



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
MESTRADO ACADÊMICO EM COMPUTAÇÃO

BRUNO DA SILVA PINHO

PLANEJAMENTO NÃO-DETERMINÍSTICO PARA O GERENCIAMENTO DO
AGENTE DE DIÁLOGO PLANTÃO CORONAVÍRUS

QUIXADÁ

2024

BRUNO DA SILVA PINHO

PLANEJAMENTO NÃO-DETERMINÍSTICO PARA O GERENCIAMENTO DO AGENTE
DE DIÁLOGO PLANTÃO CORONAVÍRUS

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

Orientadora: Profa. Dra. Maria Viviane de Menezes

QUIXADÁ

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)

- P723 Pinho, Bruno da Silva.
Planejamento não-determinístico para o gerenciamento do agente de diálogo plantão coronavírus. /
Bruno da Silva Pinho. – 2024.
85 f. : il. color.
- Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2024.
Orientação: Profa. Dra. Maria Viviane de Menezes.
1. Coronavírus. 2. Planejamento Automatizado. 3. Algoritmo não determinístico. I. Título.
CDD 005
-

BRUNO DA SILVA PINHO

PLANEJAMENTO NÃO-DETERMINÍSTICO PARA O GERENCIAMENTO DO AGENTE
DE DIÁLOGO PLANTÃO CORONAVÍRUS

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

Aprovada em: ____/____/____.

BANCA EXAMINADORA

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

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

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

RESUMO

Conversação natural é um aspecto fundamental do comportamento inteligente e, por essa razão, os *agentes de diálogo* têm sido por décadas um importante tópico de pesquisa em Inteligência Artificial (IA). Os *chatbots* são implementações destes sistemas e foram amplamente adotados, particularmente na recente pandemia de COVID-19. Como exemplo desta ampla adoção, temos o agente de diálogo *Plantão Coronavírus* que foi utilizado pelo Governo do Estado do Ceará para auxiliar no atendimento a pacientes durante os momentos críticos da pandemia. Em geral, as tarefas executadas por um agente de diálogo consistem em: (i) compreender os elementos da entrada do usuário (ii) a partir destes elementos, *selecionar uma ação a ser tomada* e; (iii) a partir dessa ação, gerar uma resposta ao usuário. Embora existam métodos bem estabelecidos para a compreensão da linguagem natural, como algoritmos de aprendizado de máquina que utilizam raciocínio estatístico, o gerenciamento do diálogo apresenta desafios para sistemas que adotam esse tipo de raciocínio, uma vez que não garantem a confiabilidade das respostas nem fornecem explicações sobre o processo de raciocínio. Esses requisitos, no entanto, são fundamentais para sistemas críticos, como os da área da saúde. Dessa forma, tem sido explorado nos últimos anos o uso de métodos de raciocínio simbólico bem consolidados na IA para a tarefa de gerenciamento do diálogo, como Planejamento Automatizado que é uma abordagem de *raciocínio sobre ações*. Em Planejamento, as ações do agente podem ser determinísticas ou terem efeitos incertos, ações não-determinísticas. Este trabalho apresenta um gerenciador de diálogos para o Plantão Coronavírus baseado na abordagem de Planejamento Automatizado com Ações Não-Determinísticas, o que reflete as diversas possibilidades de respostas do usuário. As principais contribuições deste trabalho são: (i) o processo de aquisição do domínio de planejamento com ações não-determinísticas, baseado nos dados de conversas entre pacientes e *bot*; (ii) a arquitetura do sistema que combina aprendizado de máquina para as tarefas de compreensão da linguagem natural e planejamento automatizado para as tarefas de gerenciamento do diálogo; (iii) a implementação da arquitetura proposta utilizando o Rasa como sistema de conversação e um planejador não-determinístico estado da arte para elaboração automática dos possíveis caminhos de diálogos e (iv) a avaliação conduzida simulando se as soluções obtidas pelo gerenciador poderiam efetivamente conduzir os diálogos que ocorreram no sistema.

Palavras-chave: coronavírus; planejamento automatizado; algoritmo não determinístico.

ABSTRACT

Natural conversation is a fundamental aspect of intelligent behavior, and for this reason, dialogue agents have been an important research topic in Artificial Intelligence (AI) for decades. Chatbots are implementations of these systems and have been widely adopted, particularly during the recent COVID-19 pandemic. As an example, we have *Plantão Coronavírus* dialogue agent, which was used by the Government of the State of Ceará to assist in patient care during critical moments of the pandemic. In general, the tasks performed by a dialogue agent are: (i) understanding the elements of the user input, (ii) based on these elements, selecting an action to be taken, and (iii) generating a response to the user based on this action. Although there are well-established methods for natural language understanding, based on statistical reasoning, such as machine learning algorithms, dialogue management poses challenges for this kind of reasoning, once they do not guarantee the reliability of responses or provide explanations about the reasoning process. These requirements, however, are very important for critical systems, such as those related to healthcare. Therefore, the use of well-established symbolic reasoning methods in AI has been explored in recent years for dialogue management tasks, such as Automated Planning, which is an approach to reasoning about actions. In Planning, the agent's actions can be deterministic or have uncertain effects, i.e., non-deterministic actions. This work presents a dialogue manager for *Plantão Coronavírus* based on the approach of Automated Planning with Non-Deterministic Actions, which models the different possibilities of user responses. The main contributions of this work are: (i) the acquisition of the non-deterministic planning domain, based on dialogues between patients and the bot; (ii) the system architecture that combines machine learning for natural language understanding tasks and automated planning for dialogue management tasks; (iii) the implementation of the proposed architecture using Rasa as a conversational system and a state-of-the-art non-deterministic planner for automatic generation of possible dialogue paths; and (iv) the evaluation conducted by simulating whether the solutions obtained by the manager could effectively conduct the dialogues that occurred in the system.

Keywords: coronavirus; automated planning; non-deterministic algorithm.

LISTA DE FIGURAS

Figura 1 – Arquitetura básica de um Sistema de Diálogo	17
Figura 2 – Exemplo de um módulo domain da ferramenta Rasa	22
Figura 3 – Exemplo definição de intenção no módulo nlu do Rasa.	23
Figura 4 – Exemplo de ação customizada criada no Rasa.	23
Figura 5 – Modelagem de caminhos que um usuário pode seguir em um fluxo do diálogo.	24
Figura 6 – Esquema de um Planejador.	27
Figura 7 – Aplicando ação a <i>ask-determination</i> ao estado s_0	29
Figura 8 – Esquema de um Planejador.	33
Figura 9 – Esquema de um Planejador.	33
Figura 10 – Esquema de um Planejador.	34
Figura 11 – Uma ação em PDDL para verificar um tipo de consulta	38
Figura 12 – Plano gerado para uma consulta	39
Figura 13 – Exemplo de consulta do usuário que não foi encontrada.	39
Figura 14 – Exemplo de consulta do usuário que não foi encontrada.	39
Figura 15 – Ação Praise utilizada para elogiar o usuário.	41
Figura 16 – Arquitetura proposta para o Sistema de Diálogo, utilizando o Rasa como SC e o Planejador como GD.	43
Figura 17 – Exemplo de pré-condição de uma ação no domínio gerenciador do chatbot Plantão Coronavírus.	47
Figura 18 – Exemplo de efeitos de uma ação.	48
Figura 19 – Exemplo de adição de predicados do usuário nas pré-condições e efeitos das ações.	49
Figura 20 – Exemplo de ação não-determinística <i>ask-*</i>	52
Figura 21 – Exemplo de ação não-determinística <i>show-*</i>	53
Figura 22 – Exemplo de ação determinística <i>error-*</i>	53
Figura 23 – Exemplo de ação <i>call-*</i> não-determinística do domínio da COVID.	54
Figura 24 – Exemplo de ação <i>finish-service</i>	55
Figura 25 – Exemplo de ação <i>health-agent-takes-control</i>	55
Figura 26 – Exemplo de ação não-determinística <i>start-dialog</i>	56
Figura 27 – Exemplo de ação determinística <i>start-online-service</i>	57
Figura 28 – Arquitetura do Gerenciador de Diálogo proposto neste trabalho.	58

Figura 29 – Mapeamento da variáveis de contexto e variáveis de estado.	59
Figura 30 – Escolha de uma ação utilizando o estado e a política	59
Figura 31 – Escolha de uma ação utilizando o estado e a política	60
Figura 32 – Implementação do modulo domain para o chatbot do Plantão Coronavírus .	67
Figura 33 – Exemplo ilustrativo de implementação no módulo nlu	68
Figura 34 – Exemplo de implementação no módulo stories	68
Figura 35 – Implementação da função name.	69
Figura 36 – Implementação da função action_text	69
Figura 37 – Implementação da função run.	70
Figura 38 – Grafo da Politica	81
Figura 39 – Parte inicial da Politica	82
Figura 40 – Estado de aceitação da Politica	82
Figura 41 – Simulação 1	83
Figura 42 – Simulação 2	84
Figura 43 – Simulação 3	85

LISTA DE TABELAS

Tabela 1 – Comparação entre os trabalhos relacionados.	41
Tabela 2 – Mapeamento de declarações do usuário para predicados.	44
Tabela 3 – Mapeamento de textos do <i>chatbot</i> para ações.	45
Tabela 4 – Fluxo de ações e predicados do usuário	46
Tabela 5 – Mapeamento de pré-condições e efeitos de um fluxo de diálogo.	49
Tabela 6 – Resultado das simulações dos cenários de teste usando a política de gerenciamento obtida.	75

LISTA DE ABREVIATURAS E SIGLAS

SC Serviço de Conversação

GD Gerenciador de Diálogo

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Objetivos	14
1.2	Organização do texto	15
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	Sistemas de Diálogos	16
2.1.1	<i>Watson Assistant</i>	20
2.1.2	<i>Dialogflow</i>	20
2.1.3	<i>Amazon Lex</i>	21
2.1.4	<i>Microsoft Bot</i>	21
2.1.5	<i>RASA</i>	22
2.2	O Chatbot Plantão Coronavírus	24
2.3	Planejamento Automatizado	27
2.3.1	<i>Planejamento Não-Determinístico</i>	30
2.3.2	<i>Paradigmas de Aquisição de Domínio de Planejamento</i>	32
3	TRABALHOS CORRELATOS	35
3.1	Automating the Generation of Goal-Oriented Dialogue Managers for Healthcare	35
3.2	Planning for Goal-Oriented Dialogue Systems	36
3.3	A Generic Dialog Agent for Information Retrieval Based on Automated Planning within a Reinforcement Learning Platform	37
3.4	Cognitive-Inspired Conversational-Strategy Reasoner for Socially-Aware Agents	40
3.5	Comparação entre os trabalhos	41
4	GERENCIAMENTO DO CHATBOT PLANTÃO CORONAVÍRUS USANDO PLANEJAMENTO AUTOMATIZADO	42
4.1	Aquisição do Domínio de Planejamento para Gerenciamento do ChatBot Plantão Coronavírus	43
4.1.1	<i>Descrição do Dataset de Conversação do ChatBot Plantão Coronavírus</i>	43
4.1.2	<i>Obtenção dos Predicados e Ações do Domínio a partir dos Dados Textuais de Diálogo</i>	44

4.1.3	<i>O Domínio de Planejamento Gerenciador do ChatBot da COVID</i>	50
4.2	Arquitetura do Gerenciador do Diálogo baseado em Planejamento Automatizado	57
4.2.1	<i>Mapeamento de Variáveis de Estado e Variáveis de Contexto</i>	58
4.2.2	<i>Execução da Ação</i>	59
4.2.3	<i>Monitoramento da Execução da Política</i>	60
4.3	Implementação	61
4.3.1	<i>Implementação do Gerenciador de Diálogo (GD)</i>	61
4.3.2	<i>Integração do Gerenciador de Diálogos ao Sistema de Conversação Rasa</i> .	66
5	AVALIAÇÃO DO GERENCIADOR DE DIÁLOGO	71
5.1	Construção automática da árvore de diálogo	71
5.2	Gerenciamento do Diálogo Utilizando a Política Obtida	72
5.2.1	<i>Cenários de Teste</i>	72
5.2.2	<i>Resultados das simulações</i>	74
6	CONCLUSÕES E TRABALHOS FUTUROS	76
	REFERÊNCIAS	78
	APÊNDICES	81
	APÊNDICE A – GRAFO DA POLITICA GERADA PELO PLANEJADOR PRP	81

1 INTRODUÇÃO

Conversação natural é um aspecto fundamental do comportamento inteligente e, por essa razão, os *agentes de diálogo*, também conhecidos como *chatbots*, têm sido por décadas um importante tópico de pesquisa em Inteligência Artificial (IA) (Grice, 1975; McTear, 1987; Weizenbaum, 1966; McTear, 2020; Mamatha *et al.*, 2021; IBM, 2022; Amazon, 2022a; Google, 2022a; Microsoft, 2022; Apple, 2022; Rasa, 2022). Os *chatbots* são a aplicação mais recente destes sistemas e foram amplamente adotados, particularmente na recente pandemia de COVID-19. Os diálogos podem apresentar-se em sua forma mais simples - conhecidos como diálogos de uma-pergunta-uma-resposta, tais como: perguntas com respostas prontas (e.g, ‘como utilizo esta ferramenta?’) ou bate-papos (e.g, ‘olá!’) - até sua forma mais sofisticada, denominados **diálogos com múltiplas iterações e orientados à meta** (Botea *et al.*, 2019), presentes, por exemplo, em sistemas de e-commerce, planejamento de viagens, orientação de carreira e telemedicina.

A geração automática de diálogos com múltiplas iterações e orientados à meta é uma tarefa complexa, apresentando desafios em pesquisa científica e grande interesse da indústria e governos (Muisse *et al.*, 2019; McTear, 2020). Do ponto de vista da pesquisa, o uso de métodos de aprendizado de máquina está bem consolidado para as tarefas de compreensão da linguagem (DEEP-DIAL, 2018). No entanto, a tarefa de gerenciamento do diálogo ainda é um problema em aberto, uma vez que não é resolvido de forma satisfatória com as técnicas utilizadas atualmente (DEEP-DIAL, 2018), as quais são: *aprendizagem de máquina* ou *árvores de diálogos* (Muisse *et al.*, 2019). O gerenciamento do diálogo baseado em algoritmos de aprendizagem de máquina depara-se com os problemas de como *fornecer explicações* sobre o motivo pelo qual uma determinada resposta foi dada ao usuário e como gerar e manter dados de treinamento para esses sistemas. Já o gerenciamento baseado em árvores de análise é um gerenciamento baseado em regras (Weizenbaum, 1966; Ljunglöf, 2009; Rich; Sidner, 2012) no qual o *projetista do diálogo* deve descrever manualmente todos os possíveis caminhos de um diálogo, tentando associar uma possível entrada do usuário a uma ação do agente. Como o usuário pode dar entradas diferentes em uma conversa, geralmente estas árvores possuem muitas ramificações e, quanto mais complexo o diálogo for, maior e mais complexa é a tarefa de construção desta árvore. Assim, há uma necessidade de avaliar a utilização de **métodos baseados em raciocínio**, bem consolidados na Inteligência Artificial, em conjunto com métodos que utilizam aprendizagem de máquina na construção de sistemas de diálogos projetados para solucionar problemas em conjunto com humanos de uma maneira que possam (DEEP-DIAL, 2018): (i) explicar seu

raciocínio; (ii) ajudar o usuário a decidir sobre escolhas entre alternativas e; (iii) enfrentar os padrões éticos exigidos em ambientes da vida real.

Atualmente, as principais ferramentas de construção de agentes de diálogo comerciais são: *Watson Assistant* (IBM, 2022), *Dialogflow* (Google, 2022a), *Lex* (Amazon, 2022a) e *Microsoft Bot Framework* (Microsoft, 2022) e *Rasa* (Rasa, 2022). Existem trabalhos comparando as ferramentas em termos de fatores como facilidade de uso, integração com plataformas, idiomas suportados, precisão no processamento de linguagem natural, qualidade da documentação e custo. O *Dialogflow*, ferramenta proprietária do Google, é visto como a ferramenta mais fácil para desenvolvedores com menos experiência em codificação que desejam criar aplicações básicas. O *Amazon Lex* assume que o desenvolvedor têm uma base sólida em aprendizado de máquina e em desenvolvimento de sistemas de diálogo, enquanto o *IBM Watson Assistant* e o *Microsoft Bot Framework* são mais direcionados para corporações e empresas. O *Rasa* é diferente das ferramentas mencionadas anteriormente por ser uma ferramenta de código aberto. É também preferida e utilizada por muitos desenvolvedores (McTear, 2020).

Todas essas ferramentas citadas anteriormente geram agentes de diálogos por meio da enumeração exaustiva do espaço de fluxos de diálogo ou com soluções de propósito especial para tarefas específicas. No entanto, se a interação com o usuário desvia do padrão, essas ferramentas deparam-se com a limitação de que os caminhos de conversação devem ser explicitamente capturados em *árvores de diálogos* (Muisse *et al.*, 2019).

Planejamento Automatizado é a subárea de Inteligência Artificial que estuda o processo deliberativo de **raciocínio sobre ações** para que um agente possa alcançar suas metas (Ghallab *et al.*, 2004). Um *planejador* é uma algoritmo capaz de obter um plano de ações, a partir da análise de uma descrição declarativa da dinâmica do *ambiente*, do estado inicial do agente e da meta a ser alcançada. A área de planejamento clássico considera que todas as ações do agente possuem efeitos determinísticos, isto é, uma ação quando executada em um estado conduz o agente para apenas um estado sucessor. No entanto, no mundo real, as ações podem ter efeitos não-determinísticos, isto é, quando uma dada ação é executada em um estado não sabemos ao certo que estado sucessor o agente pode alcançar (Cimatti *et al.*, 2003; Pereira, 2007; Muise *et al.*, 2013; Menezes *et al.*, 2014; Santos *et al.*, 2022). A comunidade de planejamento utiliza a linguagem declarativa PDDL (*Planning Domain Description Language*) como linguagem padrão para a descrição de domínios e problemas de planejamento (Helmert, 2009).

Recentemente, (Muisse *et al.*, 2019) propôs a utilização da abordagem de planeja-

mento automatizado para a tarefa de geração de agentes de diálogos de uma forma drasticamente diferente do que têm sido feito na indústria: no lugar de explicitamente escrever e manter árvores de diálogos ou utilizar algoritmos de aprendizagem de máquina os autores utilizam o arcabouço de *Planejamento Automatizado* para especificar de forma declarativa ações, que são intrinsecamente não determinísticas, pois o efeito de cada ação depende do texto digitado pelo usuário no diálogo. Essa estratégia permite obter automaticamente o fluxo do diálogo, eliminando a necessidade de uma definição explícita das interações. Tal sistema foi incorporado na ferramenta IBM Watson.

O Plantão Coronavírus é um *chatbot* projetado pelo Governo do Estado do Ceará para atendimento a pacientes com suspeita de COVID-19 nos momentos mais críticos da pandemia durante os anos de 2020 e 2021, quando a escassez de profissionais de saúde disponíveis para atendimento era notória e houve um aumento significativo no número de casos da doença (Mont'Alverne, 2021). Este *chatbot* auxilia na identificação de possíveis casos da doença e também fornece orientação especializada por meio de profissionais de saúde, além de disponibilizar informações sobre as medidas necessárias para lidar com a pandemia (Cardoso, 2020).

1.1 Objetivos

O objetivo geral deste trabalho é a modelagem, implementação e avaliação de um gerenciador de diálogos para o *chatbot* Plantão Coronavírus utilizando a abordagem de Planejamento Não-Determinístico. Os objetivos específicos são:

- Modelagem do domínio de planejamento em PDDL com predicados e ações não-determinísticas a partir dos dados de conversa entre usuário e *bot*¹;
- Proposta de uma arquitetura híbrida para o agente de diálogo que utiliza *Aprendizado de Máquina* para a tarefa de compreensão da linguagem natural e *Planejamento Automatizado* para a tarefa de gerenciamento do diálogo;
- Implementação da arquitetura proposta utilizando o Rasa como sistema de conversação e um planejador não-determinístico estado-da-arte para obtenção dos planos de diálogo;
- Avaliação dos planos gerados, simulando se estes eram capazes de conduzir diálogos reais.

¹ Os diálogos analisados foram coletados durante o período de utilização da ferramenta. É importante ressaltar que esses dados são sigilosos e, portanto, neste trabalho, apresentamos apenas os resultados obtidos a partir da análise desses diálogos, sem revelar qualquer informação específica de qualquer usuário do sistema.

1.2 Organização do texto

Este trabalho está organizado como descrito a seguir:

- O Capítulo 2 apresenta a fundamentação teórica, incluindo os tópicos Sistemas de Diálogos, O *chatbot* Plantão Coronavírus e Planejamento Automatizado;
- O Capítulo 3 apresenta os trabalhos relacionados com este trabalho, os quais também utilizam o arcabouço de planejamento automatizado em Sistemas de Diálogos;
- O Capítulo 4 descreve o processo de aquisição do domínio de planejamento gerenciador do *chatbot* Plantão Coronavírus; a arquitetura e detalhes de implementação do agente de diálogo proposto;
- O Capítulo 5 mostra a avaliação do gerenciador de diálogo, com simulações que tem o objetivo de verificar se os planos gerados automaticamente pelo gerenciador de diálogos seriam capazes de conduzir diálogos reais e;
- O Capítulo 6 apresenta as conclusões e propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Diálogos

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). *ELIZA* (Weizenbaum, 1966) é conhecido como o primeiro *chatbot*. Que era baseado em texto, e tentava simular uma psicoterapeuta, muitas vezes de forma convincente. Com o passar do tempo, aspectos mais avançados do diálogo foram sendo investigados. Atualmente, as grandes empresas de tecnologia da informação mantêm assistentes pessoais (Amazon, 2022b; Google, 2022b; Apple, 2022) que são formas bem elaboradas de comunicação entre máquina e usuário (Khan; Das, 2018).

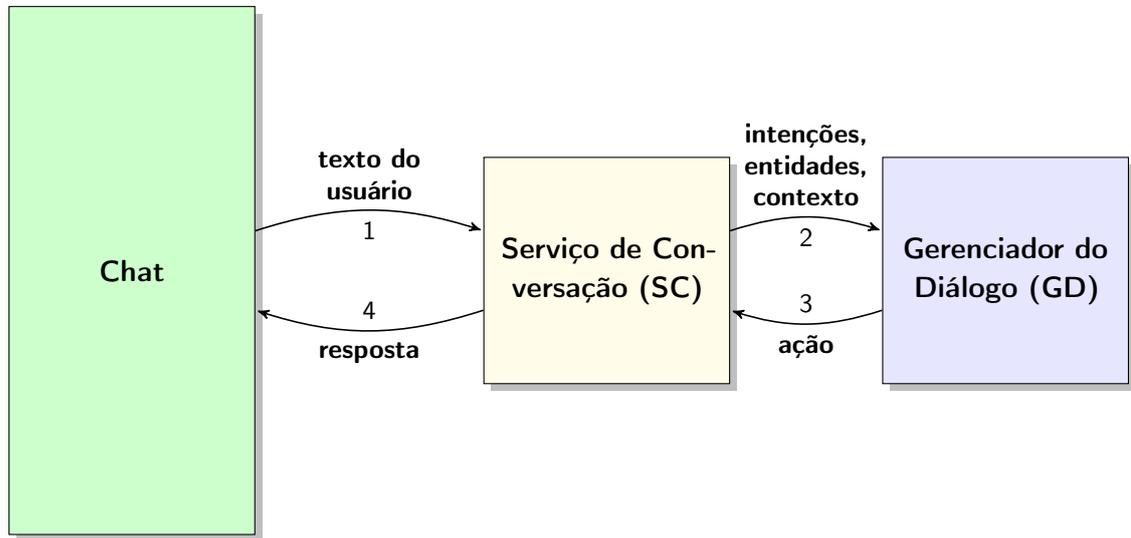
A Figura 1 apresenta a arquitetura básica de um sistema de diálogo. Nesta arquitetura há o módulo *Chat* que recebe o texto¹ digitado pelo usuário na interface do programa. Este texto é chamado de *declaração do usuário (user utterance)*. Na seta 1 da Figura 1 a *declaração do usuário* é enviada para o módulo *Serviço de Conversação (SC)*. No SC as declarações do usuário são classificadas em *intenções (intents)* que descrevem como as declarações do usuário devem ser categorizadas. Por exemplo, uma declaração como “*Gostaria de um sorvete de morango.*” pode ser classificada como uma intenção “*PEDIDO-DE-SORVETE*”. Ainda no SC é feito o reconhecimento das *entidades (entities)* que são variáveis estruturadas dentro das declarações do usuário (Botea *et al.*, 2019). Elas podem ser de uso geral como nomes, lugares, números, mas também podem ser definidas de forma específica para um determinado domínio. Por exemplo, na declaração “*Gostaria de um sorvete de morango.*”, a palavra “*morango*” pode ser classificada como uma entidade “*\$sabor*”. O SC, baseado no histórico de conversação, extrai o *contexto (context)* 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 *Gerenciador de Diálogo (GD)*, como podemos ver na seta 2 da Figura 1. O GD é o componente central de um sistema de diálogo (McTear, 2020). 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 (action)* o sistema vai executar. Após a escolha da ação, o GD envia para o SC a ação, como podemos ver na seta 3 da

¹ A entrada pode ser também a fala do usuário. No entanto, neste trabalho estamos considerando apenas a entrada por texto.

Figura 1. No SC, a ação é transformada em texto, o qual é enviado para o módulo *Chat*, como podemos ver na seta 4.

Figura 1 – Arquitetura básica de um Sistema de Diálogo



Fonte: adaptado de (McTear, 2020)

De acordo com Botea *et al.* (2019), o gerenciamento de diálogo pode se enquadrar em duas categorias:

- (i) o que utiliza árvores de diálogo e;
- (ii) o que utiliza aprendizado de máquina.

Nos sistemas cujo o gerenciamento do diálogo é baseado em árvore de diálogos, cada caminho que o usuário pode seguir no diálogo é pré-definido para que o sistema tenha controle do diálogo. Assim, uma árvore de diálogo pode tornar-se muito complexa a medida que se tem mais opções de conversa, inviabilizando sua construção manual. Além disso, pode existir muita repetição, nas quais interações de diálogo semelhantes podem ser necessárias em diferentes pontos da conversa (Botea *et al.*, 2019).

Nos sistemas que utilizam aprendizado de máquina para a tarefa de gerenciamento do diálogo, o gerenciamento é feito por meio de algoritmos que computam a próxima resposta, baseando-se em exemplos ou em históricos de interações. Estes sistemas permitem o desenvolvimento rápido de um diálogo complexo sem a necessidade da tarefa de construir uma árvore de diálogo. No entanto, as desvantagens incluem a falta de controle do proprietário do sistema, e os resultados potencialmente imprevisíveis. Em certos domínios, a incerteza e o risco de respostas erradas podem ser aceitáveis, mas em domínios como saúde, finanças e recursos humanos são

exigidas garantias e previsibilidade de resultados (Botea *et al.*, 2019).

Os *sistemas de diálogos* podem ainda ser *não orientados à meta* ou *orientados à meta* (McTear, 2020). Em um diálogo não orientado à meta o usuário e o sistema envolvem-se em uma interação conversacional geral (também conhecida como *chit-chat*). Nos diálogos orientados à meta, o usuário e o sistema envolvem-se em uma interação para completar alguma tarefa. O Exemplo 2.1.1 mostra um diálogo não orientado à meta. Já o Exemplo 2.1.2 mostra um diálogo no qual o usuário precisa planejar sua viagem. Para isso, o sistema faz perguntas para obter informações necessárias para alcançar o objetivo desejado.

Exemplo 2.1.1 *Um trecho de diálogo não orientado à meta.*

USUÁRIO: *Como você está hoje?*

AGENTE: *Ah. Tudo está indo extremamente bem. Como você está?*

USUÁRIO: *Nada mal.*

AGENTE: *Fico feliz em escutar isso. Quais as novidades?*

USUÁRIO: *O tempo não parece muito bom hoje.*

AGENTE: *Onde você gostaria de estar?*

USUÁRIO: *Em algum lugar quente e ensolarado.*

Fonte: traduzido de (McTear, 2020).

Exemplo 2.1.2 *Um trecho de diálogo orientado à meta.*

USUÁRIO: *Olá! Gostaria de comprar uma passagem.*

AGENTE: *Para onde você que ir?*

USUÁRIO: *São Paulo.*

AGENTE: *Tem preferência por aeroporto em São Paulo?*

USUÁRIO: *Não.*

AGENTE: *De onde você vai partir?*

USUÁRIO: *Fortaleza.*

AGENTE: *Qual a data que você deseja viajar?*

USUÁRIO: *Próxima quinta.*

AGENTE: *Ok. Temos essas opções:*

(i) *vôo direto XX-1243 às 13:00h, valor R\$800, chegando em Congonhas e;*

(ii) *vôo direto YY-3213 às 20:00h, valor R\$600,00, chegando em Guarulhos.*

USUÁRIO: *Quero o vôo da tarde.*

AGENTE: *Você deseja escolher assento agora?*

USUÁRIO: *Não.*

AGENTE: *Ok. Clique aqui para inserir os dados de pagamento.*

AGENTE: *Pagamento confirmado! O voo XX-1234 foi reservado. Boa viagem!*

Fonte: elaborado pelo autor.

Os sistemas de diálogos também podem ser classificados de acordo com sobre quem tem o controle das interações conversacionais (McTear, 2020) como:

- *Diálogo controlado pelo sistema.* O sistema controla o diálogo. Existem alguns tipos de diálogos controlado pelo sistemas:
 - diálogos em que o sistema inicia a interação de forma proativa. Por exemplo, para lembrar o usuário a tomar sua medicação.
 - diálogos iniciados por um usuário em busca de instruções. Por exemplo, em um aplicativo de receitas online, onde o sistema disponibiliza um conjunto de instruções com poucas entradas do usuário, exceto para solicitar a próxima instrução ou repetir uma instrução.
 - diálogos iniciados por um usuário requisitando um serviço. Por exemplo, para fazer uma compra de passagem aérea, o sistema assume o controle da interação e faz perguntas para coletar informações do usuário e ajudá-lo a concluir a tarefa.
- *Diálogo multi-turno.* Nestas interações o sistema e o usuário podem assumir um turno no diálogo. Este tipo de diálogo geralmente não é suportado em sistemas atualmente implantados, mas é o foco de muitas pesquisas (McTear, 2020).

No Exemplo 2.1.3 temos um diálogo multi-turno, no qual, dependendo do momento do diálogo, o usuário ou agente assumem o comando para realizar uma inspeção veicular.

Exemplo 2.1.3 *Diálogo multi-turno de inspeção veicular.*

USUÁRIO: *Olá! Gostaria iniciar a inspeção.* ← *Usuário está no comando*

AGENTE: *Vamos começar!*

USUÁRIO: *O freio está ok.*

AGENTE: *Freio ok! Anotado!*

USUÁRIO: *Qual o próximo passo?* ← *Troca de turno*

AGENTE: *Verifique o nível do óleo, por favor.* ← *Agente está no controle*

USUÁRIO: *O nível do óleo está bom.*

AGENTE: *Óleo ok! Anotado!*

AGENTE: *Verifique a pressão da vedação da embreagem.*

USUÁRIO: *Quais as opções?*

AGENTE: *Apertado, solto, precisa de reparo.*

USUÁRIO: *Está solta;*

AGENTE: *A vedação da embreagem está solta! Anotado!*

AGENTE: *Verifique agora a calibragem dos pneus.*

USUÁRIO: *Vou assumir agora ok?* ← *Troca de turno*

AGENTE: *Tudo bem! Como desejar.*

USUÁRIO: *Os pneus da frente não estão calibrados.* ← *Usuário está no controle*

AGENTE: *Certo, pneus dianteiros não calibrados.*

USUÁRIO: *Vejo aqui também que precisa trocar as velas de ignição*

AGENTE: *Trocar velas, anotado.*

Fonte: adaptado de (Muisse *et al.*, 2019)

Os principais *frameworks* para o desenvolvimento de sistemas de diálogos (McTear, 2020) disponíveis atualmente no mercado são descritos a seguir. Eles utilizam algoritmos de aprendizado de máquina para as tarefas de Processamento de Linguagem Natural (NLP), Compreensão da Linguagem Natural (NLU) e Gerenciamento do diálogo (McTear, 2020; Rasa, 2022).

2.1.1 *Watson Assistant*

O IBM *Watson Assistant* (IBM, 2022) é um serviço comercial de código fechado, disponibilizado pela IBM, por meio da sua infraestrutura de nuvem *IBM Cloud*. O Watson reconhece linguagem natural para criação de chatbots, no qual usa classificação de intenção, reconhecimento de entidade e entendimento do contexto. Também possui uma biblioteca com algumas intenções e entidades já pré definidas para serem usadas. Para utilizar este serviço é necessário criar uma conta na *IBM Cloud*.

2.1.2 *Dialogflow*

O *DialogFlow* (Google, 2022a) é uma plataforma de código fechado para a criação de chatbots que fornece interatividade para criar um conjunto de entradas do usuário e suas

respostas esperadas por meio de uma interface gráfica. Na plataforma são usados recursos nativos de aprendizado de máquina para entender as entradas do usuário, reconhecer entidades e determinar que ação deve ser executada. O Dialogflow disponibiliza uma interface no ambiente web. Esta interface é utilizada pelo o usuário para criar agentes, intenções, entidades e contextos. A plataforma também possui uma biblioteca de entidades pré-definidas e uma intenção *fallback* que serve para captar algo digitado que não é esperado pelo sistema. Para poder usar o *Dialogflow* é necessário ter uma conta do Google.

2.1.3 Amazon Lex

O *Amazon Lex* (Amazon, 2022a) é um serviço comercial de código fechado para criação de interfaces de conversa para aplicativos que permite o uso de voz e texto. Este serviço é disponibilizado por meio da infraestrutura de nuvem na *Amazon Web Services* (AWS). O serviço permite especificar o fluxo de conversa básico e a ferramenta ajusta dinamicamente as respostas no diálogo. O Lex utiliza técnicas de aprendizado de máquina para entender a intenção do usuário e preencher *slots*. O Amazon Lex utiliza conceitos de intenções e *slots*, onde tenta armazenar informações importantes no *slot* para controlar o fluxo do dialogo. O Amazon Lex também simula o fluxo do diálogo fornecendo exemplos para o usuário. Para acessar tal serviço é necessário ter uma conta na AWS.

2.1.4 Microsoft Bot

O *Bot Framework* (Microsoft, 2022) é um serviço gerado pela Microsoft e possui seu próprio kit de desenvolvimento de software, o que permite os desenvolvedores criar *chatbots* que conversam de forma livre ou com interações guiadas, incluindo texto simples ou cartões que contêm texto, imagens e botões de ação. Para utilizar o *framework* é necessário ter um bom conhecimento de programação. No *Bot Framework* é possível controlar o fluxo de diálogo, mas para poder fazer classificação de intenções e outras funções relacionadas a processamento de linguagem natural é necessário acessar serviços externos como chamadas a APIs que forneçam esse tipo de serviços.

2.1.5 RASA

Rasa (Rasa, 2022) é um *framework* de código aberto que usa aprendizado de máquina para criar *chatbots* que podem entender contextos (Lacerda, 2019). Sua implementação é feita na linguagem de programação Python. Utilizando dados de conversa, este framework é capaz de gerar respostas para um usuário, classificar intenções e extrair entidades. Como o Rasa não é um serviço na web, não é necessário criar nenhuma conta para ter acesso a ferramenta. O Rasa é gratuito e pode ser alterado de acordo com a vontade do programador (Viceri, 2020).

A implementação de um *chatbot* no Rasa é feita utilizando módulos, em que cada módulo é responsável por uma função específica (Mamatha *et al.*, 2021), como apresentaremos a seguir:

1. **domain:** Neste módulo é definido todos os elementos que compõem o *chatbot* como as ações, entidades e intenções. A Figura 2 ilustra as intenções (*intents*) e as ações nas quais são estabelecidas as repostas (*responses*) ao usuário.

Figura 2 – Exemplo de um módulo domain da ferramenta Rasa

```
intents:
- greet
- goodbye
- affirm
- deny
- mood_great
- mood_unhappy
- bot_challenge

responses:
utter_greet:
- text: "Hey! How are you?"

utter_cheer_up:
- text: "Here is something to cheer you up:"
  image: "https://i.imgur.com/nGF1K8f.jpg"

utter_did_that_help:
- text: "Did that help you?"
```

Fonte: (Rasa, 2022)

2. **nlu:** É o módulo onde um conjunto de intenções é definido para que a intenção extraída das declarações do usuário seja correspondida com qualquer uma das declarações neste conjunto. Neste módulo é definido exemplos de textos que correspondem à intenção. Na Figura 3 temos uma intenção de saudação denominada “*greet*” e exemplos de texto que são categorizados como esta intenção (*hey, hello, hi, hello there*).

Figura 3 – Exemplo definição de intenção no módulo nlu do Rasa.

```
- intent: greet
examples: |
  - hey
  - hello
  - hi
  - hello there
```

Fonte: (Rasa, 2022)

3. **actions:** No módulo *actions* é possível escrever ações personalizadas que podem fazer chamadas a, por exemplo, serviços externos. Na Figura 4 temos a ação “*action_hello_world*” que, quando executada, exibirá o texto “Hello World!” na tela.

Figura 4 – Exemplo de ação customizada criada no Rasa.

```
class ActionHelloWorld(Action):

    def name(self) -> Text:
        return "action_hello_world"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        dispatcher.utter_message(text="Hello World!")

        return []
```

Fonte: (Rasa, 2022)

4. **stories:** É o módulo que contém as possíveis interações dos diálogos. Neste módulo é definido qual ação o *chatbot* deve executar para cada intenção classificada e o fluxo do diálogo. A Figura 5 ilustra alguns caminhos que o diálogo pode seguir. Há dois caminhos principais nesta interação. O primeiro é o “caminho feliz“, onde cada passo é cuidadosamente delineado. Inicialmente, o usuário inicia uma saudação com uma intenção de cumprimento (greet), seguido por uma ação do chatbot expressando um cumprimento (utter_greet). Em seguida, o diálogo avança com a intenção do usuário de estar de ótimo humor (mood_great), e o ramo conclui com uma resposta do chatbot manifestando felicidade (utter_happy). O segundo caminho é o “caminho triste“, que se inicia de forma semelhante ao caminho feliz, mas, em vez da intenção (mood_great), é empregada a intenção de estar infeliz (mood_unhappy), indicando que o usuário não está muito contente. Em seguida, o chatbot realiza a ação de animar (utter_cheer_up),

seguida por uma ação de auxílio (`utter_did_that_help`). Posteriormente, o usuário expressa uma afirmação (`affirm`) e, por fim, o ramo é encerrado com a ação do chatbot declarando felicidade (`utter_happy`).

Figura 5 – Modelagem de caminhos que um usuário pode seguir em um fluxo do diálogo.

```

1  version: "3.1"
2
3  stories:
4
5  - story: happy path
6    steps:
7      - intent: greet
8      - action: utter_greet
9      - intent: mood_great
10     - action: utter_happy
11
12 - story: sad path 1
13   steps:
14     - intent: greet
15     - action: utter_greet
16     - intent: mood_unhappy
17     - action: utter_cheer_up
18     - action: utter_did_that_help
19     - intent: affirm
20     - action: utter_happy
21

```

Fonte: (Rasa, 2022)

2.2 O Chatbot Plantão Coronavírus

Este trabalho propõe o uso de planejamento automatizado para a tarefa de gerenciamento de diálogo no *chatbot* Plantão Coronavírus (Mont'Alverne, 2021). Esse *chatbot* foi disponibilizado para uso pelo Governo do Estado do Ceará² para auxiliar no atendimento aos cidadãos durante a pandemia de COVID-19. De abril de 2020 a fevereiro de 2021 foram mais 400 mil acessos ao sistema e cerca de 42 mil pessoas atendidas neste período da pandemia, onde a situação estava crítica com poucos profissionais de saúde disponíveis para atendimento e com muitos casos de pessoas infectadas (Mont'Alverne, 2021).

Por meio deste *chatbot* é possível que um cidadão busque informações sobre o coronavírus; busque informações sobre locais de atendimentos físicos e; informe sintomas ao sistema para ser encaminhado a conversar com um profissional de saúde, seja médico, enfermeiro, psicólogo, dentre outros. O Exemplo 2.2.1 mostra o trecho de diálogo inicial do *chatbot*. no qual há quatro possíveis escolhas para o usuário: (1) atendimento online; (2) informações; (3) cuidados com a saúde mental e; (4) outros. Na primeira opção, o usuário deseja ter uma consulta com um profissional de saúde. Na segunda opção é o caso do usuário apenas desejar obter

² <https://coronavirus.ceara.gov.br/>

informações sobre a doença. A terceira opção é para mostrar informações relacionadas sobre os cuidados com a saúde mental. E, a quarta opção *Outros* é disponibilizada caso o usuário deseje fazer alguma denúncia ou falar diretamente com a Central de Atendimento da Ouvidoria do Estado ³.

Neste *chatbot* temos que o diálogo inicia entre usuário (humano) e o *bot* e depois pode migrar para um diálogo entre usuário (humano) e profissional de saúde (humano). O Exemplo 2.2.2 mostra que o usuário optou pelo atendimento online e ao informar seus sintomas foi constatado que ela possa estar infectado com o coronavírus. Deste modo, o agente começa a fazer algumas perguntas para obter informações do usuário e, ao fim deste questionamento, há uma troca de turno com um profissional de saúde humano. Quando o profissional de saúde conclui o atendimento, o turno é novamente passado ao *bot* para finalizar o diálogo.

Exemplo 2.2.1 *Um trecho de diálogo - opções de escolhas iniciais dadas ao usuário.*

USUÁRIO: *Olá!*

AGENTE: *Olá! Bem-vindo(a) ao serviço de atendimento online.*

AGENTE: *Do que você precisa agora?*

1. *Atendimento online sobre o novo Coronavírus (COVID-19)*
2. *Informações sobre o novo Coronavírus (COVID-19)*
3. *Cuidados com a Saúde Mental*
4. *Outros*

Fonte: elaborado pelo autor

Exemplo 2.2.2 *Um trecho de diálogo - troca de turno entre bot e profissional de saúde humano.*

USUÁRIO: *1. Gostaria de um Atendimento Online.* ← *Usuário está no comando*

AGENTE: *É muito importante responder todas as perguntas e completar o atendimento. Ao final, se for necessário, uma equipe de saúde estará pronta para conversar e cuidar de você.* ← *troca de turno*

AGENTE: *Você está com algum desses sintomas?* ← *Agente está no comando*

1. *Falta de ar*
2. *Mal-estar intenso*
3. *Tosse incontrolável*
4. *Alargamento das narinas durante a respiração*
5. *Febre acima de 37.8C persistente por mais de 48h*

³ www.ouvidoria.ce.gov.br

6. *Confusão mental*

7. *Nenhum dos sintomas acima.*

USUÁRIO: *Falta de ar, Mal-estar intenso e Febre acima de 37.8*

AGENTE: *Há quantos dias os sintomas começaram?*

USUÁRIO: *Há 3 dias.*

AGENTE: *De acordo com suas respostas e com o protocolo do Ministério da Saúde, você pode estar com o novo Coronavírus (COVID-19).*

AGENTE: *Vamos direcioná-lo(a) para nossa equipe de saúde. Digite 'Compartilhar minha localização' para compartilhar onde você está. Ao compartilhar sua localização, você ajuda o Governo a te proteger e proteger os outros.*

USUÁRIO: *compartilhar localização.*

AGENTE: *Qual o seu nome?*

USUÁRIO: *É Bruno da Rocha Silveira.*

...

AGENTE: *Me informe o seu número de telefone, por favor?*

Não esqueça do DD. Exemplo: (82) 90000-00000 ou 82912345678.

USUÁRIO: (88)99468421 ← *troca de turno*

AGENTE HUMANO: *Olá Bruno! me chamo Paula e vou te atender ...* ← *Agente humano no controle*

USUÁRIO: *Olá Paula! Eu ...*

...

AGENTE HUMANO: *Tudo pronto. Agora vou encerrar seu atendimento. OK?*

USUÁRIO: *Ok!* ← *troca de turno*

AGENTE: *Certo! Antes de encerrar peço 1 minutinho para que você avalie nosso atendimento (digite um número de 1 a 10), sendo:* ← *Agente está no comando*

1. Odiei. - 10. Amei!

USUÁRIO: *9*

AGENTE: *Seu atendimento foi finalizado!*

Fonte: elaborado pelo autor

Em um diálogo com o usuário humano muitos imprevistos podem ocorrer: o usuário pode não necessariamente responder de forma apropriada as perguntas do *bot*. Nestes casos, o chatbot, sempre que recebe um texto que não corresponde ao contexto atual da conversa, envia

uma mensagem informando que não entendeu a mensagem do usuário e, em seguida, repete a mensagem enviada anteriormente. O Exemplo 2.2.3 ilustra a situação em que o usuário informou algo para o *bot* que correspondia ao contexto da conversa naquele momento. Já no Exemplo 2.2.4, temos que o usuário informou algo que o *bot* não entendeu como correspondente ao contexto do diálogo naquele momento.

Exemplo 2.2.3 *Trecho de diálogo - o usuário responde como esperado a pergunta do bot.*

AGENTE: *Qual o seu nome completo?*

USUÁRIO: *Meu nome é Bruno da Silva Pinho.*

AGENTE: *Qual a sua data de nascimento?*

Fonte: elaborado pelo autor

Exemplo 2.2.4 *Trecho de diálogo - o usuário responde de forma inesperada a pergunta do bot.*

AGENTE: *Qual o seu nome completo?*

USUÁRIO: *fsfsfffsf*

AGENTE: *Não conseguir entender o que você quis dizer.*

AGENTE: *Qual o seu nome completo?*

Fonte: elaborado pelo autor

2.3 Planejamento Automatizado

Planejamento Automatizado (Ghallab *et al.*, 2004) é uma subárea da Inteligência Artificial dedicada a encontrar plano de ações que alcançam os objetivos de um agente inteligente (Ghallab *et al.*, 2004). O *planejador* (Figura 6) é um algoritmo que recebe como entrada uma descrição da dinâmica do ambiente (*domínio de planejamento*), um *estado inicial* e uma *meta* do agente e obtém um *plano de ações* capaz de levar o agente do estado inicial para a meta. Caso não seja possível obter este plano, o planejador devolve *fracasso*.

Figura 6 – Esquema de um Planejador.



Fonte: (Pereira, 2007)

Uma forma de descrever domínios de planejamento de forma compacta é utilizando linguagens de ações. Uma das linguagens de ações mais conhecidas é a linguagem PDDL

(*Planning Domain Description Language*) (McDermott, 2000). Em PDDL, o domínio contém *predicados e ações*, definidas por pré-condições e efeitos.

Exemplo 2.3.1 *Domínio PDDL de um planejamento de viagem.*

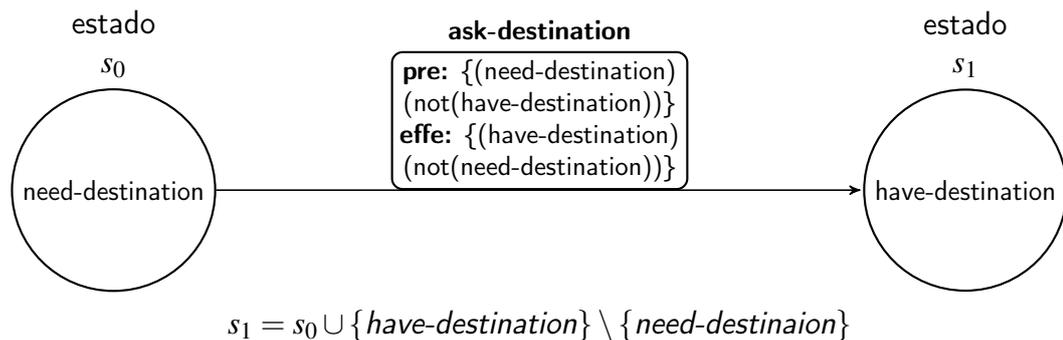
```
(define domain trip_planning
  (:predicates
    (have-destination)
    (need-destination)
    ...
  )
  ; E.g, "Pra onde voce gostaria de ir?"
  (:action ask-destination
    :precondition ( and (need-destination)
                       ( not (have-destination))
                     )
    ;E.g., "Whistler, BC."
    :effect ( and (have-destination)
                  ( not (need-destination))
                )
  )
  ...
)
```

Fonte: Elaborado pelo autor.

No Exemplo 2.3.1 é ilustrado uma representação do domínio do *planejamento de viagem* em PDDL. Este domínio consiste em um agente de diálogo cujo objetivo é fornecer serviços de reserva de viagem. Neste domínio temos um conjunto de predicados que representam se uma informação relacionada ao diálogo foi capturada ou não. Por exemplo, o predicado *have-destination* indica que o agente possui a informação sobre o destino da viagem. Quando se tem esta informação, o predicado se torna verdadeiro. Neste domínio o agente pode executar a ação *ask-destination* que consiste em perguntar para o usuário qual lugar ele gostaria de viajar.

Os estados do mundo são representados por um conjunto átomos proposicionais. Utiliza-se a suposição do *mundo fechado* em que representa-se, em cada estado, apenas o que é *verdadeiro*, sendo que o que não está na descrição do estado é considerado *falso*. As ações que o agente pode executar são descritas por meio de pré-condições e efeitos. As pré-condições são definidas por meio de um conjunto de átomos proposicionais que precisam ser verdadeiros para que se possa executar tal ação em um dado estado. Cada ação executada modifica o estado atual, gerando um estado sucessor por meio dos *efeitos*. No Exemplo 2.3.1, a ação *ask-destination* possui as pré-condições *need-destination* e *not(have-destination)*. Os efeitos desta ação são *have-destination* e *not(need-destination)*. Quando uma ação é aplicada em um estado, é gerado um novo estado a partir do estado anterior, adicionando na descrição deste o que ficou verdadeiro com o efeito da ação e removendo o que ficou falso com o efeito da ação (Ghallab *et al.*, 2004). Por exemplo, na Figura 7, temos um estado s_0 no qual temos que é verdade $\{need-destination\}$. Para que a ação *ask-destination* seja aplicada em s_0 é necessário que s_0 satisfaça suas pré-condições: $\{need-destination, not(have-destination)\}$. Assim, como s_0 satisfaz as pré-condições da ação, podemos gerar um novo estado s_1 a partir da descrição do estado s_0 , adicionando *have-destination* e removendo *need-destination*, como podemos ver na Figura 7.

Figura 7 – Aplicando ação a *ask-destination* ao estado s_0 .



Fonte: Elaborado pelo autor

Problemas de planejamento são atribuídos a um domínio, definindo-se um estado inicial e uma meta a ser alcançada. O estado inicial é uma descrição completa, em que se define todos os predicados que são verdadeiros. A meta é uma descrição parcial em que se informa apenas o que precisa ser alcançado pelo agente. Assim, podemos ter vários estados que

satisfazem a descrição na meta (Ghallab *et al.*, 2004).

No Exemplo 2.3.2 é mostrado a descrição de um problema de planejamento no domínio de viagem em PDDL. Na linha 2, temos o estado inicial contendo $\{(need-destination), (need-dates), need-departure\}$. Na linha 3, temos a meta que o agente precisa alcançar, isto é, um estado no qual $\{confirm-trip\}$ seja verdade, indicando que a viagem foi confirmada.

Exemplo 2.3.2 Problema do domínio do planejamento de viagem

```

1 (define (problem trip_planning)
2   (init:(need-destination)(need-dates)(need-departure))
3   (goal:(confirm-trip))
4 )

```

Fonte: Elaborado pelo autor.

Um *plano* é a sequência de ações para que o agente consiga sair do estado inicial e alcance um estado meta.

O planejamento clássico supõe que o ambiente evolui de forma *determinística*, ou seja, não há efeitos incertos nas ações executadas pelo agente, e o ambiente é alterado por consequência apenas destas ações (Ghallab *et al.*, 2004). Porém no mundo real temos que uma ação podem ter efeitos incertos. Por exemplo, no domínio de planejamento de viagem, temos que a ação *ask-destination* possui apenas um efeito. Desta forma, espera-se que o usuário responda a ação apenas de uma forma. Porém sabemos que no mundo real um usuário pode dar diferentes respostas para uma mesma pergunta. Desta forma é necessário assumirmos que em domínios para gerenciamento de agentes de diálogos as ações são devem ser não-determinísticas (Muisse *et al.*, 2019).

2.3.1 Planejamento Não-Determinístico

Em um domínio de um planejamento não-determinístico as ações podem ter efeitos incertos. No Exemplo 2.3.3 é mostrado uma ação não-determinística do domínio de planejamento de viagem. Os efeitos não-determinístico da ação *ask-destination* representam as diferentes formas de como um usuário pode responder a pergunta de qual lugar deseja viajar: ele pode responder informando o local (*have-destination*) que gostaria de viajar (por exemplo, respondendo com o nome de uma cidade); finalizando a conversa se não for o caso de querer

planejar a viagem (*not need-destination*), (por exemplo, respondendo “*para nenhum lugar agora*”); ou ainda digitando algo que precisa ser esclarecido (*must-clarify*), (por exemplo, respondendo algo que não seja possível reconhecer como “*fhdwhgads*”). Para expressar os efeitos não determinísticos em PDDL, utiliza-se a palavra reservada *one-of* que significa a possibilidade de um destes efeitos ocorrer.

Exemplo 2.3.3 *Ação não determinística no domínio de planejamento de viagem na linguagem PDDL*

```

; E.g, "Pra onde voce gostaria de ir?"
(:action ask-destination
  :precondition ( and (need-destination)
                    ( not (have-destination))
                )
  :effect ( oneof
            ;E.g., "Whistler, BC."
            ( and (have-destination))
            ;E.g., "Nenhum lugar agora."
            ( and ( not (need-destination)) )
            ;E.g., "fhdwhgads"
            ( and (must-clarify))
          )
)

```

Fonte: adaptado de (Muisse *et al.*, 2019)

Devido à existência de efeitos incertos nas ações, um plano para um problema não-determinístico pode resultar em múltiplos caminhos (Pereira, 2007). Um plano solução para este tipo de problema é uma *politica*, definida por um mapeamento de estados em ações. Dessa forma a politica define qual é a ação mais adequada para ser executada em cada estado resultante da aplicação de uma ação não-determinística.

De acordo com Cimatti *et al.* (2003), uma politica pode ser classificada em três tipos diferentes:

- **Política fraca:** consiste de uma politica que pode alcançar um estado meta, mas devido ao não-determinístico não possui garantia de que o faça.

- **Política forte:** consiste de uma política onde todos os caminhos possíveis do plano de execução alcançam a meta em um número finito de passos, ou seja, independentemente do não-determinismo a política garante alcançar sem ciclos o estado meta.
- **Política forte-cíclica:** consiste de uma política que alcança um estado meta pressupondo que sua execução conseguirá sair, em algum momento, de todos os ciclos existentes.

Um problema de planejamento em um domínio não-determinístico é definido da mesma forma que um problema em planejamento clássico, onde temos um estado inicial e uma meta que se deseja alcançar. Na literatura, há alguns planejadores que se propõem a resolver problemas de planejamento não determinístico, tais como: PRP (*Planner for Relevant Policies*) (Muisse *et al.*, 2012) e o planejador PACTL (Planejador Simbólico Baseado na lógica temporal α -CTL) (Pereira, 2007; Menezes *et al.*, 2014; Santos *et al.*, 2022). redcolocar também o do cimatti 2003 (Planejador Simbólico Baseado na lógica temporal CTL)

Neste trabalho, utilizaremos o planejador PRP, o qual é um planejador estado-da-arte que utiliza a técnica de *determinização* das ações, na qual é criada uma nova ação determinística para cada efeito não-determinístico de uma ação e é utilizado um planejador clássico para obter uma solução (Santos, 2018). O PRP tenta encontrar primeiro uma política forte ou forte-cíclica antes de retornar uma política fraca) (Santos *et al.*, 2022).

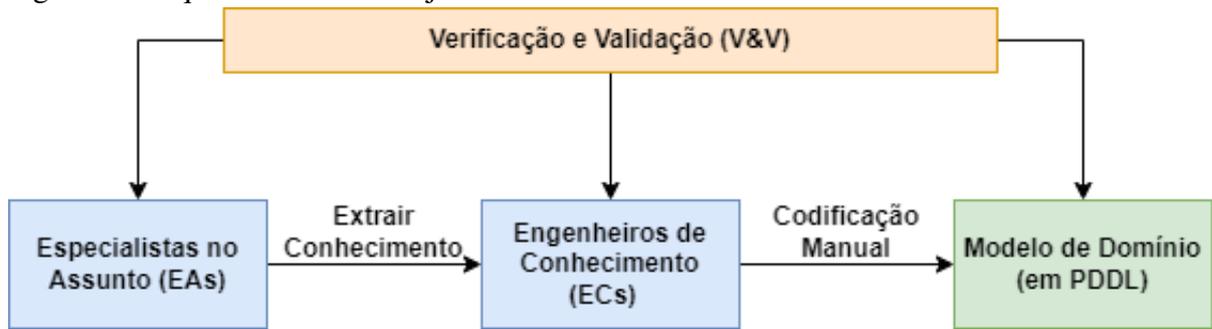
2.3.2 Paradigmas de Aquisição de Domínio de Planejamento

Segundo (Miglani; Yorke-Smith, 2020), há três paradigmas de aquisição de domínios de planejamento:

- **Codificação manual.** É realizado a codificação de forma manual por meio de uma colaboração cuidadosa entre *Especialistas no Assunto* (EAs) e *Engenheiros de Conhecimento* (ECs) (Eriksson, 1992). Neste tipo de codificação Tanto o EAs como o ECs são responsáveis pela Verificação e a Validação do domínio modelado. Deste modo ambos podem propor melhorias para o domínio modelado até que o mesmo esteja de acordo com o que se é desejado.

A Figura 8 ilustra o processo de codificação manual. Onde o EAs passa o seu conhecimento para ECs e através do conhecimento extraído o ECs modela o domínio desejado, e após a modelagem é feita a verificação e a validação domínio, caso haja melhorias a se fazer o processo é repetido novamente.

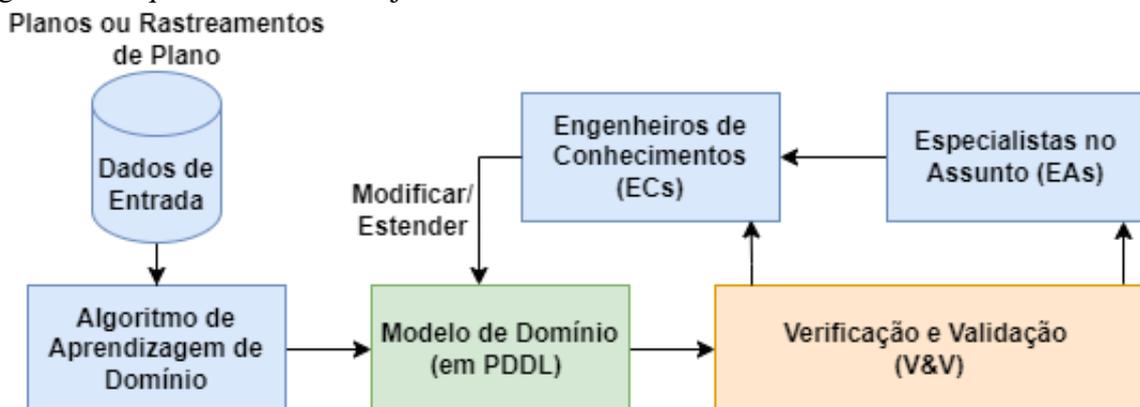
Figura 8 – Esquema de um Planejador.



Fonte: (Miglani; Yorke-Smith, 2020)

- **Codificação automatizada usando entrada estruturada.** É uma abordagem automatizada que aprende modelos de domínio a partir de histórico de planos (*plan traces*) através de algoritmo de aprendizagem de domínio. Porém esta abordagem necessita de uma grande quantidade substancial de dados estruturados para formar modelos significativos (Jilani *et al.*, 2014). A Figura 9 ilustra o processo desta abordagem, onde é passado para o algoritmo de aprendizado de domínio os históricos de planos. Então o algoritmo gera um modelo do domínio. Apesar de ser uma tarefa automatizada, ainda há a necessidade dos EAs e ECs para fazer a verificação e validação do modelo, assim podendo propor melhorias.

Figura 9 – Esquema de um Planejador.

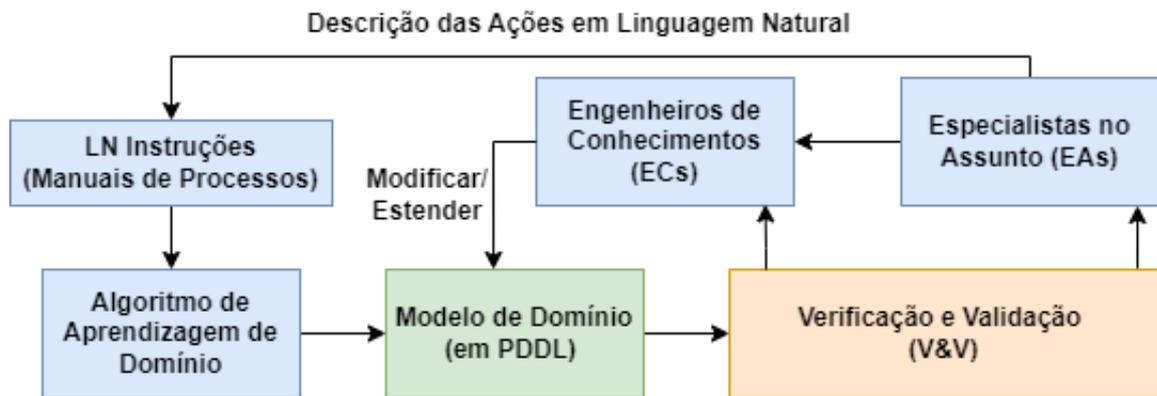


Fonte: (Miglani; Yorke-Smith, 2020)

- **Codificação automatizada usando entrada não estruturada.** É uma abordagem automatizada que aprende modelos de domínio a partir de dados não estruturados em *Linguagem Natural*. Neste abordagem é recebido como entrada um plano escrito manualmente que são instruções em linguagem natural, e por meio de algoritmos de aprendizagem é feito a extração sequencias de ação, assim gerando o modelo do domínio (Miglani; Yorke-Smith, 2020).

Na Figura 10 é ilustrado como é feito a codificação, Onde temos que os EAs são des-
 paisáveis por criar as instruções em linguagem natural. Estas instruções passa por um
 algoritmo de aprendizagem de domínio, que gera uma modelo. A verificação do modelo e
 a validação fica a cargo dos EAs e ECs, assim podendo os mesmos propor melhorias.

Figura 10 – Esquema de um Planejador.



Fonte: (Miglani; Yorke-Smith, 2020)

3 TRABALHOS CORRELATOS

3.1 Automating the Generation of Goal-Oriented Dialogue Managers for Healthcare

No trabalho de Teixeira (2022) é feito o levantamento de Abordagens Baseadas em planos para a gestão de diálogo onde foi conduzido um levantamento do estado-da-arte em abordagens baseado em planos para gerenciar agentes de diálogos. No levantamento feito pelo autor foram identificados as abordagens de planejamento utilizados mais recentes, levantando as características de planejamento que justificam a adoção dessa abordagem em sistemas de diálogos, e os desafios enfrentados no desenvolvimento de gerenciador de diálogos baseados em planos, no qual conduziu os criadores deste trabalho a utilizarem a abordagem de planejamento automatizado não determinístico.

Neste trabalho também foi realizado a aquisição de domínio em PDDL, onde se desenvolveu uma ontologia que formaliza o paradigma conversacional orientado para objetivos. A odontologia desenvolvidas pelo os autor visa modelar semanticamente e descrever os conceitos de um diálogo, o que auxilia na compreensão e gerenciamento do fluxo de um diálogo, e desta forma contribuir para a criação do domínio em PDDL, de forma que o domínio é criado a partir da ontologia. O autor introduziu uma framework baseado em agentes para a seleção das ações mais uteis em cada estado do diálogo. Segundo o autor esta framework contribuiu para a geração de um diálogo multi-turno que utiliza o mínimo de perguntas possíveis par recuperar informações de forma rápidas, porém cautelosas.

No trabalho foi feito a integração de planejamento automatizado e gerenciamento de informações, onde foi desenvolvido o OntoPlanDm, que é uma abordagem que une planejamento automatizado e gerenciamento de informação para a geração automatizada de planos que são utilizados para o gerenciamento de uma agente de diálogo voltado para a saúde. A abordagem proposta foi integrada em uma arquitetura de sistemas de diálogo que fornece componentes de linguagem natural.

Foi desenvolvido um protótipo de uma agente de diálogo implementando o OntoPlanDM, no qual foi denominado de Puffbot, no qual o mesmo é uma agente de conversacional projetado para apoiar o monitoramento de doenças asmáticas. Neste trabalho o agente de diálogo proposto foi testado e avaliado por especialistas em saúde e usuários finais.

O estudo realizado por Teixeira (2022) difere do presente trabalho em alguns aspectos. Embora ambos se concentrem no campo da saúde, suas abordagens se situam em domínios

diferentes. Enquanto os autores se dedicaram ao monitoramento de doenças asmáticas, nossa pesquisa está focada e relacionada relacionados ao Coronavírus. Além disso, em Teixeira (2022), a aquisição do domínio é conduzida por meio de ontologias, ao passo que em nossa investigação, tal aquisição foi realizada manualmente.

3.2 Planning for Goal-Oriented Dialogue Systems

No trabalho de Muise *et al.* (2019) é proposto o uso de planejamento automatizado na tarefa de *gerenciar diálogos* que são *multi-turnos e orientados a metas*. Assim, é possível eliminar a necessidade da especificação de árvores de diálogo para esta tarefa de gerenciamento. Os autores modelam as ações do *chatbot* como ações não-determinísticas em PDDL.

A modelagem proposta abstrai informações incluídas no contexto. Parte dos predicados são utilizados com variáveis correspondentes no contexto para que seja capaz de decidir sobre por quais ramos do diálogo deve-se prosseguir. Por exemplo, na declaração do usuário “*Eu quero ir para Paris*”, Paris poderia ser classificada como uma entidade *\$location-destination* que armazena a palavra “*Paris*”. No entanto, não é necessário guardar explicitamente o valor da localização, apenas é assumido que se tem um valor de contexto, utilizando o predicado *have-location-destination*. Na modelagem do domínio é utilizado os seguintes predicados:

- *OK-** - para indicar se uma *flag* booleana mantém-se verdadeira.
- *HAVE-** - para indicar que se tem um valor de contexto
- *MAYBE-** - para indicar incerteza de um valor de contexto.
- *GOAL* - predicado especialmente designado para a meta.

No trabalho a classificação de intenções e entidades é feita por meio do *Watson Assistant* (IBM, 2022) que assume o papel do serviço de conversação.

Na modelagem também é feito o mapeamento da ações que podem ser executada pelo *chatbot* para ações PDDL. Cada ação possui um conjunto de efeitos que representam a forma de como o contexto será atualizado. Os efeitos de uma ação modelam como o usuário pode responder ou, de forma mais geral, como o mundo pode evoluir em resposta a qualquer ação realizada pelo agente. No trabalho os autores apresenta ações básicas:

- *Ações de diálogo* - correspondem a mensagens enviadas para um usuário em uma conversa. Os efeitos não determinísticos do dialogo correspondem a como o usuário pode responder.
- *Ações Web* - correspondem a chamadas de serviços externos, como por exemplo uma chamada a uma API. Os efeitos não determinísticos correspondem a como o serviço pode

responder.

- *Ações de Sistema* - correspondem a ações relacionadas ao raciocínio do próprio agente.

A modelagem em PDDL é utilizada para gerar um política utilizando um planejador não-determinístico PRP (Muise *et al.*, 2012). A política gerada é denominada de *plano de diálogo*. Este plano é utilizado para gerenciar as ações do *chatbot*.

O gerenciamento é feito usando o plano de diálogo gerado pelo planejador, onde a partir do plano é escolhida qual ação será executada pelo o agente. Após a execução da ação, é obtido um resultado que pode ser uma declaração do usuário. Com o resultado, é feita a escolha de qual dos efeitos não-determinístico da ação ocorreu. Após a escolha, é atualizado o estado do mundo e o contexto. Este processo é repetido até que se consiga chegar a meta.

Nosso trabalho é baseado no trabalho de (Muise *et al.*, 2019), onde utilizaremos a modelagem de ações e arquitetura baseada na proposta pelos autores para produzir uma política para o gerenciamento do diálogo do *chatbot* Plantão Coronavírus. O gerenciador proposto por Muise *et al.* (2019) é uma ferramenta comercial de código fechado (até o presente momento) e, no artigo publicado, são dadas apenas algumas noções de como o gerenciador de diálogos foi construído. Os autores afirmam que a proposta é apenas uma maneira de modelar um *chatbot* utilizando PDDL, e que a mesma pode ser adaptada de acordo com a necessidade de quem está modelando o domínio. Assim, neste trabalho utilizamos a ferramenta Rasa (código aberto) como o Sistema de Conversação e para a tarefa de gerar a política utilizamos também o Planejador PRP. Assim, tanto o domínio PDDL modelado como a implementação do gerenciador de diálogos estão disponíveis em <https://github.com/brunopinho321/Domain-Chatbot-Covid>

3.3 A Generic Dialog Agent for Information Retrieval Based on Automated Planning within a Reinforcement Learning Platform

O trabalho de Pallagani e Srivastava (2021) propõe o uso de *planejamento automatizado e aprendizado por reforço* na construção de um sistema de diálogo para recuperação da informação (*information retrieval*) em base de dados. Segundo os autores, as formas atuais de se fazer essa recuperação, por meio da linguagem SQL ou por buscas no conjunto de dados, não são acessíveis a diversos usuários de base de dados. Assim, propõem o uso de sistemas de diálogos para tornar os dados acessíveis.

Os autores também justificam que a construção de sistemas de diálogos utilizando apenas métodos baseados em aprendizagem requer grandes quantidades de dados de treinamento

e resultam em sistemas que são específicos para um domínio de aplicação. A abordagem proposta permite escalar para novas fontes de dados e explorar as diversas formas de integração das técnicas de planejamento automatizado e aprendido por reforço. Uma das arquiteturas propostas é o planejador ser o responsável por buscar estratégias de resposta que são multi-turno e orientadas a metas (sem prévio treinamento dos dados) e; o módulo de aprendizado por reforço ser utilizado para automatizar a seleção de fontes de dados durante uma interação com o usuário.

Do lado de planejamento, a modelagem proposta utiliza *ações determinísticas com efeitos condicionais* escritas em PDDL para a tarefa de pesquisa das informações, como é o caso da ação CHECK-QUERY-TYPE na Figura 11. Nesta ação, a pré-condição é ter a consulta do usuário e os efeitos dependem se a query obtida é exata (*is-query-exact*) ou parcial (*is-query-partial*). Os efeitos condicionais são expressos em PDDL com o uso da palavra reservada *when*. A meta do problema de planejamento, por sua vez, refere-se à intenção reconhecida do usuário. O domínio e o problema de planejamento são submetidos ao planejador da nuvem *Solver.Planning.Domains*¹ que gera um plano a ser seguido pelo agente de diálogo. A Figura 12 mostra um exemplo de plano gerado para o agente de diálogo.

Figura 11 – Uma ação em PDDL para verificar um tipo de consulta

```
(:action CHECK-QUERY-TYPE
  :parameters (?x)
  :precondition ( and (status-have-user-query ?x))
  :effect ( and ( when status-have-user-query ?x)
                (status-have-exact-query ?x))
                ( when (is_query_partial ?x)
                (status-have-partial-query ?x))
  )
)
```

Fonte: adaptado de (Pallagani; Srivastava, 2021)

¹ <http://solver.planning.domains/>

Figura 12 – Plano gerado para uma consulta

```

GET-USER-QUERY user-dialogue
GET-DB-ACCESS user-dialogue
CHECK-QUERY-TYPE user-dialogue
PERFORM-PARTIAL-MATCH-DISAMBIGUATION user-dialogue
DISAMBIGUATE-HIGHEST-HIERARCHY-NODE user-dialogue
DISAMBIGUATE-DEPENDENT-HIERARCHY-NODE user-dialogue

```

Fonte: adaptado de (Pallagani; Srivastava, 2021)

As Figuras 14 e 13 mostram exemplos de diálogos de recuperação da informação na base de dados ICD (*International Statistical Classification of Diseases and Related Health Problems*) gerados pelo sistema proposto. Na Figura 13, o sistema encontra a informação que o usuário procura na base dados. Já na Figura 14 é mostrada que a informação que o usuário procura não está na base de dados.

Figura 13 – Exemplo de consulta do usuário que não foi encontrada.

Query do Usuário:	Resposta Final do Bot	Conclusão da Tarefa
Qual é o código para infarto agudo do miocárdio?	O código para infarto agudo do miocárdio é - I21	Sim (no escopo)

Fonte: adaptado de (Pallagani; Srivastava, 2021)

Figura 14 – Exemplo de consulta do usuário que não foi encontrada.

Consulta do Usuário:	Resposta Final do Bot	Conclusão da Tarefa
Qual o código para ataque do coração?	ataque do coração não está presente na fonte de dados	Não (fora do escopo)

Fonte: adaptado de (Pallagani; Srivastava, 2021)

O trabalho de (Pallagani; Srivastava, 2021) diferencia-se do nosso trabalho, pois é utilizado a abordagem de planejamento determinístico para a tarefa de recuperar informações em base de dados de acordo com uma consulta do usuário. Já em nosso trabalho é utilizado a abordagem de planejamento não-determinístico para a tarefa de gerenciamento de diálogo.

3.4 Cognitive-Inspired Conversational-Strategy Reasoner for Socially-Aware Agents

O trabalho de Romero *et al.* (2017) propõe um módulo para sistemas de diálogos chamado *Social Reasoner* que, baseado na meta do usuário, decide a estratégia e estilo de conversação apropriados com o objetivo de fortalecer a relação entre o usuário e sistema (*rapport*), fazendo com que o usuário se sinta mais a vontade para fornecer informações para o sistema conseguir atingir a meta.

O sistema proposto seleciona automaticamente, utilizando uma abordagem de planejamento determinístico, entre sete estratégias de conversação que impactam positivamente o relacionamento entre o usuário e sistema, as quais são:

1. *Self-Disclosure (SD)* - revelar informações pessoais, para diminuir a distância social;
2. *Question Elicitation of Self-Disclosure (QESD)* - que é usado para encorajar o outro interlocutor a se auto-revelar;
3. *Reference to Shared Experiences (RSE)* - que indexa a história comum;
4. *Praise (PR)* - que serve para aumentar a auto-estima do ouvinte e, portanto, a coesão interpessoal;
5. *Adhere to Social Norm (ASN)*, que aumenta a coordenação ao aderir às expectativas de comportamento guiadas por normas socioculturais;
6. *Violation of Social Norm (VSN)*, onde as normas gerais são intencionalmente violadas para acomodar as expectativas comportamentais dos outros; e
7. *Acknowledgement (ACK)* - uma forma de mostrar que o interlocutor está ouvindo.

Por exemplo, para obter a meta “saudações” o sistema pode produzir estilos diferentes de diálogos, dependendo da estratégia adotada:

- ASN: “Oi, Eu sou o Bruno. Posso saber qual o seu nome?”;
- SD: “Oi! Fico feliz que esteja aqui”;
- VSN: “E aí cara?!” ou;
- PR: “Oi, Eu sou o Bruno. É uma honra te encontrar aqui.”

Os autores modelam ações com pré-condições e efeitos referentes a cada uma das sete estratégias de conversação. A Figura 15 ilustra a ação que modela a estratégia *praise* (PR). Esta ação é utilizada para elogiar o usuário e fazer com que a relação entre o usuário e o sistema aumente. No caso, a ação possui um pré-condição chamada *low-rapport*, que serve para indicar que a relação entre o usuário e o sistema está baixa. Nesta ação temos o efeito que indica que a relação entre o usuário e o sistema aumentou e foi mantida (*not (low-rapport)*),

(*rapport-increased*), (*rapport-maintained*)) e também temos que a intensão do sistema foi executada, como indicam os predicados (*pr-system*) e (*pr-history*).

Figura 15 – Ação Praise utilizada para elogiar o usuário.

```
(:action Praise
  :precondition (... , (low-rapport), ...)

  :effect ( and not(low-rapport),
            (pr-system),
            (pr-history),
            (rapport-increased),
            (rapport-maintained), ... )
)
```

Fonte: adaptado de (Romero *et al.*, 2017)

O trabalho de (Romero *et al.*, 2017) usa *planejamento automatizado* para a tarefa de escolher uma estratégia de comunicação com o usuário, o que pode ser uma das tarefas de um gerenciador de diálogos. Para isso, utiliza planejamento determinístico o que se diferencia também da abordagem aqui proposta que utiliza planejamento não-determinístico.

3.5 Comparação entre os trabalhos

Na Tabela 1 é feita a comparação entre os trabalhos relacionados destacando a diferença entre eles na tarefa que a abordagem de planejamento automatizado assume no sistema de diálogo bem como o tipo de domínio considerado.

Tabela 1 – Comparação entre os trabalhos relacionados.

Trabalho	Tarefa	Domínio	Ferramentas
(Teixeira, 2022)	Gerenciamento do diálogo	Não-determinístico	Planejador PRP;
(Muisse <i>et al.</i> , 2019)	Gerenciamento do diálogo	Não-determinístico	Planejador PRP; <i>Watson Assistant</i>
(Pallagani; Srivastava, 2021)	Recuperação de informações	Determinístico	Planejador da nuvem Solver.Planning.Domains
(Romero <i>et al.</i> , 2017)	Escolher estratégia de comunicação com o usuário	Determinístico	não menciona o planejador utilizado.
Este trabalho	Gerenciamento do diálogo	Não-determinístico	Planejador PRP; Rasa

Fonte: Elaborado pelo autor

4 GERENCIAMENTO DO CHATBOT PLANTÃO CORONAVÍRUS USANDO PLANEJAMENTO AUTOMATIZADO

Neste capítulo, mostraremos os detalhes da construção do gerenciador do *chatbot* Plantão Coronavírus utilizando Planejamento Automatizado. Especificamente, iremos mostrar:

- O processo de aquisição do domínio de planejamento gerenciador do *chatbot* a partir dos dados de conversas entre humano (paciente) e *bot* e entre humano (paciente) e humano (profissional de saúde) e;
- A arquitetura do gerenciador de diálogo, bem como o seu funcionamento e integração com o sistema de conversação.

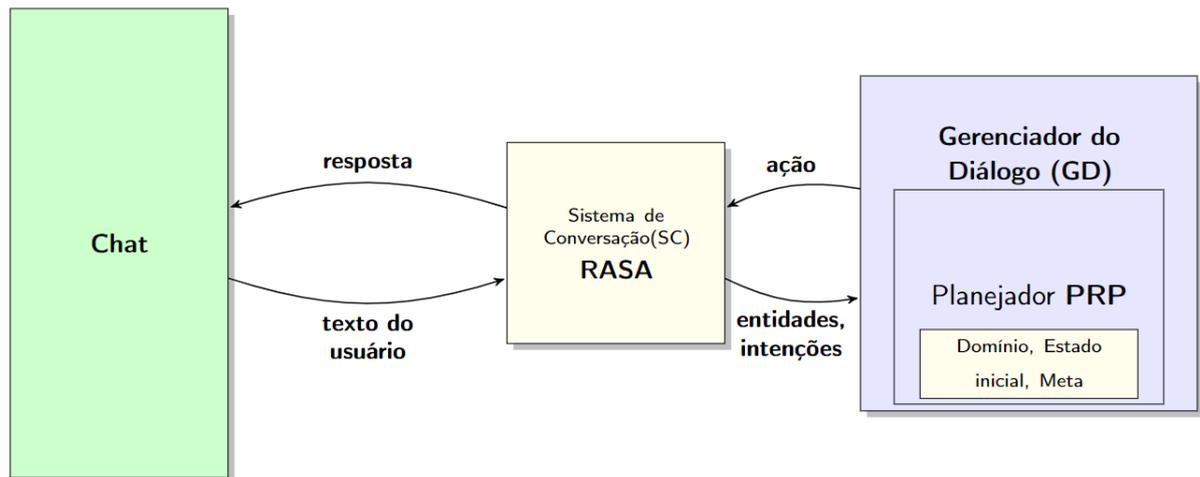
O processo de aquisição do domínio de planejamento do agente gerenciador de diálogo para o *chatbot* Plantão Coronavírus resultou na definição dos predicados e as ações na linguagem PDDL. Para isso, utilizamos a abordagem de aquisição de domínios com **codificação manual**, apresentada na Seção 2.3.2, a partir dos dados de conversas do *chatbot* disponíveis em linguagem natural. Tais dados são sigilosos e, assim, para preservação da privacidade dos usuários, os exemplos apresentados neste trabalho são fictícios, criados para fins ilustrativos, porém representam os dados reais coletados.

Além disso, nossa proposta de implementação da arquitetura do *chatbot* utiliza planejamento automatizado na tarefa de gerenciamento do diálogo e é baseada na modelagem de ações proposta por Muise *et al.* (2019). Neste caso, utilizamos:

- a ferramenta Rasa (Rasa, 2022), 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.

A Figura 16 ilustra a arquitetura do sistema proposto em que o Rasa assume o papel do *Serviço de Conversação* (SC) e o planejador PRP é utilizado para compor o *Gerenciador do Diálogo* (GD). Nesta arquitetura, o Rasa é responsável por realizar a classificação das *intenções* e das *entidades* a partir das declarações do usuário e, também, transformar as ações vindas do GD em respostas para o usuário. O GD fornecerá como entrada para o planejador PRP: o domínio, o estado atual do mundo e a meta do usuário. O planejador obterá uma política, a qual é então enviada para o GD. Este, por sua vez, utilizará a política como guia para gerenciar o diálogo.

Figura 16 – Arquitetura proposta para o Sistema de Diálogo, utilizando o Rasa como SC e o Planejador como GD.



Fonte: elaborado pelo autor

4.1 Aquisição do Domínio de Planejamento para Gerenciamento do *ChatBot* Plantão Coronavírus

Conforme mencionado anteriormente, optamos por uma abordagem de aquisição manual do domínio de planejamento. Neste processo, foram utilizados dados reais provenientes de interações entre usuários com o *chatbot* Plantão Coronavírus. Esses dados foram analisados para identificar quais ações e predicados do domínio modelado em PDDL. Nesta seção forneceremos uma explicação mais detalhada sobre o processo utilizado para aquisição deste domínio.

4.1.1 Descrição do Dataset de Conversação do *ChatBot* Plantão Coronavírus

Este *dataset* contém registros de interações de um *chatbot* com usuários durante os anos de 2020 a 2021.

Este *dataset* representa um conjunto de mensagens enviadas por um *bot* para diferentes usuários, organizadas por sessão e sequência de envio. As mensagens incluem saudações, explicações sobre privacidade e perguntas sobre sintomas de saúde. A estrutura dos dados facilita a análise de conversas individuais e a compreensão da interação do *bot* com os usuários. O *dataset* possui as seguintes colunas:

- id: Identificação única do registro de interação.
- cri: Data e hora da criação do registro.
- atu: Data e hora da última atualização do registro.
- idp: Identificação única do usuário.

- ori: Informação sobre a origem da mensagem (se é do *bot* ou do usuário).
- txt: Conteúdo da mensagem.

O *dataset* é composto por 3000 diálogos, no qual foi realizada uma análise manual de 40 diálogos. Os dados contidos no *dataset* foram utilizados para analisar o comportamento do *chatbot* e para a modelagem do domínio em PDDL.

4.1.2 Obtenção dos Predicados e Ações do Domínio a partir dos Dados Textuais de Diálogo

De modo geral, o processo de aquisição ocorreu como descrito a seguir:

- os *nomes das ações* foram extraídos dos *textos enviado pelo chatbot* ao usuário;
- os *predicados* foram extraídos tanto das *declarações do usuário*, os quais denominamos de *predicados do usuário*, como dos nomes das ações, os quais denominados de *predicados de controle de fluxo* e;
- as *precondições e efeitos das ações*, por sua vez, foram extraídos dos fluxos de diálogos.

Os *predicados do usuário* são obtidos analisando as *intenções* e as *entidades* contidas nas *declarações do usuário*, de modo que para cada intenção e entidade temos um predicado relacionado. Por exemplo, a declaração do usuário "*Meu nome é Bruno da Silva Pinho.*" corresponde à *intenção "send-user-name"* e o texto "*Bruno da Silva Pinho*" é uma *entidade "\$have-name"*. Assim, introduzimos dois predicados "*SEND-USER-NAME*"¹ e "*HAVE-NAME*" no domínio de planejamento. Tais predicados tornam-se verdadeiros quando temos que no diálogo a intenção e a entidade ocorreram. A Tabela 2 contém exemplos de mapeamento de declarações do usuário para predicados do usuário.

Tabela 2 – Mapeamento de declarações do usuário para predicados.

Declaração do Usuário	Intenção	Entidade	Predicados do Usuário
"Meu nome é Bruno da Silva Pinho"	send-info-name	\$have-patient-name	SEND-INFO-NAME, HAVE-PATIENT-NAME
"Sou Mulher"	send-info-gender	\$have-patient-gender-f	SEND-INFO-GENDER, HAVE-PATIENT-GENDER-F
"Prefiro não dizer meu genero"	send-info-gender		SEND-INFO-GENDER
"Bom dia"	user-initiative		USER-INITIATIVE
"Gostaria de um atendimento online"	can-start-dialog		CAN-START-DIALOG
"Estou com falta de ar"	send-info-symptoms	\$have-patient-symptoms	SEND-INFO-SYMPTOMS, HAVE-PATIENT-SYMPTOMS
...

Fonte: Elaborado pelo autor

Os *nomes das ações* do domínio de planejamento são extraídos dos *textos do chatbot*,

¹ Utilizamos letras maiúsculas para os predicados com o objetivo de diferenciá-los das entidades e intenções contidas das declarações dos usuários.

bem como a classificação em ação *determinística* ou *não-determinística*. Os textos do *chatbot* que são perguntas ao usuário ou escolhas que o usuário deve fazer correspondem a *ações não-determinísticas*, uma vez que o usuário pode responder as perguntas ou fazer as escolhas de diversas maneiras possíveis. Já os textos que não requerem que o usuário declare algo correspondem a *ações determinísticas*. O processo inicia com a **atribuição de rótulos para as ações que representa cada texto**. Por exemplo, o texto (“*Qual o seu nome completo?*”) é rotulado como o nome de ação (“ask-user-name”), ou seja, quando o GD manda para SC a ação (“ask-user-name”), o SC deve mandar para o usuário o texto (“*Qual o seu nome completo?*”). Com a classificação dos textos em ações temos um mapeamento de um para um de textos do *chatbot* em ações. A Tabela 3 mostra alguns exemplos deste mapeamento.

Tabela 3 – Mapeamento de textos do *chatbot* para ações.

texto do chatbot	ação	tipo
"Qual o seu nome completo? "	ask-user-name	não-determinística
"Não conseguir entender o que você quis dizer"	error-treatment	determinística
"Olá! Bem-vindo(a) ao serviço de atendimento online."	show-welcome-message	determinística
"Do que você precisa agora? - Atendimento online sobre o novo Coronavírus (COVID-19) - Informações sobre o novo Coronavírus (COVID-19) - Cuidados com a Saúde Mental - Outras informações"	start-dialog	não-determinística
...

Fonte: Elaborado pelo autor

Com os nomes das ações definidos, obtemos os *predicados de controle de fluxo*. Para cada ação do domínio, introduzimos um predicado *CAN-DO-* mais o nome da ação com o objetivo de controlar o fluxo do diálogo. Por exemplo, para a ação *ask-patient-symptoms* é criado um predicado *CAN-DO-ASK-PATIENT-SYMP TOMS* referente àquela ação. Toda ação do domínio possui um predicado de controle de fluxo, exceto a primeira ação que é disparada por meio do predicado *USER-INITIATIVE*, o qual torna-se verdadeiro quando o usuário inicia o diálogo com o *chatbot*.

As *pré-condições* e os *efeitos* das ações são extraídos a partir dos *fluxos dos diálogos*. Os fluxos do diálogo são conversas entre o *chatbot* e o usuário. No Exemplo 4.1.1 temos um fluxo de diálogo em que o usuário inicia o diálogo em busca de um atendimento online.

Exemplo 4.1.1 *Um fluxo de diálogo.*

USUÁRIO: *Olá!*

AGENTE: *Olá! Bem-vindo(a) ao serviço de atendimento online.*

AGENTE: *Do que você precisa agora?*

1. *Atendimento online sobre o novo Coronavírus (COVID-19)*
2. *Informações sobre o novo Coronavírus (COVID-19)*
3. *Cuidados com a Saúde Mental*
4. *Outros*

USUÁRIO: *1. Gostaria de um Atendimento Online.*

AGENTE: *É muito importante responder todas as perguntas e completar o atendimento. Ao final, se for necessário, uma equipe de saúde estará pronta para conversar e cuidar de você.*

AGENTE: *Você está com algum desses sintomas?*

1. *Falta de ar*
2. *Mal-estar intenso*
3. *Tosse incontrolável*
4. *Alargamento das narinas durante a respiração*
5. *Febre acima de 37.8C persistente por mais de 48h*
6. *Confusão mental*
7. *Nenhum dos sintomas acima.*

USUÁRIO: *Falta de ar, Mal-estar intenso e Febre acima de 37.8*

AGENTE: *Há quantos dias os sintomas começaram?*

...

Fonte: elaborado pelo autor

Na Tabela 4 destacamos na coluna *Ação* as ações correspondentes ao texto que o bot emitiu no fluxo do diálogo do Exemplo 4.1.1 e na coluna *Predicados do Usuário* os predicados do usuário que se tornaram verdadeiros neste fluxo de diálogo.

Tabela 4 – Fluxo de ações e predicados do usuário

Agente	Ação	Predicados do Usuário
USER		USER-INITIATIVE
BOT	show-welcome-message	
BOT	start-dialog	
USER		CAN-START-ONLINE-SERVICE
BOT	start-online-service	
BOT	ask-patient-symptoms	
USER		SEND-PATIENT-SYMPTOMS, HAVE-PATIENT-SYMPTOMS
BOT	ask-patient-how-many-days-symptoms	

Fonte: Elaborado pelo autor

Como mencionado anteriormente, toda ação possui um *predicado de controle de fluxo* correspondente e para que essa ação ocorra este predicado deve ser verdadeiro. Assim, este predicado deve fazer parte da pré-condição da ação. A Figura 17 ilustra a ação *ask-patient-symptoms* na qual introduzimos o *predicado de controle de fluxo CAN-DO-ASK-PATIENT-SYMP TOMS* em suas pré-condições.

Figura 17 – Exemplo de pré-condição de uma ação no domínio gerenciador do chatbot Plantão Coronavírus.

```

; E.g, "Voce esta com algum desses sintomas?
... "
(:action ask-patient-symptoms
  :precondition ( and
                 (CAN-DO-ASK-PATIENT-SYMP TOMS)
                 ...
               )
  :effect ( oneof
           ...
         )
)

```

Fonte: elaborado pelo autor

Com os fluxos de diálogos, podemos verificar a sequência na qual os predicados tornam-se verdadeiros ou falsos. Por exemplo, no fluxo da Tabela 4 temos que a ação *ask-patient-how-many-days-symptoms* só pode ocorrer se a ação *ask-patient-symptoms* tiver ocorrido antes. Deste modo, podemos dizer que a ação anterior possui um efeito que torna o predicado da pré-condição da ação posterior verdadeiro. Para exemplificar, suponha que há duas ações *A* e *B*, de modo que a ação *B* ocorra após a ação *A* nos fluxos dos diálogos. Deste modo a ação *A* deve conter um efeito *e*, em que a pré-condição *CAN-DO-B* de *B* seja verdadeira. Sendo assim é adicionado na ação *A* um novo efeito *e* onde o predicado *CAN-DO-B* deve ser verdadeiro. Aplicando este raciocínio para as ações *ask-patient-how-many-days-symptoms* e *ask-patient-symptoms*, conseguimos os resultado que pode ser observado na Figura 18.

Figura 18 – Exemplo de efeitos de uma ação.

```

(:action A
  :precondition ( and
                 (CAN-DO-A)
                 ...
               )
  :effect ( oneof
           ;e
           ( and
             (CAN-DO-B)
             ...
           )
           ...
        )
)
(:action B
  :precondition ( and
                 (CAN-DO-B)
                 ...
               )
  :effect ( oneof
           ...
        )
)

```

Fonte: elaborado pelo autor

Além disso, adicionamos predicados nas pré-condições e efeitos das ações também observando as entradas dos usuários que estão entre duas ações. Temos que os predicados provenientes da entrada do usuário são adicionados no efeito da ação anterior que contem o predicado *CAN-DO-** da ação posterior, e também, é adicionado como pré-condição da ação posterior. Na Tabela 4 entre as ações *ask-patient-symptoms* e *ask-patient-how-many-days-symptoms* temos os predicados *SEND-PATIENT-SYMPTOMS* e *HAVE-PATIENT-SYMPTOMS* provindos do texto do usuário. Deste modo adicionamos estes predicados no efeito da ação *ask-patient-symptoms* que contem o predicado *CAN-DO-ASK-PATIENT-HOW-MANY-DAYS-SYMPTOMS*, e também os adicionamos nas pré-condições da ação *ask-patient-how-many-days-symptoms*, como pode ser observado na Figura 19.

Figura 19 – Exemplo de adição de predicados do usuário nas pré-condições e efeitos das ações.

```

(:action ask-patient-symptoms
  :precondition ( and
    (CAN-DO-ASK-PATIENT-SYMP TOMS)
    ...
  )
  :effect ( oneof
    ;e
    ( and
      (CAN-DO-ASK-PATIENT-HOW-MANY-DAYS-SYMP TOMS)
      (SEND-PATIENT-SYMP TOMS)
      (HAVE-PATIENT-SYMP TOMS)
    )
    ...
  )
)

(:action ask-patient-how-many-days-symptoms
  :precondition ( and
    (CAN-DO-ASK-PATIENT-HOW-MANY-DAYS-SYMP TOMS)
    (SEND-PATIENT-SYMP TOMS)
    (HAVE-PATIENT-SYMP TOMS)
  )
  :effect ( oneof
    ...
  )
)

```

Fonte: elaborado pelo autor

Na Tabela 5 temos o exemplo da obtenção das pré-condições e efeitos dos fluxos de diálogo do Exemplo 4.1.1.

Tabela 5 – Mapeamento de pré-condições e efeitos de um fluxo de diálogo.

Talker	Ação	Precondição	Efeitos	Predicados do usuário
USER				USER-INITIATIVE
BOT	show-welcome-message	USER-INITIATIVE	0:[CAN-DO-START-DIALOG]	
BOT	start-dialog	CAN-DO-START-DIALOG	0:[CAN-DO-START-ONLINE-SERVICE, CAN-START-ONLINE-SERVICE]	
USER				CAN-START-ONLINE-SERVICE
BOT	start-online-service	CAN-DO-START-ONLINE-SERVICE, CAN-START-ONLINE-SERVICE	0:[CAN-DO-ASK-PATIENT-SYMP TOMS]	
BOT	ask-patient-symptoms	CAN-DO-ASK-PATIENT-SYMP TOMS	0:[CAN-DO-ASK-PATIENT-HOW-MANY-DAYS-SYMP TOMS, SEND-PATIENT-SYMP TOMS, HAVE-PATIENT-SYMP TOMS]	
USER				SEND-PATIENT-SYMP TOMS, HAVE-PATIENT-SYMP TOMS
BOT	ask-patient-how-many-days-symptoms	CAN-DO-ASK-PATIENT-HOW-MANY-DAYS-SYMP TOMS, SEND-PATIENT-SYMP TOMS, HAVE-PATIENT-SYMP TOMS		

Fonte: Elaborado pelo autor

Assim como todo processo de modelagem de domínios de planejamento, seja ele automático ou manual, alguns refinamentos são feitos a posteriori, como por exemplo removendo

predicados que são redundantes ou adicionando predicados para um maior controle das ações.

4.1.3 O Domínio de Planejamento Gerenciador do ChatBot da COVID

Nesta seção apresentaremos o domínio gerenciador do *chatbot*, resultante do processo de modelagem descrito anteriormente. Foram obtidos **89 predicados e 60 ações**.

Em relação aos predicados, temos no domínio:

- Predicados correspondentes a entidades, os quais são iniciados com a palavra *HAVE-**. Estes predicados tornam-se verdadeiros quando a entidade é obtida. Por exemplo, o predicado *HAVE-NAME* torna-se verdadeiro quando o nome do usuário é obtido no diálogo.
- Predicados correspondentes a intenções, os quais são divididos em 6 grupos:
 1. Os que iniciam com *SEND-INFO-** tornam-se verdadeiros quando temos intenções em que o usuário deseja enviar alguma informação. Por exemplo, o predicado *SEND-INFO-NAME*, no qual este predicado torna-se verdadeiro quando o usuário tem a intenção de enviar informação do seu nome.
 2. Os que iniciam com *CAN-SHOW-INFO-** tornam-se verdadeiros quando temos intenções em que o usuário deseja ver alguma informação. Por exemplo, o predicado *CAN-SHOW-INFO-MAIN-SYMPTOMS*, onde o usuário tem a intenção de ver os principais sintomas da doença.
 3. Os que iniciam com *CAN-BACK-** tornam-se verdadeiros quando temos intenções em que o usuário deseja voltar para um determinado ponto do diálogo. Em nossa modelagem possuímos apenas um predicado deste tipo que é *CAN-BACK-DIALOG*, e o mesmo serve para o usuário retorna para o menu principal do chat, quando o mesmo tem esta intenção.
 4. Os que iniciam com *CAN-GO-ERROR-** tornam-se verdadeiros quando temos alguma intenção do usuário que não é classificável pelo *chatbot*, ou seja, algo que não foi possível o agente entender, e que está fora do contexto do diálogo. Em nossa modelagem possuímos apenas um predicado deste tipo que é *CAN-GO-ERROR-TREATMENT*.
 5. Os que iniciam com *CAN-START-** tornam-se verdadeiros quando o usuário quer iniciar um serviço. Por exemplo, uma consulta online que em nossa modelagem utilizamos o predicado *CAN-START-ONLINE-SERVICE*.
 6. Predicado *USER-INITIATIVE* é verdadeiro quando o usuário tem a intenção de iniciar

uma conversa com o agente de diálogo.

- Predicados de controle de fluxo. Estes predicados começam com a palavra *CAN-DO*, e são responsáveis por bloquear uma ação até que outra ação tenha sido executada, ou até que alguma informação desejada tenha sido capturada. Por exemplo, o predicado *CAN-DO-ASK-PATIENT-INFO-BIRTHDAY*.
- Predicado de chamada de sistemas externos: Estes predicados iniciam com *CAN-CALL*, e são verdadeiros quando o agente precisa fazer uma chamada de serviço externo. Por exemplo, chamar um sistema de diagnóstico, que em nossa modelagem utilizamos o predicado *CAN-CALL-DIAGNOSTIC-SYSTEM*.
- Predicados que indicam que o diálogo iniciou. Este predicado possui o nome *STARTED* e torna-se verdadeiro quando é iniciado o diálogo.
- Predicado *GOAL*: Este predicado se torna verdadeiro quando se chega a um estado objetivo, ou seja quando se é alcançado o objetivo. Este predicado também simboliza que chegamos ao fim do diálogo.

Em relação às ações, temos no domínio:

- Ações que iniciam com *ask-**. Estas ações indicam que o agente precisa fazer uma pergunta para o usuário. A Figura 20 mostra um exemplo de uma ação deste tipo. As ações *ask* são ações não-determinísticas, pois requerem uma entrada do usuário e este pode ou não digitar o que se espera receber.

Figura 20 – Exemplo de ação não-determinística *ask*-*.

```

; E.g, "Qual o seu nome completo?"
(:action ask-patient-info-name
  :precondition ( and
    (CAN-DO-ASK-PATIENT-INFO-NAME)
    ( not (HAVE-PATIENT-NAME))
    ( not (SEND-INFO-NAME))
    ( not (CAN-GO-ERROR-TREATMENT))
  )
  :effect ( oneof
    ;E.g., "Meu nome e Bruno da Silva Pinho"
    ( and
      (HAVE-PATIENT-NAME)
      (SEND-INFO-NAME)
      (CAN-DO-ASK-PATIENT-INFO-BIRTHDAY)
    )
    ;E.g., "fsfsffsfsf"
    ( and
      (CAN-GO-ERROR-TREATMENT)
    )
  )
)
)

```

Fonte: elaborado pelo autor

- ações que iniciam com *show*-. Estas ações indicam que o agente precisa mostrar alguma informação para o usuário. A Figura 21 mostra um exemplo de uma ação, a qual mostra um texto com as informações sobre os sintomas da COVID. Essa ação também é não-determinística representando como o usuário pode responder a pergunta, voltando para o menu inicial onde poderá interagir mais com o agente ou optando por finalizar o diálogo, se o objetivo é apenas visualizar as informações sobre os sintomas da doença.

Figura 21 – Exemplo de ação não-determinística *show-**.

```

; E.g, "Os principais sintomas da doenca sao ..."
(:action show-info-about-main-symptoms
  :precondition ( and
    (CAN-SHOW-INFO-MAIN-SYMPTOMS)
    ( not(CAN-GO-ERROR-TREATMENT))
  )
  :effect ( oneof
    ;E.g., "voltar para o menu inicial"
    ( and
      (CAN-START-DIALOG)
      ( not (CAN-GO-ERROR-TREATMENT))
      (CAN-BACK-DIALOG)
      ( not(CAN-SHOW-INFO-MAIN-SYMPTOMS))
    )
    ;E.g., "finalizar a conversa"
    ( and
      (CAN-SHOW-INFO-END-CONVERSATION)
      ( not (CAN-SHOW-INFO-MAIN-SYMPTOMS))
    )
  )
)
)

```

Fonte: elaborado pelo autor

- ações que iniciam com *error-**. Estas ações indicam que o agente deve mostrar uma mensagem de que não entendeu a declaração do usuário. Esta é uma ação determinística como mostrado na Figura 22, representando que o agente deve realizar um tratamento de erro.

Figura 22 – Exemplo de ação determinística *error-**.

```

; E.g, "Nao conseguir entender o que voce quis dizer"
(:action error-treatment
  :precondition ( and
    (CAN-GO-ERROR-TREATMENT))
  )
  :effect (
    ( not (CAN-GO-ERROR-TREATMENT))
  )
)
)

```

Fonte: elaborado pelo autor

- ações que iniciam com *call*-. Estas ações indicam que o agente precisa fazer uma chamada de sistema. A Figura 23 mostra uma ação na qual é feita uma chamada a um sistema externo de diagnósticos. Nesta ação temos que os efeitos não determinísticos representam se o sistema diagnosticou a pessoa com COVID ou não.

Figura 23 – Exemplo de ação *call*-* não-determinística do domínio da COVID.

```
(:action call-diagnostic-system
  :precondition ( and
    (CAN-CALL-DIAGNOSTIC-SYSTEM)
    ( not (HAVE-DIAGNOSTIC-SYSTEM))
    ( not (CAN-GO-ERROR-TREATMENT))
    (HAVE-PATIENT-SYMPTOMS)
    (HAVE-PATIENT-DAYS-SYMPTOMS)
  )
  :effect ( oneof
    ( and
      (HAVE-DIAGNOSTIC-SYSTEM)
      ( not (CAN-CALL-DIAGNOSTIC-SYSTEM))
      (CAN-DO-ASK-LOCATION)
    )
    ( and
      (NOT (HAVE-DIAGNOSTIC-SYSTEM))
      (CAN-SHOW-INFO-END-CONVERSATION)
      ( not (CAN-CALL-DIAGNOSTIC-SYSTEM))
    )
  )
)
```

Fonte: elaborado pelo autor

- ação *finish-service*. Esta ação determinística indica que o serviço de diálogo será finalizado. A Figura 24 ilustra as pré-condições e efeitos dessa ação.

Figura 24 – Exemplo de ação *finish-service*.

```
(:action finish-service
  :precondition ( and
    (CAN-DO-FINISH-SERVICE)
    ( not(GOAL))
  )
  :effect (
    ( and
      (GOAL)
    )
  )
)
```

Fonte: elaborado pelo autor

- ação *health-agent-takes-control*. Esta ação não-determinística indica que um agente de saúde humano vai assumir o lugar do agente virtual no diálogo com o usuário. Observe na Figura 25 que um efeitos representa o fato de o agente de saúde humano ainda está assumindo o lugar do agente virtual, e o outro efeito representa que o atendimento com o agente de saúde humano terminou e o diálogo pode ser finalizado.

Figura 25 – Exemplo de ação *health-agent-takes-control*.

```
(:action health-agent-takes-control
  :precondition ( and
    (CAN-DO-HUMAN-TAKE-CONTROL-DIALOG)
    ( not(CAN-GO-ERROR-TREATMENT))
  )
  :effect ( oneof
    ( and
      (CAN-DO-HUMAN-TAKE-CONTROL-DIALOG)
    )

    ( and
      (CAN-SHOW-INFO-END-CONVERSATION)
      ( not (CAN-DO-HUMAN-TAKE-CONTROL-DIALOG))
    )
  )
)
```

Fonte: elaborado pelo autor

- ação *start-dialog*. Esta ação não-determinística representa o menu inicial de opções. Nesta ação temos que os quatro primeiros efeitos não determinísticos representam opções do

menu e o ultimo efeito representa que o usuário digitou algo que o agente não entendeu, como pode ser visto na Figura 26.

Figura 26 – Exemplo de ação não-determinística *start-dialog*.

```

; E.g, Do que voce precisa agora?
(:action start-dialog
  :precondition ( and
    (STARTED)
    (CAN-DO-START-DIALOG)
    ( not (CAN-GO-ERROR-TREATMENT) )
  )
  :effect ( oneof
    ; E.g, 1. Atendimento online sobre o novo
      Coronavirus
    ( and
      (CAN-START-ONLINE-SERVICE)
      ( not (CAN-DO-START-DIALOG) )
      ( not (CAN-BACK-DIALOG) )
    )
    ; E.g, 2. Informaces sobre o novo Coronavirus
    ( and
      (CAN-SHOW-INFO-ABOUT-COVID)
      ( not (CAN-DO-START-DIALOG) )
      ( not (CAN-BACK-DIALOG) )
    )
    ; E.g, 3. Cuidados com a Saude Mental
    ( and
      (CAN-SHOW-INFO-MENTAL-HEALTH)
      ( not (CAN-DO-START-DIALOG) )
      ( not (CAN-BACK-DIALOG) )
    )
    ; E.g, 4. Outros
    ( and
      (CAN-SHOW-INFO-OTHERS)
      ( not (CAN-DO-START-DIALOG) )
      ( not (CAN-BACK-DIALOG) )
    )

    ( and
      (CAN-DO-START-DIALOG)
      (CAN-GO-ERROR-TREATMENT)
    )
  )
)

```

Fonte: elaborado pelo autor

- ação *start-online-service*. Esta ação determinística representa o início de um atendimento online. Observe na Figura 27 que o efeito da ação contém o predicado *CAN-DO-ASK-PATIENT-SYMPTOMS*, sinalizando que a ação de perguntar os sintomas para o usuário pode ser executada em seguida.

Figura 27 – Exemplo de ação determinística *start-online-service*.

```
(:action start-online-service
  :precondition ( and
    (CAN-START-ONLINE-SERVICE)
    ( not (CAN-GO-ERROR-TREATMENT))
  )
  :effect ( oneof
    ( and
      ( not (CAN-START-ONLINE-SERVICE))
      (CAN-DO-ASK-PATIENT-SYMPTOMS)
    )
  )
)
```

Fonte: elaborado pelo autor

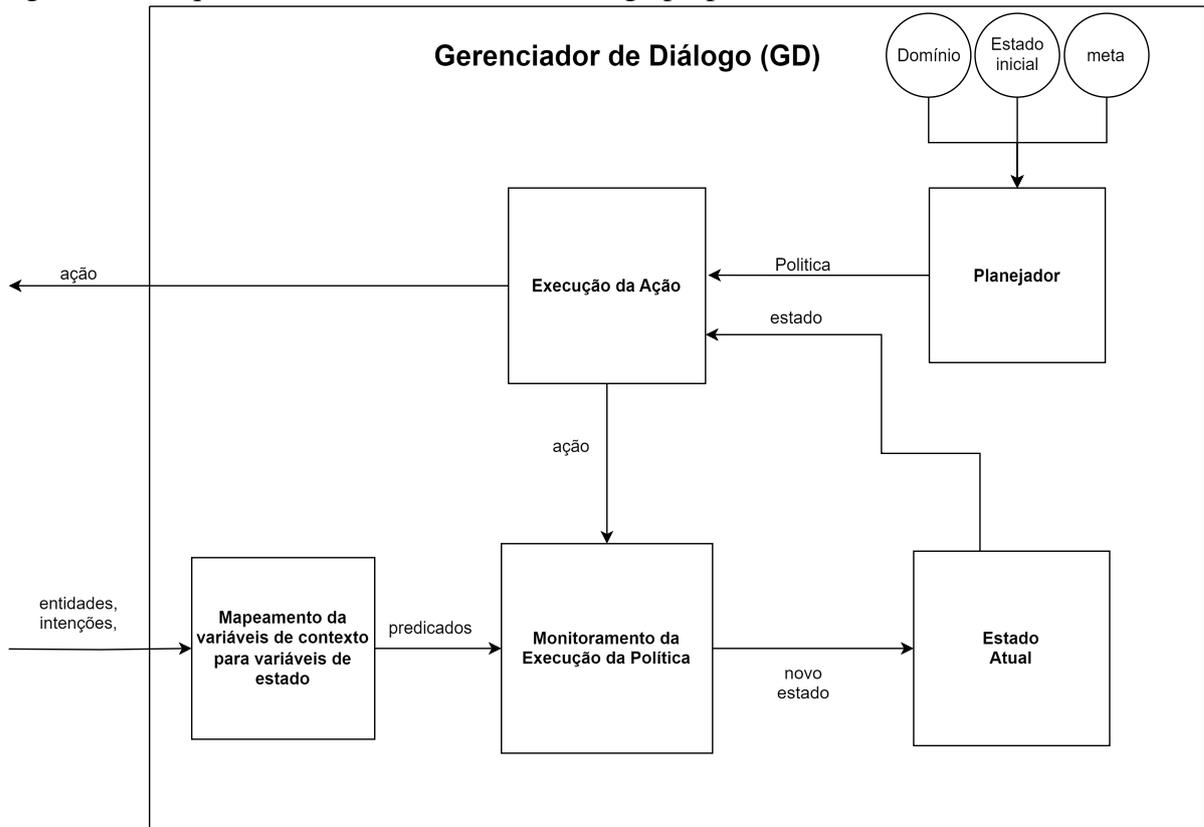
O domínio completo pode ser acessado no seguinte link: <<https://github.com/brunopinho321/Domain-Chatbot-Covid/blob/main/Domain.pddl>>

4.2 Arquitetura do Gerenciador do Diálogo baseado em Planejamento Automatizado

A Figura 28 ilustra a arquitetura de funcionamento do GD proposto neste trabalho. A arquitetura possui cinco módulos, a saber:

1. **Planejador:** Gera a política que conterà as ações que o *chatbot* irá executar.
2. **Estado Atual:** Representa o conjunto de predicados que são verdadeiros naquele momento do diálogo.
3. **Execução da Ação:** Utiliza as informações contidas no estado atual e na política para escolher qual ação deve ser executada e enviá-la ao SC.
4. **Mapeamento de variáveis de contexto para variáveis de estado:** Responsável por relacionar as intenções e entidades aos predicados.
5. **Monitoramento da Execução da Política:** Responsável por monitorar qual efeito da ação ocorreu utilizando as informações provenientes do SC e atualizar o Estado Atual.

Figura 28 – Arquitetura do Gerenciador de Diálogo proposto neste trabalho.



Fonte: Elaborado pelo autor

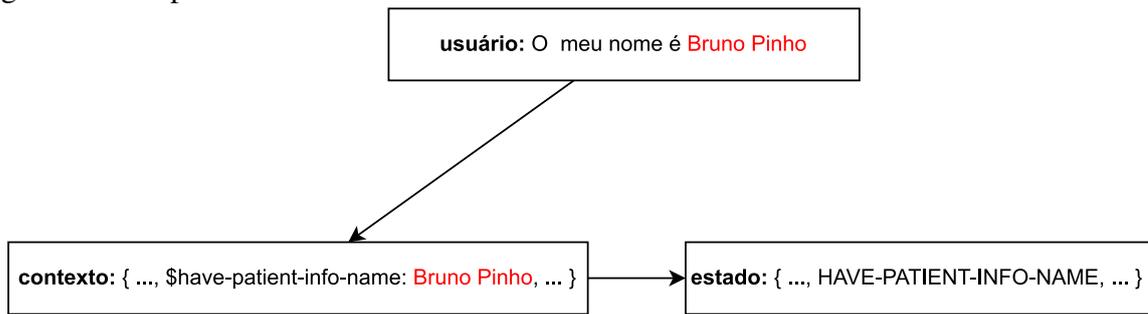
A seguir, serão descritos os detalhes do funcionamento do gerenciador do diálogo.

4.2.1 Mapeamento de Variáveis de Estado e Variáveis de Contexto

No gerenciamento do *chatbot* as duas principais tarefas são: geração da política, o que é feita pelo planejador, e execução/monitoramento da política gerada que consiste em escolher as ações que o *chatbot* irá realizar com base nas entradas do usuário. Assim, é preciso ter uma correspondência entre *estado* e *contexto*. No estado temos um conjunto de predicados que são verdadeiros e são utilizados para decidir qual ação da política será executada. Deste modo, realizamos um mapeamento entre as *variáveis de estado* e as *variáveis de contexto*. Este mapeamento é essencial para decidir com base na entrada do usuário quais predicados se tornaram verdadeiros, pois quando algum valor de intenção ou de entidade é obtido, temos que o predicado correspondente torna-se verdadeiro.

Na Figura 29, temos um exemplo de como foi feito este mapeamento: do contexto foi capturado o valor da entidade *\$havepatient-info-name* e; no estado temos que essa entidade é representada por um predicado *HAVE-PATIENT-INFO-NAME*.

Figura 29 – Mapeamento da variáveis de contexto e variáveis de estado.

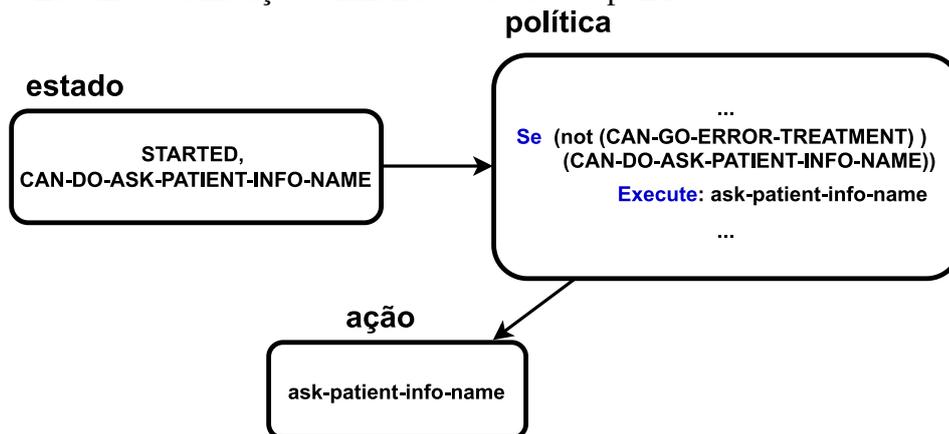


Fonte: elaborado pelo autor

4.2.2 Execução da Ação

Para escolher a ação que será executada pelo Rasa utilizamos a informação do estado atual e a política elaborada pelo planejador. A Figura 30 ilustra um exemplo de como a ação é escolhida. Nesta figura, temos o **estado** que contém os predicados {STARTED, CAN-DO-ASK-PATIENT-INFO-NAME} que são verdadeiros em um momento do diálogo. Com base nestes predicados, buscamos na **política** qual **ação** deve ser executada. A Figura 30 mostra que, de acordo com a política, para executar a **ação** start-dialog é necessário que o predicado CAN-GO-ERROR-TREATMENT seja falso e que o predicado CANT-DO-ASK-PATIENT-INFO-NAME seja verdadeiro.

Figura 30 – Escolha de uma ação utilizando o estado e a política



Fonte: elaborado pelo autor

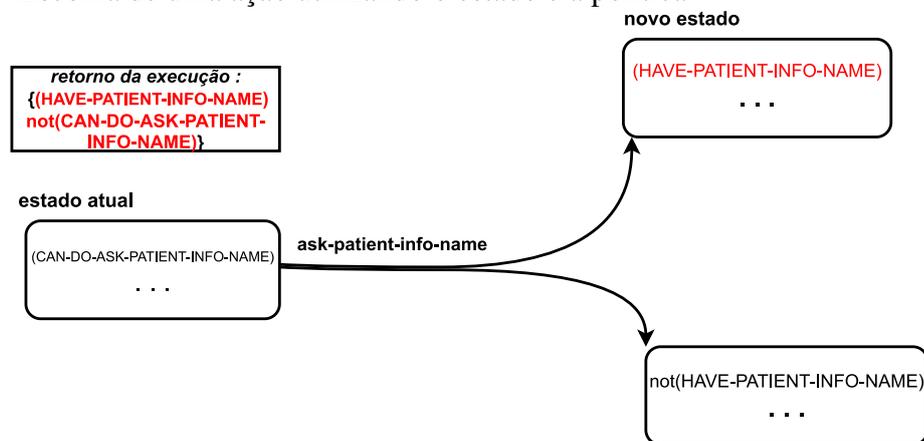
Com a ação escolhida, ela é enviada para o Rasa onde é transformada em texto para o usuário. Após a ação ser executada, é esperado um retorno que é uma declaração do usuário. Esta declaração passa pelo Rasa e é feita a classificação das *entidades* e *intenções*, e em seguida, no GD cada *entidade* e *intenção* é associada a um predicado. Este conjunto de predicados

retornado chamamos de *retorno da execução*, e será utilizado para determinar qual dos efeitos da ação não-determinística ocorreu. Por exemplo, considere que a ação `ask-patient-info-name` é transformada pelo Rasa no texto “Qual o seu nome completo?”. O usuário pode responder este texto com a declaração “O meu nome é Bruno Pinho”, no qual o texto “Bruno Pinho” é classificado pelo Rasa como uma entidade *\$have-patient-info-name*. Esta entidade é associada a um predicado `HAVE-PATIENT-INFO-NAME`, que será utilizado para determinar qual efeito ocorreu.

4.2.3 Monitoramento da Execução da Política

O monitoramento de qual efeito ocorreu é feito comprando-se o *retorno da execução* com os efeitos não-determinísticos da ação executada. Com isso, é possível saber qual dos efeitos não-determinísticos ocorreu. Após esse passo, é feita a atualização do estado, onde são adicionados os predicados que se tornaram verdadeiros, e são removidos os que se tornaram falsos, gerando assim um novo estado. Este novo estado será utilizado para escolher qual a próxima ação será executada.

Figura 31 – Escolha de uma ação utilizando o estado e a política



Fonte: elaborado pelo autor

Na Figura 31 temos um estado atual em que a ação não determinística `ask-patient-info-name` foi aplicada. Com o *retorno da execução* que contém os predicados `HAVE-PATIENT-INFO-NAME` e `not(CAN-DO-ASK-PATIENT-INFO-NAME)`, é determinado qual dos efeitos não-determinísticos será utilizado para gerar o novo estado.

4.3 Implementação

Como mencionado anteriormente, no presente trabalho foi utilizada a ferramenta Rasa para o serviço de conversação. O gerenciador do diálogo, por sua vez, foi implementado utilizando a linguagem Python e escolhe ações com base na política criada pelo planejador PRP.

4.3.1 Implementação do Gerenciador de Diálogo (GD)

Aqui apresentaremos a implementação do GD, consistindo em seis funções. A política obtida pelo planejador é utilizada para consultar sobre as ações e seus efeitos não determinísticos.

1. Função de conversão de entidades e intenções para predicados (TO_PREDICATES).

Esta função recebe uma lista de entidades e intenções como entrada e retorna como saída a lista de predicados correspondentes, segundo um mapeamento previamente realizado. No algoritmo apresentado, na linha 2, há uma lista V que contém tanto as intenções quanto as entidades. Na linha 3, é inicializada uma lista vazia de predicados. O código continua na linha 4, onde um loop é iniciado para percorrer todos os elementos contidos em V . Dentro desse loop, na linha 5, cada elemento é transformado em um predicado utilizando um mapeamento, e na linha 6, esse predicado é adicionado à lista *Predicados*. Por fim, na linha 8, a lista completa de predicados, *Predicados*, é retornada como resultado do processo.

Algoritmo 1: TO_PREDICATES

Entrada: Intenções, Entidades

Saída: Predicados

1 **início**

2 $V \leftarrow [Intenes \cup Entidades]$

3 $Predicados = [\emptyset]$

4 **para cada** $v_i \in V$ **faça**

5 $predicado \leftarrow \text{TRANSFORMAEMPREDICADO}(v_i)$

6 $Predicados.ADICIONA(predicado)$

7 **fim**

8 **retorna** $Predicados$

9 **fim**

Fonte: Elaborado Pelo Autor

2. Função de determinar qual efeito da ação não determinística foi realizado (REALIZATION)

- Esta função recebe como entrada uma ação, intenções e entidades e, com base nestas, verifica qual efeito da ação ocorreu. As intenções e entidades são transformadas pelo Algoritmo 1 em predicados e, em seguida, para cada efeito não-determinístico da ação é comparado se todos os predicados derivados da declaração do usuário estão contidos em algum destes efeitos não determinísticos. Se sim, é retornado o efeito não determinístico que contém todos os predicados da declaração do usuário. Neste algoritmo, na linha 2, verifica-se se a ação é determinística. Se for, o efeito da ação é retornado, pois ela possui apenas um efeito. Caso contrário, uma lista *Efeitos* é iniciada, contendo todos os efeitos da *Ação*, conforme mostrado na linha 5. Em seguida, na linha 6, outra lista, *Predicados*, é iniciada para armazenar os predicados que serão convertidos pelo Algoritmo 1, utilizando as *Intenções e Entidades*.

Na linha 7, inicia-se um loop que percorre todos os efeitos pertencentes à lista de *Efeitos*. Dentro deste loop, na linha 8, verifica-se se o efeito contém todos os *Predicados*. Se sim, o efeito é retornado. Caso contrário, o loop continua até percorrer todos os efeitos contidos na lista *Efeitos*, ou até encontrar um efeito que contenha todos os predicados da lista *Predicados*.

Por fim, se não for encontrado um efeito que contenha todos os predicados, é retornado um valor vazio na linha 12.

Algoritmo 2: REALIZATION	
Entrada: Ação , Intenções , Entidades	
Saída: Um Efeito da Ação	
1	início
2	se Ação <i>é determinística</i> então
3	retorna Ação.Efeito
4	fim
5	Efeitos \leftarrow Ação.Efeitos
6	Predicados \leftarrow TO_PREDICATES (Intenções, Entidades)
7	para cada efeito \in Efeitos faça
8	se efeito.CONTEM(Predicados) então
9	retorna efeito
10	fim
11	fim
12	retorna \emptyset
13	fim

Fonte: Elaborado Pelo Autor

3. **Função de atualização do estado do mundo (UPDATE)** - Esta função atualiza o estado do mundo de modo que ela utiliza o algoritmo 2 que retorna qual efeito representa a entrada do usuário. Com este efeito, são adicionados no estado do mundo os predicados que se tornaram verdadeiros e são removidos os que se tornaram falsos. Neste algoritmo, recebemos como entrada um estado do mundo, uma ação, uma lista de intenções e uma lista de entidades. Na linha 2, utilizamos o algoritmo 2 para determinar qual efeito da ação ocorreu, resultando em uma lista de predicados, os quais são armazenados na lista Efeito. Na linha 3, verificamos se a lista Efeito não está vazia. Se estiver vazia, na linha 4, retornamos o Estado inalterado. Em seguida, na linha 6, iniciamos um loop que percorre todos os predicados contidos na lista Efeito. Para cada predicado, na linha 7, verificamos se ele é verdadeiro. Se for, na linha 8, verificamos se o Estado não contém o predicado; se não contiver, o predicado é adicionado ao Estado.
- Na linha 13, verificamos se o Estado já contém o predicado. Se sim, na linha 14, verificamos se este predicado é falso; se for, o predicado é removido do Estado.
- Por fim, na linha 20, retornamos o Estado e o Efeito resultante do processamento.

Algoritmo 3: UPDATE	
Entrada: Estado, Ação, Intenções, Entidades	
Saída: Estado, Efeito	
1	início
2	Efeito ← REALIZATION (Ação, Intenções, Entidades)
3	se Efeito = [∅] então
4	retorna Estado
5	fim
6	para cada predicado ∈ Efeito faça
7	se predicado = <i>Verdadeiro</i> então
8	se !Estado.CONTEM(predicado) então
9	Estado.ADICIONA(predicado)
10	fim
11	fim
12	senão
13	se Estado.CONTEM(predicado) então
14	se predicado = <i>Falso</i> então
15	Estado.REMOVA(predicado)
16	fim
17	fim
18	fim
19	fim
20	retorna Estado, Efeito
21	fim

Fonte: Elaborado Pelo Autor

4. **Função de espera da entrada do usuário WAIT_USER_UTTER** - Esta função aguarda a entrada do usuário e atualizar o estado do mundo, utilizando o algoritmo 3. A função verifica se o estado do mundo está vazio, caso esteja é adicionado ao estado as entradas do usuário convertida para predicados. Caso não esteja vazio, é feita a atualização do estado do mundo com as entradas do usuário. Neste algoritmo, na linha 2, é verificado se o Estado está vazio. Se estiver, na linha 3, é então verificado se a lista de intenções não está vazia. Se isso for verdadeiro, o algoritmo 1 é utilizado para converter as intenções e as

entidades em predicados, os quais são então adicionados ao estado do mundo, representado por Estado. Na linha 6, o Estado é retornado.

Se o Estado não estiver vazio, na linha 8, o algoritmo 3 é utilizado para atualizá-lo, retornando o efeito que ocorreu. Por fim, na linha 9, são retornados o Estado atualizado e o efeito resultante.

Algoritmo 4: WAIT_USER_UTTER

Entrada: Ação, Intenções, Entidades

Saída: Estado do mundo S, Efeito

1 **início**

2 **se** Estado = $[\emptyset]$ **então**

3 **se** Intenções $\neq [\emptyset]$ **então**

4 Estado \leftarrow TO_PREDICATES (Intenções, Entidades)

5 **fim**

6 **retorna** Estado

7 **fim**

8 Estado, Efeito \leftarrow UPDATE (Estado, Ação, Intenções, Entidades)

9 **retorna** Estado, Efeito

10 **fim**

Fonte: Elaborado Pelo Autor

5. **Função que retorna qual ação deve ser executada (ACTION_EXECUTION)**- Está função é responsável por retorna um conjunto de ações a partir da política, de modo que a política escolhe qual ação deve ser executada de acordo com os predicados do estado do mundo. Neste algoritmo temos que é escolhido uma ação, se a mesma for determinística, ela é adicionada em uma lista de ações, e o estado do mundo é atualizado com os efeitos da mesma, e este processo se repete até ser selecionado uma ação não determinística, deste modo temos que a ultima ação da lista de ações é sempre não-determinística. Isto é necessário pois as ações determinísticas não precisam da entrada do usuário para atualizar o estado do mundo. Já a ultima ação por ser não-determinística, é necessário uma entrada do usuário para determinar qual efeito ocorreu. Está ultima ação sempre é passada para o algoritmo 4 que espera entrada do usuário para atualizar o estado do mundo com o efeito adequado a entrada do usuário. Neste algoritmo, na linha 2, selecionamos uma Ação da política com base no estado atual. Em seguida, na linha 3, verificamos se esta Ação está

vazia. Se for o caso, retornamos uma lista vazia conforme indicado na linha 4.

Caso a Ação não seja vazia, na linha 6, verificamos se ela é determinística. Se for, atualizamos o Estado utilizando o Algoritmo 3 e, em seguida, na linha 8, retornamos a Ação e realizamos uma chamada recursiva.

Por fim, se a Ação não for determinística, simplesmente a retornamos, como demonstrado na linha 11.

Algoritmo 5: ACTION_EXECUTION

Entrada: Estado do mundo (Estado), Política gerada pelo planejador não-determinístico

Saída: Uma lista de Ações

```

1 início
2   Ação ← Política.Retorne_Ação(Estado)
3   se Ação = ∅ então
4     retorna [∅]
5   fim
6   se Ação é determinística então
7     Estado ← UPDATE(Estado, Ação, [∅], [∅])
8     retorna [Ação] ∪ ACTION_EXECUTION(Estado, Política)
9   fim
10  senão
11   retorna [Ação]
12  fim
13 fim

```

Fonte: Elaborado Pelo Autor

4.3.2 Integração do Gerenciador de Diálogos ao Sistema de Conversação Rasa

Como mencionado anteriormente, utilizamos o Rasa como SC. Para tal, utilizamos os dados de conversa do Plantão Coronavírus para treinar um modelo para identificar intenções e entidades a partir das entradas do usuário. A seguir, mostraremos como cada um dos módulos do Rasa foram programados para o funcionamento do *chatbot*.

1. **Módulo domain.** A Figura 32 mostra como foi realizada a implementação deste módulo no qual foram identificadas 56 intenções (*user-initiative*, *can-start-online-service*, ...) e 13 entidades (*have-patient-name*, *have-patient-cpf*, ...) e uma ação customi-

zada (`global-action`) para gerenciar os textos que são exibidos para o usuário, a partir das ações provenientes do GD.

Figura 32 – Implementação do modulo domain para o chatbot do Plantão Coronavírus

```

intents:
  - user-initiative
  - can-start-online-service
  - can-show-info-mental-health
  - can-show-info-about-covid
  | | ...

entities:
  - have-patient-name
  - have-patient-cpf
  - have-patient-birthday
  - have-patient-symptoms
  | | ...

actions:
  - global_action

```

Fonte: Elaborado pelo autor

2. **Módulo nlu** - Neste modulo é onde colocamos alguns exemplos de textos que reprimam as entidades e as intenções que estão contidas em nosso domínio. Estes textos de exemplo são provenientes dos dados de conversa entre paciente e *bot*. Assim, para cada entidade e intenção contida no módulo domain, temos que dar exemplos de texto para que os algoritmos do rasa possam aprender a identificar as entidades e intenções a partir de novas entradas do usuário, ou em outras palavras, aprende a entender os textos de entrada do usuário. Na Figura 33 temos dois exemplos ilustrativos de como é feita esta implementação no módulo nlu. No primeiro exemplo da figura damos exemplos de textos que simbolizam a intenção de como o usuário pode iniciar um diálogo. Já no segundo exemplo temos que os textos que representam a intenção do usuário de enviar o seu nome, além disso também é identificado a entidade que representa o nome do usuário.

Figura 33 – Exemplo ilustrativo de implementação no módulo nlu

```

- intent: user-initiative
examples: |
  - Olá
  - oi
  - bom dia
  - boa tarde
...

- intent: send-info-name
examples: |
  - [Bruno da Silva Pinho>{"entity" : "have-patient-name"}
  - Meu nome é [Bruno da Silva Pinho>{"entity" : "have-patient-name"}
  - Eu sou [Bruno da Silva Pinho>{"entity" : "have-patient-name"}
...

```

Fonte: Elaborado pelo autor

3. **Módulo stories** - Neste módulo é onde colocamos todos os ramos da conversa, onde para cada intenção identificada pelo Rasa associamos a um ação. Porém em nossa implementação possuímos apenas uma ação global que encapsula todas as ações que agente pode fazer e cabe ao GD escolher qual ação deve ser executada.

Figura 34 – Exemplo de implementação no módulo stories

```

stories:
- story: main path
  steps:
  - intent: user-initiative
  - action: global_action

  - intent: can-start-online-service
  - action: global_action

  - intent: can-show-info-others
  - action: global_action

  - intent: can-show-info-mental-health
  - action: global_action

...

```

Fonte: Elaborado pelo autor

4. **Módulo actions** - Neste módulo há uma única ação global que contém todos os textos

que podem ser enviados para o usuário. Esta ação é responsável por receber as entidades e intenções provindas das declarações do usuário e enviá-las para o GD. A ação global também é responsável por escolher qual texto será enviado para o usuário a partir da ação provinda do GD. Para isso, temos um mapeamento de um para um de ação contidas no domínio PDDL para textos que devem ser enviado para o usuário. Estes mapeamento é utilizado para saber quais textos devem ser enviados para o usuário quando é recebido uma ação do GD.

A ação global no Rasa é implementado utilizando a linguagem de programação *python* onde ela é uma classe que possui as seguintes funções:

- a) Função name: Esta função é responsável por retornar o nome da ação global, como é mostrado na Figura 35.

Figura 35 – Implementação da função name.

```
def name(self) -> Text:
    return 'global_action'
```

Fonte: Elaborado pelo autor

- b) Função action_text: Esta função recebe como entrada uma ação e retorna o texto que deve ser mandado para o chat do usuário. A Figura 36 ilustra a função action_text.

Figura 36 – Implementação da função action_text

```
def action_text(self, action):
    self.text = ""

    if action == "error-treatment":
        return "Não conseguir entender o que você quis dizer"

    elif action == "show-welcome-message":
        self.text = '''
        Olá! Bem-vindo(a) ao serviço de atendimento online da Secretaria da Saúde do Estado do Ceará!
        '''
        return self.text

    elif action == "start-dialog":
        self.text = '''
        Do que você precisa agora?
        - 🧑‍⚕️ Atendimento online sobre o novo Coronavírus (COVID-19)
        - ? Informações sobre o novo Coronavírus (COVID-19)
        - ❤️ Cuidados com a Saúde Mental
        - Outras informações
        '''
        return self.text
```

Fonte: Elaborado pelo autor

- c) Função run: Esta é a função que tem como objetivo receber as entidades e intenções

da entrada do usuário e enviá-las para o GD, bem como receber as ações retornadas pelo GD e transformá-las em texto e enviá-los para o chat do usuário. A Figura 37 mostra a implementação desta função, na qual são capturados os nomes das intenções e as entidades da entrada do usuário. No Rasa existe um intenção padrão para capturar entradas do usuário que estão fora do contexto, essa intenção possui o nome de **nlu_fallback**, quando se é capturado essa intenção, o nome dela é alterado para **can-go-error-treatment** para ficar de acordo com a modelagem proposta. Em seguida, é verificado se o estado do mundo está vazio. Caso esteja a entrada do usuário será considerado como **user-initiative**, e então é chamado a função **wait_user_utter** do GD para atualizar os valores do estado do mundo. Em seguida, é chamada a função **action_execution** que retorna as ações que serão transformadas em texto. E, por fim, é feita a transformação das ações em textos e é mandado para o usuário.

Figura 37 – Implementação da função run.

```
def run(self, dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
    intent = tracker.latest_message['intent']
    ents = tracker.latest_message['entities']
    entities = []

    for entity in ents:
        if not(entity['entity'] in entities):
            entities.append(entity['entity'])

    intent_name = intent.get('name')

    if intent_name == "nlu_fallback":
        intent_name = 'can-go-error-treatment'

    if GD.state == []:
        intent_name = 'user-initiative'
    estado, outcome = GD.wait_user_utter(self.actions[-1], intent=intent_name, entities=entities)

    self.actions = GD.action_execution()
    text = ""
    for action in self.actions:
        text += self.action_text(action)

    dispatcher.utter_message(text)
```

Fonte: Elaborado pelo autor

5 AVALIAÇÃO DO GERENCIADOR DE DIÁLOGO

Neste capítulo apresentamos os resultados obtidos por meio da avaliação do gerenciador de diálogo baseado em planejamento automatizado, o qual foi integrado ao sistema de conversação do *chatbot* Plantão Coronavírus.

Segundo Teixeira (2022), não há consenso sobre uma estratégia de avaliação que seja mais adequada para avaliar sistemas de diálogo construídos para as mais diversas finalidades, tornando essa tarefa bastante desafiadora. Uma estratégia que tem sido adotada na literatura é a avaliação do sistema por usuários humanos. No entanto, não é esperado que esses usuários finais entendam as características específicas de cada módulo do sistema, seja ele o módulo de entendimento da linguagem natural ou o *módulo gerenciador de diálogo*. Teixeira (2022) conclui ainda que não existe referências ou protocolos oficiais disponíveis que possam guiar uma avaliação que não requeira usuários humanos.

Devido às dificuldades elencadas, iremos realizar as seguintes avaliações no sistema proposto:

- Viabilidade de construção automática da política pelo planejador *versus* construção manual da árvore de diálogo;
- Qualidade das políticas geradas, verificando se seriam capazes de gerenciar diálogos reais.

Os experimentos foram conduzidos utilizando-se um notebook com processador Intel® Core™ i7-7500U velocidade 2.70GHz., um disco de um terabyte e 8 gigabytes de memória RAM.

5.1 Construção automática da árvore de diálogo

O uso de um gerenciador de diálogos baseado em planejamento automatizado justifica-se quando não é possível usar um gerenciador baseado em aprendizado de máquina (devido a questões de confiabilidade e capacidade de reportar explicações, por exemplo) e também quando não é desejável ou viável construir a árvore de diálogos manualmente (Muisse *et al.*, 2019).

A modelagem do domínio de planejamento para gerenciamento do *chatbot* Plantão Coronavírus resultou em 89 predicados e 60 ações¹. O problema de planejamento consiste no estado inicial em que apenas o predicado *USER-INITIATIVE* é verdadeiro e a meta é alcançar

¹ Veja o domínio completo em <https://github.com/brunopinheiro321/Domain-Chatbot-Covid/blob/main/Domain.pddl>. Acesso em 6 de outubro de 2023

um estado no qual o predicado *GOAL* é verdadeiro. Utilizamos o planejador PRP (Muisse *et al.*, 2012) para obtenção da solução do problema e obtivemos uma política com 168 estados e 334 transições, a qual consiste em todos os possíveis caminhos de diálogo para o diálogo sair do estado inicial e alcançar o estado meta. A política gerada pode ser vista no grafo da Figura 38 no Apêndice A e corresponde à árvore de diálogo que deveria ser feita manualmente. Neste grafo temos que o estado inicial está representado pela cor azul e o estado objetivo está representado pela cor verde (estados ampliados respectivamente nas Figuras 39 e 40). O tempo de geração da política pelo planejador foi de 0.065 segundos. Um diálogo é um possível caminho da política, conforme a interação com o usuário.

5.2 Gerenciamento do Diálogo Utilizando a Política Obtida

Para a avaliação da qualidade do gerenciamento proporcionado pela política gerada, foram feitos três cenários de teste distintos, representando conversas entre o usuário e o *bot*, com base nos diálogos reais. Para isso utilizamos o conceito de *persona*, onde uma *persona* é um personagem fictício, modelo hipotético de um grupo de usuários reais, criada para descrever um usuário típico (Barbosa; Silva, 2010). Em cada cenário temos uma *persona*. Essa avaliação permitirá verificar se o gerenciamento está ocorrendo de forma adequada.

5.2.1 Cenários de Teste

A seguir descrevemos detalhadamente os cenários de teste:

- **Cenário 1:** Neste cenário é seguido o fluxo mais comum que se encontra nos diálogos reais nos quais temos que um usuário deseja fazer uma consulta online e deste modo ele precisa passar suas informações para o *chatbot* de forma que o mesmo lhe direcione a um agente humano e, por fim, conclua o diálogo.

Neste cenário, observamos um usuário homem de 26 anos que deseja realizar uma consulta online. Ao interagir com o *chatbot*, o usuário cumprimenta o *bot*. Neste momento, o *chatbot* dá boas vindas e apresenta um menu contendo quatro opções. O usuário opta pela a opção de atendimento online. A partir daí, o *chatbot* solicita algumas informações. O usuário envia as informações a medida que vão sendo solicitadas tais como os sintomas que está sentindo e há quantos dias esses sintomas estão ocorrendo. O usuário opta por não compartilhar sua localização, fazendo com que seja solicitado o CEP. Após informar o

CEP, o usuário fornece seu CPF, nome e data de nascimento.

Prosseguindo, o usuário declara que tem comorbidades (problemas respiratórios), informa seu gênero e fornece seu número de telefone. A partir desse ponto, ele é transferido para um agente de saúde humano para dar continuidade ao seu atendimento.

Ao término do seu atendimento com o agente humano, o usuário é transferido para o *chatbot* onde é solicitado que o mesmo avalie o serviço, atribuindo uma nota de 1 a 10. Uma vez informada a nota, o atendimento é finalizado.

- **Cenário 2:** Esse cenário é bastante semelhante ao anterior, porém temos que o usuário pode cometer erros e fornecer entradas que o *chatbot* não consiga reconhecer. A política deve ser capaz de gerenciar tais entradas que não são reconhecidas.

Neste cenário, observamos uma mulher de 36 anos que deseja realizar uma consulta online. Ao interagir com o chatbot, a usuária digita sua intenção de obter informações sobre a doença. Neste momento o *chatbot* dá boas vindas e apresenta um menu contendo quatro opções. Ela opta pela opção de atendimento online. A partir daí, o *chatbot* solicita algumas informações à usuária. Ela fornece as informações a medida que o chatbot solicita, informando os sintomas que está sentindo e há quantos dias esses sintomas se manifestaram. Além disso, ela compartilha sua localização e fornece seu CPF e nome. Ao tentar inserir sua data de nascimento, a usuária comete um equívoco, digitando esta data de forma incorreta. No entanto, ao receber a notificação do erro fornecida pelo *chatbot*, ela corrige a informação.

Prosseguindo, a usuária declara que não possui nenhuma comorbidade, informa seu gênero e esclarece que não está grávida. Ao tentar fornecer seu número de celular, ocorre outro erro, mas, novamente, após ser alertada, a usuária insere o número corretamente. A partir desse ponto, ela é transferida para um agente de saúde humano para dar continuidade ao atendimento.

Ao término do processo, após a conclusão do atendimento pelo agente de saúde humano, a usuária é transferida de volta para o *chatbot*, onde é solicitado que avalie o serviço, atribuindo uma nota de 1 a 10. Uma vez informada a nota, o atendimento é finalizado.

- **Cenário 3:** Este cenário a conversa segue por um fluxo que não é comum nos diálogos reais, porém a política deve ser capaz de gerenciar estes caminhos menos comuns.

Neste cenário, observamos um usuário em busca de informações sobre o novo coronavírus. Ao iniciar a interação com o *chatbot*, o usuário inicia com uma saudação. O *chatbot*

responde com boas vindas e apresenta um menu com quatro opções.

O usuário então seleciona a opção relacionada às informações sobre o novo coronavírus. Em seguida, apresenta nove sub-opções, das quais o usuário escolhe a quarta, digitando “*O que fazer quando sentir os sintomas?*”. Nesse momento, o *chatbot* exibe outras nove opções relacionadas com a pergunta do usuário.

O usuário decide selecionar a oitava opção, digitando “*Todo caso de covid precisa ser internado no hospital?*”. O *chatbot* então fornece informações relevantes sobre a questão. Em seguida, o usuário opta por retornar ao menu inicial, escolhe novamente a opção de informações sobre o novo coronavírus e, em seguida seleciona a terceira opção, digitando “*Quais são os principais sintomas?*”.

O *chatbot* prontamente exibe as informações que o usuário deseja obter sobre os principais sintomas da COVID-19. Satisfeito com as respostas fornecidas pelo *bot*, o usuário opta por encerrar o diálogo. Neste ponto, é solicitado que o usuário atribua uma nota de 1 a 10 ao serviço prestado. Após a avaliação ser informada, o atendimento é finalizado.

5.2.2 *Resultados das simulações*

Aqui apresentaremos os resultados dos cenários que foram simulados, mostrando o *caminho* percorrido no grafo representando a política, e o *tamanho da solução* (i.e, número de ações no caminho).

Na Tabela 6 podemos observar, para cada cenário, o tamanho da solução que consiste no número de ações que levam do estado inicial até o estado meta, e a sequência de ações percorrida. No primeiro cenário, foi obtido uma solução de tamanho 20, já no segundo a solução teve tamanho 26, e na terceira, a solução resultante é de tamanho 10. Os resultados das simulações podem ser vistos nas Figuras 41, 42, e 43 no Apêndice A. Nestas figuras, o estado inicial é representado pela cor azul e o estado de aceitação é representado pela a cor verde. Já o caminho do estado inicial até o estado de aceitação está ilustrado com a cor vermelha.

Apesar dos resultados da simulações 1 e 2, terem muitas ações em comum que representam os fluxos mais comuns que ocorrem nos diálogos reais, temos que os caminhos são distintos. Na simulação 2 foi simulado um erro que o usuário cometeu ao se comunicar com o *bot*, neste caso é possível observar que toda vez que o chat não entende o que o usuário digitou, capturado por um dos efeitos não-determinísticos da ação, o *bot* consegue tomar uma ação para este erro. Já no resultado da simulação 3 o resultado obtido conseguiu chegar ao estado objetivo,

como ilustra as figuras citadas acima.

Desta forma, temos que a política gerada pelo planejador, utilizando o domínio modelado, foi capaz de gerar uma árvore de forma automática, que abrange todos os possíveis fluxos de diálogos. Além disso, o algoritmo de gerenciamento do diálogo, que utiliza a política presente neste trabalho, mostrou-se eficaz ao conseguir escolher as ações adequadas para cada cenário das simulações.

Tabela 6 – Resultado das simulações dos cenários de teste usando a política de gerenciamento obtida.

Cenário	Tamanho da solução	Caminho
1	20	show-welcome-message, start-dialog, start-online-service, ask-patient-symptoms, ask-patient-how-many-days-symptoms, call-diagnostic-system, show-info-diagnostic-positive, ask-share-location, ask-postal-code, show-info-calling-health-professional, ask-patient-info-cpf, ask-patient-info-name, ask-patient-info-birthday, ask-patient-comorbidities, ask-patient-info-gender, ask-patient-info-phone-number, call-health-agent, health-agent-takes-control, ask-user-grade, finish-service
2	26	show-welcome-message, start-dialog, start-online-service, ask-patient-symptoms, ask-patient-how-many-days-symptoms, call-diagnostic-system, show-info-diagnostic-positive, ask-share-location, show-info-calling-health-professional, ask-patient-info-cpf, ask-patient-info-name, ask-patient-info-birthday, error-treatment, ask-patient-info-birthday, ask-patient-comorbidities, error-treatment, ask-patient-comorbidities, ask-patient-info-gender, ask-patient-info-pregnant-woman, ask-patient-info-phone-number, error-treatment, ask-patient-info-phone-number, call-health-agent, health-agent-takes-control, ask-user-grade, finish-service
3	10	show-welcome-message, start-dialog, show-info-about-covid, show-info-about-when-feel-the-symptoms, show-info-about-the-cases-need-hospitalization, start-dialog, show-info-about-covid, show-info-about-main-symptoms, ask-user-grade, finish-service

Fonte: Elaborado pelo autor

6 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação propomos uma abordagem para o agente de diálogo Plantão Coronavírus, um sistema de conversação do Governo do Estado do Ceará que foi utilizado para atendimento de pacientes nos momentos críticos da pandemia de COVID-19. O sistema proposto utiliza: (i) Aprendizado de Máquina para as tarefas relacionadas ao reconhecimento da linguagem natural; (ii) Planejamento Automatizado para as tarefas relacionadas ao gerenciamento do diálogo.

De posse dos dados dos diálogos que ocorreram utilizando o sistema de conversação, realizamos a modelagem do domínio de planejamento não-determinístico em PDDL, inspirada na modelagem proposta por Muise *et al.* (2019). Os predicados PDDL foram obtidos por meio das intenções e entidades do usuário contidas em textos de diálogos entre paciente e *bot*. Já as ações não-determinísticas PDDL foram provenientes dos textos de diálogos referentes às ações do *chatbot*. Tais ações foram modeladas com efeitos não-determinísticos para simular as diversas possibilidades de comportamento do usuário nas interações com o *bot*. O gerenciador foi integrado ao RASA, que operou como o Serviço de Conversação (SC) do *chatbot*, sendo orientado pelas ações da política obtida automaticamente pelo planejador não-determinístico PRP (*Planning for Relevance Policies*). O sistema foi avaliado realizando-se simulações para verificar se a política obtida pelo planejador seria capaz de conduzir diálogos similares aos reais. Em suma, as principais contribuições deste trabalho são:

- modelagem do domínio de planejamento em PDDL, obtido a partir da análise dos dados de conversa;
- proposta de arquitetura do sistema que integra o módulo de reconhecimento da linguagem natural com o módulo de gerenciamento baseado em planejamento automatizado;
- implementação da arquitetura utilizando o Rasa como sistema de conversação e o planejador PRP como gerador automático dos caminhos de diálogos e;
- avaliação do sistema proposto utilizando simulações para verificar se os caminhos da política são capazes de conduzir as conversas que ocorreram entre paciente e *bot*.

A abordagem de Planejamento Não-Determinístico mostrou-se viável para o gerenciamento do agente de diálogo Plantão Coronavírus, sendo uma alternativa mais apropriada para gerenciamento de agentes de diálogos em sistemas críticos, como sistemas de saúde, do que a abordagem de Aprendizado de Máquina, que não é capaz de garantir a previsibilidade das respostas dadas e tampouco de explicar como elas foram obtidas. Para ter respostas mais previsí-

veis e passíveis de serem explicáveis para sistemas desta natureza costuma-se utilizar árvores de diálogos para a tarefa de gerenciamento do diálogo. No entanto, a construção manual destas árvores é uma tarefa muito custosa para diálogos extensos. Assim, mostramos neste trabalho que Planejamento Automatizado é uma abordagem adequada para **gerar automaticamente** a árvore de diálogo e, por se tratar de uma técnica de raciocínio simbólico, tem a capacidade também de explicar as respostas fornecidas ao usuário.

Vale ressaltar que a aquisição do domínio de planejamento, a partir dos dados de conversa, foi realizada de forma manual. Neste sentido, possibilidades de trabalhos futuros para essa pesquisa devem explorar a aquisição automática do domínio de gerenciamento a partir dos dados de diálogo. Outra possibilidade de trabalhos futuros é incorporar na arquitetura proposta um módulo para fornecer explicações para as respostas do usuário. Além disso, é importante ainda explorar a avaliação do sistema proposto, por exemplo, realizando simulações do desempenho da política em relação a todos os diálogos reais disponíveis, que ocorreram no período de utilização do sistema.

REFERÊNCIAS

- AMAZON. **Amazon Lex**: Ia conversacional para chatbots. 2022. Disponível em: <https://aws.amazon.com/pt/lex/>. Acesso em: 24 maio. 2022.
- AMAZON. **Descubra tudo o que você pode fazer com a Alexa**. 2022. Disponível em: <https://www.amazon.com.br/b?ie=UTF8&node=19949683011>. Acesso em: 03 jun. 2022.
- APPLE. **A Siri faz mais. Mesmo antes de você pedir**. 2022. Disponível em: <https://www.apple.com/br/siri/>. Acesso em: 03 jun. 2022.
- BARBOSA, S.; SILVA, B. **Interação humano-computador**. [S. l.]: Elsevier Brasil, 2010.
- BOTEA, A.; MUISE, C.; AGARWAL, S.; ALKAN, O.; BAJGAR, O.; DALY, E.; KISHIMOTO, A.; LASTRAS, L.; MARINESCU, R.; ONDREJ, J. *et al.* **Generating dialogue agents via automated planning**. 2019.
- CARDOSO, A. **Plantão Coronavírus**: 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: 08 abr. 2023.
- CIMATTI, A.; PISTORE, M.; ROVERI, M.; TRAVERSO, P. **Weak, strong, and strong cyclic planning via symbolic model checking**. [S. l.]: Elsevier, 2003. v. 147. 35–84 p.
- DEEP-DIAL. **Reasoning and Learning for Human-Machine Dialogue**. 2018. Disponível em: <https://aaai.org/Conferences/AAAI-18/ws18workshops/#ws14>. Acesso em: 24 nov. 2021.
- ERIKSSON, H. **A survey of knowledge acquisition techniques and tools and their relationship to software engineering**. [S. l.]: Elsevier, 1992. v. 19. 97–107 p.
- GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: theory and practice**. [S. l.]: Elsevier, 2004.
- GOOGLE. **Dialogflow**: Crie experiências de conversa entre dispositivos e plataformas. 2022. Disponível em: <https://cloud.google.com/dialogflow/>. Acesso em: 24 maio. 2022.
- GOOGLE. **Google Home**: Smart home automation from google. 2022. Disponível em: <https://home.google.com/welcome/>. Acesso em: 03 jun. 2022.
- GRICE, H. P. **Logic and conversation**. [S. l.]: Brill, 1975. 41–58 p.
- HELMERT, M. **Concise finite-domain representations for PDDL planning tasks**. [S. l.]: Elsevier, 2009. v. 173. 503–535 p.
- IBM. **Watson Assistant**: The ai assistant that solves customer problems the first time. 2022. Disponível em: <https://www.ibm.com/cloud/watson-assistant>. Acesso em: 24 maio. 2022.
- JILANI, R.; CRAMPTON, A.; KITCHIN, D. E.; VALLATI, M. **Automated knowledge engineering tools in planning: state-of-the-art and future challenges**. [S. l.: s. n.], 2014.
- KHAN, R.; DAS, A. **Build better chatbots**. [S. l.]: Springer, 2018.
- LACERDA, A. R. T. d. **Rasa-ptbr-boilerplate**: Floss project that enables brazilian portuguese chatbot development by non-experts. 2019.

- LJUNGLÖF, P. **Dialogue management as interactive tree building**. [S. l.], 2009.
- MAMATHA, M. *et al.* Chatbot for e-commerce assistance: based on rasa. **Turkish Journal of Computer and Mathematics Education (TURCOMAT)**, v. 12, n. 11, p. 6173–6179, 2021.
- MCDERMOTT, D. M. **The 1998 AI planning systems competition**. 2000. 35–35 p.
- MCTEAR, M. **The articulate computer**. [S. l.]: Blackwell Publishers, Inc., 1987.
- MCTEAR, M. **Conversational AI: dialogue systems, conversational agents, and chatbots**. [S. l.]: Morgan & Claypool Publishers, 2020. v. 13. 1–251 p.
- MENEZES, M. V. de; BARROS, L. N. de; PEREIRA, S. do L. **Symbolic regression for non-deterministic actions**. 2014.
- MICROSOFT. **Microsoft Bot Framework: A comprehensive framework for building enterprise-grade conversational ai experience**. 2022. Disponível em: <https://dev.botframework.com/>. Acesso em: 24 maio. 2022.
- MIGLANI, S.; YORKE-SMITH, N. **Nltopddl One-shot learning of pddl models from natural language process manuals**. [S. l.]: ICAPS, 2020.
- MONT'ALVERNE, A. S. S. **"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: 08 abr. 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**. 2019.
- MUISE, C.; MCILRAITH, S.; BECK, C. Improved non-deterministic planning by exploiting state relevance. **Proceedings of the International Conference on Automated Planning and Scheduling**, 2012.
- MUISE, C. J.; BECK, J. C.; MCILRAITH, S. A. **Flexible Execution of Partial Order Plans With Temporal Constraints**. [S. l.: s. n.], 2013. 2328–2335 p.
- PALLAGANI, V.; SRIVASTAVA, B. **A Generic Dialog Agent for Information Retrieval Based on Automated Planning Within a Reinforcement Learning Platform**. 2021.
- PEREIRA, S. d. L. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. Tese (Doutorado) – Universidade de São Paulo, 2007.
- RASA. **Rasa: Open source conversational ai**. 2022. Disponível em: <https://rasa.com/>. Acesso em: 03 jun. 2022.
- RICH, C.; SIDNER, C. L. **Using collaborative discourse theory to partially automate dialogue tree authoring**. [S. l.], 2012. 327–340 p.
- ROMERO, O. J.; ZHAO, R.; CASSELL, J. **Cognitive-Inspired Conversational-Strategy Reasoner for Socially-Aware Agents**. [S. l.]: IJCAI, 2017. 3807–3813 p.
- SANTOS, V. B. d.; BARROS, L. N. d.; PEREIRA, S. d. L.; MENEZES, M. V. d. Symbolic fond planning for temporally extended goals. **Workshop**, 2022.

SANTOS, V. B. dos. **Planejamento baseado em Verificação Simbólica de Modelos**. 2018.

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

TEIXEIRA, M. S. **Automating the Generation of Goal-Oriented Dialogue Managers for Healthcare**. Tese (Doutorado), 2022.

VICERI. **RASA**: Conheça o software em python utilizado para criação de chatbots. 2020. Disponível em: <https://viceri.com.br/insights/rasa-conheca-o-software-em-python-utilizado-para-criacao-de-chatbots/>. Acesso em: 29 jun. 2022.

WEIZENBAUM, J. **ELIZA—a computer program for the study of natural language communication between man and machine**. [S. l.]: ACM New York, NY, USA, 1966. v. 9. 36–45 p.

APÊNDICE A – GRAFO DA POLITICA GERADA PELO PLANEJADOR PRP

Figura 38 – Grafo da Política

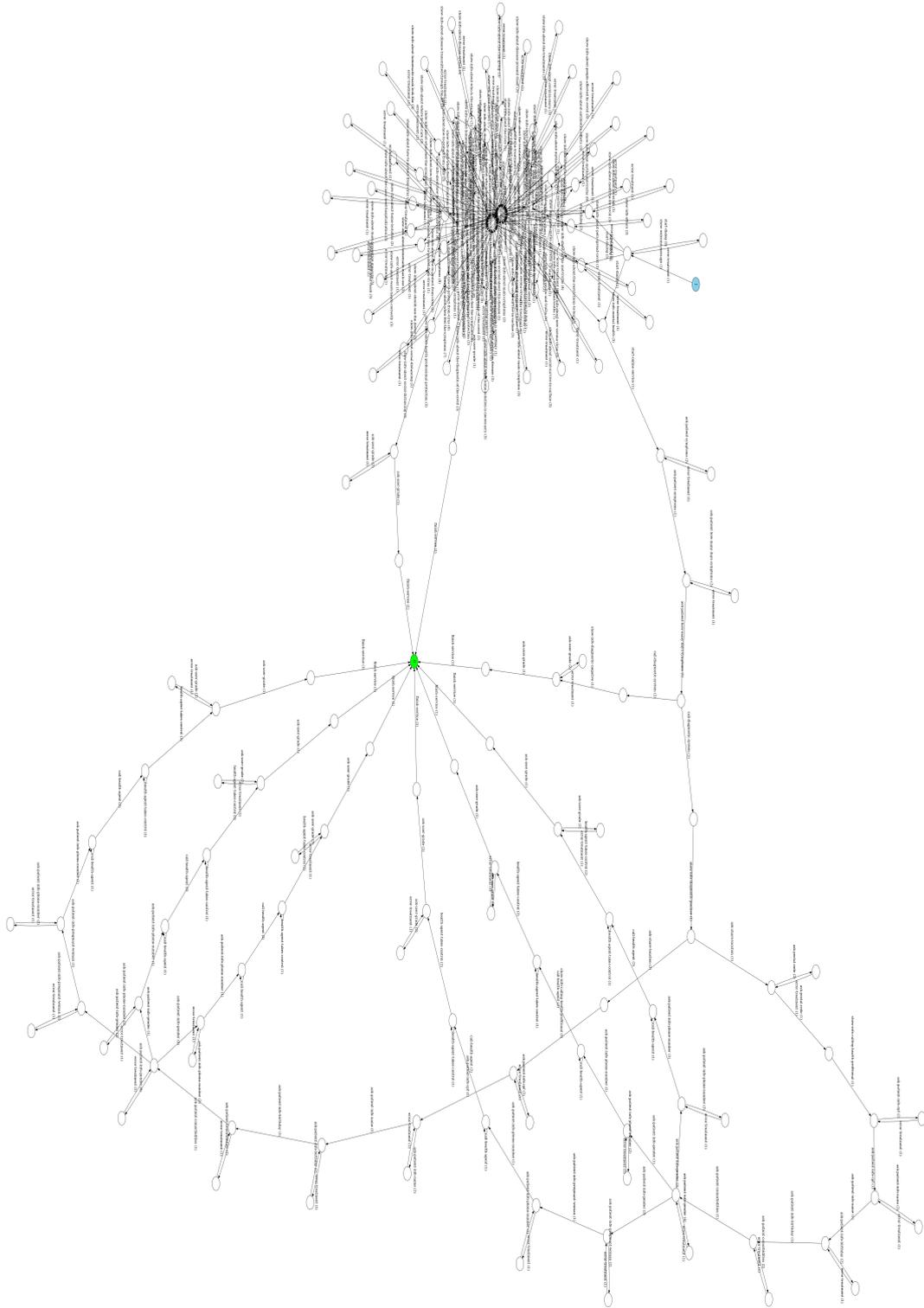


Figura 42 – Simulação 2

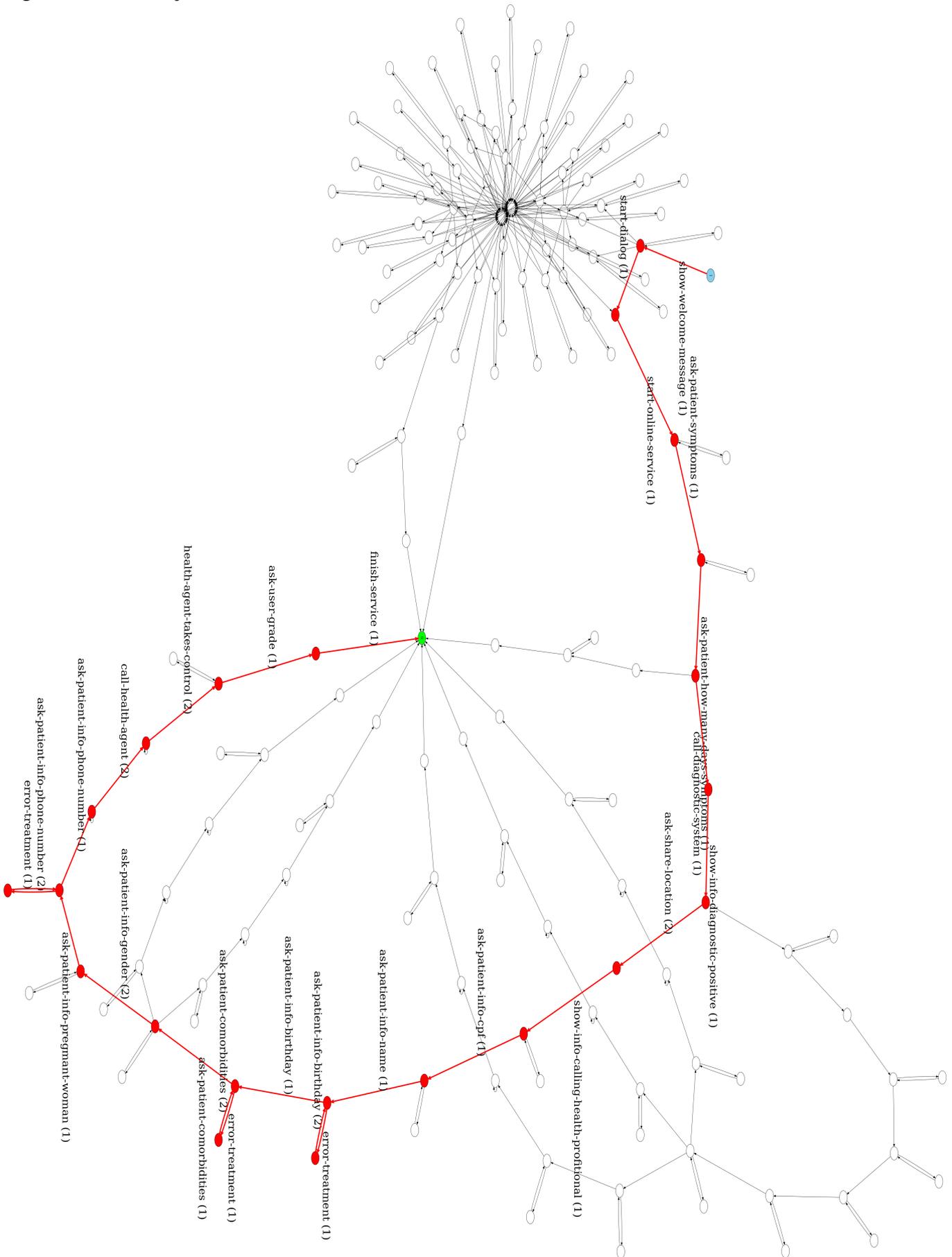


Figura 43 – Simulação 3

