UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RODRIGO MACHADO DOS SANTOS

ESPECIFICAÇÃO DE PREFERÊNCIAS DE PLANOS USANDO METAS ESTENDIDAS NA LÓGICA ALPHA-CTL

QUIXADÁ

2019

RODRIGO MACHADO DOS SANTOS

ESPECIFICAÇÃO DE PREFERÊNCIAS DE PLANOS USANDO METAS ESTENDIDAS NA LÓGICA ALPHA-CTL

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

Orientadora: Prof. Dra. Maria Viviane de Menezes

QUIXADÁ

2019

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

S238e Santos, Rodrigo Machado dos.
Especificação de preferências de planos usando metas estendidas na lógica alpha-CTL / Rodrigo Machado dos Santos. – 2019.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2019.
Orientação: Profa. Dra. Maria Viviane de Menezes.

1. Planejamento automatizado. 2. Verificação de modelos. 3. Lógica temporal. I. Título.
CDD 004

RODRIGO MACHADO DOS SANTOS

ESPECIFICAÇÃO DE PREFERÊNCIAS DE PLANOS USANDO METAS ESTENDIDAS NA LÓGICA ALPHA-CTL

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

Aprovada em: ___/___/___

BANCA EXAMINADORA

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

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

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

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

À minha mãe Fabianne, por sempre fazer de tudo por mim. Seu amor incondicional e as suas cobranças me tornam uma pessoa melhor.

Ao meu pai Wladimir, por todas as simples palavras de incentivo. O senhor é o braço forte e a mão amiga que eu preciso.

Ao meu irmão Matheus, por ser, por toda a minha vida, o meu melhor amigo. Eu, definitivamente, seria uma pessoa pior se você não existisse e isso me faz a todo momento querer ser um irmão melhor para você.

À todos os meus parentes pelo incentivo e demonstração de carinho sempre que me viam, em especial a minha vó Fátima.

Aos meus amigos, tanto os que estão mais distantes quanto as novas amizades que fiz durante a graduação. Em especial, à Iana Mary, que além de ser minha melhor amiga, é uma companheira para a vida.

À todos os meus professores da época de IFRN como o Aílton Câmara, Diego Nascimento, Rodolfo Costa e Marjorie Ramos, que me incentivaram a seguir carreira na área da computação e que nunca esquecerei.

Aos professores Régis Pires, Ticiana Linhares, Lívia Almada, Diana Braga, Marcos Dantas, João Marcelo, Francicleber Martins e Joel Ramiro, pelos ensinamentos, cobranças e lições de vida que carregarei daqui para frente.

Aos técnico-administrativos e aos terceirizados que fazem do campus Quixadá o que ele realmente é, o melhor, mais bonito e mais organizado campus da UFC.

Aos professores Davi Romero, Leliane Barros e Alexandre Matos, pelas suas valiosas contribuições a este trabalho.

Ao professor Paulo de Tarso, por ter sido o melhor coordenador de curso que um universitário pode ter, além de ser um professor inspirador.

À minha orientadora, professora e conselheira Viviane Menezes, por ser tudo isso e minha maior fonte de inspiração profissional. Muitas vezes precisou ser mais que uma orientadora e se tornou uma incentivadora, para tirar forças de mim que nem eu mesmo sabia que tinha. Se um dia eu me tornar professor, quero ser para meus alunos o que você foi para mim. Minha eterna gratidão e reconhecimento por tudo.

À Deus, pela oportunidade da vida e por estar sempre ao meu lado.

"O Ministério da Saúde adverte, fazer TCC faz mal à saúde e pode levar a consequências graves como: gastrite nervosa, psoríase e outras. Livre-se das drogas, o texto acadêmico é uma delas."

(Rudi Santos)

RESUMO

Um dos aspectos do comportamento humano mais importante é planejar tarefas, sejam elas quais forem, em busca de alcançar alguma meta. Conhecendo isso, desenvolver algoritmos capazes de planejar de forma autônoma é um dos principais objetivos de diversas pesquisas realizadas em Inteligência Artificial, em particular, na subárea chamada de *Planejamento automatizado*. Nesta subárea, a tarefa que queremos que o algoritmo planeje de forma autônoma é chamado de *problema de planejamento*. A maioria dos planejadores da literatura são capazes de elaborar planos para *metas de alcançabilidade simples*, especificadas por meio de conjuntos de estados finais desejados. Planejadores capazes de lidar com especificações de metas mais expressivas (i.e., *metas estendidas*) têm despertado cada vez mais interesse na comunidade de planejamento. Em Pereira e Barros (2007), definiu-se a lógica α-CTL para expressar problemas de planejamento com metas de alcançabilidade estendidas. Outra abordagem que têm ganhando destaque na literatura é o *Planejamento com Preferências* (BAIER; MCILRAITH, 2008; SMITH, 2004; BRIEL *et al.*, 2004; NIGENDA; KAMBHAMPATI, 2005; BENTON *et al.*, 2007; BOUTILIER *et al.*, 2004; BRAFMAN *et al.*, 2006). Essa abordagem torna possível especificar propriedades de um plano, que o distinguem como de ”alta qualidade” ou ”o preferível”. Apesar das áreas de *planejamento com metas estendidas* e *planejamento com preferências sobre planos* serem abordagens distintas, elas parecem trabalhar sobre questões similares e ambas utilizam arcabouços teóricos em suas formulações tais como lógicas temporais. Até onde sabemos, na literatura o relacionamento entre estas áreas não é bem estabelecido, mas este trabalho teve como objetivo estabelecer uma relação entre elas e encontrar uma forma de utilizar o arcabouço de planejamento com α-CTL para solucionar problemas de planejamento com preferências.

Palavras-chave: Planejamento automatizado. Verificação de modelos. Lógica temporal.

ABSTRACT

One of the most important aspects of human behavior is to plan tasks, whatever they may be, in order to achieve some goal. Knowing this, developing algorithms capable of autonomous planning is one of the main objectives of several researches done in Artificial Intelligence, in particular, in subarea called *Automated Planning*. In this subarea, the task we want the algorithm to autonomously plan for is called the *planning problem*. Most planners in the literature are able to devise plans for *simple reachability goals* specified through desired end state sets. Planners able to handle more expressive goal specifications (i.e., extended goals) have increasingly aroused interest in the planning community. In (PEREIRA; BARROS, 2007), the α-CTL logic was defined to express planning issues with extended reachability goals. Another approach that has gained prominence in the literature is *Planning with Preferences* (BAIER; MCILRAITH, 2008; SMITH, 2004; BRIEL *et al.*, 2004; NIGENDA; KAMBHAMPATI, 2005; BENTON *et al.*, 2007; BOUTILIER *et al.*, 2004; BRAFMAN *et al.*, 2006). This approach makes it possible to specify properties of a plan, which distinguish it as ”high quality” or ”preferable”. Although the areas of planning with extended goals and planning with preferences over plans are distinct approaches, they seem to work on similar issues and both use theoretical frameworks in their formulations such as temporal logics. As far as we know, in the literature the relationship between these areas is not well established, but this work aimed to establish a relationship between them and find a way to use the α-CTL planning framework to solve planning problems with Preferences.

Keywords: Automated planning. Model checking. Temporal logic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Um dos aspectos do comportamento humano mais importante é planejar tarefas, sejam elas quais forem, em busca de alcançar alguma meta. Conhecendo isso, desenvolver algoritmos capazes de planejar de forma autônoma é um dos principais objetivos de diversas pesquisas realizadas em Inteligência Artificial, em particular, na subárea chamada de *Planejamento automatizado*. Nesta subárea, a tarefa que queremos que o algoritmo planeje de forma autônoma é chamado de *problema de planejamento*.

Um *problema de planejamento* é definido por meio do seu *domínio*, que identifica como o ambiente pode ser modificado com base nas ações que o agente pode exercer; do *estado inicial* onde o agente se encontra; e de uma *meta* a ser alcançada. A solução para um problema de planejamento é um *plano*, ou seja, uma sequência de ações que leva o agente do estado inicial para um estado que satisfaz a meta (SANTOS, 2018). A Figura 1 ilustra um robô aspirador de pó autônomo *Roomba*[1] que deve planejar sua rota de limpeza.

Figura 1 – Roomba: um robô aspirador de pó.

Fonte: (IROBOT, 2019)

Exemplo 1. *Roomba é um robô aspirador de pó que deve planejar sua rota de limpeza de modo que sempre que sua bateria estiver fraca, ele esteja suficientemente próximo a uma estação de*

[1] Disponível em: https://bit.ly/2Hz8CfW

recarga. Dessa forma, ele poderá dirigir-se a essa estação, recarregar sua bateria, e retornar ao ponto em que estava para continuar a limpeza. Observe que as estações de recarga não precisam estar na rota de limpeza planejada pelo robô, mas devem ser facilmente acessíveis a partir dessa rota (PEREIRA; BARROS, 2007).

A maioria dos planejadores da literatura são capazes de elaborar planos para *metas de alcançabilidade simples*, especificadas por meio de conjuntos de estados finais desejados. No caso do *Roomba*, o estado final desejado é o ambiente totalmente limpo, porém um plano que apenas se preocupe com a limpeza da sala não é suficiente para ele, já que sua meta é mais expressiva. Planejadores capazes de lidar com especificações de metas mais expressivas (i.e., *metas estendidas*) têm despertado cada vez mais interesse na comunidade de planejamento. Em Pereira e Barros (2007), definiu-se a lógica α-CTL para expressar problemas de planejamento com metas de alcançabilidade estendidas. No Exemplo 1 podemos observar que a meta do *Roomba* é um tipo de meta estendida, pois não há como especificá-la usando apenas um conjunto de estados finais a serem alcançados ao término da execução do plano.

Outra abordagem que têm ganhando destaque na literatura é o *Planejamento com Preferências* (BAIER; MCILRAITH, 2008; SMITH, 2004; BRIEL *et al.*, 2004; NIGENDA; KAMBHAMPATI, 2005; BENTON *et al.*, 2007; BOUTILIER *et al.*, 2004; BRAFMAN *et al.*, 2006). Essa abordagem torna possível especificar propriedades de um plano, que o distinguem como de ”alta qualidade” ou ”o preferível”. Dada alguma tarefa a ser alcançada, os usuários podem ter preferências sobre quais metas alcançar e sob quais circunstâncias, o que é denominado na literatura de *preferências de metas* (BAIER; MCILRAITH, 2008; BOUTILIER *et al.*, 2004; BRAFMAN *et al.*, 2006). Eles também podem ter preferências sobre como as metas são alcançadas - propriedades do mundo que devem ser alcançadas, mantidas ou evitadas durante a execução do plano, o que é denominado na literatura de *preferências de planos* (BAIER; MCILRAITH, 2008; SMITH, 2004; BRIEL *et al.*, 2004; NIGENDA; KAMBHAMPATI, 2005; BENTON *et al.*, 2007).

Apesar das áreas de *planejamento com metas estendidas* e *planejamento com preferências sobre planos* serem abordagens distintas, elas parecem trabalhar sobre questões similares e ambas utilizam arcabouços teóricos em suas formulações tais como lógicas temporais. Até onde sabemos, na literatura o relacionamento entre estas áreas não é bem estabelecido.

As perguntas que queremos responder com este trabalho são:

- "*Quais as semelhanças e diferenças entre os formalismos utilizados para a área de plane-*

jamento para metas de alcançabilidade estendidas e planejamento com preferências?"

- "*Seria possível expressar preferências de planos usando um arcabouço de planejamento para metas estendidas?* Como?"

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é estabelecer o relacionamento entre as áreas de planejamento para metas de alcançabilidade estendidas e planejamento com preferências sobre planos.

1.1.2 Objetivos Específicos

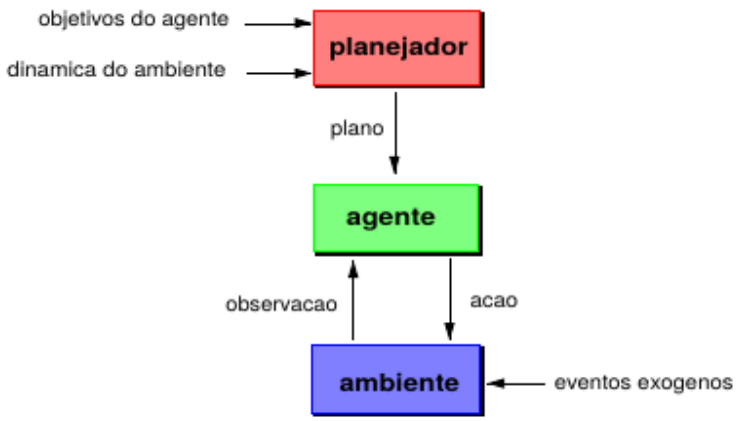
- Estudo da área de planejamento para metas de alcançabilidade estendidas.
  - estudo de planejamento como verificação de modelos (PEREIRA; BARROS, 2007).
  - estudo sobre a especificação de metas estendidas com a lógica temporal α-CTL (PEREIRA; BARROS, 2008).
- Especificação de propriedades sobre preferências de planos da literatura Son *et al.* (2015), Baral e Zhao (2004) como metas estendidas em α-CTL.
- Utilização do arcabouço de planejamento com α-CTL para solucionar problemas de planejamento com preferências.
- Análise comparativa das soluções obtidas pelo planejador baseado em α-CTL com as soluções obtidas pelos planejadores estado da arte da área de preferência de planos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Planejamento Automatizado

Planejamento Automatizado Ghallab *et al.* (2004) é um campo da Inteligência Artificial que estuda o processo de planejamento de tarefas, sob um ponto de vista computacional, visando a implementação de planejadores. Como podemos ver na Figura 2 Pereira (2007), um *planejador* é um sistema capaz de sintetizar um *plano* de ações, a partir da análise de uma descrição formal dos objetivos de um *agente* e da dinâmica de seu *ambiente*. Comportando-se dessa forma, o agente deve ser capaz de conduzir a evolução do ambiente, a despeito da ocorrência de eventos exógenos, de modo que seus objetivos possam ser atingidos (PEREIRA; BARROS, 2007).

Figura 2 – Planejamento Automatizado



Fonte: (PEREIRA, 2007)

2.2 Planejamento Clássico

Visando simplificar o problema de planejamento, a abordagem clássica (WELD, 1994) faz as seguintes suposições:

- (i) o ambiente evolui deterministicamente;
- (ii) o estado do ambiente muda apenas como efeito das ações do agente;
- (iii) a meta do agente é levar o ambiente a um estado final desejado.

Um domínio de planejamento clássico é descrito por um conjunto finito de estados do ambiente, um conjunto finito de ações que modificam o ambiente e uma função de transição

de estados que mapeia as mudanças que ocorrem de um estado para outro estado após a execução de uma ação.

Definição 2.2.1. *(Domínio de planejamento) Dado um conjunto de proposições $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, um domínio de planejamento Ghallab et al. (2004) é definido por uma tupla de três elementos $D =< S, L, T >$ em que:*

- *$S \neq \varnothing$ é um conjunto infinito de estados possíveis do ambiente;*
- *$L : S \rightarrow 2^{\mathbb{P}}$ é a função de transição de rotulação dos estados, que descreve quais proposições atômicas são verdadeiras em um estado e;*
- *$T : Sx\mathbb{A} \rightarrow S$ é uma função de transição de estados.*

A definição de domínio baseada no espaço de estados (Definição 2.2.1) corresponde à representação explícita do domínio por meio de um grafo direcionado em que os vértices representam os possíveis estados do domínio ($s \in S$) e as arestas, rotuladas por ações ($a \in \mathbb{A}$), representam todas as possíveis transições entre estados $((s, a, s') \in T)$. A Figura 3, apresenta um exemplo de um domínio de planejamento clássico representado por um grafo. Neste exemplo, $S = \{s_0, s_1, s_2, s_3\}$ representa os possíveis estados do domínio e as arestas direcionadas, rotuladas pelas ações do conjunto $\mathbb{A} = \{a_1, a_2, a_3, a_4, a_5\}$, representam as transições de estados. Por exemplo, a ação a_1 quando executada no estado s_0 leva o agente ao estado s_1. Quando uma ação a_j pode ser executada em um estado s_i dizemos que a_j é aplicável em s_i.

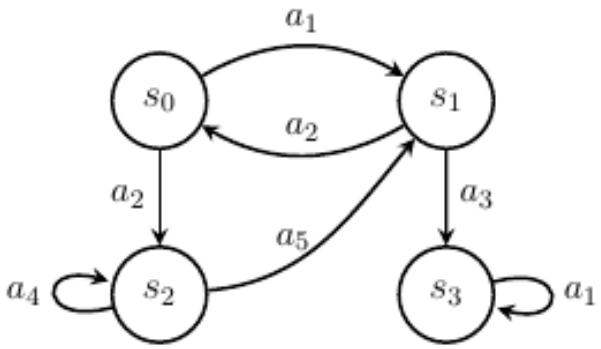
As ações possuem uma relação de *pré-condições* e *efeitos* que as fazem levar o agente de um estado a outro. As *pré-condições* são proposições atômicas que precisam ser verdadeiras no estado atual que o agente se encontra para que uma ação seja considerada aplicável para aquele estado. Os *efeitos* são proposições atômicas que serão acrescentadas ou removidas do estado atual para ”gerar” o conjunto de proposições que formam o novo estado. Ou seja, as pré-condições de a_1 são satisfeitas no estado s_0 e os efeitos da ação levam para o estado s_1.

Dado um domínio de planejamento, um *problema de planejamento* é definido em termos: (i) do estado inicial e; (ii) da meta de planejamento. Formalmente, um problema de planejamento é dado como na Definição 2.2.2.

Definição 2.2.2. *(Problema de planejamento) Dado um domínio de planejamento D, um problema de planejamento é definido por uma tupla $P = (D, s_0, \phi)$ onde:*

- *D é um domínio de planejamento (com assinatura $\mathbb{P}$ e $\mathbb{A}$);*
- *s_0 é o estado inicial do problema, e;*

Figura 3 – Domínio de planejamento clássico representado por um grafo



Fonte: (SANTOS, 2018)

- 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.*

Dado um problema de planejamento *P*, sua solução é denominada de *plano*: uma sequência de ações $\langle a_1, a_2, a_3, ..., a_n \rangle \in \mathbb{A}$, que quando executadas a partir do estado inicial, permitem que o agente alcance algum estado $g \in G$.

Quando se trata de automatizar a obtenção de um plano para um problema de planejamento, os pesquisadores e desenvolvedores da área costumam modelar os problemas em um linguagem denominada *PDDL* (Planning Domain Description Language), definindo quais são os objetos, o estado inicial e a meta.

A *International Planning Competition* (IPC) é um evento bienal organizado no contexto da *International Conference on Automated Planning and Scheduling* (ICAPS), que tem vários objetivos, incluindo a análise e o avanço do estado da arte em sistemas de planejamento automatizados; fornecer novos conjuntos de dados para serem utilizados pela comunidade de pesquisa como pontos de referência para avaliar diferentes abordagens do planejamento automatizado; enfatizar novas questões de pesquisa no planejamento; e promover a aceitação e aplicabilidade do planejamento de tecnologia. Ela se trata de uma competição na qual vários candidatos submetem seus planejadores para uma série de desafios em domínios de planejamento diferentes, que consistem em encontrar planos, caso o problema seja solucionável, ou retornar que o problema não é solucionável. Um desses vários domínios, é o *rovers*, que define o problema de planejamento para o robô explorador de Marte.

Neste domínio, o robô *rovers* tem que explorar Marte, coletando amostras de rochas e solo para fazer análise e tirar fotos do local. A meta do robô é enviar à estação espacial os resultados das análises obtidas e as fotos que ele conseguiu tirar. A Figura 4 demonstra como um

problema de planejamento para o domínio do *rovers* é definido em *PDDL*. Primeiro, define-se os `(:objects )` (objetos) do problema, que são basicamente as propriedades que são verdadeiras ou falsas nos estados do problema. Em seguida, é definido o *(:init)* (estado inicial), no qual é descrito, uma por uma, as propriedades que serão verdadeiras no estado inicial do problema. Por último, é definido a `(:goal )` (meta), que nada mais é do que uma conjunção de propriedades do problema, que define a fórmula da meta desse problema.

Figura 4 – Exemplo de arquivo *PDDL* que descreve o problema de planejamento de planos no domínio do robô de marte

```
(define (problem roverprob1) (:domain rover)
    (:objects ...  )
    (:init (visible waypoint1 waypoint0) ...  )
    (:goal (and
        (communicated_soil_data waypoint2)
        (communicated_rock_data waypoint3)
        (communicated_image_data objective1 high_res)))
)
```

Fonte: Elaborado pelo autor

2.3 Planejamento com Preferências

Na área de Planejamento com Preferências (*Preference-Based Planning (PBP)* Baier e McIlraith (2008)), relaxa-se a suposição (iii) da abordagem clássica, isto é, a suposição de que *a meta do agente é levar o ambiente a um estado final desejado*. Nesta abordagem, a descrição do problema pode ser enriquecida para expressar preferências em relação ao alcance da meta estabelecendo:

- preferências sobre quais metas são preferíveis para alcançar (no caso de não ser possível alcançar todas elas), ou ainda,
- preferências *de que caminhos tomar para que a meta seja satisfeita*.

O primeiro caso é estudado pela área de *planejamento com preferências sobre metas* Brafman *et al.* (2006), Boutilier *et al.* (2004), Baier e McIlraith (2008), Santos e Menezes (2017), também conhecida como *preferências quantitativas*. Tais abordagens são utilizadas quando se tem interesse em definir que uma meta *x* é preferível que uma meta *y*. Já o segundo caso é estudado pela área de *planejamento com preferências sobre planos* Baier e McIlraith (2008), Smith (2004), Briel *et al.* (2004), Nigenda e Kambhampati (2005), Benton *et al.* (2007), Son *et al.* (2015), também conhecida como *preferências qualitativas*. Tais algoritmos tem como

objetivo especificar propriedades que identifiquem em que situações certos planos são preferíveis a outros.

Exemplo 2. *(Preferências sobre Planos) Eu prefiro ir de avião até a cidade de uma conferência do que ir de carro, uma vez que a viagem de carro pode ser muito longa. No entanto, se não existir voos para o mesmo dia da conferência e a viagem de carro levar no máximo três horas, então eu prefiro ir de carro (SON et al., 2015).*

O Exemplo 2 mostra preferências do usuário sobre *como* chegar a uma dada cidade: a *preferência* em viajar de carro se a conferência for em um local próximo, mas se for distante, a *preferência* em ir de avião. Note que a meta do usuário é chegar na cidade, mas há preferências sobre que caminhos tomar para se alcançar esta meta.

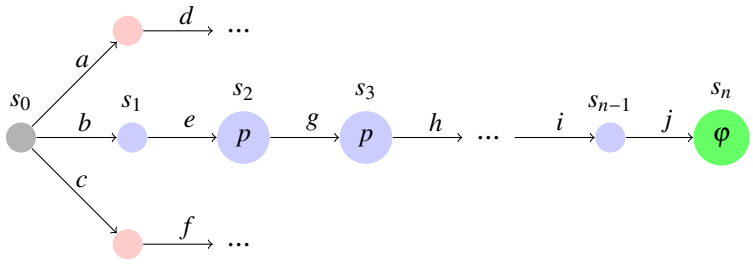
Segundo Baier e McIlraith (2008), a abordagem de preferência de planos é utilizada quando, dado um problema de planejamento P, se tem o interesse em definir uma relação especifica que um plano $\pi_i =< a_i, a_{i+1}, \cdots, a_n >$ é preferível que o plano $\pi_j =< a_j, a_{j+1}, \cdots, a_m >$, sendo que estes dois planos alcançam um estado que satisfaz a meta φ. Dessa forma, as abordagens de preferências de planos estabelecem uma ordem $\prec$ para determinar que o plano π_i é preferível que π_j se e somente se $\pi_i \prec \pi_j$.

O trabalho Sohrabi *et al.* (2009) aborda os quatro tipos de operadores para se definir preferências de planos na linguagem `PDDL`:

- `(sometime (`p`))` significa encontrar um plano em que a propriedade p seja verdadeira em algum estado no caminho para se alcançar a meta φ.
- `(always (`p`))` significa encontrar um plano em que a propriedade p seja verdadeira em todos os estados no caminho para se alcançar a meta φ.
- `(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 φ.
- `(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 φ.

A Figura 5 demonstra uma subestrutura de domínio de planejamento que satisfaz a preferência `(sometime (`p`))`, porque durante o caminho $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$ a propriedade p é verdadeira nos estados s_2 e s_3 antes de alcançar a meta φ em s_n. Perceba que mesmo que p fosse verdade em apenas um dos estados entre s_2 e s_3 ou que fosse verdade apenas em s_{n-1}, essa

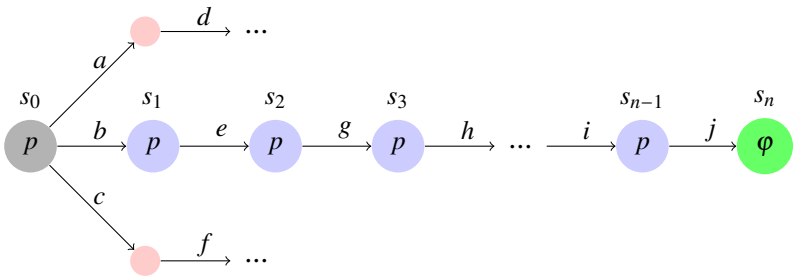
Figura 5 – Problema de planejamento com preferências de planos que satisfaz o operador `(sometime (p))`



Fonte: Elaborado pelo autor

preferência ainda seria satisfeita. Por outro lado, se todos os estados de s_0 até s_{n-1} não tivessem p como verdade, então essa meta não seria satisfeita.

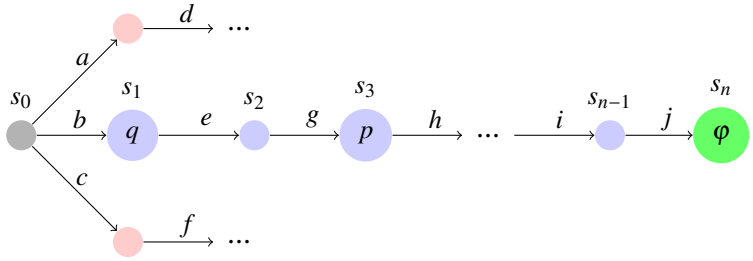
Figura 6 – Problema de planejamento com preferências de planos que satisfaz o operador `always (p)`



Fonte: Elaborado pelo autor

A Figura 6 demonstra uma subestrutura de domínio de planejamento que satisfaz a preferência `(always (p))`, porque durante o caminho $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$ a propriedade p é verdadeira em todos os estados antes de alcançar a meta φ em s_n. Se pelo menos um dos estados deste caminho não tivesse p como verdade, então essa preferência não seria aceita.

Figura 7 – Problema de planejamento com preferências de planos que satisfaz o operador `sometime-before (p),(q)`

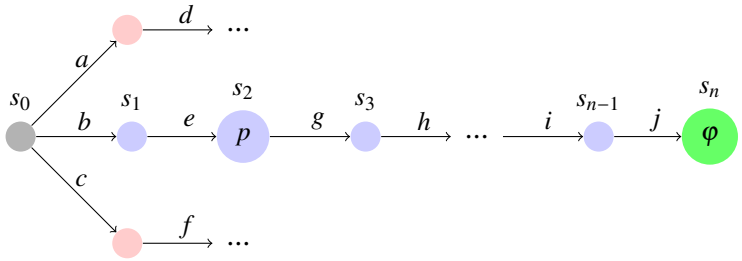


Fonte: Elaborado pelo autor

A Figura 7 demonstra uma subestrutura de domínio de planejamento que satisfaz a preferência `(sometime-before (p), (q))`, porque durante o caminho $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$

a propriedade q é verdadeira no estado s_1, que é anterior ao estado no qual a propriedade p é verdadeira, s_3, que por sua vez é anterior ao estado que alcança a meta φ, s_n. Se p não fosse verdadeiro em algum estado anterior a meta ou se q não fosse verdade em algum estado anterior ao estado que p é verdade, então essa preferência não seria satisfeita.

Figura 8 – Problema de planejamento com preferências de planos que satisfaz o operador `at-most-once` (p)



Fonte: Elaborado pelo autor

A Figura 8 demonstra uma subestrutura de domínio de planejamento que satisfaz a preferência `(at-most-once (p))`, porque durante o caminho $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$ a propriedade p é verdadeira apenas no estado s_2 antes de alcançar a meta φ em s_n. Se em nenhum estado do caminho a propriedade p fosse verdadeira, ainda sim a preferência seria satisfeita. Por outro lado, se p fosse verdade em algum outro estado do caminho, além de s_2, então essa preferência não seria satisfeita.

A 5º edição da *IPC* disponibilizou em seu site[1] os domínios de planejamento com preferências qualitativas utilizados pelos competidores. Entre eles, está o *rovers*, que retrata o mundo dos robôs exploradores de Marte.

Na versão dos problemas com preferências qualitativas, o robô de Marte deve, durante a tarefa de tentar alcançar a meta, obedecer algumas propriedades referentes ao caminho que ele toma para alcançar essa meta. A Figura 9 demonstra como este problema é descrito em *PDDL*. Primeiro, define-se os objetos, estado inicial e meta (como feito no domínio clássico da Figura 4), e, em seguida, define-se as preferências para o problema. Neste caso, deseja-se alcançar a meta passando somente por estados nos quais a propriedade `(at_soil_sample waypoint0)` é verdadeira.

[1] Disponível em: http://zeus.ing.unibs.it/ipc-5/

Figura 9 – Exemplo de arquivo *PDDL* que descreve o problema de planejamento com preferências de planos no domínio do robô de marte

```
(define (problem roverprob1) (:domain rover)
    (:objects ...  )
    (:init (visible waypoint1 waypoint0) ...  )
    (:goal (and
        (communicated_soil_data waypoint2)
        (communicated_rock_data waypoint3)
        (communicated_image_data objective1 high_res)))
    (:constraints
        (preference p1 (always (at_soil_sample waypoint0))))
)
```

Fonte: Elaborado pelo autor

2.4 Planejamento para Metas de Alcançabilidade Estendidas

Até o momento, tratamos as metas de um problema de planejamento como se elas só fossem *metas de alcançabilidade simples*, que expressam condições a serem satisfeitas apenas no estado final da execução do plano. Porém, existem também as *metas de alcançabilidade estendidas*, que podem expressar condições que devem ser satisfeitas ao longo da execução do plano.

Em planejamento, uma *meta estendida* é uma especificação de meta que estende a expressividade de uma meta simples. Metas estendidas constituem uma classe abrangente de metas que inclui, por exemplo, metas que especificam comportamentos de programas de controle (ZILLER; SCHNEIDER, 2005), metas que especificam qualidades mistas de solução (BARAL; ZHAO, 2006), metas que especificam ordem de preferência entre submetas (KRULWICH, 1992), dentre outras metas.

Metas de alcançabilidade estendidas constituem uma subclasse da classe de metas estendidas. Uma meta de alcançabilidade estendida, além de especificar uma condição a ser alcançada ao final da execução de uma política, também estabelece uma condição a ser preservada (ou evitada) em todos os estados visitados durante a execução de um plano ou política (PEREIRA; BARROS, 2008).

Definição 2.4.1. *Uma meta de alcançabilidade estendida é um par de fórmulas* (φ_1, φ_2)*, em que* φ_1 *especifica uma condição a ser preservada (ou evitada) durante a execução de um plano ou política e* φ_2 *é uma condição a ser alcançada ao final da execução desse plano ou política.*

Um problema de planejamento para meta de alcançabilidade estendida é definido por uma tupla $P = (D, s_0, (\varphi_1, \varphi_2))$*, onde:*

- $D = (S, L, T)$ *é um domínio de planejamento qualitativo com assinatura* $(\mathbb{P}, \mathbb{A})$*;*
- $s_0 \in S$ *é o estado inicial do ambiente;*
- (φ_1, φ_2) *é uma meta de alcançabilidade estendida, definida sobre* $\mathbb{P}$*.*

Retornando ao Exemplo 1, podemos perceber que a meta do Roomba não é somente encontrar a sala limpa no final da sua execução, como seria o caso se a meta fosse de alcançabilidade simples. O robô aspirador de pó, além de cuidar da limpeza do ambiente, tem que ir monitorando seu consumo de energia durante toda a execução, para que quando ele perceba que a bateria esteja acabando, retorne até o ponto de recarga e após carregado, retome a limpeza da sala de onde parou, caracterizando assim, uma meta de alcançabilidade estendida.

Algumas variações interessantes de metas de alcançabilidade estendidas são:

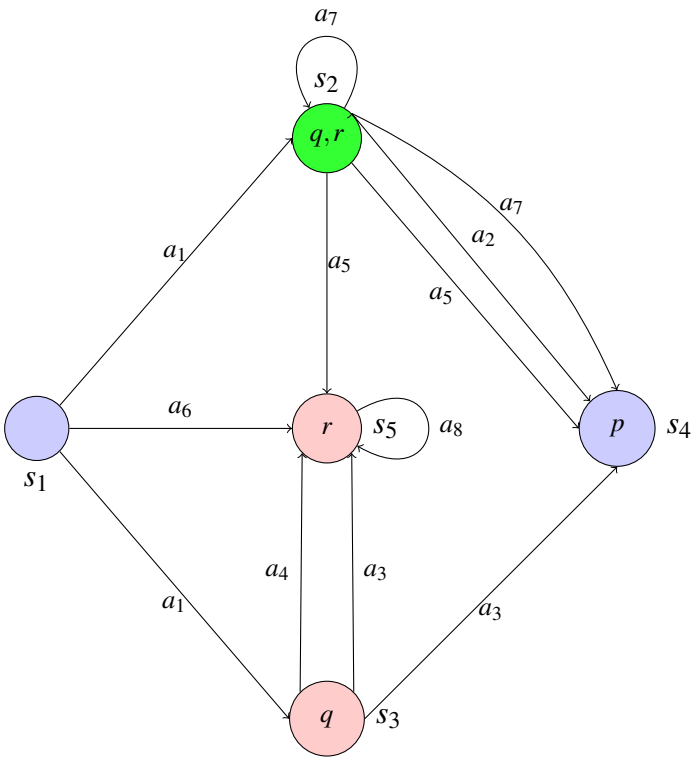
- $(\top, \varphi_2)$: alcançar a propriedade φ_2 (meta de alcançabilidade simples);
- (φ_1, φ_2): alcançar a propriedade φ_2, preservando a propriedade φ_1;
- $(\neg\varphi_1, \varphi_2)$: alcançar a propriedade φ_2, evitando a propriedade φ_1;
- $(\varphi_1 \wedge \neg\varphi_1', \varphi_2)$: alcançar a propriedade φ_2, preservando a φ_1 e evitando φ_1'.

2.5 Planejamento Sob Incerteza

Na área de Planejamento sob Incerteza Pereira e Barros (2007), relaxa-se as suposições (i) e (ii) da abordagem clássica, isto é, as suposições de que *o ambiente evolui deterministicamente* e a suposição de que *o estado do ambiente muda apenas como efeito das ações do agente*. Nesta abordagem, as ações do agente possuem efeitos incertos, resultantes da relação entre o *agente*, que planeja suas ações para atingir suas metas, e o *ambiente*, que com suas intenções desconhecidas, geram eventos exógenos que modificam os efeitos das ações dos agentes (PEREIRA; BARROS, 2007). Na área de planejamento sob incerteza, os efeitos das ações do agente podem ser não-determinísticos (CIMATTI *et al.*, 2003) ou probabilísticos (KUSHMERICK *et al.*, 1995). Esta seção irá discorrer sobre ações com efeitos não-determinísticos.

Na Figura 10, podemos perceber que para uma mesma ação, caminhos distintos podem ser alcançados, o que caracteriza efeitos não-determinísticos para as ações do agente. Por essa razão, a solução para um problema de planejamento sob incerteza não pode ser expressa sob a forma de planos como em planejamento clássico e em planejamento com preferências. Isso se dá porque uma vez que ao executar uma ação em um estado, o agente não sabe, a priori, exatamente em que estado estará no futuro. Considere, por exemplo, o estado s_1 na Figura 10, ao

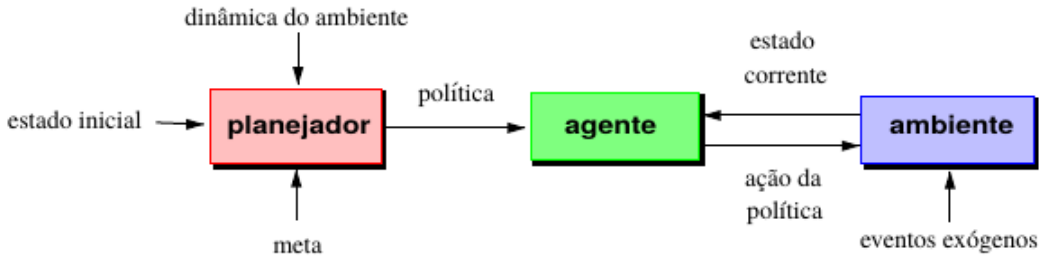
Figura 10 – Sistema de Transição de Estados Rotulado por Ações



Fonte: Adaptado de (BARAL; ZHAO, 2004)

escolher a ação a_1, o agente não sabe se, no próximo instante, estará no estado em s_2 ou em s_3 e isto pode comprometer o restante da sequência de ações dada por um plano.

Figura 11 – Modelo conceitual de planejamento sob incerteza



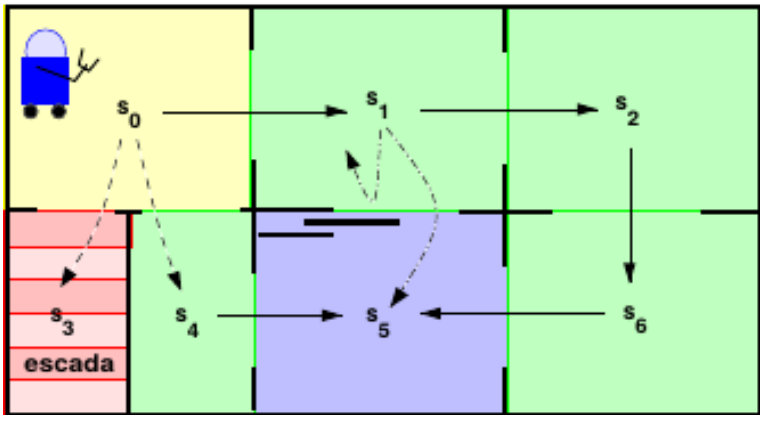
Fonte: (PEREIRA; BARROS, 2007)

Desta forma, a solução para planejamento com ações não-determinísticas é chamada de *política* Pereira e Barros (2007). Uma política é um mapeamento de estados em ações, sintetizado a partir da especificação formal da dinâmica do ambiente (i.e., domínio de planejamento), do estado inicial desse ambiente e da meta do agente nesse ambiente. Intuitivamente, uma política define um padrão de comportamento para o agente: em cada instante, ele observa o estado corrente do ambiente e executa a ação mais apropriada para esse estado, conforme

especificado pela política (PEREIRA; BARROS, 2007).

Definição 2.5.1. *(Qualidade das Soluções) Dado um problema de planejamento sob incerteza em que a meta de planejamento é especificada por meio de um conjunto de estados finais desejáveis G, uma solução para esse problema é uma política que permite ao agente alcançar um dos estados em G (PEREIRA; BARROS, 2007). Entretanto, como uma mesma política pode gerar diferentes caminhos de execução num ambiente não-determinístico, uma questão que surge é a seguinte: "que garantia um agente pode ter de que, seguindo uma determinada política, ele realmente alcançará sua meta?"*

Figura 12 – Um ambiente de planejamento sob incerteza



Fonte: (PEREIRA; BARROS, 2007)

Exemplo 3. *A Figura 12 ilustra um ambiente de planejamento sob incerteza. Nesse ambiente, quando o agente tenta mover-se de s_0 para s_4, ele pode acabar entrando em s_3 (onde há uma escada que o impede de voltar); e quando o agente tenta mover-se de s_1 para s_5, ele pode acabar permanecendo em s_1 (caso a porta automática que separa essas duas salas se feche antes que ele consiga passar). Suponha que a meta do agente nesse ambiente, que encontra-se inicialmente no estado s_0, seja alcançar o estado s_5. Para esse problema, há três soluções distintas, com qualidades também distintas:*

- π_1 = *{(s_0, entrar-em-s_1),(s_1, entrar-em-s_2),(s_2, entrar-em-s_6),(s_6, entrar-em-s_5)}*
- π_2 = *{(s_0, entrar-em-s_1),(s_1, entrar-em-s_5)}*
- π_3 = *{(s_0, entrar-em-s_4),(s_4, entrar-em-s_5)}*

Seguindo a política π_1, o agente necessariamente alcança a meta de planejamento (solução de qualidade "forte"); seguindo a política π_2, o agente alcança a meta, desde que, em algum momento, ele consiga ser rápido o suficiente para passar antes que a porta automática

se feche (solução de qualidade "forte-cíclica"); e, finalmente, seguindo a política π_3, o agente pode alcançar sua meta, mas não tem nenhuma garantia disso (solução de qualidade "fraca").

2.6 Planejamento como Verificação de Modelos

A abordagem formal de verificação de modelos (*model checking*) (CIMATTI *et al.*, 2003; GIUNCHIGLIA; TRAVERSO, 1999; PEREIRA; BARROS, 2008; MENEZES *et al.*, 2014; SANTOS, 2018; CIMATTI *et al.*, 1998; BERTOLI *et al.*, 2001) têm sido o principal formalismo para obtenção de políticas para planejamento com ações não-determinísticas, uma área conhecida como *planejamento como verificação de modelos*.

Verificação de Modelos Clarke *et al.* (1999), Müller-Olm *et al.* (1999) é uma abordagem formal para verificar se um sistema de transição de estados finito satisfaz uma determinada propriedade. Para resolver um problema de verificação de modelos computacionalmente, o sistema de transição finito deve ser representado formalmente por meio de uma estrutura de Kripke (KRIPKE, 1963) (Definição 2.6.1) e a propriedade a ser verificada deve ser especificada por uma fórmula lógica, por exemplo, em alguma lógica temporal (PNUELI, 1977; EMERSON; CLARKE, 1982; HUTH; RYAN, 2008; BARAL; ZHAO, 2004; PEREIRA; BARROS, 2008).

Definição 2.6.1. *(Estrutura de Kripke) Seja $\mathbb{P}$ um conjunto finito não vazio de proposições atômicas. Uma estrutura de Kripke sobre $\mathbb{P}$ é uma tupla $M =< S, L, T >$, onde (KRIPKE, 1963):*

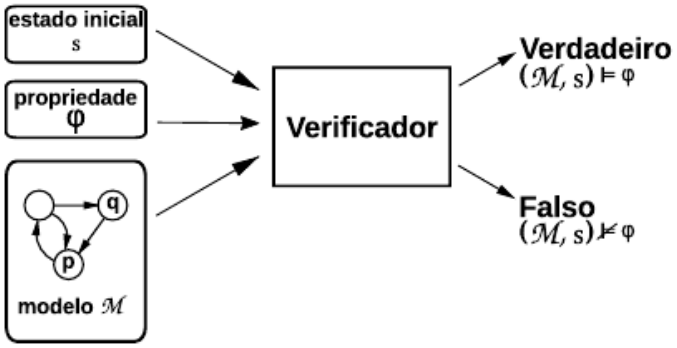
- *S é um conjunto finito não vazio de estados;*
- *$L : S \to 2^{\mathbb{P}}$ é uma função de interpretação de estados; e*
- *$T : S \to 2^S$ é uma função de transição de estados.*

Definição 2.6.2. *(Caminho) Um caminho em um modelo $\mathcal{M} = \langle S, L, T \rangle$ é uma sequência infinita de estados $s_1, s_2, s_3, \cdots$ em S tal que para cada i temos $s_i \to s_{i+1}$ (HUTH; RYAN, 2008).*

Quando um estado $s \in S$ é designado como *estado inicial*, a Estrutura de Kripke pode ser desdobrada numa árvore infinita, enraizada em s, denominada *árvore de computação.*

Um verificador de modelos (Figura 13) é um algoritmo que recebe como entrada um modelo M, uma propriedade ϕ a ser verificada e um estado s, e visita sistematicamente todos os estados do modelo verificando a validade da propriedade a partir de s. Se o modelo satisfaz a propriedade especificada, o algoritmo devolve verdadeiro; caso contrário, devolve falso (SANTOS, 2018).

Figura 13 – Verificador de modelos

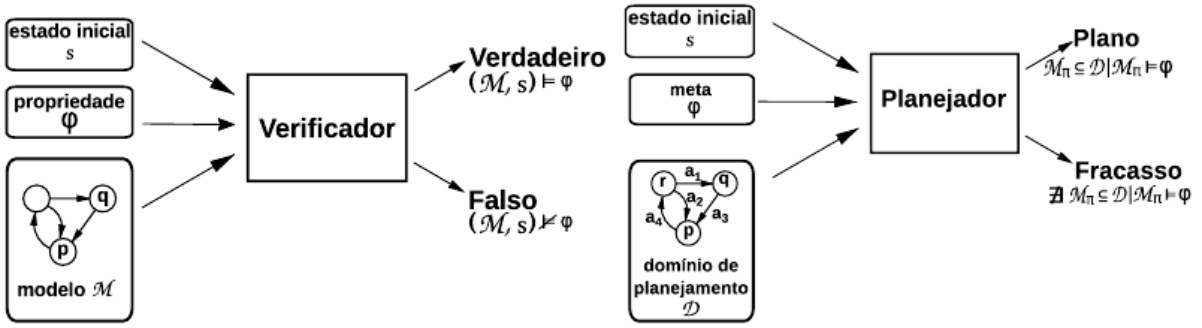


Fonte: (SANTOS, 2018)

A ideia de planejamento baseado em verificação de modelos é resolver um problema $P =< D, s_0, \phi >$ de maneira formal, herdando técnicas de verificação de modelos para visitar estados que satisfazem a meta ϕ a partir de s_0 (GIUNCHIGLIA; TRAVERSO, 1999).

A Figura 14 ilustra algumas diferenças entre um verificador de modelos e um planejador. Em vez de uma estrutura de Kripke, no planejamento baseado em verificação de modelos temos um modelo cujas transições são rotuladas pelas ações, isto é, o modelo $D = \{S, A, T\}$. Além disso, enquanto um verificador de modelos verifica a validade de uma fórmula ϕ em um modelo M (Figura 14a), um planejador procura um submodelo de D $(M_\pi \subseteq D)$, isto é, uma política, que satisfaça a meta ϕ a partir do estado inicial (Figura 14b).

Figura 14 – Verificador de modelos versus planejador baseado em verificação de modelos



(a) *Verificador de modelos.* **(b)** *Planejador baseado em verificação de modelos.*

Fonte: (SANTOS, 2018)

Além disso, essa técnica permite resolver problemas com nos quais propriedades que devem ser satisfeitas (ou evitadas) nos estados visitados durante a execução do plano, as chamadas metas de alcançabilidade estendidas (SANTOS, 2018; PEREIRA; BARROS, 2008; GHALLAB *et al.*, 2004).

2.6.1 Lógicas Temporais

Lógicas Temporais são lógicas modais cuja modalidade são capazes de expressar aspectos temporais. A ideia da lógica temporal é que uma fórmula não é estaticamente verdadeira ou falsa em um modelo. Em vez disso, os modelos contêm diversos estados e uma fórmula pode ser verdadeira em alguns estados e falsa em outros. Assim, a noção estática de verdade é substituída por uma noção *dinâmica*, na qual as fórmulas podem mudar seu valor lógico ao passo que o sistema evolui de estado em estado (HUTH; RYAN, 2008).

Nesta seção iremos apresentar as lógicas temporais LTL (PNUELI, 1977), CTL (EMERSON; CLARKE, 1982), CTL* (HUTH; RYAN, 2008), π-CTL* (BARAL; ZHAO, 2004) e α-CTL (PEREIRA; BARROS, 2007), que são utilizadas no contexto de metas estendidas, devido a expressividade que elas podem fornecer a esse tipo de meta em um problema de planejamento.

2.6.1.1 A lógica de tempo linear - LTL

A lógica de tempo linear LTL (do inglês, *Linear Temporal Logic*) (PNUELI, 1977) é uma lógica cujas fórmulas permitem especificar propriedades sobre caminhos numa árvore de computação.

Definição 2.6.3. *(Sintaxe da Lógica LTL) A sintaxe da Lógica LTL é definida como a seguir: (PNUELI, 1977)*

$$\varphi \doteq \top \mid \bot \mid p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \circ\varphi_1 \mid \Box\varphi_1 \mid \Diamond\varphi_1 \mid \varphi_1 \sqcup \varphi_2$$

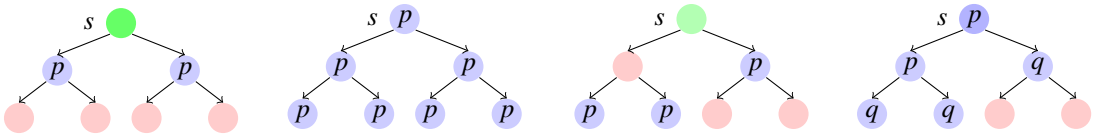
Intuitivamente, $\circ$ refere-se ao próximo estado do caminho; $\Box$ refere-se a todos os estados do caminho; $\Diamond$ refere-se a finalmente um estado no caminho; e $\sqcup$ refere-se à obrigatoriedade da verdade de uma fórmula no caminho até que outra seja verdade neste caminho.

Definição 2.6.4. *(Satisfação da fórmula LTL em um caminho) Sejam $\mathcal{M} = \langle S, L, T \rangle$ um modelo e $\pi = s_1 \rightarrow s_2 \rightarrow \cdots$ um caminho em $\mathcal{M}$, a relação de satisfação de uma fórmula em um caminho é definida como a seguir: (HUTH; RYAN, 2008)*

- $\pi \vDash \top$.
- $\pi \nvDash \bot$.
- $\pi \vDash p$ *sss* $p \in L(s_1)$.
- $\pi \vDash \neg\phi$ *sss* $\pi \nvDash \varphi$.

- $\pi \vDash \varphi_1 \wedge \varphi_2$ *sss* $\pi \vDash \varphi_1$ *e* $\pi \vDash \varphi_2$.
- $\pi \vDash \varphi_1 \vee \varphi_2$ *sss* $\pi \vDash \varphi_1$ *ou* $\pi \vDash \varphi_2$.
- $\pi \vDash \varphi_1 \rightarrow \varphi_2$ *sss* $\pi \vDash \varphi_2$ *sempre que* $\pi \vDash \varphi_1$.
- $\pi \vDash \circ\varphi$ *sss* $s_2 \vDash \varphi$.
- $\pi \vDash \Box\varphi$ *sss, para todo* $s \in \pi$ $s \vDash \varphi$.
- $\pi \vDash \Diamond\varphi$ *sss, existe algum i no caminho tal que* $s_i \vDash \varphi$.
- $\pi \vDash \varphi \sqcup \psi$ *sss, existe algum i tal que* $s_i \vDash \psi$ *e para todo* $j = 1, \cdots, i-1$, *temos que* $s_j \vDash \varphi$.

Figura 15 – Semântica dos operadores temporais da lógica LTL



(a) $(D,s) \vDash \circ p$ (b) $(D,s) \vDash \Box p$ (c) $(D,s) \vDash \Diamond p$ (d) $(D,s) \vDash p \sqcup q$

Fonte: Adaptado de (PEREIRA; BARROS, 2007)

Definição 2.6.5. *(Semântica da Lógica LTL) Seja* $D =< S, L, T >$ *uma estrutura de Kripke,* $s \in S$ *um estado dessa estrutura e* φ *uma fórmula LTL, definimos* $(D,s) \vDash \varphi$ *se para todo caminho* π *de D iniciando em s, temos que* $\pi \vDash \varphi$.

A Figura 15 ilustra a semântica dos operadores temporais da lógica LTL. Sendo s um estado inicial, pode-se constatar que: em (a) p é verdade em todos os caminhos da árvore de computação; em (b) p é verdade em todos os estados de todos os caminhos da árvore de computação; em (c) p é finalmente verdade em todos os caminhos da árvore de computação; em (d), p é verdade até que q seja verdade em todos os caminhos da árvore de computação.

2.6.1.2 A lógica de tempo ramificado - CTL

Na seção anterior, vimos que as lógica LTL quantifica as fórmulas universalmente ao longo de caminhos. Assim, propriedades que afirmam a existência de um caminho não podem ser expressas em LTL. Desta forma, a lógica CTL (do inglês, *Computation Tree Logic*) (EMERSON; CLARKE, 1982) resolve este problema permitindo a quantificação explícita (existencial ou universal) de caminhos (HUTH; RYAN, 2008). Na lógica CTL, os operadores temporais da lógica devem ser imediatamente precedidos por um quantificador de caminho ($\exists$ ou $\forall$).

Definição 2.6.6. *(Sintaxe da Lógica CTL) A sintaxe da Lógica CTL é definida como a seguir: (EMERSON; CLARKE, 1982)*

$$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid$$
$$\forall \circ \varphi_1 \mid \exists \circ \varphi_1 \mid \forall \Box \varphi_1 \mid \exists \Box \varphi_1 \mid \forall \Diamond \varphi_1 \mid \exists \Diamond \varphi_1 \mid \forall(\varphi_1 \sqcup \varphi_2) \mid \exists(\varphi_1 \sqcup \varphi_2)$$

Intuitivamente, $\exists\circ$ refere-se ao *próximos estado de algum caminho*; $\forall\circ$ refere-se aos *próximos estados de todos os caminhos*; $\exists\Box$ refere-se a *todos os estados de algum caminho*; $\forall\Box$ refere-se a *todos os estados de todos os caminhos*; $\exists\Diamond$ refere-se a *algum estado em algum caminho*; $\forall\Diamond$ refere-se a *algum estado em todos os caminhos*; $\exists\sqcup$ refere-se à existência de um caminho em que uma fórmula é obrigatoriamente verdade até que outra seja verdade neste caminho e; $\forall\sqcup$ a refere-se que uma fórmula deve ser obrigatoriamente verdade até que outra seja verdade em todos os caminhos da árvore de computação.

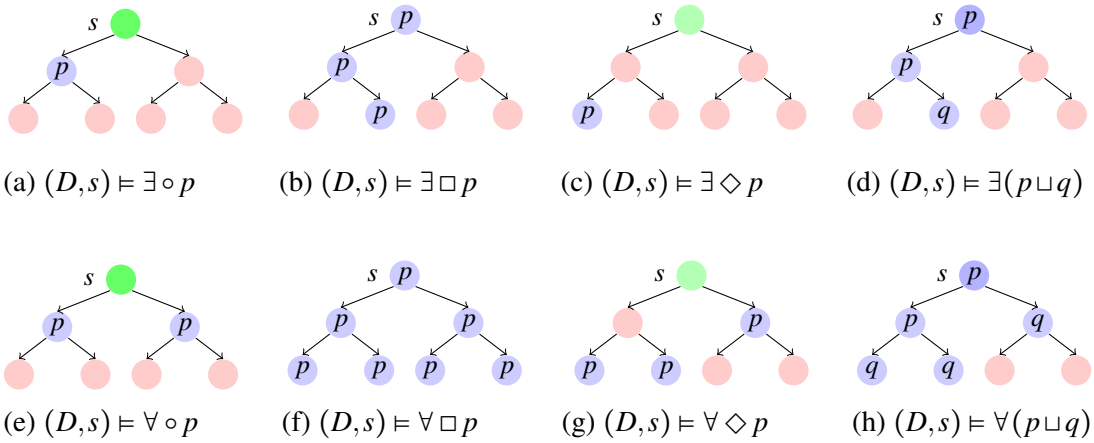
Definição 2.6.7. *(Semântica da Lógica CTL) Sejam $D =< S, L, T >$ uma estrutura de Kripke, $s \in S$ um estado e, φ uma fórmula CTL. A relação $(D,s) \vDash \varphi$ é definida por (HUTH; RYAN, 2008):*

- $(D,s) \vDash \top$ *e* $(D,s) \nvDash \bot$
- $(D,s) \vDash p$ *sss* $p \in L(s)$
- $(D,s) \vDash \neg\varphi$ *sss* $(D,s) \nvDash \varphi$
- $(D,s) \vDash \varphi_1 \wedge \varphi_2$ *sss* $(D,s) \vDash \varphi_1$ *e* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \varphi_1 \vee \varphi_2$ *sss* $(D,s) \vDash \varphi_1$ *ou* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \varphi_1 \rightarrow \varphi_2$ *sss* $(D,s) \nvDash \varphi_1$ *ou* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \forall \circ \varphi_1$ *sss para todo* s_1 *tal que* $(s,s_1) \in T$*, temos* $(D,s_1) \vDash \varphi_1$
- $(D,s) \vDash \exists \circ \varphi_1$ *sss para algum* s_1 *tal que* $(s,s_1) \in T$*, temos* $(D,s_1) \vDash \varphi_1$
- $(D,s) \vDash \forall \Box \varphi_1$ *é verdade sss para todos os caminhos* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$*, onde* $s_1 = s$*, e para todos os* s_i *ao longo deste caminho, temos* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \exists \Box \varphi_1$ *é verdade sss existe um caminho* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$*, onde* $s_1 = s$*, e para todos os* s_i *ao longo deste caminho, temos* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \forall \Diamond \varphi_1$ *é verdade sss para todos os caminhos* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$*, onde* $s_1 = s$*, existe algum* s_i *ao longo deste caminho, tal que* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \exists \Diamond \varphi_1$ *é verdade sss existe algum caminho* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$*, onde* $s_1 = s$*, e para algum* s_i *ao longo deste caminho, temos* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \forall(\varphi_1 \sqcup \varphi_2)$ *é verdade sss para todos os caminhos* $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$*, onde* $s_1 = s$*, este caminho satisfaz* $\varphi_1 \sqcup \varphi_2$*, isto é, existe algum* s_i *ao longo deste caminho tal que*

$(D,s_i) \vDash \varphi_2$ *e, para cada* $j < i$*, temos* $(D,s_j) \vDash \varphi_1$

- $(D,s) \vDash \forall(\varphi_1 \sqcup \varphi_2)$ *é verdade sss para todos os caminhos* $s_1 \to s_2 \to s_3 \to \ldots$*, onde* $s_1 = s$*, este caminho satisfaz* $\varphi_1 \sqcup \varphi_2$ *como especificado no item anterior*

Figura 16 – Semântica dos operadores temporais da lógica CTL



(a) $(D,s) \vDash \exists \circ p$ (b) $(D,s) \vDash \exists \Box p$ (c) $(D,s) \vDash \exists \Diamond p$ (d) $(D,s) \vDash \exists(p \sqcup q)$

(e) $(D,s) \vDash \forall \circ p$ (f) $(D,s) \vDash \forall \Box p$ (g) $(D,s) \vDash \forall \Diamond p$ (h) $(D,s) \vDash \forall(p \sqcup q)$

Fonte: (PEREIRA; BARROS, 2007)

A Figura 16 ilustra a semântica dos operadores temporais da lógica CTL. Sendo s um estado inicial, pode-se constatar que: em (a) existe um caminho em que p é verdade, para o próximo estado do caminho; em (b) existe um caminho em que p é verdade em todos os estados; em (c) existe um caminho em que p é finalmente verdade; em (d) existe um caminho em que p é verdade até que q seja verdade; em (e) para todos os caminhos, p é verdade, para o próximo estado do caminho; em (f) para todos os caminhos, p é verdade em todos os estados dos caminhos; em (g) para todos os caminhos, p será finalmente verdade; e em (h) para todos os caminhos, p será verdade até que q seja verdade.

*2.6.1.3 A lógica que une os poderes de expressão da LTL e CTL - CTL**

A lógica CTL permite a quantificação explícita dos caminhos e, neste aspecto, é mais expressiva do que a LTL. No entanto, ela não permite que se selecione um subconjunto de caminhos através de uma fórmula, como a LTL. Sob este aspecto a LTL é mais expressiva. Por exemplo, na LTL podemos dizer "todos os caminhos que têm um p ao longo dele também tem um q" escrevendo $\Diamond p \to \Diamond q$. Não é possível escrever isso na CTL devido à restrição de que todo $\Diamond$ tem que estar associado a um $\forall$ ou $\exists$. A fórmula $\forall \Diamond p \to \forall \Diamond q$ significa algo bem diferente: ela diz que "se todos os caminhos têm um p ao longo deles, então todos os caminhos têm um q

ao longo deles" (HUTH; RYAN, 2008).

Definição 2.6.8. *(Sintaxe da Lógica CTL*) Seja p um átomo proposicional, φ uma fórmula de estado e β uma fórmula de caminho. A sintaxe de CTL* envolve duas classes de fórmulas (HUTH; RYAN, 2008):*

- *fórmulas de estado, que são avaliadas em estados:*

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \forall\beta \mid \exists\beta$$

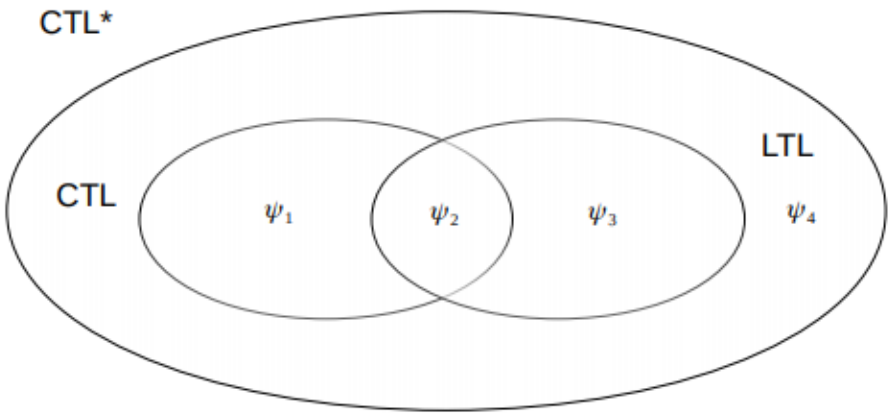
  em que β é qualquer fórmula de caminho.

- *fórmulas de caminho, que são avaliadas ao longo de caminhos:*

$$\beta ::= \varphi \mid \neg\beta_1 \mid \beta_1 \wedge \beta_2 \mid \beta_1 \vee \beta_2 \mid \beta_1 \rightarrow \beta_2 \mid \circ\beta_1 \mid \Box\beta_1 \mid \Diamond\beta_1 \mid \beta_1 \sqcup \beta_2$$

CTL^* é uma lógica que combina os poderes de expressão da LTL e da CTL, retirando a restrição da CTL de que todo operador temporal $(\circ, \Box, \Diamond, \sqcup)$ tenha que estar associado a um único quantificador de caminhos $(\forall, \exists)$ (HUTH; RYAN, 2008). A Figura 17 ilustra a relação entre as lógicas até aqui abordadas: LTL, CTL e CTL^*.

Figura 17 – Os poderes de expressão das lógicas LTL, CTL e CTL^*



Fonte: (HUTH; RYAN, 2008)

2.6.1.4 A lógica de tempo ramificado baseada em ações - α-CTL

Vimos anteriormente que a Lógica CTL é útil quando se deseja expressar propriedades que afirmam a existência de um caminho. Porém, o trabalho de Pereira e Barros (2007) mostra que esta lógica não é adequada para a área de planejamento automatizado, uma vez que ela não é capaz de representar as ações que estão por trás das transições entre os estados. Como

forma de contornar essa limitação, Pereira e Barros (2007) propôs uma extensão para a lógica CTL, denominada α-CTL. Esta nova lógica possui uma semântica baseada em ações, isto é, o modelo formal do sistema não é uma Estrutura de Kripke e sim um Sistema de Transições de Estados cujas transições são rotuladas pelas ações.

Para diferenciar as fórmulas α-CTL daquelas em CTL são utilizados um conjunto diferenciado de símbolos ”pontuados” para representar os operadores temporais:

Definição 2.6.9. *(Sintaxe α-CTL) Seja $\mathbb{P}$ o conjunto de símbolos proposicionais. A sintaxe de α-CTL é definida como (PEREIRA; BARROS, 2007):*

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

Intuitivamente, $\exists\odot$ refere-se a *existe um caminho que, para alguma ação, p é verdade*; $\forall\odot$ refere-se a *para todos os caminhos, para alguma ação, p é verdade*; $\exists\boxdot$ refere-se a *existe um caminho que, para alguma ação, p será sempre verdade*; $\forall\boxdot$ refere-se a *para todos os caminhos, para alguma ação, p será sempre verdade*; $\exists\diamondsuit$ refere-se a *existe um caminho que, para alguma ação, em algum estado p será verdade*; $\forall\diamondsuit$ refere-se a *para todos os caminhos, para alguma ação, em algum estado p será verdade*; $\exists\sqcup$ refere-se a *existe um caminho, para alguma ação, em que uma fórmula deve ser verdade até que uma outra seja* e; $\forall\sqcup$ refere-se a *para todos os caminhos, para alguma ação, uma fórmula deve ser obrigatoriamente verdade até que outra seja verdade* (PEREIRA; BARROS, 2007).

Definição 2.6.10. *(Sistema de Transição de Estados Rotulado por Ações) Seja $\mathbb{P}$ um conjunto finito não vazio de átomos proposicionais e $\mathbb{A}$ um conjunto não vazio de ações, um sistema de transições de estados rotulado por ações é uma tupla $M =< S, L, T >$, em que (PEREIRA; BARROS, 2007):*

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

Definição 2.6.11. *(Semântica α-CTL) Seja $M =< S, L, T >$ 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:*

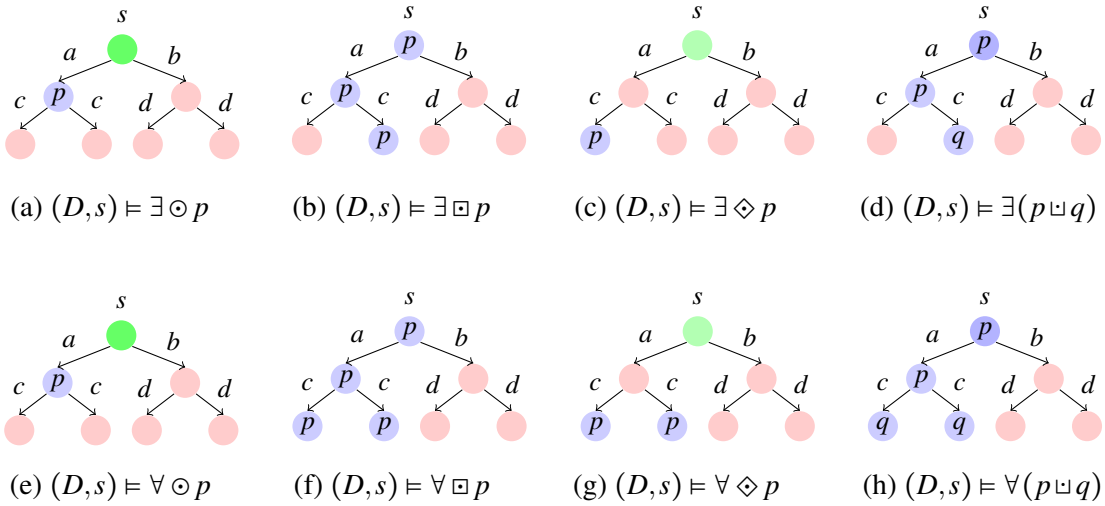
- $(D, s) \vDash \top$ *e* $(D, s) \nvDash \bot$

- $(D,s) \vDash p$
- $(D,s) \vDash \neg p$ *sss* $(D,s) \nvDash p$
- $(D,s) \vDash \varphi_1 \wedge \varphi_2$ *sss* $(D,s) \vDash \varphi_1$ *e* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \varphi_1 \vee \varphi_2$ *sss* $(D,s) \vDash \varphi_1$ *ou* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \varphi_1 \rightarrow \varphi_2$ *sss* $(D,s) \nvDash \varphi_1$ *ou* $(D,s) \vDash \varphi_2$
- $(D,s) \vDash \forall \odot \varphi_1$ *sss existe ação* $a \in \mathbb{A}$ *tal que para todo* s_1 *em que* $(s,a,s_1) \in T$*, temos* $(D,s_1) \vDash \varphi_1$
- $(D,s) \vDash \exists \odot \varphi_1$ *sss existe ação* $a \in \mathbb{A}$ *tal que para algum* s_1 *em que* $(s,a,s_1) \in T$*, temos* $(D,s_1) \vDash \varphi_1$
- $(D,s) \vDash \forall \boxdot \varphi_1$ *é verdade sss existe ação* $a \in \mathbb{A}$ *em que todos os estados* s_i *ao longo de todos caminhos iniciando em* (s,a,s_1)*, temos* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \exists \boxdot \varphi_1$ *é verdade sss existe ação* $a \in \mathbb{A}$ *em que todos os estados* s_i *ao longo de algum caminho iniciando em* (s,a,s_1)*, temos* $(D,s_i) \vDash \varphi$
- $(D,s) \vDash \forall \diamondsuit \varphi_1$ *é verdade sss existe ação* $a \in \mathbb{A}$ *tal que para todos os caminhos iniciando em* (s,a,s_1) *existe um estado* s_i *tal que* $(D,s_i) \vDash \varphi$*.*
- $(D,s) \vDash \exists \diamondsuit \varphi_1$ *é verdade sss existe ação* $a \in \mathbb{A}$ *tal que existe um caminho iniciando em* (s,a,s_1) *no qual existe um estado* s_i *tal que* $(D,s_i) \vDash \varphi$*.*
- $(D,s) \vDash \forall(\varphi_1 \sqcup \varphi_2)$ *sss existe ação* $a \in \mathbb{A}$ *tal que para todos os caminhos iniciando em* (s,a,s_1) *existe algum estado* s_i *ao longo deste caminho tal que* $(D,s_i) \vDash \varphi_2$ *e, para cada* $j < i$*, temos* $(D,s_j) \vDash \varphi_1$*.*
- $(D,s) \vDash \forall(\varphi_1 \sqcup \varphi_2)$ *sss existe ação* $a \in \mathbb{A}$ *tal que para algum caminho iniciando em* (s,a,s_1) *existe algum estado* s_i *ao longo deste caminho tal que* $(D,s_i) \vDash \varphi_2$ *e, para cada* $j < i$*, temos* $(D,s_j) \vDash \varphi_1$*.*

2.6.1.5 A Lógica CTL para políticas - π-CTL**

Vimos anteriormente que a lógica CTL* une o poder de expressividade da LTL com a CTL. No entanto, o trabalho de Baral e Zhao (2004) mostra que esta lógica não é capaz de expressar políticas para problemas de planejamento não-determinísticos. Isso se deve por causa da semântica dos operadores $\exists$ e $\forall$, que estão vinculados à relação geral de transição e não a uma determinada política. Para contornar essa limitação, Baral e Zhao (2004) propõe introduzir novos operadores, $\forall_\pi$ e $\exists_\pi$ que tratam caminhos apenas para uma política π em consideração.

Figura 18 – Semântica dos operadores temporais da lógica α-CTL



(a) $(D,s) \vDash \exists \odot p$ (b) $(D,s) \vDash \exists \boxdot p$ (c) $(D,s) \vDash \exists \diamondsuit p$ (d) $(D,s) \vDash \exists (p \sqcup q)$

(e) $(D,s) \vDash \forall \odot p$ (f) $(D,s) \vDash \forall \boxdot p$ (g) $(D,s) \vDash \forall \diamondsuit p$ (h) $(D,s) \vDash \forall (p \sqcup q)$

Fonte: (MENEZES, 2014)

Definição 2.6.12. *(Sintaxe π-CTL*) Seja p um átomo proposicional, φ uma fórmula de estado e β uma fórmula de caminho. A sintaxe de π-CTL* é definida como: (BARAL; ZHAO, 2004)*

- *fórmulas de estado, que são avaliadas em estados:*

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \forall\beta \mid \exists\beta \mid \forall_\pi\beta \mid \exists_\pi\beta$$

- *fórmulas de caminho, que são avaliadas ao longo de caminhos:*

$$\beta ::= \varphi \mid \neg\beta_1 \mid \beta_1 \wedge \beta_2 \mid \beta_1 \vee \beta_2 \mid \beta_1 \rightarrow \beta_2 \mid \circ\beta_1 \mid \Box\beta_1 \mid \Diamond\beta_1 \mid \beta_1 \sqcup \beta_2$$

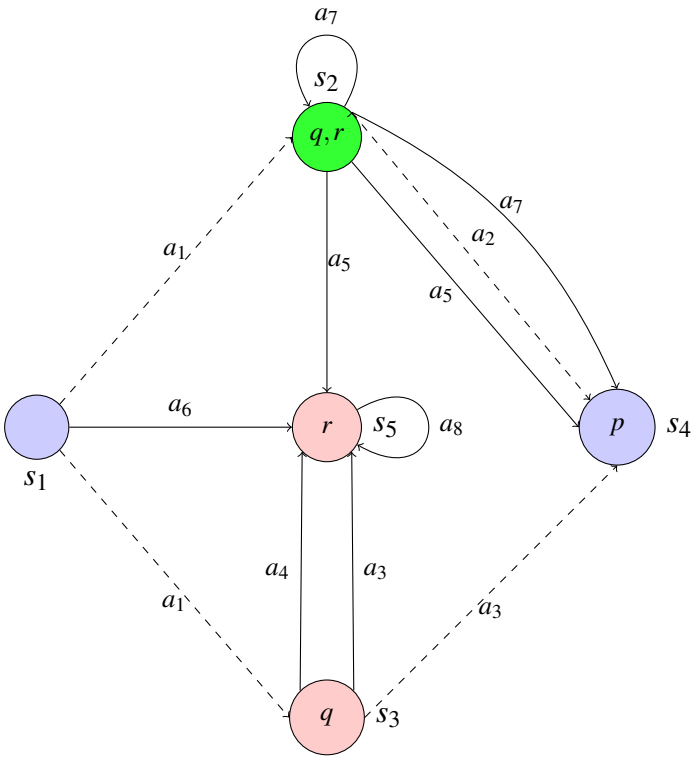
Intuitivamente, $\exists\circ$ refere-se a um *próximo estado de algum caminho*; $\forall\circ$ refere-se aos *próximos estados de todos os caminhos*; $\exists_\pi\circ$ refere-se a um *próximo estado de algum caminho da política em consideração*; $\forall_\pi\circ$ refere-se aos *próximos estados de todos os caminhos da política em consideração*; $\exists\Box$ refere-se a *todos os estados de algum caminho*; $\forall\Box$ refere-se a *todos os estados de todos os caminhos*; $\exists_\pi\Box$ refere-se a *todos os estados de algum caminho da política em consideração*; $\forall_\pi\Box$ refere-se a *todos os estados de todos os caminhos da política em consideração*; $\exists\Diamond$ refere-se a *algum estado em algum caminho*; $\forall\Diamond$ refere-se a *algum estado em todos os caminhos*; $\exists_\pi\Diamond$ refere-se a *algum estado em algum caminho da política em consideração*; $\forall\Diamond$ refere-se a *algum estado em todos os caminhos da política em consideração*; $\exists\sqcup$ refere-se à existência de um caminho em que uma fórmula é obrigatoriamente verdade até que outra seja verdade neste caminho e; $\forall\sqcup$ a refere-se que uma fórmula deve ser obrigatoriamente verdade até que outra seja verdade em todos os caminhos da árvore de computação; $\exists_\pi\sqcup$ refere-se à existência de um caminho em que uma fórmula é obrigatoriamente verdade até que outra seja

verdade neste caminho da política em consideração e; $\forall_\pi \sqcup$ refere-se a uma fórmula que deve ser obrigatoriamente verdade até que outra seja verdade em todos os caminhos da árvore de computação da política em consideração.

Na semântica da Logica $\pi - CTL^*$ consideraremos: um sistema de transição de estados rotulado por ações $D =< S, L, T >$, representando o domínio de planejamento; um problema de planejamento $P =< D, s_0, \varphi >$ em que s_0 é o estado inicial e um φ é uma fórmula descrevendo a meta de planejamento; e π é uma política solução para o problema P. Dizemos ainda que T_π são as transições induzidas pela política.

A Figura 19 ilustra o sistema de transição de estados $D =< S, L, T >$, representando um domínio de planejamento, em que os estados em $S = \{s_1, s_2, s_3, s_4, s_5\}$ são rotulados por elementos de $\mathbb{P} = \{p, q, r\}$ e as transição T são rotuladas pelas ações do conjunto $\mathbb{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$. Dado um problema de planejamento neste domínio em que o estado inicial é s_1 e a meta é alcançar um estado em que p é verdade. Uma possível solução para este problema é dada pela política forte $\pi_1 = \{(s_1, a_1), (s_2, a_2), (s_3, a_3)\}$. Chamamos de T_π as transições induzidas por uma política π. Na Figura 19 as transições T_{π_1} são ilustradas por arestas tracejadas.

Figura 19 – Sistema de transição de estados rotulado por ações



Fonte: Adaptado de (BARAL; ZHAO, 2004)

Definição 2.6.13. *(Semântica π-CTL*) Seja $D =< S, L, T >$ um sistema de transição de estados rotulado por ações; e seja π uma política para o problema de planejamento $P =< D, s, \varphi >$.*

A verdade das fórmulas de estado é definida em relação a tripla (D, s, π) em que:

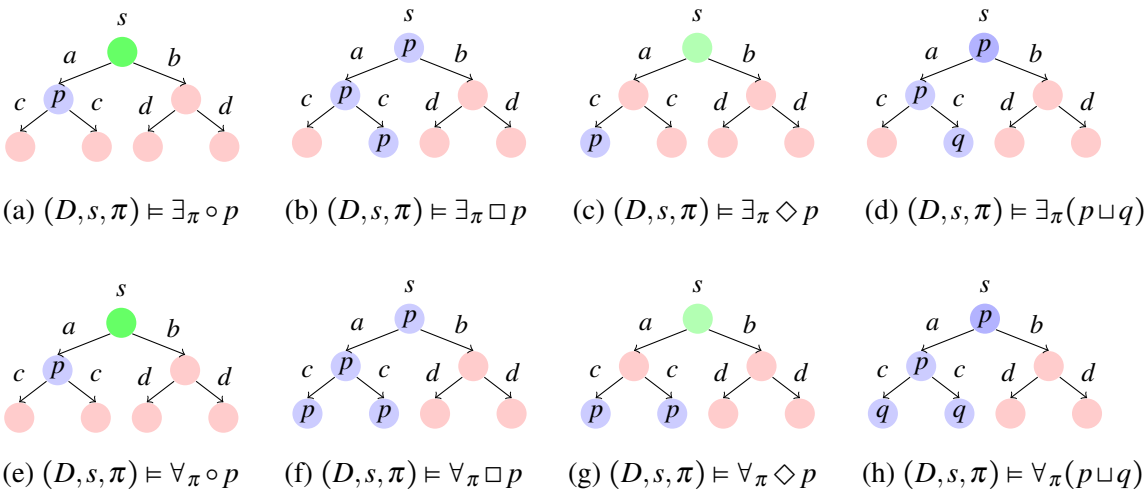
1. $(D, s, \pi) \vDash p$ *se e somente se* p *é verdade em* s
2. $(D, s, \pi) \vDash \neg\varphi$ *se e somente se* $(D, s, \pi) \nvDash \varphi$
3. $(D, s, \pi) \vDash \varphi_1 \wedge \varphi_2$ *se e somente se* $(D, s, \pi) \vDash \varphi_1$ *e* $(D, s, \pi) \vDash \varphi_2$
4. $(D, s, \pi) \vDash \varphi_1 \vee \varphi_2$ *se e somente se* $(D, s, \pi) \vDash \varphi_1$ *ou* $(D, s, \pi) \vDash \varphi_2$
5. $(D, s, \pi) \vDash \exists\beta$ *se e somente se existe um caminho* σ *em* T *começando por* s *tal que* $(D, s, \pi, \sigma) \vDash \beta$
6. $(D, s, \pi) \vDash \forall\beta$ *se e somente se para todos os caminhos* σ *em* T *começando por* s *tem* $(D, s, \pi, \sigma) \vDash \beta$
7. $(D, s, \pi) \vDash \exists_\pi\beta$ *se e somente se existe um caminho* σ *em* T_π *começando por* s *tal que* $(D, s, \pi, \sigma) \vDash \beta$
8. $(D, s, \pi) \vDash \forall_\pi\beta$ *se e somente se para todos os caminhos* σ *em* T_π *começando por* s *tem* $(D, s, \pi, \sigma) \vDash \beta$

A verdade das fórmulas de caminho é definida com respeito ao 4-tupla (D, s, π, σ), onde σ é um caminho $(s, s_1, \ldots)$.

1. $(D, s, \pi, \sigma) \vDash \varphi$ *se e somente se* $(D, s, \pi) \vDash \varphi$
2. $(D, s, \pi, \sigma) \vDash \neg\ \beta$ *se e somente se* $(D, s, \pi, \sigma) \nvDash \beta$
3. $(D, s, \pi, \sigma) \vDash \beta_1 \wedge \beta_2$ *se e somente se* $(D, s, \pi, \sigma) \vDash \beta_1$ *e* $(D, s, \pi, \sigma) \vDash \beta_2$
4. $(D, s, \pi, \sigma)) \vDash \beta_1 \vee \beta_2$ *se e somente se* $(D, s, \pi, \sigma) \vDash \beta_1$ *ou* $(D, s, \pi, \sigma) \vDash \beta_2$
5. $(D, s, \pi, \sigma) \vDash \circ\beta$ *se e somente se* $(D, s_1, \pi, \sigma) \vDash \beta$
6. $(D, s, \pi, \sigma) \vDash \Box\beta$ *se e somente se* $(D, s_k, \pi, \sigma) \vDash \beta$ *para todo* s_k *do caminho;*
7. $(D, s, \pi, \sigma) \vDash \Diamond\beta$ *se e somente se* $(D, s_k, \pi, \sigma) \vDash \beta$ *para algum* s_k *do caminho;*
8. $(D, s, \pi, \sigma) \vDash \beta_1 \sqcup \beta_2$ *se e somente existe um* s_k *no caminho tal que* $(D, s_k, \pi, \sigma) \vDash \beta_2$*, e para todo estado* s_j *anterior a* s_k*, temos que* $(D, s_j, \pi, \sigma) \vDash \beta_1$

Os operadores $\exists$ e $\forall$, por não raciocinarem sobre políticas, terão seus comportamentos iguais a quando usados na lógica CTL*. A Figura 20 ilustra a semântica dos operadores $\exists_\pi$ e $\forall_\pi$ da π-CTL*. Sendo s um estado inicial, pode-se constatar que: em (a) existe um caminho em que p é verdade, para o próximo estado do caminho, na política em consideração; em (b) existe um caminho em que p é verdade em todos os estados, na política em consideração; em (c) existe um caminho em que p é finalmente verdade, na política em consideração; em (d)

Figura 20 – Semântica dos operadores temporais da lógica π-CTL*



(a) $(D,s,\pi) \models \exists_\pi \circ p$ (b) $(D,s,\pi) \models \exists_\pi \Box p$ (c) $(D,s,\pi) \models \exists_\pi \Diamond p$ (d) $(D,s,\pi) \models \exists_\pi (p \sqcup q)$

(e) $(D,s,\pi) \models \forall_\pi \circ p$ (f) $(D,s,\pi) \models \forall_\pi \Box p$ (g) $(D,s,\pi) \models \forall_\pi \Diamond p$ (h) $(D,s,\pi) \models \forall_\pi (p \sqcup q)$

Fonte: Adaptado de (MENEZES, 2014)

existe um caminho em que p é verdade até que q seja verdade, na política em consideração; em (e) para todos os caminhos, p é verdade, para o próximo estado do caminho, na política em consideração; em (f) para todos os caminhos, p é verdade em todos os estados dos caminhos, na política em consideração; em (g) para todos os caminhos, p será finalmente verdade, na política em consideração; e em (h) para todos os caminhos, p será verdade até que q seja verdade, na política em consideração.

2.7 Algoritmos de Planejamento como Verificação de Modelos na Lógica α-CTL

Nesta seção, descreveremos pseudo-códigos dos algoritmos de planejamento como verificação de modelos baseados na lógica α-CTL que realiza o raciocínio sobre ações para retornar o conjunto de estados que satisfaz uma fórmula α-CTL (PEREIRA; BARROS, 2008).

Os algoritmos baseados em verificação de modelos realizam uma busca exaustiva regressiva no espaço de estados. Nestes algoritmos São utilizadas duas funções auxiliares: (i) a função `SAT` Pereira e Barros (2008) que recebe um átomo proposicional p e retorna o conjunto de estados que satisfaz p e; (ii) a função *regressão* Menezes (2014) que recebe um conjunto de estados X e computa um passo da busca regressiva, retornando o conjunto de todos os estados que possuem transições chegando em X. Os algoritmos também consideram que o problema de planejamento $P = \langle D, s_0, \varphi \rangle$ é uma variável global, acessível a todas as funções.

O Algoritmo 1 é o pseudo-código do algoritmo baseado em verificação de modelos para o operador *existe finalmente*. Este algoritmo recebe uma propriedade p do problema, e

retorna o conjunto de estados que satisfazem a fórmula $\exists \diamond p$, ou seja, um conjunto de estados que satisfazem a preferência *existe finalmente p*.

Algoritmo 1: SAT-EF(p)

Entrada: A propriedade p do problema de planejamento

Saida: Conjunto de estados que satisfaz $\exists \diamond p$

```
1 begin
2     X := S ;
3     Y := SAT(p);
4     while X != Y do
5         X := Y ;
6         Y := Y ∪ regressao(Y);
7     end
8     return Y;
9 end
```

O Algoritmo 1 inicia na linha 2, a variável X recebe o conjunto finito S de todos os estados do problema de planejamento. Na linha 3, a variável Y recebe o conjunto de estados que satisfazem a propriedade p, computados pela função `SAT` (PEREIRA; BARROS, 2008). Em cada iteração do laço das linhas 4-7, um passo da busca regressiva é computado (linha 6) até que nenhum novo estado seja alcançado, isto é, quando X for igual a Y. Na linha 8, o algoritmo retorna então o conjunto de todos os estados a partir dos quais finalmente podemos alcançar p, tais estados estão armazenados na variável Y.

O Algoritmo 2 é o pseudo-código do algoritmo baseado em verificação de modelos para o operador *existe globalmente*. Este algoritmo recebe uma propriedade p do problema, e retorna o conjunto de estados que satisfaz a fórmula $\exists \boxdot p$, ou seja, um conjunto de estados que satisfazem a preferência *existe globalmente p*.

O Algoritmo 2 tem o processo de execução bem semelhante ao do Algoritmo 1. A diferença está na linha 6, na qual o SAT-EG calcula a interseção entre dos conjuntos Y e o conjunto de estados retornados pela regressão de Y. Isso é devido ao fato de que no operador globalmente a propriedade p deve ser verdadeira em todos os estados do caminho.

O Algoritmo 3 representa o algoritmo baseado em verificação de modelos para o operador *existe até que*. Este algoritmo recebe a propriedade ψ, que representa a preferência do problema, seja ela *always* ou *sometime*, e retorna o conjunto de estados que satisfaz a fórmula

Algoritmo 2: SAT-EG(p)

Entrada: A propriedade p do problema de planejamento

Saida: Conjunto de estados que satisfaz $\exists \boxdot p$

```
1  begin
2      X := S;
3      Y := SAT(p);
4      while X != Y do
5          X := Y;
6          Y := Y ⋂ regressao(Y);
7      end
8      return Y;
9  end
```

$\exists(\psi \sqcup \varphi)$, sendo φ a meta do problema de planejamento.

Algoritmo 3: SAT-EU(ψ, φ)

Entrada: A fórmula em ψ, representando o conjunto de estados que satisfazem a preferência, e a meta de planejamento φ.

Saida: Conjunto de estados que satisfaz $\exists(\psi \sqcup \varphi)$

```
1  begin
2      W := SAT(ψ);
3      X = S ;
4      Y := SAT(φ);
5      while (X != Y) do
6          X := Y;
7          Y := Y ⋃ (W ⋂ regressao(Y));
8          if s0 ∈ Y then
9              return Y;
10         end
11     end
12     return falha.;
13 end
```

O Algoritmo 3 inicia na linha 2, a variável W recebe o conjunto de estados que satisfazem a fórmula da preferência, ψ, computados pela função SAT (PEREIRA; BARROS,

2008). Na linha 3, a variável X recebe o conjunto de estados do problema de planejamento. Na linha 4, a variável Y recebe o conjunto de estados que satisfazem a meta φ do problema, também computado por `SAT` (PEREIRA; BARROS, 2008). Em cada iteração do laço das linhas 5-11, um passo da busca regressiva é computado (linha 7) até que nenhum novo estado seja alcançado, isto é, quando X for igual a Y. Na linha 9, o algoritmo retorna, então, o conjunto de todos os estados a partir da meta de planejamento, φ, tendo a busca regressiva alcançado o estado inicial, s_0, e satisfazendo a fórmula de preferência para o caminho, ψ. Tais estados estão armazenados na variável Y. Diferentemente dos outros dois algoritmos, o Algoritmo 3 deve retornar o plano para o problema de planejamento, então caso a busca regressiva não consiga alcançar o estado inicial, o algoritmo retorna uma falha, como definido na linha 12.

Os algoritmos apresentados nesta seção são lineares no tamanho do modelo. No entanto, o tamanho do modelo propriamente dito é, com frequência, exponencial no número de átomos proposicionais do domínio de planejamento. A tendência do espaço de estados se tornarem muito grandes é chamada *problema da explosão do espaço de estados* e uma das formas de contornar tal problema é utilizar estruturas de dados eficientes denominadas *Diagramas de Decisão Binários* (BDDs) (HUTH; RYAN, 2008). Os algoritmos que usam tais estrutura de dados são denominados algoritmos de *planejamento como verificação simbólica de modelos*.

Uma inspeção nos algoritmos desta seção mostra que eles consistem na manipulação de conjuntos de estados. Assim, para a implementação da versão simbólica destes algoritmos, tais conjuntos de estados são armazenados usando *BDDs*. Ademais, as operações entre conjuntos tais como *união* e *intersecção* são implementadas em termos de operações lógicas tais como conjunção e disjunção de fórmulas (HUTH; RYAN, 2008). Uma outra importante operação a ser levada em consideração é o raciocínio simbólico sobre ações, definidas com pré-condições e efeitos, implementado por meio da operação de regressão simbólica de ações (MENEZES, 2014).

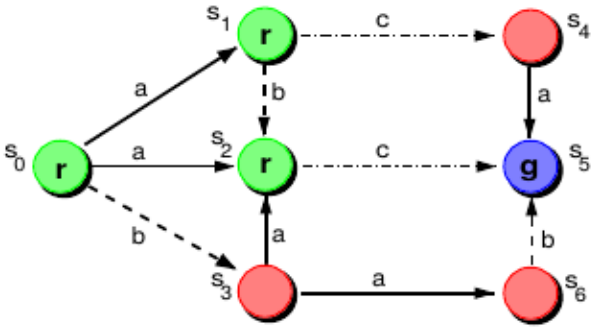
3 TRABALHOS RELACIONADOS

3.1 Expressando metas estendidas com a lógica α-CTL em domínios não-determinísticos

O trabalho de Pereira e Barros (2008) mostrou que, apesar da lógica CTL ser o principal formalismo adotado na área de planejamento como verificação de modelos, a semântica desta lógica não é adequada para tratar problemas com metas de alcançabilidade estendidas.

Exemplo 4. *(Inadequação da Lógica CTL para Problemas com Metas Estendidas) Considere o domínio D Pereira e Barros (2008) ilustrado na Figura 21 e um problema de planejamento com metas estendidas* $P =< D, s_0, (r, g) >$ *em que o agente inicialmente no estado* s_0 *deseja alcançar garantidamente um estado que satisfaça a propriedade g, passando apenas por estados que satisfaçam a propriedade r. Essa meta estendida poderia ser especificada na lógica CTL pela fórmula* $\forall(r \sqcup g)$*, que é verdade em um estado somente se todo caminho que parte desse estado finalmente alcança g, preservando r ao longo de toda a sua extensão. Entretanto, temos que* $(D, s_0) \not\models \forall(r \sqcup g)$*, pois como a lógica CTL não "enxerga" as ações que causam as transições entre estados tal fórmula não é verdade em* s_0 *visto que o estado* s_3 *não satisfaz r. Isso significa que, um planejador baseado em CTL não seria capaz de obter uma solução para um problema no domínio D com a meta* $\forall(r \sqcup g)$*; e, consequentemente, devolveria fracasso.*

Figura 21 – O grafo de transição para o domínio de planejamento *D*



Fonte: (PEREIRA; BARROS, 2008)

Para contornar esta limitação da lógica CTL, o trabalho de (PEREIRA; BARROS, 2008) propôs a lógica α-CTL, que é uma extensão da lógica CTL capaz de "enxergar" as ações por trás das transições entre estados. Pereira e Barros (2008) propõe também algoritmos capazes de obter soluções para problemas de planejamento não-determinísticos com estes tipos de metas.

A capacidade da lógica α-CTL expressar a noção de propriedades de caminho que devem ser preservadas (ou evitadas) nos levou ao questionamento se os algoritmos de planejamento baseados em α-CTL também poderiam ser utilizados para obtenção de soluções para os problemas da área de planejamento com preferências de planos. Além disso, devemos investigar se podemos tratar preferências de planos em domínios com ações não-determinísticas, o que só pode ser feito para domínios com ações determinísticas na maioria dos trabalhos da área de preferências de planos.

O trabalho de Pereira e Barros (2008) propõe as principais fundamentações e algoritmos utilizados neste trabalho: a lógica α-CTL e os algoritmos de verificação de modelos para tratar metas estendidas. As principais diferenças entre o trabalho de Pereira e Barros (2008) e este trabalho é que utilizamos algoritmos de verificação simbólica de modelos baseada em ações, enquanto que os algoritmos propostos em Pereira e Barros (2008) são algoritmos que raciocinam sobre as transições do domínio de planejamento. Ademais, caracterizamos as preferências qualitativas como as metas estendidas apresentadas por (PEREIRA; BARROS, 2008). Neste sentido, este trabalho apresenta mais uma classe de problemas de planejamento que pode ser solucionada com os algoritmos propostos por (PEREIRA; BARROS, 2008).

3.2 Expressando metas estendidas com a lógica $\pi - CTL^*$ em domínios não-determinísticos

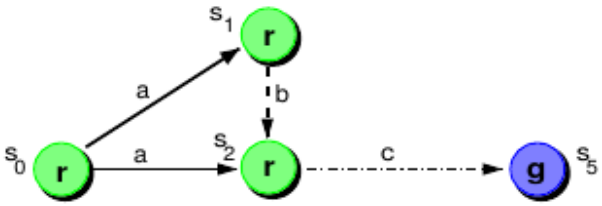
Assim como a lógica CTL, a lógica CTL* também não "enxerga" as ações que rotulam as transições entre os estados, e desta forma, também é inadequada para tratar problemas de planejamento não determinísticos com metas estendidas. Para contornar tal limitação, Baral e Zhao (2004) propôs uma extensão da lógica CTL^*, chamada π-CTL*.

A lógica π-CTL* raciocina sobre uma subestrutura do domínio de planejamento induzida por uma política π. Considere o Exemplo 4, no qual temos o Problema de Planejamento $P =< D, s_0, (r, g) >$, e uma política $\pi = \{(s_0, a), (s_1, b), (s_2, c)\}$ que garantidamente alcança um estado que satisfaz g passando apenas por estados que satisfazem r. A Figura 22 ilustra a subestrutura de D (Figura 21) induzida pela política π.

Os algoritmos baseados em π-CTL* são adequados para *verificar* se um dada subestrutura do domínio de planejamento satisfaz certas propriedades em relação a uma dada política π. No entanto, tais algoritmos não são capazes de *obter* uma política, que é a solução para um problema de planejamento não determinístico.

Desta forma, a semelhança entre este trabalho e o trabalho de Baral e Zhao (2004) é

Figura 22 – Subestrutura do Domínio D induzido por uma política π



Fonte: (PEREIRA; BARROS, 2008)

que ambos são capazes de expressar metas estendidas, propriedades que devem ser alcançadas ou evitadas durante o caminho para alcance da meta. No entanto, o trabalho de Baral e Zhao (2004) não é capaz de obter tais soluções, como podemos fazer com os algoritmos propostos neste trabalho.

3.3 SGPlan: Planejador para Preferência e Restrições

O trabalho Hsu *et al.* (2006) descreve o SGPlan5, planejador que foi o primeiro colocado, na trilha de planejamento com preferências qualitativas da 5ª edição da Competição Internacional de Planejamento (IPC). Esse planejador particiona um grande problema de planejamento em subproblemas, cada um com sua própria submeta. O particionamento em submetas é eficaz para solucionar grandes problemas de planejamento, porque cada subproblema particionado envolve um espaço de busca substancialmente menor que o do problema original. Os autores do SGPlan desenvolveram métodos para representar um problema de planejamento de uma forma multi-valorada e executar o particionamento no espaço transformado. A representação multi-valorada leva a heurísticas mais eficazes para resolver preferências de meta e caminho e restrições temporais. O planejador é baseado em algoritmos de otimização combinatória e tenta minimizar ou maximizar as restrições do problema correspondente a cada preferência qualitativa. O planejador é adequado para tratar problemas de planejamento com ações determinísticas.

O planejador SGPlan5 utiliza métodos baseados em otimização combinatória na tentativa de elaborar planos que maximizem a quantidade de preferências qualitativas satisfeitas enquanto que os algoritmos utilizados neste trabalhos utilizam-se de métodos formais para obtenção do plano com preferências. Outra diferença fundamental é que o SGPlan5 é capaz de raciocinar apenas sobre domínios com ações determinísticas enquanto que os algoritmos utilizados neste trabalho podem ser estendidos para obtenção de preferências qualitativas em domínios com ações não-determinísticas.

3.4 HPlan-P: Planejador baseado em Busca Heurística para Preferências Estendidas

O trabalho Baier *et al.* (2006) descreve o HPlan-P, planejador que foi o segundo colocado na trilha de planejamento com preferências qualitativas da 5ª edição da Competição Internacional de Planejamento (IPC). Este planejador é capaz de obter planos para problemas com preferências temporais estendidas, compilando as preferências em autômatos de estados finitos não-determinísticos cujas condições de aceitação denotam o alcance da preferência descrita pelo autômato. Os autômatos são representados no problema de planejamento por meio de predicados e ações adicionais. Com essas preferências compiladas, o planejador utiliza heurísticas para guiá-lo em direção a planos que satisfazem as preferências.

As semelhanças entre o planejador HPlan-P e o proposto neste trabalho é que ambos são baseado em métodos formais: o primeiro raciocina sobre autômatos finitos não-determinísticos enquanto o último raciocina sobre estruturas de Kripke representando o domínio de planejamento. A principal diferença é que o HPlan-P utiliza-se de busca heurística para encontrar planos e os algoritmos utilizados neste trabalho realizam busca regressiva simbólica e exaustiva.

3.5 MIPS-BDD: Planejamento Simbólico Ótimo

O trabalho Edelkamp (2006) descreve o MPIS-BDD, planejador que foi 3º colocado, na área dos domínios com preferências qualitativas, da 5ª edição da Competição Internacional de Planejamento (IPC). O planejador compila as expressões de caminho e de estado na lógica LTL . O planejador realiza uma busca em largura de custo ótimo para obtenção de planos satisfazendo as restrições de preferências.

O sistema desenvolvido por (EDELKAMP, 2006) se assemelha ao proposto neste trabalho por utilizar verificação simbólica de modelos para solucionar problemas com preferências qualitativas. No entanto, O MIPS é baseado na lógica LTL, que não possui ações em sua semântica, e o planejador proposto neste trabalho é baseado em α-CTL cuja semântica é baseada em ações. Além disso, o MIPS realiza o raciocínio simbólico sobre as *transições* do domínio de planejamento e não diretamente sobre as ações com pré-condições e efeitos, como fazemos neste trabalho.

3.6 Comparação entre os trabalhos relacionados e o proposto

A Tabela 1, apresentada a seguir, explica resumidamente a relação entre os trabalhos relacionados e o que está sendo proposto neste trabalho.

Tabela 1 – Comparação entre os trabalhos relacionados e o proposto

Abordagem	Tipo de ações que os algoritmos conseguem raciocinar.	Obtém soluções para problemas com preferência de planos qualitativas?
α-CTL (PEREIRA; BARROS, 2008)	Não-determinísticas	Não
π-CTL* (BARAL; ZHAO, 2004)	Não-determinísticas	Não
SGPlan (HSU *et al.*, 2006)	Determinísticas	Sim
HPPlan-P (BAIER *et al.*, 2006)	Não-determinísticas	Sim
MIPS-BDD (EDELKAMP, 2006)	Determinísticas	Sim
Este trabalho	Não-Determinísticas	Sim

Fonte: Elaborado pelo autor.

4 ESPECIFICAÇÃO DE PREFERÊNCIAS DE PLANOS USANDO METAS ESTENDIDAS NA LÓGICA α-CTL

Este trabalho tem como objetivo verificar se o arcabouço de planejamento como verificação de modelos na lógica α-CTL pode ser utilizado para obtenção de planos para problema de *Planejamento com Preferências de Planos*. Estas áreas, apesar de distintas, são capazes de expressar a noção de propriedades que devem ser preservadas ou mantidas durante o caminho para alcance da meta.

Na Seção 2, vimos que as Lógicas π-CTL* e α-CTL são lógicas temporais com semântica baseada em ações e, por isto, são mais adequadas para verificar propriedades em um domínio de planejamento. No entanto, a lógica π-CTL* somente é adequada para realizar a verificação se uma solução (política) satisfaz ou não uma dada propriedade. Como tal lógica necessita utiliza a política em sua semântica, não é possível utilizá-la para *obter* tal política, Assim, de todos os formalismos apresentados na fundamentação, a Lógica α-CTL é a que possui a capacidade de expressar a noção de propriedades de caminho que devem ser preservadas (ou evitadas) e também que esta lógica pode ser utilizadas para a obtenção das soluções, isto é, da subestrutura do domínio que satisfaz uma dada propriedade.

Devido a isso, escolhemos e propomos o uso da lógica α-CTL para especificar problemas de planejamento com preferências de planos, também conhecida como preferências qualitativas. Como já vimos, um problema de planejamento com preferências de planos pode ter quatro tipos de operadores: `always`, `sometime`, `sometime-before` e `at-most-once`. Neste trabalho, iremos mostrar como expressar os operadores `always` e `sometime` usando fórmulas na lógica $\alpha - CTL$.

4.1 Especificação da preferência sometime em α-CTL

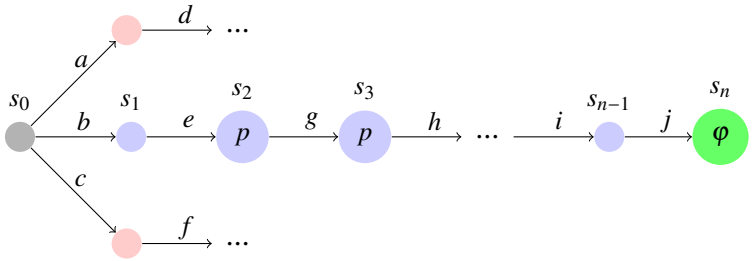
A preferência (sometime (p)) expressa que devemos encontrar um plano em que a propriedade p seja verdadeira em algum estado no caminho para se alcançar a meta φ. Usando a Definição 2.6.11 (Semântica da Lógica α-CTL), podemos perceber que a sentença "em algum momento p é verdade" pode ser modelada com o operador *finalmente* ($\diamond$), resultando na fórmula $\exists \diamond p$ (existe um caminho no qual *finalmente* p é verdade). No entanto, para especificar a preferência desejada, esta fórmula deve ser verdadeira *durante* o caminho de alcance da meta φ. Desta forma, usaremos também o operador *até que* ($\sqcup$) para expressar esta noção sobre o caminho. A Definição 4.1.1 mostra como expressamos a propriedade de preferência de planos

`sometime` em α-CTL.

Definição 4.1.1. *(Especificação da Preferência Sometime) Dado um problema de planejamento $P = \langle D, s_0, \varphi \rangle$ sobre um conjunto de proposições $\mathbb{P}$ e um conjunto de ações $\mathbb{A}$, a preferência de planos (`sometime` (p)), em que $p \in \mathbb{P}$, pode ser escrita como a meta estendida em α-CTL:*

$$\exists((\exists \diamond p) \sqcup \varphi)$$ □

Figura 23 – Problema de planejamento com preferências de planos que satisfaz o operador (`sometime` (p))



Fonte: Elaborado pelo autor

A Figura 23 demonstra uma subestrutura de planejamento que satisfaz a preferência (`sometime` (p)). Nela existe um caminho, $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$, no qual p é *finalmente* verdade, ao mesmo tempo que este caminho leva ao alcance da meta. Adotando s_0 como o estado inicial, no caminho anteriormente citado, a propriedade p é verdade no estado s_2, tornando a fórmula $\exists \diamond p$ verdadeira. Como essa propriedade é verdadeira antes da meta ser alcançada, então podemos concluir que existe um caminho no qual $\exists \diamond p$ é verdade *até que* a meta seja verdade, logo $\exists(\exists \diamond p \sqcup \varphi)$ é satisfeita.

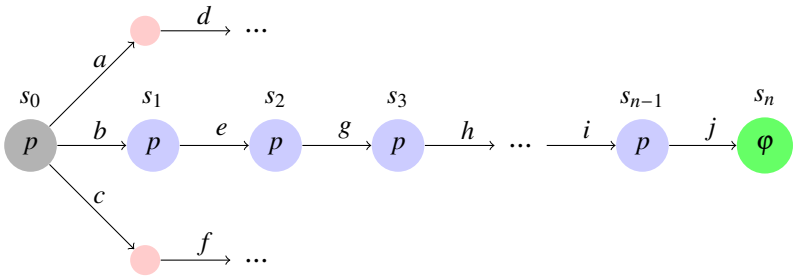
4.2 Especificação da preferência always em α-CTL

A preferência `always` (p) expressa que devemos encontrar um plano em que em a propriedade p seja verdadeira em todos os estados no caminho para se alcançar a meta φ. Usando a Definição 2.6.11 (Semântica da Lógica $\alpha - CTL$), podemos perceber que a sentença "em todos os estados p é verdade" pode ser modelada com o operador *globalmente* ($\boxdot$), resultando na fórmula $\exists \boxdot p$ (existe um caminho no qual *globalmente* p é verdade). No entanto, para especificar a preferência desejada, esta fórmula deve ser verdadeira durante o caminho de alcance da meta φ. Desta forma, usaremos também o operador *até que* ($\sqcup$) para expressar esta noção sobre o caminho. A Definição 4.2.1 mostra como expressamos a propriedade de preferência de planos `always` em α-CTL.

Definição 4.2.1. *(Especificação da Preferência Always) Dado um problema de planejamento* $P = \langle D, s_0, \varphi \rangle$ *sobre um conjunto de proposições* $\mathbb{P}$ *e um conjunto de ações* $\mathbb{A}$*, a preferência de planos (*`always` (p)*), em que* $p \in \mathbb{P}$*, pode ser escrita como a meta estendida em* α*-CTL:*

$$\exists((\exists \boxdot p) \sqcup \varphi)$$ □

Figura 24 – Problema de planejamento com preferências de planos que satisfaz o operador `always` (p)



Fonte: Elaborado pelo autor

A Figura 24 demonstra uma subestrutura de planejamento que satisfaz a preferência `(always (`p`))`. Nela existe um caminho, $\{s_0, s_1, s_2, s_3, \cdots, s_{n-1}, s_n\}$, no qual p é *globalmente* verdade, ao mesmo tempo que este caminho leva ao alcance da meta. Adotando s_0 como o estado inicial, no caminho anteriormente citado, a propriedade p é verdade em todos os estados do caminho anteriores a meta, tornando a fórmula $\exists \boxdot p$ verdadeira. Como essa propriedade é verdadeira antes da meta ser alcançada, então podemos concluir que existe um caminho no qual $\exists \boxdot p$ é verdade *até que* a meta seja verdade, logo $\exists(\exists \boxdot p \sqcup \varphi)$ é satisfeita.

Por enquanto, trabalharemos apenas com a especificação de preferências `always` e `sometimes`. Concluímos que a preferência `(sometime-before (`p`), (`q`))` pode ser especificada com a fórmula $\exists(\exists(q \sqcup p) \sqcup \varphi)$. Já a preferência `(at-most-once (`$p$`))` acreditamos que a lógica α-CTL não é expressiva o suficiente para especificar a noção de ”caso a propriedade seja verdade em algum estado do caminho, ela não pode ser verdade em nenhum outro até alcançar a meta”, pois a propriedade p deve ser verdadeira no máximo uma vez durante o caminho, assim como caminhos que levam a meta, mas que a propriedade não é verdadeira em nenhum dos estados.

5 IMPLEMENTAÇÃO E ANÁLISE EXPERIMENTAL

Nesta seção, apresentamos informações sobre a implementação dos algoritmos de planejamento baseado em verificação simbólica de modelos na lógica α-CTL na tentativa de solucionar problemas de planejamento com preferências de planos, bem como os resultados da análise experimental realização.

5.1 Implementação

Os algoritmos de planejamento como verificação simbólica de modelos foram implementados[1] utilizando a linguagem `Java`, na IDE Eclipse. Utilizamos também a biblioteca `JavaBdd` para representação e cálculo simbólico do conjunto de estados na busca regressiva.

A implementação foi feita diretamente a partir dos pseudocódigos dos algoritmos de planejamento baseados em verificação simbólica de modelos na lógica α-CTL e teve por objetivo mostrar que este arcabouço pode ser utilizado diretamente para solucionar uma nova classe de problemas de planejamento: *planejamento com preferências qualitativas*.

5.2 Análise Experimental

Nesta seção apresentamos em detalhes o domínio utilizado na análise experimental, o domínio do robô de marte, e apresentamos os resultados obtidos com a execução dos algoritmos implementados.

5.2.1 Domínio Rovers

O domínio *Rovers* aborda as tarefas dos robôs exploradores de Marte. Em resumo, o robô possui pontos específicos de Marte para explorar, e equipado de uma câmera e coletores de amostra de solo e rocha, ele precisa tirar fotos dos pontos e/ou coletar as amostras requeridas. Após as coletas, o robô deve enviar esses os dados para a estação espacial.

A versão deste domínio envolvendo preferências qualitativas foi dada como entrada para os planejadores que competiram da 5º edição da Competição Internacional de Planejamento (IPC). O domínio é composto por vinte e um arquivos em formato *PDDL*. O primeiro deles é o que define o domínio de planejamento, como definido na Figura 25. Os outros vinte arquivos são

[1] Disponível em: https://github.com/RodMdS/search-backwards-in-preferences-plan

diferentes problemas de planejamento que trabalham com preferência qualitativas, a Figura 9 mostra um exemplo de como é definido um desses arquivos.

Figura 25 – Exemplo de arquivo *PDDL* que descreve o domínio de planejamento rovers

```
(define (domain Rover)
    (:requirements :typing :constraints :preferences)
    (:types rover waypoint store camera mode lander objective)
    (:predicates (at ?x - rover ?y - waypoint)
        (at_lander ?x - lander ?y - waypoint)
        // outros predicados são definidos a seguir
        ...
    )
    (:action navigate
        :parameters (?x - rover ?y - waypoint ?z - waypoint)
        :precondition (and (can_traverse ?x ?y ?z)
                (available ?x)
                (at ?x ?y)
                (visible ?y ?z))
        :effect (and (not (at ?x ?y))
                (at ?x ?z))
    )
    // outras ações são definidas a seguir
    ...
)
```

Fonte: Elaborado pelo autor

Na segunda linha da Figura 25, está escrito a expressão `:preferences` que define que este domínio será utilizado para problema de planejamento com preferências qualitativas. Na terceira linha é definidos os tipos que as variáveis do domínio poderão ter, utilizando-se a palavra reservada `(:types)`. Em seguida, são definidos os `(:predicates)` (propriedades), que são as propriedades referentes ao mundo: localização do robô, localização do veículo espacial, etc. Dessa forma, o planejador utiliza os parâmetros para criar os vários átomos proposicionais (instanciados) referente a cada predicado. Por último, são definidas as ações, por meio da utilização da palavra reservada `(:action)`, contendo: o nome da ação, a expressão `:parameters` que define os parâmetros de entrada da ação; a expressão (:precondition), que define uma conjunção das pré-condições da ações; a expressão `(:effect)`, que define uma conjunção das propriedades que são os efeitos dessa ação.

A princípio, um arquivo de domínio é o mesmo utilizado em planejamento clássico. A diferença nas entradas para problemas de planejamento com preferências está na definição dos

Figura 26 – Exemplo de arquivo *PDDL* que descreve as preferências qualitativas de um problema de planejamento

```
(define (problem roverprob1) (:domain rover)
    (:objects ...  )
    (:init ...  )
    (:goal (and ...  ))
    (:constraints
        (and (preference a0 (always (at_soil_sample waypoint0)))
             (preference a1 (always (at_rock_sample waypoint1)))
             (preference e0 (sometime (at rover0 waypoint0)))
             (preference e1 (sometime (have_soil_analysis rover0 waypoint0)))
             (preference e2 (sometime (have_rock_analysis rover0 waypoint1)))
             (preference o0 (at-most-once (at rover0 waypoint3)))
             (preference o1 (at-most-once (at rover0 waypoint1)))
             (preference sb3
                  (sometime-before (at rover0 waypoint2) (have_soil_analysis rover0 way-
point0)))
             (preference sb7
                  (sometime-before (have_soil_analysis rover0 waypoint2) (full rover0store)))
...
    (:metric minimize
            (* (is-violated sb7) 4.28133)
            (* (is-violated sb3) 9.22133)
            (* (is-violated o1) 5.434)
            (* (is-violated o0) 9.804)
            (* (is-violated e2) 10.0447)
            (* (is-violated e1) 10.2093)
            (* (is-violated e1) 10.2093)
            (* (is-violated e0) 12.35)
            (* (is-violated a1) 8.208)
            (* (is-violated a0) 16.53)
    )))
```

Fonte: Elaborado pelo autor

arquivos do problema.

A Figura 26 demonstra como é definido um arquivo PDDL que descreve o problema de planejamento com preferências qualitativas. O arquivo que descreve o problema de planejamento *rover-01* possui 19 definições de preferências, sendo elas 2 do tipo `always` e 3 do tipo `sometime`. Além disso, cada preferência possui uma função que associa um valor a ela sobre o quão desinteressante é que essa preferência seja violada. O objetivo dos algoritmos competidores é minimizar essa função de violação de preferências.

5.2.2 Resultados dos Experimentos

Os experimentos foram realizados em um notebook Acer Aspire A515-51 com 8GB de memória RAM, 1TB de HD, Processador Intel i5-7200U com 2.50GHz e sistema operacional Ubuntu 18.04.3 64 bits.

O objetivo dos experimentos é verificar se o arcabouço de *Planejamento com Verificação Simbólica de Modelos* com a lógica α-CTL pode ser utilizado para encontrar planos em problemas de planejamento com preferências qualitativas. Devido a isso, foi o escolhido o problema rovers-01 do domínio do `rovers` para que estes testes fossem realizados.

A partir do arquivo de entrada rovers-01, geramos 3 arquivos de problemas cada um contendo uma preferência `sometime` e 2 arquivo de problema contendo uma preferência `always`. Todas as entradas geradas possuíam o mesmo conjunto de propriedades, estado inicial e a meta de planejamento. O que os diferenciavam, eram as preferências que deveriam ser satisfeitas no decorrer do plano.

A Tabela 2 exibe o tempo de execução, para cada um dos cinco problemas gerados, que os algoritmos utilizados neste trabalho utilizaram para verificar e retornar a subestrutura do domínio de planejamento contendo um plano com preferências qualitativas. Para todos os problemas, todos os planos encontrados possuem tamanho 8. Este é o mesmo tamanho de plano produzido pelo algoritmo vencedor (HSU *et al.*, 2006) da competição para este tipo de problema.

Tabela 2 – Tempo de execução para os cinco problemas gerados no domínio do robô de marte.

Problema	Preferência	Tempo de execução (ms)
rover-01-a0	ALWAYS-AT_SOIL_SAMPLE-WAYPOINT0	157
rover-01-a1	ALWAYS-AT_ROCK_SAMPLE-WAYPOINT1	107
rover-01-e0	SOMETIME-AT-ROVER0-WAYPOINT0	105
rover-01-e1	SOMETIME-HAVE_SOIL_ANALYSIS-ROVER0-WAYPOINT0	62
rover-01-e2	SOMETIME-HAVE_ROCK_ANALYSIS-ROVER0-WAYPOINT1	125

Fonte: Elaborado pelo autor.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho mostrou que o arcabouço de planejamento com verificação simbólica de modelos baseado na lógica α-CTL pode ser utilizado para solucionar problemas de planejamento com preferências de planos. Estas áreas, apesar de distintas, são capazes de expressar a noção de propriedades que devem ser preservadas ou mantidas durante o caminho para alcance da meta. Os algoritmos baseados em α-CTL foram os escolhidos para obtenção de soluções para o problema de planejamento com preferências qualitativas por, além de ser um formalismo apropriado para expressar propriedades que devem ser mantidas ou evitadas ao longo de um caminho, também é apropriada para obter tais soluções, o que não é possível realizar com a lógica π-CTL*.

Utilizamos, então, a lógica α-CTL para expressar as preferências qualitativas utilizadas nos domínios de planejamento com preferências. Esses domínios definem quatro tipos de operadores para definição de uma preferência, *always*, *sometime*, *at-most-once* e *sometime-before*. Porém, neste trabalho, nos propusemos a especificar apenas as preferências *always* e *sometime*.

Para realizar os experimentos do nosso trabalho, utilizamos os domínios disponibilizados pela 5º edição da IPC. Apesar desses domínios apresentarem muitas preferências a serem atendidas, estas não foram utilizadas em conjunto pois em muitas vezes resultavam em problemas sem solução. Os algoritmos da literatura de planejamento com preferências tratam este tipo de problema como problemas de otimização combinatória, minimizando a quantidade de preferências que deixam de ser atendidas. No entanto, o intuito deste trabalho não visava atender todas as preferências mas sim mostrar que e como cada tipo de preferência de caminho é correspondente a uma meta estendida em α-CTL. E, além disso, que os algoritmos de planejamento como verificação simbólica de modelos podem ser utilizados para solucionar este tipo de problema.

Um ponto positivo para os algoritmos utilizados neste trabalho é que eles são, também, adequados para obter soluções para problemas de planejamento em domínios com ações não-determinísticas. A maioria dos algoritmos da literatura de planejamento com preferências não podem ser facilmente estendidos para solucionar problemas com este tipo de ações.

Para os trabalhos futuros, ficam a necessidade de pensar em uma alternativa para expressar os operadores `at-most-once` e `sometime-before` em α-CTL e encontrar uma forma de utilizar as diversas preferências, definidas nos problemas de planejamento, em conjunto, para

que assim o planejador possa escolher quais preferências seguir ou não baseados na definição do problema. Além disso, mais testes da abordagem proposta neste trabalho precisam ser realizadas, como testes em outros problemas do domínio *rovers* e nos outros diversos domínios disponibilizados pela competição.

REFERÊNCIAS

BAIER, J.; HUSSELL, J.; BACCHUS, F.; MCILRAITH, S. Planning with temporally extended preferences by heuristic search. **ICAPS**, p. 20, 2006.

BAIER, J. A.; MCILRAITH, S. A. Planning with preferences. **AI Magazine**, American Association for Artificial Intelligence, v. 29, n. 4, p. 25–37, 2008.

BARAL, C.; ZHAO, J. Goal specification in presence of non-deterministic actions. In: **ECAI**. [S.l.: s.n.], 2004. v. 16, p. 273.

BARAL, C.; ZHAO, J. Goal specification, nondeterminism and quantifying over pçolicies. In: MENLO PARK, CA; CAMBRIDGE, MA; LONDON; AAAI PRESS; MIT PRESS; 1999. **Proceedings of the National Conference on Artificial Intelligence**. [S.l.], 2006. v. 21, n. 1, p. 231.

BENTON, J.; BRIEL, M. van den; KAMBHAMPATI, S. Finding admissible bounds for over-subscription planning problems. In: **Proceedings of ICAPS-07 Workshop on Heuristics for Domain independent Planning**. [S.l.: s.n.], 2007.

BERTOLI, P.; CIMATTI, A.; ROVERI, M.; TRAVERSO, P. Planning in nondeterministic domains under partial observability via symbolic model checking. In: **IJCAI**. [S.l.: s.n.], 2001. v. 2001, p. 473–478.

BOUTILIER, C.; BRAFMAN, R. I.; DOMSHLAK, C.; HOOS, H. H.; POOLE, D. Cp-nets: A tool for representing and reasoning withconditional ceteris paribus preference statements. **Journal of artificial intelligence research**, [S.l], v. 21, p. 135–191, 2004.

BRAFMAN, R. I.; DOMSHLAK, C.; SHIMONY, S. E. On graphical modeling of preference and importance. **Journal of Artificial Intelligence Research**, [S.l], v. 25, p. 389–424, 2006.

BRIEL, M. V. D.; SANCHEZ, R.; DO, M. B.; KAMBHAMPATI, S. Effective approaches for partial satisfaction planning. In: **AAAI**. [S.l.: s.n.], 2004. p. 562–569.

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

CIMATTI, A.; ROVERI, M.; TRAVERSO, P. Strong planning in nondeterministic domains via model checking. In: **AIPS**. [S.l.: s.n.], 1998. v. 98, p. 36–43.

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

EDELKAMP, S. Optimal symbolic pddl3 planning with mips-bdd. **5th International Planning Competition Booklet (IPC-2006)**, [S.l], p. 31–33, 2006.

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.

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

GIUNCHIGLIA, F.; TRAVERSO, P. Planning as model checking. In: SPRINGER. **European Conference on Planning**. [S.l.], 1999. p. 1–20.

HSU, C.-W.; WAH, B. W.; HUANG, R.; CHEN, Y. New features in sgplan for handling preferences and constraints in pddl3. 0. In: CITESEER. **Proceedings of the Fifth International Planning Competition**. [S.l.], 2006. p. 39–42.

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.

IROBOT. **Roomba 966 Vacuuming Robot**. 2019. Disponível em: https://bit.ly/2Hz8CfW. Acesso em: 20.nov.2019.

KRIPKE, S. A. Semantical considerations on modal logic. [S.l.: s.n.], 1963.

KRULWICH, B. Planning for soft goals. In: ELSEVIER. **Artificial Intelligence Planning Systems**. [S.l.], 1992. p. 289–290.

KUSHMERICK, N.; HANKS, S.; WELD, D. S. An algorithm for probabilistic planning. **Artificial Intelligence**, Elsevier, v. 76, n. 1-2, p. 239–286, 1995.

MENEZES, M. V. **Mudanças em problemas de planejamento sem solução**. Tese (Doutorado) — Instituto de Matemática e Estatística. Universidade de São Paulo, São Paulo, 2014.

MENEZES, M. V.; BARROS, L. N.; PEREIRA, S. L. Symbolic regression for nondeterministic actions. **Learning and Nonlinear Models**, [S.l], v. 12, p. 98–114, 2014.

MÜLLER-OLM, M.; SCHMIDT, D.; STEFFEN, B. Model-checking: a tutorial introduction. In: **Proceedings of the 6th International Symposium on Static Analysis, SAS'99**. London, UK: Springer, 1999.

NIGENDA, R. S.; KAMBHAMPATI, S. Planning graph heuristics for selecting objectives in over-subscription planning problems. In: **ICAPS**. [S.l.: s.n.], 2005. p. 192–201.

PEREIRA, S. do L. **Planejamento não-determinístico baseado em Verificação de Modelos**. Tese (Doutorado) — Instituto de Matemática e Estatística. Universidade de São Paulo, São Paulo, 2007.

PEREIRA, S. do L.; BARROS, L. N. de. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. Tese (Doutorado) — Instituto de Matemática e Estatística. 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**, v. 16, n. 3, p. 327–344, Jun 2008. ISSN 1573-7454. Disponível em: https://doi.org/10.1007/s10458-008-9034-0. Acesso em: 20.mar.2019.

PNUELI, A. The temporal logic of programs. In: IEEE. **18th Annual Symposium on Foundations of Computer Science (sfcs 1977)**. [S.l.], 1977. p. 46–57.

RUSSEL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. 3rd edt. ed. [S.l.: s.n.], 2016.

SANTOS, R. M.; MENEZES, V. Utilizando preferências para mudança da meta de planejamento. **Resumos Estendidos Encontros Universitários da UFC 2017**, Campus da UFC em Quixadá, v. 7, p. 24–28, 2017.

SANTOS, V. B. **Planejamento baseado em verificação simbólica de modelos**. Dissertação (Mestrado) — Instituto de Matemática e Estatística. Universidade de São Paulo, São Paulo, 2018.

SMITH, D. E. Choosing objectives in over-subscription planning. In: **ICAPS**. [S.l.: s.n.], 2004. v. 4, p. 393.

SOHRABI, S.; BAIER, J. A.; MCILRAITH, S. A. Htn planning with preferences. In: **Twenty-First International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2009.

SON, T. C.; PONTELLI, E.; BARAL, C. A non-monotonic goal specification language for planning with preferences. In: **Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation**. [S.l.]: Springer, 2015. p. 202–217.

WELD, D. S. An introduction to least commitment planning. **AI magazine**, [S.l], v. 15, n. 4, p. 27–27, 1994.

ZILLER, R.; SCHNEIDER, K. Combining supervisor synthesis and model checking. **ACM Transactions on Embedded Computing Systems (TECS)**, ACM, v. 4, n. 2, p. 331–362, 2005.