Uma Proposta de Tradução de Autômatos Temporais para Agentes BDI

Lucas S. Karau[1], **Rafael C. Cardoso**[2], **André P. Borges**[1], **Gleifer V. Alves**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Ponta Grossa, PR, Brasil

[2]University of Aberdeen
Aberdeen, United Kingdom

lucaskarau@alunos.utfpr.edu.br, rafael.cardoso@abdn.ac.uk

{apborges,gleifer}@utfpr.edu.br

Abstract. *Temporal automata are a suitable structure for verifying reactive behaviours, while the BDI agent structure covers a more cognitive and rational aspect of decision-making. This article aims to move from a purely reactive system to a decision-making system based on beliefs and desires, allowing for a more rational representation of an agent selecting a plan to follow. Based on this, the following paper has as its main goal to present a proposal for translating a temporal automata, represented using the real-time modelling tool UPPAAL, into a BDI agent, using the structure of the MASPY framework as reference.*

Resumo. *Autômatos temporais são uma estrutura adequada para verificação de comportamentos reativos, enquanto a estrutura de agentes BDI abrange uma parte mais cognitiva e racional de tomada de decisão. Este artigo tem o objetivo de partir de um sistema puramente reativo para um sistema de tomada de decisões baseada em crenças e objetivos, permitindo representar um agente de forma mais racional selecionando um plano a ser seguido. Com isso, o presente artigo tem como objetivo apresentar uma proposta de tradução de um autômato temporal, representado utilizando a ferramenta de modelagem em tempo real UPPAAL, para um agente BDI, utilizando a estrutura do framework MASPY como referência.*

1. Introdução

Enquanto autômatos temporais são eficazes para tratar restrições temporais [Alur 1999], a arquitetura BDI, baseada em crenças, desejos e intenções, oferece uma modelagem mais adequada para representar decisões complexas e racionais [Bratman 1987]. O autômato temporal, juntamente com seus detalhes, é exemplificado utilizando-se a ferramenta de modelagem de sistemas em tempo real UPPAAL [Larsen et al. 1995], na qual foi selecionado para exemplo o modelo *2doors*, que possui os autômatos *Door* e *User*, devido a sua simplicidade (porém ainda abrangendo grande parte dos elementos de um autômato temporal), suas conexões via canais de sincronização e devido a sua fácil visualização como uma situação real, representando uma simples simulação de abertura e fechamento de uma porta.

O seguinte artigo propõe traduzir um autômato temporal para um agente BDI, pretendendo unir a verificação de situações críticas baseadas no tempo com a tomada de decisão mais racional e próxima do pensamento humano. Isso é, partir de um sistema mais estático e que não trata imprevisibilidades para um sistema BDI que possibilita maior abstração de comportamentos e é mais eficiente em tomadas de decisão em situações críticas. A proposta de tradução de um autômato temporal para agente BDI será realizada tendo como base a estrutura de código no framework MASPY [Mellado et al. 2023]. Tal framework permite o desenvolvimento de sistemas multiagentes BDI, sendo escolhido por sua eficiente e simplificada implementação em Python, além de permitir definir explicitamente o(s) agente(s) juntamente com suas crenças, desejos e planos. No escopo deste artigo, a proposta aqui descrita representa um "esboço" da tradução de autômatos para planos de um agente BDI, mas não é traduzido efetivamente para um código funcional Python. A efetiva tradução de arquivos XML (arquivos em UPPAAL) para arquivos Python (MASPY) será realizada em trabalhos futuros.

2. Definições

Em [Alur 1999] é apresentada uma descrição detalhada de como um autômato temporal utiliza do tempo e dos estados para representar o comportamento de um determinado sistema. Para isso, são utilizadas as restrições de tempo, e para exemplificá-las, utilizam-se os relógios. Um autômato temporal possui relógios para controlar o tempo decorrido em um estado do autômato à taxa de uma unidade [Alves et al. 2021]. Além disso, relógios podem possuir restrições, utilizadas quando um determinado relógio tem seu valor comparado a um valor natural, delimitando assim um limite temporal.

A ideia inicial de agentes BDI surgiu com o trabalho do filósofo Bratman, descrito em [de Silva et al. 2022] como uma representação do pensamento racional prático humano. Na arquitetura BDI, um agente possui crenças/informações, desejos/objetivos e intenções. As intenções são os planos que o agente se compromete a realizar para concluir seus objetivos baseados nas crenças que possui.

Para representar um agente BDI, o framework em Python MASPY (Multi-Agent System for PYthon) [Mellado et al. 2023] foi selecionado, o qual objetiva facilitar o desenvolvimento de sistemas multiagentes baseados no paradigma BDI. Neste momento, o necessário para propor a tradução é a estrutura de agentes e seus planos. Um plano em MASPY contém um "evento gatilho", um contexto para ativação do plano e o seu corpo. O contexto pode possuir qualquer quantidade de crenças ou objetivos os quais o agente deve ter para executar determinado plano [Mellado et al. 2023].

Para a definição de planos em MASPY é utilizado um "decorador" *@pl*, que deve possuir uma mudança para o plano ser ativado (*gain*, *lose* ou *test*), as informações que mudaram (o "gatilho" do plano, podendo conter *Belief(s)* ou *Goal(s)*), e opcionalmente um contexto para executar o plano de fato (também *Belief(s)* ou *Goal(s)*).

3. Modelagem

Para exemplificar um autômato temporal, será utilizado o modelo *2doors* da ferramenta UPPAAL. A Figura 1 representa os dois autômatos (*Door* e *User*, respectivamente), os quais se conectam por meio de canais de sincronização (*pushed* e *closed*) com a finalidade de trocarem mensagens e informações, baseando-se nelas para tomar determinadas ações.

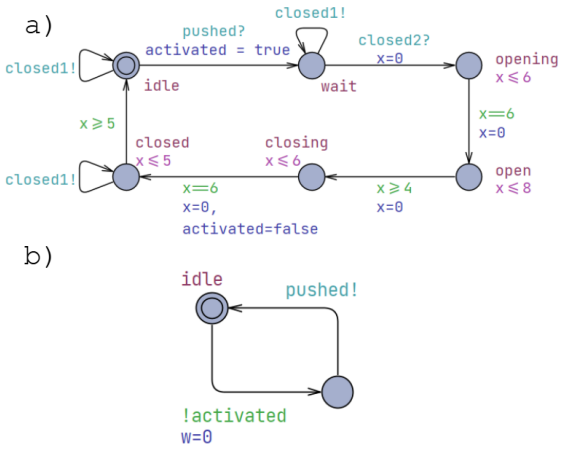

Figura 1. Autômatos Door e User

O autômato da Figura 1.a) representa *Door*, o qual inicia no estado `idle`. No estado `wait` é esperada a confirmação de que a outra porta (segunda instância de *Door*) está fechada (`closed2?`), assim o relógio x recebe 0 e o agente passa ao estado `opening`.

As restrições de relógio $x <= 6$ e $x == 6$ em relação à x significam que de `opening` para `open` x deverá ter valor 6. Agora, na transição de `open` para `closing` o relógio x deverá estar associado a um valor entre 4 e 8 unidades, devido às restrições $x <= 8$ e $x >= 4$. Após trocar para o estado `closing`, x recebe 0 novamente.

Para sair de `closing` para `closed`, x deverá ter valor 6. Quando é feita a transição de estados, x recebe 0 e a variável `activated` recebe valor `false`, descrevendo que o agente parou de realizar a "abertura e fechamento" da porta. Ao fim, no estado `closed`, a mensagem `closed1!` é enviada informando que a primeira instância de *Door* está fechada e é possibilitada a abertura da segunda instância de *Door*.

Agora, o autômato da Figura 1.b) representa *User*. Ele inicia no estado `idle`, e o operador *User* é ativado quando o agente que coordena o autômato *Door* correspondente estiver desativado (`!activated`), assim o relógio w é resetado para iniciar a contagem de tempo de abertura e fechamento da porta e é enviada uma mensagem ao autômato *Door* informando que *User* iniciou a abertura da porta (`pushed!`). Após o envio da mensagem, a variável `activated` recebe `true` e *User* retorna ao estado de repouso novamente.

4. Proposta de Tradução

Na seção 3 é demonstrada uma breve explicação de como o autômato temporal em UPPAAL realiza suas ações. É possível observar que um autômato temporal é um modelo que exige certas restrições e exceções para uma determinada transição de estado ocorrer, não abordando tomadas de decisão que deveriam ocorrer em situações críticas. Por outro lado, uma estrutura BDI possibilita essa abordagem.

Nesse artigo, a realização de uma proposta de tradução de um autômato temporal para uma especificação em planos BDI tem como motivação e objetivo gerar um código de agente MASPY a partir de um modelo já verificado formalmente através da ferramenta *model checker* UPPAAL.

Tomando como base a estrutura de planos de agentes BDI no framework MASPY e parte de cada autômato da Figura 1.a) e 1.b), é realizado uma proposta de tradução partindo dos autômatos em UPPAAL para os Códigos 1 e 2 em MASPY.

```
1  class Door(Agent):
2      def __init__(self, agt_name):
3          super().__init__(agt_name)
4          self.add(Belief("x"))
5          self.add(Belief("activated"), false)
6          self.send("User", tell, ("activated", false))
7          self.add(Goal("idle"))
8          self.send("Door2", achieve, Goal("closed1"))
9
10     @pl(gain, Goal("idle"), Goal("pushed"))
11     def idle(self, src):
12         self.send("Door2", achieve, Goal("closed1"))
13         self.add(Belief("activated", true))
14         self.add(Goal("wait"))
15
16     @pl(gain, Goal("wait"), Goal("closed2"))
17     def wait(self, src):
18         self.add(Belief("x", 0))
19         self.add(Goal("opening"))
20
21     @pl(gain, Goal("opening"), Belief("x", x == 6))
22     def opening(self, src):
23         self.add(Belief("x", 0))
24         self.add(Goal("open"))
```

Código 1. Agente "Door" em MASPY

```
1  class User(Agent):
2      def __init__(self, agt_name):
3          super().__init__(agt_name)
4          self.add(Belief("w"))
5          self.add(Goal("idle"))
6
7      @pl(gain, Goal("idle"), Belief("activated", false))
8      def idle(self, src):
9          self.add(Belief("w", 0))
10         self.send("Door", achieve, Goal("pushed"))
```

Código 2. Agente "User" em MASPY

Primeiramente, definem-se as classes *Door* e *User* como `Agent`, transformando os dois autômatos em agentes. Para a estrutura de tradução para crenças iniciais, objetivos iniciais e planos do agente foi utilizada como base a Figura 2 de [Markovicz et al. 2024]. O autômato possui um Estado Inicial (em *Door* é `idle`), o qual é traduzido como objetivo inicial no agente na linha 7 do Código 1, as variáveis iniciais declaradas no UPPAAL são colocadas como crenças do agente (relógio x e variável `activated`) nas linhas 4 e 5 do Código 1.

Para comunicar ao outro agente *User* que a porta está fechada e não foi inici-

ada a abertura, é enviada a mensagem da linha 8 do Código 1 com a estrutura presente no Código 3 [Mellado et al. 2023]. Sendo `target` o nome do agente destinatário, `directive` o que será adicionado ao `target`, utilizando `tell` para adicionar crença e `achieve` para objetivo, e `info` a crença ou objetivo a ser enviado.

```
1     self.send(<target>, <directive>, <info>)
```

Código 3. Comunicação entre Agentes em MASPY

Para representar uma transição de estados do autômato no código de agente, o plano é executado traduzindo o estado em que o autômato se encontra como o evento gatilho, e as mensagens recebidas ou as restrições de relógio ou variáveis como o contexto (opcional). Por exemplo, no primeiro plano do agente *Door* representado na linha 10 do Código 1, o evento gatilho tem o objetivo (estado no autômato) `idle`, e o contexto sendo o objetivo `pushed` (mensagem no autômato) enviado pelo agente *User*.

O corpo do plano de `idle` (linhas 12 a 14) possui a representação de uma mensagem do autômato *Door1* para outra instância do autômato *Door2* como uma adição do objetivo `closed1` a um segundo agente *Door*. Além disso, a crença `activated` recebe uma atualização de valor para `true`. E na linha 14 a adição de um novo objetivo, que seria a representação da transição para o estado `wait`.

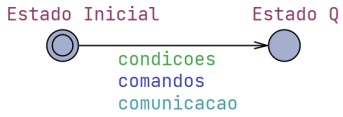

Figura 2. Visão Geral do Autômato

5. Conclusão

Com o objetivo de propor uma transformação de um autômato com verificações de tempo para um sistema de agentes de pensamento mais racional e detalhado, ainda incluindo uma estrutura de decisão baseada no tempo, e tomando como referência as estruturas de [Markovicz et al. 2024] e [Mellado et al. 2023], foi descrita uma proposta de tradução de forma teórica no formato MASPY de um autômato temporal para um agente BDI utilizando como exemplo o autômato *2doors* em UPPAAL. A tradução de um arquivo XML (UPPAAL) para um código de fato em Python no framework MASPY será realizada em trabalhos futuros.

Agradecimentos: Este trabalho tem financiamento do projeto: 444568/2024-7, CNPq/MCTI/FNDCT Nº 22/2024, *Programa Conhecimento Brasil – Apoio a Projetos em Rede com Pesquisadores Brasileiros no Exterior.*

Referências

Alur, R. (1999). Timed automata. In *Lecture Notes in Computer Science*.

Alves, G. V., Dennis, L., and Fisher, M. (2021). A double-level model checking approach for an agent-based autonomous vehicle and road junction regulations. *Journal of Sensor and Actuator Networks*.

Bratman, M. E. (1987). *Intention, plans, and practical reason*. Harvard University Press, Cambridge, Mass.

de Silva, L., Meneguzzi, F., and Logan, B. (2022). Bdi agent architectures: A survey. *arXiv preprint arXiv:2206.11990*.

Larsen, K. G., Pettersson, P., and Yi, W. (1995). Uppaal - model checker software.

Markovicz, J., Alves, G., and Borges, A. (2024). Criação de agentes bdi a partir de modelos do uppaal. In *Anais do XVIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 174–179, Porto Alegre, RS, Brasil. SBC.

Mellado, A., Fidler, I., Borges, A., and Alves, G. (2023). Maspy: Towards the creation of bdi multi-agent systems. In *Anais do XVII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 106–117, Porto Alegre, RS, Brasil. SBC.