não-determinístico para produzir uma política para P' e; (v) finalmente, converte a política resultante em uma solução para P.

Nosso trabalho assemelha-se à proposta de (CAMACHO *et al.*, 2017) em dois aspectos. Primeiro, resolve o mesmo tipo de problema no mesmo ambiente: **planejamento com preferências qualitativas em domínios não determinísticos**. Segundo, ambos os trabalhos fazem o uso de lógicas temporais para representar as preferências qualitativas. Especificamente, (CAMACHO *et al.*, 2017) utiliza a lógica LTL enquanto o nosso trabalho utiliza a lógica α-CTL. Há, no entanto, uma grande vantagem em utilizar a lógica α-CTL ao tratar com problemas não determinísticos, uma vez que esta lógica possui a semântica capaz de expressar as ações que rotulam as transições entre os estados (PEREIRA, 2007), o que não é possível fazer diretamente com a lógica LTL. Assim, a principal diferença entre a abordagem de (CAMACHO *et al.*, 2017) e a proposta neste trabalho é que não precisamos fazer uma representação intermediária, com os autômatos, e nem é necessário realizar a *determinização* das ações, uma vez que nossos

algoritmos trabalham diretamente com ações não determinísticas. Além disso, não é possível com a lógica LTL especificar a qualidade da política a ser obtida: se fraca, forte ou forte-cíclica.

3.2 On Compiling Away PDDL3 Soft Trajectory Constraints without Using Automata

O trabalho apresenta uma abordagem para planejamento com **preferências qualitativas em ambientes determinísticos** que converte um problema de *planejamento com preferências* Π em um problema de *planejamento sem preferências* Π'. Para cada preferência, a transformação de Π em Π' adiciona uma nova meta que é falsa no estado inicial de Π'. A nova meta pode ser alcançada por uma ação que tem custo zero, mas requer que a preferência seja satisfeita, ou uma por ação que tem custo igual a utilidade da preferência e pode ser executada somente se a preferência é falsa. Devido ao método de transformação proposto, que não usa autômatos para representar especificação e verificação de sistemas, qualquer planejador clássico pode ser utilizado para resolver problemas com preferências qualitativas sem a necessidade de alteração de seus algoritmos.

Nosso trabalho assemelha-se à proposta de (PERCASSI; GEREVINI, 2019) por tratar preferências qualitativas, isto é, preferências sobre as trajetórias de caminho. Outra semelhança é que em nossa abordagem também não utilizamos autômatos para representar estas restrições e nem há necessidade, a princípio, de modificação dos algoritmos de verificação para tal. Com a semântica da lógica α-CTL, é possível especificar as preferências e utilizar diretamente os algoritmos de verificação de modelos para obtenção dos planos. No entanto, há duas principais diferenças entre nosso trabalho e a abordagem de (PERCASSI; GEREVINI, 2019). Primeiro, exploramos preferências em ambiente *não determinísticos* enquanto que a abordagem de (PERCASSI; GEREVINI, 2019) é proposta para ambientes determinísticos. Segundo, nossa abordagem não realiza transformações de um problema de planejamento com preferências em um sem preferências. Tratamos diretamente as preferências com os algoritmos de verificação de modelos.

3.3 Collaborative Planning with Encoding of Users' High-level Strategies

Na abordagem de (KIM *et al.*, 2017) as **preferências qualitativas** do usuário são codificadas como fórmulas na Lógica Temporal de Tempo Linear (LTL). Os autores utilizam dois **domínios determinísticos** da IPC, *Zenotravel* e *Satellite*. O primeiro lida com roteamento

de veículos para o transporte de passageiros. O segundo, aborda o problema de alocação de satélites para observação espacial. Assim, estratégias dos usuários como "*idosos e acompanhantes devem sentar lado a lado no avião*" ou "*o nível de combustível de um dado veículo não deve ficar abaixo de um dado limite*" são codificadas com preferências do tipo `always`, `sometime`, `at-most-once` e `at-end`. As preferências compõem uma função de recompensa que o planejador deve maximizar. Além disso, esta abordagem utiliza variáveis numéricas, sendo possível codificar preferências como "*o nível de combustível de um dado veículo não deve ficar abaixo de 10*".

Nosso trabalho assemelha-se à proposta de (KIM *et al.*, 2017) por abordar preferências qualitativas, isto é, condições na trajetória que o usuário gostaria que fossem atendidas. Outra semelhança é que, em ambos os trabalhos, as preferências são especificadas usando lógicas temporais. No entanto, há algumas diferenças entre nosso trabalho e a abordagem de (KIM *et al.*, 2017). Primeiro, não usamos uma função de recompensa para maximizar as preferências atendidas. Em nosso trabalho, caso a fórmula temporal seja satisfeita, devolvemos uma política ou, caso não seja possível satisfazê-la, devolvemos falha. Segundo, em nossa abordagem usamos variáveis proposicionais para especificação de estados, enquanto (KIM *et al.*, 2017) faz uso de variáveis numéricas. Dessa forma, para especificamos que "*durante o percurso o nível de combustível de um dado veículo não deve ficar abaixo de 10*", precisamos definir uma variável proposicional que será verdadeira se o limite foi atingido ou falsa, caso contrário. Terceiro, em nossa abordagem utilizamos a lógica α-CTL, enquanto (KIM *et al.*, 2017) utiliza a lógica LTL. Por fim, em nossa abordagem, lidamos com problemas em ambientes não determinísticos, enquanto (KIM *et al.*, 2017) aborda problemas em ambientes determinísticos.

3.4 Especificação de preferências de planos usando metas estendidas na lógica alpha-CTL

Nesse trabalho (SANTOS, 2019) apresenta uma abordagem de planejamento que lida com **preferências qualitativas em ambientes determinísticos**. Preferências do tipo *o agente deverá visitar no máximo uma vez uma determinada área* ou *em algum momento futuro o agente deverá coletar amostras de solo em uma determinada área* são codificadas por operadores modais como *always*, *sometime*, *at-most-once*. Esses operadores modais são especificados na lógica $\alpha - CTL$. Nessa abordagem, o autor utiliza algoritmos baseados na verificação de modelos para realizar uma busca simbólica regressiva no espaço de estados, com o uso de *Diagramas de*

Decisão Binárias.

Nosso trabalho assemelha-se à proposta de (SANTOS, 2019) em vários aspectos. Primeiro, por abordar preferências qualitativas. Segundo, as preferências são especificadas por operadores modais especificados na lógica $\alpha - CTL$. Terceiro, os algoritmos desenvolvidos faz uso do processo de verificação de modelos. Por fim, no uso de diagramas de decisão binárias. No entanto, a principal diferença entre nosso trabalho e a abordagem de (SANTOS *et al.*, 2019) está no fato de lidarmos com problemas em ambientes não determinísticos, enquanto (SANTOS *et al.*, 2019) aborda problemas em ambientes determinísticos.

3.5 Generalized Planning: Non-Deterministic Abstractions and Trajectory Constraints

(BONET *et al.*, 2019) propõe uma solução para planejamento generalizado, que é uma abordagem de na qual um mesmo plano pode funcionar para vários domínios. Nessa abordagem o objetivo é encontrar uma solução que resolve de uma só vez um conjunto de problemas. Por exemplo, no domínio do mundo dos blocos é possível pensar na seguinte solução: " se o bloco x não estiver limpo, pegue o bloco limpo acima de x e coloque-o na mesa". Essa é uma solução geral no sentido de que alcança a meta limpo(x) para muitos problemas. De fato, para qualquer instância do domínio do mundo do blocos.

Assim, (BONET *et al.*, 2019) propõem a geração de políticas para múltiplos domínios, por meio de uma técnica que traduz um conjunto de problemas para um único problema abstrato. Esse problema abstrato captura então a estrutura comum dos problemas concretos. A estrutura global dos problemas pode ser capturada por meio de **preferências qualitativas** expressas como fórmulas na lógica LTL. Ademais, (BONET *et al.*, 2019) mostra que para uma ampla classe de problemas que envolvem variáveis que podem ser aumentadas ou diminuídas, as restrições de trajetória podem ser compiladas, reduzindo o planejamento generalizado a **planejamento não determinístico**.

Esse trabalho assemelha-se ao nosso por lidar com preferências em ambientes não determinísticos. Entretanto, nosso trabalho utiliza a lógica α−CTL diferentemente do trabalho de (BONET *et al.*, 2019) que utiliza a lógica de tempo linear. Além disso, nosso trabalho não faz a redução em classes de problemas.

3.6 Symbolic FOND Planning for Temporally Extended Goals

(SANTOS *et al.*,) propõem um algoritmo de planejamento para resolver problemas de **planejamento não-determinísticos** com metas estendidas temporalmente (metas complexas) contemplando a qualidade da política. O planejador utiliza o arcabouço da verificação de modelos simbólica em combinação com a lógica α−CTL para resolver problemas de planejamento com metas complexas.

Nosso trabalho é uma aplicação da proposta de metas complexas em ambientes não-determinísticos (SANTOS *et al.*,) para expressar preferências qualitativas. Nosso trabalho também utiliza os algoritmos propostos por (SANTOS *et al.*, 2019) em domínios não-determinísticos e com preferências.

3.7 Resumo dos trabalhos relacionados

A Tabela 1 mostra a comparação entre os trabalhos relacionados e esta pesquisa. As principais características consideradas são: se a abordagem é não determinística ou determinística e se as preferências são especificadas em LTL ou $\alpha - CLT$.

Tabela 1 – Comparação entre os trabalhos relacionados e o proposto.

Trabalhos	Abordagem de planejamento		Especificação de preferências	
	Determinística	Não determinística	LTL	α-CTL
(CAMACHO *et al.*, 2017)	-	✓	✓	-
(PERCASSI; GEREVINI, 2019)	✓	-	-	-
(KIM *et al.*, 2017)	✓	-	✓	-
(SANTOS, 2019)	✓	-	-	✓
(BONET *et al.*, 2019)	-	✓	✓	-
(SANTOS *et al.*,)	✓	✓	-	✓
Este trabalho	-	✓	-	✓

Fonte: Elaborado pelo autor.

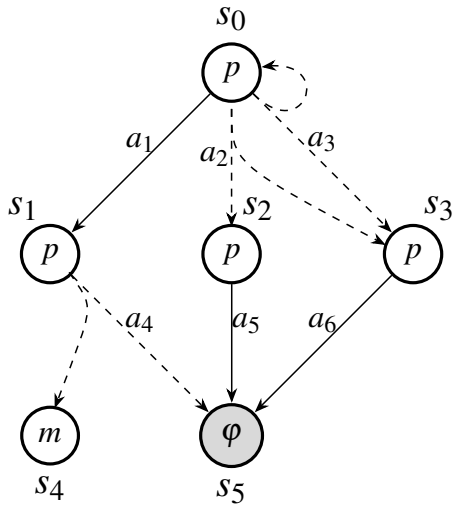
4 EXPRESSANDO PREFERÊNCIAS SOBRE POLÍTICAS NA LÓGICA α−CTL

Neste capítulo mostramos como expressar as preferências sobre políticas na lógica temporal α−CTL. Preferências sobre políticas são propriedades que devem ser preservadas ou evitadas ao longo do caminho para o alcance da meta. **Até o nosso conhecimento, este trabalho é um primeiro passo no sentido de estender a noção de preferências qualitativas** (que podem ser do tipo *always*, *sometime*, *sometime-before* e *at-most-once*) **em ambientes não determinísticos para contemplar também a qualidade da política obtida**, seja ela *fraca*, *forte* ou *forte-cíclica*. Especificamente, aplicamos a noção de metas estendidas em α-CTL (SANTOS *et al.*,) para expressar preferências.

A preferência *always*(p) expressa que a propriedade p deve ocorrer em todos os estados que levam à meta. Considere o problema de planejamento $P =< D, s_0, \varphi >$ em que D é o domínio da Figura 23, s_0 é o estado inicial e a meta φ é satisfeita no estado s_5. As ações não-determinísticas deste domínio são a_2, a_3 e a_4 (representadas por linhas pontilhadas na Figura 23). Observe que:

- A política fraca $(s_0, a_1), (s_1, a_4)$ atende a preferência *always p* do usuário;
- A política forte $(s_0, a_2), (s_2, a_6), (s_3, a_7)$ atende a preferência *always p* do usuário;
- A política forte-cíclica $(s_0, a_3), (s_3, a_7)$ atende a preferência *always p* do usuário.

Figura 23 – Domínio de planejamento não-determinístico. Ações determinísticas são representadas em linhas contínuas e ações não determinísticas, em linhas pontilhadas.



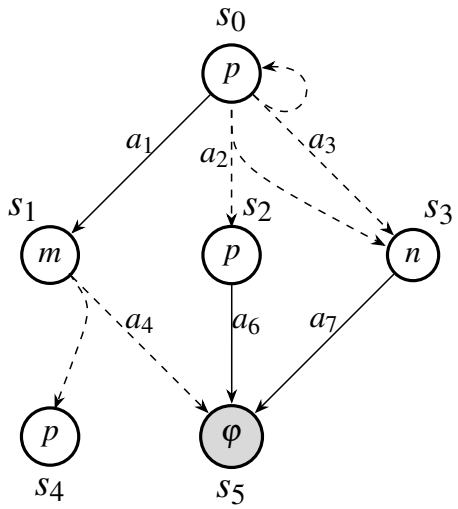
Fonte: Elaborado pelo autor

Agora, considere o problema de planejamento $P' =< D', s_0, \varphi >$ em que D' é o domínio

da Figura 24, s_0 é o estado inicial e a meta φ é satisfeita no estado s_5. Observe que:

- A política fraca $(s_0, a_1), (s_1, a_4)$ não atende a preferência *always p* do usuário;
- A política forte $(s_0, a_2), (s_2, a_6), (s_3, a_7)$ não atende a preferência *always p* do usuário;
- A política forte-cíclica $(s_0, a_3), (s_3, a_7)$ não atende a preferência *always p* do usuário;

Figura 24 – Domínio de planejamento não-determinístico. Ações determinísticas são representadas em linhas contínuas e ações não determinísticas, em linhas pontilhadas.



Fonte: Elaborado pelo autor

4.1 Especificando preferência *always* em políticas fracas

Como vimos, uma política fraca para um problema de planejamento não determinístico garante que pelo menos uma sequência de estados, obtidos após a execução de uma política, alcança o estado meta. Por sua vez, preferências qualitativas especificam condições que devem ser evitadas ou mantidas ao longo dos estados ao longo do caminho para o alcance da meta. Nesta seção, definimos a preferência qualitativa `always` para uma política fraca para um problema de planejamento não determinístico e a seguir apresentamos uma fórmula na lógica α-CTL que especifique essa preferência.

Definição 4.1.1 (Preferência *always* em políticas fracas) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico, π uma política fraca para $\mathcal{D}$, $S_{\mathcal{D}[\pi]}$ a estrutura de execução induzida pela política fraca π. A preferência always(p) ($p \in \mathbb{P}$) para uma política fraca é atendida em $S_{\mathcal{D}[\pi]}$ se, e somente se existir algum caminho de execução $\mathcal{P}_\pi$ em que $\forall k, 0 \leq k \leq i-1$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$.* ■

Definição 4.1.2 (Políticas fracas com preferência *always* em α-CTL) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico com assinatura ($\mathbb{P}, \mathbb{A}$) e always(p) ($p \in \mathbb{P}$) uma preferência do usuário em P. Esta preferência para uma política fraca pode ser expressa usando a seguinte fórmula $\alpha-CTL$:*

$$\exists[p \sqcup \varphi].$$ ■

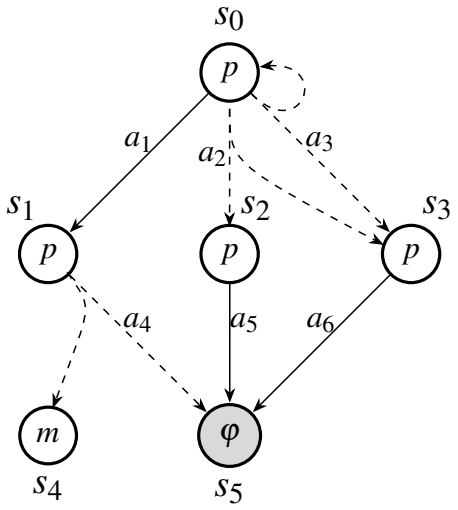
Para provar que a fórmula α-CTL $\exists[p \sqcup \varphi]$ especifica a preferência *always*(p) para um problema de planejamento $P = \langle \mathcal{D}, s_0, \varphi \rangle$ devemos mostrar que é possível obter um submodelo $S_{\mathcal{D}[\pi]} \subseteq \mathcal{D}$ induzido por uma política fraca π, onde:

1. $s_0 \in S_{\mathcal{D}[\pi]}$.
2. Existe um caminho de execução $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$ que alcança o estado meta $s_i \vDash \varphi$, a partir do estado inicial s_0 e que $\forall k, 0 \leq k \leq i-1$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$.

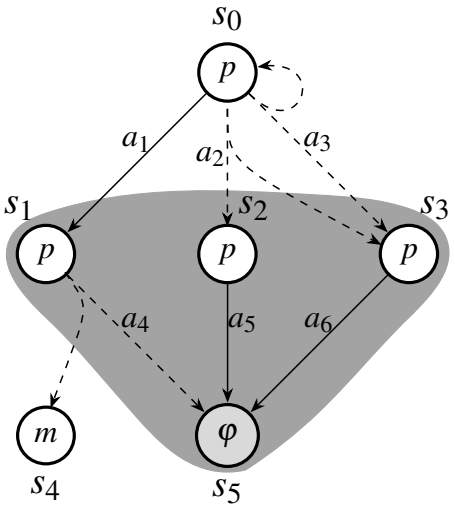
Prova 4.1.1 *Considere o problema de planejamento não determinístico $P =< D, s_0, \varphi >$. Assuma que $(\mathcal{D}, s_0) \vDash \exists(p \sqcup \varphi)$. De acordo com a semântica da lógica α-CTL (Definição 2.13), existe uma ação $a \in \mathbb{A}$ tal que, para um caminho $\mathcal{P}_\pi$ iniciado em s_0 existe um estado s_i (para $s_i \geq 0$) ao longo deste caminho tal que $s_i \vDash \varphi$ e, para cada $0 \leq k < i$, temos $(\mathcal{D}, s_k) \vDash p$. Logo $\mathcal{P}_\pi$ é um caminho de execução (Definição 2.2.6). Consequentemente, existe uma $S_{\mathcal{D}[\pi]}$ estrutura de execução induzida pela política π (Definição 2.2.5) tal que $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$. Assim, temos que $s_0 \in S_{\mathcal{D}[\pi]}$, atendendo a condição 1. Como existe um $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$, que alcança o estado meta $s_i \vDash \varphi$, a partir do estado inicial s_0 e que $\forall k, 0 \leq k < i$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$, temos que a condição 2 é atendida.* ■

Veja na Figura 25 um exemplo de computação do conjunto de estados que satisfaz a fórmula $\alpha - CTL$ $\exists[p \sqcup \varphi]$. O Algoritmo 2 inicia computando o conjunto de estados $Y = \{s_5\}$ (linha 4) que satisfaz φ e o conjunto de estados $W = \{s_0, s_1, s_2, s_4, s_5\}$ (linha 2) que satisfaz p (Figura 25a). Na primeira iteração, computa a regressão fraca do conjunto de estados que satisfaz φ e realiza a intersecção destes com o conjunto de estados que satisfaz p, Figura 25b. Esta intersecção é acumulada na variável $Y = \{s_5, (s_1, a_4), (s_2, a_5), (s_3, a_6)\}$ (linha 7, Algoritmo 2) em cada passo, até que um ponto-fixo seja alcançado. Após a segunda iteração o ponto fixo é alcançado e o valor de $Y = \{s_5, (s_1, a_4), (s_2, a_5), (s_3, a_6), (s_0, a_1), (s_0, a_2), (s_0, a_3)\}$ é obtido, Figura 25c.

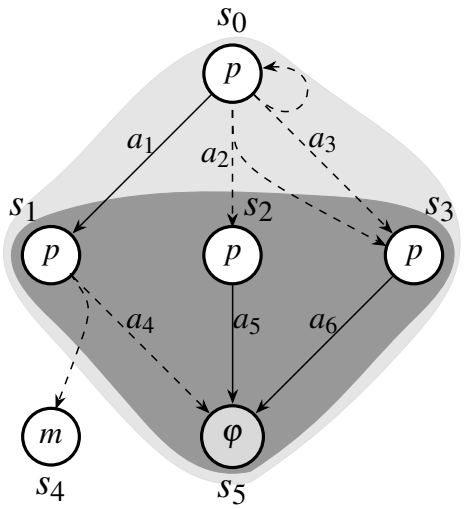
Figura 25 – Computando um submodelo com uma política fraca com preferência always(p).



(a) Estado meta



(b) Primeira iteração para $\exists(p \sqcup \varphi)$



(c) Segunda iteração para $\exists(p \sqcup \varphi)$

Fonte: Elaborado pelo autor

4.2 Especificando políticas fortes com preferência *always*

Uma política forte para um problema de planejamento não determinístico garante que todas as sequências de estados, obtidos após a execução da política, alcançam a meta. Por sua vez, preferências qualitativas especificam condições que devem ser evitadas ou mantidas ao longo do caminho para o alcance da meta. Nesta seção, definimos como obter a solução forte para um problema com preferência `always`.

Definição 4.2.1 (Preferência *always* em políticas fortes) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico, π uma política forte para $\mathcal{D}$, $S_{\mathcal{D}[\pi]}$ a estrutura de execução acíclica induzida pela política forte π. A preferência always(p) ($p \in \mathbb{P}$) para uma política forte é aten-*

dida em $S_{\mathcal{D}[\pi]}$ se, e somente se para todos os caminhos de execução $\mathcal{P}_\pi$, temos que $s_i \vDash \varphi$ e $\forall k, 0 \leq k \leq i-1, s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$. ■

Definição 4.2.2 (Políticas forte com preferência *always* em α-CTL) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico com assinatura $(\mathbb{P}, \mathbb{A})$ e always(p) $(p \in \mathbb{P})$ uma preferência do usuário em P. Esta preferência para uma política forte pode ser expressa usando a seguinte fórmula $\alpha-CTL$:*

$$\forall [p \sqcup \varphi].$$ ■

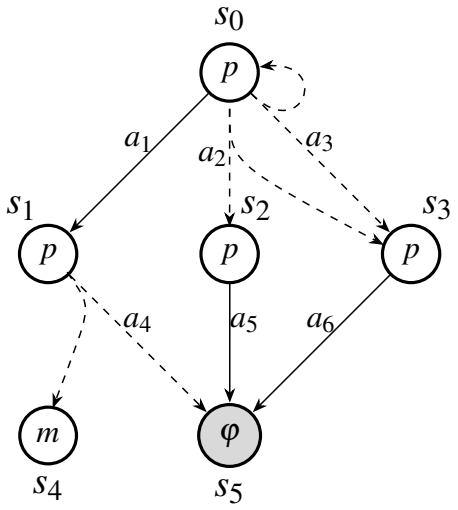
Para provar que a fórmula α-CTL $\forall [p \sqcup \varphi]$ especifica a preferência *always*(p) para uma política forte em um problema de planejamento $P = \langle \mathcal{D}, s_0, \varphi \rangle$ devemos mostrar que é possível obter um submodelo $S_{\mathcal{D}[\pi]} \subseteq \mathcal{D}$ induzido por uma política forte π, onde:

1. $s_0 \in S_{\mathcal{D}[\pi]}$.
2. Todo caminho de execução $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$ temos que $\forall k, 0 \leq k \leq i-1$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$.

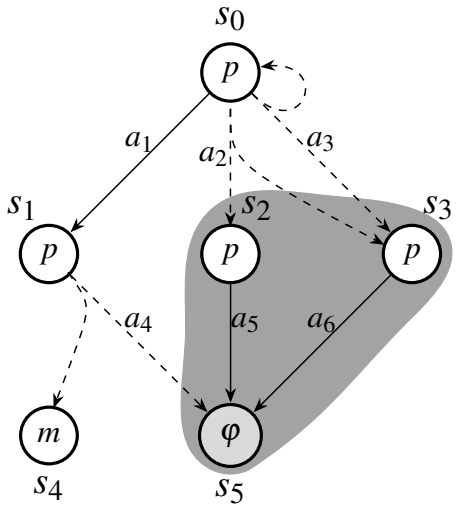
Prova 4.2.1 *Considere o problema de planejamento não determinístico $P =< D, s_0, \varphi >$. Assuma que $(\mathcal{D}, s_0) \vDash \forall (p \sqcup \varphi)$. De acordo com a semântica da lógica α-CTL (Definição 2.12), existe uma ação $a \in \mathbb{A}$ tal que, para todo o caminho $\mathcal{P}_\pi$ iniciado em s_0 existe um estado s_i (para $s_i \geq 0$) ao longo de cada caminho tal que $s_i \vDash \varphi$ e, para cada $0 \leq k < i$, temos $(\mathcal{D}, s_k) \vDash p$. Logo $\mathcal{P}_\pi$ é um caminho de execução (Definição 2.2.6). Consequentemente, existe uma $S_{\mathcal{D}[\pi]}$ estrutura de execução induzida pela política π (Definição 2.2.5) tal que todo $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$. Assim, temos que $s_0 \in S_{\mathcal{D}[\pi]}$, atendendo a condição 1. Como todo $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$ alcança o estado meta $s_i \vDash \varphi$, a partir do estado inicial s_0, e que $\forall k, 0 \leq k < i$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$, temos que a condição 2 é atendida.* ■

Veja na Figura 26 um exemplo de computação do conjunto de estados que satisfaz a fórmula $\alpha - CTL$ $\forall [p \sqcup \varphi]$. O Algoritmo 3 inicia computando o conjunto de estados $Y = \{s_5\}$ (linha 4) que satisfaz φ e o conjunto de estados $W = \{s_0, s_1, s_2, s_4\}$ (linha 2) que satisfaz p (Figura 26). Na primeira iteração, computa a regressão forte do conjunto de estados que satisfaz φ e realiza a intersecção destes com o conjunto de estados que satisfaz p, Figura 26b. Esta intersecção é acumulada na variável $Y = \{s_5, (s_2, a_6), (s_3, a_7)\}$ (linha 7, Algoritmo 3) em cada passo, até que um ponto-fixo seja alcançado. Após a segunda iteração o ponto fixo é alcançado e o valor de $Y = \{s_5, (s_2, a_6), (s_3, a_7), (s_0, a_2), (s_0, a_3)\}$ é obtido, Figura 26c.

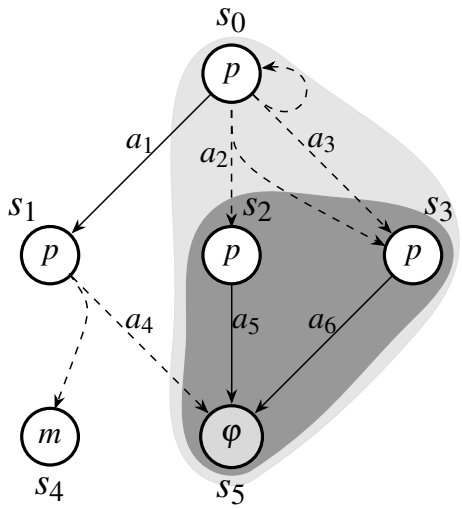
Figura 26 – Computando um submodelo com uma política forte com preferência always(p).



(a) Estado meta



(b) Primeira iteração para $\forall(p \sqcup \varphi)$



(c) Segunda iteração para $\forall(p \sqcup \varphi)$

Fonte: Elaborado pelo autor

4.3 Especificando políticas forte-cíclicas com preferência *always*

Uma política forte-cíclica para um problema de planejamento não determinístico garante que todas as sequências de estados, obtidos após a execução da política, sempre alcançam a meta sob a hipótese de que sua execução eventualmente conseguirá sair de todos os ciclos existentes. Nesta seção, definimos como especificar a preferência `always` para uma política forte cíclica.

Definição 4.3.1 (Preferência *always* em políticas forte-cíclicas) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico, π uma política forte-cíclica para $\mathcal{D}$, $S_{\mathcal{D}[\pi]}$ a estrutura de execução induzida pela política forte-cíclica π. A preferência always(p) ($p \in \mathbb{P}$) para uma polí-*

tica forte-cíclica é satisfeita em $S_{\mathcal{D}[\pi]}$ se, e somente se para todos os caminhos de execução $\mathcal{P}_\pi$, temos que $\forall k, 0 \leq k \leq i-1, s_k \in \mathcal{P}_{\pi_i}$ e $s_k \vDash p$. Além disso, é possível alcançar o estado meta a partir de cada estado $s_k \in \mathcal{P}_\pi$. ■

Definição 4.3.2 (Políticas forte-cíclica com preferência *always* em α-CTL) *Seja $P = \langle \mathcal{D}, s_0, \varphi \rangle$ um problema de planejamento não determinístico com assinatura $(\mathbb{P}, \mathbb{A})$ e always(p) ($p \in \mathbb{P}$) uma preferência do usuário em P. Esta preferência para uma política forte cíclica pode ser expressa usando a seguinte fórmula $\alpha-CTL$:*

$$\forall \boxdot \exists [p \sqcup \varphi].$$ ■

Para mostrar que a fórmula α-CTL $\forall \boxdot \exists (p \sqcup \varphi)$ especifica a preferência *always*(p) para um problema de planejamento $P = \langle \mathcal{D}, s_0, \varphi \rangle$ devemos mostrar que é possível obter um submodelo $S_{\mathcal{D}[\pi]} \subseteq D$, onde:

1. $s_0 \in S_{\mathcal{D}[\pi]}$.
2. Todo caminho de execução $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$ temos que $\forall k, 0 \leq k \leq i-1$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$.
3. É possível sair do ciclo e alcançar o estado meta a partir de qualquer estado de um caminho de execução.

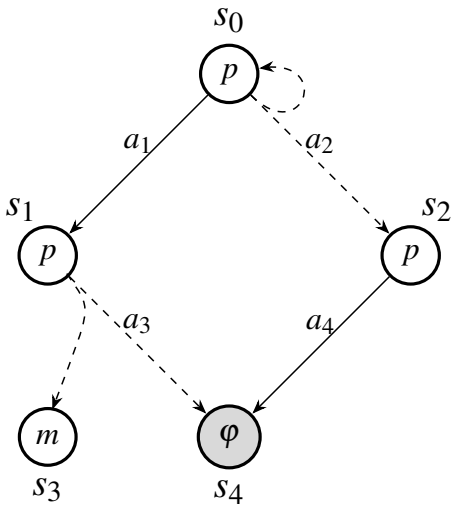
Prova 4.3.1 *Considere o problema de planejamento não determinístico $P =< D, s_0, \varphi >$. Assuma que $(\mathcal{D}, s_0) \vDash \forall \boxdot \exists (p \sqcup \varphi)$. De acordo com a semântica da lógica α-CTL (Definição 2.13), existe uma ação $a \in \mathbb{A}$ tal que, para algum caminho $\mathcal{P}_\pi$ iniciado em s_0 existe um estado s_i (para $s_i \geq 0$) ao longo deste caminho tal que $s_i \vDash \varphi$ e, para cada $0 \leq k < i$, temos $(\mathcal{D}, s_k) \vDash p$. Logo, todo caminho $\mathcal{P}_\pi$ é um caminho de execução (Definição 2.2.6). Consequentemente, existe uma $S_{\mathcal{D}[\pi]}$, estrutura de execução induzida por uma política π, tal que $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$. Assim, temos que $s_0 \in S_{\mathcal{D}[\pi]}$ atendendo a condição 1. Como existe um $\mathcal{P}_\pi \in S_{\mathcal{D}[\pi]}$, que alcança o estado meta $s_i \vDash \varphi$, a partir do estado inicial s_0 e que $\forall k, 0 \leq k < i$, $s_k \in \mathcal{P}_\pi$ e $s_k \vDash p$, temos que a condição 2 é atendida.*

Por outro lado, de acordo com a semântica da lógica α-CTL (Definição 2.8), existe uma ação $a \in \mathbb{A}$ tal que, em todos os estados s_i (para $i \geq 0$) ao longo de todos os caminhos iniciado em s_0 temos a possibilidade de se alcançar um estado meta passando apenas por estados em a propriedade p é verdadeira. Assim, mostramos que a condição 3 é atendida. ■

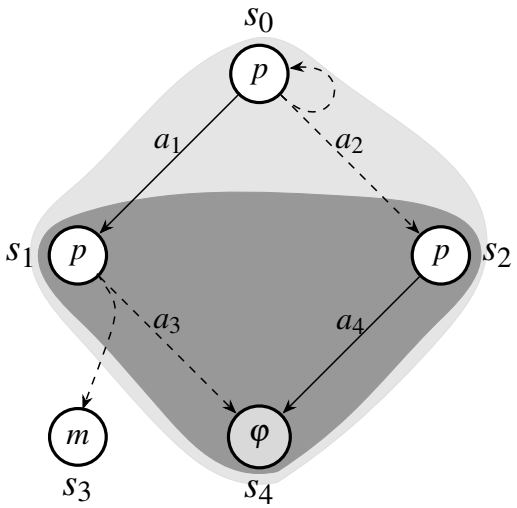
Veja na Figura 27 um exemplo de computação do conjunto de estados que satisfaz a fórmula $\alpha - CTL$ $\forall \boxdot \exists [p \sqcup \varphi]$. O Algoritmo 2 inicia computando o conjunto de estados $Y = \{s_5\}$

(linha 4) que satisfaz φ e o conjunto de estados $W = \{s_0, s_1, s_2, s_4\}$ (linha 2) que satisfaz p (Figura 25a). Na Figura 27b apresentamos a computação da fórmula $\exists(p \sqcup \varphi)$, conforme explicado anteriormente. O algoritmo 1 computa a regressão forte do conjunto de estados que satisfaz $\exists(p \sqcup \varphi)$ até o ponto fixo é alcançado e o valor de $Y = \{s_5, (s_2, a_6), (s_3, a_7), (s_0, a_2), (s_0, a_3)\}$ é obtido, Figura 26c.

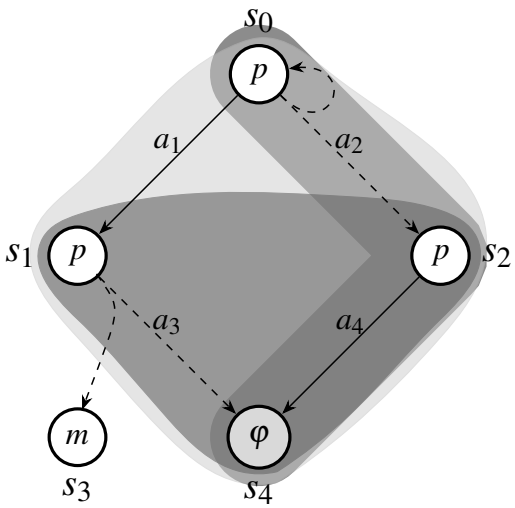
Figura 27 – Computando um submodelo com uma política forte-cíclica com preferência always(p).



(a) Estado meta



(b) computando $\exists(p \sqcup \varphi)$



(c) computando $\forall \boxdot (p \sqcup \varphi)$

Fonte: Elaborado pelo autor

5 ANÁLISE EXPERIMENTAL

A análise experimental realizada neste trabalho tem por objetivo verificar a viabilidade de se obter políticas em problemas de planejamento não determinísticos e com preferências qualitativas do tipo `always`. Os dois aspectos importantes da construção do trabalho para os experimentos são listados a seguir.

- Usamos uma versão do domínio *Rover* com preferências, proveniente da 5^a Competição Internacional de Planejamento (*International Planning Competition* - IPC). No entanto, este domínio é um domínio determinístico e, assim, o modificamos para incluir ações não determinísticas. Assim, obtemos um domínio com preferências qualitativas e não-determinístico. Até o nosso conhecimento, **não há na literatura domínios *benchmarks* com essas duas características**.
- Modificamos o planejador PACTL(PEREIRA, 2007) para realizar as chamadas para os algoritmos capazes de obter o conjunto de estados que satisfazem as fórmulas α-CTL capazes de: (i) expressar as preferências `always` e; (ii) expressar a qualidade da política obtida, se fraca, forte ou forte-cíclica. Essas chamadas foram implementadas em Java, utilizando a biblioteca de BDDs `JavaBdd`.

Os experimentos foram realizados em um servidor *Xeon Quad Core* com sistema operacional *Debian* versão 22, com 16GB de RAM e processador.

5.1 Gerando Problemas de Planejamento Não-Determinístico com Preferências Qualitativas.

Como dito anteriormente, a versão do domínio *Rovers* usada nos experimentos é baseada na versão determinística com preferências qualitativas, proveniente da 5ª Competição Internacional de Planejamento (IPC). As seguintes modificações foram realizadas neste domínio:

- inclusão de efeitos não-determinísticos em ações do domínio.
- geração de uma versão de problema para cada preferência `always` do problema original.
- inclusão da opção de escolha da qualidade da política no problema: se fraca, forte ou forte-cíclica.

A ação escolhida para ser modificada com efeitos não determinísticos foi a ação *navigate*. A Figura 28 apresenta a ação em seu formato original com efeitos determinísticos. A Figura 29 apresenta a ação *navigate* modificada para inserir o não determinismo. A palavra

reservada *oneof* indica que há duas possibilidades de deslocamento para o agente. O agente poderá desloca-se para a localização `?y` ou, caso contrário, poderá deslocar-se para a localização `?z`.

Figura 28 – Um trecho do domínio Rovers determinístico.

```
(define(domain Rover)
        ⋮
  (:action navigate
    :parameters (?r ?x ?y)
    :precondition (and (at ?r ?x)
                       (visible ?x ?y)
                       (can_traverse ?r ?x ?y)
                       (available ?r))
    :effect ( and (at ?r ?y)
                  (not(at ?r ?x))
              )
  )
                              ⋮
```

Fonte: International Planning Competition.

Figura 29 – Um trecho do domínio Rovers modificado para incluir efeitos não-determinísticos na ação.

```
(define(domain Rover)
        ⋮
  (:action navigate
    :parameters (?r ?x ?y ?z)
    :precondition (and (at ?r ?x)
                       (visible ?x ?y)
                       (can_traverse ?r ?x ?y)
                       (can_traverse ?r ?x ?Z)
                       (available ?r))
    :effect (oneof (and ( (at ?r ?y)
                       (not(at ?r ?x))
                           )
                    )
                    (and ( (at ?r ?z)
                           (not (at ?r ?x))
                           )
                    )
              )
  )
                              ⋮
```

Fonte: Elaborado pelo autor

Para a geração das instâncias dos problemas de planejamento usamos um *parser* modificado, baseado no código fonte JavaFF[1] (COLES *et al.*, 2008). O arquivo gerado pelo *parser* foi então modificado para incluir a opção da qualidade da política escolhida pelo usuário: se fraca, forte ou forte-cíclica. Assim, para cada um dos 20 arquivos de problemas do

[1] Disponivel: https://github.com/tonyallard/JavaFF

domínio original produzimos 5 instâncias com preferências `always`. Veja, por exemplo, na Tabela 2 que para o arquivo o problema p01.pddl, produzimos os problemas: p01Always1.pddl, p01Always2.pddl, p01Always3.pddl, p01Always4.pddl e p01Always5.pddl. Para cada uma dessas instâncias verificamos se é possível obter uma política fraca, uma política forte-cíclica e uma política forte.

Os arquivos de problemas variam em níveis de dificuldade como número de regiões a serem percorridas para alcançar as metas, a quantidade de metas e o número de agentes envolvidos.

5.2 Resultados

A Tabela 2 apresenta o tempo de execução para obtenção de um submodelo para políticas fraca, forte-cíclica e forte com preferência qualitativas *always* a partir de uma instância de um problema de planejamento. Ao lado de cada política encontra-se o número de passos ou iterações necessárias para se alcançar a política. Na coluna relativa a política forte-cíclica a quantidade de passos corresponde a quantidade de passos para computar uma política fraca mais a quantidade passos para computar a regressão forte.

Tabela 2 – Tempo de execução em milisegundos e o número de passos necessários para computar um submodelo com uma política fraca com preferência qualitativa always(p) a partir de uma instância de um problema de planejamento.

Arquivo - p01.pddl							
Instância	**Preferência**	**Política**					
		Fraca		**F. Cíclica**		**Forte**	
		T(*ms*)	**Passos**	**T(*ms*)**	**Passos**	**T(*ms*)**	**Passos**
p01Always1.pddl	always-at_soil_sample-waypoint0	1813	22	2127	26	1469	14
p01Always2.pddl	always-at_soil_sample-waypoint1	2161	25	2375	29	1619	17
p01Always3.pddl	always-at_soil_sample-waypoint3	1789	21	2053	25	1531	17
p01Always4.pddl	always-at_rock_sample-waypoint1	2178	25	2791	29	1689	17
p01Always5.pddl	always-at_rock_sample-waypoint6	1694	22	2098	26	1541	14
Arquivo - p02.pddl							
p02Always1.pddl	always-at_soil_sample-waypoint9	5399	20	6597	25	5031	18
p02Always2.pddl	always-at_soil_sample-waypoint4	8523	22	10841	27	8430	22
p02Always3.pddl	always-at_rock_sample-waypoint7	5911	20	6939	25	5677	20
p02Always4.pddl	always-at_rock_sample-waypoint5	8352	22	10214	27	7654	22
p02Always5.pddl	always-at_rock_sample-waypoint3	6006	18	7928	23	6434	20
Arquivo - p03.pddl							
p03Always1.pddl	always-at_soil_sample-waypoint3	10752	24	12131	27	8095	24
p03Always2.pddl	always-at_soil_sample-waypoint1	25320	28	27912	31	14180	24
p03Always3.pddl	always-at_rock_sample-waypoint3	25103	28	27759	31	13331	24

Instância	Preferência	Política					
		Fraca		F. Cíclica		Forte	
		T(*ms*)	Passos	T(*ms*)	Passos	T(*ms*)	Passos
p03Always4.pddl	always-at_soil_sample-waypoint0	13743	25	14135	28	9491	21
p03Always5.pddl	always-at_rock_sample-waypoint4	25227	26	26723	29	12958	22
Arquivo - p04.pddl							
p04Always1.pddl	always-at_soil_sample-waypoint0	3324	16	4857	22	2960	16
p04Always2.pddl	always-at_soil_sample-waypoint6	3011	16	4393	22	2872	16
p04Always3.pddl	always-at_rock_sample-waypoint8	3768	18	4479	22	3694	18
p04Always4.pddl	always-at_soil_sample-waypoint9	3450	16	4455	22	2791	16
p04Always5.pddl	always-at_rock_sample-waypoint9	4826	21	6128	28	4390	21
Arquivo - p05.pddl							
p05Always1.pddl	always-at_soil_sample-waypoint3	3264	23	3655	31	3163	23
p05Always2.pddl	always-at_soil_sample-waypoint6	3291	23	3645	31	3444	23
p05Always3.pddl	always-at_rock_sample-waypoint9	2996	23	3759	31	3363	23
p05Always4.pddl	always-at_soil_sample-waypoint4	2499	21	3228	29	3077	21
p05Always5.pddl	always-at_rock_sample-waypoint7	2664	21	3200	29	2903	21
Arquivo - p06.pddl							
p06Always1.pddl	always-at_soil_sample-waypoint10	2517596	35	-	-	-	-
p06Always2.pddl	always-at_soil_sample-waypoint19	1345996	32	-	-	-	-
p06Always3.pddl	always-at_soil_sample-waypoint5	1761539	31	-	-	-	-
p06Always4.pddl	always-at_rock_sample-waypoint15	1179631	35	-	-	-	-
p06Always5.pddl	always-at_rock_sample-waypoint17	879975	33	-	-	-	-
Arquivo - p07.pddl							
p07Always1.pddl	always-at_soil_sample-waypoint11	28712	21	39680	26	24313	21
p07Always2.pddl	always-at_soil_sample-waypoint13	18485	20	26105	25	13732	18
p07Always3.pddl	always-at_soil_sample-waypoint10	29511	21	38699	26	23688	21
p07Always4.pddl	always-at_rock_sample-waypoint8	26602	21	37960	26	18497	21
p07Always5.pddl	always-at_rock_sample-waypoint11	26966	21	38233	26	18124	21
Arquivo - p08.pddl							
p08Always1.pddl	always-at_soil_sample-waypoint5	9352	36	9834	40	7153	34
p08Always2.pddl	always-at_soil_sample-waypoint7	7295	31	7669	35	6043	30
p08Always3.pddl	always-at_soil_sample-waypoint0	7078	32	7266	36	6257	32
p08Always4.pddl	always-at_rock_sample-waypoint6	7527	32	8114	36	6615	31
p08Always5.pddl	always-at_rock_sample-waypoint4	9451	36	10038	40	7563	34
Arquivo - p09.pddl							
p09Always1.pddl	always-at_soil_sample-waypoint1	-	-	-	-	-	-
p09Always2.pddl	always-at_soil_sample-waypoint4	-	-	-	-	-	-
p09Always3.pddl	always-at_soil_sample-waypoint14	-	-	-	-	-	-
p09Always4.pddl	always-at_soil_sample-waypoint12	-	-	-	-	-	-
p09Always5.pddl	always-at_soil_sample-waypoint11	-	-	-	-	-	-
Arquivo - p10.pddl							
p10Always1.pddl	always-at_soil_sample-waypoint5	-	-	-	-	-	-
p10Always2.pddl	always-at_soil_sample-waypoint3	-	-	-	-	-	-
p10Always3.pddl	always-at_soil_sample-waypoint1	-	-	-	-	-	-
p10Always4.pddl	always-at_rock_sample-waypoint6	-	-	-	-	-	-
p10Always5.pddl	always-at_rock_sample-waypoint8	-	-	-	-	-	-

Instância	Preferência	Política					
		Fraca		F. Cíclica		Forte	
		T(*ms*)	Passos	T(*ms*)	Passos	T(*ms*)	Passos
Arquivo - p11.pddl							
p11Always1.pddl	always-at_soil_sample-waypoint4	-	-	-	-	-	-
p11Always2.pddl	always-at_soil_sample-waypoint6	-	-	-	-	-	-
p11Always3.pddl	always-at_rock_sample-waypoint7	-	-	-	-	-	-
p11Always4.pddl	always-at_rock_sample-waypoint9	-	-	-	-	-	-
p11Always5.pddl	always-at_rock_sample-waypoint2	-	-	-	-	-	-
Arquivo - p12.pddl							
p12Always1.pddl	always-at_soil_sample-waypoint8	-	-	-	-	-	-
p12Always2.pddl	always-at_soil_sample-waypoint6	-	-	-	-	-	-
p12Always3.pddl	always-at_rock_sample-waypoint9	-	-	-	-	-	-
p12Always4.pddl	always-at_soil_sample-waypoint0	-	-	-	-	-	-
p12Always5.pddl	always-at_rock_sample-waypoint8	-	-	-	-	-	-
Arquivo - p13.pddl							
p13Always1.pddl	always-at_soil_sample-waypoint12	-	-	-	-	-	-
p13Always2.pddl	always-at_rock_sample-waypoint3	-	-	-	-	-	-
p13Always3.pddl	always-at_rock_sample-waypoint5	-	-	-	-	-	-
p13Always4.pddl	always-at_soil_sample-waypoint13	-	-	-	-	-	-
p13Always5.pddl	always-at_soil_sample-waypoint10	-	-	-	-	-	-
Arquivo - p14.pddl							
p14Always1.pddl	always-at_soil_sample-waypoint8	7299	23	8176	31	6841	23
p14Always2.pddl	always-at_soil_sample-waypoint3	4928	21	1237	1602	4122	21
p14Always3.pddl	always-at_soil_sample-waypoint11	5405	21	1295	1333	4626	21
p14Always4.pddl	always-at_soil_sample-waypoint14	5291	19	1381	1547	4218	19
p14Always5.pddl	always-at_soil_sample-waypoint7	4759	21	1470	1483	4043	21
Arquivo - p15.pddl							
p15Always1.pddl	always-at_rock_sample-waypoint5	-	-	-	-	-	-
p15Always2.pddl	always-at_rock_sample-waypoint14	-	-	-	-	-	-
p15Always3.pddl	always-at_rock_sample-waypoint6	-	-	-	-	-	-
p15Always4.pddl	always-at_rock_sample-waypoint12	-	-	-	-	-	-
p15Always5.pddl	always-at_rock_sample-waypoint13	-	-	-	-	-	-
Arquivo - p16.pddl							
p16Always1.pddl	always-at_soil_sample-waypoint13	-	-	-	-	-	-
p16Always2.pddl	always-at_rock_sample-waypoint6	-	-	-	-	-	-
p16Always3.pddl	always-at_soil_sample-waypoint11	-	-	-	-	-	-
p16Always4.pddl	always-at_soil_sample-waypoint14	-	-	-	-	-	-
p16Always5.pddl	always-at_soil_sample-waypoint10	-	-	-	-	-	-
Arquivo - p17.pddl							
p17Always1.pddl	always-at_soil_sample-waypoint12	-	-	-	-	-	-
p17Always2.pddl	always-at_soil_sample-waypoint13	-	-	-	-	-	-
p17Always3.pddl	always-at_soil_sample-waypoint10	-	-	-	-	-	-
p17Always4.pddl	always-at_rock_sample-waypoint13	-	-	-	-	-	-
p17Always5.pddl	always-at_rock_sample-waypoint14	-	-	-	-	-	-
Arquivo - p18.pddl							
p18Always1.pddl	always-at_soil_sample-waypoint10	-	-	-	-	-	-

Instância	Preferência	Política					
		Fraca		F. Cíclica		Forte	
		T(*ms*)	Passos	T(*ms*)	Passos	T(*ms*)	Passos
p18Always2.pddl	always-at_soil_sample-waypoint12	-	-	-	-	-	-
p18Always3.pddl	always-at_soil_sample-waypoint13	-	-	-	-	-	-
p18Always4.pddl	always-at_rock_sample-waypoint8	-	-	-	-	-	-
p18Always5.pddl	always-at_rock_sample-waypoint13	-	-	-	-	-	-
Arquivo - p19.pddl							
p19Always1.pddl	always-at_soil_sample-waypoint12	23624	31	25324	38	19351	31
p19Always2.pddl	always-at_soil_sample-waypoint15	49479	34	53520	40	254993	33
p19Always3.pddl	always-at_soil_sample-waypoint18	48912	34	54024	40	250590	33
p19Always4.pddl	always-at_rock_sample-waypoint5	29303	31	29982	37	27236	31
p19Always5.pddl	always-at_rock_sample-waypoint4	48022	34	81287	40	279942	33
Arquivo - p20.pddl							
p20Always1	always-at_soil_sample-waypoint14	-	-	-	-	-	-
p20Always2	always-at_soil_sample-waypoint11	-	-	-	-	-	-
p20Always3	always-at_soil_sample-waypoint8	-	-	-	-	-	-
p20Always4	always-at_rock_sample-waypoint10	-	-	-	-	-	-
p20Always5	always-at_rock_sample-waypoint15	-	-	-	-	-	-

Fonte: Elaborado pelo autor

Na Tabela 2, o símbolo "-" indica que o tempo de execução passou do limite estipulado, que foi de **35 minutos**. Assim, para esses problemas, não é possível saber se não há uma solução ou se não foi possível encontrá-la.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, propomos o uso da lógica α-CTL para expressar preferências qualitativas ao longo do caminho para alcance da meta de planejamento em domínios com ações não determinísticas. Para isso, propomos fórmulas da lógica para expressar preferências do tipo `always` para políticas fracas, fortes e forte-cíclicas. Utilizamos os algoritmos simbólicos de verificação de modelos α-CTL para realizar a análise experimental no domínio *benchmark* Rovers. Este é um domínio com preferências, porém determinístico. Por esta razão, precisou ser adaptado no sentido de incluir efeitos não-determinísticos nas ações. Até o nosso conhecimento, este é o primeiro trabalho que propõe o uso de algoritmos capazes de obter políticas com preferências e que também incluam a noção de qualidade: se fraca, forte ou forte-cíclica.

Como trabalhos futuros, pretendemos verificar se a lógica α-CTL também é apropriada para expressar outros tipos de preferências como *sometime, at-most-once e sometime-before* e se os algoritmos de verificação de modelos são capazes de obter políticas para essas preferências em domínios *benchmarks*. Ademais, pretendemos estender o domínio utilizado nos experimentos utilizando os outros domínios com preferências qualitativas, modificados com ações não determinísticas, utilizados na IPC-5 (GEREVINI; LONG, 2006).

REFERÊNCIAS

AKINTUNDE, M. E. **Planning for ctl***:specified temporally extended goals via model checking. [S.l: s.n], 2017.

BAIER, J. A.; BACCHUS, F.; MCILRAITH, S. A. A heuristic search approach to planning with temporally extended preferences. In: **Proceedings of the 20th international joint conference on Artifical intelligence**. [S.l.: s.n.], 2007. p. 1808–1815.

BAIER, J. A.; BACCHUS, F.; MCILRAITH, S. A. A heuristic search approach to planning with temporally extended preferences. **Artificial Intelligence**, Elsevier, v. 173, n. 5-6, p. 593–618, 2009.

BAIER, S.; MCILRAITH, S. A. Htn planning with preferences. In: **21st Int. Joint Conf. on Artificial Intelligence**. [S.l.: s.n.], 2009. p. 1790–1797.

BONET, B.; GIACOMO, G. D.; GEFFNER, H.; RUBIN, S. Generalized planning: Non-deterministic abstractions and trajectory constraints. **arXiv preprint** arXiv:1909.12135.[S.l], 2019.

BRYANT, R. E. Binary decision diagrams. In: **Handbook of model checking**. [S.l.]: Springer, 2018. p. 191–217.

CAMACHO, A.; TRIANTAFILLOU, E.; MUISE, C.; BAIER, J. A.; MCILRAITH, S. A. Non-deterministic planning with temporally extended goals: Ltl over finite and infinite traces. In: **Thirty-First AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017.

CIMATTI, A.; PISTORE, M.; ROVERI, M.; TRAVERSO, P. Weak, strong, and strong cyclic planning via symbolic model checking. **Artificial Intelligence**, Elsevier, v. 147, n. 1-2, p. 35–84, 2003.

COLES, A.; COLES, A. Lprpg-p: Relaxed plan heuristics for planning with preferences. In: **Twenty-First International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2011.

COLES, A.; FOX, M.; HALSEY, K.; LONG, D.; SMITH, A. Managing coordination in temporal planning using planner-scheduler interaction. **To appear, Artificial Intelligence.** [S.l], v. 10, 2008.

DELGADO, K. V.; SANNER, S.; BARROS, L. N. D. Efficient solutions to factored mdps with imprecise transition probabilities. **Artificial Intelligence**, Elsevier, v. 175, n. 9-10, p. 1498–1527, 2011.

EDMUND, M.; CLARKE, J.; GRUMBERG, O.; PELED, D. A. Model checking. **MIT Press.** [S.l], p. 314, 1999.

EMERSON, E. A.; CLARKE, E. M. Using branching time temporal logic to synthesize synchronization skeletons. **Science of Computer programming**, Elsevier, v. 2, n. 3, p. 241–266, 1982.

FIKES, R. E.; NILSSON, N. J. Strips: A new approach to the application of theorem proving to problem solving. **Artificial intelligence**, Elsevier, v. 2, n. 3-4, p. 189–208, 1971.

GEREVINI, A.; LONG, D. Preferences and soft constraints in pddl3. In: **ICAPS workshop on planning with preferences and soft constraints**. [S.l.: s.n.], 2006. p. 46–53.

GEREVINI, A. E.; HASLUM, P.; LONG, D.; SAETTI, A.; DIMOPOULOS, Y. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. **Artificial Intelligence**, Elsevier, v. 173, n. 5-6, p. 619–668, 2009.

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

GRUMBERG, O.; CLARKE, E.; PELED, D. **Model checking**. [S.l.]: MIT press Cambridge, 1999.

HELMERT, M. The fast downward planning system. **Journal of Artificial Intelligence Research**, v. 26, p. 191–246, 2006.

HOFFMANN, J. Ff: The fast-forward planning system. **AI magazine**, v. 22, n. 3, p. 57–57, 2001.

HOFFMANN, J. Everything you always wanted to know about planning. In: SPRINGER. **Annual Conference on Artificial Intelligence**. [S.l.], 2011. p. 1–13.

HSU, C.-W.; WAH, B. W.; HUANG, R.; CHEN, Y. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In: **IJCAI**. [S.l.: s.n.], 2007. p. 1924–1929.

HSU, K.-C.; RUBIES-ROYO, V.; TOMLIN, C. J.; FISAC, J. F. Safety and liveness guarantees through reach-avoid reinforcement learning. **arXiv preprint** arXiv:2112.12288. [S.l], 2021.

HUTH, M.; RYAN, M. **Lógica em ciência da computação**: modelagem e argumentação sobre sistemas. 2. ed. Rio de Janeiro: [s.n.], 2008. Tradução e revisão técnica Valéria de Magalhães Iório.

JORGE, A.; MCILRAITH, S. A. *et al.* Planning with preferences. **AI Magazine**, v. 29, n. 4, p. 25–25, 2008.

KAUTZ, H.; SELMAN, B. Unifying sat-based and graph-based planning. In: **IJCAI**. [S.l.: s.n.], 1999. v. 99, p. 318–325.

KELLER, T.; EYERICH, P. Prost: Probabilistic planning based on uct. In: **ICAPS**. [S.l.: s.n.], 2012. p. 119–127.

KIM, J.; BANKS, C. J.; SHAH, J. A. Collaborative planning with encoding of users' high-level strategies. In: **Thirty-First AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017.

KINDLER, E. Safety and liveness properties: A survey. **Bulletin of the European Association for Theoretical Computer Science**, Citeseer, v. 53, n. 268-272, p. 30, 1994.

KRIPKE, S. A. A completeness theorem in modal logic. **The journal of symbolic logic**, Cambridge University Press, v. 24, n. 1, p. 1–14, 1959.

LAMPORT, L. Proving the correctness of multiprocess programs. **IEEE transactions on software engineering**, IEEE, n. 2, p. 125–143, 1977.

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

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. **PDDL-the planning domain definition language**. [S.l.]: Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational . . . , 1998.

MENEZES, Maria Viviane de. **Mudanças em Problemas de Planejamento sem Solução.** 2014. 127 f. Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2014.

MOHAMMED, B.; MOUHOUB, M.; ALANAZI, E.; SADAOUI, S. Managing weighted preferences with constraints in interactive applications. In: **ICINCO (1)**. [S.l.: s.n.], 2018. p. 265–270.

MUISE, C.; MCILRAITH, S.; BELLE, V. Non-deterministic planning with conditional effects. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2014. v. 24, n. 1.

MUISE, C. J.; MCILRAITH, S. A.; BECK, C. Improved non-deterministic planning by exploiting state relevance. In: **Twenty-Second International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2012.

PERCASSI, F.; GEREVINI, A. E. On compiling away pddl3 soft trajectory constraints without using automata. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2019. v. 29, n. 1, p. 320–328.

PEREIRA, S. d. L. **Planejamento sob incerteza para metas de alcancabilidade estendidas**. Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2007.

PEREIRA, S. L.; BARROS, L. N. de. A logic-based agent that plans for extended reachability goals. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 16, n. 3, p. 327–344, 2008.

RICHTER, S.; WESTPHAL, M.; HELMERT, M. Lama 2008 and 2011. In: **International Planning Competition**. [S.l.: s.n.], 2011. p. 117–124.

RINTANEN, J. Regression for classical and nondeterministic planning. In: **ECAI 2008**. [S.l.]: IOS Press, 2008. p. 568–572.

SANNER, S. **Relational dynamic influence diagram language (RDDL):** language description. [s.l: s.n], 2011. p. 59–79. Disponível em: http://users. cecs. anu. edu. au/ssanner/IPPC. Acesso em: 10 out. 2021.

SANNER, S.; BOUTILIER, C. Probabilistic planning via linear value-approximation of first-order mdps. **ICAPS 2006**, Citeseer, p. 74, 2006.

SANTHANAM, G. R.; BASU, S.; HONAVAR, V. Representing and reasoning with qualitative preferences for compositional systems. **Journal of Artificial Intelligence Research**, v. 42, p. 211–274, 2011.

SANTHANAM, G. R.; BASU, S.; HONAVAR, V. Representing and reasoning with qualitative preferences: Tools and applications. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, Morgan & Claypool Publishers, v. 10, n. 1, p. 1–154, 2016.

SANTOS, R. M. d. **Especificação de preferências de planos usando metas estendidas na lógica alpha-ctl**. In: . [S.l.: s.n.], 2019.

SANTOS, V. B. d.; BARROS, L. N. d. Pactl-sym: um planejador baseado em verificação simbólica de modelos. In: **Encontro Nacional de Inteligência Artificial e Computacional - ENIAC**. [S.l.]: SBC, 2017.

SANTOS, V. B. dos; BARROS, L. N. de; PEREIRA, S. do L.; VIVIANE, M. **Symbolic fond Planning for Temporally Extended Goals**. [s.l: s.n],2022.

SANTOS, V. M. B. dos; BARROS, L. N. de; MENEZES, M. V. de. Symbolic planning for strong-cyclic policies. In: IEEE. **2019 8th Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.], 2019. p. 168–173.

XIE, F.; MÜLLER, M.; HOLTE, R.; IMAI, T. Type-based exploration with multiple search queues for satisficing planning. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2014. v. 28, n. 1.