UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

LUÍS FERNANDO OLIVEIRA SOUSA

OBTENÇÃO DE DADOS ESTRUTURADOS PARA AQUISIÇÃO AUTOMÁTICA DO DOMÍNIO DE PLANEJAMENTO GERENCIADOR DO AGENTE DE DIÁLOGO PLANTÃO CORONAVÍRUS

QUIXADÁ

2024

LUÍS FERNANDO OLIVEIRA SOUSA

OBTENÇÃO DE DADOS ESTRUTURADOS PARA AQUISIÇÃO AUTOMÁTICA DO DOMÍNIO DE PLANEJAMENTO GERENCIADOR DO AGENTE DE DIÁLOGO PLANTÃO CORONAVÍRUS

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

Orientadora: Profª. Drª. Maria Viviane de Menezes.

QUIXADÁ

2024

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

S697o Sousa, Luís Fernando Oliveira.
Obtenção de dados estruturados para aquisição automática do domínio de planejamento gerenciador do agente de diálogo plantão coronavírus / Luís Fernando Oliveira Sousa. – 2024.
47 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2024.
Orientação: Profa. Dra. Maria Viviane de Menezes.

1. Planejamento automatizado. 2. Aquisição de domínios. 3. Agentes de Diálogo. I. Título.
CDD 005

LUÍS FERNANDO OLIVEIRA SOUSA

OBTENÇÃO DE DADOS ESTRUTURADOS PARA AQUISIÇÃO AUTOMÁTICA DO DOMÍNIO DE PLANEJAMENTO GERENCIADOR DO AGENTE DE DIÁLOGO PLANTÃO CORONAVÍRUS

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

Aprovada em: __/__/____

BANCA EXAMINADORA

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

Profª. Drª. Lívia Almada Cruz
Universidade Federal do Ceará (UFC)

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 Deus por me resguardar, acolher e conduzir intensamente nesse período, ajudando a superar todos os obstáculos.

À minha mãe, Regina, por estar sempre ao meu lado. Seu amor incondicional e exemplo me tornam uma pessoa melhor. O seu esforço para me proporcionar uma educação de qualidade desde cedo foi fundamental para a minha trajetória.

Ao meu pai, Adriano, agradeço profundamente por todo o apoio e incentivo ao longo da minha vida. Nos momentos mais desafiadores, sua presença me trouxe força e tranquilidade.

Ao meu irmão, André, pelo apoio tanto na área acadêmica quanto na vida como um todo. Por ser meu companheiro nas brincadeiras e alegrias da vida, assim como nos momentos sérios e mais desafiadores.

À minha orientadora, Viviane Menezes, pela excelente orientação, incentivo, paciência e pelo vasto conhecimento compartilhado.

Aos professores da banca examinadora, Lívia Almada e Davi Romero, pelo tempo dedicado, pelas valiosas colaborações e sugestões.

Aos professores da UFC Quixadá, por todo o ensinamento, orientação e preparo que me proporcionaram para a vida profissional, deixo meus sinceros agradecimentos.

Aos colegas da turma de graduação, pelas reflexões, críticas e sugestões recebidas ao longo do curso.

E todos que de alguma forma diretamente ou indiretamente contribuíram para minha formação pessoal e profissional.

"Seja a mudança que você quer ver no mundo."

(Mahatma Ganhdi)

RESUMO

Planejamento Automatizado é a subárea da Inteligência Artificial (IA) que estuda o processo deliberativo de escolha de ações para que um agente inteligente alcance suas metas. Um planejador é um algoritmo do tipo *resolvedor geral de problemas*, que recebe como entrada uma descrição em alto nível do agente e do ambiente (domínio de planejamento) e fornece como saída uma sequência de ações (plano) que leva o agente de um estado inicial para um estado que satisfaça suas metas. A modelagem de um domínio de planejamento pode incluir o fato de que as ações tem efeitos incertos, efeitos não-determinísticos. Nestes casos, a solução para um problema de planejamento com ações não-determinística é denominada política: um mapeamento que revela para cada possível estado que o agente alcançar qual a ação que deve ser tomada. A aquisição de um domínio de planejamento requer conhecimentos sobre linguagens de ações e sobre o ambiente de aplicação, necessitando de colaboração entre especialistas em planejamento e no domínio de aplicação. Este processo pode ser realizado manualmente ou de forma automática, com algoritmos que aprendem a descrição do domínio a partir de dados, conhecidos como rastros de planos (*plan traces*). Este trabalho visa construir uma base de dados de *plan traces* (dados estruturados) a partir da política gerenciadora do agente de diálogo Plantão Coronavírus, desenvolvido pelo Governo do Estado do Ceará para auxiliar na comunicação entre cidadãos e profissionais de saúde durante a pandemia de COVID-19. A base de dados servirá para, no futuro, ser possível realizar a aquisição automática deste domínio de planejamento.

Palavras-chave: planejamento automatizado; aquisição de domínios; agentes de diálogo.

ABSTRACT

Automated Planning is a subfield of Artificial Intelligence (AI) that focuses on the deliberative process of selecting actions for an intelligent agent to achieve its goals. A planner is an algorithm capable of solving general problems, taking as input a high-level description of the agent and the environment (planning domain) and providing as output a sequence of actions (plan) that leads the agent from an initial state to a state that satisfies its goals. Acquiring a planning domain is a complex task, as it requires knowledge of action languages and the application environment, necessitating collaboration between planning specialists and domain experts. This process can be performed manually or automatically, using algorithms that learn the domain description from data, known as plan traces. This work aims to build a database of plan traces (structured data) from the policy generated by the PRP Planner, avoiding the need to extract data from natural language information. The policy used in this work was extracted through dialogues from the chatbot named Plantão Coronavírus, developed to assist the Government of the State of Ceará in communication between citizens and health professionals during the COVID-19 pandemic. By analyzing the policy generated by the PRP Planner, it was possible to more accurately obtain the start and end states for each action and to construct a solid database of plan traces, contributing to advances in the automatic acquisition of planning domains.

Keywords: automated planning; domain acquisition; dialogue agents.

LISTA DE FIGURAS

SUMÁRIO

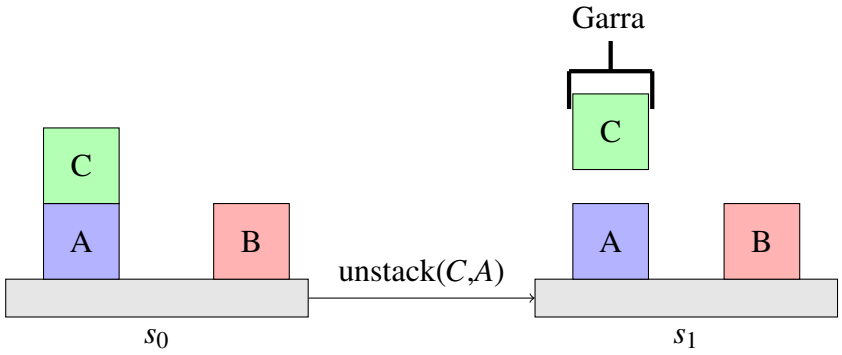
1 INTRODUÇÃO

Planejamento é um processo que seleciona e organiza ações antecipando seus resultados esperados, visando atingir os objetivos propostos da melhor forma possível. *Planejamento Automatizado* é a subárea da Inteligência Artificial (IA) que estuda computacionalmente esse processo (Ghallab *et al.*, 2004).

Um domínio de planejamento é uma descrição da dinâmica do ambiente e do agente. Ele pode ser descrito formalmente por meio de um sistema de transição de estados em que os estados são rotulados por átomos proposicionais, descrevendo as propriedades do agente e do ambiente, e as transições são rotuladas por ações, representando as habilidades do agente neste ambiente (Russell; Norvig, 2010).

Considere o domínio de planejamento denominado *Mundo dos Blocos*, no qual uma garra robótica deve movimentar blocos que estão sobre outros blocos ou sobre a mesa. Cada estado representa uma possível configuração de 3 blocos (A, B e C) e as ações são responsáveis pela mudança de um estado para outro. A Figura 1 ilustra dois possíveis estados neste domínio. No estado s_0 temos que é verdade que o bloco C está sobre o bloco A, os blocos A e B estão sobre a mesa e os blocos B e C estão limpos (não há blocos sobre eles). A ação *unstack*(C, A), quando executada em s_0, desempilha o bloco C que está sobre A levando ao estado sucessor s_1 no qual os blocos A e B continuam sobre a mesa, a garra está segurando o bloco C, o bloco A está limpo (não há mais blocos sobre ele) e o bloco B continua limpo.

Figura 1 – Ação determinística no mundo dos blocos.



Fonte: Elaborado pelo autor

No entanto, a descrição de domínios de planejamento, por meio de sistemas de transições de estados, torna-se inviável para problemas reais, uma vez que o número de estados cresce exponencialmente com o número de proposições (Ghallab *et al.*, 2004). Assim, a

comunidade de Planejamento Automatizado (ICAPS, 2023) utiliza *linguagens de ações* para *representar implicitamente* o domínio de planejamento por meio de *ações* com *pré-condições* e *efeitos*. Exemplos de *linguagens de ações* são: PDDL (*Planning Domain Description Language*) (McDermott *et al.*, 1998), STRIPS (*Stanford Research Institute Problem Solver*) (Fikes; Nilsson, 1971) e RDDL (*Relational Dynamic Influence Diagram Language*) (Sanner *et al.*, 2010). Neste trabalho usaremos PDDL por ser a linguagem mais utilizada na Competição Internacional de Planejamento (IPC - *International Planning Competition*) (Vallati *et al.*, 2015).

Em PDDL (Haslum *et al.*, 2019), um domínio é descrito por meio de *predicados* e esquema de *ações*. Os *predicados* são usados para representar propriedades do mundo que podem ser verdadeiras ou falsas em um determinado estado. Cada ação é composta por *pré-condições*, que são proposições que devem ser verdadeiras para que a ação possa ser executada em um estado, e por *efeitos*, que descrevem como os estados serão alterados após a execução da ação (Ghallab *et al.*, 2004).

A Figura 2 ilustra um trecho do domínio do Mundo dos Blocos em PDDL. Os predicados deste domínio são: *on*(?*x* ?*y*) determina se é verdade que um bloco *x* esteja sobre um bloco *y*, *ontable*(?*x*) determina se é verdade que o bloco *x* está sobre a mesa, *clear*(?*x*) determina se é verdade que o bloco *x* está limpo, *holding*(?*x*) determina se é verdade que a garra está segurando o bloco *x*, *handempty* determina se é verdade que a garra está vazia, não está segurando nenhum bloco. A ação *unstack* faz com que a garra segure um bloco que está empilhado sobre outro bloco. Além dessa ação, há no domínio as seguintes ações: *stack* que empilha um bloco que a garra está segurando sobre o outro bloco; *pick-up* na qual a garra pega um bloco que está sobre a mesa e *put-down* na qual a garra solta o bloco que está segurando sobre a mesa.

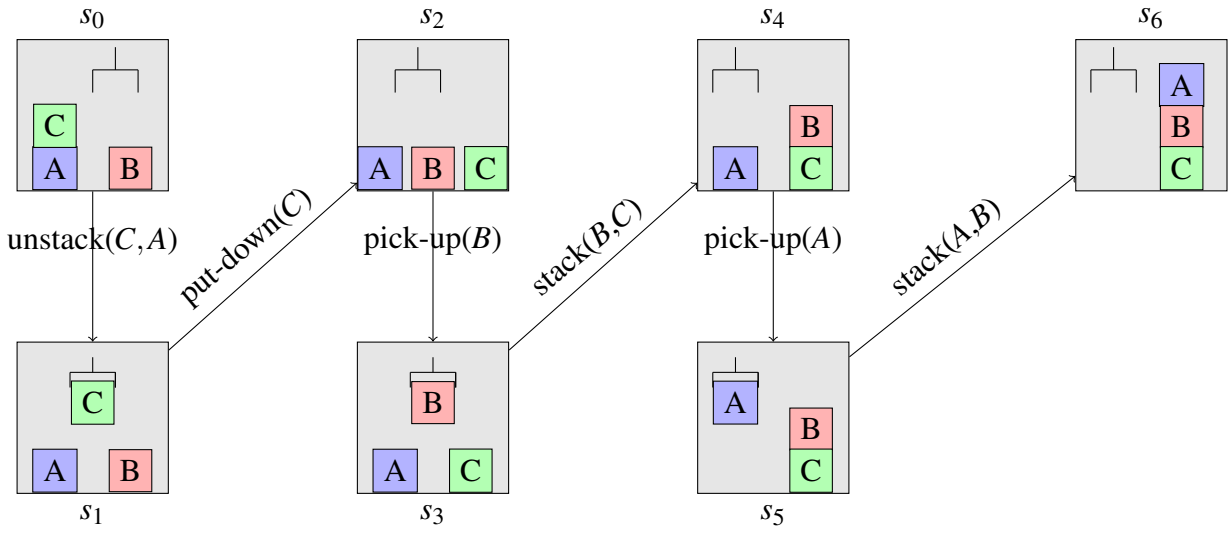
Figura 2 – Trecho do domínio do mundo dos blocos em PDDL.

```
(define (domain blocks)
(:predicates (on ?x ?y) (ontable ?x) (clear ?x) (holding ?x) (handempty))
(:action unstack
     :parameters (?x ?y)
     :precondition (and (clear ?x) (on ?x ?y) (handempty))
     :effect (and (holding ?x) (not (handempty)) (not (on ?x ?y)))
)
...
```

Fonte: Elaborado pelo autor.

Um *plano* é uma sequência de ações que quando executadas levam o agente do estado inicial para um estado meta (Ghallab *et al.*, 2004). Considerando o estado s_0 da Figura 3 e a meta sendo obter a configuração de blocos em que o bloco *A* esteja sobre o bloco *B* e o bloco *B* esteja sobre o bloco *C* (como pode ser visto no estado s_6), um possível plano é: `unstack (C, A), put-down (C), pick-up (B), stack (B, C), pick-up (A), stack (A, B).`

Figura 3 – Estados e transições do domínio de planejamento do mundo dos blocos.

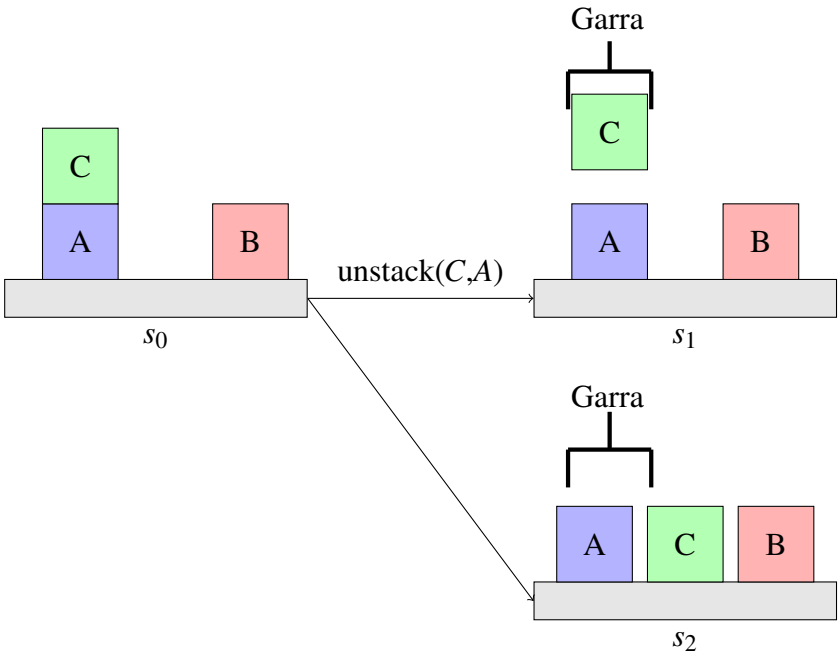


Fonte: Elaborado pelo autor.

Os conceitos sobre planejamento vistos anteriormente, referem-se ao *Planejamento Clássico* no qual supõe-se que o ambiente evolui de forma determinística, isto é, sem qualquer incerteza em relação aos efeitos das ações executadas pelo agente (Ghallab *et al.*, 2004). No entanto, há situações em que os efeitos das ações do agente são não-determinísticos, uma área denominada *Planejamento Não-Determinístico* (Cimatti *et al.*, 2003; Pereira; Barros, 2007; Muise *et al.*, 2012; Santos *et al.*, 2019; Santos *et al.*, 2022). A Figura 4 ilustra uma situação em que a ação `unstack` tem efeitos incertos: pode ser bem sucedida na tarefa de desempilhar o bloco *C* sobre o bloco *A* (levando para o estado s_1) em que a garra está segurando o bloco *C* ou pode falhar nesta execução, desempilhando o bloco *C* sobre o bloco *A*, *mas derrubando o bloco C sobre a mesa*, levando ao estado s_2. A solução para um problema de planejamento com ações não-determinística é denominada **política**. O planejador PRP (do inglês, *Planning for Relevant Policies*) é o planejador estado-da-arte para obtenção de políticas para problemas de planejamento não-determinístico (Muise *et al.*, 2012).

Como vimos, para que se possa planejar, é necessário ter um domínio modelado

Figura 4 – Ação não-determinística no mundo dos blocos.



Fonte: Elaborado pelo autor

em uma linguagem de ações. No entanto, **a aquisição de domínios de planejamento é uma tarefa difícil, que necessita de conhecimentos específicos sobre o domínio a ser modelado (Segura-Muros *et al.*, 2021).** Na literatura, há os seguintes paradigmas de aquisição de domínios de planejamento: (a) codificação manual, (b) codificação automática usando **dados estruturados** (*plan traces*) (Segura-Muros *et al.*, 2021), (c) codificação automática usando dados em linguagem natural não estruturada (Miglani; Yorke-Smith, 2020).

A codificação manual requer conhecimentos específicos sobre os domínios, que são fornecidos por especialistas no assunto. Assim, eles tornam-se responsáveis por todo o processo de Engenharia do Conhecimento, lidando com a aquisição, formulação, validação e manutenção do planejamento para a produção do modelo de domínio (Shah *et al.*, 2013). Na codificação automática, são utilizados algoritmos de aprendizado de máquina (Russell; Norvig, 2010) para aprender domínios a partir de dados que podem ser estruturados ou não-estruturados. Neste trabalho, temos como objetivo a obtenção de **dados estruturados** denominados *plan traces*. Um *plan trace* é uma sequência de estado-ação, representando o histórico de execução de um plano, desde o estado inicial até o estado objetivo (Zhuo; Kambhampati, 2013). É importante ressaltar que na codificação automática com dados não-estruturados o objetivo é obter os *plan traces* a partir de dados em linguagem natural.

Na Figura 3, temos o seguinte *plan trace*: $\langle$ (s_0, `unstack (C, A)`, s_1); (s_1, `put-down (C)`, s_2); (s_2, `pick-up (B)`, s_3); (s_3, `stack (B, C)`, s_4); (s_4, `pick-up (A)`, s_5); (s_5, `stack`

Figura 5 – Exemplo de *plan tace*, referente a Figura 3

(a) Plano

Início	Fim	*Ação*
s0	s1	*unstack (C, A)*
s1	s2	*put-down (C)*
s2	s3	*pick-up (B)*
s3	s4	*stack (B, C)*
s4	s5	*pick-up (A)*
s5	s6	*stack (A, B)*

(b) Lista de Estados

Estado	*Predicados*
s0	*(ontable A)*, *(ontable B)*, *(on C A)*, *(handempity)*
s1	*(ontable A)*, *(ontable B)*, *(holding C)*
s2	*(ontable A)*, *(ontable B)*, *(ontable C)*, *(handempity)*
s3	*(ontable A)*, *(ontable C)*, *(holding B)*
s4	*(ontable A)*, *(ontable C)*, *(on B C)*, *(handempity)*
s5	*(ontable C)*, *(on B C)*, *(holding A)*
s6	*(ontable C)*, *(on B C)*, *(on A B)*, *(handempty)*

Fonte: Elaborada pelo Autor

`(A, B)`, s_6) $\rangle$. Para ser melhor utilizado por algoritmos de aprendizagem de máquina, os *plan traces* são especificados em duas partes principais (Segura-Muros *et al.*, 2021): (a) plano e (b) lista de estados, conforme mostrado na Figura 5. Nessa figura temos a parte (a) na qual são listadas as ações pertencentes a um plano solução com informações, para cada ação, do seu estado de início (de onde a ação parte) e fim (para onde a ação leva). A primeira linha da parte (a) do *plan trace*, "s_0 s_1 *unstack (C, A)*" é referente a transição (s0,*unstack (C, A)*, s1) na Figura 3. Já a parte (b) do *plan trace* corresponde aos predicados que se sabe que são verdadeiros em cada um dos estados do domínio da Figura 3 (informação parcial do estado) . Conforme (Segura-Muros *et al.*, 2021), essas informações não precisam ser completas, significando que se um dado predicado não consta na definição do estado não se sabe se ele é verdadeiro ou falso.

Recentemente, a abordagem de Planejamento Automatizado têm sido usada na tarefa de gerenciamento de Sistemas de Diálogos (*chatbots*) (Muise *et al.*, 2019), uma vez que algoritmos de aprendizagem de máquina não são confiáveis para gerenciar diálogos em sistemas críticos (como saúde, por exemplo) e nem são capazes de fornecer explicações para suas respostas (Muise *et al.*, 2019; DEEP-DIAL, 2018). **As ações de gerenciamento de diálogo são em sua essência não-determinísticas**, pois não se sabe *a priori* que tipo de resposta o usuário irá fornecer em um diálogo. O trabalho de Pinho (2024) propõe o uso de um gerenciador de diálogos baseado em planejamento e **realiza a aquisição manual** de um domínio com ações não-determinísticas para gerenciar o *chatbot* Plantão Coronavírus. Este trabalho utiliza o planejador

PRP (Muise *et al.*, 2012) para obtenção automática da política que representa todos os possíveis diálogos neste domínio. Ademais, (Silva, 2024) realiza **a obtenção dos *plan traces* a partir de dados de conversa em linguagem natural** realizadas neste *chatbot*.

O **Plantão Coronavírus**[1] é um *chatbot*, proposto pelo Governo do Estado do Ceará, para auxiliar no atendimento de pacientes com suspeita de COVID-19. Este *chatbot* ajuda na detecção de possíveis casos da doença, oferecendo orientação especializada através de profissionais de saúde (Cardoso, 2020). Seu uso tornou-se particularmente relevante durante os momentos mais críticos da pandemia de COVID-19 nos anos de 2020 e 2021, quando havia um aumento significativo nos casos da doença e escassez de profissionais de saúde disponíveis para atendimento (Mont'alvarine, 2021).

1.1 Objetivos

1.1.1 Objetivo Geral

Obter dados estruturados (*plan traces*) a partir da política do gerenciador do *chatbot* Plantão Coronavírus para serem utilizados por algoritmos de aquisição automática de domínios de planejamento.

1.1.2 Objetivos Específicos

- A partir da política do gerenciador do *chatbot* Plantão Coronavírus, obter a parte (a) (*plano*) de cada um dos *plan traces*, isto é, as informações de estado de início e fim para cada ação de cada um dos caminhos da política.
- A partir da política do gerenciador do *chatbot* Plantão Coronavírus, obter a parte (b) (*lista de estados*) de cada um dos *plan traces*, isto é, as informações que se é possível saber sobre os predicados que descrevem cada um dos estados da política.
- Realizar comparativo entre os *plan traces* obtidos a partir da política com os *plan traces* obtidos a partir dos dados de conversa em linguagem natural (Silva, 2024).

1.1.3 Organização do texto

Este trabalho está organizado de forma a apresentar: no Capítulo 2, a fundamentação teórica incluindo os tópicos Planejamento Automatizado, Aquisição de Domínios de Planeja-

[1] https://coronavirus.ceara.gov.br/

mento e Gerenciamento de Agentes de Diálogo; no Capítulo 3, os trabalhos relacionados com essa pesquisa, que utilizam *plan traces* para aquisição automática de domínios de planejamento; no Capítulo 4, a metodologia; no Capítulo 5 os resultados e; no Capítulo 6 as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Planejamento Automatizado

Planejamento Automatizado é a subárea da Inteligência Artificial (IA) que estuda modelos e algoritmos para obtenção de um plano de ações para que um agente inteligente alcance suas metas (Ghallab *et al.*, 2004). A abordagem clássica de planejamento supõe que o ambiente evolui de maneira determinística. Assim, as ações executadas pelo agente não sofrem interferências do ambiente (Pereira; Barros, 2007). Na prática existem situações em que a suposição de determinismo adotada pelo planejamento clássico se mostra inadequada. Na verdade, o ambiente de planejamento pode evoluir de forma não-determinística levando em consideração que as ações do agente tem efeitos incertos (Muise *et al.*, 2012).

2.1.1 Planejamento Determinístico

Um domínio de planejamento determinístico pode ser descrito formalmente por meio de um sistema de transição de estados como mostrado na Definição 1.

Definição 1 *(Domínio de Planejamento Determinístico) Dados um conjunto de átomos proposicionais* $\mathbb{P}$ *e um conjunto de ações* $\mathbb{A}$*, um domínio de planejamento pode ser representado por um sistema de transição de estados* $D = \langle S, L, T \rangle$ *em que (Pereira; Barros, 2007):*

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

No entanto, essa abordagem torna-se impraticável em situações reais devido ao grande número de estados possíveis do sistema de transição de estados. Por isso, linguagens como a PDDL (McDermott *et al.*, 1998) foram desenvolvidas para representar domínios de planejamento de forma sucinta. Em PDDL, as ações são definidas por pré-condições e efeitos. A Figura 2 ilustra um trecho do domínio do Mundo dos Blocos em PDDL.

No planejamento determinístico, o objetivo do agente é alcançar determinados estados do ambiente onde condições específicas são atendidas, os chamados estados metas (Pereira; Barros, 2007). Assim, um problema específico dentro do domínio é definido com as adições de um estado inicial e um objetivo, como mostra a Definição 2.

Definição 2 *(Problema de Planejamento Determinístico) Dado um domínio de planejamento D, um problema de planejamento (Ghallab et al., 2004) é definido por uma tupla* $P = \langle D, s_0, \varphi \rangle$ *em que:*

- *D é o domínio de planejamento sobre um conjunto de proposições* $\mathbb{P}$ *e um conjunto de ações* $\mathbb{A}$*;*
- $s_0 \in S$ *é o estado inicial do problema;*
- φ *é um fórmula em lógica proposicional que descreve a meta de planejamento.*

Um plano é a sequência de ações necessárias para atingir o objetivo, enquanto o fracasso indica que não foi encontrada uma solução para o problema.

Definição 3 *(Plano) Seja* $P = \langle D, s_0, \varphi \rangle$ *um problema de planejamento em um domínio* $D = \langle S, L, T \rangle$ *com ações determinísticas. Um plano é uma sequência de ações* $\pi = \langle a_1, ..., a_k \rangle$*, onde* $k \geq 0$*, em que* a_1 *é uma ação aplicada no estado inicial* s_0 *e* a_k *é uma ação que alcança um estado* $s_g \models \varphi$*. (Ghallab et al., 2004).*

2.1.2 Planejamento Não-Determinístico

Um domínio de planejamento não-determinístico pode ser descrito formalmente por meio de um sistema de transição de estados como mostrado na Definição 4.

Definição 4 *(Domínio de Planejamento Não-Determinístico) Dados um conjunto de átomos proposicionais* $\mathbb{P}$ *e um conjunto de ações* $\mathbb{A}$*, um domínio de planejamento pode ser representado por um sistema de transição de estados* $D = \langle S, L, T \rangle$ *em que (Pereira; Barros, 2007):*

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

Em PDDL, para representar os efeitos não-determinísticos de uma ação é utilizada a palavra reservada `one-of`. Veja na Figura 6 a descrição da ação não-determinística `unstack`. Nela, podem ocorrer as seguintes situações:

- (i) `(holding ?x) (not (handempty)) (not (on ?x ?y))` *ou*
- (ii) `(ontable ?x) (not (on ?x ?y))`

Na situação (i) a garra é bem sucedida na sua tarefa de desempilhar o bloco *x* sobre o bloco *y*, uma vez que: ela está segurando o bloco *x* (`(holding ?x)`), não está com as “mãos

vazias" ((not (handempty))) e o bloco x não está sobre o bloco y ((not (on ?x ?y))). Na situação (ii), a garra também desempilha x sobre y ((on ?x, ?y)), mas derruba o bloco x sobre a mesa ((ontable ?x)).

Figura 6 – Domínio do mundo dos blocos com ação não-determinística.

```
(define (domain blocks)
(:predicates (on ?x ?y) (ontable ?x) (clear ?x) (holding ?x) (handempty))
(:action unstack
     :parameters (?x ?y)
     :precondition (and (clear ?x) (on ?x ?y) (handempty))
     :effect (oneof (and (holding ?x) (not (handempty)) (not (on ?x ?y))
                    (and (ontable ?x) (not (on ?x ?y)))
     )
)
...
```

Fonte: Adaptada de (Ghallab *et al.*, 2004)

Da mesma forma que na abordagem determinística, um problema de planejamento com ações não-determinísticas é formulado ao incluir um estado inicial e um objetivo específico dentro de um determinado contexto ou domínio. No entanto, uma solução para um problema de planejamento não-determinístico é descrita pela consideração das múltiplas execuções possíveis de um plano, que é definido como uma política, conforme mostrado na Definição 5.

Definição 5 *(Política) Seja* $P = \langle D, s_0, \varphi \rangle$ *um problema de planejamento em um domínio* $D = \langle S, L, T \rangle$ *com ações não-determinísticas. Uma política* π *é uma função parcial* $\pi : S \rightarrow A$ *que mapeia estados em ações; tal que, para cada estado* $s \in S$ *se* π *está definido para* s *então* $\pi(s) \in \{a \in \mathbb{A} : T(s,a) \neq \emptyset\}$ *(Pereira; Barros, 2007).*

Uma política pode ser: fraca, forte ou forte-cíclica. Uma política fraca é uma solução que é possível de atingir o estado meta, porém não é garantido devido à incerteza associada aos efeitos das ações. Uma política forte é caracterizada por assegurar o alcance de um estado meta, mesmo diante das incertezas inerentes ao ambiente. Já uma política forte-cíclica corresponde a uma solução em que é garantido o alcance de um estado meta, mas que inclui a possibilidade de existirem ciclos no percurso (Cimatti *et al.*, 2003).

2.2 Aquisição de domínios de planejamento

A aquisição de domínios de planejamento é uma tarefa complexa, principalmente devido à necessidade de possuir um amplo conhecimento sobre as particularidades dos domínios a serem modelados (Segura-Muros *et al.*, 2021). Nesse cenário, criar um modelo de domínio para aplicação no mundo real exige uma colaboração entre especialistas no assunto, conhecidos como SMEs (*Subject Matter Experts*), e engenheiros de conhecimento, chamados de KEs (*Knowledge Engineers*). No entanto, esse processo é demorado e suscetível a erros (Miglani; Yorke-Smith, 2020).

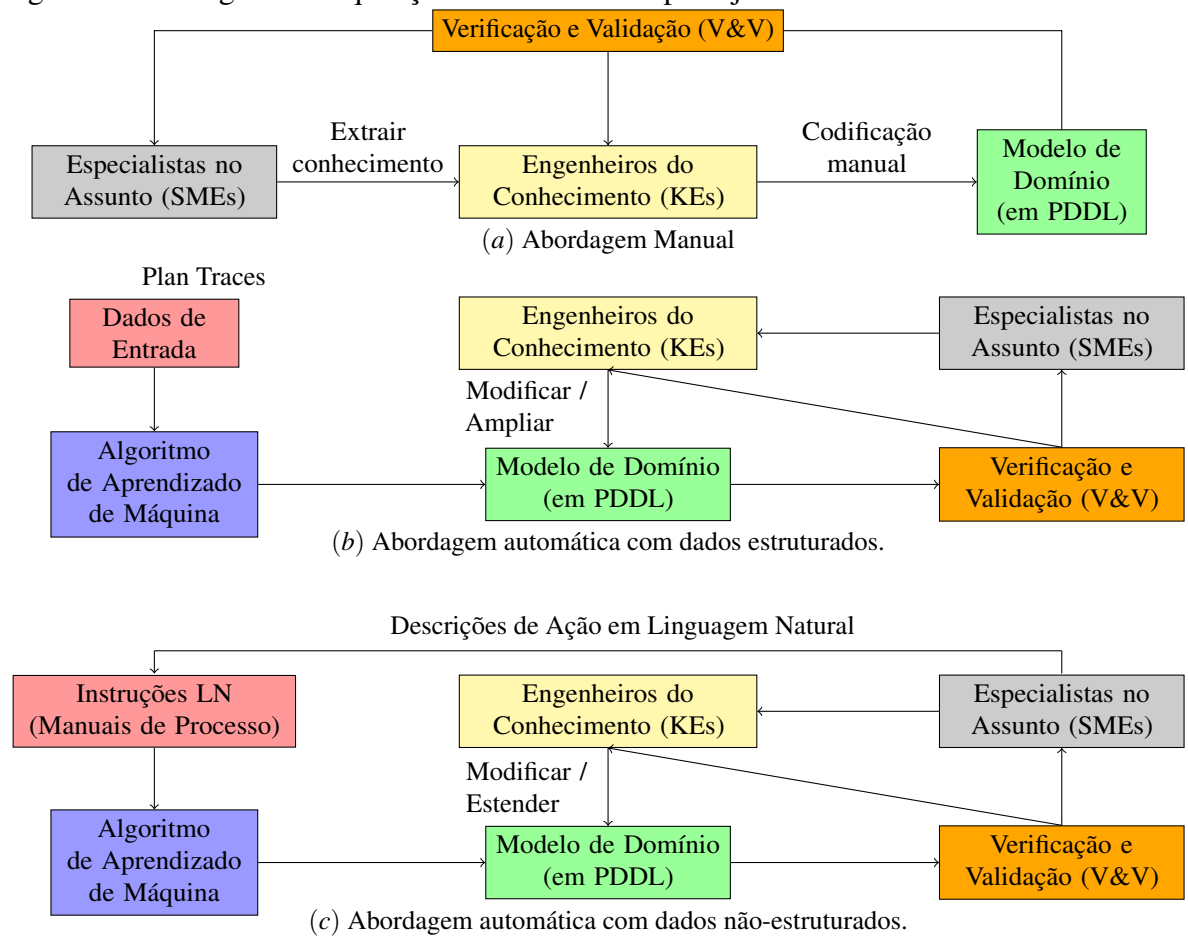
A Figura 7 ilustra os principais paradigmas de aquisição de domínios de planejamento (Miglani; Yorke-Smith, 2020):

- abordagem manual;
- abordagem automática a partir de dados estruturados (*plan traces*) (Segura-Muros *et al.*, 2021);
- abordagem automática baseada em dados não estruturados a partir de dados em Linguagem Natural (Miglani; Yorke-Smith, 2020).

Na *abordagem manual* há a intensa participação de especialistas no assunto, uma vez que requer conhecimento profundo nos domínios em questão. Nesse sentido, eles assumem a responsabilidade por todas as etapas do processo de Engenharia do Conhecimento, englobando a coleta, formulação, validação e manutenção do plano para desenvolver o modelo de domínio (Shah *et al.*, 2013). A principal preocupação associada a essa abordagem é a possível introdução de erros durante o processo de codificação manual, uma vez que o conhecimento dos domínios do mundo real é limitado pela perspectiva humana (Arora *et al.*, 2018). Na Figura 7(a) temos o fluxograma referente a esta abordagem, na qual os especialistas extraem o conhecimento que é passado para os engenheiros que fazem a codificação manual, gerando um modelo de domínio em PDDL. O modelo gerado passa por um processo de verificação e validação e, a partir desse ponto, pode ser feito o fluxo novamente para obter um domínio mais preciso.

A *abordagem automática a partir de dados estruturados* obtém o modelo de domínio a partir de históricos de planos (*plan traces*) (Miglani; Yorke-Smith, 2020). Um *plan trace* é uma sequência de estados e ações, destacando-se para cada ação seu pré-estado e pós-estados. Define-se por pré-estado o estado anterior a execução de uma ação e pós-estado o estado posterior a execução de uma ação (Segura-Muros *et al.*, 2021). A Figura 5 na introdução do trabalho ilustra um *plan trace* para o domínio do mundo dos blocos.

Figura 7 – Paradigmas de aquisição de domínios de planejamento.



(a) Abordagem Manual

(b) Abordagem automática com dados estruturados.

(c) Abordagem automática com dados não-estruturados.

Fonte: Adaptada de (Silva, 2024)

Em muitas situações do mundo real, os *plan traces* são extraídos ou gerados por sistemas já disponíveis no mercado (Zhuo *et al.*, 2019). A Figura 7(b) ilustra o fluxograma desta abordagem, na qual os dados de entrada no formato de *plan traces* são submetidos ao algoritmo de aprendizado de máquina que fornece como saída o domínio em PDDL. O modelo gerado passa por um processo de verificação e validação, onde os engenheiros podem fazer modificações ou ampliar o domínio gerado se necessário.

A *abordagem automática a partir de dados não estruturados* baseia-se na utilização de dados em linguagem natural como entrada para a aquisição de modelos de domínio para planejamento. Esses dados estão disponíveis, seja na forma de instruções, manuais de processos ou descrições de ações. Essa forma de entrada representa uma maneira natural e intuitiva de interagir com os usuários e é uma maneira eficaz de compreender os elementos de um modelo de domínio e criar exemplos de planos (Miglani; Yorke-Smith, 2020). A Figura 7(c) ilustra o fluxograma desse processo de aquisição, no qual os dados de entrada (Instruções em Linguagem Natural) são submetidos ao algoritmo de aprendizado de máquina que geram como saída o

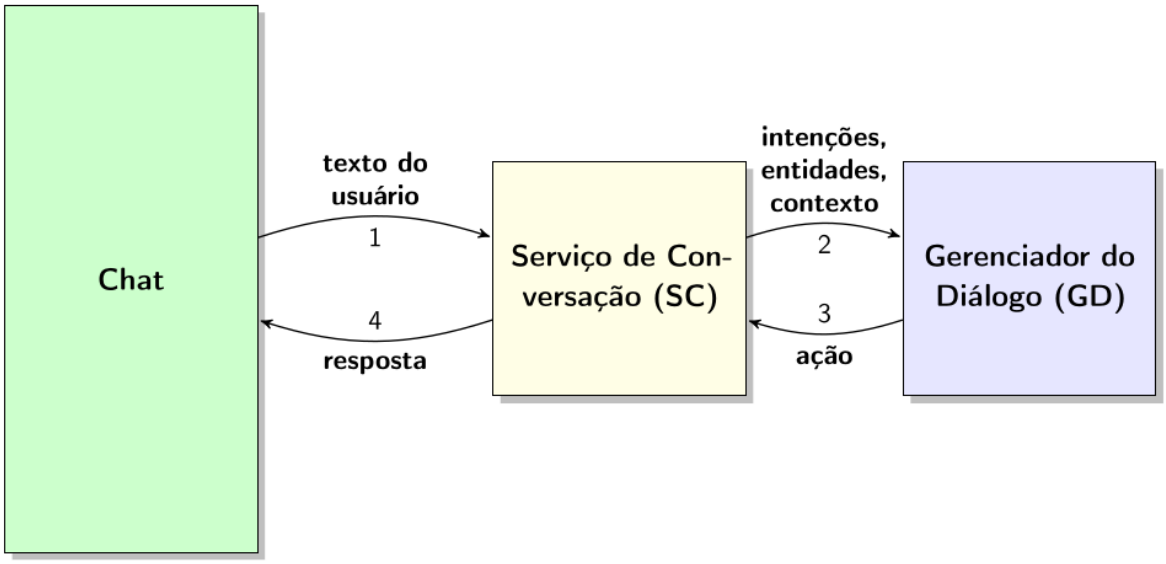
domínio em PDDL. O modelo gerado passa por um processo de verificação e validação, em que os Engenheiros do Conhecimento podem fazer modificações ou ampliar o domínio se necessário. Além disso os Especialistas no Assunto podem adicionar descrições de ação em Linguagem Natural para a entrada.

2.3 Gerenciador de Diálogos do Plantão Coronavírus baseado em Planejamento Automatizado

Sistemas de diálogo, também conhecidos como *chatbots*, são programas projetados para interagir com o usuário, buscando manter a percepção de que se está conversando com outro ser humano (Tede; Barros, 2016). A Figura 8 apresenta a arquitetura básica de um sistema de diálogo. Nesta arquitetura há o módulo *Chat* que recebe o texto digitado pelo usuário na interface do programa. Este texto é chamado de *declaração do usuário*. Na seta 1 da Figura 8 a *declaração do usuário* é enviada para o módulo *SC* (*Sistema de Conversação*). No *SC* as declarações do usuário são classificadas em *intenções*, que descrevem como as declarações do usuário devem ser categorizadas. Ainda no *SC* é feito o reconhecimento das *entidades* que são variáveis estruturadas dentro das declarações do usuário. Elas podem ser de uso geral como nomes, lugares, números, etc. O *SC*, baseado no histórico de conversação, extrai o *contexto* da conversa, o qual pode conter informações que foram ditas até o momento no diálogo. As intenções, as entidades e o contexto são enviados do *SC* para o módulo **GD** (*Gerenciador do Diálogo*), como podemos ver na seta 2 da Figura 8. O *GD* é o componente central de um sistema de diálogo. Ele controla o fluxo do diálogo, decidindo o que será dito para o usuário com base nas suas entradas. O *GD* utiliza então as intenções e as entidades para atualizar o contexto e decidir, com base no contexto atual, qual *ação* o sistema vai executar. Após a escolha da ação, o *GD* envia para o *SC* a ação, como podemos ver na seta 3 da Figura 8. No *SC*, a ação é transformada em texto, o qual é enviado para o módulo *Chat*, como podemos ver na seta 4 (Pinho, 2024).

Recentemente, a abordagem de Planejamento Automatizado têm sido usada na tarefa do gerenciamento do diálogos (Muise *et al.*, 2019), uma vez que algoritmos de aprendizagem de máquina podem não ser confiáveis para gerenciar diálogos em sistemas críticos (como saúde, por exemplo) e nem são capazes de fornecer explicações para suas respostas (Muise *et al.*, 2019; DEEP-DIAL, 2018). As ações de gerenciamento de diálogo são em sua essência não-determinísticas, pois não se sabe *a priori* que tipo de resposta o usuário irá fornecer em um diálogo. O trabalho de Pinho (2024) propõe o uso de um gerenciador de diálogos baseado em

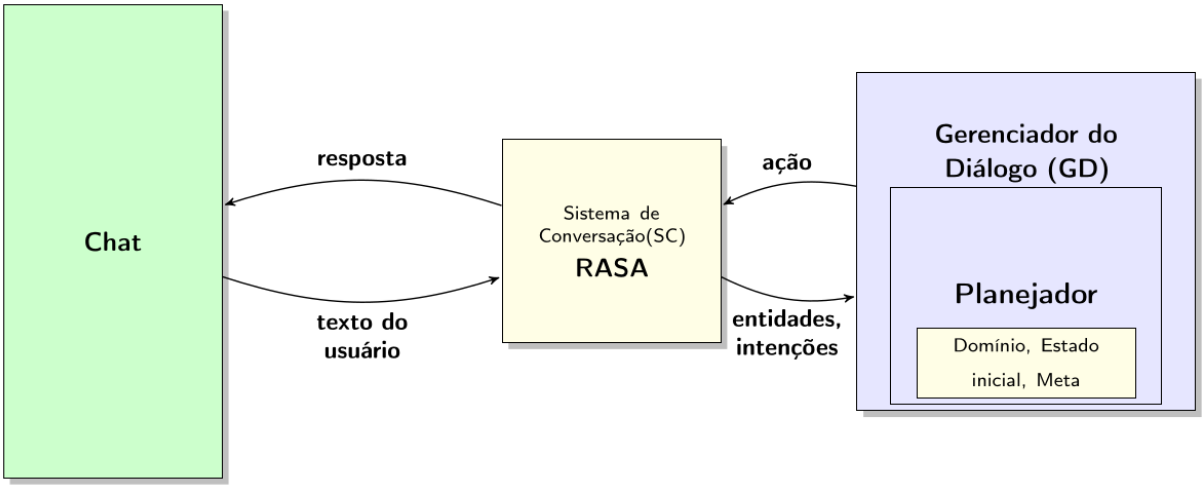
Figura 8 – Arquitetura básica de um sistema de diálogo



Fonte: (Pinho, 2024)

planejamento e **realiza a aquisição manual** de um domínio com ações não-determinísticas para gerenciar o *chatbot* Plantão Coronavírus. Este trabalho utiliza o planejador PRP (do inglês, *Planning for Relevance Policies*), um planejador estado-da-arte para solucionar problemas de planejamento com ações não-determinísticas (Muise *et al.*, 2012), para obtenção automática da política que representa todos os possíveis diálogos neste domínio. Ademais, (Silva, 2024) realiza **a obtenção dos *plan traces* a partir de dados de conversa em linguagem natural** realizadas neste mesmo *chatbot*.

Figura 9 – Arquitetura proposta para o Sistema de Diálogo, utilizando o Rasa como SC e o Planejador como GD



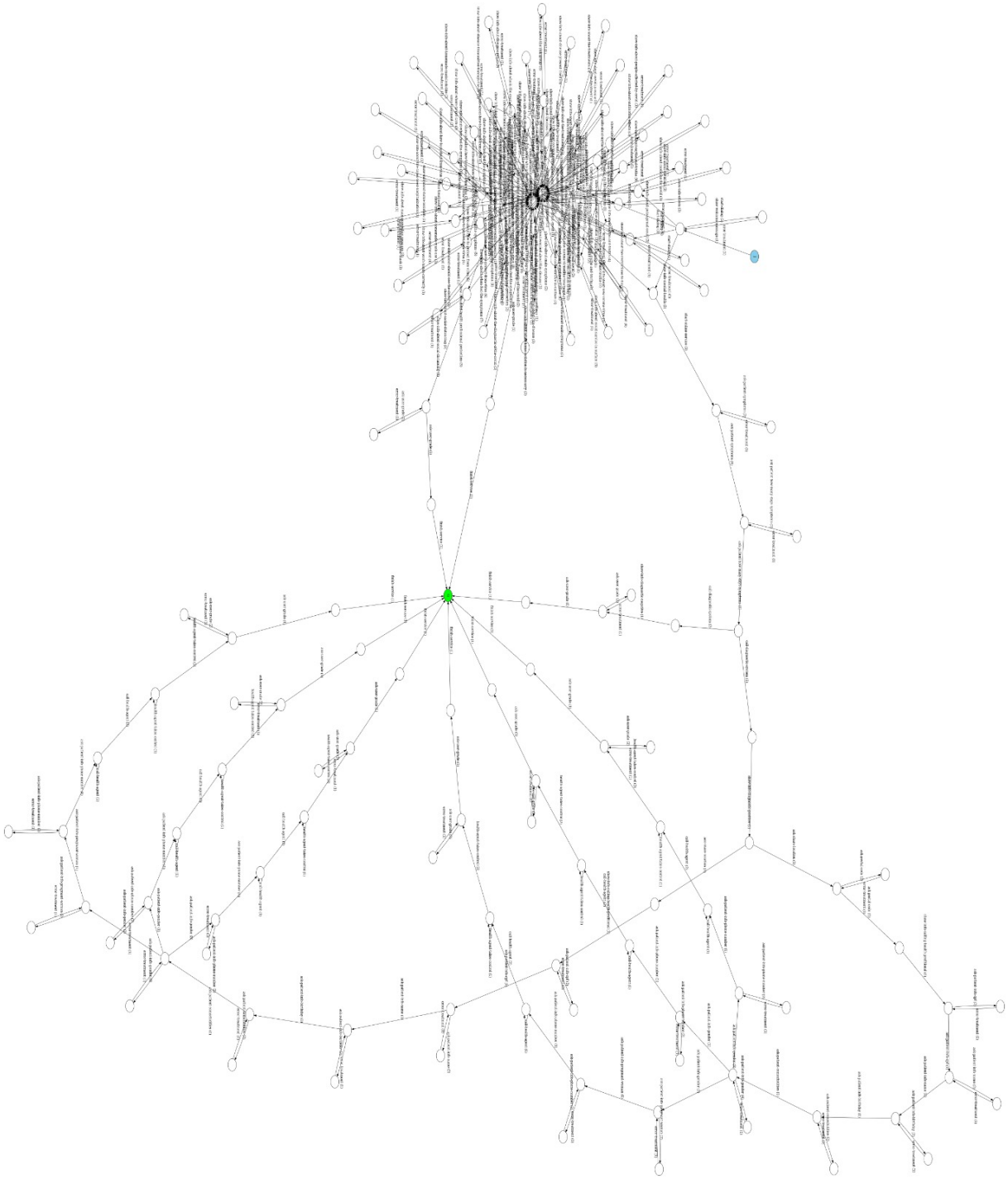
Fonte: (Pinho, 2024)

A solução proposta por (Pinho, 2024) para a arquitetura do *chatbot* (Figura 9) utiliza planejamento automatizado na tarefa de gerenciamento do diálogo e é baseada na modelagem de ações proposta por Muise et al. (2019). Neste caso, foram utilizados:

- a ferramenta Rasa, por ser uma ferramenta grátis e de código aberto, na tarefa de serviço de conversação no lugar da ferramenta IBM Watson utilizada por Muise *et al.* (2019) e;
- o planejador PRP (Planner for Relevant Policies) (Muise *et al.*, 2013) para obtenção de políticas para o gerenciamento do diálogo por ser considerado o planejador estado-da-arte para solucionar problemas de planejamento não-determinísticos.

Por fim, foi possível obter automaticamente a politica com 168 estados e 88 ações correspondente a todos os caminhos possíveis do diálogo, conforme pode ser visto na Figura 10. Nesta figura, o estado inicial está destacado em azul e a meta na cor verde.

Figura 10 – Política do *chatbot* Plantão Coronavírus



Fonte: (Pinho, 2024)

3 TRABALHOS RELACIONADOS

3.1 Discovering Relational and Numerical Expressions from Plan Traces for Learning Action Models

Segura-Muros *et al.* (2021) propõem o algoritmo `PlanMiner` para aquisição de um domínio de planejamento com ações determinísticas, usando **aprendizado de máquina com dados estruturados** (*plan traces*). O `PlanMiner` transforma o problema de aquisição de domínios em um problema de classificação (Kotsiantis *et al.*, 2006). De acordo com os autores, a abordagem de classificação é adotada porque as características que definem as pré-condições e os efeitos das ações estão relacionadas aos estados do ambiente. Esses estados incluem o estado anterior à aplicação da ação (pré-estado) e o estado resultante após a aplicação da ação (pós-estado).

O processo de aquisição do domínio pelo `PlanMiner` ocorre em quatro fases:

- *Extração de Dados*: Inicialmente, o sistema recebe um conjunto de *plan traces* de entrada e os transforma em uma coleção de conjuntos de dados. Dado que as estruturas de dados utilizadas em algoritmos convencionais de aprendizado de máquina são distintas do formato PDDL, é necessário realizar uma conversão para adequar a entrada ao formato de dados apropriado.
- *Descoberta de novas informações*: A partir dos conjuntos de dados gerados na etapa anterior, o `PlanMiner` utiliza técnicas de regressão simbólica (Manly; Alberto, 2016) para gerar novos conhecimentos e melhorar os conjuntos de dados. Essa etapa é fundamental, uma vez que, para aprender expressões aritméticas e relacionais, é necessário incorporar novos conhecimentos de forma explícita nos conjuntos de dados.
- *Aquisição de Modelos de Classificação*: Utilizando os conjuntos de dados como entrada, o `PlanMiner` emprega um algoritmo de classificação para desenvolver um modelo de classificação para cada conjunto de dados. As suposições incorporadas nesses modelos de classificação definem as características essenciais necessárias para representar um conjunto de estados intermediários.
- *Geração de Domínios de Planejamento*: Por fim, os modelos de classificação são processados para criar um conjunto de modelos de ação. Pré-condições e efeitos são extraídos de cada modelo de classificação para construir um esquema de ação. O resultado final do `PlanMiner` é um domínio PDDL ao combinar os modelos de ação adquiridos.

A abordagem de Segura-Muros *et al.* (2021) assemelha-se com o objetivo deste trabalho, uma vez que ambas fazem uso de *plan traces*. No entanto, existem diferenças significativas entre os trabalhos. O PlanMiner (Segura-Muros *et al.*, 2021) faz a aquisição de um domínio com ações determinísticas a partir de dados estruturados (*plan traces*) para aprender domínios *benchmarks* e este trabalho foca na obtenção de *plan traces*, que podem ser utilizados posteriormente para a aquisição de domínios com ações não-determinísticas, voltados para o gerenciamento de agentes de diálogo.

3.2 NLtoPDDL: One-Shot Learning of PDDL Models from Natural Language Process Manuals

A abordagem NLtoPDDL (Miglani; Yorke-Smith, 2020) realiza a aquisição automática de domínios de planejamento a partir de dados não estruturados, os quais são representados por meio de textos em Linguagem Natural. A abordagem utiliza como entrada manuais de processos nos quais os dados são apresentados em forma de linhas de instrução. O processo de aquisição dos domínios é realizado por algoritmos de aprendizado de máquina em duas fases: (i) extração de sequências de ações para posterior extração de *plan traces* a partir dos dados não estruturados e (ii) aquisição automatizada do domínio usando os dados estruturados adquiridos na fase anterior. Os domínios aprendidos possuem ações determinísticas.

A abordagem do trabalho assemelha-se com a proposta deste trabalho, pois ambas fazem uso de *plan traces* que podem ser usados para fazer a aquisição automática de domínios de planejamento. No entanto, existem diferenças significativas entre as duas abordagens, uma das quais é a entrada de dados, uma vez que o trabalho de (Miglani; Yorke-Smith, 2020) usa manuais de processos como entrada, já este trabalho usa os dados da política para obter os *plan traces*. Também, no trabalho de (Miglani; Yorke-Smith, 2020) os domínios aprendidos são domínio com ações determinísticas relacionados a guias e manuais de processos e, neste trabalho, o foco é na obtenção de *plan traces*, que podem ser utilizados posteriormente para a aquisição de domínios com ações não-determinísticas, voltados para o gerenciamento de agentes de diálogo.

3.3 LOUGA: Learning Planning Operators using Genetic Algorithms

Kučera e Barták (2018) propõem um algoritmo, denominado LOUGA, para aprender ações determinísticas a partir de dados estruturados (*plan traces*). O LOUGA recebe como

entrada o estado inicial e um conjunto de *plan traces* e tem como objetivo aprender uma função de transição de estado correta de acordo com as sequências observadas de ações e estados, contidas nos *plan traces*. Para isso, utiliza um algoritmo genético para aprender os efeitos das ações e um algoritmo *ad-hoc* para aprender as precondições das ações.

A abordagem do trabalho assemelha-se com a proposta deste trabalho, pois ambas fazem uso de *plan traces* que podem ser usados para fazer a aquisição automática de domínios de planejamento. No entanto, existem diferenças entre as duas abordagens, uma das quais é o tipo de domínio aprendido, que na abordagem de (Kučera; Barták, 2018) são domínios *benchmarks* determinísticos e, neste trabalho, queremos realizar a aquisição de *plan traces* que podem ser usados para realizar a aquisição de domínios para gerenciamento de agentes de diálogos com ações não-determinísticas. Ademais, o tipo de algoritmo utilizado na aquisição também é diferente, no qual (Kučera; Barták, 2018) utiliza algoritmos genéticos para aprender as ações e este trabalho pretende adquirir os *plan traces* para serem utilizados como entrada de algoritmos de aquisição automática de domínios de planejamento.

3.4 Comparação dos Trabalhos Relacionados

O Quadro 1 apresenta a comparação entre os trabalhos relacionados e o trabalho proposto. As principais características consideradas são: os dados utilizados como fonte de conhecimento para a abordagem, se o domínio é determinístico ou não-determinístico e o tipo de aplicação na qual os domínios adquiridos são inseridos.

Quadro 1 – Comparação dos Trabalhos Relacionados.

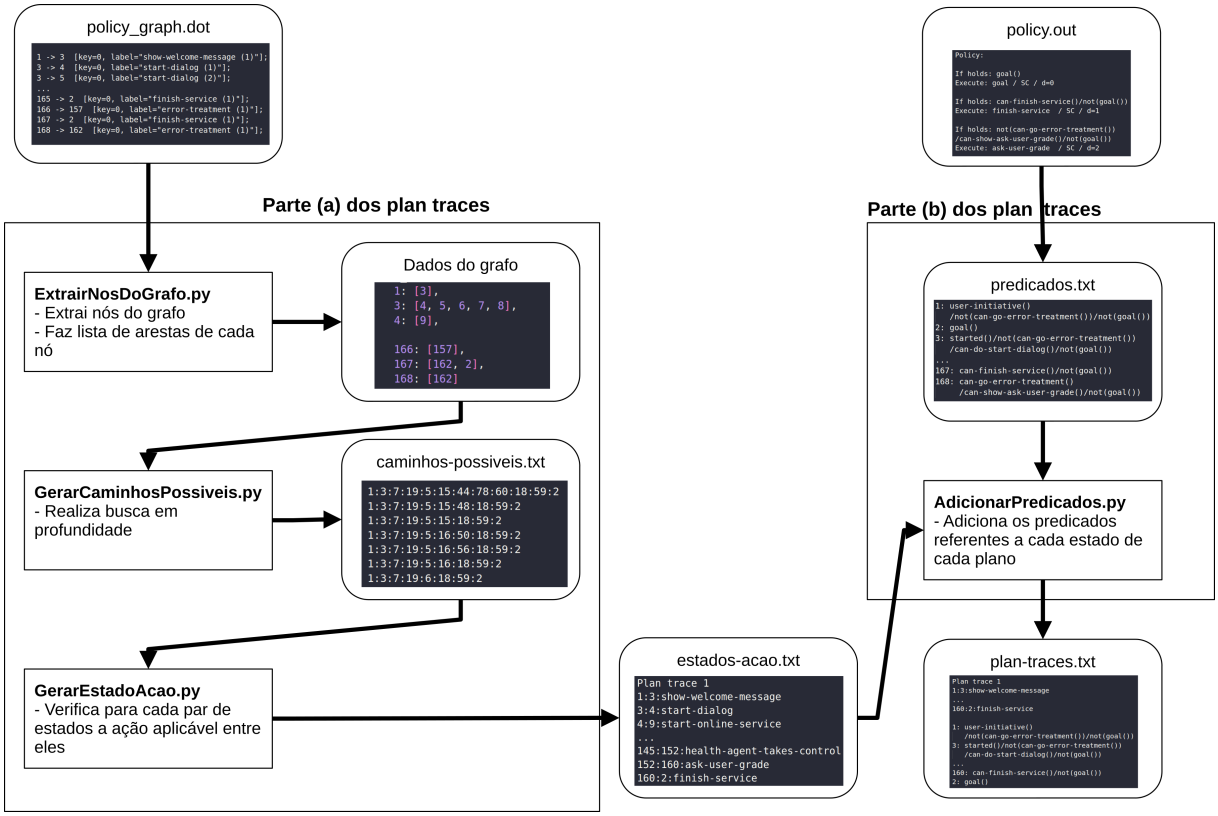
Trabalho	**Fonte de aquisição de conhecimento**	**Tipo de Domínio**	**Aplicação**
(Segura-Muros *et al.*, 2021)	*Plan traces*	Determinístico	Domínios *benchmarks* de planejamento
(Miglani; Yorke-Smith, 2020)	Manuais de processos	Determinístico	Guias e manuais
(Kučera; Barták, 2018)	*Plan traces*	Determinístico	Domínios *benchmarks* de planejamento
Trabalho proposto	*Plan traces*	Não-determinístico	Sistemas de diálogo

Fonte: Elaborado pelo autor.

4 METODOLOGIA

A proposta deste trabalho é obter dados estruturados, denominados *plan traces*, os quais são fundamentais para realizar aquisição automática de domínios de planejamento (Miglani; Yorke-Smith, 2020; Segura-Muros *et al.*, 2021). Os dados aqui em questão são provenientes da política gerada pelo gerenciador de diálogos do *chatbot* Plantão Coronavírus (Pinho, 2024). As etapas a serem realizadas neste projeto são descritas a seguir e ilustradas na Figura 11.

Figura 11 – Arquitetura do sistema que recebe como entrada dados da política e elabora um arquivo *.txt* contendo os *plan traces*.



Fonte: Elaborada pelo Autor

4.1 Obtenção da parte (a) (plano) de cada *plan trace*

Para obter os planos (parte (a)) dos *plan traces* o sistema recebe como entrada o arquivo "policy_graph.dot", obtido por meio da execução do planejador PRP (Muise *et al.*, 2012). Um exemplo do aquivo de entrada pode ser observado na Figura 12, em que existe os estados do grafo representados de forma numérica e com o label sendo o nome da ação de transição aplicada entre os estados. A partir desse arquivo, um programa *Python* realiza a extração dos dados do

grafo. Esse algoritmo primeiro identifica todos os nós existentes no grafo e, em seguida, lista as arestas correspondentes a cada nó.

Figura 12 – Exemplo do arquivo de entrada `policy_graph.dot`

```
1 -> 3  [key=0, label="show-welcome-message (1)"];
3 -> 4  [key=0, label="start-dialog (1)"];
...
167 -> 2  [key=0, label="finish-service (1)"];
168 -> 162  [key=0, label="error-treatment (1)"];
```

Fonte: Elaborada pelo Autor

O processo inicia-se com a leitura do arquivo da política. O algoritmo percorre cada linha do arquivo, extraindo o nó de origem e o nó de destino. Esses nós são então adicionados a uma lista, caso ainda não tenham sido registrados. A separação das informações é feita utilizando um delimitador específico de *string*, o que permite ao algoritmo identificar e armazenar as arestas que conectam os nós.

Com as arestas devidamente registradas, os dados são passados para outro programa Python, denominado `GerarCaminhosPossiveis.py`, que realiza uma busca no espaço de estados da política. Após a identificação de todos os caminhos possíveis, esses dados são submetidos a um terceiro programa *Python*, denominado `GerarAcaoEstado.py`. Este verifica cada par de estados consecutivos e, utilizando os dados das ações associadas, identifica qual ação corresponde a cada par de estados. Esse processo resulta na geração do plano (parte (a)) dos *plan traces* como pode ser observado na Figura 11, que inclui a sequência de estados e as respectivas ações que os conectam.

4.2 Obtenção dos predicados que descrevem cada estado da política

Para obter a lista de estados (parte (b)) dos *plan traces*, foi utilizado o arquivo *policy.out*, gerado pelo planejador PRP. Esse arquivo contém informações sobre os predicados associados a cada estado da política, como pode ser observado na Figura 13. As linhas com *"If holds"* representam os predicados, que em seguida tem a ação que pode ser executada quando esses predicados são verdadeiros. Para cada ação mapeada existem predicados referentes a essa ação. A partir desses dados, foi possível compilar uma lista de todos os estados com seus

respectivos predicados, como ilustrado no arquivo *predicados.txt*.

Figura 13 – Exemplo do arquivo de entrada `policy.out`

```
If holds: goal()
Execute: goal / SC / d=0
If holds: can-finish-service()/not(goal())
Execute: finish-service  / SC / d=1
If holds: not(can-go-error-treatment())
/can-show-ask-user-grade()/not(goal())
Execute: ask-user-grade  / SC / d=2
```

Fonte: Elaborada pelo Autor

Com os predicados identificados, o próximo passo foi integrar essas informações aos dados gerados no plano (parte (a)), que continham os estados e ações de cada *plan trace*. Para isso, foi desenvolvido um programa *Python* que recebe como entrada tanto o arquivo de estados e ações *estados-acao.txt* quanto o arquivo de predicados *predicados.txt*. Esse algoritmo foi responsável por adicionar os predicados correspondentes a cada estado de cada *plan trace*.

O programa, denominado `AdicionarPredicados.py`, começa lendo as informações dos arquivos de entrada. Ele percorre as linhas do arquivo *estados-acao.txt* para saber cada pré-estado e cada pós-estado envolvidos nas ações dos planos. Para cada um desses estados, o programa consulta o arquivo *predicados.txt* para construir a lista de predicados para cada um dos estados.

A saída é um conjunto de *plan traces* em que cada *plan trace* contém a parte (a) (plano com nomes de pré-estado, pós-estado e ação) e a parte (b) (lista de estados - contendo o nome dos estados envolvidos na parte (a) seguido dos predicados que descrevem cada estado).

4.3 Avaliação dos *plan traces* obtidos

A avaliação dos *plan traces* gerados a partir do grafo da política gerenciadora do *chatbot Plantão Coronavírus* será realizada comparativamente aos *plan traces* gerados a partir dos diálogos em linguagem natural (Silva, 2024). O objetivo é realizar uma análise quantitativa e qualitativa dos *plan traces* produzidos pelas duas abordagens.

Na análise quantitativa, consideramos a quantidade de *plan traces* gerados pelas duas abordagens, identificando quais *plan traces* são comuns às duas abordagens e quais são

exclusivos de uma delas. Na análise qualitativa, consideramos o quão completa é a descrição dos estados (parte (b) dos *plan traces*) para cada um dos estados contidos nos *plan traces*.

5 OBTENÇÃO DE DADOS ESTRUTURADOS (PLAN TRACES) PARA AQUISIÇÃO AUTOMÁTICA DO DOMÍNIO DE PLANEJAMENTO GERENCIADOR DO CHATBOT PLANTÃO CORONAVÍRUS

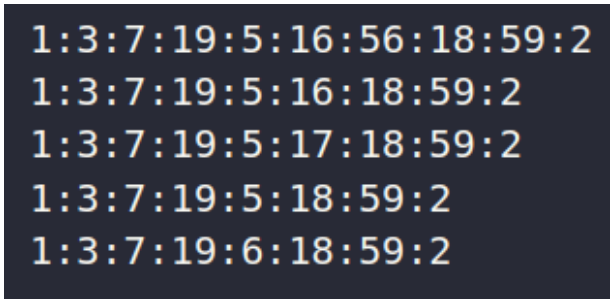
Nesta seção são apresentados os resultados obtidos de cada uma das fases da metodologia deste trabalho. As implementações foram feitas em uma máquina equipada com um processador Ryzen 5 3500U CPU 2.1GHz, 12GB de memória RAM, e 512GB de disco, rodando em um sistema operacional Ubuntu 24.04 LTS. Esta configuração está relacionada a todas as fases de extração e manipulação dos dados referentes a este trabalho.

5.1 Obtenção da parte (a) (plano) de cada *plan trace*

Para a obtenção da parte (a) de cada *plan trace*, utilizamos os dados da política contidos no arquivo "policy_graph.dot". Esse arquivo forneceu uma representação em grafo direcionado da política do chatbot, onde cada nó correspondia a um estado e cada aresta representava uma ação de transição entre estados.

Utilizando uma busca mo espaço de estados da política, foram explorados todos os possíveis caminhos do grafo, do estado inicial até a meta, ignorando *loops*. A decisão de não considerar *loops* foi tomada para simplificar a análise, focando nos caminhos ideais previstos pela política. Essa abordagem permitiu identificar todas as sequências de estados que levam o agente ao seu objetivo, resultando em 457 caminhos. Como mostrado na Figura 14, cada caminho mostra a trajetória completa do estado inicial até a meta.

Figura 14 – Exemplo de caminhos possíveis.



Fonte: Elaborado pelo Autor

Cada um dos **457 caminhos** identificados foi então convertido em um *plan trace*. Esses *plan traces* foram representados como pares de estados e a ação de transição entre eles, detalhando cada passo do agente ao longo do trajeto. A Figura 15 ilustra essa representação, onde cada transição entre pares de estados é descrita pela ação correspondente.

Figura 15 – Exemplo de estados com ação de transição.

```
Plan trace 457
1:3:show-welcome-message
3:7:start-dialog
7:19:show-info-others
19:6:start-dialog
6:18:show-info-mental-health
18:59:ask-user-grade
59:2:finish-service
```

Fonte: Elaborado pelo Autor

5.2 Obtenção dos predicados que descrevem cada estado da política

Para a obtenção dos predicados, foram analisados em detalhes um total de **168 estados** presentes na política. Cada estado foi descrito com base em seus respectivos predicados, vindos do aquivo "policy.out", fornecendo informações sobre as condições e características que definem cada etapa do diálogo do agente. Na Figura 16 é exemplificado como os estados foram separados e tiveram seus predicados mapeados. Os predicados foram fundamentais para entender o contexto de cada estado e a lógica por trás das transições entre eles.

Figura 16 – Exemplo de predicados em cada estado.

```
1: user-initiative()/not(can-go-error-treatment())/not(goal())

2: goal()

3: started()/not(can-go-error-treatment())/can-do-start-dialog()/not(goal
())

4: not(have-patient-days-symptoms())/not(have-patient-symptoms())/not
(have-diagnostic-system-positive())/can-start-online-service()/not
(can-go-error-treatment())/not(goal())
```

Fonte: Elaborado pelo Autor

Cada *plan trace* obtido na parte (a) foi então associado aos estados analisados. Para cada *plan trace*, foi identificado quais dos 168 estados faziam parte do caminho do estado inicial à meta. Essa análise permitiu uma visão detalhada não apenas das ações que o agente realiza, mas também dos estados que ele atravessa, enriquecendo a compreensão do comportamento do agente ao longo dos diálogos. Ao final foi possível ter 457 *plan traces* completos, determinado os pares de estados, a ação responsável pela transição entre os estados e os predicados correspondentes a

cada estado. Podem ser acessados em: https://github.com/fernanddo-s/plan-traces-chatbot-covid

5.3 Avaliação dos *plan traces* obtidos

Para avaliar os *plan traces* obtidos a partir da política utilizada para gerenciar os diálogos (dados estruturados), realizamos um comparativo com os *plan traces* obtidos por (Silva, 2024) a partir dos dados de diálogo (não estruturados). O objetivo foi mapear quais caminhos da política os diálogos reais percorrem, a fim de validar se a política gerenciadora do *chatbot* contempla esses diálogos. Essa validação é importante, já que o domínio utilizado para gerar a política foi construído manualmente com base em apenas cerca de 30 diálogos reais (Pinho, 2024), contendo 89 predicados e 60 ações.

Assim, para realizar esse mapeamento foram utilizados *scripts* desenvolvidos na linguagem *Python* com o objetivo de automatizar o processo. Os arquivos de texto resultantes em ambas as abordagens foram normalizados para possibilitar a comparação automática.

Em relação à quantidade de *plan traces*, foram **457** *plan traces* completos obtidos por este trabalho versus 1125 *plan traces* do trabalho de (Silva, 2024). Por outro lado levando em consideração o quesito qualitativo, os *plan traces* deste trabalho são completos contendo a descrição de estados e a sequência de ações que levam o agente de um estado inicial do diálogo para a meta, como pode ser observado na Figura 17. Enquanto os *plan traces* vindos dos diálogos não possuem os detalhes de descrição de estados por meio de predicados, levando em consideração as ações correspondentes partindo do estado inicial até a meta, como pode ser observado na Figura 18.

Figura 17 – Exemplo de *plan trace* completo

```
Plan trace 457
1:3:show-welcome-message
3:7:start-dialog
7:19:show-info-others
19:6:start-dialog
6:18:show-info-mental-health
18:59:ask-user-grade
59:2:finish-service

1: user-initiative()/not(can-go-error-treatment())/not(goal())
2: goal()
3: started()/not(can-go-error-treatment())/can-do-start-dialog()/not(goal())
6: started()/can-show-info-mental-health()/not(can-go-error-treatment())/not(can-back-dialog())/not(goal())
7: can-show-info-others()/not(can-go-error-treatment())/not(goal())
18: not(can-go-error-treatment())/can-show-ask-user-grade()/not(goal())
19: started()/not(can-go-error-treatment())/can-do-start-dialog()/not(goal())
59: can-finish-service()/not(goal())
```

Fonte: Elaborado pelo Autor

Figura 18 – Exemplo de caminho do diálogo completo

```
Caminho do diálogo 1
show-welcome-message
start-dialog
start-online-service
ask-patient-symptons
ask-patient-how-many-days-symptoms
show-info-diagnostic-positive
ask-share-location
ask-postal-code
show-info-calling-health-profitional
ask-patient-info-cpf
ask-patient-info-name
ask-patient-info-birthday
error-treatment-patient-info-birthday
ask-patient-comorbidities
ask-patient-info-gender
ask-patient-info-phone-number
call-health-agent
health-agent-takes-control
finish-service
ask-user-grade
show-info-about-follow-program
ask-user-check
```

Fonte: (Silva, 2024)

Com relação aos 1125 diálogos analisados é possível observar que, 779 diálogos não correspondem a um caminho completo na política, saindo do estado inicial até a meta. Esta parcela de diálogos caracteriza-se por incompletudes durante a interação, como conversas que foram interrompidas Figura 19.

Em 474 diálogos, os usuários cometeram algum tipo de erro na inserção das informações (diálogos com *loop*), que por sua vez caracterizam o não-determinismo das ações. No entanto, são *plan traces* completos que levam o agente do estado inicial para a meta.

Além disso, o comparativo com os *plan traces* provenientes dos diálogos reais permitiu encontrar **21 novas ações** que não estavam mapeadas na política original. Estas ações correspondem à transições de estados que devem ser inseridas no grafo da política.

Fazendo uma comparação direta entre os *plan traces*, foi possível observar ainda que 50 diálogos ocorreram sem erros de inserção do usuário, percorreram 2 caminhos da política Figura 20 e Figura 22 e seus respectivos *plan traces* que podem ser acompanhados na Figura 21

Figura 19 – Exemplo de caminho do diálogo incompleto

```
Caminho do diálogo 7
show-welcome-message
start-dialog
start-online-service-v1
start-online-service-v2
ask-patient-symptoms-v1
ask-patient-symptons-v3
ask-patient-how-many-days-symptoms
show-info-diagnostic-positive-v2
ask-share-location-v3
ask-postal-code
error-treatment-postal-code
error-treatment-postal-code
error-treatment-postal-code
error-treatment-postal-code
ask-patient-info-state
ask-patient-info-city
show-info-calling-health-profitional-v1
ask-patient-info-cpf
```

Fonte: (Silva, 2024)

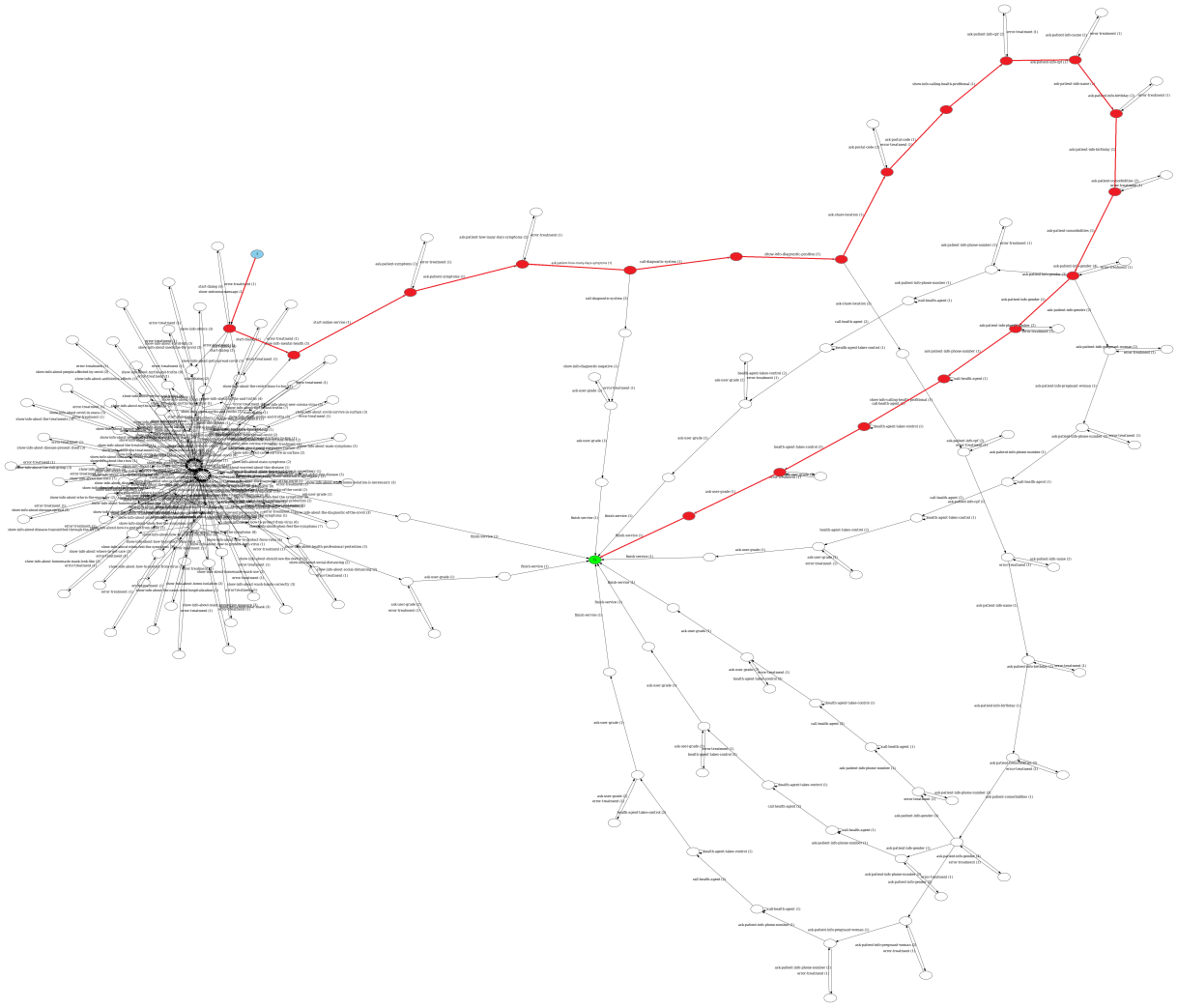
e Figura 23.

O caminho 1 mostrado na Figura 21, representa um usuário homem que deseja realizar uma consulta online. Ele interage com o chatbot, fornece as informações necessárias, incluindo sintomas, CPF, CEP, entre outras, e é transferido para um agente de saúde humano. Após o atendimento, ele avalia o serviço e por fim o serviço é finalizado.

O caminho 2 mostrado na Figura 23, representa uma usuária mulher que deseja realizar uma consulta online. Ela interage com o chatbot, fornece as informações, como data de nascimento, número de telefone. Ela é então transferida para um agente de saúde humano, e ao final do atendimento, avalia o serviço e por fim o serviço é finalizado. ao informar o gênero o usuário pode ter uma ação adicional caso seja mulher, uma vez que a ação responsável por verificar se essa mulher etá gravida, o que não é usado caso o usuário seja homem, como pode ser visto na diferença no fluxo das Figuras 20 e 22.

Esses resultados indicam que a política gerenciadora do *chatbot* é capaz de representar uma parte significativa dos diálogos reais, cobrindo casos em que os usuários seguem o fluxo corretamente, assim como aqueles em que erros ocorrem durante a interação.

Figura 20 – Exemplo de caminho da política



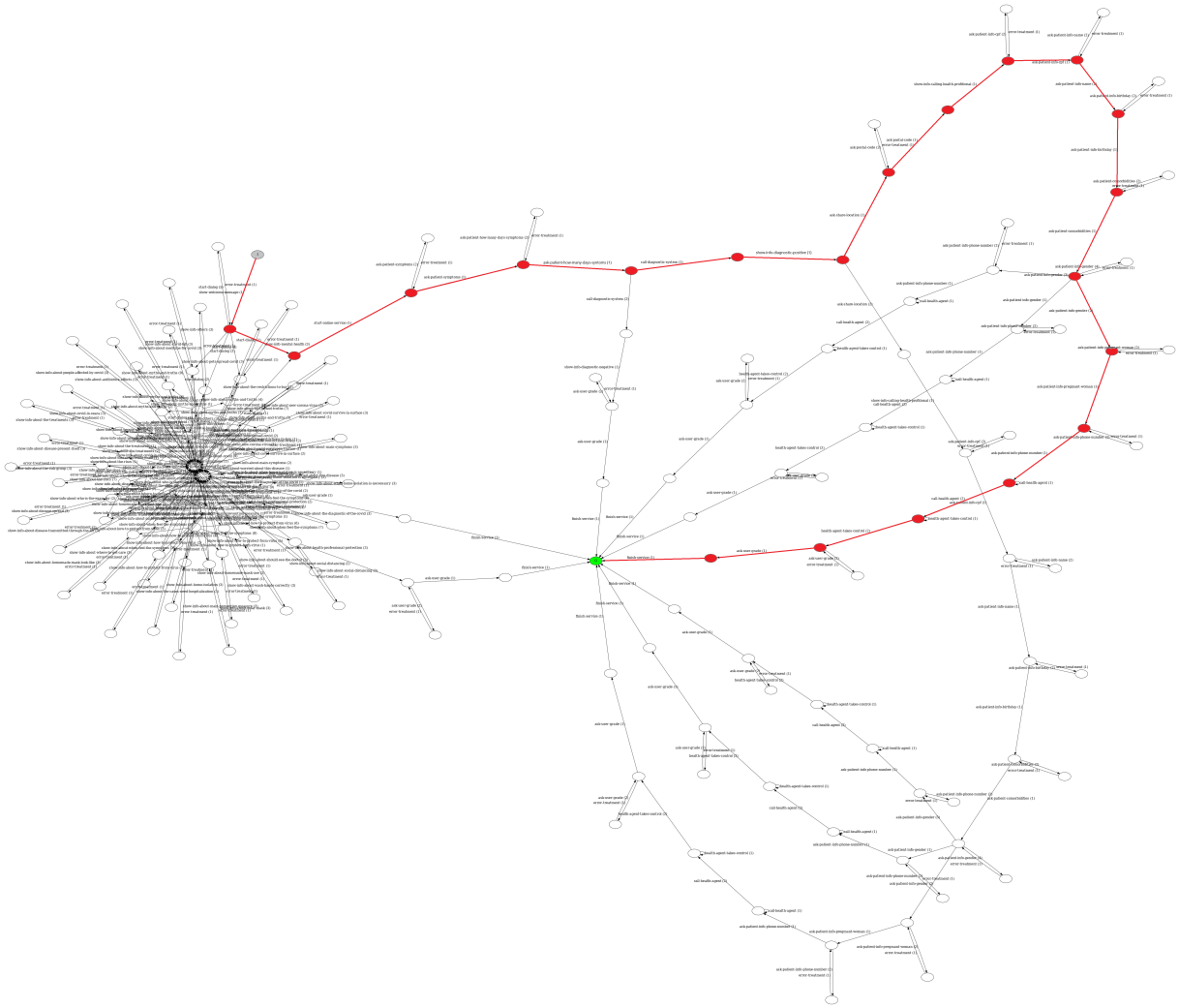
Fonte: Adpatada de (Pinho, 2024)

Figura 21 – Exemplo de *plan traces* do caminho da política

```
Plan trace 1
1:3:show-welcome-message
3:4:start-dialog
4:9:start-online-service
9:23:ask-patient-symptoms
23:62:ask-patient-how-many-days-symptoms
62:91:call-diagnostic-system
91:95:show-info-diagnostic-positive
95:97:ask-share-location
97:101:ask-postal-code
101:104:show-info-calling-health-profitional
104:107:ask-patient-info-cpf
107:111:ask-patient-info-name
111:115:ask-patient-info-birthday
115:119:ask-patient-comorbidities
119:125:ask-patient-info-gender
125:135:ask-patient-info-phone-number
135:145:call-health-agent
145:152:health-agent-takes-control
152:160:ask-user-grade
160:2:finish-service
```

Fonte: Elaborada pelo Autor

Figura 22 – Exemplo de caminho da política



Fonte: Adpatada de (Pinho, 2024)

Figura 23 – Exemplo de caminho da política

```
Plan trace 2
1:3:show-welcome-message
3:4:start-dialog
4:9:start-online-service
9:23:ask-patient-symptoms
23:62:ask-patient-how-many-days-symptoms
62:91:call-diagnostic-system
91:95:show-info-diagnostic-positive
95:97:ask-share-location
97:101:ask-postal-code
101:104:show-info-calling-health-profitional
104:107:ask-patient-info-cpf
107:111:ask-patient-info-name
111:115:ask-patient-info-birthday
115:119:ask-patient-comorbidities
119:126:ask-patient-info-gender
126:137:ask-patient-info-pregmant-woman
137:146:ask-patient-info-phone-number
146:153:call-health-agent
153:162:health-agent-takes-control
162:167:ask-user-grade
167:2:finish-service
```

Fonte: Elaborada pelo Autor

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho mostrou como gerar *plan traces* para a aquisição automática de domínios de planejamento, a partir da política gerenciadora do *chatbot* Plantão Coronavírus. O trabalho de (Pinho, 2024) efetuou a construção manual do domínio de planejamento com base em um conjunto limitado de diálogos reais (cerca de 30 diálogos) e utilizou o planejador PRP para obter automaticamente a política de gerenciamento do diálogo. No entanto, até o momento, não se sabia o quão representativa essa política era em relação à quantidade total de diálogos efetuados no sistema real (cerca de 7.136 diálogos).

Nesta pesquisa, compararam-se 1125 *plan traces* extraídos de diálogos reais produzidos por (Silva, 2024) com 457 *plan traces* completos (parte (a) e parte (b)) obtidos a partir da política elaborada manualmente por (Pinho, 2024). Os resultados indicaram que a política modelada representa de forma satisfatória os diálogos reais, cobrindo quase a totalidade dos cenários analisados. O comparativo permitiu ainda verificar que algumas ações presentes nos diálogos reais não estavam representadas na política, sendo necessário a adição de novas arestas no grafo da política. Assim as principais contribuições, deste trabalho são:

- A elaboração de uma base de dados formada por 457 *plan traces* completos, que poderão servir para efetuar a aquisição automática de domínios de planejamento com ferramentas como *PlanMinner* (Segura-Muros *et al.*, 2021).
- Uma análise de cobertura que comprovou a abrangência da política em relação aos diálogos reais, assegurando que o modelo manual de gerenciamento de diálogos representa uma boa parte das interações reais do sistema.
- A constatação que os diálogos reais contém algumas ações que não estão representadas na política, tendo esta que ser incrementada por novas transições de estados.

Para trabalhos futuros, propõe-se a integração dos *plan traces* produzidos neste estudo com aqueles gerados automaticamente a partir dos diálogos de (Silva, 2024). Essa integração formará uma base de dados apropriada para a aquisição automática do domínio de planejamento, que posteriormente será comparado ao domínio adquirido manualmente (https://github.com/brunopinho321/Domain-Chatbot-Covid).

REFERÊNCIAS

ARORA, A.; FIORINO, H.; PELLIER, D.; MÉTIVIER, M.; PESTY, S. A review of learning planning action models. **The Knowledge Engineering Review**, Cambridge University Press, v. 33, p. e20, 2018.

Cardoso. **Plantão Coronavirus: Governo do Ceará lança canal de Whatsapp para atender a população**. 2020. Disponível em: https://www.saude.ce.gov.br/2020/04/13/plantao-coronavirus-governo-do-ceara-lanca-canal-de-whatsapp-para-atender-a-populacao/. Acesso em: 09 de out 2023.

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

DEEP-DIAL. **Reasoning and Learning for Human-Machine Dialogue**. 2018. Disponível em: https://sites.google.com/view/deep-dial-2019/deep-dial-2018. Acesso em: 05 de setembro 2023.

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

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

HASLUM, P.; LIPOVETZKY, N.; MAGAZZENI, D.; MUISE, C.; BRACHMAN, R.; ROSSI, F.; STONE, P. **An introduction to the planning domain definition language**. [*S. l.*]: Springer, 2019. v. 13.

ICAPS. **International Conference on Automated Planning and Scheduling**. 2023. Disponível em: https://www.icaps-conference.org/.

KOTSIANTIS, S. B.; ZAHARAKIS, I. D.; PINTELAS, P. E. Machine learning: a review of classification and combining techniques. **Artificial Intelligence Review**, Springer, v. 26, p. 159–190, 2006.

KUČERA, J.; BARTÁK, R. Louga: learning planning operators using genetic algorithms. In: SPRINGER. **Knowledge Management and Acquisition for Intelligent Systems: 15th Pacific Rim Knowledge Acquisition Workshop, PKAW 2018, Nanjing, China, August 28-29, 2018, Proceedings 15**. [*S. l.*], 2018. p. 124–138.

MANLY, B. F.; ALBERTO, J. A. N. **Multivariate statistical methods: a primer**. [*S. l.*]: Chapman and Hall/CRC, 2016.

MCDERMOTT, D.; GHALLAB, M.; HOWE, A.; KNOBLOCK, C.; RAM, A.; VELOSO, M.; WELD, D.; WILKINS, D. Pddl-the planning domain definition language. New Haven, CT, 1998.

MIGLANI, S.; YORKE-SMITH, N. Nltopddl: One-shot learning of pddl models from natural language process manuals. In: **Proc. of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS). ICAPS**. [*S. l.*: *s. n.*], 2020.

Mont'alvarine. **Plantão Coronavírus é importante ferramenta no combate à Covid-19 e na assistência à saúde**. 2021. Disponível em: https://www.ceara.gov.br/2021/02/18/plantao-coronavirus-e-importante-ferramenta-no-combate-a-covid-19-e-na-assistencia-a-saude/. Acesso em: 06 de out 2023.

MUISE, C.; CHAKRABORTI, T.; AGARWAL, S.; BAJGAR, O.; CHAUDHARY, A.; LASTRAS-MONTANO, L. A.; ONDREJ, J.; VODOLAN, M.; WIECHA, C. Planning for goal-oriented dialogue systems. **arXiv preprint arXiv:1910.08137**, 2019.

MUISE, C.; MCILRAITH, S.; BECK, C. Improved non-deterministic planning by exploiting state relevance. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [*S. l.*: *s. n.*], 2012. v. 22, p. 172–180.

MUISE, C. J.; BECK, J. C.; MCILRAITH, S. A. Flexible execution of partial order plans with temporal constraints. In: **IJCAI**. [*S. l.*: *s. n.*], 2013. p. 2328–2335.

PEREIRA, S. do L.; BARROS, L. N. de. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. Tese (Doutorado) – Tese de Doutorado, Universidade de Sao Paulo, Instituto de Matemática e . . . , 2007.

PINHO, B. da S. **Planejamento não-determinístico para o gerenciamento de agentes de diálogos orientados à meta**. 2024.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence a modern approach**. [*S. l.*]: London, 2010.

SANNER, S. *et al.* Relational dynamic influence diagram language (rddl): Language description. **Unpublished ms. Australian National University**, v. 32, p. 27, 2010.

SANTOS, V. B. d.; BARROS, L. N. d.; PEREIRA, S. d. L.; MENEZES, M. V. d. Symbolic fond planning for temporally extended goals. In: **Workshop**. [*S. l.*: *s. n.*], 2022.

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

SEGURA-MUROS, J. Á.; PÉREZ, R.; FERNÁNDEZ-OLIVARES, J. Discovering relational and numerical expressions from plan traces for learning action models. **Applied Intelligence**, Springer, v. 51, n. 11, p. 7973–7989, 2021.

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

SILVA, B. P. da. **OBTENÇÃO DE DADOS ESTRUTURADOS A PARTIR DE DIÁLOGOS PARA AQUISIÇÃO AUTOMÁTICA DO DOMÍNIO DE PLANEJAMENTO GERENCIADOR DO CHATBOT PLANTÃO CORONAVÍRUS**. 2024.

TEDE, P. C. d. A. R.; BARROS, F. de A. Agentes inteligentes conversacionais: conceitos básicos e desenvolvimento. **Sociedade Brasileira de Computação**, 2016.

VALLATI, M.; CHRPA, L.; GRZEŚ, M.; MCCLUSKEY, T. L.; ROBERTS, M.; SANNER, S. *et al.* The 2014 international planning competition: Progress and trends. **Ai Magazine**, v. 36, n. 3, p. 90–98, 2015.

ZHUO, H. H.; KAMBHAMPATI, S. Action-model acquisition from noisy plan traces. In: **Twenty-Third International Joint Conference on Artificial Intelligence**. [*S. l.*: *s. n.*], 2013.

ZHUO, H. H.; PENG, J.; KAMBHAMPATI, S. Learning action models from disordered and noisy plan traces. **arXiv preprint arXiv:1908.09800**, 2019.