UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GEOVÁ JUNIO DA SILVA TAVARES

UMA ABORDAGEM BASEADA NO *FLUENT CALCULUS* PARA A AXIOMATIZAÇÃO DOS CASOS JURÍDICOS DO DIREITO DAS SUCESSÕES

Recife
2022

GEOVÁ JUNIO DA SILVA TAVARES

UMA ABORDAGEM BASEADA NO FLUENT CALCULUS PARA A AXIOMATIZAÇÃO DOS CASOS JURÍDICOS DO DIREITO DAS SUCESSÕES

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional

Orientador: Frederico Luiz Gonçalves de Freitas
Coorientador: Cleyton Mário de Oliveira Rodrigues

Recife
2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

T231a Tavares, Geová Junio da Silva
Uma abordagem baseada no fluent calculus para a axiomatização dos casos jurídicos do direito das sucessões / Geová Junio da Silva Tavares. – 2022.
75 f.: il., fig., tab.

Orientador: Frederico Luiz Gonçalves de Freitas.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.

Inclui referências, apêndices e anexo.

1. Inteligência computacional. 2. Axiomatização. I. Freitas, Frederico Luiz Gonçalves de (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-156

GEOVÁ JUNIO DA SILVA TAVARES

“UMA ABORDAGEM BASEADA NO FLUENT CALCULUS PARA A AXIOMATIZAÇÃO DOS CASOS JURÍDICOS DO DIREITO DAS SUCESSÕES”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 17 de agosto de 2022.

_____

Orientador: Prof. Dr. Frederico Luiz Gonçalves de Freitas

BANCA EXAMINADORA

_____

Profa. Dra. Anjolina Grisi de Oliveira
Centro de Informática / UFPE

_____

Prof. Dr. Ricardo Cícero de Carvalho Rodrigues
Departamento de Direito / UNINASSAU

_____

Prof. Dr. Cleyton Mário de Oliveira Rodrigues
Faculdade de Formação de Professores de Garanhuns / UPE

Dedico esse trabalho a minha filha Lara por ter sido a maior motivação.

AGRADECIMENTOS

Agradeço a todos que contribuíram para esse trabalho e em especial aos orientadores por essa oportunidade.

RESUMO

O Direito de Sucessão brasileiro passa por um processo de longa transformação que, naturalmente, acompanha a evolução dos valores morais de uma sociedade. Essa evolução, porém, traz diversas regras e exceções que causam confusão entre leigos e até mesmo pelos operadores do Direito. Além das diversas regras e exceções existentes no texto legal, para algumas situações a legislação ainda é omissa e fica a cargo do judiciário decidir de acordo com o caso concreto, tornando assim essa matéria ainda mais complexa e de difícil elucidação. As razões para querer formalizar o Direito de Sucessão com a ajuda da linguagem formal e a criação de sistemas capazes de inferir premissas em situações de elevada carga de regras e restrições são inúmeras. A formalização melhora a compreensão exata e específica de um texto porque ajuda a destacar ambiguidades, leituras ou interpretações não intencionais, aumenta a compreensão precisa e evita má interpretação do texto legal proveniente das frequentes alterações e revisões da legislação e jurisprudência. Isso resulta em uma expansão do acesso à informação, crescimento da produtividade, qualificação das informações, melhora no controle de prazos, aumento da eficiência e redução de custos. Nessa interdisciplinaridade do Direito e da Inteligência Artificial o propósito é desembaraçar problemas com a complexidade dos sistemas jurídicos, as antinomias da norma jurídica, o alto volume de informações, e a modelagem do raciocínio jurídico. Este trabalho explora, portanto, como o conhecimento jurídico e o raciocínio jurídico do Direito de Sucessão se comportam ao serem formalizados pela abordagem conhecida como *Fluents Calculus*.

Palavras-chave: fluent calculus; representação do conhecimento; axiomatização; direito de sucessão; automação jurídica.

ABSTRACT

The Brazilian Succession Law goes through a process of long transformation that, naturally, follows the evolution of the moral values of a society. This evolution, however, brings several rules and exceptions that cause confusion among lay people and even legal professionals. In addition to the various rules and exceptions existing in the legal text, for some situations the legislation is still silent and it is up to the judiciary to decide according to the specific case, thus making this matter even more complex and difficult to elucidate. The reasons for wanting to formalize the Law of Succession with the help of formal language and the creation of systems capable of inferring premises in situations with a high load of rules and restrictions are numerous. Formalization improves the exact and specific understanding of a text because it helps to highlight ambiguities, unintended readings or interpretations, increases accurate understanding and avoids misinterpretation of legal text arising from frequent amendments and revisions of legislation and jurisprudence. This results in an expansion of access to information, productivity growth, information qualification, improvement in deadline control, increased efficiency and cost reduction. In this interdisciplinarity of Law and Artificial Intelligence, the purpose is to untangle problems with the complexity of legal systems, the antinomies of the legal norm, the high volume of information, and the modeling of legal reasoning. This work explores how the legal knowledge and legal reasoning of the Law of Succession behave when formalized by Fluents Calculus.

Keywords: fluent calculus; knowledge representation; axiomatization; succession law; legal automation.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Sucessão implica na ideia de transmissão de bens (TELLES, 1985). Entende-se sucessão por substituir, tomar o lugar de outrem na esfera dos fenômenos jurídicos. De forma abstrata, na sucessão, há a substituição do titular de um direito. Há de se delimitar uma divisão do conceito jurídico de sucessão que se faz entre vivos, como em um contrato de compra e venda, e a sucessão que deriva da morte (*causa mortis*), momento em que os direitos e obrigações da pessoa que morre são transferidos para seus herdeiros e legatários. Há, então, nessa segunda definição de sucessão uma noção de continuação dos direitos da pessoa falecida (o autor da herança) para a pessoa do sucessor (o legatário) (VENOSA, 2018).

Para esse estudo foi considerada a íntima relação e dependência do direito de sucessão com o direito de família, este também positivado no Código Civil de 2002 (BRASIL, 2002a). Isso porque o direito de família traz conceitos e definições que refletem nas regras e restrições daquele. Sinteticamente, a família pode ser definida como o organismo social a que pertence o homem pelo nascimento, casamento, filiação ou afinidade, que se encontra inserido em determinado momento histórico, observada a formação política do Estado, a influência dos costumes e da civilização (MALUF, 2010).

A família abrange os indivíduos chamados pela lei para herdar uns dos outros. São eles: os parentes em linha reta, ascendentes e descendentes *ad infinitum*, os cônjuges, os companheiros e os colaterais até o 4º grau, segundo a disposição constante nos artigos 1829, I a IV – que institui o ordem da vocação hereditária; art.1839; art 1843 e art. 1790 do Código Civil Brasileiro (BRASIL, 2002a). Inovou o Código Civil atual em considerar como herdeiro necessário não só os descendentes e os ascendentes, mas também o cônjuge sobrevivente em seu art. 1845. Inovou também a atual Constituição Federal em reconhecer como entidade familiar não só a família “legítima” constituída pelos laços matrimoniais, mas também aquela oriunda da união estável e da monoparentalidade, conferindo a estas um caráter de legitimidade (DINIZ, 2007).

Em Leite (2003) aponta-se o crescimento do conceito de família nuclear sobre a consideração da família patriarcal, sendo reflexo das mudanças oriundas do Código Civil de 2002 nas regras sobre a partilha de bens, visando a garantia de maior segurança para o cônjuge sobrevivente, “valorizando situações jurídicas

existenciais, beneficiando aqueles que ajudaram a construir um patrimônio e que pela sistemática do antigo Código Civil, acabavam sendo prejudicados na divisão dos bens, passando a privilegiar o círculo familiar mais restrito". Observa-se, portanto, que o direito de sucessões abrange diversas singularidades e estas, quando não bem observadas, podem levar ao atraso em possíveis tomadas de decisão, levando ao acúmulo de processos nas mais diversas esferas do direito.

De acordo com dados do Conselho Nacional de Justiça (CNJ), no ano de 2019 o Brasil registrava cerca de 77,1 milhões de processos em tramitação à espera de uma solução definitiva. Nesse mesmo ano de 2019, o Poder Judiciário recebeu 20,2 milhões de ações originárias, o que corresponde a um percentual de 3,3% a mais que o ano anterior. Além desse congestionamento, conta-se ainda com a morosidade, que refere-se ao tempo de tramitação do processo. De acordo com o relatório de 2020 do CNJ, o Poder Judiciário apresenta um tempo médio de tramitação dos processos pendentes de cinco anos e dois meses (BARCELLOS, 2021).

Assim o Direito de Sucessão surge nessa pesquisa como um domínio para a aplicação da Inteligência Artificial, que tem o potencial de impactar as profissões jurídicas. Em estudo desenvolvido por Maranhão (2021) é possível observar que a Inteligência Artificial (IA) é um tema cada vez mais presente nos debates jurídicos brasileiros. Desse modo entende-se que para a construção de uma justiça célere e eficiente, que facilite o acesso à justiça, deve se utilizar da tecnologia, em especial do uso da IA (FEFERBAUM; SILVA, 2018).

Para representar uma aplicação no domínio jurídico, podemos destacar o paradigma da Inteligência Artificial dos sistemas baseados em conhecimento que são sistemas inteligentes que realizam inferências a partir de representações internas de conhecimento. Essas representações do conhecimento são construídas de tal forma que o sistema pode atualizar sua base de conhecimento a partir de informações coletadas do ambiente, aplicando as mesmas regras de inferência para lidar com situações semelhantes (RUSSELL; NORVIG, 2010).

Em Ciência da Computação, um algoritmo é qualquer procedimento de computador bem definido que possua algum valor agregado na qualidade de suas entradas (*input*), gerando outros valores na saída (*output*), de forma que pode ser utilizado como uma ferramenta para resolver um determinado problema (CORMEN

et al., 2009). Entre outras aplicações, os algoritmos são empregados em programas de computador para a tomada de decisões e alocação de recursos a partir de grandes conjuntos de dados. Entre esses algoritmos, destaca-se particular crescimento nos últimos anos os chamados algoritmos de Inteligência Artificial, que utilizam técnicas específicas para a construção de sistemas capazes de agir racionalmente diante de situações específicas (MANZANO; OLIVEIRA, 2005).

Como essa crescente evolução da ideia de família que reflete na sucessão, é natural surgirem novas regras que definem esse fenômeno, o que pode causar uma sobrecarga de restrições que dificultam o entendimento acerca desse processo sucessório. Dessa forma a construção de sistemas inteligentes artificiais se torna essencial para o controle e automação desses processos, decidindo automaticamente sobre quem é um herdeiro, qual fração da herança lhe é cabida, facilitando assim a atividade sucessória e o acesso à informação (NAKAMITI, 2009).

A construção de agentes inteligentes, com base em lógica de predicados, tem sido uma das vertentes do estudo da Inteligência Artificial. Assim o estudo de um formalismo que seja capaz de raciocinar sobre as diferentes e atualizadas regras e exceções que permeiam o direito de sucessão é imperativo para o desenvolvimento de sistemas que possam contribuir para a automação desse processo de definir quem são os herdeiros e suas respectivas partes na herança (McCARTHY, 1963).

Neste contexto, este presente estudo foca no Cálculo de Fluente (THIELSCHER, 2000) (do inglês, *Fluent Calculus*) uma axiomatização em Lógica de Segunda Ordem, capaz de uma grande variedade de tarefas de raciocínio. O *Fluent Calculus* foi inicialmente proposto como uma linguagem de especificação e sistema para resolver dificuldades intrínsecas enfrentadas na robótica cognitiva, como incerteza, não determinismo, ramificações e concorrência.

1.1 CONTEXTUALIZAÇÃO E PROBLEMÁTICA

O direito sucessório surge, inicialmente, com as formações familiares (MEDEIROS, 1997). O conhecimento da evolução histórica do direito das sucessões é qualificado a partir do direito romano. A Lei das XII Tábuas concedia liberdade ao *pater familias* (Termo da Roma Antiga referente ao pai de família) de dispor dos seus bens para depois da morte. Mas, se falecesse sem testamento, a sucessão seguia um rito respeitando a prioridade de três classes de herdeiros: *sui*, *agnati* e *gentiles*.

Os *heredi sui et necessarii* eram os filhos sob o poder do pater e que se tornavam *sui iuris* com sua morte: os filhos, os netos, incluindo-se também, nessa qualificação, a esposa. Os *agnati* eram os parentes mais próximos do falecido. Entende-se estes como o colateral de origem exclusivamente paterna, como o irmão consanguíneo, o tio que fosse filho do avô paterno, e o sobrinho, filho desse mesmo tio. A herança não era deferida a todos os *agnados*, mas ao mais próximo no momento da morte (*agnatus proximus*) (GOMES, 2002).

Em seguida, a religião e o parentesco sistematizaram a lógica do processo sucessório. A partir daí os integrantes da família eram definidos de acordo com a sua orientação religiosa, assim, a relação de parentesco era firmada pelo culto no qual os integrantes da família pertenciam. A religião então era a fundamentação do direito sucessório (RIZZARDO, 2005).

No Brasil, várias leis foram implementadas para versarem sobre o tema das sucessões. Destaca-se a Lei Feliciano Pena (BRASIL, 1907d), a qual fez mudança significativa na vocação hereditária, termo que se refere a convocação da pessoa com direito à herança para que receba o patrimônio deixado pela pessoa falecida, acrescentando o cônjuge e garantindo-o prioridade sobre os demais (OLIVEIRA, 1952). Com a entrada em vigor do Código Civil, Lei 3.071 (BRASIL, 1916c), ficou entendido que somente a família constituída do casamento e filhos dentro do casamento seriam legítimos para gozo dos direitos de sucessão.

A partir da constituição cidadã de 1988 e do conjunto de decisões e entendimentos jurisprudenciais e mesmos doutrinários acerca do tema, essas e outras discriminações foram superadas diante da própria evolução da sociedade (DANTAS, 1991). A Constituição Federal trouxe duas importantes disposições atinentes ao direito sucessório: a do art. 5º, XXX, que inclui entre as garantias fundamentais o direito de herança; e a do art. 227, § 6º, que assegura a paridade de direitos, inclusive sucessórios, entre todos os filhos, havidos ou não da relação do casamento, assim como por adoção. A Lei n. 10.406, de 10 de janeiro de 2002, instituiu o vigente Código Civil Brasileiro, apresentando, como mencionado, inúmeras inovações, destacando-se a inclusão do cônjuge como herdeiro necessário e concorrente com descendentes e ascendentes (GONÇALVES, 2020).

A própria ideia de família tomou outra dimensão no contexto atual, estendendo-se além da família tradicional, proveniente do casamento, para outras

modalidades, as vezes informais, tendo em vista o respeito à dignidade do ser humano, o momento histórico contemporâneo, a evolução dos costumes, o diálogo internacional, a descoberta de novas técnicas científicas, a tentativa da derrubada de mitos e preconceitos (MALUF, 2010).

Diante da constante evolução das regras e exceções existentes no direito das sucessões, que formam um conjunto complexo de informações, muitas vezes torna limitado o direito fundamental de acesso à justiça que é expressamente previsto na própria Constituição Federal Brasileira (BRASIL, 1988a). Neste contexto, a Inteligência Artificial pode ser utilizada para a otimização desse processo. Entende-se por IA uma disciplina científica que se apropria das características de processamento da computação com a finalidade de formar métodos abstratos para automatizar as atividades perceptivas, cognitivas e manipulativas através de um computador. Além disso, o computador permite que seja explorado de forma mais eficiente certas dimensões de tarefas, tanto por sua enorme capacidade de retenção de informação quanto pela sua alta capacidade de processamento. Logo, proporciona maior capacidade de lidar com tarefas mais complexas (NILSON, 2014).

O poder judiciário é um dos três poderes que compõem o Estado Democrático de Direito. Segundo Alvim (2015), o Judiciário é o poder incumbido por preservar a ordem jurídica e manter a paz social, portanto é o responsável, dentre outras funções que exerce, por obter a composição da lide, mediante a aplicação da lei. Acontece que em muitos casos, há uma demora nesta resolução do conflito que se dá por meio de uma sentença e com a satisfação do direito no seu cumprimento. Assim, a Justiça acaba sendo considerada morosa e gerando uma insatisfação popular (RAMOS, 2017).

Entre outras causas que podem gerar essa morosidade ao Judiciário, ressalta-se o alto número de processos judiciais em curso e sendo ajuizados diariamente, o limitado quadro de funcionários resultando em uma quantidade insuficiente de magistrados e servidores e o excesso de demandas e atribuições (POLITIZE, 2017).

Para o direito de sucessão é no processo de inventário que o juiz decidirá todas as questões de direito, tais como o inventário dos bens deixados pelo falecido, a definição dos herdeiros legítimos e testamentários, o pagamento das dívidas deixadas, o recolhimento dos tributos, a partilha entre outros. Esse processo

também é marcado pela morosidade, o que pode gerar inutilização da herança, a deterioração do patrimônio, e até mesmo o aumento dos conflitos familiares.

A tecnologia pode ser um fator de fundamental importância na celeridade e pode ser utilizada como uma aliada dos servidores e dos jurisdicionados. Um exemplo é o sistema de penhora online, fruto de um convênio entre o Banco Central do Brasil e o Judiciário, denominado BACENJUD. Um outro exemplo é o algoritmo Victor do STF baseado em aprendizado de máquinas que tem por finalidade analisar os temas de repercussão geral da Corte, contribui para dar maior eficiência na análise de processos, com economia de tempo e de recursos humanos. Nessa aplicação, as tarefas que os servidores do Tribunal levam, em média, 44 minutos, podem ser feitas em cinco segundos (PRESCOTT; MARIANO, 2019). Nesse sentido, com um maior uso de tecnologia, a informatização e digitalização dos processos, o uso de intimações por meio de serviços de mensageria privada, a realização de audiências por videoconferências, o uso de Inteligência Artificial na análise de processos e recursos pode ajudar a reduzir essa morosidade dos processos no Poder Judiciário.

1.2 QUESTÃO DE PESQUISA

Como o formalismo do *Fluent Calculus* é capaz de lidar com regras e exceções do direito de sucessão?

1.3 HIPÓTESES

O uso do formalismo *Fluent Calculus* pode ser utilizado para axiomatização e raciocínio do direito de sucessão.

1.4 OBJETIVOS

Essa dissertação tem como objetivo geral explorar a utilização do formalismo do *Fluent Calculus* no domínio do direito de sucessão, como forma de raciocinar e apoiar a tomada de decisão pelos operadores do direito. Como objetivos específicos, destacam-se:

- Levantar o conhecimento sobre direito de sucessão, a partir de pesquisas bibliográficas e documentais;
- Descrever as regras relativas à ordem de vocação;
- Descrever a crescente integração e dependência do conhecimento jurídico frente às novas tecnologias da informação;
- Desenvolver uma proposta de aplicação *Fluent Calculus* em casos de tomadas de decisões no contexto dos direitos das sucessões;
- Axiomatizar, através do *Fluent Calculus*, a norma da legislação que versa sobre a vocação hereditária do direito de sucessão;
- Avaliar a proposta desenvolvida através de estudos de casos e entrevistas com especialistas em direito de sucessões.

1.5 ORGANIZAÇÃO DO TRABALHO

Esse trabalho está organizado da forma como se segue:

- No Capítulo 2, o referencial teórico deste trabalho, são apresentados os conceitos e definições literárias sobre os objetos de estudo;
- No Capítulo 3 é apresentada a metodologia utilizada;
- No Capítulo 4 apresenta-se a axiomatização do direito das sucessões na linguagem de programação FLUX, componente indispensável para a construção do sistema, provendo a conversão do conhecimento construído para axiomas lógicos;
- No Capítulo 5 é avaliado o modelo proveniente da axiomatização do Capítulo 4, seguida da avaliação do modelo a partir de estudos de caso;
- No Capítulo 6 são apresentadas as conclusões deste trabalho, alguns trabalhos futuros e as publicações submetidas com esta pesquisa.

2 REFERENCIAL TEÓRICO

Na seção 2.1 será tratado o Direito de Sucessão com base na legislação vigente e jurisprudência relevante que versa sobre o tema. Na seção 2.2 será apresentado uma introdução ao formalismo do *Fluent Calculus*. Na 2.3 as ideias essenciais da linguagem FLUX(*Fluent Executor*), ambos originais da obra *Reasoning Robots* de Thielscher(2005), além de conceitos provenientes de artigos científicos do mesmo autor.

2.1 DIREITO DAS SUCESSÕES

O termo sucessão, em sentido geral, é uma transição na qual uma pessoa assume o lugar da outra, substituindo assim a titularidade de determinado bem. Num negócio de compra e venda, por exemplo, o comprador é a pessoa que sucede ao vendedor, adquirindo a titularidade e direitos a que este pertencia. Neste caso, trata-se de uma sucessão inter vivos. Contudo, no Direito das Sucessões, que é um dos objetos de estudo deste trabalho, o termo sucessão é empregado em sentido estrito para designar tão somente a sucessão causa mortis, a sucessão decorrente da morte de alguém (GONÇALVES, 2013).

Este ramo do Direito disciplina o conjunto de regras referente a transmissão do patrimônio do autor da herança, também conhecido como o de cujus, a seus sucessores, também chamado de herdeiros. A Constituição Federal assegura, em seu artigo quinto, inciso trinta, o direito de herança, e o Código Civil disciplina as regras do direito das sucessões (DINIZ, 2014).

A sucessão é uma relação jurídica complexa dotada de várias regras e condições, que podem também ser consideradas fases pelas quais deve passar até atingir a sua finalidade, que é a transferência de titularidade dos bens de alguém que morreu à seus herdeiros. Tais fases podem ser definidas como a morte do autor da herança ou abertura da sucessão, a devolução da herança ou vocação hereditária, sobrevivência e idoneidade do sucessor, aquisição ou aceitação da herança (CAMPOS, 1997).

Destarte, o direito de sucessão é ramo integrante da parte especial do Direito Civil que trata da alocação de todos os bens e obrigações contraídas pelo de cujus, cumprindo-nos ressaltar que tal ramo do direito abriga apenas as relações entre

pessoas físicas, vez que apenas estas podem exprimir suas disposições de última vontade. Direito hereditário ou das sucessões, como visto, é o complexo dos princípios segundo os quais se realiza a transmissão do patrimônio de alguém que deixou de existir. Essa transmissão constitui a sucessão; o patrimônio transmitido denomina-se herança; e quem a recebe se diz herdeiro (BEVILAQUA, 1899). Nas seções seguintes serão descritos elementos e as fases essenciais do fenômeno sucessório.

2.1.1 Conceitos e Espécies

A transferência de direitos e obrigações pode acontecer por ato realizado ainda pelo seu titular em vida, assim como em razão da morte. A primeira é conhecida como transmissão inter vivos, enquanto a segunda, como transmissão *mortis causa*. Quanto à origem dessa sucessão em razão da morte de alguém, pode ser classificada como legítima ou testamentária conforme art. 1.786 do Código Civil Brasileiro (BRASIL, 2002a).

A sucessão legítima, também doutrinariamente conhecida como *ab intestato* (sem testamento), é aquela derivada imediatamente da lei, que se encarrega de indicar quais pessoas serão consideradas titulares hereditários. Ocorrerá sempre que o falecido não tiver deixado testamento ou quando este negócio jurídico for julgado nulo ou caduco. A sucessão testamentária, por sua vez, é definida pela disposição de última vontade do de cujus, expressa em testamento, elaborado de acordo com as condições estabelecidas por lei, no qual o próprio autor da herança elege os seus sucessores (VENOSA, 2010a).

2.1.2 Da abertura da Sucessão, do Autor da Herança e dos Herdeiros

O Código Civil no seu artigo sexto define que: “A existência da pessoa natural termina com a morte; presume-se esta, quanto aos ausentes, nos casos em que a lei autoriza a abertura de sucessão definitiva.” Nesse momento, abre-se a sucessão e o direito subjetivo da herança é passado automaticamente aos herdeiros do de cujus. (CC, art. 1.784). Essa transmissão automática consiste no princípio da *saisine*, segundo o qual com a morte do autor da herança transmite-se ao sucessor o domínio e a posse da herança. Importante ressaltar que apesar do dessa

transmissão ser automática, a herança é tratada como uma bem imóvel e indivisível para os herdeiros até o momento da partilha (VENOSA, 2003).

Maria Helena Diniz, na obra Direito Civil Brasileiro, leciona que: “A morte natural é o cerne de todo direito sucessório, pois ela determina a abertura da sucessão, uma vez que não se compreende sucessão, sem o óbito do de cujus, dado que não há herança de pessoa viva". (DINIZ, 2004). Assim, verifica-se que, os momentos, morte, abertura e transmissão da herança aos herdeiros, se dão em um mesmo momento (SILVA, 2012).

A morte é a causa do fenômeno sucessório. O agente principal que com seu falecimento dar início a esse processo de sucessão é chamado autor da herança ou de cujus, também podendo ter outras denominações, como falecido, morto, defunto ou finado. Por sua vez, os que recebem o patrimônio deixado pelo de cujus são qualificados como sucessores ou herdeiros. Por fim, o patrimônio que é conjunto de bens, direitos e obrigações, que alguém deixa ao morrer é denominado herança ou acervo hereditário, podendo também sê-lo, na ótica processual, espólio (MONTEIRO, 2016).

De modo geral, herdeiro é o sucessor que recebe a totalidade ou fração aritmética do patrimônio do autor da herança, ou seja, herdeiros são os sucessores da pessoa falecida. Por sua vez, os herdeiros são classificados como legítimos ou testamentários, existindo entre aqueles, ainda, os necessários e os facultativos. É importante ressaltar que o objeto desse trabalho se concentra na identificação desses herdeiros legítimos, tendo em vista que são neles que recai a maior complexidade de regras em sua determinação e vocação (THEODORO, 2016).

O herdeiro pode ser designado por lei ou por testamento. E o legatário pode sê-lo apenas por ato do testador. Assim, na sucessão legítima, encontraremos somente herdeiros, enquanto, na sucessão testamentária, poderemos nos deparar com herdeiros e legatários, inclusive em concorrência.

Os herdeiros necessários são aqueles que não podem, salvo por motivo justo, ser excluídos da sucessão por vontade do testador, pertencendo-lhes, de pleno direito, metade do acervo hereditário. São herdeiros necessários os descendentes - por exemplo, filhos e netos -, os ascendentes - por exemplo, pais e avós, o cônjuge e também o companheiro. Por sua vez, os herdeiros facultativos são aqueles que podem ser excluídos da sucessão, independentemente de motivo justo, por vontade

do testador, bastando que este disponha de todo o seu patrimônio sem os contemplar. Os herdeiros facultativos são os colaterais - por exemplo, irmãos, primos e tios (BRASIL, 2002b).

Por fim, o herdeiro após a sua vocação à sucessão, é necessário que ele decida pela a aceitação da herança ou pela sua renúncia. A aceitação é o ato pelo qual o herdeiro ratifica o recebimento do acervo hereditário (art. 1.804, Código Civil Brasileiro). Em contraponto, a renúncia ocorre quando o herdeiro declara, expressamente, que não quer aceitar, preferindo-se conservar completamente estranho à sucessão (OLIVEIRA, 1952).

2.2 FLUENT CALCULUS

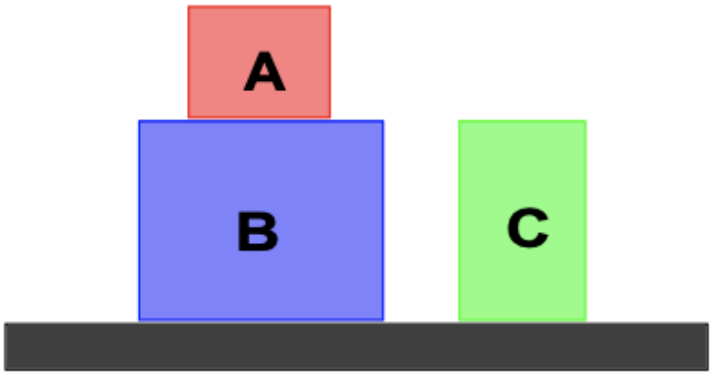
O *Fluent Calculus* é definido como um formalismo matemático de programação lógica, fundamentado em lógica de primeira ordem (ver Anexo A), para raciocinar em ambiente de domínios dinâmicos, fornecendo uma metodologia para especificar e raciocinar sobre estados e ações. Sua estrutura é baseada na junção de quatro elementos: os *fluents*, *actions*, *states* e *situations*. Um *fluent* é um componente atômico que descreve um estado que pode ser alterado no decurso do tempo. O *state* é uma coleção de *fluents* que se referem a um determinado momento de um mundo estável (ou seja, o mundo que não está em transição). As *actions* são ações elementares referentes às coisas que podem mudar o estado. Uma situation por sua vez é composta por uma sequência de ações.

As bases para o *Fluent Calculus* foram estabelecidas por Thielscher (1999) com a introdução de axiomas de atualização de estado como uma combinação de uma representação baseada em estado com elementos do *Situation Calculus* por McCARTHY (1963).

Ademais, o *Fluent Calculus* é estruturado como um formalismo de lógica de predicados que estabelece as bases teóricas para a programação de agentes inteligentes. Na lógica de predicados, a noção de objeto é usada num sentido bastante amplo. Objetos podem ser concretos (pessoa, relógio) ou abstratos (a vida). Objetos podem ainda ser atômicos ou compostos (um teclado ou um composto de teclas). Em síntese, um objeto pode ser qualquer coisa a respeito da qual pode-se dizer algo. Por convenção, nomes de objetos são escritos com inicial minúscula e é assumido que nomes diferentes denotam objetos diferentes.

Um predicado, por sua vez, denota uma relação entre objetos de um determinado contexto de discurso. Por exemplo, no contexto ilustrado na Figura 1, podemos dizer que o bloco a está sobre o bloco b usando o predicado sobre e escrevendo sobre(a, b); para dizer que o bloco b é azul, podemos usar o predicado cor e escrever cor(b, azul) e, para dizer que o bloco b é maior que o bloco c, podemos usar o predicado maior e escrever maior(b, c). Por convenção, os nomes de predicados também são escritos com inicial minúscula.

Figura 1 - Blocos empilhados sobre uma mesa.



Fonte: PEREIRA (2022, pág. 1).

Na programação desses agentes, ações significam interações destes com seus ambientes. As ações podem alterar o mundo exterior ou o status do próprio agente físico. Por exemplo, uma ação de sensoriamento pode apenas fornecer ao agente as informações de seu ambiente ou uma única ação pode ser muito complexa no nível do agente físico. As ações são tomadas como entidades elementares no nível de programação de agentes no *Fluent Calculus*. Todo programa agente produz uma sequência de ações a serem executadas em uma determinada situação que é o momento do mundo em um tempo particular (RICH; KNIGHT, 1995).

Para facilitar a compreensão acerca desse formalismo, considere um problema no qual um robô esteja em um corredor com vários escritórios em fila. O robô é um mensageiro automático, cuja tarefa é recolher e entregar correspondência interna trocada entre os escritórios. O robô está equipado com três compartimentos, cada um dos quais pode ser preenchido com uma dessas correspondências. Ademais, pode haver muito mais solicitações de entrega do que o robô pode realizar de uma só vez, dada sua capacidade limitada.

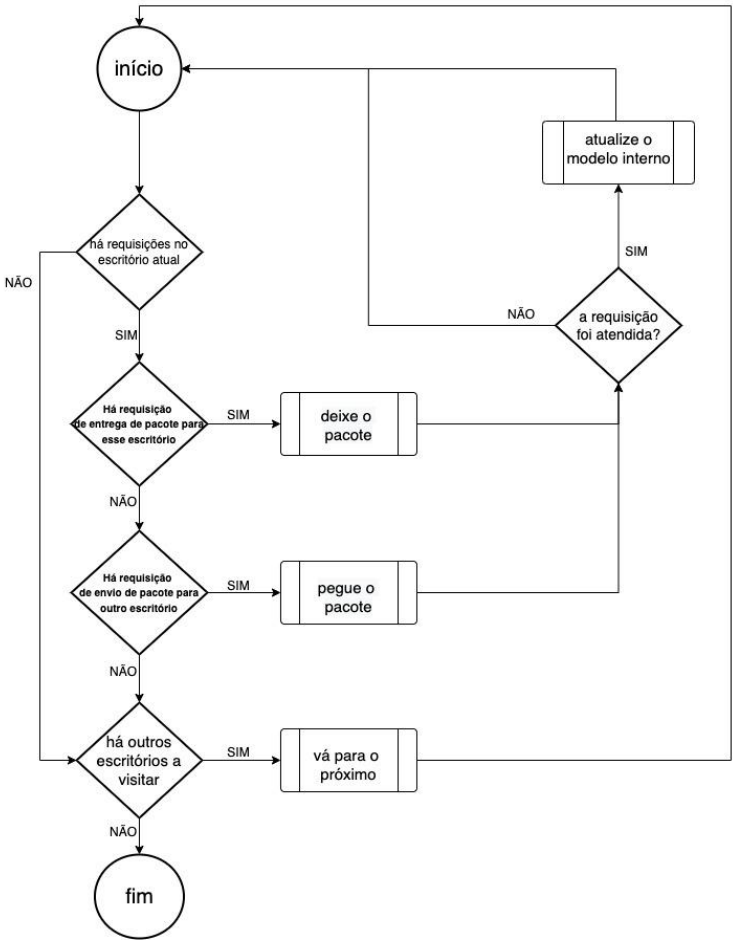
A estratégia para o robô de entrega é representada em uma algoritmo e implementada em uma pseudo-linguagem a seguir:

```
laço
      se possível deixar um pacote
            então deixe
      senão se possível pegar um pacote
            então pegue
atualize o modelo interno
      senão pode pegar ou deixar um pacote no escritório atual
            então vá para o próximo escritório
      senão pare
fim laço
```

Esse algoritmo demonstra como o robô avalia as condições que dependem do estado atual. Isso porque para decidir sobre sua próxima ação, ele precisa saber o conteúdo atual do seu compartimento, os pedidos que ainda estão abertos e sua localização atual. Como essas propriedades mudam constantemente à medida que o programa avança e nem todas podem ser detectadas diretamente, o robô precisa acompanhar o que faz à medida que se move. Para isso, ele vai manter um modelo interno do ambiente, que ao longo da execução do programa transmite as informações necessárias sobre a localização atual de todos os pacotes que ainda não foram entregues.

O modelo precisa ser atualizado após cada ação de acordo com os efeitos da ação. No que diz respeito ao cenário da Figura 3, por exemplo, se o robô começar colocando um determinado pacote em um de seus compartimentos, isso será registrado como uma modificação no modelo, ou seja, removendo a solicitação correspondente e adicionando as informações em que compartimento foi colocado o pacote em questão. Da mesma forma, sempre que um pacote é retirado de um compartimento, isso também é atualizado no modelo interno, para que o robô saiba que esse compartimento pode ser preenchido novamente.

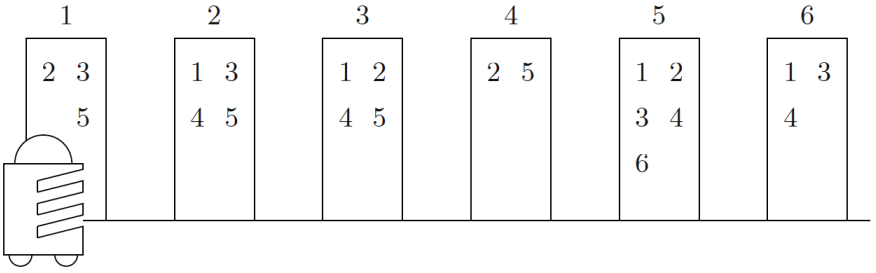
Figura 2 - Fluxograma do robô de entrega.



Fonte: Própria.

A Figura 3 mostra um cenário de exemplo em um ambiente com seis escritórios e um robô com as três malas de correio. Um total de 21 pacotes precisam ser entregues. A questão é como escrever um programa de controle que envie o robô de um lado para o outro no corredor e diga a ele onde coletar e deixar pacotes para que, no final, todas as solicitações dos escritórios sejam atendidas.

Figura 3 - O estado inicial de um problema do robô de entrega, com um total de 21 solicitações.

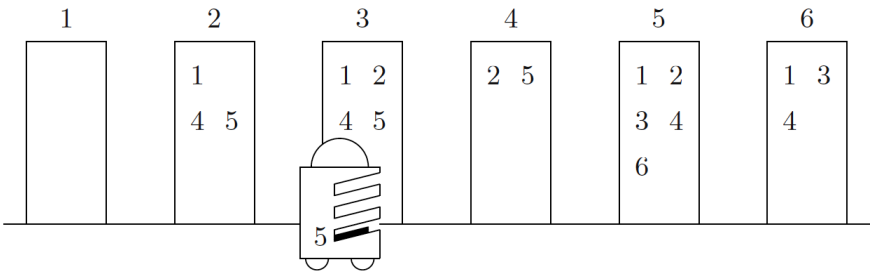


Fonte: THIELSCHER (2005, pág. 2).

A estratégia traçada para o robô é sempre que ele se encontra em algum escritório para o qual carrega um ou mais pacotes, eles são entregues. Por outro lado, se o robô estiver em algum lugar onde os itens ainda estão esperando para serem coletados, ele pega arbitrariamente o maior número possível, ou seja, até a sua capacidade. Se mais nenhum pacote puder ser descartado ou coletado em seu local atual, o robô toma uma decisão arbitrária de se mover no corredor em direção a algum escritório para o qual tenha correspondência ou onde a correspondência ainda esteja esperando para ser coletada.

Seguindo essa estratégia, o robô da Figura 4, por exemplo, escolhe pegar todos os três pacotes na primeira sala e depois passar para a sala número 2, onde pode entregar um deles. A partir daí, ele pode selecionar o pacote endereçado da sala 2 para a 3 e seguir para esta. Chegando ao escritório número 3, o robô pode entregar tanto o pacote que vem do escritório 1 quanto o do escritório 2. Isso deixa o robô com dois compartimentos vazios, que ele enche novamente, e assim por diante até finalizar todas as requisições.

Figura 4 - Estado do robô ao chegar no escritório 3.



Fonte: THIELSCHER (2005, pág. 13).

Neste programa exige que o robô de entregas avalie as condições que dependem do estado atual. Isso porque para decidir sobre sua próxima ação, ele sempre precisa saber o conteúdo atual de seus compartimentos, os pedidos que ainda estão abertos e sua localização atual. Como essas propriedades mudam constantemente à medida que o programa avança, o robô precisa acompanhar o que faz à medida que se move. Para isso, ele vai manter um modelo interno do ambiente, que ao longo da execução do programa transmite as informações necessárias sobre a localização atual de todos os pacotes que ainda não foram entregues.

O modelo precisa ser atualizado após cada ação de acordo com os efeitos da ação. No que diz respeito ao cenário da Figura 2, por exemplo, se o robô começar

colocando um determinado pacote em um de seus compartimentos, isso será registrado como uma modificação do modelo, ou seja, removendo a solicitação correspondente e adicionando as informações em que no compartimento foi colocado o pacote em questão. Da mesma forma, sempre que um pacote é retirado de um compartimento, isso também deve ser atualizado para que o robô saiba que esse espaço pode ser preenchido novamente.

O formalismo do *Fluent Calculus* tem esse nome devido um dos seus principais componentes, os *Fluents*. Assim, fornece meios para codificar modelos internos para agentes e descrever ações e seus efeitos. Como o nome sugere, é capaz de calcular a atualização de um modelo sempre que uma ação é executada (THIELSCHER, 2005).

Por convenção, as propriedades atômicas dos estados são chamadas de *Fluents*, sugerindo assim sua natureza fugaz com o passar do tempo. Logo um *Fluent* é um subtipo ou propriedade atômica desse estado que pode ser manipulada pelo agente. Manter um modelo do ambiente durante a execução do programa reflete a maneira como os vários *Fluents* mudam devido ao desempenho das ações. O *Fluent Calculus* recebe o nome desses componentes do estado e fornece meios para calcular as mudanças que eles sofrem quando o agente atua. De acordo com Thielscher (2005):

Definição: Um tripla $< F, \circ, \varnothing >$ é uma assinatura de estado *Fluent Calculus*, onde:

- F conjunto finito e não vazio de símbolos de função em ordenação *FLUENT*;
- $\circ$: ESTADO × ESTADO → ESTADO (geralmente escrito em notação infixa); e
- $\varnothing$: ESTADO

Exemplo: Numerando os escritórios e as correspondências do robô de entrega, um estado na entrega será descrito com a ajuda dos quatro fluentes a seguir:

Em: N → FLUENT $Em(r)$ = o robô está na sala r

Vazio: N → FLUENT *Vazio*(*b*)= compartimento do robô *b* vazio

Transporta : N × N → FLUENT *Transporta*(*b*,*r*) = compartimento *b* tem um pacote para *r*

Solicitação : N × N → FLUENT *Solicitação*(*r1*,*r2*) = solicitação de *r1* para *r2*

Assim, o estado inicial representado na figura 2 pode então ser formalizado por este termo:

Em(1) ∘ (Vazio(1) ∘ (Vazio(2) ∘ (Vazio(3) ∘ (Solicitação(1,2) ∘ (Solicitação(1,3) ∘ . . . ∘ Solicitação(6, 4) . . .)))))

Além disso, as condições que se referem a possibilidade de uma ação ocorrer no estado atual são baseadas na noção de *fluents* a serem mantidos em determinados estados. Por exemplo, é impossível para o robô de entrega colocar algum pacote no seu compartimento *b* se o *fluent* *Vazio(b)* não se mantiver no estado atual. De uma mesma forma, o pacote de algum compartimento *b* pode ser entregue apenas no caso de *Transporta(b, r)* e *Em(r)* se manterem no mesmo escritório *r*, ou seja, o robô precisa estar no próprio escritório ao qual o pacote é endereçado.

Nos programas de agentes, as condições que se referem ao estado do mundo são baseadas na noção de que um *Fluent* deve ser mantido. Como essa noção é onipresente nas fórmulas de *Fluent Calculus*, ela é abreviada como uma Equação que segue:

$$Holds(f : FLUENT, z : STATE) = (\exists z')\ z = f \circ z'$$

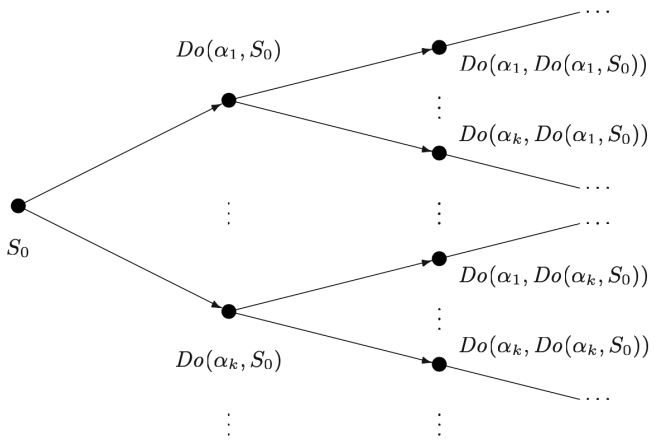
Holds(f,z) é verdade se existe um estado *z'* onde o estado *z* é uma composição do fluente *f* and *z'*.

2.1.1 Ações e Situações

Na programação de agentes, ações significam as interações do agente com seu ambiente. As ações podem mudar o mundo exterior, por exemplo, quando um robô pega um pacote ou um agente de software solicita um produto pela Internet. Outras ações apenas alteram o status do próprio agente físico, por exemplo, quando um robô se move para uma nova posição. As ações também podem apenas fornecer ao agente informações sobre o ambiente, por exemplo, quando um robô detecta se uma porta está aberta ou um agente de software compara preços em diferentes lojas online. Enquanto uma única ação pode ser um comportamento muito complexo no nível do agente físico, as ações são tomadas como entidades elementares no nível dos programas do agente e no *Fluent Calculus*.

A programação do agente é determinante para acionar o agente a fazer a ação certa no momento certo. Todo programa agente deve produzir uma sequência de ações a serem executadas pelo agente. Ao falar sobre os diferentes estágios de uma execução real de um programa, se refere à sequência de ações que foram executadas até certo ponto como uma situação. A constante especial `S0` denota a situação inicial, na qual o agente ainda não fez nenhuma interação com o mundo exterior. O construtor `Do(a,s)` então mapeia uma ação `a` e uma situação `s` para a situação após a execução da ação. Portanto, as sequências de ação são termos aninhados na forma `Do(an,... , Do(a1, S0)...)`. As situações podem ser visualizadas como os nós de uma árvore enraizada em `S0`; conforme Figura 5. Cada ramificação nesta árvore é uma execução potencial de um programa para o agente.

Figura 5 - Árvore de situações.



Fonte: THIELSCHER (2005, pág. 12).

A maioria das condições em programas de agente é avaliada em relação ao estado em que o ambiente está no momento em que a instrução do programa é executada. A função padrão *State(s)* é usada para denotar esses estados nos diferentes estágios da execução de um programa, ou seja, nas diferentes situações *s*. Completando a *signature*, o predicado padrão *Poss(a,z)* é usado para especificar as condições no estado *z* para que a ação *a* seja executável pelo agente.

Definição: Uma tupla $S \cup \langle A, S0, Do, State, Poss \rangle$ é uma assinatura de *Fluent Calculus* se:

- *S* ***assinatura*** do estado;
- *A* conjunto finito e não vazio de símbolos de função na ação de classificação;
- S_0 : SIT;
- *Do* : ACTION X SIT → SIT ;
- *State* : SIT → STATE;
- *Poss* : ACTION x STATE.

Uma ação é um termo do tipo ACTION, e situação é um termo do tipo SIT.

Exemplo: As ações do robô de entrega serão indicadas pelo seguintes três símbolos:

Pegar : N × N → AÇÃO *Pegar(b,r)* = coloque em *b* o pacote para *r*

Entregar : N → AÇÃO *Entregar(b)* = entregue o conteúdo do compartimento *b*

Ir : N → AÇÃO *Ir(d)* = se ir para o próximo escritório

Aplicado ao cenário da Figura 2, um programa pode atingir a seguinte situação após nove etapas, onde o robô já executou três das 21 tarefas de entrega (ver Figura 3):

Faça(*Entregar*(*2*), ***Faça***(*Entregar*(1), ***Faça***(*Ir*(*d*),
Faça(*Pegar*(*1, 3*), ***Faça***(*Entregar* (1), ***Faça***(*Ir*(*d*),
Faça(*Pegar*(*3, 5*), ***Faça***(*Pegar*(*2, 3*), ***Faça***(*Pegar*(*1, 2*), S_0)))))))))

2.1.2 Axiomas de atualização de estado

Os chamados axiomas de pré-condição são usados para especificar formalmente as circunstâncias sob as quais uma ação é possível em um estado.

Considere uma assinatura de cálculo fluente com funções de ação A, e seja $A \in A$. Um axioma de pré-condição para A é uma fórmula:

$$Poss(A(x),z) \equiv \Pi(z)$$

Onde $\Pi(z)$ é uma fórmula de estado com variáveis livres para x e z.

Para manter seu modelo interno atualizado, os agentes precisam conhecer não apenas as pré-condições de suas ações, mas também como elas afetam o ambiente. Com esse conhecimento, o agente pode atualizar seu modelo após cada ação para refletir as alterações efetuadas.

As mudanças que uma ação causa são especificadas por axiomas que definem a atualização dos estados. Sob a condição de que a ação $A(x)$ em questão seja possível em uma situação s, o chamado axioma de atualização de estado para esta ação relaciona o estado resultante, $State(Do(A(x),s))$, para o estado atual, $State(s)$. Mais precisamente, os efeitos positivos e negativos da ação são lançados em uma equação de atualização que codifica a diferença entre esses dois estados. Algumas ações podem ter efeitos condicionais, assim os axiomas de atualização de estado podem, portanto, combinar várias equações de atualização, cada uma das quais se aplica condicionalmente.

Thielscher define axiomas de atualização de estado que são baseados no trabalho de Reiter(1991). Ele descreve axiomas de atualização de estado, que definem os efeitos da ação A no cálculo fluente, como segue:

$$Poss(A(x),s) \wedge \Delta(x,State(s)) \supset State(Do(A(x),State(s)) = State(s) \circ \theta^{+} - \theta^{-}$$

Para a Equação, temos que:

- $\Delta(x,State(s))$ define as pré-condições adicionais para a mudança definida de *fluents*.

- θ^+ define os efeitos positivos (ou seja, os novos fluents adicionados);
- θ^- define os efeitos negativos na situação (ou seja, os fluents que deixaram de ser verdadeiros), atualizando o estado adequadamente.

Sob a suposição de que os efeitos positivos e negativos são disjuntos, os axiomas de atualização de estado descrevem apenas os *fluents* que mudam.

2.3 FLUX

A linguagem de programação FLUX, acrônimo formado pelos os segmentos *Fluent Executor*, foi desenvolvida e apresentada por Thielscher (2004). É uma linguagem construída com base no PROLOG para definição de agentes inteligentes que raciocinam sobre suas ações e que são capazes de lidar com estados incompletos utilizando o formalismo do *Fluent Calculus*. Permite escrever programas concisos e modulares para agentes que baseiam suas decisões em um modelo de mundo explícito. Para manter esse modelo atualizado durante a execução do programa, todo programa agente requer uma codificação da axiomatização do domínio subjacente, em particular os axiomas de atualização de estado.

Os programas agentes em FLUX são implementados sobre um kernel, que dota os agentes robóticos com capacidades de raciocínio geral para manter e usar seu modelo interno. Como um programa lógico, o kernel pode ser facilmente verificado em relação à semântica do *Fluent Calculus*. Estruturalmente um programa em FLUX consiste na união de três componentes: Kernel, Domínio e Estratégia. O Kernel é um conjunto de regras de tratamento de restrições junto com cláusulas que modelam os axiomas do *Fluent Calculus*. O Domínio contém uma axiomatização da aplicação, dotando o agente do conhecimento da ação necessária. Por fim, a Estratégia traça um método de acordo como o agente atua, o que pode facilitar a resolução de problemas ao longo do seu funcionamento (THIELSCHER, 2005).

O kernel do FLUX é mostrado na Figura 6 (completo no Apêndice C). Consiste em três partes: uma definição para os fluentes manterem em determinados estados, uma definição em uma cláusula para atualizar um estado por uma lista de efeitos positivos e negativos e por fim uma definição para executar uma ação.

Figura 6 - Kernel do FLUX.

```
holds(F, [F|_]).
holds(F, Z) :- Z=[F1|Z1], F\==F1, holds(F, Z1).

holds(F, [F|Z], Z).
holds(F, Z, [F1|Zp]) :- Z=[F1|Z1], F\==F1, holds(F, Z1, Zp).

minus(Z, [], Z).
minus(Z, [F|Fs], Zp) :-
   (\+ holds(F, Z) -> Z1=Z ; holds(F, Z, Z1)),
   minus(Z1, Fs, Zp).

plus(Z, [], Z).
plus(Z, [F|Fs], Zp) :-
   (\+ holds(F, Z) -> Z1=[F|Z] ; holds(F, Z), Z1=Z),
   plus(Z1, Fs, Zp).

update(Z1, ThetaP, ThetaN, Z2) :-
   minus(Z1, ThetaN, Z), plus(Z, ThetaP, Z2).

execute(A, Z1, Z2) :- perform(A), state_update(Z1, A, Z2).
```

Fonte: THIELSCHER (2005, pág. 28).

O predicado *Holds*(F, Z) é utilizado para declarar que o *Fluent* F se mantém no estado Z. Esse predicado utiliza a decomposição padrão [*Head* | *Tail*] de uma lista em seu primeiro elemento e nos elementos restantes da lista. A cláusula averigua se F está presente na *Head* do estado Z, ou, de maneira recursiva, no restante de Z.

O predicado *Update*(Z1, P, N, Z2) declara que o estado Z2 é o resultado da atualização do estado Z1 por uma lista de efeitos positivos P e uma lista de efeitos negativos N. Os efeitos negativos são inferidos primeiro, para que, se ocorrer um fluente tanto em P quanto em N, então ela se manterá no estado resultante, Z2. Efeitos positivos que são conhecidos como verdadeiros em Z1 não causam mudança, assim como efeitos negativos que são conhecidos como falsos em Z1. Esse, por sua vez, utiliza na sua composição o predicado *minus*(Z1, N, Z2), que se refere aos efeitos negativos N da passagem do estado Z1 para o estado Z2, e o predicado *plus*(Z1, P, Z2) que denota os efeitos positivos P da mudança do estado Z1 para o estado Z2.

O predicado *Execute*(A, Z1, Z2) é definido para execução da ação A e, simultaneamente, atualizar o estado atual Z1 para o estado Z2 de acordo com os efeitos de A. O programa do kernel pressupõe a definição de um predicado *Perform*(A) para fazer com que o agente realize a ação A no ambiente. Nesse predicado é necessário que o programador defina um outro predicado *StateUpdate*(Z1, A, Z2) de forma que codifique os axiomas de atualização de estado para cada ação que o agente possa executar.

2.4 TRABALHOS RELACIONADOS

Alguns estudos já trabalharam também na abordagem multidisciplinar de formalizar domínio jurídico utilizando um paradigma lógico. Em Rodrigues (2019) pode-se observar, de forma inovadora, exemplos de como formalizar a representação do conhecimento jurídico e realizar inferências lógicas utilizando os padrões da Web Semântica. No seu trabalho, após um mapeamento sistemático, foi desenvolvido uma ontologia que representa o conhecimento da teoria geral do crime e outra construção ontológica denominada OntoCrime, esta que visa o formalizar conceitos gerais do Código Penal (BRASIL, 1940c).

Além dessas contribuições, destaca-se o trabalho de Oliveira (2019) desenvolvido no ambiente da pós graduação em Direito que contribuiu com um estudo de elementos técnicos e políticos para a implementação da automação de parte do exercício jurisdicional no Brasil além de realizar um experimento desenvolvendo uma ontologia para formalizar parte do Direito do Consumidor brasileiro (BRASIL, 1990), favorecendo, assim, o crescimento dessa interdisciplinaridade.

No âmbito da justiça brasileira, já é realidade as tecnologias de Inteligência Artificial. O Watson, uma plataforma criada pela IBM, foi implantada em um escritório de advocacia localizado em Recife, sendo utilizado para automatizar serviços repetitivos. Uma das suas atribuições é avaliar todo o conteúdo das mais de 100 mil peças para identificar pedidos e demandas a fim de classificar a complexidade de um processo (NUNES; MARQUES; RUBINGER, 2018). Outra ferramenta nesse sentido é a Assistente Digital do Promotor, trata-se de software desenvolvido pela empresa Softplan, cuja função é auxiliar na organização do volume de processos,

sendo capaz de fornecer análises eficientes e objetivas, auxiliando o trabalho dos promotores. O sistema facilita o entendimento dos casos e a construção de peças processuais que, após finalizadas de maneira estruturada, podem ser enviadas ao Poder Judiciário (HOFFMAN, 2018).

O Supremo Tribunal Federal, mais alta instância do poder judiciário brasileiro, já possui duas experiências com sistemas inteligentes. Trata-se do Victor que é uma Inteligência Artificial voltada para apoiar a atividade de análise de admissibilidade recursal, mediante sinalização de que um dado tema de repercussão geral, ou mais de um, se aplica ao caso dos autos. Nesse ano de 2022, também foi anunciado pela suprema corte o projeto RAFA que utiliza mecanismos de *machine learning*, *deep learning*, para aprender tarefas cognitivas de uma grande quantidade de dados e classificá-los permitindo levar os processos prioritários para a pauta com maior velocidade (SILVA et al., 2019).

Outra instituição nesse sentido a implantar um sistema inteligente foi a Advocacia Geral da União, trata-se do Sapiens. O sistema auxilia o procurador, simplificando a produção de peças processuais, de forma automatizada, o que auxilia no trabalho dos servidores, reduzindo a necessidade de seu trabalho manual. O software inteligente também sugere teses jurídicas a serem utilizadas em determinado caso concreto (LOURENÇO; MAIRINK; ALMEIDA, 2020).

Contudo, até onde se sabe, investigar as potencialidades e limites do Cálculo de Fluentes é inovador. Não foi encontrado na literatura nenhum estudo neste sentido, e entendemos que, dadas as novas possibilidades que surgem no âmbito na automação de tarefas judiciais, e considerando as tarefas de raciocínio inerentes do cálculo de fluentes, constatamos a possibilidade de enveredar por esta área de pesquisa. É possível, desta forma, entender possíveis fragilidades, bem como possibilidades de novos serviços de raciocínio na esfera do Direito.

3 METODOLOGIA

Neste capítulo será apresentada a metodologia e os métodos empregados no desenvolvimento deste trabalho. No Quadro 1 estão dispostas as fases metodológicas dessa pesquisa.

Quadro 1 - Fases metodológicas.

Fase	Ações principais	Método
Fase 1	Coletar informações junto às publicações de outros pesquisadores sobre as aplicações do *Fluents Calculus*, estudos na área da automação jurídica e aplicação	Revisão bibliográfica e documental
Fase 2	Entender no estudo de caso o fenômeno sucessório para explorar no seu próprio contexto os mesmos elementos que foram base da revisão bibliográfica	Estudo de caso único com uma unidade de análise
Fase 3	Entender a complexidade da matéria de sucessão através do estudo da legislação, doutrina e entendimentos jurisprudenciais	Análise documental
Fase 4	Obter uma apreciação, junto a especialistas na área do direito de sucessão, sobre a validade do modelo proposto, com o intuito de otimizá-lo para que retrate mais precisamente a realidade	*Member Checking*

Fonte: Própria.

3.1 DA MODALIDADE DE PESQUISA

A abordagem qualitativa centra-se na identificação das características de situações, eventos e organizações (LLEWELLYN; NORTHCOTT, 2007). De acordo com Liebscher (1998), a abordagem qualitativa é viável quando o fenômeno em estudo é complexo, de natureza social e de difícil quantificação. Para usar adequadamente a abordagem qualitativa, precisa-se aprender a observar, analisar e registrar as interações entre as pessoas e entre as pessoas e o sistema. Nesse tipo de abordagem há uma interação dinâmica “entre o mundo real e o sujeito, isto é um vínculo indissociável do mundo objetivo e a subjetividade do sujeito que não pode ser traduzida em números” (SILVA; MENEZES, 2005).

3.2 DO ESTUDO DE CASO

Nessa categoria de pesquisa o objeto é analisado de forma profunda visando à descoberta, atento a novos elementos que possam surgir. Durante o estudo, a apreensão se faz mais completa do objeto, é preciso levar em conta o contexto em que ele se situa. Assim, para compreender melhor a manifestação geral de um problema, as ações, as percepções, os comportamentos e as interações das pessoas devem ser relacionadas à situação específica onde ocorrem ou à problemática determinada a que estão ligadas. Assim retratar a realidade de forma completa e profunda. Revelando a multiplicidade de dimensões presentes, focalizando-o como um todo. Nesse tipo de abordagem enfatiza a complexidade natural das situações, evidenciando a inter-relação dos seus componentes (LÜDKE; ANDRÉ, 1986).

Como exposto anteriormente, este trabalho explora a capacidade de formalizar regras e exceções da legislação pertinente ao direito de sucessão, mais especificamente a vocação hereditária, com o formalismo do Fluent Calculus. Apesar do esforço dessa dissertação em tentar formalizar esse processo de sucessão, vale destacar que aqui é tratado na sua regra geral com base na legislação e em jurisprudência vigente e que o Direito, por si só, tem uma forte característica de ser subjetivo e cada caso deve ser analisado na sua particularidade.

Para tanto, dentro de uma abordagem de pesquisa qualitativa, (STRAUSS, CORBIN, 2008), o método de pesquisa, estudo de caso, foi adotado porque "em geral os estudos de caso representam estratégia preferida quando se colocam questões do tipo “como e por que”, quando o pesquisador tem pouco controle sobre os acontecimentos e quando o foco se encontra em fenômenos contemporâneos inseridos em algum contexto da vida real" (YIN, 2005, p. 19).

Para este estudo foi analisado um órgão público responsável por processos de inventários, que tratam sobre a partilha de bens deixados por familiares falecidos a seus respectivos herdeiros, e a unidade de análise foram dois servidores que lidam diariamente com o direito em questão.

3.3 *MEMBER CHECKING*

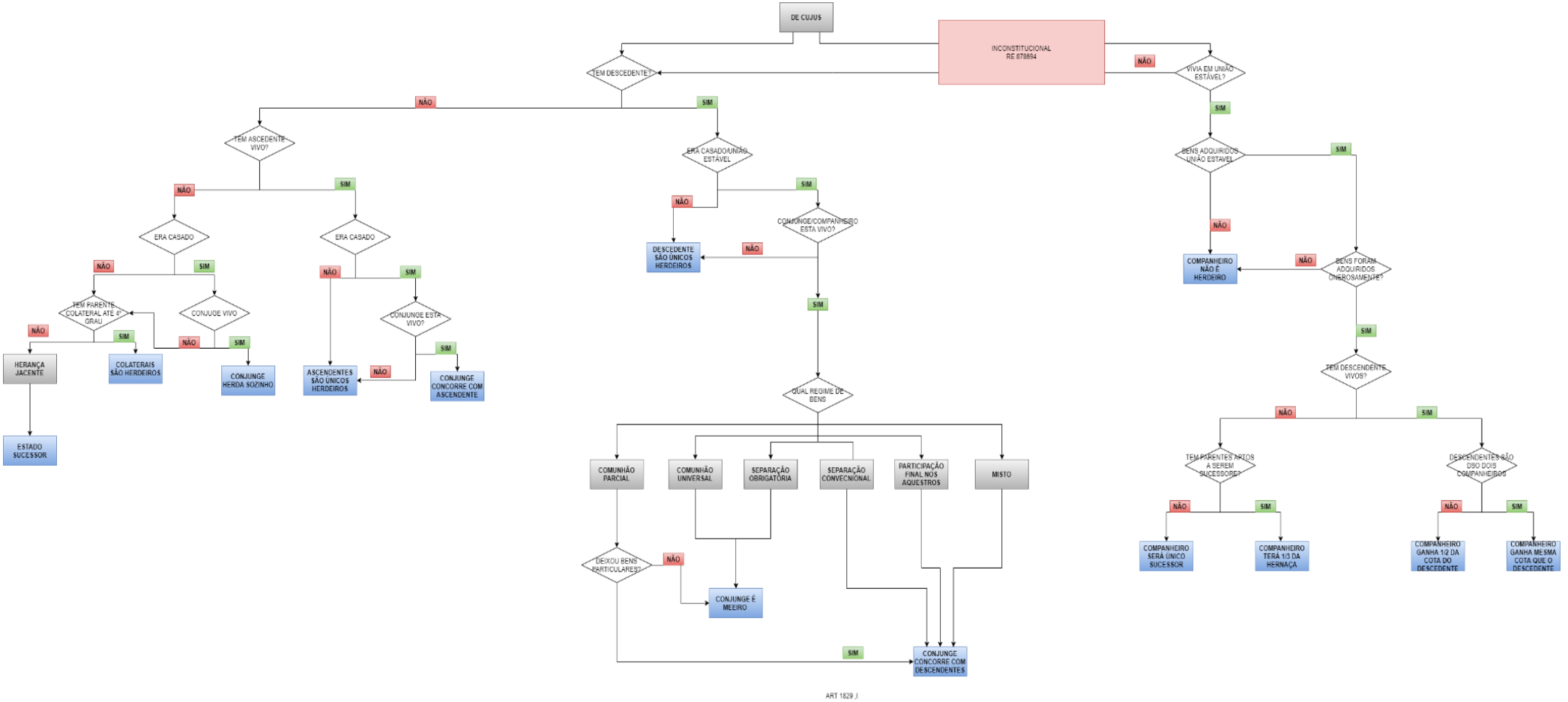
A técnica de pesquisa *Member Checking* é utilizada em pesquisa qualitativa como meio de manter a validade, credibilidade e ajudar a melhorar a precisão em um estudo. Permite que aos participantes do estudo a oportunidade de confirmar ou negar a precisão e interpretações dos dados, agregando credibilidade ao estudo qualitativo.

Nessa fase foi apresentado o modelo proposto neste trabalho para ser avaliado por especialista quanto às propriedades que compõem os axiomas lógicos. Assim com essa técnica melhora a precisão de que o formalismo retrata a realidade.

3.4 *DOS PROCEDIMENTOS METODOLÓGICOS*

Para o desenvolvimento deste trabalho, além dos métodos anteriores empregados, foram realizadas pesquisas bibliográficas e documentais em leis, doutrinas e jurisprudência para levantar o conhecimento necessário sobre o Direito de Sucessão. O resultado desse levantamento foi compilado na Figura 7.

Figura 7 - Fluxograma Direito de Sucessão.



Fonte: Própria.

4 AXIOMATIZAÇÃO DO DIREITO DAS SUCESSÕES

Nesta seção é apresentada a axiomatização do direito de sucessão, de forma concentrada e objetivando a ordem da vocação hereditária definida no Livro V, Título 2, Capítulo 1 do Código Civil Brasileiro (BRASIL, 2002a). Os axiomas foram estruturados na sintaxe da linguagem de programação FLUX versão 3.1, estando dispostos de forma resumida nos quadros a seguir (a axiomatização completa está disposta no apêndice A), em três colunas. Cada coluna informa respectivamente o nome do axioma, sua sintaxe em FLUX e seu significado.

A ferramenta utilizada para a implementação dessa linguagem foi o SWI-Prolog na sua versão 8.4.3-1 disponível em (https://www.swi-prolog.org/download/stable).

Na Tabela 1 contém os axiomas elementares(*Fluents*), que serão utilizados para construir outros predicados das Tabelas 2 e 3. Nele é descrito elementos necessários para construir as regras de pré-condição e atualização de estado.

Tabela 1 - *Fluents* do Direito de Sucessão

Nome	**Sintaxe**	**Significado**
Autor de Herança	`autor_heranca(X)`	Falecido que deixou bens
Pessoa	`pessoa(X)`	Ser humano dotado de capacidade, provido de direitos e obrigações
Morto	`morto(X)`	Pessoa morta
Vivo	`vivo(X)`	Pessoa viva
Masculino	`masculino(X)`	Pessoa do sexo masculino
Feminino	`feminino(X)`	Pessoa do sexo feminino
Cônjuge	`conjuge(X, Y)`	Pessoa unida por matrimônio

Fonte: Própria.

Na Tabela 2 é axiomatizada a estrutura de uma árvore genealógica. Neste é definido, por exemplo, o que é um pai, filho ou descendente. Essas regras são necessárias para que o próximo módulo possa fazer a correta inferência lógica.

Tabela 2 - Axiomas de árvore genealógica

Nome	**Sintaxe**	**Significado**
Pais	`pais(X, Y)`	Ascendente de primeiro grau de uma pessoa

Pai	`poss(pai(X,Y), Z) :-`
	`holds([masculino(X), pais(X,Y)], Z).`
	
	`state_update(Z1, pai(X, Y), Z2) :-`
	`update(Z1, [], [pai(X, Y)], Z2).`
		Pessoa do sexo masculino que é ascendente de primeiro grau

Mãe	`poss(mae(X,Y), Z) :-`
	`holds([feminino(X), pais(X,Y)], Z).`
	
	`state_update(Z1, mae(X, Y), Z2) :-`
	`update(Z1, [], [mae(X, Y)], Z2).`
		Pessoa do sexo feminino que é ascendente de primeiro grau

Filha	`poss(filha(X,Y), Z) :-`
	`holds([descendente(X,Y), feminino(X)], Z).`
	
	`state_update(Z1, filha(X, Y), Z2) :-`
	`update(Z1, [], [filha(X, Y)], Z2).`
		Pessoa do sexo feminino descedente de primeiro grau

Fonte: Própria.

Por fim, na Tabela 3 é definido os axiomas de pré-condição e atualização de estado necessários para definir os possíveis herdeiros na vocação hereditária.

Tabela 3 - Axiomas da vocação hereditária

Nome	**Sintaxe**	**Significado**

Herdeiros	`poss(herdeiros(X, Y), Z) :-`
	
	`or_holds([herdeiro_descendente(X,Y),`
	`herdeiro_conjuge_comunhao_parcial(X,Y),`
	`herdeiro_conjuge_separacao_convencional(X,`
	`Y), herdeiro_descendente_estirpe(X,Y),`
	`herdeiro_ascendente(X,Y),`
		Retorna todos os herdeiros existentes

	herdeiro_descendente_estirpe(X,Y), herdeiro_colateral(X,Y)], Z). **state_update**(Z1, herdeirosl(X), Z2) :- **update**(Z1, [], [herdeiros(X)], Z2)	
Vocação Hereditária	**poss**(vocacao_hereditaria(X, Y), Z) :- **or_holds**([meeiro(X,Y), herdeiros(X,Y)], Z). **state_update**(Z1, vocacao_hereditaria(X), Z2) :- **update**(Z1, [], [vocacao_hereditaria(X)], Z2)	Retorna todos os herdeiros e meeiros existentes

Fonte: Própria.

Por conseguinte, após definir os *fluents* (Tabela 1), é possível fazer a consulta com os axiomas da vocação hereditária (Tabela 3), assim sendo possível verificar os herdeiros e meeiros da sucessão. Há de se destacar alguns pontos desse formalismo: foi categorizado algumas formas de herdeiros de acordo com as circunstâncias que o levaram a essa condição. Por exemplo, o axioma *herdeiro_descendente* comporta todo aquele herdeiro que estiver na linha de descendência do *de cujus*, isto é, filho ou neto que representa e que tem direito a alguma parte na herança. Já o axioma *herdeiro_descendente_estirpe* refere-se ao herdeiro representando na herança de um descendente que morreu morrido antes do autor da herança. Essa modularização foi necessária para ser possível identificar e individualizar as regras para os herdeiros no axioma.

Para não sobrecarregar visualmente o texto, todas as regras em FLUX produzidas nesta pesquisa com suas devidas explicações constam no Apêndice A. Neste, destacamos os Fluents do Direito de Sucessão mapeados (Regime Comunhão Universal, Regime Comunhão Parcial, Regime Comunhão Separação Convencional, Regime Comunhão Separação Legal) e os Axiomas de pré condição e atualização da vocação hereditária, a saber: De cujus, Herdeiro Descendente, Herdeiro Ascendente, Herdeiro Pré-morto, Herdeiro Descendente Estirpe, Meeiro, Meeiro Universal, Meeiro Parcial, Meeiro Separação Legal, herdeiro Cônjuge

Separação Convencional, herdeiro cônjuge comunhão parcial, Herdeiro Colateral, Herdeiros e Vocação Hereditária.

5 AVALIAÇÃO DO MODELO

Nesta seção será apresentado o modelo proposto deste trabalho juntamente com sua análise a partir da aplicação no estudo de caso.

5.1 DO MODELO

Para o desenvolvimento desse modelo foram utilizados dois casos hipotéticos baseados em questões do Exame de Ordem dos Advogados do Brasil (EDITAIS E PROVAS, 2022).

5.2 CASO HIPOTÉTICO 1

O Quadro 5 descreve o caso hipotético, o questionamento com base nas regras de sucessão e a resposta resolutiva esperada.

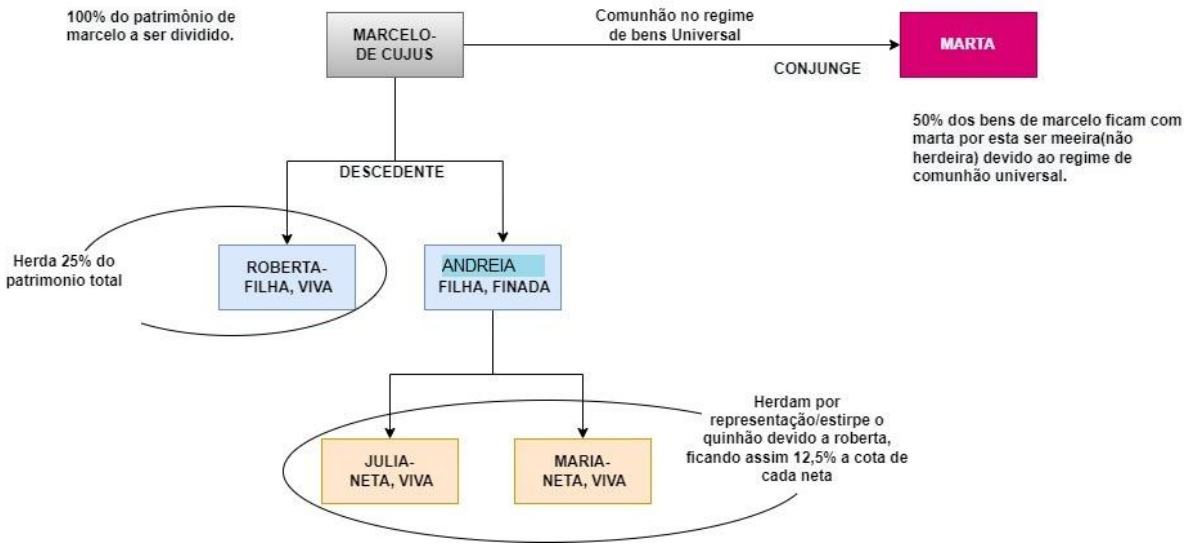
Quadro 2 - Caso hipotético 1

Caso hipotético	**Marcelo**, faleceu em 2019. O mesmo deixou **Marta**, cônjuge, que vivia em **comunhão universal de bens** e tinha como prole duas filhas, **Roberta**, viva, maior e capaz e **Andreia** já falecida em 2018, esta, por sua vez, tinha 2 filhos. Roberta também possui uma filha.
Questão	Quem são os sucessores (herdeiro ou meeiro) de Marcelo?
Resposta	Marta não participa como herdeira pois já é meeira “de metade” dos bens deixados pelo de cujus, Marta então terá direito a metade dos bens a serem sucedidos devido o regime de comunhão universal. Roberta é herdeira descendente e herdará em concorrência com as duas filhas de Andreia (a parte que cabia a Andreia, tendo em vista que Andreia é herdeira pré-morta) por estirpe (ou representação) sendo a cota dividida igualmente para cada filha.

Fonte: Própria.

Para melhor visualização e entendimento, na Figura 8 é representado em formato de diagrama o caso hipotético 1.

Figura 8 - Diagrama do caso hipotético 1.



Fonte: Própria.

Com base nesse caso hipotético e na axiomatização realizada no Capítulo 4 e no Apêndice A, foi possível analisar o comportamento do modelo proposto. Para isso, na Seção 5.2.1 foram formalizados os *fluents* com base na Tabela 1. Em seguida, na Seção 5.2.2 foram aplicadas as regras definidas na Tabela 2 (Axiomas de árvore genealógica) necessários para saber o grau parentesco entre os elementos. Por fim, na Seção 5.2.3 as condições da vocação hereditária para saber se determinada pessoa pode ser chamada a suceder.

5.2.1 *Fluents* do caso hipotético 1

Nesse momento é alimentado a base de dados definindo os elementos (as pessoas) que participam desse fenômeno sucessório, seja como autor da herança (falecido que deixou bens) ou como potencial herdeiro. Por exemplo, Marcelo, considerando ele ser a pessoa que tem bens a serem sucedidos(autor da herança), sua inserção na base de dados seria disposto como se segue:

```
autor_heranca(marcelo)
pessoa(marcelo)
morto(marcelo)
conjunge(marcelo, marta)
```

5.2.2 Regras da Árvore Genealógica do caso hipotético 1

Apresentamos aqui as regras para descobrir o parentesco entre os elementos definidos da seção anterior. Por exemplo, para saber se uma pessoa é filha do autor da herança, basta definir algumas restrições do tipo: ser do sexo feminino e ser descedente de primeiro grau do autor da herança. No formalismo abaixo foi utilizado Roberta como exemplo.

```
poss(filha(roberta, marcelo), Z)  :-
      holds([pai(marcelo, roberta), feminino(roberta)], Z)

state_update(Z1,  filha(roberta, marcelo), Z2) :-
      update(Z1, [], [filha(roberta, marcelo)], Z2)
```

Na primeira sentença para garantir que Roberta seja filha de Marcelo, expresso pelo axioma *poss,* o axioma de condição *holds* tem que ser verificada como verdade, dentro de axioma marcelo tem que o pai e roberta ser do sexo feminino. Assim com esses axiomas sendo verdade o estado é atualizado do Z1 para o Z2.

5.2.3 Regras da Vocação Hereditária do caso hipotético 1

Por fim, as regras da vocação hereditária se utilizarão dos dois módulos anteriores para definir quem é o possível herdeiro ou meeiro da sucessão. No exemplo abaixo, verifica-se se Roberta é possível herdeira de Marcelo. Para tanto, é necessário que a pessoa esteja viva e esteja na linha de descendência do de cujus(autor da herança):

```
poss(herdeiro_descendente(roberta), Z) :-
      holds(descendente(roberta), Z), holds(vivo(roberta), Z),
      holds(decujus(marcelo), Z)

state_update(Z1,  herdeiro_descendente(roberta), Z2) :-
      update(Z1, [], [herdeiro_descendente(roberta)], Z2)
```

Assim, para o exemplo acima, para verificar se Roberta é de fato herdeira de Marcelo, foi realizado na ferramenta SWI-Prolog a consulta utilizando o axioma *state_update*, onde só será possível a atualização do estado Z1 para o estado Z2 se as condições forem verificadas como verdade. Essa consulta com o seu retorno pode ser verificada na Figura 9.

Figura 9 - Consulta SWI-Prolog caso 1.

```
?- state_update(Z1, herdeiro_descendente(roberta, marcelo), Z2).
Z1 = [descendente(roberta, marcelo)|_A],
Z2 = [herdeiro_descendente(roberta, marcelo), descendente(roberta, marcelo)|_A],
```

Fonte: Própria.

5.3 CASO HIPOTÉTICO 2

Por conseguinte, o Quadro 3 descreve o segundo caso hipotético, o questionamento com base nas regras de sucessão e a resposta resolutiva esperada.

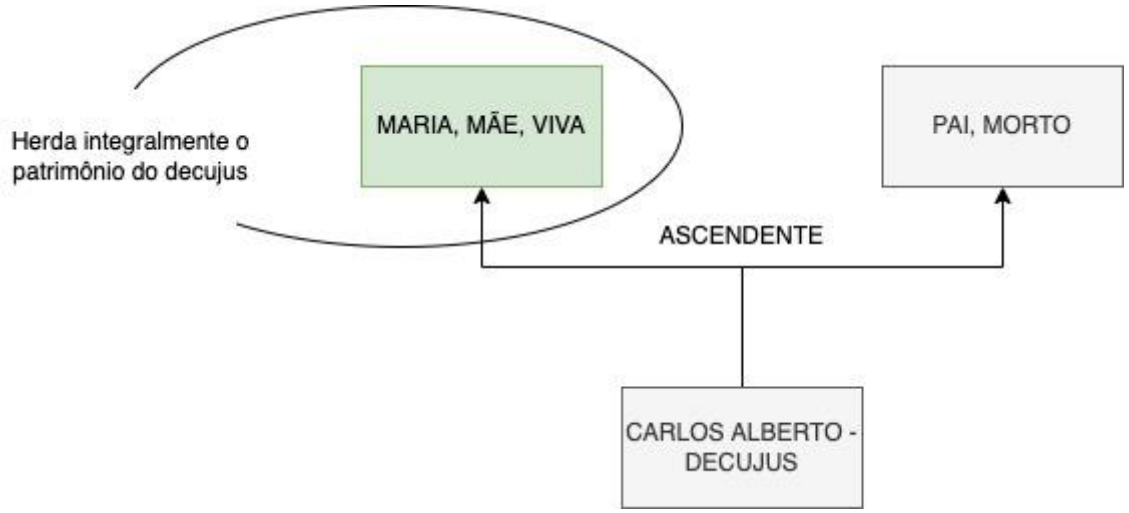
Quadro 3. Caso hipotético 2

Caso hipotético	**Carlos Alberto,** solteiro, faleceu em 15 de agosto de 2010. No momento de seu falecimento Carlo Alberto não tinha filhos, seu pai já era falecido, restando-lhe na linha ascendente apenas sua mãe, Maria, e os avós paternos.
Questão	Quem é o herdeiro de Carlos Alberto?
Resposta	O herdeiro de Carlos Alberto é apenas sua mãe, uma vez que na linha ascendente não há direito de representação. (arts. 1.836 e 1.852, CC).

Fonte: Própria.

Para melhor visualização e entendimento, na Figura 10 é representado em formato de diagrama o caso hipotético 1.

Figura 10 - Diagrama do caso hipotético 2.



Fonte: Própria.

Assim como feito na seção anterior, na Seção 5.3.1 foram formalizados os *fluents* com base na Tabela 1. Em seguida, na Seção 5.3.2 foram aplicadas as regras definidas do Tabela 2 (Axiomas de árvore genealógica) necessárias para saber o grau parentesco entre os elementos. Por fim, na Seção 5.3.3 as condições da vocação hereditária para saber se determinada pessoa pode ser chamada a suceder.

5.3.1 *Fluents* do caso hipotético 2

Nessa fase é alimentada a base de dados definindo os elementos (as pessoas) que participam desse fenômeno sucessório, seja como autor da herança(falecido que deixou bens) ou como potencial herdeiro. Por exemplo, Carlos Alberto, considerando ele ser a pessoa que tem bens a serem sucedidos(autor da herança), sua inserção na base de dados seria disposto como se segue:

```
autor_heranca(carlos_alberto)
pessoa(carlos_alberto)
morto(carlos_alberto)
vivo(maria)
```

5.3.2 Regras da Árvore Genealógica do caso hipotético 2

Nessa fase são definidas as regras para descobrir o parentesco entre os elementos definidos da seção anterior. Por exemplo, para saber se uma pessoa é mãe do autor da herança, basta definir algumas restrições como ser do sexo feminino e ser ascendente de primeiro grau do autor da herança. No formalismo abaixo foi utilizado Maria como exemplo nesse cenário de caso hipotético.

```
poss(mae(maria, carlos_alberto), Z)  :-
      holds([pais(maria, carlos_alberto), feminino(maria)], Z)

state_update(Z1,  mae(maria, carlos_alberto), Z2) :-
      update(Z1, [], [mae(maria, carlos_alberto)], Z2)
```

Na primeira sentença para garantir que Maria seja mãe de Carlos Alberto, expresso pelo axioma *poss,* o axioma de condição *holds* tem que ser verificada como verdade, Maria tem que ter sido declarada como *pais* e do sexo feminino. Assim, o estado é atualizado do Z1 para o Z2.

5.3.3 Regras da Vocação Hereditária do caso hipotético 2

Por fim, as regras da vocação hereditária se utilizarão dos dois módulos anteriores para definir quem é o possível herdeiro da sucessão. No exemplo abaixo, verifica-se se Maria é possível herdeira de Carlos Alberto. Para tanto, é necessário que a pessoa esteja viva e esteja na linha de ascendência do de cujus (autor da herança):

```
poss(herdeiro_ascendente(maria, carlos_alberto), Z) :-
      holds(mae(maria, carlos_alberto), Z), holds(vivo(maria), Z),
      holds(decujus(carlos alberto), Z)

state_update(Z1, herdeiro_ascendente(maria, carlos_alberto), Z2) :-
      update(Z1, [], [herdeiro_ascendente(maria, carlos_alberto)],
Z2)
```

Assim, para verificar se Maria é herdeira de Carlos Alberto, foi utilizado o mesmo axioma `state_update`, onde só será possível a atualização do estado Z1 para o estado Z2 se as condições forem satisfeitas. Essa consulta com o seu retorno pode ser verificada na Figura 11.

Figura 11 - Consulta no SWI-Prolog caso 2.

```
?- state_update(Z1, herdeiro_ascendente(maria, carlos_alberto), Z2).
Z1 = [ascendente(maria, carlos_alberto)|_A],
Z2 = [herdeiro_ascendente(maria, carlos_alberto), ascendente(maria, carlos_alberto)|_A],
```

Fonte: Própria.

5.4 ENTREVISTA SEMIESTRUTURADA

Nessa sessão é descrito como se deu o método principal para alcançar o objetivo dessa pesquisa. Na entrevista semiestruturada, o pesquisador segue um roteiro de perguntas previamente estabelecidas. Porém, não há rigor engessado nesse rito, o que permite combinar perguntas definidas com questões espontâneas, que surgem no decorrer da entrevista. A escolha por esse método de entrevista foi devido a flexibilidade na discussão e formulação de perguntas adicionais, além das questões que haviam sido definidas no roteiro. Para esse trabalho, a entrevista semiestruturada foi realizada com servidores de um órgão que tem jurisdição sobre o direito de sucessão, para assim explorar os fenômenos atuais inseridos no contexto dessa legislação.

5.4.1 Acesso, Unidade de Análise e Dados Coletados

No contato inicial com os servidores do órgão foi apresentado um resumo da pesquisa, contendo os objetivos do estudo e indicação prévia de dados necessários para a investigação. A opção pelos servidores, como unidade de análise e o órgão público, se deu pelo contato completo do ciclo destes com o processo de inventário. Os dados foram coletados valendo-se de entrevistas semiestruturadas. A entrevista se deu de forma remota através da plataforma Google Meet, dado a versatilidade e comodidade que essa ferramenta oferece.

Assim, mesmo utilizando um roteiro, a entrevista pode se tornar mais flexível, com a ordem das perguntas e o teor podendo ser modificadas durante o processo. Esta representa, pois, um dos instrumentos básicos para a coleta de dados, sendo reconhecida por sua importante função na produção do conhecimento científico (MANZINI, 2009).

Alguns pontos foram essenciais para a compreensão de como uma abordagem lógica seria útil ao fenômeno sucessório, tais como: a forma como os dados (ou elementos, *fluents*) chegam ao Poder Judiciário, em qual local ou momento seria mais indicado a automação e quais os pontos críticos ou mais complexos de elucidação nesse ramo do direito. Na entrevista foram coletados dados a respeito da impressão dos dois servidores acerca desses pontos levantados, a coleta de dados seguiu o roteiro que pode ser visualizado no Apêndice B.

As principais questões levantadas aos servidores estão descritas no Quadro 4.

Quadro 4 - Impressões gerais dos servidores acerca dos principais pontos questionados na entrevista.

Questões	Respostas
Como os dados chegam para o Poder Judiciário	Os dados chegam através da petição inicial com todas definições e informações necessárias à abertura da sucessão acompanhada de, quando for o caso, certidão de nascimento, casamento, comprovantes de união estável, parentesco, etc.
Onde seria propício à automação do processo sucessório	Tendo em vista que a petição inicial já chega com elementos necessários para o processo sucessório, a automação seria mais funcional e ideal aos advogados, estes que formulam a petição inicial, e subsidiariamente ao público leigo em geral que normalmente tem dificuldade em distinguir um herdeiro de meeiro, por exemplo. Aos cartórios responsáveis que fazem partilha extrajudicial.
Qual o momento ideal do processo a ser automatizado	Momento da construção da petição inicial, já que ela reúne todos os possíveis herdeiros.
Pontos críticos ou mais complexos de elucidação nesse ramo do direito	A partilha, onde há a divisão de fato dos bens aos herdeiros. Outra grande dificuldade é diferenciar um herdeiro de um meeiro.
Considerações gerais sobre o direito de sucessão e a automação	O processo no direito de sucessão é predominantemente administrativo, o que facilita o processo de automação, já que a tomada de decisão depende mais das regras da lei vigente, o Juiz seria mais um homologador, exceto em casos em que há divergência jurisprudencial ou que a lei ainda não tratou. A automa

Fonte: Própria.

Após essa entrevista foi possível aperfeiçoar o entendimento sobre o fenômeno sucessório estudado no referencial teórico. Observar a visão dos especialistas da área torna possível uma pesquisa mais próxima ao real problema

objeto deste estudo. Foi possível concluir a partir das considerações dos entrevistados que o direito de sucessão apresenta rotinas de tarefas administrativas e repetitivas o que facilita a inserção de formalismo lógico para automatizar tais tarefas. Além disso, esse tipo de sistema pode contribuir com o acesso à informação quando disponibilizados à sociedade tendo em vista a sua capacidade de lidar com diversas regras e exceções do direito de sucessão.

Todo os códigos desenvolvidos e utilizados durante essa pesquisa estão disponíveis em (https://github.com/geovatavares/codigos-da-dissertacao-flux-fluentcalculus-direito-sucessao/tree/main).

6 CONCLUSÕES

De forma inovadora, essa pesquisa relacionou duas grandes áreas de estudo, trabalhando em investigações interdisciplinares entre Direito e Ciência da Computação, combinando-as a fim de buscar um exercício jurisdicional mais autônomo e uma difusão do uso de sistemas inteligentes com o potencial de transformar a prática do Direito, não só por trazer novas questões a serem consideradas pelas profissões jurídicas, mas também pela automação de atividades jurídicas, começando por aquelas que envolvem trabalho repetitivo. O uso de formalismo lógico como o *Fluent Calculus* pode auxiliar nesse processo de axiomatização do domínio jurídico e contribuir para a automação jurídica. Além disso, pôde-se observar durante o estudo de caso que o processo do Direito de Sucessão é predominantemente administrativo, o que facilita a automatização tendo em vistas o ciclo de tarefas repetitivas observando as regras da lei vigente. Assim o Juiz ou órgão julgador seria mais responsável pela homologação dos casos que são pacíficos na lei ou há jurisprudência consolidada. Assim a construção de uma justiça mais célere e eficiente, que facilita o acesso à justiça, passa pelo uso da tecnologia, em especial no uso da Inteligência Artificial.

6.1 CONTRIBUIÇÕES

Entre as contribuições que este trabalho espera ter realizado estão:

- Um avanço nas pesquisas interdisciplinares entre Ciência da Computação e Direito no Brasil.
- Descrição das regras relativas à ordem de vocação observando o formalismo do *Fluent Calculus*.
- Avanço nos estudos de representação de conhecimento jurídico utilizando lógica.

6.2 LIMITAÇÕES

Um dos maiores desafios nessa interdisciplinaridade está em viabilizar a implantação dessa automação. Isso porque esse processo envolve engajar pessoas, redesenhar os próprios procedimentos, elaborar regras, desenvolver um projeto piloto, aprimorá-lo e implantá-lo em escala.

6.3 TRABALHOS FUTUROS

Em futuros trabalhos acadêmicos nessa área interdisciplinar há possibilidades de estudo em diversas abordagens nos mais variados ramos do Direito. Além das contribuições já citadas na seção anterior, há um vasto campo ainda a ser explorado, como:

- Avançar na representação do Direito de Sucessão com o formalismo do Fluent Calculus, em particular, envolvendo questões de especificidade nas consultas;
- Desenvolver aplicações tecnológicas para o direito de sucessão baseadas em lógica;
- Desenvolver sistema capaz de resolver a partilha no Direito de Sucessão retornando a cota parte de cada herdeiro.

6.4 SUBMISSÕES

O presente trabalho foi enviado para o ENIAC 2022 - Encontro Nacional de Inteligência Artificial e Computacional com o título *Abordagem baseada em regras para auxiliar a tomada de decisões em direito sucessório* e encontra-se em processo de submissão.

REFERÊNCIAS

ALVIM, J. **Teoria Geral do Processo**. 18. ed. Rio de Janeiro: Forense, 2015.

BARCELLOS, C. **Prazer, Victor**: Uma breve exposição sobre a utilização de Inteligência Artificial no STF. OABES, 2021. Disponível em: https://www.oabes.org.br/artigos/prazer-victor-uma-breve-exposicao-sobre-a-utilizacao-de-inteligencia-artificial-no-stf-117.html. Acesso em: 05, agosto e 2022.

BEVILAQUA, C. **Direito das successões**. Bahia: Livraria Magalhães, 1899.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Brasília, DF: Senado Federal: Centro Gráfico, 1988.

_____. **Lei 10.406, de 10 de janeiro de 2002 Institui o Código Civil**. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/2002/l10406.htm Acesso em: 21 jul. 2022.

_____. **Lei 8.078, de 11 de setembro de 1990. Dispõe sobre a proteção do consumidor e dá outras providências**. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/l8078compilado.htm.

_____. **Lei 3.071, de 1 de janeiro de 1916.** Dispõe sobre Código Civil dos Estados Unidos do Brasil. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/L3071.htm Acesso em: 06 Jul. 2020.

_____. **Decreto Lei 2.848 de 07 de dezembro de 1940**. Código Penal. Disponível em: http://www.planalto.gov.br/ccivil_03/decreto-lei/del2848compilado.htm. Acesso em: 21 jul. 2022.

_____. **Decreto Lei 1.839 de 31 de dezembro de 1907**. Regula o deferimento da herança no caso da successão . Disponível em:

https://www2.camara.leg.br/legin/fed/decret/1900 1909/decreto 1839 31 dezembro 1907580742republicacao103783 pl.html. Acesso em: 08 jul. 2021.

CAMPOS, D. **Lições de direito de família e das sucessões**. 2. ed. rev. e actual. Belo Horizonte: Del Rey, 1997.

CORMEN, T. et al. **Introduction to algorithms**. 3. ed. Cambridge: The MIT Press, 2009. p. 5-6.

DANTAS, F. **Direito de família e das sucessões**. rev. e atual. por José Gomes Bezerra Câmara e Jair Barros. Rio de Janeiro: Forense, 1991.

DINIZ, M. **Curso de Direito Civil Brasileiro**, v.6: direito das sucessões, 18 ed., São Paulo: SARAIVA, pág. 23, 2004.

_____. **Curso de direito civil brasileiro**: direito de família. 22. ed. São Paulo: Saraiva, 2007.

_____. **Curso de Direito Civil Brasileiro** – Volume 6 – 29ª Ed. Editora Saraiva, 2014.

FEFERBAUM, M; SILVA, A. **Direito e mudanças tecnológicas**: automação, Inteligência Artificial e os novos desafios do ensino jurídico. Revista de Direito e as Novas Tecnologias. Ano1. vol.1. out-dez/2018., Revista dos Tribunais. p.199-216.

GOMES, O. **Sucessões**. 12. ed. atual. por Mário Roberto Carvalho de Faria. Rio de Janeiro: Forense, 2004.

GONÇALVES, C. **Direito Civil Brasileiro**. Volume 7 – Editora Saraiva. 2013.

_____. **Direito civil brasileiro**, Direito das Sucessões, vol. 7 – 14. ed. – São Paulo : Saraiva Educação, 2020.

LEITE, E. **A nova ordem de vocação hereditária e a sucessão dos cônjuges**. In: DELGADO, Mário Luiz; ALVES, Jones Figueiredo (Coords.). Novo Código Civil: questões controvertidas. São Paulo: Método. p. 446. Série Grandes Temas do Direito Privado, v. 1), 2003.

LLEWELLYN, S.; NORTHCOTT, D. **The "singular view" in management case studies qualitative research in organizations and management**. An International Journal, v. 2, n. 3, 2007, p. 194-207.

LOURENÇO, R. T. F.; MAIRINK, C. H. P; ALMEIDA, G. H. **Inteligência artificial e Direito**. LIBERTAS: Rev. Ciênci. Soc. Apl., Belo Horizonte, v. 10, n. 2, p. 126-164, ago./dez. 2020.

LÜDKE, M.; ANDRÉ, M. **Pesquisa em Educação**: abordagens qualitativas. São Paulo: EPU, 1986.

MALUF, A. C. R. F. D. **Novas modalidades de Família na pós-modernidade**. Tese (Doutorado em Direito) – Faculdade de Direito da USP. São Paulo. p. 14. 2010.

MALUF, C. A. D.; MALUF, A. C. D. **Curso de Direito das Sucessões**. São Paulo: Saraiva, 2013.

MANZINI, E. **Análise do uso da entrevista em dissertações e teses em educação especial**. In: BAPTISTA, Cláudio Roberto; JESUS, Denise Meyrelles de (Orgs.). Conhecimento e margens: ação pedagógica e pesquisa em educação especial. Porto Alegre: Mediação/CDV/FACITEC, 2009. p. 167-188.

MANZANO, J.; OLIVEIRA, J. **Algoritmos**: lógica para desenvolvimento de programação. São Paulo: Érica, 2005.

MARANHÃO, J; FLORÊNCIO, J; ALMADA, M. **Inteligência artificial aplicada ao direito e o direito da Inteligência Artificial**. Suprema: revista de estudos constitucionais, Brasília, v. 1, n. 1, p. 154-180, jan./ jun. 2021.

MARCOS, J. **Lógica de Primeira Ordem**. 2022. Disponível em: https://www.dimap.ufrn.br/~jmarcos/courses/LC/Cap3.1.pdf. Acesso em: 26 ago. 2022.

MEDEIROS, N. **Lições de direito civil**. Belo Horizonte: Nova Alvorada, 1997.

McCARTHY, J. **Situations and Actions and Causal Laws**. Stanford Artificial Intelligence Project, Memo 2, Stanford University, CA, 1963.

MONTEIRO, W. Curso de direito civil. 45. ed. São Paulo: Saraiva, v. 6, 2016.

NAKAMITI, E. K. **Agentes inteligentes artificiais**. Dissertação (Mestrado em Comunicação e Semiótica) – Pontifícia Universidade Católica de São Paulo. São Paulo. p. 50. 2009.

NUNES, D; MARQUES, A; RUBINGER, P. **Os perigos do uso da Inteligência Artificial na advocacia**. Conjur, 2018. Disponível em: https://www.conjur.com.br/2018-jul-09/opiniao-perigos-uso-inteligencia-artificial-advocacia. Acesso em: 02 maio de 2022.

OLIVEIRA, A. **Tratado de direito das sucessões**. 4. ed. São Paulo: Max Limonad, v. 1, 1952.

OLIVEIRA, I. J. S. **DIREITO, LÓGICA E INTELIGÊNCIA ARTIFICIAL: por quê, como e em que medida automatizar a solução judicial de conflitos no Brasil**. 2019. 108 f. Tese (Doutorado em Direito) - Centro de Ciências Jurídicas, Universidade Federal de Pernambuco, Recife, 2019.

PRESCOTT, R; MARIANO, R. **Victor, a IA do STF, reduziu o tempo de tarefa de 44 minutos para cinco segundos**. 2019. Disponível em: https://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?UserActiveTemplate=site&info.... Acesso em: 11 mai. 2022.

PEREIRA, S. **Lógica de Predicados**. 2022. Disponível em: https://www.ime.usp.br/~slago/IA-logicaDePredicados.pdf. Acesso em: 25 ago. 2022.

RAMOS, L. d. O.; CUNHA, L. G.; OLIVEIRA, F. L. d.; SAMPAIO, J. O.; BUENO, R. d. L. d. S.; ÚBIDA, G. **Relatório ICJBrasil - 1º semestre / 2017**. [S.l.], 2017. Disponível em: http://bibliotecadigital.fgv.br/dspace/handle/10438/19034. Acesso em: 11 mai. 2022.

REITER, R. **The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression**, Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, 1991.

RODRIGUES, C. M. O. **Uma abordagem ontológica para simulação de ação legal e consistência semântica aplicada à legislação brasileira**. 2019. 222 f. Tese (Doutorado em Ciência da Computação) - Centro de Informática, Universidade Federal de Pernambuco, Recife, 2019.

RUSSELL, J.; NORVIG, P. **Artificial intelligence: a modern approach**. 3. ed. Upper Saddle River: Prentice Hall, 2010. p. 234.

RICH, E. and KNIGHT, K. **Inteligência Artificial**, 2a ed., Makron Books, 1995.

SILVA, E. L; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: UFSC, 2005. Disponível em: <http://tccbiblio.paginas.ufsc.br/files/2010/09/024_Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes1.pdf>. Acesso em: 12 mai. 202.

SILVA, R. **A fórmula "saisine" no Direito Sucessório**. Revista Jus Navigandi, ISSN 1518-4862, Teresina, ano 17, n. 3443, 4 dez. 2012. Disponível em: https://jus.com.br/artigos/23156. Acesso em: 6 mai. 2022.

SILVA, N.; BRAZ, F.; FERRETI, J.; CASTRO, C. **Aplicação da Inteligência Artificial no sistema jurídico brasileiro**: VICTOR e ALEI. Revista de Administración Pública del GLAP, 3(4), páginas 45-55, 2019.

STRAUSS, A.; CORBIN, J. **Pesquisa Qualitativa** - Técnicas e Procedimentos para o Desenvolvimento de Teoria Fundamentada. 2a. ed. Porto Alegre: Artmed, 2008.

POLITIZE. **3 motivos que fazem o judiciário brasileiro ser lento**. 2017. Disponível em: https://www.politize.com.br/judiciario-lento-motivos/#:~. Acesso em: 28 mai. 2022.

TELLES, G. **Direito das Sucessões**. Coimbra: Coimbra Editoras, 1985.

THEODORO, E. **Direito sucessório:** linhas gerais. Revista Jus Navigandi, ISSN 1518-4862, Teresina, ano 21, n. 4616, 20 fev. 2016. Disponível em: https://jus.com.br/artigos/34103. Acesso em: 9 mar. 2022.

THIELSCHER, M. **Reasoning Robots**: the art and science of programming robotic agents. Netherlands: Springer, 2005.

_____. **FLUX**: A Logic Programming Method for Reasoning Agents. Theory And Practice of Logic Programming. 2004.

_____. **The Fluent Calculus**, Dresden University of Technology, 2001.

_____. **Introduction to the fluent calculus**, Electronic Transactions on Artificial Intelligence, 2(3–4): pp. 179–192, 1998.

VENOSA, S. **Direito Civil**: Sucessões. 3. ed. São Paulo: Atlas. 2003.

_____. **Direito das sucessões**. 10 ed. São Paulo: Atlas, 2010.

_____. **Direito Civil, vol 06:** direito das sucessões. 18ª ed. São Paulo: Atlas, 2018.

In: OAB, Conselho Federal. **Exame de Ordem aplicado pela OAB**, 2022. Disponível em: https://examedeordem.oab.org.br/EditaisProvas?NumeroExame=0. Acesso em: 01, jun. 2022.

YIN, R. K. **Estudo de Caso**: Planejamento e Métodos. 5a. ed. Porto Alegre: [s.n.].

APÊNDICE A - AXIOMAS DO DIREITO DE SUCESSÃO EM FLUX

Quadro 1. Fluents do Direito de Sucessão

Nome	**Sintaxe**	**Significado**
Autor da Heranca	**autor_heranca**(X)	Falecido que deixou bens
Pessoa	**pessoa**(X)	Ser humano dotado de capacidade, provido de direitos e obrigações
Morto	**morto**(X)	Pessoa morta
Vivo	**vivo**(X)	Pessoa viva
Masculino	**masculino**(X)	Pessoa do sexo masculino
Feminino	**feminino**(X)	Pessoa do sexo feminino
Cônjuge	**conjunge**(X, Y)	Pessoa unida por matrimônio
Regime Comunhão Universal	**comunhao_universal**(X, Y)	Duas pessoas casadas em regime de comunhão universal de bens
Regime Comunhão Parcial	**comunhao_parcial**(X, Y)	Duas pessoas casadas em regime de comunhão parcial de bens
Regime Comunhão Separação Convencional	**comunhao_separacao_convencional**(X, Y)	Duas pessoas casadas em regime de separação convencional de bens
Regime Comunhão Separação Legal	**comunhao_separacao_legal**(X, Y)	Duas pessoas casadas em regime de separação legal de bens

Quadro 2. Axiomas de Árvore Genealógica

Nome	**Sintaxe**	**Significado**
Pais	`pais(X, Y).`	Ascendente de primeiro grau de uma pessoa

Pai	`poss( pai(X,Y), Z) :-`
	`holds([masculino(X), pais(X,Y)], Z).`
	
	`state_update(Z1, pai(X, Y), Z2) :-`
	`update(Z1, [], [pai(X, Y)], Z2).`
		Pessoa progenitor

Mãe	`poss(mae(X,Y), Z) :-`
	`holds([feminino(X), pais(X,Y)], Z).`
	
	`state_update(Z1, mae(X, Y), Z2) :-`
	`update(Z1, [], [mae(X, Y)], Z2).`
		Pessoa progenitora

Filha	`poss(filha(X,Y), Z) :-`
	`holds([descendente(X,Y), feminino(X)], Z).`
	
	`state_update(Z1, filha(X, Y), Z2) :-`
	`update(Z1, [], [filha(X, Y)], Z2).`
		Pessoa do sexo feminino descedente de primeiro grau

Filho	`poss(filho(X,Y), Z) :-`
	`holds([descendente(X,Y), masculino(X)], Z).`
	
	`state_update(Z1, filho(X, Y), Z2) :-`
	`update(Z1, [], [filho(X, Y)], Z2).`
		Pessoa do sexo masculino descedente de primeiro grau

Irmãos	`poss(irmaos(X,Y), Z) :-`
	`holds([pai(W,X), pai(W,Y), X\=Y], Z).`
	
	`state_update(Z1, filho(X, Y), Z2) :-`
	`update(Z1, [], [filho(X, Y)], Z2).`
		Pessoas que possuem o mesmo ascendente

Ascendente	`poss(ascendente(X, Y), Z) :-`
	`holds(pais(X, Y), Z).`
	
	`state_update(Z1, ascendente(X, Y), Z2) :-`
	`update(Z1, [], [ascendente(X, Y)], Z2)`
		antepassado

Ascendente	`poss(ascendente(X, Y), Z) :-` `holds(pais(X, W), ascendente(W, Y), Z).`	verificar antepassados de forma recursiva

	`state_update(Z1, ascendente(X, Y), Z2) :-`
	`  update(Z1, [], [ascendente(X, Y)],`
	`  Z2)`	

Descendente	`poss(descendente(X, Y), Z) :-`
	`  holds(ascendente(Y, X), Z).`
	
	`state_update(Z1, descendente(X, Y), Z2) :-`
	`  update(Z1, [], [descendente(X, Y)],`
	`  Z2)`
		sucessores

Quadro 3. Axiomas de pré condição e atualização da vocação hereditária

Nome	**Sintaxe**	**Significado**

De cujus	`poss(decujus(X),Z) :-`
	`  holds(morto(X), Z),`
	`holds(autor_heranca(X), Z).`
	`state_update(Z1, decujus(X), Z2) :-`
	`  update(Z1, [], [decujus(X)], Z2).`
		falecido cujos bens estão em inventário/dar início a sucessão

Herdeiro Descendente	`poss(herdeiro_descendente(X), Z) :-`
	`  holds(descendente(X), Z),`
	`holds(vivo(X), Z),`
	`  holds(decujus(Y), Z)`
	
	`state_update(Z1, herdeiro_descendente(X), Z2) :-`
	`  update(Z1, [], [herdeiro_descendente(X)], Z2)`
		Pessoa viva herdeira na linha de descendência do de cujus

Herdeiro Ascendente	`poss(herdeiro_ascedente(X), Z) :-`
	`  holds(ascendente(X), Z),`
	`holds(vivo(X), Z),`
	`  holds(decujus(Y), Z)`
	`  not_holds(herdeiro_descendente(_), Z)`
		Pessoa viva herdeira na linha de ascendência do de cujus, desde que não exista herdeiro descendente

	state_update(Z1, herdeiro_ascendente(X), Z2) :- **update**(Z1, [], [herdeiro_ascendente(X)], Z2)	
Herdeiro Pré-morto	**poss**(herdeiro_pre_morto(X), Z) :- **holds**(descendente(X), Z), **holds**(morto(X), Z), **holds**(decujus(Y), Z) **state_update**(Z1, herdeiro_pre_morto(X), Z2) :- **update**(Z1, [], [herdeiro_pre_morto(X)], Z2)	Descendente que morreu antes do decujus
Herdeiro Descendente Estirpe	**poss**(herdeiro_estirpe(X,Y), Z) :- **holds**(descendente(X), Z), **holds**(herdeiro_pre_morto(y), Z), **holds**(vivo(X), Z), **holds**(decujus(Y), Z) **state_update**(Z1, herdeiro_estirpe(X), Z2) :- **update**(Z1, [], [herdeiro_estirpe(X)], Z2)	Herdam por estirpe a parte que cabia ao herdeiro pré-morto
Meeiro	**poss**(meeiro(X, Y), Z) :- **or_holds**([meeiro_universal(X, Y), meeiro_parcial(X, Y), meeiro_separacao_legal(X, Y)], Z) **state_update**(Z1, meeiro(X, Y), Z2) :- **update**(Z1, [], [meeiro(X, Y)], Z2)	Cônjuge que recebe a título de meação Retorna as possíveis formas de meação
Meeiro Universal	**poss**(meeiro_universal(X, Y), Z) :- **holds**(conjuge(X, Y), Z), **holds**(vivo(X), Z), **holds**(comunhao_universal(X, Y), Z), **holds(**decujus(Y), Z**)**, **state_update**(Z1, meeiro_universal(X, Y), Z2) :- **update**(Z1, [], [meeiro_universal(X, Y)], Z2)	Cônjuge que recebe a título de meação proveniente de um regime de comunhão universal com o decujus
Meeiro Parcial	**poss**(meeiro_parcial(X, Y), Z) :- **holds**(conjuge(X, Y), Z), **holds**(vivo(X), Z),	Cônjuge que recebe a título de meação proveniente de um regime de comunhão

	holds(comunhao_parcial(X, Y), Z), holds(decujus(Y), Z), state_update(Z1, meeiro_parcial(X, Y), Z2) :- update(Z1, [], [meeiro_parcial(X, Y)], Z2)	parcial com o decujus
Meeiro Separação Legal	poss(meeiro_separacao_legal(X, Y), Z) :- holds(conjuge(X, Y), Z), holds(vivo(X), Z), holds(comunhao_separacao_legal(X, Y), Z), holds(decujus(Y), Z), state_update(Z1, meeiro_separacao_legal(X), Z2) :- update(Z1, [], [meeiro_separacao_legal(X)], Z2)	Cônjuge que recebe a título de meação proveniente de um regime de comunhão de separação legal com o decujus e meeiro(a) dos bens adquiridos após o casamento, desde que esses bens tenham sido adquiridos de forma onerosa por ambos e NÃO herda em concorrência com os demais herdeiros nos bens adquiridos pelo decujus antes do casamento. Fonte: Súmula 377 do STF e art. 1829 e 1640 do CC.
herdeiro Cônjuge Separação Convencional	poss(herd_conjuge_separacao_convencional(X, Y), Z) :- holds(conjuge(X, Y), Z), holds(vivo(X), Z), holds(comunhao_separacao_convencional(X, Y), Z), holds(decujus(Y), Z), state_update(Z1, herd_conjuge_separacao_convencional(X), Z2) :- update(Z1, [], [herd_conjuge_separacao_convencional(X)], Z2)	Se houver descendentes o cônjuge concorrerá com estes em iguais condições, se houver ascendentes mesma regra aplicada, se não houver nem descendentes nem ascendentes, o cônjuge herdará na integralidade os bens 1.837 cc
herdeiro cônjuge comunhão parcial	poss(herd_conjuge_parcial(X, Y), Z) :- holds(conjuge(X, Y), Z), holds(vivo(X), Z), holds(comunhao_parcial(X, Y), Z), holds(decujus(Y), Z).	O Cônjuge herdeiro concorrerá com os demais herdeiros descendentes em iguais condições nos bens particulares do de cujus, se houver ascendentes, mesma regra aplicada, se não houver nem

	state_update(Z1, herd_conjuge_parcial(X), Z2) :- **update**(Z1, [], [herd_conjuge_parcial(X)], Z2)	descendentes nem ascendentes, o cônjuge herdará na integralidade os bens. 1.837 cc
Herdeiro Colateral	**poss**(herdeiro_colateral(X, Y), Z) :- **not_holds**([herdeiro_descendente(_,Y), herdeiro_ascedente(_,Y), herdeiro_conjuge(_,Y), meeiro(_,Y)], Z), **holds**(vivo(X), Z), **holds**(parente_ate_4_grau(X, Y), Z), **holds(**decujus(Y), Z**).** **state_update**(Z1, herd_conjuge_parcial(X), Z2) :- **update**(Z1, [], [herd_conjuge_parcial(X)], Z2)	Não havendo cônjuge, descendentes ou ascendentes, são herdeiros os parentes colaterais, (os de até 4º grau: pela ordem, irmãos, sobrinhos, tios e primos). Os mais próximos excluem os remotos, exceto os sobrinhos, que têm o direito de representar os irmãos do falecido.
Herdeiros	**poss**(herdeiros(X, Y), Z) :- **or_holds**([herdeiro_descendente(X,Y), herdeiro_conjuge_comunhao_parcial(X,Y), herdeiro_conjuge_separacao_convencional(X, Y), herdeiro_descendente_estirpe(X,Y), herdeiro_ascendente(X,Y), herdeiro_descendente_estirpe(X,Y), herdeiro_colateral(X,Y)], Z). **state_update**(Z1, herdeiros(X), Z2) :- **update**(Z1, [], [herdeiros(X)], Z2)	Retorna todos os herdeiros existentes
Vocação Hereditária	**poss**(vocacao_hereditaria(X, Y), Z) :- **or_holds**([meeiro(X,Y), herdeiros(X,Y)], Z). **state_update**(Z1, vocacao_hereditaria(X), Z2) :- **update**(Z1, [], [vocacao_hereditaria(X)], Z2)	Retorna todos os herdeiros e meeiros existentes

APÊNDICE B - ROTEIRO DA ENTREVISTA

Este apêndice contém o roteiro de entrevistas semiestruturado que foi realizado para o estudo de caso, cujo objetivo era entender, na visão dos especialistas, como a automação seria útil ao fenômeno sucessório.

Questões	Respostas
Como os dados chegam para o Poder Judiciário	Os dados chegam através da petição inicial com todas definições e informações necessárias à abertura da sucessão acompanhada de, quando for o caso, certidão de nascimento, casamento, comprovantes de união estável, parentesco, etc.
Onde seria propício à automação do processo sucessório	Tendo em vista que a petição inicial já chega com elementos necessários para o processo sucessório, a automação seria mais funcional e ideal aos advogados, estes que formulam a petição inicial, e subsidiariamente ao público leigo em geral que normalmente tem dificuldade em distinguir um herdeiro de meeiro, por exemplo. Aos cartórios responsáveis que fazem partilha extrajudicial.
Qual o momento ideal do processo a ser automatizado	Momento da construção da construção da petição inicial, já que ela reúne todos os possíveis herdeiros.
Pontos críticos ou mais complexos de elucidação nesse ramo do direito	A partilha, onde há a divisão de fato dos bens aos herdeiros. Outra grande dificuldade é diferenciar um herdeiro de um meeiro.
Considerações gerais sobre o direito de sucessão e a automação	O processo no direito de sucessão é predominantemente administrativo, o que facilita o processo de automação, já que a tomada de decisão depende mais das regras da lei vigente, o Juiz seria mais um homologador, exceto em casos em que há divergência jurisprudencial ou que a lei ainda não tratou. A automa

APÊNDICE C - KERNEL COMPLETO DO FLUX

O Kernel e outros arquivos referente a essa linguagem estão disponibilizados em https://github.com/logicmoo/flux/blob/master/prolog/wumpus_2_3.pl.

```
holds(F, [F|_]).
holds(F, Z) :- nonvar(Z), Z=[F1|Z1], F\==F1, holds(F, Z1).

holds(F, [F|Z], Z).
holds(F, Z, [F1|Zp]) :- nonvar(Z), Z=[F1|Z1], F\==F1, holds(F, Z1, Zp).

cancel(F,Z1,Z2) :-
   var(Z1)    -> cancel(F,Z1), cancelled(F,Z1), Z2=Z1 ;
   Z1 = [G|Z] -> ( F\=G -> cancel(F,Z,Z3), Z2=[G|Z3]
                         ; cancel(F,Z,Z2) ) ;
   Z1 = []    -> Z2 = [].

minus_(Z, [], Z).
minus_(Z, [F|Fs], Zp) :-
   ( \+ not_holds(F, Z) -> holds(F, Z, Z1) ;
     \+ holds(F, Z)     -> Z1 = Z
                         ; cancel(F, Z, Z1), not_holds(F, Z1) ),
   minus_(Z1, Fs, Zp).

plus_(Z, [], Z).
plus_(Z, [F|Fs], Zp) :-
   ( \+ holds(F, Z)     -> Z1=[F|Z] ;
     \+ not_holds(F, Z) -> Z1=Z
                         ; cancel(F, Z, Z2), not_holds(F, Z2), Z1=[F|Z2] ),
   plus_(Z1, Fs, Zp).

update(Z1, ThetaP, ThetaN, Z2) :-
   minus_(Z1, ThetaN, Z), plus_(Z, ThetaP, Z2).

knows(F, Z) :- \+ not_holds(F, Z).

knows_not(F, Z) :- \+ holds(F, Z).

knows_val(X, F, Z) :- k_holds(F, Z), knows_val(X).

k_holds(F, Z) :- nonvar(Z), Z=[F1|Z1],
                 ( instance(F1, F), F=F1 ; k_holds(F, Z1) ).

:-local variable(known_val).
:-setval(known_val,[]).

knows_val(X) :- dom(X), \+ nonground(X), ambiguous(X) -> false.
knows_val(X) :- getval(known_val,X), X \== [], setval(known_val, []).

dom([]).
```

```
dom([X|Xs]) :- dom(Xs), ( is_domain(X) -> indomain(X)
                                        ; true ).

ambiguous(X) :-
   ( getval(known_val, Val),
     Val \== [] -> setval(known_val, [])
   ;
                 setval(known_val, X),
                 false
   ).

execute(A,Z1,Z2) :-
   is_predicate(perform/2),
   perform(A,Y)    -> ( is_predicate(ab_state_update/4) ->
                           ( Z1=[sit(S)|Z], ! ; S=[], Z=Z1 ),
                           ( state_update(Z,A,Z3,Y)
                             ; ab_res([[A,Y]|S],Z3) ),
                           !, Z2=[sit([[A,Y]|S])|Z3]
                       ;
                  state_update(Z1,A,Z2,Y) ) ;

   is_predicate(perform/3),
   perform(A,Y,E)  -> ( is_predicate(ab_state_update/4) ->
                           ( Z1=[sit(S)|Z], ! ; S=[], Z=Z1 ),
                           ( state_update(Z,A,Z3,Y), state_updates(Z3,E,Z4)
                             ; ab_res([[A,Y,E]|S],Z4) ),
                           !, Z2=[sit([[A,Y,E]|S])|Z4]
                       ;
                  state_update(Z1,A,Z,Y), state_updates(Z,E,Z2) ) ;

   A = [A1|A2]     ->
                      execute(A1,Z1,Z), execute(A2,Z,Z2) ;

   A = if(F,A1,A2) ->
                      (holds(F,Z1) -> execute(A1,Z1,Z2)
                                    ; execute(A2,Z1,Z2)) ;

   A = []          ->
                      Z1=Z2 ;

   complex_action(A,Z1,Z2).

ab_res([],Z) :- init(Z).
ab_res([S1|S],Z) :-
   ab_res(S,Z1),
   ( S1=[A,Y] -> ( state_update(Z1,A,Z,Y) ; ab_state_update(Z1,A,Z,Y) )
     ;
     S1=[A,Y,E], ( state_update(Z1,A,Z2,Y) ; ab_state_update(Z1,A,Z2,Y) ),
                 state_updates(Z2, E, Z) ).

state_updates(Z, [], Z).
state_updates(Z1, [A|S], Z2) :-
```

```
   state_update(Z1, A, Z), state_updates(Z, S, Z2).

:- local variable(plan_search_best).

plan(Problem, Z, P) :-
   setval(plan_search_best(_ : -1)),
   plan_search(Problem, Z),
   getval(plan_search_best,P:C),
   C =\= -1.

plan_search(Problem, Z) :-
    is_predicate(Problem/2) ->
       ( PlanningProblem =.. [Problem,Z,P],
         call(PlanningProblem),
         plan_cost(Problem, P, C),
         getval(plan_search_best,_:C1),
         ( C1 =< C, C1 =\= -1 -> false
                               ;
                               setval(plan_search_best,P:C), false )
         ;
         true ) ;
    PlanningProblem =.. [Problem,Z,P,Zn],
    call(PlanningProblem),
    plan_cost(Problem, P, Zn, C),
    getval(plan_search_best,_:C1),
    ( C1 =< C, C1 =\= -1 -> false
                             ;
                           setval(plan_search_best,P:C),
                           false
    )
    ;
    true.

knows(F, S, Z0) :- \+ ( res(S, Z0, Z), not_holds(F, Z) ).

knows_not(F, S, Z0) :- \+ ( res(S, Z0, Z), holds(F, Z) ).

:-local variable(known_vals).

knows_val(X, F, S, Z0) :-
   setval(known_vals,[]),
   res(S, Z0, Z),
   findall(X, knows_val(X,F,Z), T),
   getval(known_vals,T1),
   ( T1=[] -> T2=T ; intersection(T,T1,T2) ),
   setval(known_vals, T2),
   false.
knows_val(X, _, _, _) :-
   getval(known_vals, T),
   member(X, T),
   setval(known_vals, []).
```

```
res([], Z0, Z0).
res(do(A,S), Z0, Z) :-
   ( A = if_true(F)  -> res(S, Z0, Z), holds(F, Z)
   ;
                        ( A = if_false(F) -> res(S, Z0, Z),
                                             not_holds(F, Z)
                        ;
                                             res(S, Z0, Z1),
                                             state_update(Z1, A, Z, _)
                        )
   ).

causes(Z,P,N,Z2) :-
   causes(Z,P,N,Z1,P1,N1) -> causes(Z1,P1,N1,Z2)
                           ; Z2=Z.

ramify(Z1,ThetaP,ThetaN,Z2) :-
   update(Z1,ThetaP,ThetaN,Z), causes(Z,ThetaP,ThetaN,Z2).
```

ANEXO A - LÓGICA DE PRIMEIRA ORDEM

Em (MARCOS, 2015) é possível verificar que a linguagem da lógica de primeira ordem é mais expressiva que a linguagem da lógica proposicional. Por exemplo, no âmbito da lógica proposicional, as fórmulas são mais simples e os símbolos proposicionais não têm estrutura interna. Assim, cada uma das proposições ou asserções representadas por estas fórmulas é representada por meio de um símbolo, normalmente, uma letra. Há outras fórmulas que representam asserções mais complexas que se podem obter a partir destas usando apenas os conectivos lógicos como conjunção, disjunção ou implicação.

Esta situação não permite a representação da estrutura interna das proposições, nomeadamente, a representação explícita de que certas entidades têm determinadas propriedades ou estão numa determinada relação com outras entidades. Torna-se assim complicado representar asserções do tipo “existem números inteiros pares” ou “o quadrado de qualquer número inteiro não nulo e um número inteiro positivo” e tirar partido dessa representação para desenvolver raciocínios do tipo “se o quadrado de qualquer número inteiro não nulo é positivo” e “-3 é um número inteiro não nulo” então “o quadrado de -3 é um número inteiro positivo”.

Já na linguagem da lógica de primeira ordem é possível fazer referência explícita a entidades, representar funções sobre essas entidades (por exemplo, a função quadrado de) e propriedades ou predicados sobre essas entidades (por exemplo, o predicado ser número inteiro, ser número positivo ou ser número par). Permite também a quantificação sobre entidades, ou seja, permite representar asserções que exprimem a ideia de que todas as entidades possuem uma determinada propriedade ou que existem entidades que possuem uma outra propriedade.

Para definir a sintaxe de uma linguagem de primeira ordem é necessário dispor de um alfabeto. Este alfabeto introduz os símbolos que serão construídos os termos e as fórmulas dessa linguagem. Um alfabeto de primeira ordem é composto de:

- um conjunto SF de símbolos de uma função e um conjunto SP de símbolos de predicado, disjuntos, juntamente com as respectivas aridades, uma função $\tau : SF \cup SP \rightarrow IN_0$;

- um conjunto numerável X de variáveis;
- conectivos ¬ (negação), ∧ (conjunção), ∨ (disjunção) e → (implicação); quantificador universal ∀ e o quantificador existencial ∃ ;
- símbolos de pontuação (e) (parêntese esquerdo e parêntese direito) e , (vírgula).

Numa assinatura de primeira ordem são introduzidos os símbolos que representam as funções (de diferentes aridades) e os símbolos que representam os predicados (de diferentes aridades). Assim uma assinatura de primeira ordem é um par:

$$\Sigma = (SF, SP)$$

onde

$SF = \{SF\text{i}\}\text{i} \in IN_0$;

$SP = \{SP\text{i}\}\text{i} \in IN_0$;

são famílias de conjuntos indexados em IN_0 tais que os conjuntos SFi , SPi , i $\in IN_0$, são disjuntos dois a dois. Para cada i $\in IN_0$, SFi é o conjunto de símbolos de função de aridade i e SPi é o conjunto de símbolos de predicado de aridade i. Os símbolos de função de aridade 0 são também designados símbolos de constante ou constante. Os símbolos de predicados de aridade 0 são também designados símbolos proposicionais.