JEAN DE OLIVEIRA ZAHN

UM MECANISMO PARA VERIFICAÇÃO DE CONFLITOS NORMATIVOS INDIRETOS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Orientador: Prof.ª. Dr.ª VIVIANE TORRES DA SILVA

Niterói

2015

Ficha Catalográfica – Esta página deve ser removida na versão a ser entregue para a banca, mas deve ser reinserida na versão final, com a ficha catalográfica fornecida pela biblioteca. Informações sobre este processo devem ser obtidas na secretaria da pós-graduação.

JEAN DE OLIVEIRA ZAHN

UM MECANISMO PARA VERIFICAÇÃO DE CONFLITOS NORMATIVOS INDIRETOS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Aprovada em março de 2015.

BANCA EXAMINADORA

_____

Prof.ª Dr.ª Viviane Torres da Silva – Orientador
UFF – Universidade Federal Fluminense

_____

Prof. Dr. Christiano de Oliveira Braga
UFF – Universidade Federal Fluminense

_____

Prof. Dr. Sean Wolfgand Matsui Siqueira
UNIRIO – Universidade Federal do Estado do Rio de Janeiro

Niterói
2015

Aos meus familiares,

Joanízio Zahn (in memoriam)
Lúcia de Oliveira Zahn
Giselly Zahn Erthal
Aline Zahn dos Santos
Joanízio Zahn Junior
Henilson Erthal
Tiago dos Santos
Heitor Zahn Erthal
Lívia Zahn dos Santos

AGRADECIMENTOS

Entre tantos agradecimentos, o maior deles é o que tenho feito a Deus. Em diversos momentos, pensamos em desistir, mas devido a toda sabedoria e proteção divina, os resultados puderam ser alcançados. Nada me faltou durante todo esse período de pesquisa e desenvolvimento desse trabalho e, com toda a certeza, Deus esteve sempre à frente de tudo isso.

Agradeço minha mãe, Sra. *Lúcia Zahn*, por todo o apoio a mim demonstrado, durante todo o período em que estive envolvido com essa pesquisa. Sem esse apoio todo especial de mãe, jamais teria conseguido alcançar os objetivos da forma como alcancei. Agradeço meus irmãos, *Giselly Zahn*, *Aline Zahn* e *Joanízio Zahn Jr*., que sempre estiveram comigo em todos os momentos difíceis, demonstrando apoio e aconselhando nas decisões importantes. Vocês fazem parte de tudo isso. Agradeço o Prof. *Henilson Erthal* e o Pr. *Tiago dos Santos*, que sempre foram verdadeiros conselheiros em todos os momentos que precisei. Um agradecimento todo especial aos meus sobrinhos, *Heitor Zahn Erthal* e *Lívia Zahn dos Santos*, por serem verdadeiras fontes inspiradoras, dando força e motivação para a continuidade nos estudos. Agradeço os familiares em geral, pois são e sempre serão fundamentais em toda e qualquer conquista.

Não poderia deixar de agradecer a Prof.ª Dr.ª *Viviane Torres da Silva*. Sua atenção e capacidade de motivar seus alunos são uma marca, na qual levarei comigo por toda a minha vida acadêmica. Sempre muito prestativa, desenvolver este trabalho juntamente contigo foi uma honra, um verdadeiro estágio na qual eu muito aprendi. Agradeço aos professores *Leonardo Murta* e *Vanessa Braganholo*, por terem sido verdadeiros exemplos de profissionais na docência. Não posso me esquecer do Prof. *Elias Januário*, *Fernando Selleri* e *Rodrigo Fernando Shimazu*, que contribuíram me motivando a seguir carreira acadêmica. Sem dúvida, vocês foram profissionais exemplares e que muito me influenciaram.

Agradeço ao Instituto de Computação da Universidade Federal Fluminense (IC/UFF) pela oportunidade de fazer parte do corpo discente dessa importante instituição. Até aqui eu muito aprendi e compreendi. Que esse seja apenas o início de toda uma jornada juntos.

Agradeço aos meus amigos, aqueles que de uma forma indireta, contribuíram para que eu tivesse a capacidade de desenvolver o presente trabalho. Amigos são verdadeiros pilares em nossas vidas. A todos que contribuíram, deixo aqui o meu muito obrigado.

“A moralidade é a melhor de todas as regras para orientar a humanidade.”

Friedrich Nietzsche

RESUMO

Em Sistemas Multi-Agentes, normas tem sido o meio utilizado para reger o comportamento de agentes autônomos, heterogêneos e que são projetados para diferentes fins. As normas definem o que pode ser feito, o que deve ser feito e o que não pode ser feito pelos agentes do sistema. Uma das principais questões que envolve a validação de normas é a detecção de conflitos normativos. Duas normas estão em conflito quando o cumprimento de uma norma viola a outra, e vice-versa. Diversos trabalhos tem sido propostos para lidar com conflitos normativos, porém a maioria tem focado seus esforços em conflitos normativos diretos, que ocorrem quando as normas em conflito são aplicadas sobre uma mesma entidade, regendo um mesmo comportamento. Existem conflitos, chamados neste trabalhos de conflitos indiretos, que ocorrem entre normas aplicadas a entidades diferentes, porém relacionadas, regendo comportamento diferentes, porém relacionados. Neste contexto, propomos nesta dissertação uma abordagem de verificação de conflitos normativos indiretos que analisa as características do domínio da aplicação no momento da verificação de conflitos. Para isso, desenvolvemos uma estrutura de ontologia para a representação do domínio focando nas normas que governam o sistema e nos relacionamentos entre os elementos identificados nas normas como, por exemplo, relacionamentos entre as entidades e os comportamentos. Nós definimos um modelo normativo, apresentamos um conjunto de relacionamentos que podem ocorrer entre os elementos definidos nas normas. É também resultado deste trabalho o *Conflict Checker*, um mecanismo de verificação de conflitos normativos diretos e indiretos que recebe como entrada uma ontologia descrevendo as normas e os relacionamentos. Sua saída é um conjunto de pares de normas em conflito com uma justificativa para a ocorrência de cada conflito detectado. Para avaliar a capacidade do *Conflict Checker* em detectar conflitos normativos, utilizamos um cenário de compra e venda de ações, extraindo um conjunto de normas de contrato nesse seguimento e inserindo novas normas para que o problema de conflito normativo pudesse ser alcançado.

Palavras-chave: Sistemas Multi-Agentes, Sistemas Normativos, Normas, Conflitos Normativos, Ontologia, Relacionamento.

ABSTRACT

In Multi-Agent Systems, norms have been used to regulate the behaviour of autonomous, heterogeneous and independently designed agents. Norms describe what can do, must do and cannot do. One of the main issues on the specification of norms is the detection of normative conflicts. Two norms are in Conflict when the fulfilment of one norm violates the other and vice-versa. Although several Works have been proposed to deal with normative conflicts, the majority focuses on direct normative conflicts that occur when the norms apply to the same entity and govern the same behavior. There is conflicts, called in this paper of indirect conflict, which occur between norms applied to different entities, but related, regulating different behaviors, but related. In this context, we proposes in this paper an approach to check indirect normative conflicts that analyses the application domain characteristics at the time of verification. In order to do so, we developed an ontology model for the representation of a domain focusing on the norms regulating the system and relationships between the elements identified in these norms, i.e., relationships between entities and behaviors. We define a normative model, present a set of relationships that can occur between the elements defined in the norms and developed the *Conflict Checker*. The *Conflict Checker* is a verification mechanism of direct and indirect normative conflicts that takes as input an ontology describing the rules and relationships, generating a set of normative peers in conflict and one justification for the occurrence of each detected conflict. To evaluate the ability of *Conflict Checker* on the checking normative conflicts, we use the buy-sell agreement scenario buy extracting a set of contract rules and inserting new norms for the normative conflicts problem could be achieved.

Keywords: Multi-Agent System, Normative Systems, Norms, Normative Conflicts, Ontology, Relationship.

LISTA DE ILUSTRAÇÕES

Figura 1: Modelo de agente apresentado por WOOLDRIDGE (2009). 22
Figura 2: Uma pequena ontologia, seus indivíduos e propriedades. 33
Figura 3: Representação da Classe *Agent*. 34
Figura 4: Representação do relacionamento Maria é Estudante. 35
Figura 5: Diagrama UML apresentando o gerenciamento de ontologias em OWL API, conforme HORRIDGE e BECHHOFER (2011). 37
Figura 6: O ciclo dos estados e ações. 42
Figura 7: Classes base da Ontologia. 44
Figura 8: Classes base da Ontologia com referência normativa. 44
Figura 9: As classes *Context*, *Organization* e *Environment*. 45
Figura 10: Instâncias da classe *Context* e seus relacionamentos. 45
Figura 11: A classes *Entity*, *Organization*, *Role* e *Agent*. 46
Figura 12: Instâncias da classe *Entity* e seus relacionamentos. 47
Figura 13: A classe *Behavior*, *Action* e *State*. 47
Figura 14: As instâncias da classe *Behavior* e seus relacionamentos. 48
Figura 15: A classe *Norm* e suas subclasses. 48
Figura 16: NormaN1 aplicada sobre o domínio. 49
Figura 17: Exemplo de relacionamento *Hierarchy* entre Contextos. 51
Figura 18: Exemplo do relacionamento *Inhabit* entre Contextos. 52
Figura 19: Exemplo de relacionamento *Play*. 53
Figura 20: Exemplo do relacionamento *Hierarchy* entre entidades. 54
Figura 21: Exemplo de relacionamento *Ownership*. 55
Figura 22: A ação Mover e o relacionamento de *Refinement*. 57
Figura 23: *Dependecy* definido entre ações e estados. 59
Figura 24: *Composition* definido entre estados. 61
Figura 25: *Orthogonality* definido entre estados. 65
Figura 26: O *Conflict Checker*. 69
Figura 27: Entrada de informações no *Conflict Checker*. 70
Figura 28: Informações sobre a ontologia. 70
Figura 29: O resultado das verificações. 71
Figura 30: Diagrama de Classes. 75
Figura 31: Diagrama de Sequência para a extração de normas. 79

Figura 32: Código-fonte para extração de ações de um documento OWL. 80
Figura 33: Código-fonte para extração de relacionamentos entre ações. 81
Figura 34: Código-fonte para extração de normas. 81
Figura 35: Código-fonte para extração da entidade definida na norma. 81
Figura 36: Código-fonte para extração do Id da norma. 81
Figura 37: Diagrama de Sequência da aplicação das regras. 82
Figura 38: Código-fonte do método *checkTime()*. 83
Figura 39: Relacionamento *Orthogonality* entre papéis. 84
Figura 40: Diagrama de Classe contento a classe *OrthogonalRole*. 85
Figura 41: Código-fonte para extrair o relacionamento *Orthogonal* entre papéis. 85
Figura 42: Contextos e Entidades do contrato de compra e venda de ações. 88
Figura 43: Relacionamento *Refinement* no contrato. 89
Figura 44: Relacionamento *Dependency* no contrato. 90
Figura 45: Relacionamento *Composition* no contrato. 91
Figura 46: Relacionamento *Orthogonality* no contrato. 92
Figura 47: Um exemplo de norma do contrato, a NormN6. 93
Figura 48: Um exemplo de norma inserida no domínio, a NormN10. 95
Figura 49: O resultado das verificações do *Conflict Checker*. 96

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

BDI – *Belief, Desire and Intention*

EUA – *Estados Unidos da América*

IA – *Inteligência Artificial*

IC – *Instituto de Computação*

IDE – *Integrated Development Environment*

IRI – *Internationalized Resource Identifier*

NoA – *Normative Agent Architecture*

NY – *Nova York*

SMA – *Sistemas Multi-Agentes*

OV – *Organizações Virtuais*

OWL – *Web Ontology Language*

POLAR – *Policy Language for Agent Reasoning*

RJ – *Rio de Janeiro*

UFF – *Universidade Federal Fluminense*

UML – *Unified Modeling Language*

W3C – *World Wide Web Consortium*

SUMÁRIO

Capítulo 1 – INTRODUÇÃO 15

1.1 PROBLEMA 15

1.2 SOLUÇÃO 18

1.3 OBJETIVOS 19

1.4 METODOLOGIA 19

1.5 ORGANIZAÇÃO DO TEXTO 20

Capítulo 2 – REFERENCIAL TEÓRICO 22

2.1 SISTEMAS MULTI-AGENTES 22

2.2 NORMAS E SISTEMAS NORMATIVOS 24

2.2.1 MODELOS DE NORMAS 25

2.3 CONFLITOS ENTRE NORMAS 29

2.3.1 CONFLITO VS. INCONSISTÊNCIA 29

2.3.2 CONFLITOS DIRETOS 29

2.3.3 CONFLITOS INDIRETOS 30

2.4 ONTOLOGIA 31

2.4.1 PROTÉGÉ 34

2.4.2 OWL – WEB ONTOLOGY LANGUAGE 34

2.4.3 A OWL API 36

Capítulo 3 – MECANISMO PARA DETECÇÃO DE CONFLITO 39

3.1 ESTRUTURA DA NORMA 39

3.1.1 CONCEITO DEÔNTICO 40

3.1.2 CONTEXTO 41

3.1.3 ENTIDADE 41

3.1.4 COMPORTAMENTO 42

3.1.5 ATIVAÇÃO E DESATIVAÇÃO 43

3.2 ONTOLOGIA 43

3.2.1 CONTEXTO 45
3.2.2 ENTIDADE 46
3.2.3 COMPORTAMENTO 47
3.2.4 NORMA 48
3.3 RELACIONAMENTOS ENTRE OS ELEMENTOS DA NORMA 49
3.3.1 CONTEXTO 49
3.3.1.1 HIERARCHY ENTRE CONTEXTOS 50
3.3.1.2 INHABIT 51
3.3.2 ENTIDADE 52
3.3.2.1 PLAY 52
3.3.2.2 HIERARCHY ENTRE ENTIDADES 53
3.3.2.3 OWNERSHIP 54
3.3.3 COMPORTAMENTO 55
3.3.3.1 REFINEMENT ENTRE COMPORTAMENTOS 56
3.3.3.2 DEPENDENCY ENTRE COMPORTAMENTOS 58
3.3.3.3 COMPOSITION ENTRE COMPORTAMENTOS 61
3.3.3.4 ORTHOGONALITY ENTRE COMPORTAMENTOS 64
3.3.4 ATIVAÇÃO E DESATIVAÇÃO DA NORMA 66
Capítulo 4 – ESPECIFICAÇÃO E IMPLEMENTAÇÃO DO CONFLICT CHECKER 68
4.1 VISÃO GERAL DO CONFLICT CHECKER 68
4.2 ESPECIFICAÇÃO E IMPLEMENTAÇÃO 75
4.2.1 ESTRUTURA LÓGICA DAS CLASSES 75
4.2.2 A CLASSE CONFLICTCHECKER 76
4.2.3 SEQUÊNCIA DOS EVENTOS PARA VERIFICAR CONFLITOS NORMATIVOS 78
4.3 O CONFLICT CHECKER COMO UMA FERRAMENTA EXPANSÍVEL 83
Capítulo 5 – ESTUDO DE CASO E ANÁLISE DOS RESULTADOS 87

5.1 O CONTRATO DE COMPRA E VENDA 87
5.1.1 OS CONTEXTOS E ENTIDADES 87
5.1.2 OS ESTADOS E AÇÕES 89
5.2 AS NORMAS CONTIDAS NO CONTRATO 92
5.3 INSERINDO NOVAS NORMAS 94
5.4 O CONFLICT CHECKER E A EXISTÊNCIA DE CONFLITO 95
Capítulo 6 – TRABALHOS RELACIONADOS 101
6.1 KOLLINGBAUM ET AL. (2006) 101
6.2 VASCONCELOS ET AL. (2007) 102
6.3 KOLLINGBAUM ET AL. (2007) 104
6.4 VASCONCELOS ET AL. (2009) 106
6.5 APHALE ET AL. (2013) 108
6.6 SILVA, ZAHN (2013) 109
Capítulo 7 – CONCLUSÃO E TRABALHOS FUTUROS 111
7.1 RESUMO DO TRABALHO 111
7.2 CONTRIBUIÇÕES 113
7.3 LIMITAÇÕES 113
7.4 TRABALHOS FUTUROS 114
REFERÊNCIAS 116

CAPÍTULO 1 – INTRODUÇÃO

Agentes de Software são entidades computacionais interativas, autônomas e orientadas a objetivos. Em outras palavras, são capazes de realizar trocas de mensagens com outras entidades, possuem a capacidade de tomar decisões por conta própria e também são capazes de escolher quais atividades realizar para atingir seus objetivos. Essas características tornam os agentes de software "imprevisíveis" e independentes. (JENNINGS, 2000; ESTEVA et al., 2004; KOLLINGBAUM, NORMAN, 2004).

Normas têm sido o meio utilizado para reger o comportamento de agentes de software, através da descrição do que pode ser realizado (permissão), o que deve ser realizado (obrigação) e o que não deve ser realizado (proibição) (SILVA e ZAHN, 2013). No escopo da Sociologia, normas são regras que uma sociedade ou um grupo usa para definir valores apropriados e inapropriados, crenças, atitudes e comportamento (DEUTSCH e GERARD, 1955). No escopo de Sistemas Multi-Agentes (SMA), normas são regras sociais que influenciam o comportamento dos agentes (OREN et al., 2008).

As normas podem ser definidas em tempo de design juntamente com a modelagem do sistema, bem como podem ser geradas em tempo de execução por agentes que possuem essa capacidade (LÓPEZ, 2003). Nesta dissertação, todo o conjunto de normas analisado é gerado em tempo de design. Além disso, não está sendo considerado que as normas possam sofrer alterações durante o tempo de vida do sistema.

Quando um conjunto de normas é aplicado sobre um mesmo domínio, conflitos normativos podem surgir. Um conflito normativo ocorre quando o cumprimento de uma norma resulta na violação de outra norma. Assim, se um agente se depara com duas normas conflitantes, ao executar uma delas, o mesmo estará automaticamente violando a outra.

1.1 PROBLEMA

Normas têm sido o meio descritivo de grande relevância para garantir que agentes de software sejam agentes governáveis. Porém, a ocorrência de conflitos normativos é um importante problema a ser considerado.

Conflitos normativos em SMA tem sido o foco de diversas abordagens (KOLLINGBAUM et al., 2007; VASCONCELOS, KOLLINGBAUM e NORMAN, 2007; OREN et al., 2008; LI et al., 2013). No entanto, tais abordagens apresentam apenas mecanismos para a detecção de conflitos normativos diretos, não sendo eficientes para tipos de conflitos

mais complexos, tais como os que consideram as características e definições do domínio da aplicação.

Conflitos Normativos Diretos são detectáveis considerando-se apenas as definições contidas nas normas. Esse tipo de conflito ocorre quando os conceitos deônticos definidos nas normas são contraditórios, i.e., obrigação x proibição, permissão x proibição e quando os elementos contidos nas normas são iguais, i.e., conflitos entre normas que governam o mesmo comportamento executado pelas mesmas entidades). Abaixo é apresentado um conflito normativo direito entre duas normas:

```
N1: {obrigação, zona de guerra, agentes de Software, evacuar trabalhadores}
N2: {proibição, zona de guerra, agentes de Software, evacuar trabalhadores}
```

As normas N1 e N2 apresentam exatamente os mesmos elementos, pois regem o mesmo comportamento (*evacuar trabalhadores*) de uma mesma entidade (*agentes de software*) e em um mesmo contexto (*zona de guerra*). Por fim, os conceitos deônticos são opostos (*obrigação x proibição*), determinando assim um conflito direto entre tais normas.

Os Conflitos Normativos Indiretos são mais complexos e não detectáveis apenas com as definições contidas nas normas. Esse tipo de conflito exige o conhecimento sobre o domínio na qual as normas estão sendo aplicadas, pois as características desses domínios influenciam diretamente no resultado da verificação de conflitos entre normas (ver normas N3 e N4). Outra consideração importante sobre Conflitos Normativos Indiretos é que esse tipo de conflito ocorre quando os elementos das normas são diferentes, porém relacionados. É esse o tipo de informação (os relacionamentos entre os elementos) que é buscada nas definições do domínio da aplicação.

Abaixo é apresentado um exemplo de Conflito Normativo Indireto, onde a primeira norma obriga uma entidade *cabo* a *ajudar* em um contexto de *guerra*. A segunda norma proíbe um *soldado123* a prestar *auxílio* em um *campo de batalha*. Aparentemente, ambas as normas não apresentam conflito, mas ao obter o conhecimento sobre o domínio da aplicação, é detectado um conflito entre elas: um *campo de batalha* é um subcontexto de *guerra*, existindo portanto um relacionamento de hierarquia entre tais contextos. *Cabo* é um papel que está sendo desempenhado pelo *soldado123*, o que significa que existe um relacionamento entre as entidades. Por fim, o comportamento *ajudar* é um comportamento genérico que possui o comportamento *auxiliar* como um comportamento específico, havendo assim um relacionamento de hierarquia definido entre eles. Com isso, um conflito indireto pode ser detectado entre as normas N3 e N4.

N3: *{obrigação, guerra, cabo, ajudar}*
N4: *{proibição, campo de batalha, soldado123, auxiliar}*

Diferentemente dos conflitos diretos, que ocorrem apenas quando os conceitos deônticos são contraditórios, os conflitos normativos indiretos também podem ocorrer quando os conceitos deônticos são iguais (i.e., obrigação x obrigação). Isso é possível quando os comportamentos presentes no par normativo não podem ser executados em um mesmo período de tempo.

N5: *{obrigação, guerra, cabo, ajudar}*
N6: *{obrigação, campo de batalha, soldado123, fugir}*

No exemplo acima, os contextos e entidades estão relacionados, e os conceitos deônticos são iguais. Os comportamentos definidos em ambas as normas não podem ser executados em um mesmo período de tempo. Neste exemplo, o soldado *foge* ou o soldado *ajuda*. Devido as normas serem duas obrigações que não podem ser executadas em um mesmo período de tempo, um conflito normativo indireto com conceitos deônticos iguais surge. Note que até aqui não consideramos o fator tempo, ou melhor, consideramos que as normas se aplicam sobre um mesmo período de tempo.

Os relacionamentos definidos entre os elementos contidos nas normas devem ser conhecidos. Cada tipo de relacionamento possui características específicas e que devem ser consideradas. É importante ressaltar que entre dois elementos equivalentes de um par normativo, diversos tipos de relacionamentos podem existir.

No contexto desta dissertação, procura-se uma resposta à seguinte pergunta: como verificar conflitos normativos diretos e indiretos entre normas em Sistemas Multi-Agentes? Esse problema pode ser divido em quatro problemas menores:

a) Como obter conhecimento sobre o domínio da aplicação, i.e., sobre o conjunto de normas e sobre os elementos descritos nas mesmas?
b) Dado um conjunto de normas em um determinado domínio, como verificar conflitos normativos diretos?
c) Dado um conjunto de normas em um determinado domínio, como verificar conflitos normativos indiretos?
d) Após os casos de conflitos terem sido verificados, como expor os resultados da verificação ao usuário, apresentando quais normas estão e quais não estão em conflito e os motivos pelos quais um par normativo está em conflito?

1.2 SOLUÇÃO

Respondendo aos questionamentos da Seção anterior, apresentamos as seguintes soluções:

a) O conhecimento sobre o domínio da aplicação é fundamental para a verificação de conflitos indiretos. As normas a serem verificadas são parte da definição do domínio que compreende também os relacionamentos existentes entre os elementos identificados nas normas, por exemplo, relacionamentos entre as entidade e entre as ações sendo governadas. Nesta dissertação, a representação do domínio será feita utilizando-se uma ontologia que será definida nesta dissertação seguindo as características da norma e os tipos dos relacionamentos tratados nesta dissertação. Ontologias de domínio, segundo Musen (1998), são descrições formais de classes, conceitos e relacionamentos entre tais conceitos que descrevem um domínio de aplicação (conhecido também como área de aplicação). Assim, dado uma ontologia que representa o conhecimento sobre um domínio, o mecanismo deve ser capaz de compreender e tabular as informações contidas em tal ontologia;

b) Após a obtenção das normas oriundas do domínio da aplicação, deve-se aplicar algoritmos específicos para a verificação de conflitos normativos diretos. Neste caso, como apresentado anteriormente, não é necessário considerar os relacionamentos entre os elementos de um par normativo.

c) Com a obtenção do conhecimento sobre os relacionamentos entre os elementos do domínio é possível verificar conflitos indiretos entre normas que possuem elementos par-a-par diferentes, mas que estão relacionados. Com esse tipo de verificação e, dependendo da característica dos relacionamentos contidos entre os elementos das normas, conflitos podem inclusive ser detectados quando conceitos deônticos são iguais (i.g., obrigação e obrigação).

d) Desenvolver um protótipo capaz de contemplar as soluções propostas em *a*, *b* e *c*. Esse protótipo deve ser capaz de receber uma ontologia (através de um documento OWL) e detectar os conflitos normativos diretos e indiretos existentes no domínio. O protótipo deve também ser capaz de apresentar os resultados da verificação de maneira organizada, explicitando os motivos que levaram um par normativo a apresentar (ou não) um conflito.

1.3 OBJETIVOS

O objetivo principal dessa dissertação é o desenvolvimento de uma ferramenta denominada *Conflict Checker*: capaz de identificar conflitos normativos diretos e indiretos entre pares normativos em SMA, considerando as características do domínio contidas em ontologias.

Os objetivos específicos são:

- Desenvolver uma ontologia que seja modelo de domínio para a verificação de conflitos;
- Definir um modelo normativo que rege o comportamento de uma entidade em um contexto durante um período de tempo.
- Definir os relacionamentos entre os elementos de um par normativo;
- Criar regras para a verificação de conflitos entre pares de normas;
- Desenvolver um protótipo expansível a novos relacionamentos e regras de conflitos e que seja capaz de verificar conflitos entre pares normativos.

1.4 METODOLOGIA

Esse trabalho segue os procedimentos metodológicos que orientam a realização de pesquisas científicas (SANTOS, 2003), sendo divido em etapas. A primeira fase da pesquisa compreende três etapas:

i. A realização de uma revisão na bibliográfica visando trabalhos sobre normas em SMA e verificação de conflitos em Sistemas Multi-Agentes;
ii. Estudos sobre a utilização de ontologias para a representação de domínios;
iii. Estudos sobre a OWL-API (API de alto nível para a plataforma Java para criar e manipular ontologias OWL) e o seu funcionamento.

Esta fase da pesquisa teve como objetivo o levantamento de toda a fundamentação teórica para a realização das etapas seguintes. Com o levantamento dos trabalhos relacionados, buscou-se o conhecimento sobre os problemas já resolvidos e sobre os problemas ainda a resolver no tema pesquisado. Com base nestes problemas a resolver, os objetivos desta dissertação foram definidos.

A segunda fase da pesquisa compreendeu o desenvolvimento de um mecanismo para a verificação de conflitos normativos, que seja capaz de identificar as normas, os elementos e os relacionamentos de um domínio específico. Essa fase foi dividida nas seguintes etapas:

i. Definição da estrutura normativa;

ii. Definição dos relacionamentos existentes entre os elementos que compõe o modelo normativo;
iii. Levantamento de regras de verificação de conflitos normativos diretos e indiretos;
iv. Elaboração de um algoritmo em pseudocódigo baseado nas regras de verificação de conflitos elaboradas na etapa anterior;
v. Desenvolvimento de uma ontologia para a representação de um domínio específico, com base na definição das normas e dos relacionamentos (etapas i e ii);
vi. Desenvolvimento de um protótipo contento as regras de verificação de conflitos normativos (em Java).

A terceira e última fase compreende um estudo de caso utilizando normas que compõem um contrato de compra e venda do mundo real, a fim de avaliar a viabilidade do mecanismo proposto em atender as hipóteses que regem essa dissertação.

1.5 ORGANIZAÇÃO DO TEXTO

Essa dissertação está dividida em Capítulos e segue a estrutura definida a seguir:

2. **Referencial Teórico**: apresenta os conceitos fundamentais para o entendimento da solução proposta nesse trabalho. Descreve conceitos como Sistemas Multi-Agentes e suas características, definições de normas e sistemas normativos, conflitos normativos diretos e indiretos, apresenta conceitos sobre ontologias e sobre as ferramentas utilizadas no desenvolvimento desse trabalho.
3. **Mecanismo para detecção de conflito**: contém as definições sobre o modelo normativo utilizado e os elementos que compõem tal norma. Nesse Capítulo também são apresentados regras de verificação de conflito e os casos de conflitos entre pares normativos.
4. **Especificação e Implementação do *Conflict Checker***: apresenta a estrutura do *Conflict Checker*.
5. **Experimento e análise dos resultados**: apresenta um estudo de caso de verificação de conflitos, utilizando normas especificadas em um contrato do mundo real.
6. **Trabalhos relacionados**: apresenta um conjunto de trabalhos que antecederam essa dissertação e que estão relacionados com a proposta de verificar conflitos normativos.

7. **Conclusão e trabalhos futuros**: apresenta um resumo da pesquisa descrita nessa dissertação, expõe as dificuldades e limitações, as contribuições e sugestões para pesquisas futuras.

CAPÍTULO 2 – REFERENCIAL TEÓRICO

Neste Capítulo, são apresentados os conceitos fundamentais para o entendimento do restante do trabalho. Aqui serão apresentadas as fundamentações teóricas dos assuntos que serão tratados nos capítulos seguintes. Apresentamos conceitos sobre Sistemas Multi-Agentes e suas características, Sistemas Normativos e as ferramentas utilizadas para o desenvolvimento dessa dissertação.

2.1 SISTEMAS MULTI-AGENTES

Os SMA são sistemas computacionais compostos por agentes de software e que visam alcançar seus objetivos através da interação. Antes de descrever os SMA, é importante compreender as características dos agentes de software. Segundo (WOOLDRIDGE, 2009), um agente é um sistema computacional que está situado em algum ambiente e que é capaz de executar ações de maneira autônoma. O autor afirma ainda que a execução dessas ações é para cumprir com os objetivos delegados ao agente. Em outras palavras, agentes de software são programas autônomos (ou autossuficientes), sendo capazes de executar ações para atingir seus objetivos e de acordo com a percepção que o mesmo possui sobre o ambiente.

A Figura 1 apresenta o modelo de agente apresentado por WOOLDRIDGE (2009) sobre um determinado ambiente.

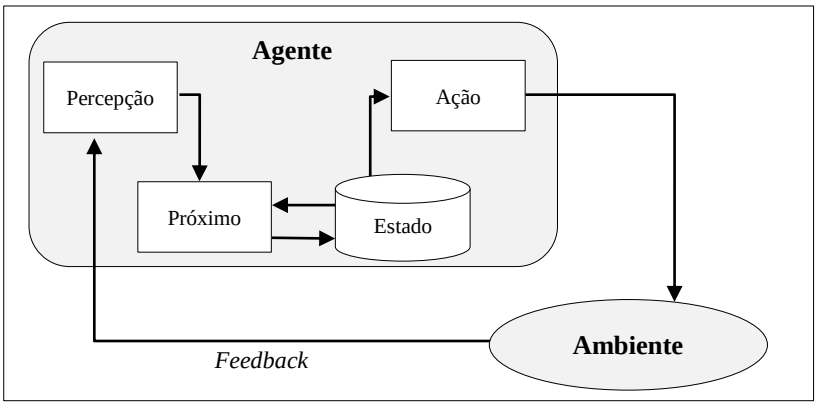

Figura 1: Modelo de agente apresentado por WOOLDRIDGE (2009).

De acordo com a Figura 1, o agente recebe um *feedback* do ambiente, o qual é captado através de sensores que notam as mudanças que estão ocorrendo. Através da informação captada pelo agente, que está em um determinado estado de execução, o mesmo decide por executar (ou não) uma ação. Se não executar a ação, a entidade pode esperar por novas percepções. Caso execute, os efeitos de sua ação podem alterar o ambiente e esta alteração pode

ser sentida pelo agente. Caso o agente não execute uma ação a partir da percepção atual, ele pode aguardar por novas percepções.

Mesmo que autonomia seja uma característica predominante em agentes de software, tais entidades usualmente necessitam interagir com outros agentes para que objetivos de maior complexidade possam ser alcançados. Com isso, a interação entre agentes de software surge com a intenção de alcançar os objetivos propostos a cada agente, derivando então o conceito de SMA.

De acordo com (DIGNUM, 2009), um SMA é constituído pelas entidades agentes, objetos e organizações. Uma organização pode ter seus próprios objetivos e, por tanto, pode ser considerada um tipo de entidade. No entanto, organizações podem também ser vistas como um conjunto de agentes (ou sub-organizações) agrupados. As três entidades apresentadas pelos autores habitam ambientes e cada entidade é capaz de armazenar informações sobre o ambiente no qual a mesma habita.

Os SMA podem ser classificados em dois tipos (BORDINI e VIEIRA, 2003):

- **Reativos**: as entidades de software que habitam tais sistemas possuem características extremamente simples. A definição de sistemas reativos partiu da analogia com a entomologia (estudos dos insetos) onde o comportamento das entidades dos sistemas é comparado ao comportamento dos insetos. Nesse tipo de SMA, os agentes não possuem a percepção direta do ambiente e do comportamento dos outros agentes. No entanto, possuem um conjunto de regras que mapeiam as percepções do ambiente a um conjunto de tarefas fazendo com que o agente possa agir sobre um esquema estimulo-resposta. Em outras palavras, o comportamento das entidades pertencentes a esse tipo de SMA pode ser modelado como um autônomo finito simples;
- **Cognitivos**: as entidades deste sistema possuem características computacionalmente complexas, como as citadas em BORDINI e VIEIRA (2003):
    - **Percepção**: o agente cognitivo é capaz de perceber as mudanças a sua volta, i.e., as mudanças que ocorrem no ambiente em que o mesmo se faz presente. Essa percepção pode ser notada através de sensores e que está representada na Figura 1;
    - **Ação**: as ações interferem no comportamento do ambiente. Cada agente é capaz de executar ações a fim de alcançar o seu objetivo. Desse modo, cada ação visa alterar o ambiente atual para um ambiente desejado associado ao objetivo do agente;

- **Comunicação**: esse é um tipo de ação fundamental para o agente. É através da comunicação que agentes podem coordenar suas ações e assim, atingir objetivos através dessa coordenação;
- **Representação**: a representação é o que o agente possui como verdade sobre o ambiente no qual habita e sobre as entidades que habitam nesse mesmo ambiente;
- **Motivação**: a motivação é a representação dos objetivos do agente, ou seja, representação de estados do ambiente que o agente deseja alcançar. A motivação é fundamental devido a autonomia dos agentes, pois é devido a essa autonomia, juntamente com a motivação, que os agentes tomam iniciativas para atingir os estados desejados;
- **Deliberação**: o agente é capaz de decidir quais caminhos percorrer, quais estados atingir e, de fato, quais serão os objetivos a serem seguidos por ele;
- **Raciocínio e aprendizagem**: são técnicas de Inteligência Artificial clássicas e que contribuem para um melhor desempenho do agente. Essa característica (raciocínio e aprendizagem) não estão presentes, necessariamente, em todos os agentes de software.

2.2 NORMAS E SISTEMAS NORMATIVOS

A autonomia e a heterogeneidade dos agentes de software exige que o comportamento de tais agentes sejam regulados por um sistema normativo. Sistemas normativos são compostos por um conjunto de normas (ou regras) que visam instruir o agente sobre aquilo que ele pode fazer (permissão), sobre aquilo que não pode fazer (proibição) e sobre aquilo que deve ser feito (obrigação). No contexto sociológico, normas são regras que uma sociedade ou um grupo utiliza para definir comportamentos, atitudes, valores e crenças, classificando-as como apropriadas ou inapropriadas (DEUTSCH e GERARD, 1955).

No contexto de SMA, normas são o meio descritivo para governar o comportamento dos agentes, em um determinado contexto e período. De acordo com (JIANG et al., 2013), as normas possuem sujeitos que devem cumprir com a mesma. Além disso, os autores também definem normas como sendo um conjunto de “o que fazer” e “o que não fazer”. Definem também que normas são fatores dependentes do ambiente institucional no qual são aplicadas. Em outras palavras, um ambiente *a* é um conjunto de estados definidos por predicados sobre aspectos como a individualidade, tempo, localização, ação e relação. Para (KOLLINGBAUM e NORMAN, 2006), agentes possuem a capacidade de desempenhar papéis e tais papéis possuem restrições sociais aplicadas através das normas.

2.2.1 MODELOS DE NORMAS

Diversas especificações normativas, linguagens de modelação, metodologias e modelos organizacionais definem o conceito de normas. Em geral, as especificações definem uma norma com uma tupla que identifica um conceito deôntico (permissão, obrigação ou proibição), uma entidade e um comportamento (execução de uma ação ou alcance de um estado). Cada abordagem possui um modelo de norma que satisfaça as necessidades do que está sendo abordado. Abaixo são apresentados os principais modelos de normas utilizados em diversas pesquisas:

- KOLLINGBAUM e NORMAN (2006): Kollingbaum e Norman não deixam a estrutura normativa claramente especificada. No entanto, podemos perceber a utilização de uma estrutura normativa que possui como elementos um conceito deôntico (permissão, obrigação e proibição), uma entidade na qual a norma está sendo aplicada e uma ação que é o comportamento que está sendo regulado. Desse modo, a estrutura normativa fica da seguinte maneira:

$$n = \{conceito\ de\hat{o}ntico, entidade, a\tilde{c}ao\}.$$

- OREN et al., (2008): Nesse trabalho é apresentada uma estrutura normativa simples, constituída apenas de um conceito deôntico, um contexto social e uma meta. A norma é do tipo N (onde $N \in \{O, P, F\}$) que é, respectivamente, uma obrigação, permissão ou proibição. A norma é imposta sobre um agente, que deve cumprir com a norma. Em geral, a norma se aplica sobre um agente, mas os autores não eliminam a possibilidade dessa norma ser aplicada sobre um papel ou um grupo de agentes. Um *contexto social* representa uma entidade social que deve executar a norma e ele é representado na estrutura normativa através da letra *c*. Por fim, a norma apresenta o elemento *meta* – g –, que é o que a entidade deve fazer (caso a norma seja uma obrigação) ou não deve fazer (caso a norma seja uma proibição). No caso da permissão, fica sobre responsabilidade do agente a execução ou não execução da *meta* estipulada pela norma. Assim, a norma N apresenta a seguinte estrutura:

$$Nc(g)$$

- VASCONCELOS, KOLLINGBAUM e NORMAN (2008): É apresentado um modelo normativo onde a norma ω é uma tupla, definida conforme a seguir:

$$\omega\ \{\nu, t_d, t_a, t_e\}\ \text{tal qual}\ \nu,$$

- $\mathbf{O}_{\alpha:\rho\varphi}$ o Γ (é uma obrigação),
- $\mathbf{P}_{\alpha:\rho\varphi}$ o Γ (é uma permissão),
- $\mathbf{F}_{\alpha:\rho\varphi}$ o Γ (é uma proibição),

onde α e ρ são termos, φ o Γ é uma fórmula atômica de primeira ordem com restrições; $t_d, t_a, t_e \in \boldsymbol{IN}$ são, respectivamente, o tempo quando ν é introduzida, quando ν é ativada e quando ν expira, $t_d \leq t_a \leq t_e$. O termo α identifica o agente da norma e o termo ρ é o papel de tal agente. Assim, a norma utilizada pelos autores possui um conceito deôntico, uma entidade, um papel, um comportamento e fatores de ativação e desativação.

- JIANG et al. (2013): Os autores utilizam uma estrutura normativa definida na sintaxe ADICO[1], que descreve a quem (entidade) a norma está aplicando a obrigação, permissão ou proibição e qual o comportamento está sendo regulado pela norma (executar uma ação ou atingir um estado), quando e onde (condição) a norma deve ser cumprida, e que o não cumprimento de tal norma leva a violações. Assim, os autores definem a norma como sendo uma tupla *n*, conforme apresentado abaixo:

$$n = \{papel, conceito\ deôntico, ação, condição\}.$$

- SILVA e ZAHN (2013): O autor apresenta uma estrutura normativa com um elevado nível de detalhamento dos elementos. Segundo o autor, a norma *n* é composta por um conceito deôntico (*deoC*) de um conjunto {obrigação ou proibição}, um contexto *c* (pertencente a um conjunto de contextos) onde a norma é definida, uma entidade *e* (de um conjunto de entidades) que é a entidade que deverá cumprir com a norma, uma ação *a* (de um conjunto de ações) que está sendo regulada, uma condição de ativação *ac* (de um conjunto de condições) que é o fator de ativação da norma, uma condição de desativação da norma *dc* e, por fim, o status *s* da norma (cumprida, violada ou nulo). Assim, esse completo modelo normativo possui a seguinte estrutura:

$$n = \{deoC, c, e, a, ac, dc, s\}.$$

Existem inúmeras abordagens que apresentam estruturas normativas, cada uma definida sobre aquilo que corresponde aos conceitos abordados em cada trabalho. Como é possível perceber, em geral, uma norma é uma tupla constituindo-se de um conceito deôntico, uma

[1] Expressão natural de normas que provê uma descrição clara das oportunidades e restrições que criam expectativas sobre o comportamento dos atores, tornando a norma um modelo intuitivo para que os atores consigam compreender o que pode ser feito e o que não pode ser feito (JIANG et al., 2013).

entidade e um comportamento. A seguir, são descritos os elementos comuns entre as normas, sob a ótica de diversas abordagens e que são citados nas descrições abaixo:

- **Conceito deôntico**: O conceito deôntico é o elemento mais comum entre as diversas abordagens. A lógica deôntica refere-se à lógica de pedidos, leis, comandos, regras, julgamento e princípios morais (MEYER e WIERINGA, 1993). Em SMA, tais conceitos têm sido usados para representar restrições de comportamento sobre as entidades, definido o que poderá ser feito (permissão), deverá ser feito (obrigação) ou não poderá ser feito (proibição). Para FIGUEIREDO (2011), o conceito deôntico é um dos principais elementos presentes em uma norma, cujo o objetivo é a identificação do tipo de restrição que está sendo definido. Nota-se então a elevada importância dos conceitos deônticos na definição uma estrutura normativa.
- **Contexto**: As normas são definidas em contextos, que determinam a área em que a mesma será aplicada. Para FIGUEIREDO (2011), a norma pode ser definida no contexto de um ambiente, o que significa que todos os agentes sendo executados em tal contexto deverão cumprir com a norma. A norma pode também ser aplicada no contexto de organização, o que significa que todas as entidades que desempenham papéis em tal organização deverão cumprir com a norma. O contexto é definido por LÓPEZ (2003) como sendo um estado do ambiente que deve ser atingido para que uma norma possa ser executada. Esse é um modo interessante de enxergar contextos, pois quando a norma se aplica sobre um contexto *c*, apenas os agentes presentes em tal contexto deverão cumprir com a norma.
- **Entidade**: A entidade é definida em uma norma a fim de identificar a quem a norma está sendo aplicada. A norma pode restringir o comportamento de uma entidade de maneira individual (por exemplo, a um único agente), ou coletiva. Nesse caso a norma restringe o comportamento de um grupo de entidades (por exemplo, todos os agentes desempenhando um determinado papel, grupos de agentes, todos os agentes presentes no sistema). Se a norma é aplicada sobre uma entidade, significa que tal entidade deve cumprir com a norma (FIGUEIREDO, 2011). Para OREN et al. (2008), entidades e contextos são mencionados de modo conjunto, como se fossem apenas um. Contextos e entidades estão diretamente ligados, pois uma entidade pertence, em toda a sua vida útil, a algum contexto no ambiente. Em outras palavras, se mencionamos entidade, logo mencionamos

o contexto onde essa entidade está presente. Assim, OREN et al. (2008) apresentam um modelo de norma que é aplicado sobre um *contexto social*, ou seja, aplicado sobre o agente que pertence a tal contexto. Em SILVA e ZAHN (2013), os autores descrevem que toda entidade possui um nome e um tipo, podendo esse tipo ser um agente, papel ou organização. Assim, se a norma é especificada sobre um agente, apenas o agente deverá cumprir com a norma. Se a norma é aplicada sobre um papel, todos os agentes que desempenham tal papel devem cumprir com a norma. Se a norma se aplica sobre uma organização, todas as entidades que desempenham papéis em tal organização deverão cumprir com as normas. A norma também pode especificar todos os agentes, o que significa que todos os agentes no referido contexto devem cumprir com a norma.

- **Comportamento**: O comportamento pode ser divido em dois tipos que se relacionam. O estado é a situação que se encontra o domínio, ou de parte dele. Para a execução de ações, a entidade que deseja executá-la, deverá atingir o estado *condição* para a execução da ação. A ação, ao fim da sua execução, leva a entidade que a executou para um novo estado, chamado de estado *efeito*. Isso ocorre devido ao fato que ações alteram o estado *condição*, gerando o estado *efeito*. Estados e ações possuem então um relacionamento definido por condição, execução e efeito (RUSSELL e NORVIG, 1995). Diversas abordagens consideram apenas a ação como o comportamento a ser restringido pela norma, como SILVA e ZAHN (2013), KOLLINGBAUM e NORMAN (2006), VASCONCELOS et al (2009), e JIANG et al. (2013). No entanto, quando o estado é considerado, ou seja, quando normas são aplicadas sobre ações e sobre estados, o relacionamento que tais comportamentos possuem entre eles devem ser considerados. A execução de uma ação pode ser comunicativa (envio ou recebimento de mensagens) ou não comunicativas (acessar ou modificar recursos, entrar em uma determinada organização, mover para outro ambiente, etc.), afirma FIGUEIREDO (2011). Um estado é o modo como se encontra uma entidade em um determinado momento no tempo, um conjunto de entidades ou o ambiente como um todo. A ação modifica o estado atual, gerando um novo estado resultante dessa modificação.
- **Evento de ativação e desativação**: Os eventos de ativação e desativação de uma norma nem sempre estão presentes nas abordagens. Os eventos de ativação e desativação pode ser, por exemplo, a execução de uma ação, um determinado

> instante no tempo, o alcance de um estado, o cumprimento ou a violação de uma norma, entre outros (ZAHN e SILVA, 2014). Uma norma possui um evento de ativação que representa o momento em que a norma passa a ter validade sobre o contexto, entidade e comportamento, i.e., o momento no qual a entidade deve cumprir a norma. O evento de desativação define quando a norma deixa de estar ativa, i.e., quando a norma é desativada e a entidade não necessita mais cumprir com exigências da norma.

2.3 CONFLITOS ENTRE NORMAS

Devido ao elevado número de normas que podem ser aplicadas sobre um mesmo domínio, conflitos normativos podem surgir. Um conflito entre duas normas ocorre quando o cumprimento de uma norma resulta na violação de outra, e vice-versa. Diversas abordagens propõem métodos para verificação de existência de conflitos entre normas (conforme será apresentado no Capítulo 6).

2.3.1 CONFLITO VS. INCONSISTÊNCIA

Antes de definir conflito e inconsistência, é importante compreender o que é um *escopo de influência*. Cada norma possui um escopo de influência, que nada mais é do que a área de atuação da norma (o contexto na qual atual, as entidades que devem cumprir com a norma, o comportamento e o tempo em que a norma permanecerá ativa). Para que um conflito ou inconsistência possa ocorrer, o escopo de influência das normas deve sobrepor umas às outras.

Além da sobreposição entre as normas, um *conflito* é caracterizado pelo confronto de duas normas, onde uma é proibitiva e a outra permissiva. Quando normas que possuem o escopo de influência que sobrepõe umas sobre as outras são proibitivas e obrigatórias, esse confronto é definido como *inconsistência*, afirma KOLLINGBAUM e NORMAN (2006).

Nesse trabalho, não adotamos o termo inconsistência entre normas. Tanto os confrontos entre proibições e permissões quando os entre proibições e obrigações chamaremos de conflito. Como será visto mais a seguir, conflitos também podem ocorrer quando os conceitos deônticos são iguais (obrigação x obrigação) e até quando uma norma define uma obrigação e outra uma permissão.

2.3.2 CONFLITOS DIRETOS

Os **conflitos diretos** surgem quando os conceitos deônticos são opostos (por exemplo, obrigação e proibição, ou permissão e proibição), quando as normas são aplicadas sobre uma

mesma entidade, sobre um mesmo comportamento, em um mesmo contexto e quando os eventos de ativação e desativação das normas implicam em uma interseção temporal.

A fim de exemplificar os conflitos, nos basearemos em um cenário apresentado em VASCONCELOS et al. (2009). Porém, antes da apresentação deste cenário, é importante compreender a estrutura utilizada para representar as normas deste cenário. As normas aplicadas no cenário possuirão a seguinte estrutura:

Template de definição das normas utilizadas nos exemplos:
N: *{conceitoDeôntico, contexto, entidade, comportamento}*

No cenário apresentado em VASCONCELOS et al. (2009), agentes de software possuem equipamentos de evacuação e fornecimento de informações para não-combatentes em zona de conflito. Diversas normas são aplicadas sobre os agentes por diferentes superiores, o que pode gerar um conflito normativo devido à falta de comunicação entre os superiores. O objetivo dos agentes é salvar humanos, no entanto, fatores externos como o clima podem dificultar a operação e fazer com que os superiores insiram novas normas que assegurem o sucesso da missão. Vamos supor que em um determinado momento as normas ***N1*** e ***N2*** estejam ativas no sistema.

N1: *{obrigação, zona de guerra, agentes de Software, evacuar trabalhadores}*
N2: *{proibição, zona de guerra, agentes de Software, evacuar trabalhadores}*

Ambas normas são aplicadas sobre a mesma entidade (*agentes de software*), sobre um mesmo comportamento (*evacuar trabalhadores*) e sobre um mesmo contexto (*zona de guerra*). Como elas estão associadas a conceitos deônticos opostos (uma norma proíbe e outra obriga) podemos afirmar que as duas normas estão em conflito.

2.3.3 CONFLITOS INDIRETOS

Os **conflitos indiretos** surgem quando os elementos de uma norma não são iguais aos respectivos elementos contidos em outra norma. No entanto, os elementos de ambas normas são relacionados. Aparentemente, o conflito não existe, mas através de uma verificação mais detalhadas dos elementos que compõem as normas e dos relacionamentos existentes entre estes elementos, o conflito pode ser desvendado. A primeira diferença entre conflitos normativos diretos e conflitos normativos indiretos é que, o segundo, não busca apenas por conflitos existentes entre normas aplicadas sobre entidades, contextos e comportamentos iguais. Os conflitos indiretos são encontrados quando analisamos as informações sobre o domínio da aplicação, i.e., quando analisamos os relacionamentos existentes entre as entidades, os

contextos e os comportamentos (conforme apresentado no Capítulo 3). Conflitos indiretos, diferentes dos conflitos diretos, podem ainda surgir quando normas são aplicadas sobre conceitos deônticos iguais (por exemplo, obrigação x obrigação). Isso é possível quando os comportamentos presentes no par normativo não são possíveis de serem executados em um mesmo período de tempo.

N3: *{obrigação, guerra, cabo, ajudar feridos}*
N4: *{proibição, campo de batalha, soldado, auxiliar feridos}*

Acima foi exposto um exemplo de conflito normativo indireto, onde a primeira norma obriga uma entidade *cabo* a *ajudar os feridos* em um contexto de *guerra*. A segunda norma proíbe um *soldado* a prestar *auxílio aos feridos* em um *campo de batalha*. Aparentemente, ambas as normas não apresentam conflito, mas ao obter o conhecimento sobre o domínio da aplicação, é detectado um conflito entre elas. Um *campo de batalha* é um subcontexto de *guerra*, existindo um relacionamento definido entre tais contextos. O *cabo* é um papel que está sendo desempenhado pelo *soldado*, o que significa que existe um relacionamento entre as entidades. Por fim, o comportamento *ajudar* é um relacionamento genérico que possui o comportamento *auxiliar* como um comportamento específico, havendo assim um relacionamento definido entre eles. Com isso, um conflito indireto é afirmando entre ambas as normas.

2.4 ONTOLOGIA

O termo Ontologia não possui um consenso comum entre os autores que lidam com o tema. No entanto, muitos concordam que ontologia é o meio de compartilhar e permitir a reutilização de conhecimento (GUARINO et al., 2009). Na Inteligência Artificial (IA), existem diversas definições para o termo ontologia. Uma das mais consensuais é apresentada em STUDER et al. (1998) onde o autor define uma ontologia como sendo uma especificação explícita e formal de uma conceptualização partilhada.

Uma das principais características do desenvolvimento de ontologia é a reutilização daquilo que já se sabe sobre o domínio. Em outras palavras, o processo custoso de elaboração do conhecimento pode ser reduzido em desenvolvimentos futuros através da reutilização daquilo que já se conhece sobre a área de conhecimento a ser desenvolvida LANÇA et al. (2010).

Ontologias passou a ser recomendação da W3C para sua utilização em Web Semântica como sendo um padrão de vocabulário para a troca de dados e informações, além de tornar o conhecimento reutilizável e facilitar a comunicação de sistemas heterogêneos (BERNERS-LEE

et al., 2001; WEB [200-?]; GRUBER, 2008). Essa recomendação motivou a utilização de ontologias para a obtenção de conhecimento sobre o domínio.

De acordo com Guarino (1997), ontologias podem ser classificadas conforme a sua função, ao grau de formalismo, aplicação, estrutura, conteúdo, entre outras maneiras. Abaixo é apresentado a classificação de ontologias de acordo com a sua função:

- **Ontologias Genéricas**: Esse tipo de ontologia não depende de um problema específico (domínio). É um tipo de ontologia que representa um conjunto de conceitos mais amplos, tais como conceitos da natureza, relativos ao espaço e ao tempo.
- **Ontologias de Domínio**: Representa conceitos específicos, de uma área (domínio) específico. A computação e a medicina são tipos de domínios específicos. Esse tipo de ontologia é uma das mais comuns.
- **Ontologias de Aplicação**: É semelhante à Ontologias de Domínio, mas são descritas para uma determinada tarefa.
- **Ontologias de Representação**: É um tipo de ontologia utilizada para representar conhecimentos.
- **Ontologias de Tarefa**: De uma maneira geral, descreve conceitos que são utilizados por processos, independente de domínio.

No contexto dessa dissertação, serão utilizadas ontologias de domínio devido suas características. Além das características apresentadas acima, ontologias de domínio são descrições formais de classes, conceitos e relacionamentos entre tais conceitos que descrevem um domínio de aplicação (conhecido também como área de aplicação), afirma MUSEN (1998).

A análise ontológica esclarece a estrutura do conhecimento. Dado um determinado domínio, sua ontologia forma o coração de qualquer sistema de representação de conhecimento para tal domínio. Sem a utilização de ontologias, ou sem a conceptualização que constitui a base do conhecimento, o vocabulário para a representação do conhecimento não pode ser desenvolvido. Desse modo, o primeiro passo ao idealizar uma efetiva representação do sistema (e vocabulário), é realizar uma efetiva análise ontológica da área da aplicação – ou domínio da aplicação (CHANDRASEKARAN et al., 1999).

Na Figura 2, é apresentado graficamente uma pequena ontologia. Esta ontologia define entidades e contextos. A classe *Entity* possui três subclasses (*Organization, Agent* e *Role* conforme a legenda), o que significa que existem três tipos de entidades: Organização, Agente e Papel. A classe *Context* possui duas subclasses (*Organization* e *Environment*, conforme a

legenda). Além de definir as classes, podemos ainda obter informações dos indivíduos e das propriedades, conforme descritas abaixo:

- *Maria* e *João* são entidades do tipo Agente;
- *Estudante* e *Pesquisador* são entidades do tipo papel;
- *Campus Praia Vermelha* é uma entidade do tipo organização;
- *Universidade* é um contexto na qual as entidades podem habitar;
- *João* é um *Pesquisador*;
- *Maria* é uma *Estudante*;
- *Estudante* e *Pesquisador* são papéis desempenhados na Organização *Campus Praia Vermelha*.

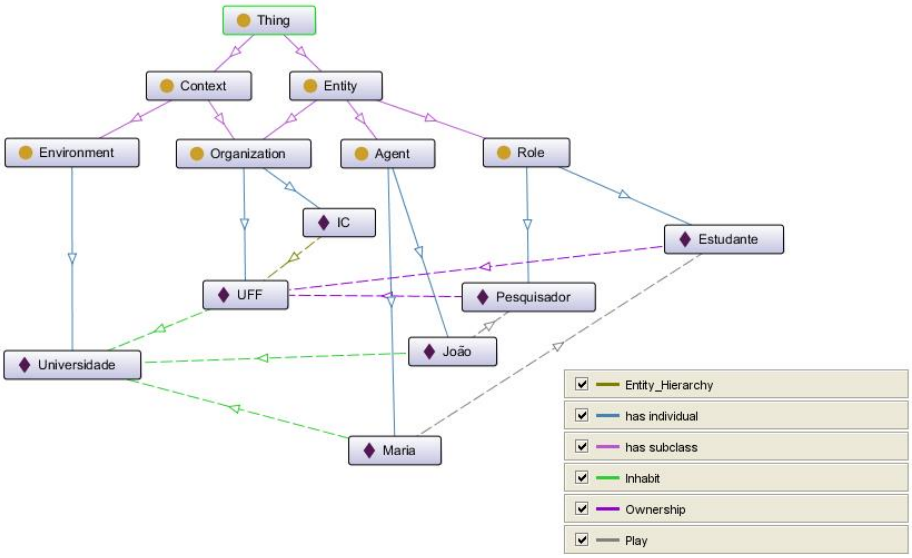

Figura 2: Uma pequena ontologia, seus indivíduos e propriedades.

Através das informações extraídas da ontologia, obtemos um conhecimento exato sobre o relacionamento entre as entidades e entre as entidades e o contexto presentes no domínio. Esse tipo de conhecimento é base para que conflitos normativos indiretos sejam encontrados.

Nesse trabalho, a ontologia representará o conhecimento sobre o domínio da aplicação no qual estão inseridos as entidades e os contextos do SMA. O objetivo da utilização de ontologia neste trabalho é a possibilidade de representação dos elementos identificados nas normas assim como dos relacionamentos entre esses elementos dentro do contexto do SMA. Mais detalhes sobre a ontologia utilizada neste trabalho serão descritos na Seção 3.2.

2.4.1 PROTÉGÉ

Para a criação e manutenção de ontologias, optou-se por utilizar o software Protégé[2]. O Protégé é um editor gratuito de ontologias e um framework para a criação de sistemas inteligentes. É uma ferramenta fortemente aceita pela comunidade acadêmica, órgãos governamentais e usuários corporativos. Através da utilização do Protégé, é possível criar soluções baseadas em conhecimento em diversas áreas, tais como a biomedicina, e-commerce e modelagem organizacional (PROTÉGÉ, [201-?]).

A escolha pela utilização do Protégé se dá por ser uma ferramenta com um elevado índice de aceitação na comunidade acadêmica e que é disponibilizada gratuitamente para pesquisadores e desenvolvimento de projetos e pesquisas científicas. Outro fator determinante é a capacidade do Protégé em exportar ontologias através de documentos OWL. Assim, o conhecimento do domínio elaborado através do Protégé pode ser exportado para um documento OWL, contendo toda a descrição das classes e conceitos presentes na ontologia.

2.4.2 OWL – WEB ONTOLOGY LANGUAGE

A OWL é uma linguagem de ontologias para a Web Semântica, que foi desenvolvida pelo *Web Ontology Working Group* da W3C (*World Wide Web Consortium*). Uma ontologia OWL descreve o domínio em termos de classes, propriedades e indivíduos, podendo ainda conter as descrições das características desses objetos (BECHHOFER, 2009). Em seu projeto original, as expectativas eram voltadas ao uso da OWL na representação de categorias de objetos e como os objetos podem estar inter-relacionados. Como uma linguagem para descrever ontologias, a OWL é capaz de representar um grupo de ontologias e seus recursos (HORROCKS et al., 2003).

A Figura 3 apresentada parte da ontologia contida na Figura 2, a fim de que seja possível compreender trechos fundamentais da sintaxe e funcionamento da linguagem OWL.

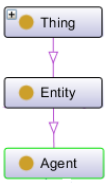

Figura 3: Representação da Classe *Agent*.

[2] http://protege.stanford.edu/

As informações dessa Hierarquia entre Classes é armazenada pela linguagem OWL, a fim de que as propriedades da ontologia sejam armazenadas. Desse modo, a OWL define esse trecho da ontologia conforme abaixo:

```
Declaration(Class(<http://www.ic.uff.br/~jzahn/ontology/Uni#Agent>))
 SubClassOf(<http://www.ic.uff.br/~jzahn/ontology/Uni#Agent>
            <http://www.ic.uff.br/~jzahn/ontology/Uni#Entity>)
```

Note que existe uma declaração do tipo Classe, seguido da IRI com o nome da classe. A IRI da ontologia está definida como `http://www.ic.uff.br/~jzahn/onto/Uni`, o que significa que todas os indivíduos, classes e propriedades são anexados ao fim dessa IRI, seguido pelo símbolo # e o nome.

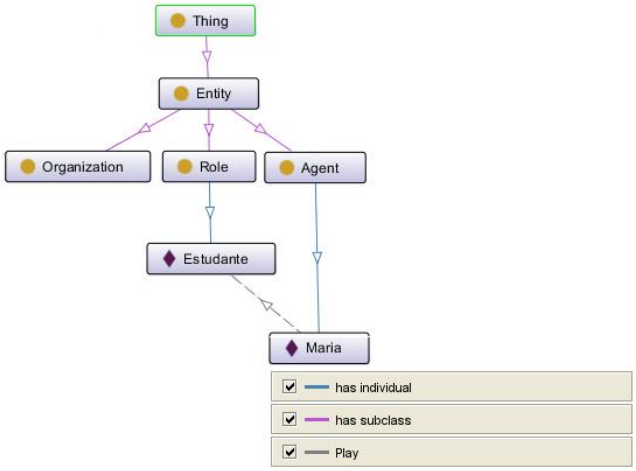

Figura 4: Representação do relacionamento Maria é Estudante.

Ontologias não são formadas apenas por classes, mas devem conter Indivíduos (objetos), propriedades (relacionamento entre os indivíduos) e as características dos indivíduos (informações adicionais contidas em cada objeto). A Figura 4 apresenta graficamente o relacionamento entre dois indivíduos instanciados na ontologia. O relacionamento entre *Maria* e *Estudante* é a Propriedade do Objeto *Play*. Isso significa que através da análise ontológica, é constatado que *Maria* é uma *Estudante*, desempenhando então esse papel. A linguagem OWL define esse tipo de relacionamento da seguinte maneira:

```
Declaration(ObjectProperty(<http://www.ic.uff.br/~jzahn/onto/Uni#Play>))
```

Após definir o `ObjectProperty` – que é o tipo de relacionamento entre os indivíduos – é necessário definir os indivíduos e a qual classe cada indivíduo pertence, conforme abaixo:

```
Declaration(NamedIndividual(<http://www.ic.uff.br/~jzahn/2014/Uni#Maria>))
```

```
ClassAssertion(<http://www.ic.uff.br/~jzahn/2014/Uni#Agent>
                <http://www.ic.uff.br/~jzahn/2014/Uni#Maria>)
```

A propriedade `NamedIndividual` cria um indivíduo qualquer. No entanto, é necessário associar esse indivíduo a uma classe correspondente. A propriedade `ClassAssertion` é quem gera essa informação, afirmando que a classe *Agent* é a classe responsável por *Maria*.

A existência de dois indivíduos no relacionamento definido pelo `ObjectProperty` é afirmada pela propriedade `ObjectPropertyAssertion`. Essa propriedade efetua essa informação conforme a seguir:

```
ObjectPropertyAssertion(<http://www.ic.uff.br/~jzahn/2014/Uni#Play>
                        <http://www.ic.uff.br/~jzahn/2014/Uni#Maria>
                        <http://www.ic.uff.br/~jzahn/2014/Uni#Estudante>)
```

As propriedades até aqui apresentadas são algumas das dezenas que a linguagem OWL possui. Toda e qualquer necessidade de representação de ontologias pode ser realizada pela linguagem OWL.

2.4.3 A OWL API

A OWL API[3] é uma *Application Programming Interface* (API) de alto nível para a plataforma Java e é referência para criar e manipular ontologias OWL (*Web Ontology Language*). A OWL API foi desenvolvida e é atualmente mantida pela Universidade de Manchester, mas que, durante o seu desenvolvimento, obteve contribuições diretas da Clarck & Parsia[4] e Ulm University[5].

De acordo com HORRIDGE e BECHHOFER (2011), a OWL API possui alguns princípios fundamentais em seu projeto, conforme listados abaixo:

- Prover uma interface de acesso a estrutura do modelo, acesso na qual somente a leitura é permitida;
- Alteração/manipulação de ontologias através de operações explícitas de alteração;
- Independência das serializações concretas;
- Uma clara separação entre os componentes oriundos de funcionalidades particulares, tal como a representação, manipulação, análise e renderização.
- Separação de declarações e inferências.

[3] http://owlapi.sourceforge.net/
[4] Empresa de desenvolvimento de tecnologias, com foco particular em OWL. http://clarkparsia.com/
[5] http://www.uni-ulm.de/in/ki/noppens

A Figura 5 apresenta um diagrama UML com a estrutura do gerenciamento de ontologias em OWL API.

Cada ontologia possui um meio pela qual é identificada. Em OWL API, as IRIs (*Internationalized Resource Identifiers*) possuem um papel fundamental quanto na identificação da ontologia, suas classes, instâncias e propriedades. No diagrama apresentado na Figura 3, essa identificação ocorre através da interface OWLOntologyID.

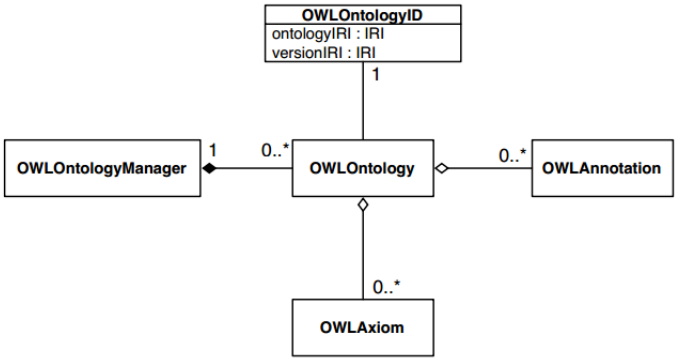

Figura 5: Diagrama UML apresentando o gerenciamento de ontologias em OWL API, conforme HORRIDGE e BECHHOFER (2011).

A interface OWLOntologyManager provê um ponto central de acesso para a criação, e para carregar, alterar e salvar ontologias, que são instancias da interface OWLOntology. Cada ontologia é criada ou carregada pelo OWLOntology. Outra característica importante é que cada instância de uma ontologia é única e possui um OWLOntologyManager particular. Todas as mudanças na ontologia são aplicadas através desse gerenciador (HORRIDGE; BECHHOFER, 2011).

O OWLOntology (Figura 5) fornece um ponto de acesso aos axiomas contidos em uma ontologia. Assim como fornece um ponto de acesso às anotações presentes na ontologia. Os axiomas são os principais componentes de uma ontologia OWL. As anotações são fundamentais para agregar informações adicionais com a ontologia, entidades e axiomas. Para isso, ontologias OWL fornecem uma interface capaz de agregar tais informações. Desse modo, é notável a importância dos axiomas e anotações, configurando então uma necessidade de um ponto de acesso fácil através da OWL API.

Nesse trabalho, o foco não está voltado a alteração de ontologias. Desse modo, grande parte das funcionalidades da OWL API não serão utilizadas. O que é necessário compreender está relacionado com consultas sobre ontologias. Nesse ponto, a OWL API por si só, não fornece um mecanismo de alto nível para consulta. No entanto, utilizando-se da interface

OWLReasoner é possível ter o conhecimento sobre as propriedades de hierarquia, determinar instâncias, determinar características das propriedades dos objetos e das propriedades dos dados, entre outros. Com tais consultas básicas, conseguimos as informações necessárias para a verificação de conflitos entre normas que se encontram no domínio em questão (HORRIDGE; BECHHOFER, 2011).

CAPÍTULO 3 – MECANISMO PARA DETECÇÃO DE CONFLITO

No Capítulo 2 foram apresentadas diversas estruturas de normas propostas em diferentes trabalhos relacionados à verificação de conflitos. Nesse Capítulo são apresentadas definições fundamentais para a detecção de conflitos normativos. Na Seção 3.1 apresentamos a estrutura normativa utilizada neste trabalho. Na Seção 3.2 apresentamos a ontologia que descreve os conceitos e os relacionamentos entre os elementos de um SMA e que também descreve as normas seguindo a estrutura normativa apresentada na Seção 3.1. Note que a parte da ontologia que descreve os conceitos e relacionamentos utilizados para representar o domínio das aplicações no escopo de SMA não pretende ser completa a ponto de contemplar todos os possíveis conceitos e relacionamentos. Os conceitos abordados são aqueles, e somente aqueles, utilizados na estrutura normativa. Os relacionamentos contemplados são aqueles apresentados em SILVA e ZAHN (2013) e ZAHN e SILVA (2014). A Seção 3.3 apresenta os relacionamentos entre os elementos das normas considerados nessa dissertação. Esse conjunto de relacionamentos contém características que podem levar um par normativo a apresentar um conflito entre elas.

3.1 ESTRUTURA DA NORMA

A estrutura normativa utilizada nesse trabalho é baseada na estrutura apresentada em SILVA e ZAHN (2013). Em SILVA e ZAHN (2013) os autores utilizam somente ações como comportamentos possíveis a serem restringidos pela norma. Nesta dissertação, estendeu-se o comportamento para que fosse possível especificar também estados e assim possibilitar especificar normas que regulam o alcance de um estado. A inclusão de um identificador para a norma se fez necessário para o processo de verificação de conflito e para a visualização dos resultados, identificando as normas que estão em conflito.

Definição 1 (*Norma*). *A norma é uma tupla*

$$Norm = \{id, D, C, \{E \cup \{_\}\}, B, Cnd_1, Cnd_2\}$$

onde id define o identificador da norma; D define o conceito deôntico (permissão(pe), obrigação(ob), proibição(pr)); C é o contexto onde a norma se aplica, E é a entidade que deverá cumprir com a norma; B é o comportamento sendo regulado (i.g., uma ação), Cnd_1 e Cnd_2 são respectivamente a condição que ativa e desativa a norma.

A norma é definida no escopo de um contexto. A entidade, cujo o comportamento está sendo regulado, deve cumprir com a norma quando executar em um contexto onde a norma está sendo definida. Fora do seu contexto, uma norma não é válida. Será considerado, nesta dissertação, que uma norma pode ser definida no contexto de uma *organização*, isto é, um grupo de agentes $o \in O$ ou de um ambiente $env \in Env$. Assim, o conjunto de possíveis contextos são definidos como $C = O \cup Env$. Uma norma regula o comportamento de um agente $a \in A$, uma organização (ou grupo de agentes $o \in O$) ou um papel $r \in R$. Agentes, papéis e organizações são elementos do conjunto $E = A \cup R \cup O$. Um comportamento sendo regulado pela norma pode ser uma ação $act \in Act$ ou um estado $st \in St$. Assim, o conjunto de comportamentos é definido como $B = St \cup Act$.

A condição de ativação $acc \in Cnd$ e a condição de desativação $dac \in Cnd$ define a ocorrência de um evento tal como uma data, a execução de uma ação ou o cumprimento de uma norma. As Seções seguintes apresentam definições sobre a utilização desses elementos no modelo normativo proposto até aqui.

3.1.1 CONCEITO DEÔNTICO

A lógica deôntica refere-se a lógica de requisitos, comandos, regras, leis, princípios morais e sentenças (MEYER; WIERINGA, 1993). Em SMA, a lógica deôntica é utilizada para descrever as modalidades das normas que restringem o comportamento das entidades executando nestes sistemas:

- **Obrigação**: Define que uma entidade tem que executar uma determinada ação ou atingir um determinado estado especificado na norma. Não executar a ação ou não atingir o estado sendo obrigado implica em uma violação da norma.
- **Permissão**: Define que uma entidade pode executar determinada ação ou atingir um determinado estado especificado na norma. A entidade pode escolher executar ou não ação e pode escolher atingir ou não o estado. Nenhuma alternativa implica em violação da norma.
- **Proibição**: Define que uma entidade não pode executar determinada ação ou atingir um determinado estado especificado na norma. Executar a ação ou atingir o estado implicam em violação da norma.

Através do conceito deôntico, a entidade possui um referencial sobre a modalidade da norma (norma proibitiva, norma permissiva, norma obrigatória), sendo então, um elemento fundamental na estrutura da norma até aqui definida.

3.1.2 CONTEXTO

O contexto é o elemento da norma responsável por restringir seu escopo de atuação. O contexto indica o quão abrangente é a norma em relação ao ambiente e às organizações do SMA. A norma só está ativa no seu contexto de atuação, i.e., a entidade cujo comportamento está sendo regulado pela norma deve cumprir com a norma somente quando a mesma está executando no contexto de atuação da norma. Fora do contexto de atuação a norma não possui poder de reger comportamentos de entidades.

A norma pode ser especificada sobre dois tipos de contextos: *organização* e *ambiente* (SILVA; ZAHN, 2013). Se a norma é especificada sobre uma organização, significa que somente as entidades que desempenham papéis em tal organização deverão cumprir com a norma. Note que uma organização pode conter *sub-organizações* (SILVA et al., 2003). Se uma norma puder ser aplicada a uma organização que possua *sub-organizações*, todas as entidades executando no contexto das *sub-organizações* devem também cumprir com a norma.

Quando a norma é aplicada à um *ambiente*, então todas as entidades que habitam tal *ambiente* deverão cumpri-la. De acordo com SILVA et al. (2003), um ambiente pode ser habitado por organizações e agentes. Sendo assim, se uma norma for definida no contexto de um ambiente, todas as organizações que habitam o ambiente e todos os agentes que executam nas organizações deste ambiente devem cumprir a norma dada.

Se a norma não especifica um contexto, considera-se então que a norma se aplica a todos os contextos do domínio, i.e., a todos os ambientes e todas organizações.

3.1.3 ENTIDADE

A entidade é o elemento do modelo normativo que define a quem a norma está sendo aplicada. É esta entidade a responsável por cumprir com a norma. Conforme mencionado no Capítulo 2, a norma pode ser aplicada à um agente, um papel ou uma organização:

- Um **agente** é um sujeito autônomo, adaptativo e interativo (SILVA et al., 2003). Essas são características consideradas como fundamentais de um agente de software. Em HUHNS e SINGH (1998) e JENNINGS (2000) os autores acrescentam características ao agente, tais como o relacionamento com outros agentes, a habilidade de aprendizado e a mobilidade, formando então um conjunto de características que contemplam um agente de software.
- Um **papel** é definido no contexto de uma organização. Um papel é um elemento que guia e restringe o comportamento de um agente em uma organização,

garantindo aos agentes que desempenham um mesmo papel os mesmos direitos, deveres e oportunidades (VASCONCELOS et al., 2009). Em SILVA et al. (2003), duas grandes propriedades que definem um papel são apresentadas: (i) um papel deve ser definido no contexto de uma organização e (ii) um papel deve ser desempenhado por um agente ou uma sub-organização da organização onde o papel foi definido. Em um domínio, diversas entidades podem estar desempenhando um papel em um determinado momento. Uma entidade pode desempenhar diversos papéis simultaneamente.

- Uma **organização** é uma entidade que agrupa outras entidades que desempenham papéis. Uma organização define os papeis que devem ser desempenhados por um agente ou sub-organizações. Da perspectiva das entidades que estão fora de uma organização, a organização pode ser visualizada como um agente.

3.1.4 COMPORTAMENTO

O comportamento identifica a ação ou o estado que está sendo regulado pela norma. De acordo com FERREIRA et al., (2009), o comportamento é a maneira de agir, podendo ser uma resposta a um estímulo, bem como não ser correspondente a estímulo algum.

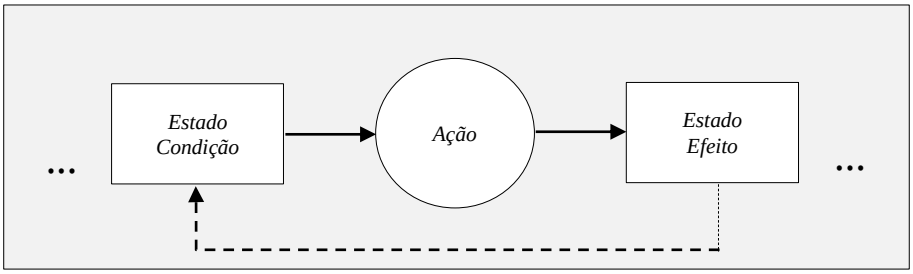

Figura 6: O ciclo dos estados e ações.

Uma **ação** é uma execução do agente que altera o estado do ambiente. A execução de uma ação gera o *estado efeito*, i.e., o estado resultante da execução da ação. Para que uma ação seja executada é necessário que o ambiente se encontre no *estado condição* dessa ação, possibilitando então que a mesma seja iniciada, conforme a Figura 6. Em outras palavras, ações possuem precondições associadas a elas, definindo as possíveis situações em que a ação pode ser executada pelo agente (WOOLDRIDGE, 2009). É importante ressaltar que um *estado efeito* pode ser um *estado condição* de outra ação, conforme é demonstrado na Figura através das linhas tracejadas (RUSSEL e NORVIG, 2010).

O **estado** de um agente define sua situação perante o ambiente em um determinado instante no tempo. De acordo com as definições apresentadas em RUSSELL (2003), cada estado é representado por uma coleção de variáveis. Ainda seguindo as afirmações dos autores, ações e estados estão intimamente ligados, pois um estado pode ser requisito para a execução de uma ação (*estado condição*), bem como o resultado ao fim da execução de uma determinada ação (*estado efeito*). Assim, um estado é consequência das modificações realizadas por ações, exceto o estado inicial do ambiente (RUSSEL e NORVIG, 2010).

3.1.5 ATIVAÇÃO E DESATIVAÇÃO

Uma norma deve ser cumprida dentro de um prazo que é definido nela mesma. Como mencionado no Capítulo 2, os eventos de ativação e desativação de uma norma podem ser: um determinado instante no tempo, o cumprimento de uma norma, a violação de uma norma, a execução de uma ação, o alcance de um estado, entre outros.

Nesse trabalho é considerado apenas o tempo como fator de ativação e desativação de uma norma. Assim, uma norma é ativa em uma determinada data (*dd/mm/aaaa*) e em uma determinada hora (*hh:mm:ss*). O mesmo se aplica para o evento de desativação da norma. Para que os demais eventos de ativação e desativação pudessem ser contemplados nesse trabalho, informações em tempo real do domínio da aplicação seriam necessários[6]. Devido a isso, optou-se apenas pela obtenção de informações do domínio não influenciadas pela dinâmica do sistema e descritas na ontologia.

Se a ativação não está especificada na norma, é considerado que a norma está ativa até o momento da sua desativação. Se a desativação não está especificada na norma, é considerado que a norma estará permanentemente ativa após a sua ativação. Se ambos os fatores de ativação e desativação não estiverem especificados, a norma estará sempre ativa.

3.2 ONTOLOGIA

A ontologia utilizada neste trabalho foi definida com base nos elementos existentes na Seção 3.1. É importante ressaltar que nosso principal objetivo ao definir a ontologia é saber se os elementos definidos na estrutura da norma estão relacionados entre si. O conhecimento sobre

[6] Por exemplo, a ativação de uma norma através de um evento de violação de outra só poderia ser afirmada se fosse possível observar um determinado sistema em tempo de execução.

estes relacionamentos é importante na verificação de conflitos normativos indiretos, como apresentado na Seção 3.3.

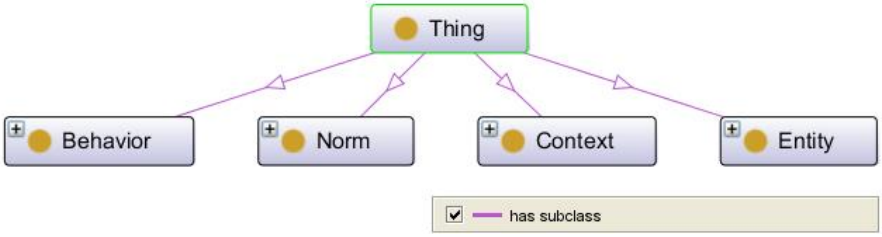

Figura 7: Classes base da Ontologia.

As principais entidades do domínio são: Contexto (*Context*), Entidade (*Entity*), Comportamento (*Behavior*) e Norma (*Norm*). Contexto, entidade e comportamento foram definidos como elementos principais, pois são os elementos definidos na tupla normativa. Não foi necessário definir conceito deôntico, identificador e os contextos de ativação e desativação como elementos principais pois são elementos muito simples, representados apenas por um texto (como propriedade da norma) e que não possuem relacionamentos com outros elementos.

Conforme a Figura 7, as classes base da ontologia são subclasses da classe *Thing*. Desse modo, *Thing* representa o conjunto de elementos contidos no domínio. Na Figura 8, a classe *Norm* é utilizada para instanciar o conjunto de normas aplicadas sobre o domínio. Ela se relaciona diretamente com as demais classes, descrevendo as restrições sobre cada elemento contido na ontologia.

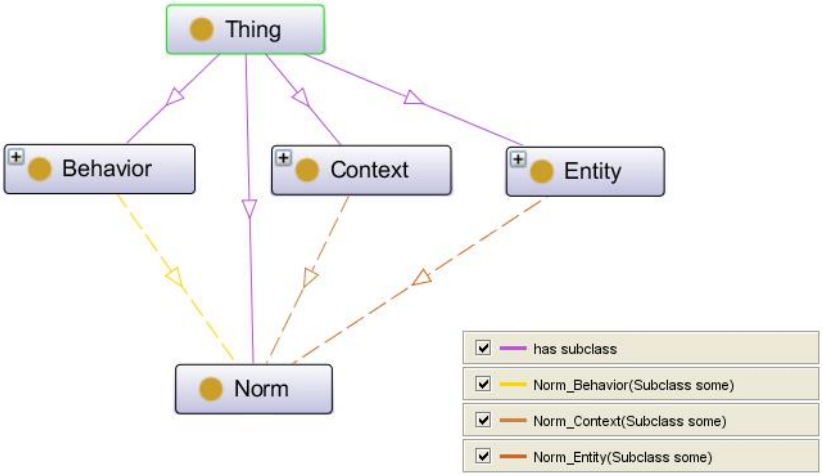

Figura 8: Classes base da Ontologia com referência normativa.

Como os elementos contidos nas normas estão diretamente ligados aos demais elementos contidos na ontologia, a classe *Norm* contém as classes *Entidade*, *Comportamento* e

Contexto, relacionamento que pode ser visualizado na ontologia através das linhas tracejadas da Figura 8.

3.2.1 CONTEXTO

O contexto, conforme já mencionado, é o elemento responsável por definir o escopo de atuação da norma. O contexto pode ser definido no escopo de *ambiente* ou *organização*. Na ontologia – Figura 9 – a classe *Context* (que representa o contexto) é uma classe abstrata que possui duas subclasses correspondentes aos tipos de *contextos* existentes no domínio – *Organization* (organização) e *Environment* (ambiente).

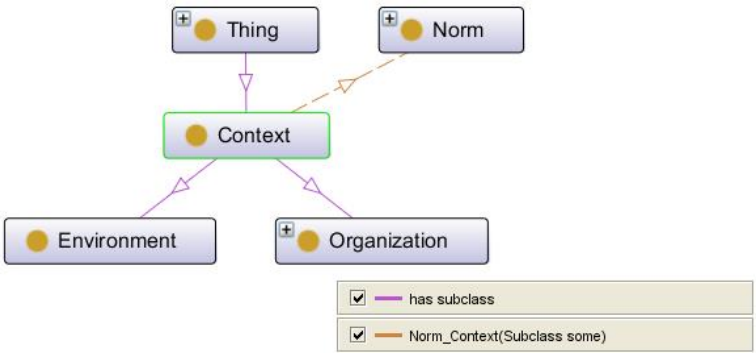

Figura 9: As classes *Context, Organization* e *Environment.*

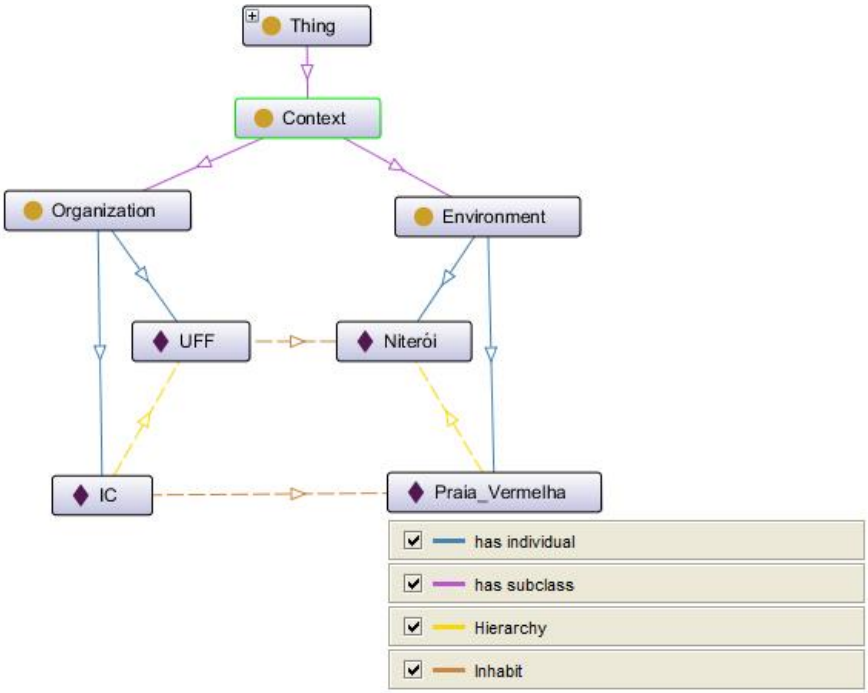

Figura 10: Instâncias da classe *Context* e seus relacionamentos.

A Figura 10 exemplifica o uso de contexto apresentando alguns contextos instanciados na ontologia, sendo eles: dois contextos do tipo *Environment* e dois contextos do tipo *Organization*. IC (Instituto de Computação) é uma *Organização* e também *sub-organização* da *Organização* UFF (Universidade Federal Fluminense), como definido pelo relacionamento H*ierarchy*. De maneira similar, o campus Praia Vermelha é um *Ambiente* e um *sub-ambiente* de Niterói, como definido pelo relacionamento *Hierarchy*. O relacionamento existente entre Niterói e UFF e, o relacionamento entre Praia Vermelha e IC é o *Inhabit*. Os relacionamentos *Hierarchy* e *Inhabit* serão definidos na Seção 3.3.1.

3.2.2 ENTIDADE

A entidade é o sujeito da norma, sendo ela o elemento ao qual a norma se aplica, i.e., é o elemento responsável por cumprir com a norma. Como mencionado na Seção 3.1.3, uma entidade pode ser um agente, um papel ou uma organização. Sendo assim, organização pode ser utilizada tanto para especificar o contexto de uma norma quanto a entidade a qual a norma se aplica. A Figura 11 apresenta a classe abstrata *Entity* (representando a entidade) que possui como subclasse as classes *Agent* (Agente), *Role* (Papel) e *Organization* (Organização).

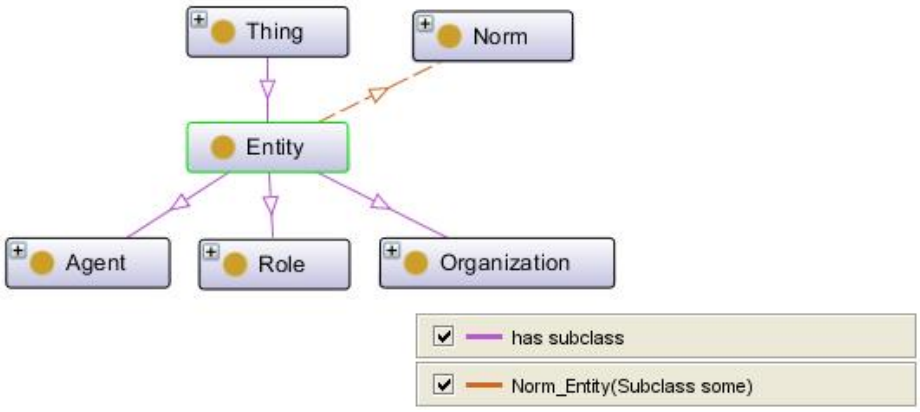

Figura 11: A classes *Entity*, *Organization*, *Role* e *Agent*.

A fim de exemplificar o uso das subclasses de *Entity*, a Figura 12 apresenta um conjunto de instâncias destas subclasses. Na Figura foram definidas duas Entity do tipo agente (*João* e *Maria*) e duas entidades do tipo papel (*Pesquisador* e *Estudante*). Dois relacionamentos que serão apresentados nas seções 3.3.2.1 e 3.3.2.3, *Play* e *Ownership* respectivamente, também estão modelados nesta Figura. O relacionamento *Play* indica que *João* desempenha o papel de *Pesquisador* e que *Maria* desempenha o papel de *Estudante* e o relacionamento *Ownership* indica que *Estudante* e *Pesquisador* são papéis definidos na organização *UFF*.

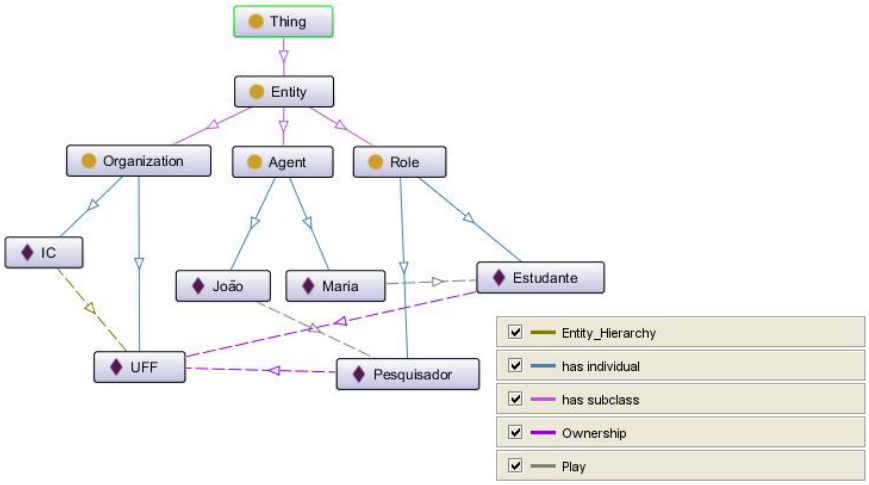

Figura 12: Instâncias da classe *Entity* e seus relacionamentos.

3.2.3 COMPORTAMENTO

A ontologia descreve também os comportamentos que são regulados pelas normas. Conforme já mencionado, o comportamento pode ser uma ação ou um estado. Assim, uma norma pode restringir a execução de uma ação ou o alcance de um estado.

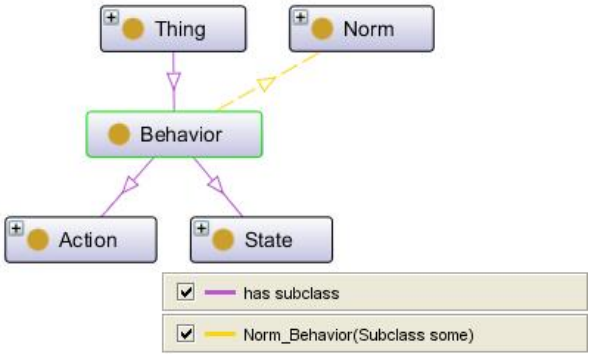

Figura 13: A classe *Behavior*, *Action* e *State*.

A classe *Behavior* (Comportamento) é abstrata e possui duas subclasses, sendo elas a *Action* (Ação) e a *State* (Estado) – Figura 13. Para exemplificar como as instâncias são alocadas e como se relacionam, a Figura 14 apresenta algumas ações e estados presentes no domínio do sistema, juntamente com os possíveis relacionamentos entre ações e estados, que serão apresentados detalhadamente na Seção 3.3. Na ontologia, é possível visualizar o relacionamento entre estados e ações devido à dependência expressada entre as instâncias. Na Figura 14, estão representadas três ações (*Parar*, *Mover* e *Caminhar*) e três estados (*Parado*, *Movendo* e *Caminhado*). No exemplo, a ação *Mover* é dependente do estado *Parado* para que possa entrar em execução. A ação *Mover* possui como efeito o estado *Movendo*. A ação *Caminhar* possui como efeito o estado *Caminhando*. Já a ação *Parar* possui como efeito o

estado *Parado*. Note que existe um relacionamento entre as ações *Caminhar* e *Mover* que é o relacionamento *Refinement*, que será definido na Seção 3.3.3.1

O exemplo da Figura 14 demonstra o modo como os relacionamentos entre estados e ações, i.e., condição da ação (*condition*) e pós-condição da ação (*effect*), são definidos conforme descrito na Seção 3.1.4.

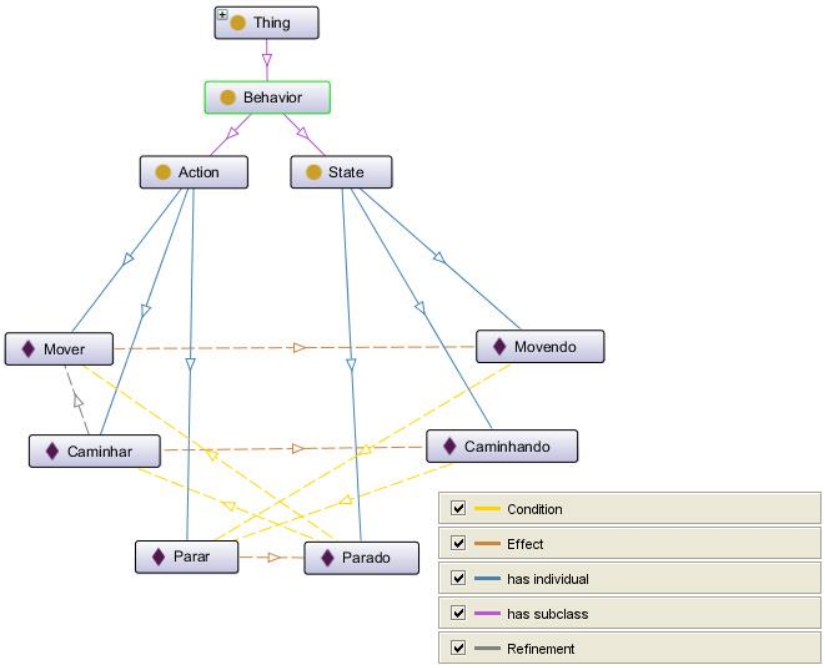

Figura 14: As instâncias da classe *Behavior* e seus relacionamentos.

3.2.4 NORMA

A norma tem como função reger o comportamento das entidades envolvidas no domínio. A ontologia representa a norma especificando diretamente as classes pertencentes ao domínio e que são elementos presentes na norma. A Figura 15 apresenta como a norma se encaixa na estrutura do sistema proposto.

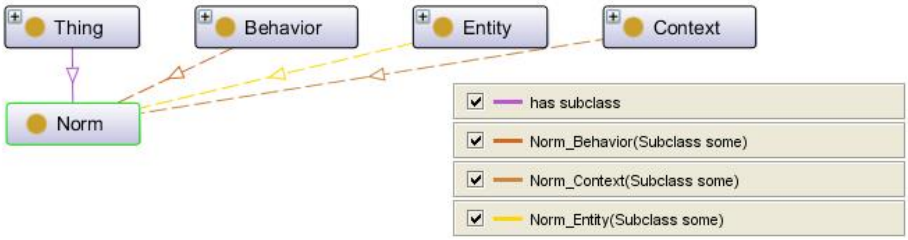

Figura 15: A classe *Norm* e suas subclasses.

Como visto na Figura, uma norma é aplicada sobre uma *Entidade*, sobre um *Contexto* e sobre um *Comportamento*. A norma contém informações em sua estrutura que definem o

conceito deôntico aplicado sobre ela (*proibição*, *permissão* ou *obrigação*), seu identificador e seu período de ativação e desativação.

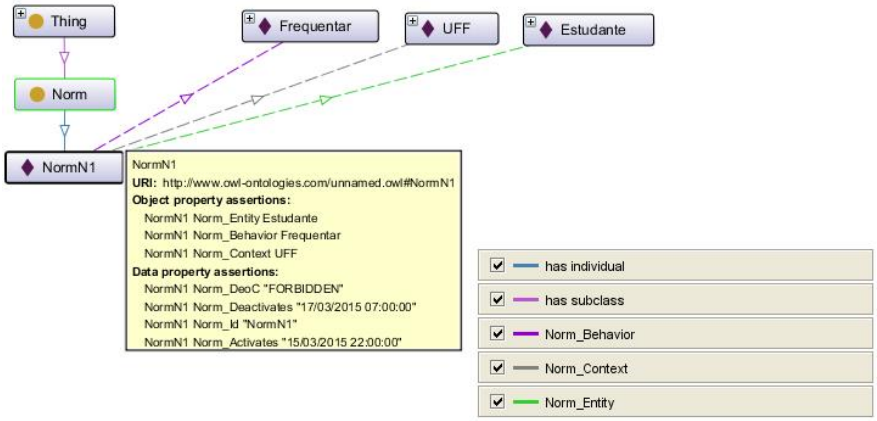

Figura 16: NormaN1 aplicada sobre o domínio.

A Figura 16 apresenta um pequeno exemplo de uma norma aplicada sobre os elementos do domínio. A *NormaN1* é aplicada sobre o papel *Estudante*, *proibindo* (*forbidden*) qualquer entidade que desempenha tal papel de *Frequentar* a UFF no período de tempo estipulado pela norma. Note que a norma é aplicada sobre o contexto da Universidade Federal Fluminense (*UFF*), definindo então que a entidade só deve cumprir com a norma quando a mesma estiver presente em tal contexto. O intervalo de tempo (*15/03/2015 22:00:00 – 17/03/2015 07:00:00*), o identificador da norma (*NormN1*) e o conceito deôntico (*Forbidden*) são informações contidas nas propriedades da norma, conforme é possível visualizar através da caixa de diálogo – Figura 16.

Utilizando a estrutura normativa apresentada na Seção 3.1, a norma apresentada na Figura 16 pode ser descrita conforme abaixo:

$$n = \{NormN1, forbidden, UFF, Student, ToAttend, 15/03/2015\ 22{:}00{:}00, 17/03/2015\ 07{:}00{:}00\}$$

3.3 RELACIONAMENTOS ENTRE OS ELEMENTOS DA NORMA

Os elementos normativos – contexto, entidade e comportamento – podem, quando diferentes, estar relacionados entre eles. O restante desse Capítulo apresenta o conjunto de relacionamentos existentes entre os elementos que compõem uma norma. Associado às definições desses relacionamentos, é apresentada a regra da propagação da norma.

3.3.1 CONTEXTO

Nesta Seção, são apresentados os relacionamentos entre os contextos considerados nesta dissertação. Existem dois tipos distintos de relacionamentos entre contextos: **Hierarchy** e

Inhabit. Associada com cada relacionamento, é definida a Regra de Propagação (RP) da norma, que define como uma norma que define determinado contexto é propagada para as entidades ou contextos relacionados com tal contexto.

3.3.1.1 HIERARCHY ENTRE CONTEXTOS

O relacionamento **Hierarchy** (*hierarquia*) entre contextos é definido com base na relação entre *supercontexto* e *subcontexto*. De acordo com FERREIRA et al. (2009), a hierarquia é uma ordem que visa priorizar um membro, poderes, categorias, entre outros. Além disso, é uma classificação que leva em consideração a relação entre superiores e dependentes.

Definição 2 *(Hierarchy entre contextos). O relacionamento Hierarchy entre contextos relaciona uma organização (chamada de superorganização) com outra organização (chamada de suborganização).*

$$\textit{Hierarchy de Contextos} \subseteq \textit{Organização} \times \textit{Organização}$$

Se uma norma é definida no escopo de uma organização e tal norma não especifica a entidade cujo o comportamento deve ser regulado, isso significa que tal norma se aplica a todas as *sub-organizações* de tal organização, conforme formalizado na Definição 3.

Definição 3 (*Propagação de Hierarchy entre contextos*). *Se a norma é definida no contexto de uma organização e não menciona uma entidade específica, tal norma se aplica a todas as sub-organizações de uma determinada organização.*

$$\exists o_1(\langle d, o_1, _, b, acc, dac\rangle \in Norm \rightarrow \forall o_2(hierarchy(o_1, o_2) \rightarrow \langle d, o_1, o_2, b, acc, dac\rangle \in Norm))$$

onde $d \in D, env \in Env, o_1 \in O, o_2 \in O, b \in B, acc \in Cnd, dac \in Cnd$.

Na Figura 17, IC é uma *sub-organização* de UFF. Desse modo, se uma norma é aplicada sobre a UFF, a norma se aplica também ao IC (e a todas as *sub-organizações* de IC, se for o caso), i.e., as entidades que desempenham papeis em IC (e em todas as *sub-organizações* de IC) deverão cumprir com a norma. No entanto, se a norma se aplica a IC, a norma não se aplica necessariamente a UFF. O relacionamento **Hierarchy** é unidirecional.

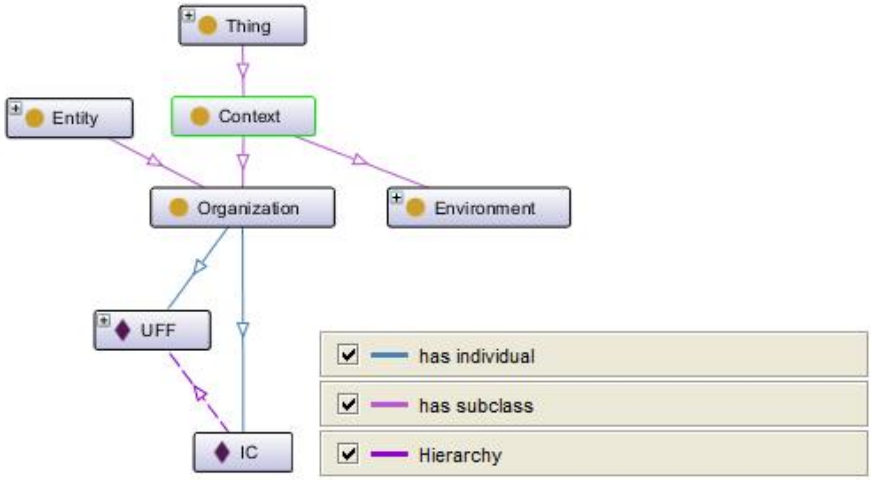

Figura 17: Exemplo de relacionamento *Hierarchy* entre Contextos.

3.3.1.2 INHABIT

O relacionamento **Inhabit** relaciona uma entidade com o ambiente onde ela habita. As entidades que executam em um contexto do tipo *ambiente* são *agentes* e *organizações*. Se uma entidade do tipo *organização* ou do tipo *agente* está contida em um contexto do tipo *ambiente*, isso significa que a *organização* habita o *ambiente* e que o relacionamento existente entre eles é o **Inhabit**.

Definição 4 (*Inhabit*). *O relacionamento Inhabit relaciona uma entidade com o ambiente no qual habita. As entidades que executam no contexto de um ambiente são agentes e/ou organizações.*

$$Inhabit \subseteq Ambiente \times (Agente \cup Organização)$$

Definição 5 (Propagação de *Inhabit*). *Se a norma é definida unicamente no contexto de um ambiente e não identifica uma entidade específica, isso significa que tal norma se aplica sobre todas as entidades que habitam o determinado ambiente.*

$$\exists env(\langle d, env, _, b, acc, dac\rangle \in Norm \rightarrow \forall e(inhabit(env, e) \rightarrow \langle d, env, e, b, acc, dac\rangle \in Norm))$$

onde $d \in D, env \in Env, e \in A \cup O, b \in B, acc \in Cnd, dac \in Cnd$.

Quando uma norma se aplica exclusivamente sobre um contexto do tipo *ambiente*, não especificando uma entidade, tal norma se aplica a todas as entidades (*agentes* e *organizações*) que *habitam* esse ambiente (Definição 5).

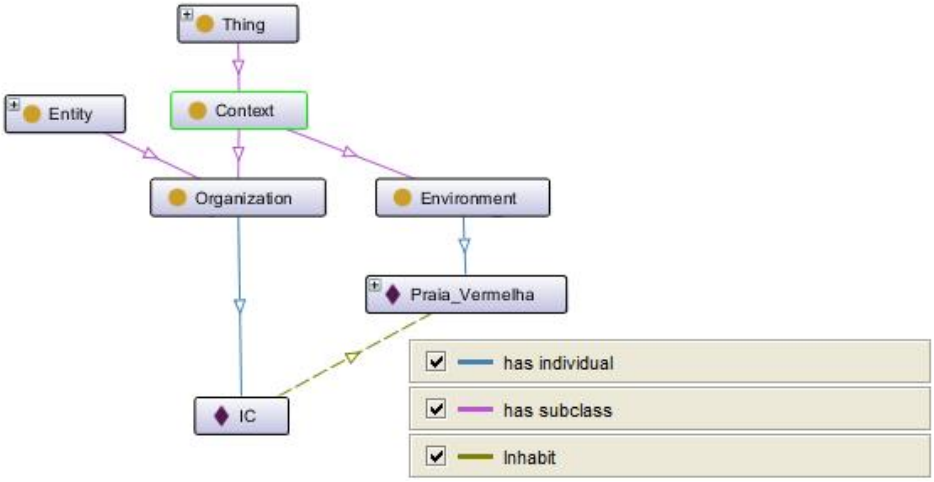

Figura 18: Exemplo do relacionamento *Inhabit* entre Contextos.

Na Figura 18 é apresentado o relacionamento **Inhabit** entre o contexto *organização* IC e o contexto *ambiente* Praia Vermelha. Note que a *organização* IC é um contexto, e também uma entidade do tipo *organização*.

3.3.2 ENTIDADE

Conforme mencionado no Capítulo 2, a norma pode ser aplicada sobre um agente, um papel ou um grupo de agentes (*organização*). Nesta Seção são formalizados os relacionamentos existentes entre as entidades. Seguindo as definições expostas em FIGUEIREDO (2011), três relacionamentos entre entidades devem ser considerados: **Play**, **Hierarchy** e **Ownership**.

3.3.2.1 PLAY

O relacionamento **Play** relaciona uma entidade e os papéis que tal entidade desempenha em uma determinada organização. As seguintes entidades podem desempenhar papéis: *agente* e *organização*.

Definição 6 *(Play). O relacionamento Play relaciona um agente ou sub-organização com os papeis que podem ser desempenhados no contexto de uma determinada organização.*

$$Play \subseteq Organização \times (Agente \cup Organização) \times Papel$$

Quando uma norma é aplicada a um papel, tal norma rege o comportamento dos agentes ou organizações que desempenham tal papel no contexto onde a norma foi definida, conforme afirmado na Definição 7.

Definição 7 *(Propagação de Play). Se a norma rege um papel em uma determinada organização, tal norma rege o comportamento de todos os agentes e sub-organizações que desempenham tal papel no contexto onde a norma está definida.*

$\exists r(\langle d, o, r, b, acc, dac\rangle \in Norm) \rightarrow \forall e(play(o, e, r) \rightarrow \langle d, o, e, b, acc, dac\rangle \in Norm))$
onde $d \in D, o \in O, r \in R, b \in B, e \in A \cup O, R_1 \subseteq R, acc \in Cnd, dac \in Cnd.$

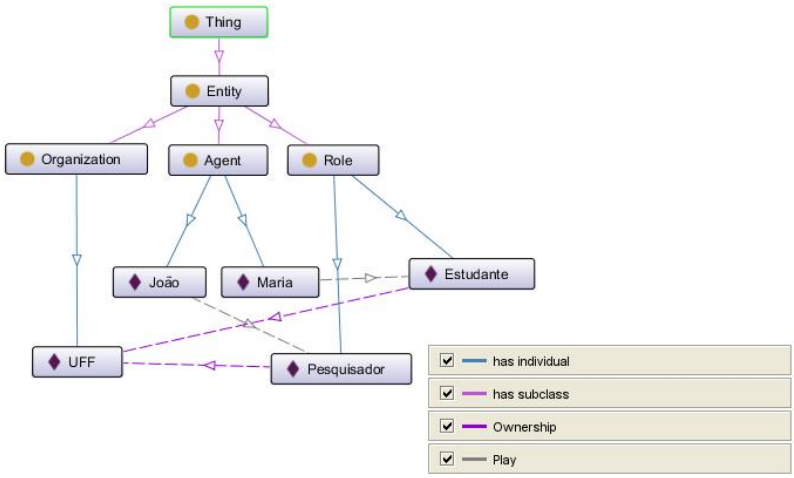

Figura 19: Exemplo de relacionamento *Play*.

Para exemplificar o relacionamento **Play** entre entidades, a Figura 19 apresenta o papel *Estudante* e o papel *Pesquisador*, o agente *Maria* e o agente *João* e o relacionamento **Play** entre *Maria* e *Estudante* e o relacionamento **Play** entre *João* e *Pesquisador*. Isso significa que *João* e *Maria* desempenham, respectivamente, os papéis *Pesquisador* e *Estudante*.

3.3.2.2 HIERARCHY ENTRE ENTIDADES

O relacionamento **Hierarchy** define uma hierarquia entre as entidades. Conforme já mencionado na Seção 3.3.1.1, a hierarquia é uma ordem que visa priorizar um membro, poderes, categorias, entre outros. Diferentemente das definições apresentadas nos relacionamentos entre contextos, nessa Seção é definida a hierarquia entre entidades.

Definição 8 *(Hierarchy entre entidades). O relacionamento Hierarchy entre entidades relaciona um papel (chamado de superentidade) com outro papel (chamado de subentidade).*

$$Hierarchy\ de\ Entidades \subseteq Papel \times Papel$$

Quando uma norma se aplica a uma entidade definida na ontologia como um *super-papel*, significa que todo e qualquer *sub-papel* presente nesse relacionamento deverá cumprir com a norma (caso um *sub-papel* específico não seja mencionado). O mesmo não ocorre no

caso de uma norma ser aplicada sobre um *sub-papel*, pois a norma não se propaga de uma *sub* para um *super-papel* (Definição 9).

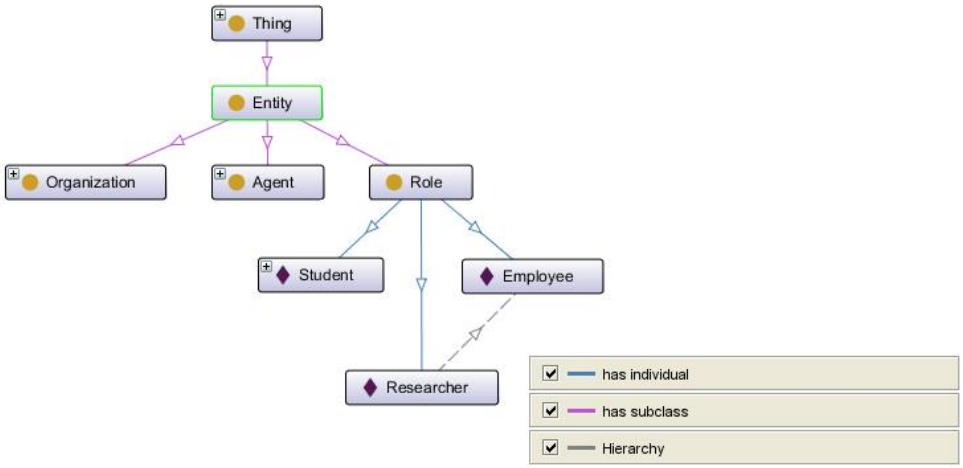

Figura 20: Exemplo do relacionamento *Hierarchy* entre entidades.

Definição 9 *(Propagação de Hierarchy entre entidades). Se a norma que rege o comportamento de um super-papel não especifica um sub-papel correspondente, significa que a norma se aplica sobre todos os sub-papéis do super-papel em questão. Os sub-papéis herdam as restrições aplicadas aos super-papéis, respeitando a hierarquia existente entre eles.*

$$\exists o_1(\langle d, o_1, _, b, acc, dac\rangle \in Norm \rightarrow \forall o_2(hierarchy(o_1, o_2) \rightarrow \langle d, o_1, o_2, b, acc, dac\rangle \in Norm))$$

onde $d \in D, env \in Env, o_1 \in O, o_2 \in O, b \in B, acc \in Cn, dac \in Cn$.

A Figura 20 apresenta o relacionamento **Hierarchy** entre os papéis *Empregado* e *Pesquisador*. Isso significa que se a norma é aplicada sobre *Empregado*, ela se aplica a todos os *sub-papéis* correspondentes. Neste caso, a norma se propaga ao papel *Pesquisador*.

3.3.2.3 OWNERSHIP

O relacionamento **Ownership** relaciona um *papel* e a *organização* onde tal *papel* é desempenhado.

Definição 10 *(Ownership). O relacionamento de Ownership relaciona um papel com a organização onde o papel está definido.*

$$Ownership \subseteq Organização \times Papel$$

Se uma norma se aplica sobre uma *organização* e não especifica um *papel* específico, significa que todos os papéis desempenhados em tal *organização* deverão cumprir com a norma.

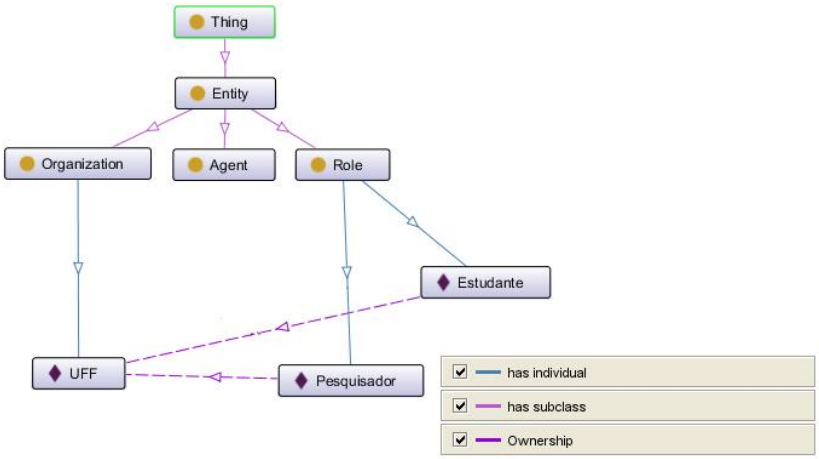

Figura 21: Exemplo de relacionamento *Ownership*.

Para exemplificar (Figura 21), considere a existência de uma norma aplicada sobre a entidade *UFF* (*organização*). Sabendo que *Estudante* é um papel desempenhado na organização *UFF*, a norma aplicada a *UFF* também se aplica a *Estudante*.

Definição 11 *(Propagação de Ownership). Se a norma é aplicada para todas as entidades em uma organização, tal norma deve ser aplicada sobre todos os papeis desempenhados em tal organização.*

$\exists o(\langle d, o, _, b, acc, dac\rangle \in Norm \rightarrow \forall r(ownership(o, r) \rightarrow \langle d, o, r, b, acc, dac\rangle \in Norm))$
onde $d \in D, o \in O, r \in R, b \in B, acc \in Cnd, and\ dac \in Cnd$.

3.3.3 COMPORTAMENTO

Como já mencionado nesse trabalho, o comportamento é o elemento responsável por identificar o que a norma está regulando, ou seja, quais os comportamentos que a norma está permitindo, obrigando ou proibindo. Esse trabalho possui quatro tipos de relacionamentos entre comportamentos: **Refinement**, **Dependency**, **Orthogonality** e **Composition**. É importante ressaltar que novos tipos de relacionamentos entre comportamentos podem surgir e que a atual abordagem pode ser estendida.

A verificação de relacionamento entre os comportamentos deverá ocorrer entre *estados* x *estados*, *estados* x *ações* e *ações* x *ações.* Isso é necessário, devido que o ciclo de uma ação (*estado condição*, *ação*, *estado efeito*) deve ser levado em consideração – conforme já mencionado na Seção 3.1.4.

3.3.3.1 REFINEMENT ENTRE COMPORTAMENTOS

O Refinement é um tipo de relacionamento que existe entre um comportamento *genérico* (*super estado* ou *super ação*) e outro comportamento *específico* (*sub estado* ou *sub ação*, que refina o comportamento genérico). A execução da *sub ação* atinge os estados alcançados pela execução da *super ação*, bem como pode atinge outros estados. O alcance de um *sub estado* atinge um *super estado*. Se existe mais de uma *sub ação* para uma determinada *super ação*, a execução de qualquer uma das respectivas *sub ações* atingem o estado da execução da *super ação*. Semelhantemente, se existe um mais de um *sub estado* para um determinado *super estado*, o alcance de qualquer *sub estado* atinge o mesmo estado do *super estado*.

Definição 12 *(Refinement). O relacionamento de Refinement relaciona um comportamento genérico (super ação ou super estado) com um comportamento específico (sub ação ou sub estado).*

$$Refinement \subseteq Comportamento\ Genérico \times Comportamento\ Específico$$

Para auxiliar na compreensão do refinamento entre comportamentos, vamos utilizar as seguintes ações que possuem esse tipo de relacionamento definido entre elas, conforme a Figura 22.

No exemplo, é definido que as ações *caminhar* e *dirigir* são ações específicas da ação *mover*. Uma ação *genérica* atinge um estado *genérico*. Do mesmo modo, uma ação *parte* atinge um estado *parte*.

O restante dessa Seção apresenta as definições para os casos de conflitos entre ações e estados que possuem como relacionamento **Refinement**. Para isso, uma análise da ação sobre um conceito deôntico será feita, bem como do estado, realizando a comparação com uma possível segunda norma, afirmando (ou não) a existência de um conflito. As análises e regras de propagação ocorrerão sob a óptica de uma *super ação* e um *super estado*.

Obrigação/Permissão: Quando a norma obriga (ou permite) um *super comportamento* ser executado ou atingido, significa que a entidade cujo o comportamento está sendo regulado deverá (ou poderá) executar ou atingir tal comportamento.

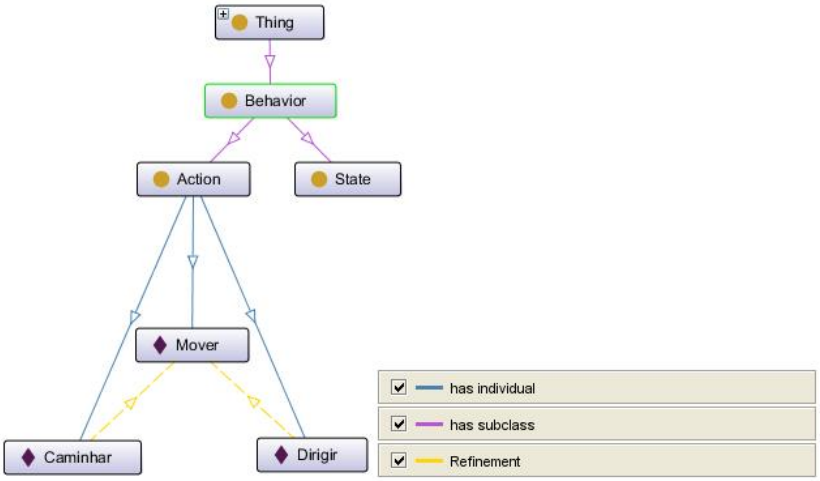

Figura 22: A ação Mover e o relacionamento de *Refinement*.

Definição 13 *(Regra de conflito para permissão e obrigação no relacionamento de Refinement). Se existe uma obrigação ou permissão aplicada sobre um super comportamento e proibições aplicadas sobre todos os sub comportamentos, tais normas estarão em conflito.*

$$\exists b_1(\langle d, c, e, b_1, acc_1, dac_1\rangle \in Norm \rightarrow \forall b_i(refinement(b_1, b_i) \rightarrow \langle pr, c, e, b_i, acc_2, dac_2\rangle \in Norm \rightarrow conflict(\langle d, c, e, b_1, acc_1, dac_1\rangle, \langle pr, c, e, b_i, acc_2, dac_2\rangle)))$$

$$onde\ \ b_i \in \{b \mid refinement(\ b_1, b_i)\}, d \in (ob, pe), c\ \in C, e \in E, b_1 e\ b_2 \in B, acc_i\ and\ dac_i \in Cnd,\ \ acc_1 \geq dac_2,\ \text{e}\ dac_1 \geq acc_2.$$

Para exemplificar a afirmação contida na Definição 13, vamos analisar a ontologia da Figura 22. Se a norma obriga (ou permite) a entidade a *mover* e, sabendo que *caminhar* e *dirigir* são ações específicas da ação *mover*, significa que se a entidade for proibida de *caminhar* e *dirigir*, um conflito ocorrerá entre as normas. Note que aqui estamos considerando o período em que a norma está ativa, considerando que as entidades são as mesmas (ou relacionadas) e que os contextos são os mesmos (ou relacionados).

Proibição: Quando a norma proíbe a execução ou o alcance do *super estado*, significa que o comportamento não pode ser executado ou atingido pela entidade que está sendo regulada pela norma. Isso deverá ser cumprido no período de tempo em que a norma está estipulada.

Definição 14 *(Regra de conflito para proibição no relacionamento de Refinement). Se existe uma proibição aplicada sobre um super comportamento e uma obrigação (ou permissão) aplicada sobre uma sub comportamento, tais normas estarão em conflito.*

$\exists b_1 \exists b_2((\langle pr, c, e, b_1, acc_1, dac_1\rangle \in Norm \wedge \langle d, c, e, b_2, acc_2, dac_2\rangle \in Norm \wedge refinement(b_1, b_2)) \rightarrow conflict(\langle pr, c, e, b_1, acc_1, dac_1\rangle, \langle d, c, e, b_2, acc_2, dac_2\rangle))$
onde $d \in \{ob, pe\}, c \in C, e \in E, b_1 e\ b_2 \in B, acc_i\ e\ dac_i \in Cnd,\ acc_1 \geq dac_2,$ e $dac_1 \geq acc_2$.

Vamos utilizar o exemplo e assumir a existência de uma norma proibindo uma entidade de se *mover*. Se tal entidade *caminhar* ou *andar*, a entidade cuja a norma está aplicada sobre ela, violará a norma proibitiva. Por outro lado, se a entidade não executar nenhuma das *sub ações* (no período de tempo em que a norma está ativa), nenhuma norma será violada, resultando no cumprimento da norma.

3.3.3.2 DEPENDENCY ENTRE COMPORTAMENTOS

A dependência entre comportamentos é um processo natural que ocorre em um ciclo de ações. Conforme mencionado no Capítulo 2, uma ação possui um estado (ou um conjunto de estados) **condição** e um estado (ou um conjunto de estados) **efeito** que é atingido ao fim da execução da ação. Uma ação pode ser executada somente se sua **condição** for alcançada, existindo então uma dependência entre a ação e a sua **condição**. O **efeito** é alcançado através da execução da ação, existindo então uma relação de dependência entre o **efeito** e a ação. Desse modo, o **efeito** somente é alcançado se a **condição** também for alcançada (e se a ação entre esses estados for executada).

Definição 15 *(Dependency). O relacionamento Dependency relaciona um comportamento (denominado comportamento cliente) com outro comportamento (denominado comportamento dependente).*

$$Dependency \subseteq Cliente \times Dependente$$

A fim de exemplificar o relacionamento de dependência entre os comportamentos ao longo da apresentação dos casos de conflitos, vamos considerar o exemplo do transporte aéreo de carga (Figura 23). Um avião, ao realizar um transporte de carga, necessita que a carga esteja em seu compartimento e que o mesmo esteja abastecido. Assim, algumas ações são necessárias para que o avião possa cumprir com seu dever (WOOLDRIDGE, 2009).

Conforme descrito na ontologia, o estado *Vazio* é um estado *cliente* para a ação *dependente Carregar*. Isso significa que, para a ação *Carregar* ser executada, o estado *Vazio* deve ser alcançado. O mesmo acontece entre o estado *Carregado*, que necessita do estado *Vazio* para que o mesmo possa ser atingido. Obviamente, essa dependência possui uma ação mediadora, que está definida na segunda tupla da ontologia, sendo então o estado *Carregado*

dependente da ação *Carregar*, pois tal ação é a responsável para que o estado seja alcançado através do seu **efeito**.

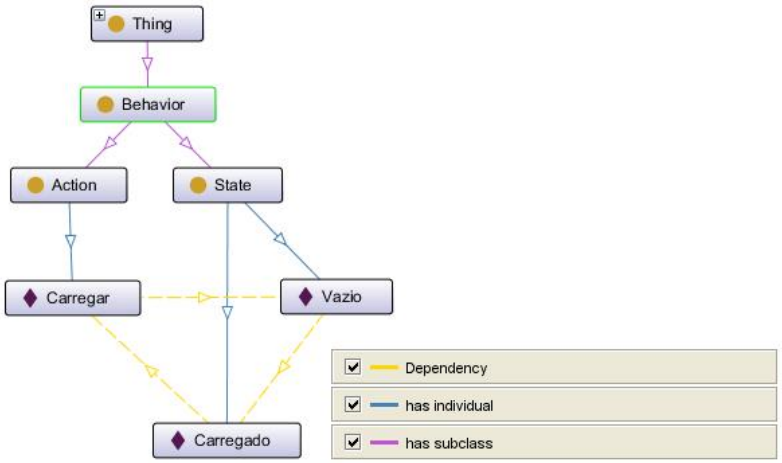

Figura 23: *Dependecy* definido entre ações e estados.

As regras de conflito do relacionamento *Dependendency* são aplicadas sobre dois pontos de vista distintos: o estado dependente (1) e a ação dependente (2).

Obrigação/Permissão:

(1) Quando a norma obriga (ou permite) o alcance de um estado, significa que a entidade cujo o comportamento está sendo regulado deve (ou pode) executar a ação cujo o **efeito** é tal estado.

Se existe somente uma ação cujo o **efeito** é o estado que deve (ou pode) ser alcançado, uma norma proibindo a execução da ação não deve ser aplicada. Se a execução da ação é proibida, significa que o estado que está sendo obrigatório (ou permitida) não poderá ser alcançado, devido a ação ser a única ação que possui o poder de levar a entidade a atingir tal estado. Assim, não existe a possibilidade do alcance do estado sem a execução da ação.

Definição 16 *(Regra de conflito para permissão e obrigação aplicada sobre um estado dependente no relacionamento Dependency). Se existe uma obrigação ou permissão aplicada sobre um estado dependente e proibições aplicadas as ações clientes do respectivo estado, tais normas estarão em conflito.*

$$\exists act \exists st((\langle pr, c, e, act, acc_1, dac_1\rangle \in Norm \wedge \langle d, c, e, st, acc_2, dac_2\rangle \in Norm \wedge dependency(act, st)) \rightarrow conflict(\langle pr, c, e, act, acc_1, dac_1\rangle, \langle d, c, e, st, acc_2, dac_2\rangle))$$

where $d \in \{ob, pe\}, c \in C, e \in E, act\ and\ st \in B, acc_i\ and\ dac_i \in Cnd,\ acc_1 > dac_2,$ and $dac_1 \geq acc_2$.

(2) Quando uma norma obriga (ou permite) a execução de uma ação, significa que a **condição** de tal ação deve ser alcançada para que a entidade seja capaz de executar a ação. Por exemplo, para que a ação *Carregar* seja executada, é necessário que o estado *Vazio* seja alcançado. A ação *Carregar* é dependente do estado *Vazio*. Assim, se existe uma norma proibindo o alcance da **condição**, essa norma estará em conflito com a norma que obriga a execução da ação. Em qualquer outro caso, o conflito não existe.

Definição 17 *(Regra de conflito para permissão e obrigação aplicada sobre uma ação relacionamento Dependency). Se existe uma obrigação ou permissão aplicada sobre uma ação, significa que nenhuma proibições não podem ser aplicadas sobre os estados condição dessa ação.*

$$\exists st \exists act((\langle pr, c, e, st, acc_1, dac_1\rangle \in Norm \wedge \langle d, c, e, act, acc_2, dac_2\rangle \in Norm \wedge dependency(st, act)) \rightarrow conflict(\langle pr, c, e, st, acc_1, dac_1\rangle, \langle d, c, e, act, acc_2, dac_2\rangle))$$

onde $d \in \{ob, pe\}, c \in C, e \in E, st\ and\ act \in B, acc_i\ and\ dac_i \in Cnd,\ acc_1 > dac_2,$ and $dac_1 \geq acc_2$.

Proibição:

(1) Quando a norma está proibindo o alcance do estado, significa que a entidade cujo o comportamento está sendo regulado não deve executar ações cujo o **efeito** é tal estado.

Se existe uma ação cujo o **efeito** é o estado que está sendo proibido, nenhuma norma permitindo ou obrigando a execução de tal ação deve existir. Para exemplificar, se existe uma norma proibindo uma entidade de alcançar o estado *Carregado*, o conflito surge quando outra norma permite (ou obriga) a execução de uma ação que possui como efeito tal estado.

Definição 18 *(Regra de conflito para proibição no relacionamento de Dependency). Se existe uma proibição aplicada sobre um estado dependente e obrigações ou permissões aplicadas as ações clientes do respectivo estado, tais normas estarão em conflito.*

$$\exists act \exists st((\langle d, c, e, act, acc_1, dac_1\rangle \in Norm \wedge \langle pr, c, e, st, acc_2, dac_2\rangle \in Norm \wedge dependency(act, st)) \rightarrow conflict(\langle d, c, e, act, acc_1, dac_1\rangle, \langle pr, c, e, st, acc_2, dac_2\rangle))$$

where $d \in \{ob, pe\}, c \in C, e \in E, act\ and\ st \in B, acc_i\ and\ dac_i \in Cnd,\ acc_1 > dac_2,$ and $dac_1 \geq acc_2$.

(2) Quando uma norma está proibindo a execução de uma ação, significa que o estado **efeito** de tal norma não será atingido pela entidade (considerando que a entidade cumpra com a norma). Se existe uma permissão ou uma obrigação aplicada sobre tal **efeito** e, não existindo

nenhuma outra ação que possua o mesmo **efeito**, as normas entrarão em conflito. Por outro lado, se existe outras ações que possuem o mesmo **efeito**, a entidade pode executar tais outras ações para atingir o estado. Assim, para que um conflito seja caracterizado, todas as outras ações com o mesmo **efeito** devem ser proibidas. Esse caso é o mesmo ocorrido na Definição 16, não havendo então, necessidade definir uma nova regra para tal caso.

3.3.3.3 COMPOSITION ENTRE COMPORTAMENTOS

A relacionamento **Composition** relaciona um comportamento chamado *todo* e comportamentos denominados *parte*. O comportamento *todo* é um comportamento que só é alcançado se todos os comportamentos *parte* também forem atingidos.

Definição 19 *(Composition). O relacionamento Composition relaciona um comportamento (denominado inteiro) com outro comportamento (denominada parte).*

$$Composition \subseteq Comportamento\ todo \times Comportamento\ parte$$

Para contribuir com o entendimento das regras de conflito aqui levantadas, vamos considerar, como exemplo, um cenário de transporte aéreo de carga. Nesse cenário, existe um avião que pode voar de um lugar para o outro apenas se o mesmo estiver carregado com a carga e quando o mesmo está *Abastecido*. Assim, podemos definir alguns estados nesse cenário, como o *Pronto para Decolar* sendo um estado todo, formado pelos estados *Carregado* e *Abastecido*, Figura 24.

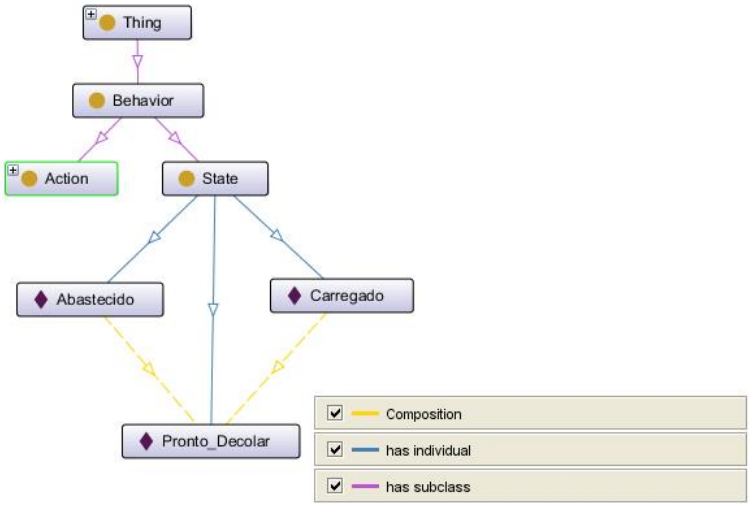

Figura 24: *Composition* definido entre estados.

Conforme definido na ontologia acima, os estados *Carregado* e *Abastecido*, quando atingidos, alcançam automaticamente o estado *todo Pronto para Decolar*. Para descrever os

casos de conflito no relacionamento *Composition* entre comportamentos, visualizaremos a partir do estado *todo* e, assim, os casos de conflitos serão estendidos para os demais tipos de comportamentos contidos nas outras normas, combinando o estado *todo* de uma norma com possíveis ações, estados *parte*, estados como condição e estados como efeito.

Obrigação/Permissão: Quando uma norma obriga (ou permite) a entidade de executar um comportamento *todo*, significa que todos os seus comportamentos *parte* devem ser executados para que a entidade possa cumprir com a norma obrigatória (ou permissiva).

Os problemas relacionados com obrigações e permissões aplicadas sobre o estados *integrais* e normas aplicadas sobre ações também devem ser considerados. Se existe uma norma que proíbe a execução de uma ação cujo **efeito** é um estado *todo*, (ou um estado *parte*), tal norma entrará em conflito com a norma que obriga (ou permite) o estado *todo* ser alcançado. No entanto, esse conflito ocorre apenas se tal ação for a única ação cujo o efeito é o alcance do estado obrigatório. Se existe outras ações cujo o efeito é o estado *todo* ou algum dos estados *parte*, essas ações devem ser verificadas a fim de reconhecer se todas também estão sendo proibidas. Se a ação cujo o **efeito** é o estado obrigatório não é executada pela entidade, tal entidade não conseguirá atingir o estado *todo*, gerando então uma violação da norma. Por outro lado, se existem outras ações cujo o efeito é o estado *todo* e que podem ser alcançadas, logo o estado *todo* ou todos os estados *parte* serão alcançados e a norma não será violada. Para exemplificar, vamos considerar que exista apenas uma ação que seja capaz de *Carregar* o avião. Assim, se existe uma norma obrigando a entidade a alcançar *Pronto para Decolar* e outra norma proibindo a entidade de executar *Carregar*, as normas estarão em conflito. O avião não pode atingir o estado *Pronto para Decolar* sem que ele esteja *Carregado*.

No caso de obrigações ou permissões aplicadas a ações cujo **efeito** é um estado *integral* obrigatório, o conflito não existirá entre as normas que obrigam (ou permitem) tais ações e a norma que obriga o alcance do estado *integral*. Por outro lado, se existe uma proibição aplicada sobre uma ação cuja **precondição** é um estado *integral*, o conflito não surge entre tal norma e a norma que obriga o alcance do estado *integral*. A entidade pode alcançar o estado *integral* e não executar a ação. Além disso, se existe uma norma obrigando ou permitindo a execução de uma ação cujo a **precondição** é um estado *integral*, tal norma não estará em conflito com a norma obrigatória aplicada sobre o estado *integral*.

Note que, se existe uma norma proibindo a execução de uma ação cujo a **precondição** é um estado *parte*, o conflito não existirá entre tal norma e a norma que obriga o alcance do

estado *integral*. A proibição afirma que a entidade não pode executar a ação, mas nada menciona sobre o alcance de tal **precondição**.

Definição 20 *(Regra de conflito para permissão e obrigação no relacionamento Composition). Se existe uma obrigação ou permissão aplicada sobre um comportamento todo e proibições aplicadas sobre qualquer comportamento parte do respectivo comportamento todo, tais normas estarão em conflito.*

$$\exists b_1 \exists b_2 ((\langle d, c, e, b_1, acc_1, dac_1\rangle \in Norm \wedge \langle pr, c, e, b_2, acc_2, dac_2\rangle \in Norm \wedge composition(b_1, b_2)) \rightarrow conflict(\langle d, c, e, b_1, acc_1, dac_1\rangle, \langle pr, c, e, b_2, acc_2, dac_2\rangle))$$

onde $d \in \{ob, pe\}, c \in C, e \in E, b_1 and\ b_2 \in B, acc_i\ and\ dac_i \in Cnd,\ acc_2 \geq dac_1,$ e $dac_1 \geq dcc_2$.

Proibição: Se uma norma proíbe o alcance de um estado *todo*, isso não significa que os estados *parte* do estado *todo* também estão sendo proibidos. A violação da norma proibitiva ocorrerá se todos os estados *parte* forem alcançados no mesmo período estipulado pela norma. Se existe ao menos um estado *parte* que não está sendo obrigado (ou permitido) de ser alcançado no mesmo período da norma que proíbe o alcance do estado *todo*, o conflito é inexistente.

Os casos de conflitos que podem surgir quando uma proibição é aplicada sobre um estado *todo* e outras normas são aplicadas sobre ações também devem ser analisados. Se existe uma norma proibindo a execução de uma ação cuja **condição** ou o **efeito** são estados *parte* (ou estado *todo*), tais normas nunca entrarão em conflito com a proibição que está sendo aplicada sobre o estado *todo*. Por outro lado, se uma obrigação ou permissão é aplicada sobre uma ação cuja **condição** é um estado *parte*, o conflito entre tal obrigação (ou permissão) com a norma que está proibindo o alcance do estado *todo* não ocorre. Nesse caso, a entidade é livre do conflito devido que a mesma é proibida de alcançar o estado *todo*, mas não é proibida de atingir os estados *parte* isoladamente. Assim, a entidade é capaz de cumprir com a proibição aplicada sobre o estado *todo* e também cumprirá com a obrigação (ou permissão) aplicada sobre a execução da ação. Por exemplo, se existe uma norma obrigando (ou permitindo) a execução da ação *Descarregar*, a **condição** *Carregado* deve ser alcançada, o que não significa que o estado *Pronto para Decolar* também tenha sido alcançado. O estado *Pronto para Decolar* é alcançado somente quando os estados *parte Abastecido* e *Carregado* também são alcançados.

Se existe uma norma obrigando ou permitindo a execução de uma ação cujo **efeito** é um estado *parte*, o conflito não surgirá entre tal norma obrigatória (ou permissiva) e a norma que proíbe o alcance do estado *todo*. A entidade pode cumprir com a obrigação – executando a ação

– sem violar a norma que define a proibição do alcance do estado *todo*. Porém, é importante verificar se existem normas que obrigam ou permitem a execução das ações cujo **efeito** inclui todos os estados *parte*. Se todas essas normas são cumpridas, a entidade alcançará todos os estados *parte*, caracterizando então o alcance automático do estado *todo* que está sendo proibido. Em tal caso, o conflito é existente entre as normas obrigatórias (ou permissivas) aplicadas sobre as ações cujo o **efeito** são os estados *parte* e a norma que proíbe o alcance do estado todo.

Para exemplificar, vamos considerar a existência de uma norma obrigando (ou permitindo) a execução da ação *Carregar* ou uma norma obrigando (ou permitindo) a execução da ação *Suprir*. Não existe conflito entre tais normas e a norma que define a proibição sobre o estado *todo Pronto para Decolar*. Contudo, se a execução de ambas as ações estão sendo obrigatórias no mesmo período em que o estado *Pronto para Decolar* está sendo proibido, as normas estarão em conflito.

Definição 21 *(Regra de conflito para proibições no relacionamento Composition). Se existe uma proibição aplicada sobre um comportamento todo e obrigações (e/ou permissões) aplicadas sobre todos os comportamentos parte do respectivo comportamento todo, tais normas estarão em conflito.*

$$\exists b_1(\langle pr, c, e, b_1, acc_1, dac_1\rangle \in Norm \rightarrow \forall b_i(composition(b_1, b_i) \rightarrow conflict(\langle pr, c, e, b_1, acc_1, dac_1\rangle, \langle d, c, e, b_i, acc_2, dac_2\rangle)))$$

$$onde\ \ b_i \in \{b \mid composition(b_1, b)\}, d \in (ob, pe), c\ \in C, e \in E, b_1 and\ b_2 \in B, acc_i\ and\ dac_i \in Cnd,\ \ acc_2 \geq acc_1,\ \text{e}\ dac_1 > dac_2.$$

3.3.3.4 ORTHOGONALITY ENTRE COMPORTAMENTOS

O relacionamento *Orthogonality* entre comportamentos é caracterizado por comportamentos que não podem ser executados em períodos de tempo relacionados.

Definição 22 *(Orthogonality). O relacionamento Orthogonality relaciona um comportamento que não pode ser executado simultaneamente com outro comportamento.*

$$Orthogonality \subseteq Comportamento \times Comportamento$$

Para exemplificar os casos de conflito que serão descritos no decorrer dessa Seção que descreve a ortogonalidade entre comportamentos, vamos considerar os exemplos de estados e ações utilizados anteriormente. Os estados *Voando* e *Taxiando* não podem ser atingidos pela

entidade em um mesmo período de tempo, pois o avião (entidade em questão) não é capaz de *voar* e *taxiar* ao mesmo tempo.

Na ontologia, os estados *Voando* e *Taxiando* são estados ortogonais, sendo assim não podem ser atingidos em um mesmo período de tempo. Para completar o restante do exemplo, o ciclo das duas ações (Conforme apresentado na Seção 3.1.4) foram definidos (*Voando, Aterrissando, Taxiando,* e *Taxiando, Decolar, Voando*) e serão utilizados como exemplo no decorrer dessa Seção. Os casos de conflito serão apresentados com base na norma que é aplicada sobre o alcance do estado.

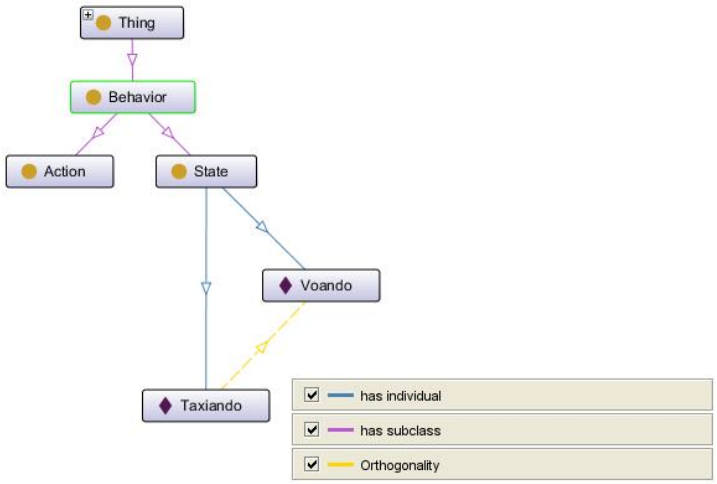

Figura 25: *Orthogonality* definido entre estados.

Obrigação/Permissão: Se existe uma norma obrigando (ou permitindo) o alcance de um estado e outra norma obrigando (ou permitindo) o alcance de outro estado que é ortogonal ao primeiro, tais normas estarão em conflito. Por exemplo, se existe uma obrigação aplicada sobre o alcance do estado *Voando* e outra norma obrigando a mesma entidade a alcançar o estado *Taxiando* em um mesmo período de tempo, tais normas estarão em conflito.

Se existe uma norma obrigando o alcance de um estado e outra norma obrigando (ou permitindo) a execução de uma ação cujo o **efeito** é o estado da outra norma, tais normas estarão em conflito. Ambos os estados não podem ser atingidos por uma mesma entidade em um mesmo período de tempo. Então, a entidade não pode cumprir com a norma que obriga o estado e executar a ação sem violá-la. Por exemplo, se existe uma norma obrigando a execução da ação *Aterrissando* cujo o efeito é o estado *Taxiando* e existe outra norma obrigando o alcance do estado *Voando,* o conflito existe entre tais normas. Por outro lado, se existe uma norma obrigando o alcance de um estado e outra norma proibindo a execução de uma ação cujo o **efeito** é o estado que está sendo obrigatório, o conflito não é afirmado entre tais normas.

No caso de uma norma estar obrigando o alcance de um estado e outra norma estar obrigando (ou permitindo) a execução de uma ação cujo a **precondição** é um estado ortogonal ao estado da primeira norma, o conflito entre tais normas não existe. Para exemplificar, se existe uma norma obrigando o alcance do estado *Voando* e existe outra norma obrigando a execução da ação *Decolar*, tais normas não estão em conflito. Nesse caso, a execução da ação contribuirá diretamente com o alcance do estado que está sendo obrigado pela primeira norma.

Se existe uma norma obrigando o alcance de um estado e outra norma proibindo a execução de uma ação cujo a **precondição** é um estado ortogonal ao estado sendo regulado na primeira norma, o conflito pode existir. Tal **precondição** é ortogonal ao estado que deve ser alcançado. Por exemplo, a **precondição** da ação *Decolar* é o estado *Taxiando* que é ortogonal ao estado *Voando*. Se *Decolar* está sendo proibida e *Voando* está sendo obrigatória, as normas apresentarão conflito se a única ação que pode ser executada e que leva ao alcance do estado *Voando* é *Decolar*.

Definição 23 *(Regra de conflito para permissão e obrigação no relacionamento Orthogonality). Se existe uma obrigação ou permissão aplicada sobre um comportamento e permissões ou aplicadas sobre um comportamento ortogonal ao primeiro, tais normas estarão em conflito.*

$$\exists b_1 \exists b_2 ((\langle ob, c, e, b_1, acc_1, dac_1\rangle \in Norm \wedge \langle d, c, e, b_2, acc_2, dac_2\rangle \in Norm \wedge orthogonality(b_1, b_2)) \rightarrow conflict(\langle ob, c, e, b_1, acc_1, dac_1\rangle, \langle d, c, e, b_2, acc_2, dac_2\rangle))$$

onde $d \in \{ob, pe\}, c \in C, e \in E, b_1 and\ b_2 \in B, acc_i\ e\ dac_i \in Cnd,\ acc_1 \geq dac_2,$ e $dac_1 \geq acc_2$.

3.3.4 ATIVAÇÃO E DESATIVAÇÃO DA NORMA

Uma norma deve ser cumprida dentro de um prazo que está definido em suas propriedades. Como mencionado no Capítulo 2, os fatores de ativação e desativação de uma norma podem ser indicações como: um determinado tempo que define o momento da ativação e o momento da desativação, o cumprimento de uma norma, a violação de uma norma, a execução de uma ação, o alcance de um estado, entre outros.

O verificador de conflitos considerará o **tempo** como fator de ativação e desativação de uma norma. A ativação e desativação de uma norma pode ser dada através de um período no tempo. Com isso, tal norma é independente de qualquer outro evento que não seja o tempo que determina o seu período de ativação. Se duas normas são aplicadas sobre períodos iguais ou que possuem intersecção, significa que um possível conflito entre elas pode ocorrer. Quando a

norma especifica tempo, tal especificação pode ser dada através da definição de uma data e hora (*dd/mm/aaaa 00:00:00*) ou todos os dias (*everyday*). Se a primeira opção é definida, tanto o momento da ativação quanto o da desativação devem ser especificados. Caso seja o dia de hoje ou todos os dias, o período de ativação é implícito, não precisando ser mencionado o momento em que a norma se tornará inativa.

CAPÍTULO 4 – ESPECIFICAÇÃO E IMPLEMENTAÇÃO DO CONFLICT CHECKER

O *Conflict Checker* é a ferramenta de verificação de conflitos normativos, desenvolvido especialmente para a verificação de conflitos diretos e indiretos se baseando na ontologia que define as normas e os relacionamentos entre os elementos da norma.

As Seções desse Capítulo apresentam as propriedades e funcionamento do *Conflict Checker*, bem como as especificações de implementação. As ferramentas utilizadas para o desenvolvimento desse mecanismo de verificação de conflitos são softwares gratuitos e disponíveis na Internet. A linguagem *Java* é a linguagem de programação utilizada, juntamente com a OWL-API – descrita na Seção 2.4.3. O ambiente de desenvolvimento utilizado foi o NetBeans IDE em sua versão 8.0, que oferece total suporte para a utilização da linguagem Java em conjunto com a API. No decorrer desse Capítulo, alguns diagramas UML serão apresentados afim de especificar o funcionamento do *Conflict Checker* e a sua estrutura.

4.1 VISÃO GERAL DO CONFLICT CHECKER

O *Conflict Checker* foi desenvolvido a fim de verificar conflitos entre normas, baseando-se na representação do conhecimento do domínio para que conflitos indiretos possam ser detectados.

A Figura 26 apresenta uma visão da interface gráfica do *Conflict Checker*. Na parte inferior encontram-se três botões que controlam o funcionamento do mecanismo: *reset*, *upload* e *execute*.

O botão *reset* é responsável por limpar todas as informações de ontologias e de resultados de verificações contidas na memória. Em outras palavras, o botão *reset* reinicializa o mecanismo, deixando-o em seu estado inicial. É importante ressaltar que uma das limitações desse mecanismo é a capacidade de processar apenas um arquivo OWL por vez. Caso exista a necessidade da análise de outro arquivo OWL, o C*onflict Checker* deverá ser reinicializado para que isso seja possível. O *reset* se torna ativo somente quando uma ontologia está armazenada na memória.

O botão *upload* abre ao usuário o explorador de arquivos do sistema operacional, para que o usuário seja capaz de informar ao *Conflict Checker* o local onde se encontra o arquivo OWL a ser analisado – Figura 27. Quando um arquivo OWL é selecionado através do explorador de arquivos, as informações sobre as normas e os relacionamentos existentes entre os elementos contidos nas normas são organizados automaticamente no primeiro grande campo:

o campo *Ontology* – Figura 28. O campo *Ontology* é responsável por apresentar as informações dos elementos que compõem a ontologia, bem como os relacionamentos que esses elementos possuem. No contexto desse trabalho e considerando as definições contidas no Capítulo 3, nesse campo estarão contidas informações tais como as entidades, contextos, comportamentos e os relacionamento entre esses elementos.

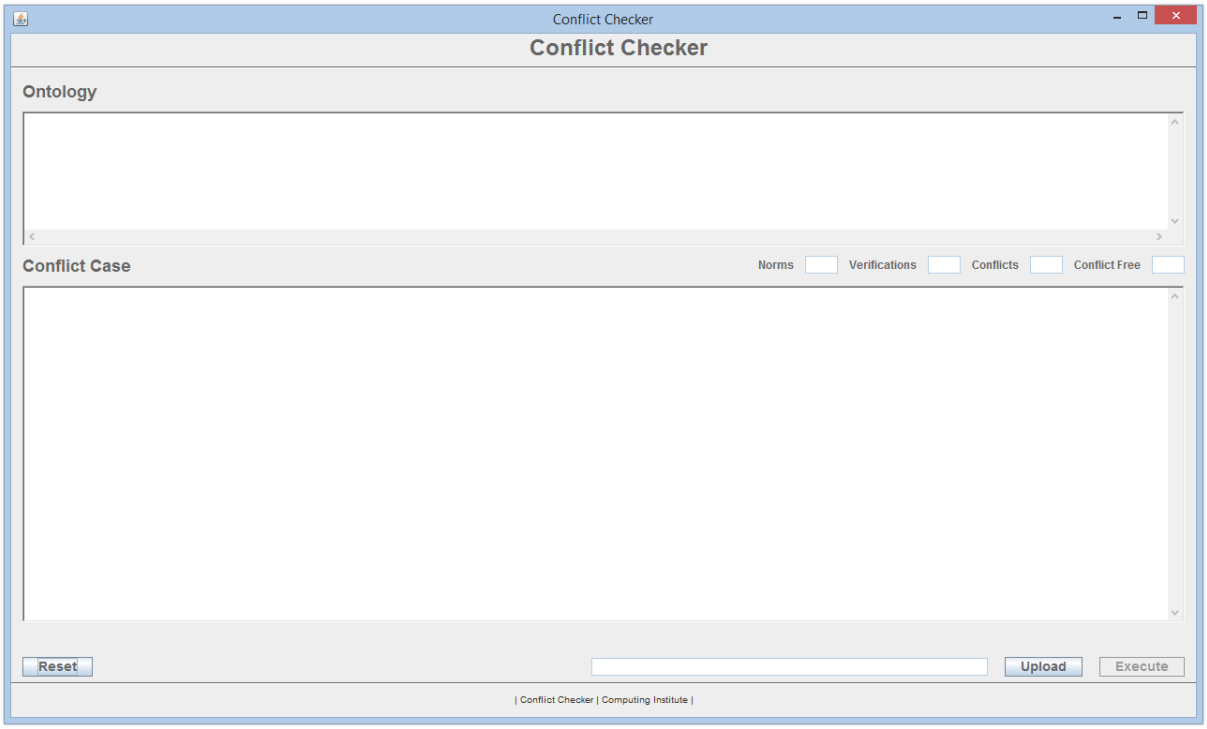

Figura 26: O *Conflict Checker*.

O botão *execute*, quando ativado, aplica as regras de verificação de conflitos normativos definidos no Capítulo 3. Aplicar as regras significa organizar as normas em pares, de modo a contemplar todas combinações de pares possíveis. Após a verificação de todo o conjunto de pares normativos, os resultados de conflitos entre pares normativos são expostos no segundo grande campo da aplicação: o campo *Conflict Case* – Figura 29. Além de informar quais normas estão em conflito, são apresentados também os motivos do conflito.

Nos resultados da verificação de conflitos, os pares normativos que não apresentam conflitos também são expostos. Além de informar quais os pares que não apresentam conflito, o *Conflict Checker* também apresenta o(s) motivo(s) do conflito não existir.

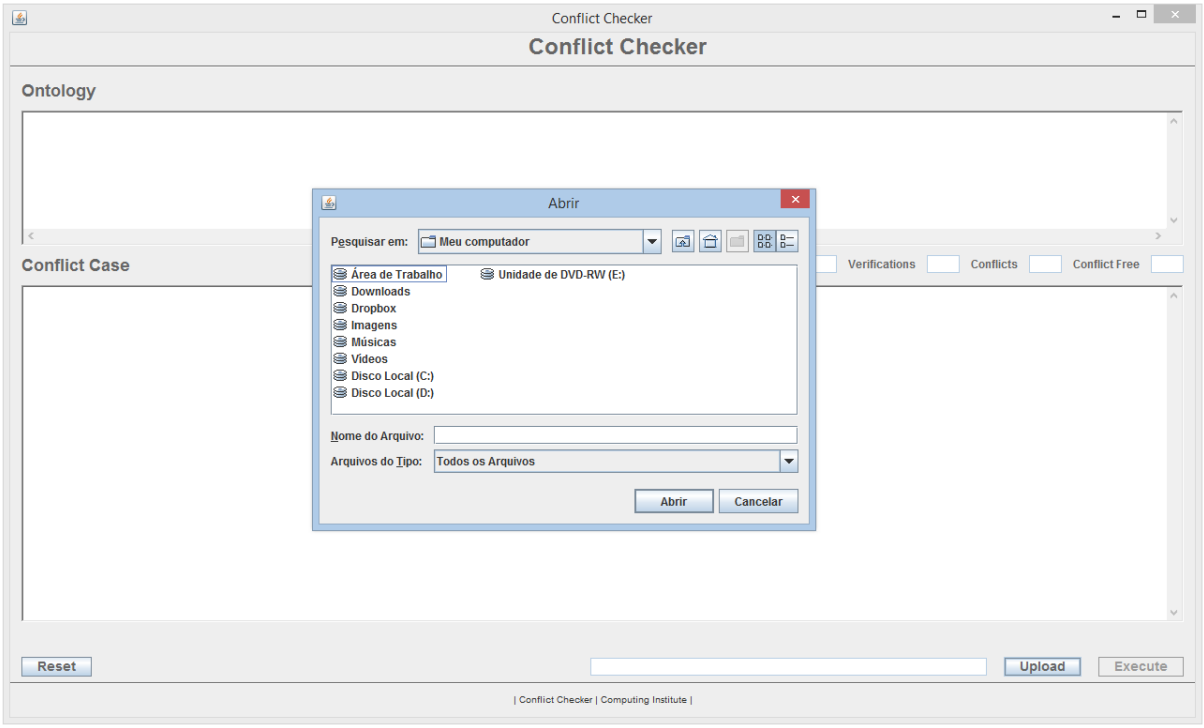

Figura 27: Entrada de informações no *Conflict Checker*.

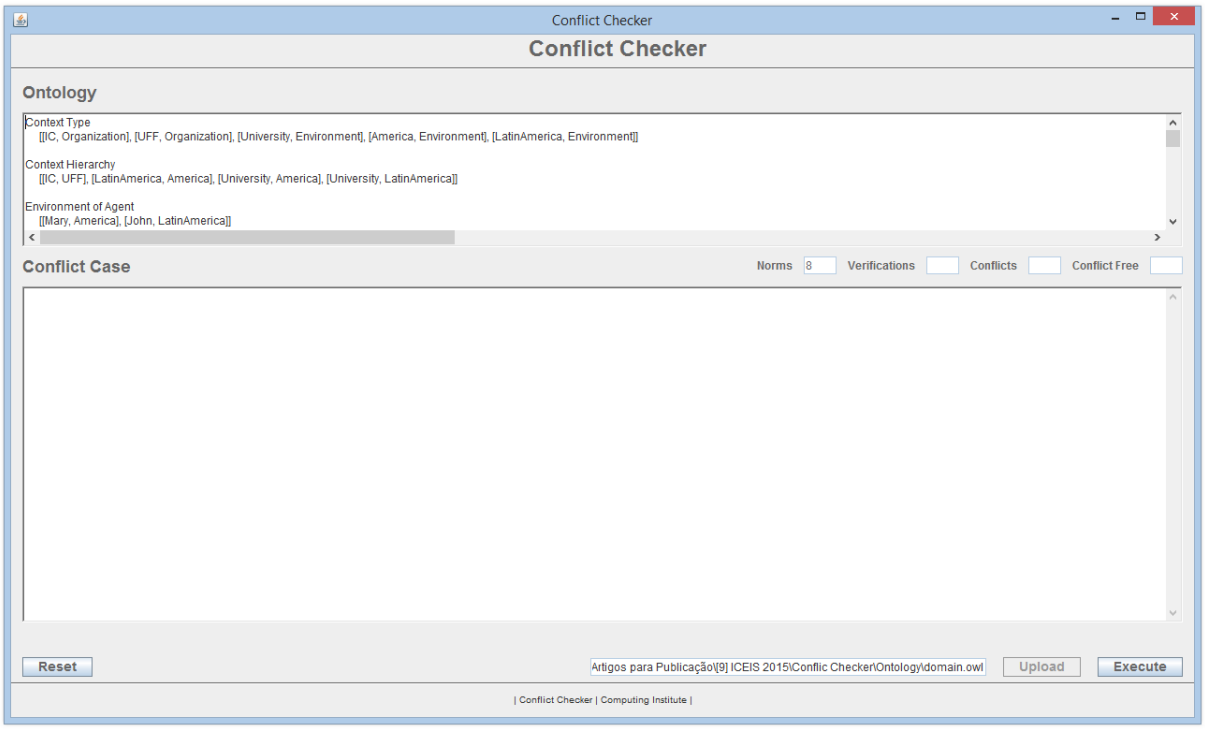

Figura 28: Informações sobre a ontologia.

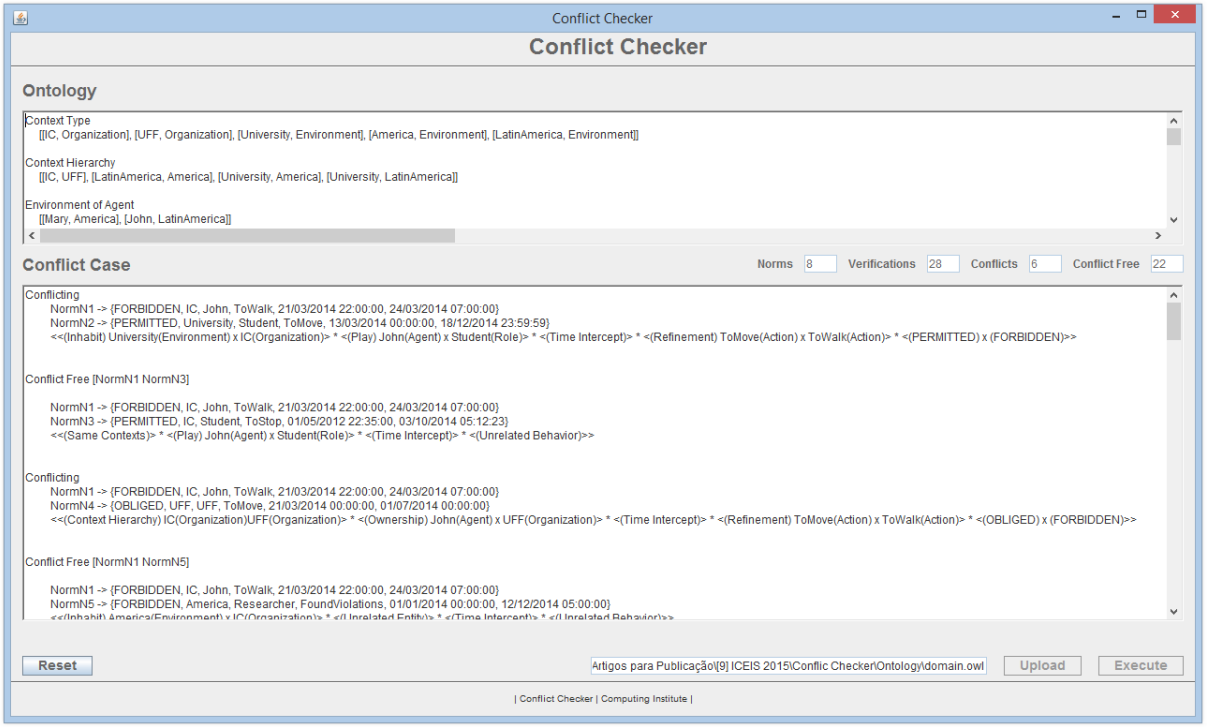

Figura 29: O resultado das verificações.

A interface ainda possui quatro pequenos campos localizados no centro-direito do protótipo que apresentam informações estatísticas sobre o arquivo OWL que está sendo analisado. Os campos possuem as seguintes funções:

- **Norms**: expõe o número total de normas contidas no arquivo OWL;
- **Verifications**: apresenta o número total de verificações necessárias para que todos os pares normativos possíveis sejam contemplados. As normas são organizadas em pares únicos e que não se repetem. Além disso, a mesma norma não é verificada com ela mesma. Com isso, o número total de verificações é obtido através da fórmula:

$$verificações = \frac{número\ de\ normas(número\ de\ normas - 1)}{2}$$

- **Conflicts**: contém o número de casos de conflitos contidos no conjunto de normas verificadas;
- **Conflict Free**: contém o número de pares normativos que não apresentam conflitos no conjunto de normas verificadas.

As informações de saída são os resultados das verificações da existência de conflito. O *Conflict Checker*, ao fim da verificação, expõe todas as informações sobre a existência de

conflito ou a ausência do mesmo na segunda caixa de texto, logo abaixo das informações sobre a ontologia. A Figura 29 apresenta um exemplo de verificação de conflitos através do mecanismo do *Conflict Checker*.

Quando contextos entre duas normas se relacionam, o retorno da verificação de conflitos refere-se ao tipo de relacionamento existente entre o contexto do par normativo, seguido pelo nome do contexto da primeira norma e o seu tipo e, por fim, o nome do contexto e o tipo do contexto contido na segunda norma. Se os contextos são iguais, o verificador informa que os contextos são iguais (`Same Context`). Se os contextos são diferentes e não relacionados, o verificador afirma que não existem ligações entre o par de contextos contidos no par normativo (`Unrelated Context`). Abaixo, segue a estrutura utilizada (separada por elementos) para formalizar a existência ou ausência de relacionamentos entre contextos:

```
Contextos relacionados:
<(relacionamento) Contexto da Norma 1 (tipo) x Contexto da Norma 2 (tipo)>

Contextos iguais:
<(Same Context)>

Contextos diferentes e não relacionados:
<(Unrelated Context)>
```

Quando entidades se relacionam, o retorno da verificação de conflitos refere-se ao tipo de relacionamento existente entre as entidades envolvidas no par normativo, seguido pelo nome do contexto da primeira norma e o seu tipo, por fim, o nome da entidade e o tipo de entidade contida na segunda norma. Se as entidades são iguais, o verificador informa que os contextos são iguais (`Same Entity`). Se as entidades não são iguais e não estão relacionadas por algum tipo de relacionamento, o verificador afirma que não existem ligações entre o par de entidades contidas no par normativo (`Unrelated Entities`).

```
Entidades relacionadas:
<(relacionamento) Entidade da Norma 1 (tipo) x Entidade da Norma 2 (tipo)>

Entidades iguais:
<(Same Entity)>

Entidades diferentes e não relacionados:
<(Unrelated Entities)>
```

Se os comportamentos se relacionam, o retorno da verificação de conflitos refere-se ao tipo de relacionamento existente entre os comportamentos envolvidos no par normativo, seguido pelo nome do contexto da primeira norma e o seu tipo, por fim, o nome do comportamento e o tipo do comportamento contido na segunda norma. Se os comportamentos são iguais, o verificador informa que os comportamentos são iguais (`Same Behavior`). Se os

comportamentos não são iguais e não estão relacionados por algum tipo de relacionamento, o verificador afirma que não existem ligações entre o par de comportamentos contidos no par normativo (`Unrelated Entities`).

```
Comportamentos relacionados:
<(relacionamento) Comportamento da Norma 1 (tipo) x Comportamento da Norma 2 (tipo)>

Comportamento iguais:
<(Same Behavior)>

Comportamentos diferentes e não relacionados:
<(Unrelated Behaviors)>
```

Quando fatores de tempo são relacionados, o retorno afirma apenas que tais tempos estão relacionados (`Time Intercept`). Esse é o único retorno para afirmar que as normas se relacionam através dos fatores de tempo. Fatores de tempo podem também não se relacionar, retirando a possibilidade de conflito entre as normas que compõem o par normativo (`Unrelated Time`).

```
Fatores de tempo relacionados:
<(Time Intercept)>

Fatores de tempo não relacionados:
<(Unrelated Time)>
```

Com a caracterização dos motivos pelos quais os relacionamentos aconteceram (ou não aconteceram) em cada elemento que compõe a norma, a tupla que justifica a existência (ou não existência) dos conflitos pode então ser gerada. O par normativo também é exposto juntamente com a tupla que justifica o caso de conflito (ou que não apresenta conflito). Apresentar o par normativo juntamente com a justificativa contribui para um melhor entendimento dos resultados expostos pelo *Conflict Checker*. A informação sobre conflitos é expressada da seguinte maneira:

```
Conflicting [NormId, NormId]
      NormId -> {D, C, E, B, Cnd, Cnd}
      NormId -> {D, C, E, B, Cnd, Cnd}

<<(Context)> * <(Entity)> * <(Time)> * <(Behavior)> * <(Deontic Concept)>>
```

Para exemplificar, duas normas são aplicadas:

```
N = {NormN23, FORBIDDEN, Rio de Janeiro, Comprador, Comprar
                                  12/02/2015 07:00:00, 14/02/2015 12:00:00}

N = {NormN12, PERMITTED, Macaé, João, Comercializar, _, _}
```

A norma *NormN23* é aplicada no contexto do *Rio de Janeiro*, afirmando que a entidade *comprador* está *proibido* de *comprar* durante o período entre *12/02/2015 07:00:00* e 14/02/20015 12:00:00. A *NormN12* é aplicada sobre o *João* no contexto de *Macaé*, permitindo a entidade de *comercializar* durante todo o tempo (sem restrições de tempo). Considerando que *Macaé* é uma cidade do estado do *Rio de Janeiro* (*Hierarchy* de contextos), considerando que *João* desempenha o papel de *Comprador* (*Play*), considerando que *comprar* é uma ação que especializa a ação *comercializar* (*Refinement*) e que os períodos de tempo se interceptam, o conflito é afirmado, sendo expressado pelo *Conflict Checker* da seguinte maneira:

```
Conflicting [Norm23, Norm12]
      Norm23 -> {FORBIDDEN, Rio de Janeiro, Comprador, Comprar,
                                  12/02/2015 07:00:00, 14/02/2015 12:00:00}
      Norm12 -> {PERMITTED, Macaé, João, Comercializar, _, _}

<<(Hierarchy) Rio de Janeiro(Organization) x Macaé(Organization)> *
 <(Play) Comprador(Role) x João(Agent)> *
 <(Time Intercept)> *
 <(Refinement) Comprar(Action) x Comercializar (Action)> *
 <(FORBIDDEN) x (PERMITTED)>>
```

Por outro lado, caso o conflito entre as normas seja inexistente, o resultado também é apresentado. Para exemplificar, considere as mesmas normas *NormN23* e *NormN12*. No entanto, *João* não desempenha mais o papel de *comprador*, ou seja, o relacionamento *Play* deixou de existir entre as entidades.

```
Conflict Free [Norm23, Norm12]
      Norm23 -> {FORBIDDEN, Rio de Janeiro, Comprador, Comprar,
                                  12/02/2015 07:00:00, 14/02/2015 12:00:00}
      Norm12 -> {PERMITTED, Macaé, João, Comercializar, _, _}

<<(Hierarchy) Rio de Janeiro(Organization) x Macaé(Organization)> *
 <(Unrelated Entities)> *
 <(Time Intercept)> *
 <(Refinement) Comprar(Action) x Comercializar (Action)> *
 <(FORBIDDEN) x (PERMITTED)>>
```

Apesar dos elementos normativos serem relacionados, bastou que as entidades não apresentassem relacionamento entre elas para que as normas estivessem livres de conflitos normativos.

Ao fim da verificação, a análise pode ser realizada, a fim de tomar iniciativas para que as normas sejam alteradas e que os conflitos deixem de existir. Após a exibição dos resultados, o mecanismo de verificação de conflitos proposto até aqui encerra o ciclo de funcionamento, aguardando ser finalizado ou reiniciado.

4.2 ESPECIFICAÇÃO E IMPLEMENTAÇÃO

Nesta Seção são apresentados os detalhes da implementação do mecanismo proposto. A especificação dessa implementação ocorrerá através de *diagramas UML*, juntamente com trechos fundamentais do código fonte do *Conflict Checker*.

4.2.1 ESTRUTURA LÓGICA DAS CLASSES

A Figura 30 contém a estrutura lógica das classes do *Conflict Checker*. É importante destacar a classe *Ontology*, pois tal classe é responsável por extrair e armazenar as informações oriundas da ontologia.

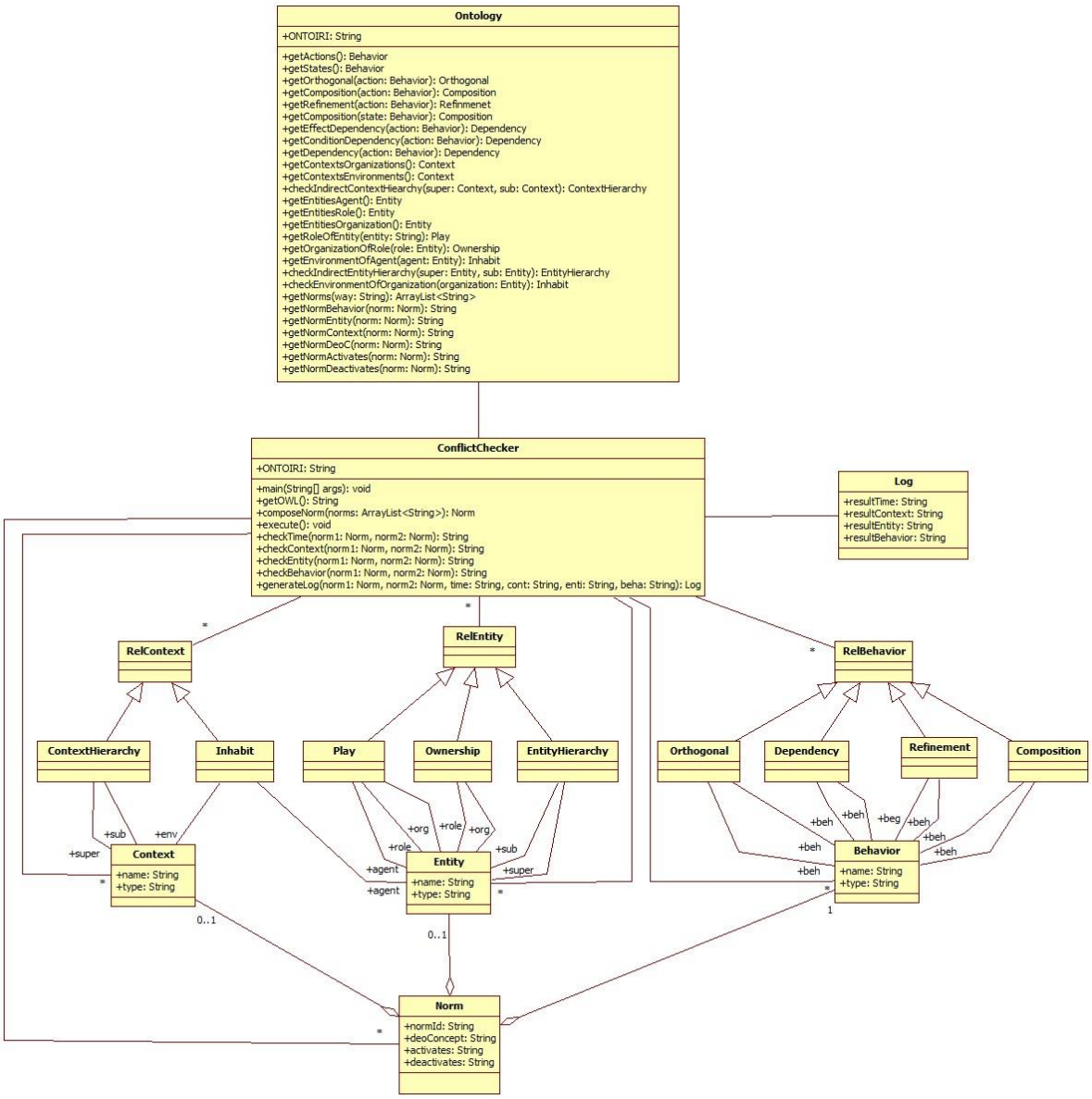

Figura 30: Diagrama de Classes.

A classe *Ontology* é responsável por extrair as informações a respeito das normas e dos relacionamentos entre os elementos existentes no domínio. O método *getNorms(way:String)*, por exemplo, cria uma lista contendo todas as normas existentes. Essa lista é utilizada pelo método *composeNorm(norms:ArrayList<String>)*, que é responsável por coordenar a extração de todas as informações referentes a tal norma.

As classes *Context*, *Entity* e *Behavior* são classes que instanciam, respectivamente, os contextos, as entidades e os comportamentos presentes no domínio.

A classe *RelContext* é uma classe abstrata, portanto, não é instanciada. Tal classe possui duas subclasses definidas em um relacionamento de herança: *ContextHierarchy* e *Inhabit*.

Os relacionamentos entre entidades são definidos nas classes *Play*, *Ownership* e *EntityHierarchy*. Estas classes são subclasses da *RelEntity*.

Já os relacionamentos entre comportamentos são definidos na classe *RelBehavior*, que possui como subclasses as classes *Orthogonal*, *Dependency*, *Refinement* e *Composition*.

A classe *Log* instancia os resultados das verificações. Para cada par normativo verificado, uma nova instância é gerada com os motivos pelos quais o conflito existiu (ou não existiu). Os resultados dessas verificações são expostos ao final da verificação da existência de conflito, conforme já mencionado na Seção 4.1.

A aplicação das regras ocorre na classe *ConflictChecker*. Os quatro métodos da classe verificam, respectivamente, o período de ativação e desativação das normas (*checkTime()*), o contexto definido no par normativo (*checkContext()*), as entidades (*checkEntity()*) e os comportamentos sendo regidos pelas normas (*checkBehavior()*). A classe *ConflictChecker* será melhor detalhada na Seção 4.2.2.

4.2.2 A CLASSE CONFLICTCHECKER

Aplicar as regras de verificação de conflitos consiste em analisar as informações contidas no domínio da aplicação, i.e., os relacionamentos entre contextos, entidades e comportamentos, juntamente com o conjunto de normas existentes em tal domínio. É importante mencionar que, a verificação de conflitos nesse trabalho ocorre através da comparação de pares normativos. No entanto, nos relacionamentos *Refinement* (Seção 3.3.3.1) e *Composition* (Seção 3.3.3.3) necessitamos verificar se outras normas podem alterar o resultado da existência de conflitos.

A classe *ConflictChecker* é a classe principal do mecanismo proposto. Esta classe contém o método *Main()*, o método *execute()*, o método *composeNorm()*, entre outros métodos importantes. O método *composeNorm()* é o responsável por criar as instâncias das normas. Este

método invoca os métodos da classe *Ontology*, buscando saber quais os elementos (*contexto*, *entity* e *behavior*) que a norma se aplica. Esse método busca também pelos atributos de uma norma, invocando por exemplo, o método *getNormActivates()* que extrai o tempo em que a norma é ativa.

O método *execute()* é responsável por inicializar e coordenar a aplicação das regras de verificação de conflitos sobre o conjunto de normas extraído na ontologia. Os quatro métodos que contém as regras são:

- **CheckTime(norm1: Norm, norm2: Norm): String**: é o método responsável por aplicar as regras referentes ao tempo de ativação e desativação de um par normativo. Esse método recebe duas normas distintas, verifica a existência de intersecção do tempo entre elas e, retorna a descrição que afirma o não relacionamento entre os períodos de tempo, caso as normas não compartilham de um mesmo período de tempo, ou retorna a descrição que afirma que os períodos de tempo estão relacionados, para o caso das normas apresentarem intersecção entre os períodos de ativação – conforme as regras apresentadas na Seção 3.3.4.
- **CheckContext(norm1: Norm, norm2: Norm): String**: é o método que aplica as regras apresentadas na Seção 3.3.1. O método *execute()* aciona o método *checkContext()*. Tal método verifica a existência de relacionamentos entre os contextos definidos nas normas, retornando descrição que afirma o não relacionamento entre contextos, caso os contextos não sejam relacionados, ou a descrição que afirma qual o relacionamento existente entre os contextos, para contextos que apresentam relacionamento entre eles.
- **CheckEntity(norm1: Norm, norm2: Norm): String**: as definições apresentadas na Seção 3.3.2 são aplicadas nesse método. O método *execute()* ativa o método *checkEntity()*. Esse método verifica a existência dos relacionamentos que envolvem as entidades contidas em um par normativo. Caso um relacionamento exista, o método retorna a descrição que afirma a existência do relacionamento e qual é esse relacionamento, caso contrário, retorna a descrição que afirma a não existência de relacionamento entre as entidades.
- **CheckBehavior(norm1: Norm, norm2: Norm): String**: os relacionamentos *Refinement*, *Dependency*, *Composition* e *Orthogonality* são verificados nesse

método. O método *checkBehavior()* retorna a *String* contendo o tipo de relacionamento presente entre os comportamentos, caso os comportamentos estejam relacionados, e retorna a descrição que afirma a não existência de relacionamento entre os comportamentos, no caso desses relacionamentos não existirem. Juntamente com a verificação dos comportamentos de um par normativo, esse método verifica a combinação dos conceitos deônticos com o tipo de relacionamento associado, conforme descrito na Seção 3.3.3. É importante mencionar que em alguns tipos de relacionamentos entre comportamentos (*Refinement* e *Composition*), em alguns casos, considerar apenas o par normativo não é satisfatório para a constatação de conflitos. Esse método possui a capacidade de verificar (nos relacionamentos *Refinement* e *Composition*) um número maior de normas simultaneamente.

O método *generateLog()* recebe um par normativo, juntamente com todas as descrições geradas pelos métodos anteriores, para que o *Log* da verificação possa ser gerado. Para cada par normativo, um *Log* é gerado, descrevendo todas as causas da existência (ou não) do conflito.

Embora as normas sejam verificadas em pares, os pares normativos não se repetem, pois a ordem em que as normas se encontram em um par normativo não altera o resultado da existência de conflito. Desse modo, pode-se afirmar que as normas são organizadas em uma matriz *n* x *n*, onde *n* é o número de normas existentes no domínio. No entanto, apenas os pares normativos contidos acima da diagonal principal são verificados.

4.2.3 SEQUÊNCIA DOS EVENTOS PARA VERIFICAR CONFLITOS NORMATIVOS

O processo de verificação de conflitos normativos ocorre em três etapas bem simples: a primeira (i) é a responsável por obter as informações do arquivo OWL, seguido pela segunda etapa (ii) que constitui na aplicação das regras de verificação de conflitos e, por último, a terceira etapa (iii) é responsável por emitir os resultados ao usuário.

O primeiro evento (obter informações do arquivo OWL) corresponde em retirar as informações do documento OWL e organizar no *Conflict Checker*. Essas informações são os elementos contidos no domínio, os relacionamentos presentes entre eles e as normas. Conforme exposto na Figura 30, a classe *Ontology* possui um conjunto de métodos responsáveis por extrair as informações contidas no documento OWL. A Figura 31 apresenta o Diagrama de Sequência da UML, mostrando a sequência de eventos e troca de mensagens no processo de extração de normas definidas na Ontologia.

O processo inicia quando o usuário insere um documento OWL no *Conflict Checker* (método *getOWL()*). Esse método ativa primeiramente o método *getNorms(way:String)*, que é o método responsável por listar todas as normas existentes no domínio. Com a lista de normas, o método *getOWL()* invoca o método *composeNorm()*, que dará início ao processo de preenchimento dessa norma. Isso consiste em buscar pelo contexto definido na norma, pela entidade e pelo comportamento sendo regido por tal norma. Caso esses elementos encontrados ainda não tenham sido instanciados, uma instância na classe correspondente é criada. Os atributos da norma (*conceito deôntico*, *fator de ativação e desativação*) também são buscados, através dos métodos (*getNormDeoC()*, *getNormActivates()* e *getNormDeactivates()*). Após isso, a norma é instanciada na classe *Norm* e o processo se repete enquanto houverem normas na lista de normas.

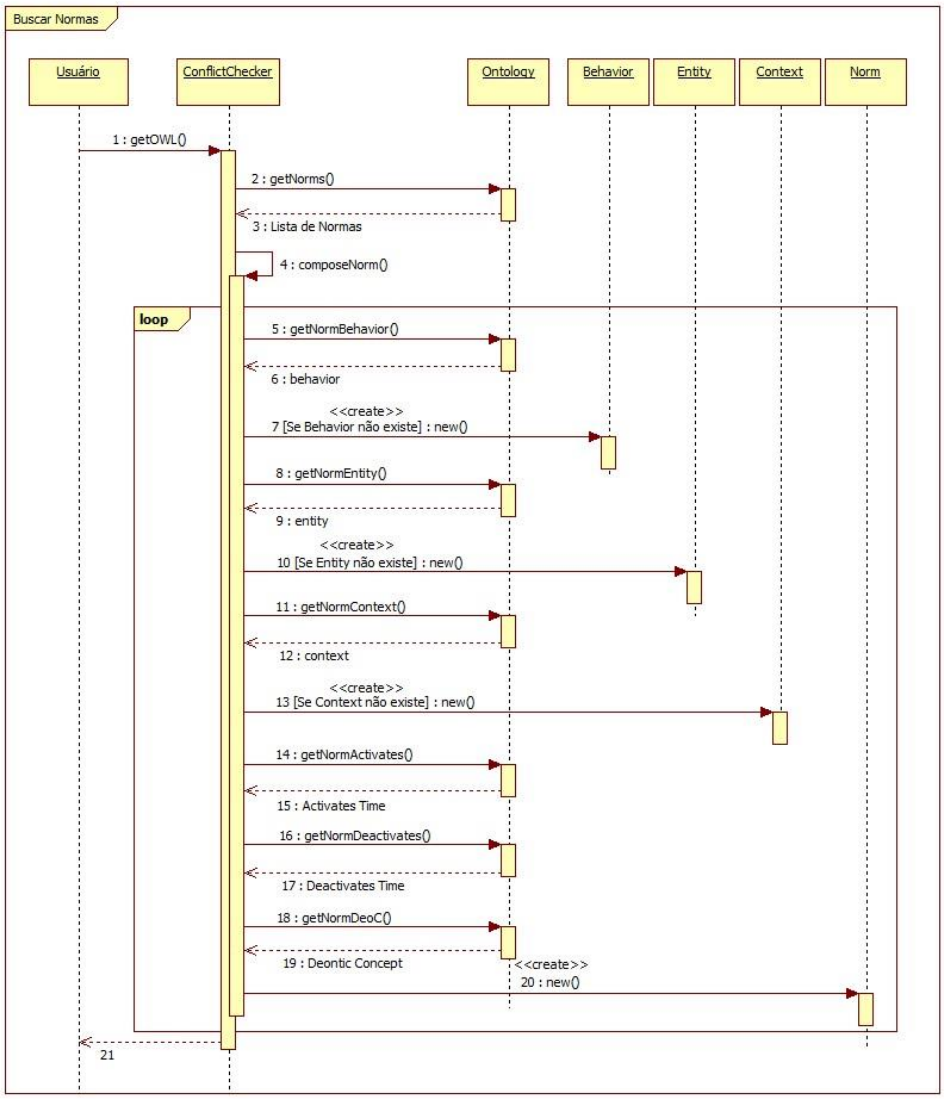

Figura 31: Diagrama de Sequência para a extração de normas.

Após todas as normas terem sido extraídas e instanciadas, o *Conflict Checker* busca pelos demais elementos contidos nas normas, bem como os relacionamentos existentes entre eles. Esse processo é iniciado através da busca por ações e estado, seguido pela extração de contextos e entidades.

A Figura 32 apresenta o código responsável por buscar as ações presentes no domínio. O código está contido no método *GetActions()*, que busca apenas pelas ações, sem distinguir relacionamentos existentes entre elas. As instâncias de uma classe definida na Ontologia podem facilmente ser obtidas. Para isso, é necessário especificar a *IRI* da classe ao *Reasoner*, conforme a primeira linha do código contido na Figura. O *Reasoner* retorna um *NodeSet* contendo os *Individuals* da classe especificada. O método *GetFlattened()* pega todos as entidades contidas no *NodeSet*, listando-as uma a uma.

```
OWLClass action = fac.getOWLClass(IRI.create(ONTOIRI + "#Action"));
NodeSet<OWLNamedIndividual> ontoAction = reasoner.getInstances(action, true);
Set<OWLNamedIndividual> actions = ontoAction.getFlattened();
```

Figura 32: Código-fonte para extração de ações de um documento OWL.

As instâncias de estados são criadas pelo método *getStates()*. Semelhante ao método *getActions()*, o método de extração de estados também ativa os métodos de verificação de relacionamento entre os comportamentos. A única diferença são os métodos que envolvem o relacionamento *Dependency* (*getEffectDependency()* e *getConditionDependency()*), pois ambos os métodos retornam comportamentos do tipo *estado*.

Para cada comportamento encontrado no documento OWL, todos os relacionamentos possíveis entre tais comportamentos são verificados. O mesmo ocorre para os demais elementos presentes na norma (*contexto* e *entidade*).

A Figura 33 apresenta o código-fonte para extrair as ações que contém o relacionamento *Dependency* definido entre elas. Assim como na Figura 32, a classe *Actions* é especificada, afirmando que os *Individuals* de interesse estão contidos nessa classe. O método *getObjectPropertyValues(OWLNamedIndividual, OWLObjectProperty)* busca pelas propriedades contidas entre dois objetos. Em outras palavras, o método busca pelos relacionamentos definidos entre os objetos da classe *Action*. Note que existe apenas uma diferença entre extrair os objetos de uma classe e extrair objetos que se relacionam com os objetos de uma classe. Essa diferença está na segunda linha do código-fonte da Figura 33, onde definimos a *IRI* do *objectProperty* que será extraída.

```
OWLNamedIndividual actions = fac.getOWLNamedIndividual(IRI.create(ONTOIRI + "#" + action));
OWLObjectProperty dependency = fac.getOWLObjectProperty(IRI.create(ONTOIRI + "#Dependency"));
NodeSet<OWLNamedIndividual> actionsDependency = reasoner.getObjectPropertyValues(actions, dependency);
Set<OWLNamedIndividual> values = actionsDependency.getFlattened();
```

Figura 33: Código-fonte para extração de relacionamentos entre ações.

A extração das normas definidas no documento OWL ocorre de maneira semelhante. As normas são instâncias da classe *Norm*, bastando então, informar a *IRI* de tal classe para que o *Reasoner* seja capaz de buscar os objetos da *Norm*. A Figura 34 apresenta o código para a extração das normas. Note que a *IRI* ao fim da declaração na primeira linha define que os *Individuals* de interesse são os contidos na classe *Norm* (*#Norm*).

```
OWLClass norm = fac.getOWLClass(IRI.create(ONTOIRI + "#Norm"));
NodeSet<OWLNamedIndividual> individualsNodeSet = reasoner.getInstances(norm, true);
Set<OWLNamedIndividual> individuals = individualsNodeSet.getFlattened();
```

Figura 34: Código-fonte para extração de normas.

A norma é aplicada sobre uma *entidade*, em um determinado *contexto*, regendo um *comportamento*. Desse modo, uma norma apresenta um relacionamento com um *contexto*, com uma *entidade* e com um *comportamento*. A Figura 35 contém o código-fonte para extrair a entidade que foi definida na norma. Note que no método *getObjectPropertyValues()* são dois os valores passados como parâmetro. O primeiro parâmetro é a *IRI* da instância da norma em questão. O segundo parâmetro é a *IRI* definida no *OWLObjectProperty* (primeira linha).

```
OWLObjectProperty entity = fac.getOWLObjectProperty(IRI.create(ONTOIRI + "#Norm_Entity"));
NodeSet<OWLNamedIndividual> normEntity = reasoner.getObjectPropertyValues(norm, entity);
Set<OWLNamedIndividual> valuesEntity = normEntity.getFlattened();
```

Figura 35: Código-fonte para extração da entidade definida na norma.

Finalizando a extração das informações correspondes à norma, as propriedades definidas na própria norma devem ser extraídas. Essas informações são: o id da norma (*Norm_Id*), o conceito deôntico (*Norm_DeoC*), o fator de ativação (*Norm_Activates*) e o fator de desativação da norma (*Norm_Deactivates*). Essas informações são extraídas pelo método *getDataPropertyValues()* (Figura 36), que recebe como parâmetro a norma de interesse (*norm*) e a *IRI* da propriedade (*id*).

```
OWLDataProperty id = fac.getOWLDataProperty(IRI.create(ONTOIRI + "#Norm_Id"));
Set<OWLLiteral> normId = reasoner.getDataPropertyValues(norm, id);
```

Figura 36: Código-fonte para extração do Id da norma.

Após as características do domínio serem totalmente extraídas pelo *Conflict Checker*, o mecanismo inicia a segunda etapa do seu funcionamento: a aplicação das regras. Essa etapa ocorre quando o usuário pressiona o botão *Execute*, ativando o método *execute()*. Tal método é responsável por convocar os métodos responsáveis por aplicar as regras de verificação de conflito: *checkTime()*, *checkContext()*, *checkEntity()* e *checkBehavior()*. A Figura 37 apresenta a sequência de eventos para a verificação de conflitos.

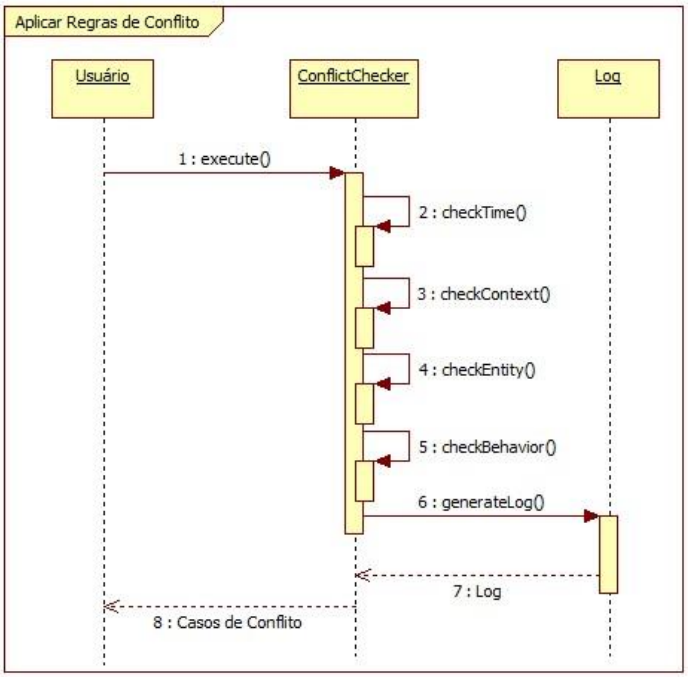

Figura 37: Diagrama de Sequência da aplicação das regras.

O método *execute()* controla a chamada dos demais métodos. O primeiro método a ser ativado é o *checkTime()*. Esse método (Figura 38) contém uma sequência lógica para verificar se os tempos definidos nas normas se relacionam. Primeiramente, os períodos de ativação e desativação da norma são comparados através do método *compareTo()*. Esse método possui a capacidade de comparar dois valores e retornar 0 caso os valores sejam iguais, -1 se o valor passado como parâmetro for maior que o valor comparado e 1 se o valor passado como parâmetro for menor que o valor comparado. Após isso, é verificado se uma das normas foi aplicada sobre todo o tempo (*Everyday*). Caso seja verdade, as normas apresentam relacionamento nos tempos de ativação e desativação. Caso seja falso, um conjunto de condições lógicas são aplicadas para verificar se os períodos de tempo se interceptam.

O retorno do método *checkTime()* é uma *String* que afirma se os tempos das normas apresentam relacionamento ou não. A *String* é passada, ao final da verificação de todos os

elementos da norma, como parâmetro para o método *generateLog()*, onde analisará cada retorno e instanciará um objeto na classe *Log*, com os motivos que levaram a norma a apresentar (ou não) um conflito normativo.

```
public static String checkTime(Norm n1, Norm n2) {
    String retorno;
    int act = n1.getActivates().compareTo(n2.getActivates());
    int deact = n1.getDeactivates().compareTo(n2.getDeactivates());

    if (n1.getActivates().equals("Everyday") || n2.getActivates().equals("Everyday")) {
        retorno = "<(Time Intercept)>";
        return retorno;
    } else {
        if ((act < 0 && deact < 0) || (act < 0 && deact > 0) || (act > 0 && deact > 0) || (act > 0 && deact < 0)) {
            retorno = "<(Time Intercept)>";
        return retorno;
        } else {
            if ((act == 0 && (deact < 0 || deact == 0 || deact > 0)) || (deact == 0 && (act < 0 || deact == 0 || deact > 0))) {
                retorno = "<(Time Intercept)>";
        return retorno;
            }
        }
    }
    retorno = "<(Unrelated Time)>";
    return retorno;
}
```

Figura 38: Código-fonte do método *checkTime()*.

Ao final de toda a verificação, um conjunto de objetos da classe *Log* foi gerado, descrevendo todos os casos de conflitos e casos que não apresentaram conflitos normativos. Com isso, o resultado pode ser apresentado ao usuário, conforme apresentado na Seção 4.1.

4.3 O CONFLICT CHECKER COMO UMA FERRAMENTA EXPANSÍVEL

O *Conflict Checker* foi desenvolvido para que seja expansível a novos relacionamentos que possibilitem detectar novos tipos de conflitos normativos. Essa expansividade deve ocorrer também no modelo da ontologia, mantendo a compatibilidade entre o protótipo e a ontologia definida no documento OWL. Nesta Seção, daremos um breve exemplo de como o *Conflict Checker* pode ser expansível a novos relacionamentos e casos de conflito.

O *Conflict Checker*, até aqui, é capaz de detectar conflitos normativos reconhecendo cinco tipos de relacionamentos entre contextos e entidades e 4 tipos de relacionamentos envolvendo comportamentos. Podemos pressupor a inserção de um novo tipo de relacionamento envolvendo entidades, afim de exemplificar a expansividade do mecanismo.

O relacionamento entre entidades ortogonais será o novo relacionamento entre papeis, denominado então como relacionamento **Orthogonality** entre papéis. Esse relacionamento define dois papéis que não podem ser desempenhados em um mesmo período de tempo (i.e., em um tribunal, nenhuma entidade pode desempenhar o papel de *réu* e *juiz* simultaneamente). O modelo de ontologia não sofreria nenhuma mudança brusca. Um novo relacionamento entre entidades deverá ser criado, gerando um novo *objectProperty* que define o relacionamento

Orthogonality entre entidades (papéis). A Figura 39 apresenta um exemplo de relacionamento **Orthogonality** ente papéis, definido na ontologia como *OrthogonalityRole*.

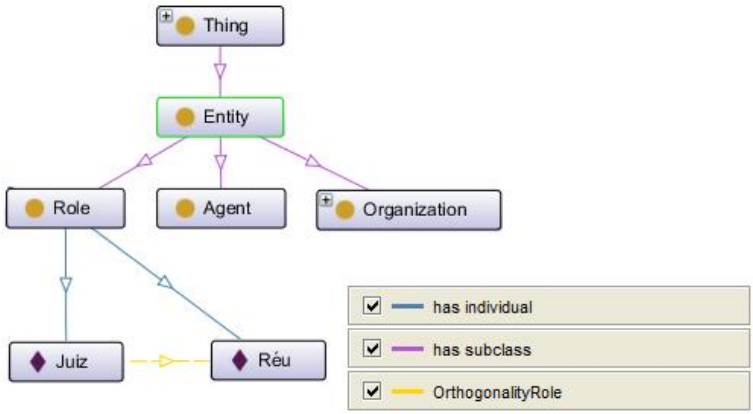

Figura 39: Relacionamento *Orthogonality* entre papéis.

Para que um conflito entre normas ocorra devido o relacionamento **Orthogonality** entre papéis, é necessário que ambas as normas estejam obrigando ou permitindo um agente a desempenhar esses papéis em um mesmo período de tempo. O agente não será capaz de desempenhar ambos os papéis no período de tempo definido, violando assim, pelo menos uma das normas.

A Figura 40 contém o Diagrama de Classe UML com a classe *OrthogonalRole* definida. Em se tratando de classe, a única alteração foi a inserção do novo relacionamento entre papéis, através da classe *OrthogonalRole*. Já nos métodos para extração dos relacionamentos entre os elementos normativos, a única diferença foi a inserção do método *getOrthogonalRole()*.

A estrutura das classes e métodos não sofre muitas alterações. No entanto, o método *checkEntity()* deverá sofrer alterações, inserindo condições que correspondam ao caso de conflito normativo devido ao relacionamento entre as entidades ortogonais. Pequenas alterações no modo como os resultados serão apresentados também deverão ocorrer para que o novo relacionamento possa ser apresentado caso o mesmo seja um dos motivadores do conflito. Seguindo o modelo de resultados da verificação de conflitos apresentado na Seção 4.1, o relacionamento **Orthogonality** entre papéis pode ser apresentado conforme abaixo:

```
<(OrthogonalRole) Réu(Role) x Juiz(Role)>
```

A descrição acima é apresentada quando um conflito ocorre e um dos motivos que levaram um par normativo ao conflito é o relacionamento que as mesmas possuem entre entidades ser o de ortogonalidade. Note que neste relacionamento, os conceitos deônticos

devem ser verificados para que um conflito ocorra, já que esse conflito não ocorre entre normas com modalidades opostas (i.e., permissão x proibição ou obrigação x proibição).

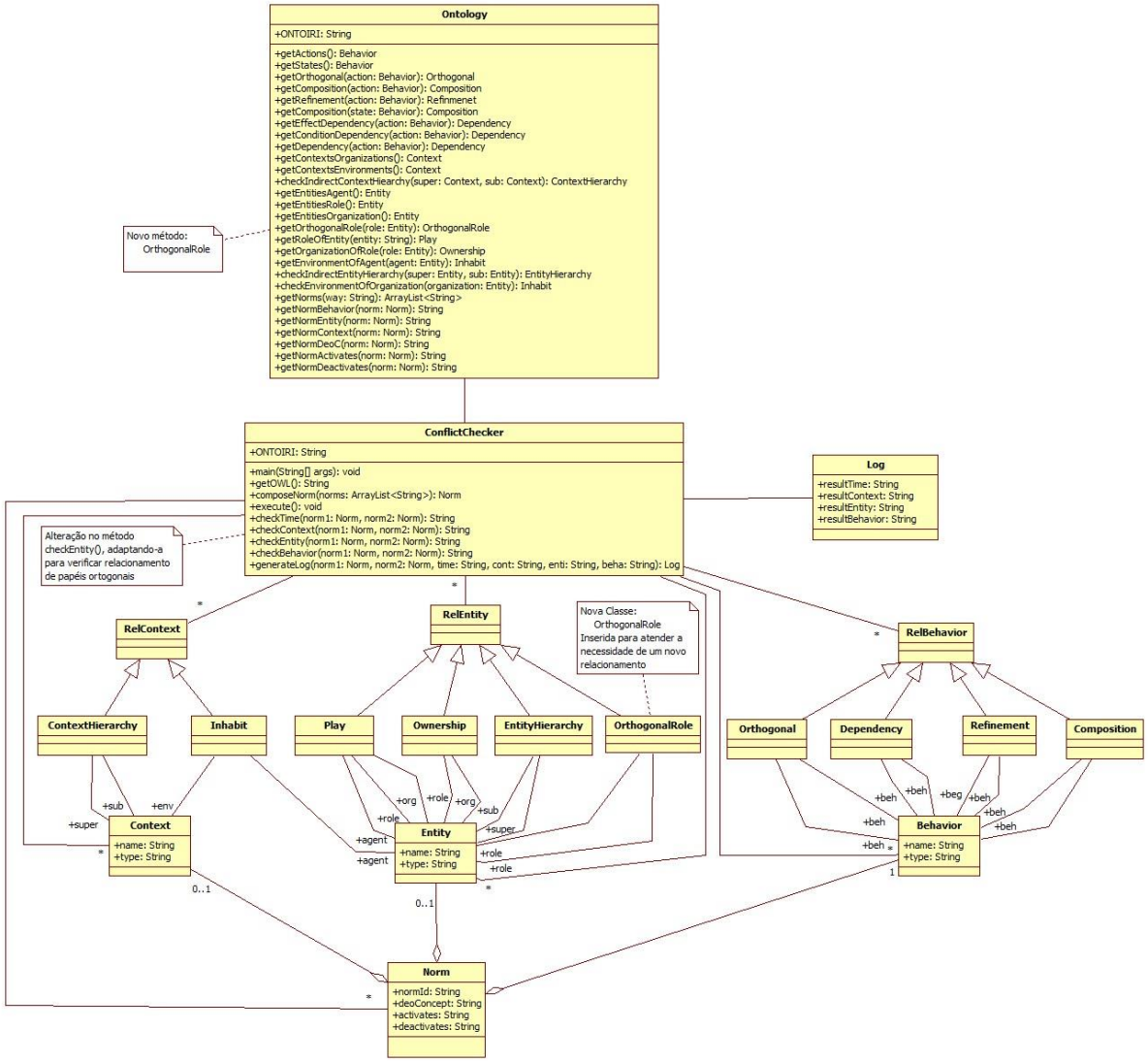

Figura 40: Diagrama de Classe contento a classe *OrthogonalRole*.

No método *getOrthogonalRole*, o código fonte para extrair o relacionamento ortogonal entre dois papeis é apresentado na Figura 41. O método recebe um papel como parâmetro e busca os papéis ortogonais a ele. Caso o papel definido em uma norma seja ortogonal ao papel definido em outra e, caso as modalidades das normas sejam semelhantes (i.e., obrigação x permissão), as entidades possuem o relacionamento **Orthogonal** definido entre elas.

```
OWLNamedIndividual entities = fac.getOWLNamedIndividual(IRI.create(ONTOIRI + "#" + entity));
OWLObjectProperty orthogonalRole = fac.getOWLObjectProperty(IRI.create(ONTOIRI + "#OrthogonalRole"));
NodeSet<OWLNamedIndividual> ontoRole = reasoner.getObjectPropertyValues(entities, orthogonalRole);
Set<OWLNamedIndividual> values = ontoRole.getFlattened();
```

Figura 41: Código-fonte para extrair o relacionamento *Orthogonal* entre papéis.

Através do exemplo apresentado nesta Seção, percebemos que é possível adaptar o *Conflict Checker* a novos tipos de relacionamentos, seja ele entre contextos, entidades ou comportamentos. As regras de verificação de entidades também passam por alterações, verificando se o relacionamento de ortogonalidade entre papeis é existente entre os papéis definidos nas normas.

CAPÍTULO 5 – ESTUDO DE CASO E ANÁLISE DOS RESULTADOS

Neste Capítulo será especificado o estudo de caso proposto para a aplicação de normas e verificação de conflitos normativos. Para isso, foi analisado um contrato real de compra e venda de ações entre duas empresas brasileiras do ramo de aviação civil. Um contrato de compra e venda é um contrato oneroso, entre duas pessoas ou entre várias pessoas, representando duas partes, uma das quais se obriga a transferir o domínio à outra, e esta a pagar-lhe o preço estipulado (NETTO, 2010).

5.1 O CONTRATO DE COMPRA E VENDA

Um contrato de compra e venda de ações visa formalizar um acordo entre vendedor e comprador, tendo como objeto de negócio, ações que representam os valores de uma empresa.

O contrato em questão possui um conjunto de normas que devem ser cumpridas pelas entidades envolvidas. Um número elevado de normas é aplicado sobre qualquer contrato, afim de que todas as exigências e restrições sejam contempladas, não favorecendo quaisquer partes envolvidas no acordo. Um número reduzido e simplificado de normas será utilizado nesse estudo de caso, visando clarificar e simplificar questões que envolvem o direito jurídico.

Como é de se esperar em um contrato de compra e venda de ações consolidado, nenhum conflito existe entre as normas aplicadas sobre tal contrato. Por isso, propomos a inserção de algumas normas contratuais, criando assim, uma espécie de extensão do atual contrato. As Seções seguintes descrevem os elementos existentes no contrato original, assim como as normas inseridas em tal contrato.

5.1.1 OS CONTEXTOS E ENTIDADES

No contrato de compra e venda de ações em questão, os trâmites jurídicos ocorrem no ambiente *América*, que possui dois sub-ambiente: *América do Sul* e *América do Norte*. Esses três estão relacionados pelo relacionamento *Hierarchy* (Figura 26). Esses *ambientes* não são explicitamente especificados no contrato, mas buscou-se cria-los para que todos os relacionamentos entre contextos e entidades sejam expostos nesse estudo de caso.

Duas jurisdições são envolvidas em tal contrato. Os *Estados Unidos da América* (EUA) é uma entidade do tipo organização, que habita o ambiente *América do Norte*. O *Brasil* também é uma entidade do tipo organização, mas que habita o ambiente *América do Sul*.

O Fórum responsável pelos trâmites jurídicos nos *EUA* é o Fórum do estado de Nova York. O Estado de *Nova York* é uma sub-organização da entidade *EUA*, havendo então um

relacionamento *Hierarchy* entre essas organizações. Do mesmo modo, o estado do *Rio de Janeiro* é uma sub-organização da entidade *Brasil*.

Devido os nomes das empresas envolvidas em tal acordo não contribuírem e não alterarem com os resultados das verificações de conflito, tais nomes serão omitidos. Desse modo, as empresas envolvidas são denominadas *Entidade A* (denominada *vendedor*) e a *Entidade B* (denominado *comprador*).

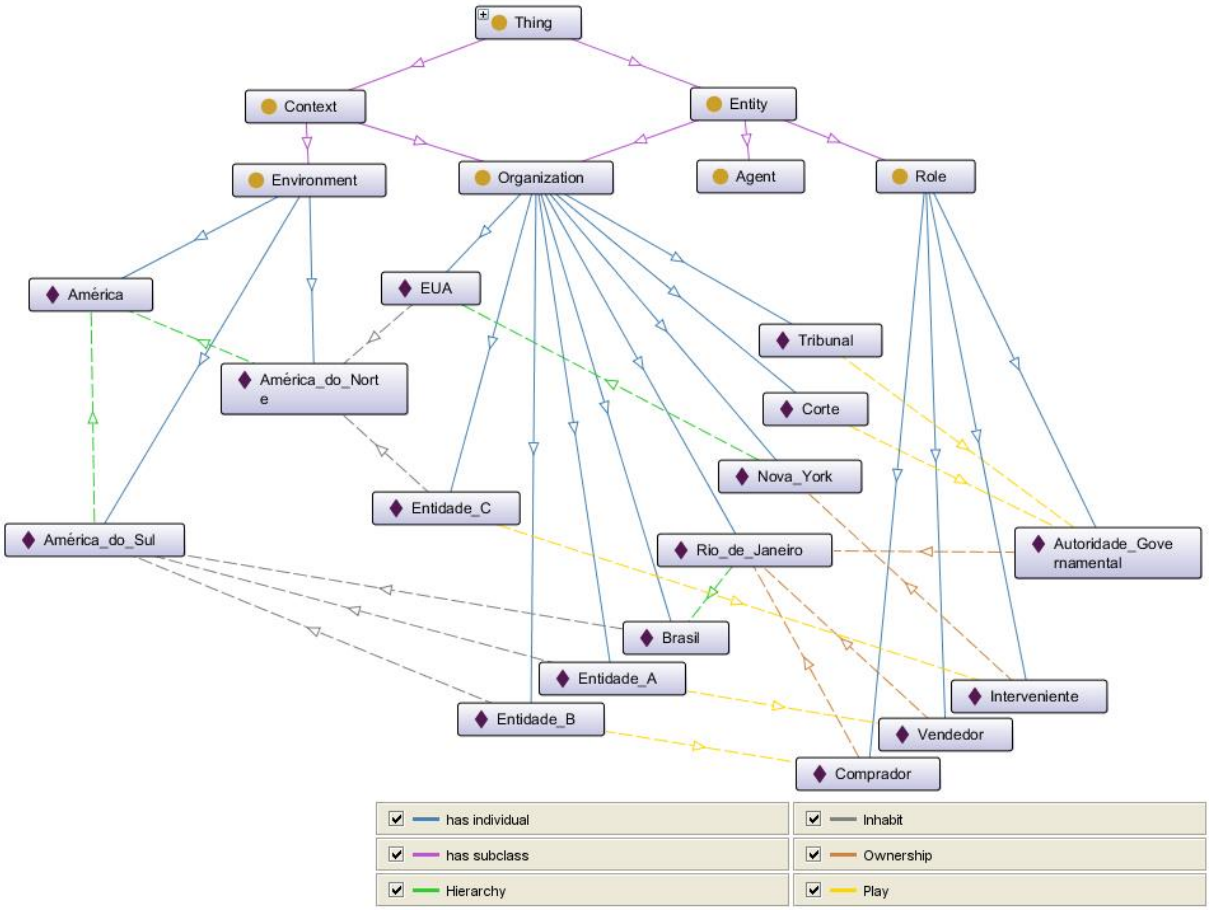

Figura 42: Contextos e Entidades do contrato de compra e venda de ações.

As entidades *Entidade A* e *Entidade B* habitam o ambiente *América do Sul*. Os papéis *Comprador* e *Vendedor* são papéis desempenhados na jurisdição brasileira. Desse modo, esses papéis se relacionam com a organização *Rio de Janeiro* (RJ) através do **Ownership**.

Existe no contrato, uma organização interveniente entre o *comprador* e o *vendedor*. De acordo com NETTO (2010), um interveniente é um terceiro que paga a letra de câmbio em lugar do sacado, ou que faz o pagamento por qualquer dos coobrigados. Ele é um interessado que paga a dívida em seu nome. Em outras palavras, tal empresa é o intermediador entre as partes envolvidas na negociata. A *Interveniente* é uma entidade que habita o ambiente *América*

do Norte e desempenha papel sob a jurisdição da organização *Nova York* (NY), cujo nome também será omitido, sendo aqui chamado de *Entidade C*.

Para finalizar a identificação das entidades e contextos, no contrato são mencionadas as entidades *Tribunal* e *Corte*. Essas entidades são mencionadas no contrato como uma *Autoridade Governamental* (papel). Diversas outras entidades foram definidas como *Autoridade Governamental*, mas para a realização desse estudo de caso, as duas entidades mencionadas são suficientes. *Brasil* é a organização onde a *Autoridade Governamental* é desempenhada.

5.1.2 OS ESTADOS E AÇÕES

Em um contrato, normas são definidas sobre escopos de atuação (*contextos*), sobre entidades cujos comportamentos estão sendo regulados e sobre os comportamentos que podem, não podem e devem ser executados. Nesta Seção, é apresentado um conjunto de ações e estados contidos no contrato, assim como os relacionamentos existentes entre eles.

O contrato apresenta a ação *cooperar*, tendo em vista que as entidades devem *cooperar* para que o comum acordo seja legítimo. No próprio contrato é definido que *cooperar* é uma *Super Ação* e que possui diversas *Sub Ações*, tais como *disponibilizar informações*, *disponibilizar documentos*, *envidar esforços*, entre outras. Com isso, possuímos o primeiro conjunto de ações que apresentam o relacionamento **Refinement** entre elas, conforme.

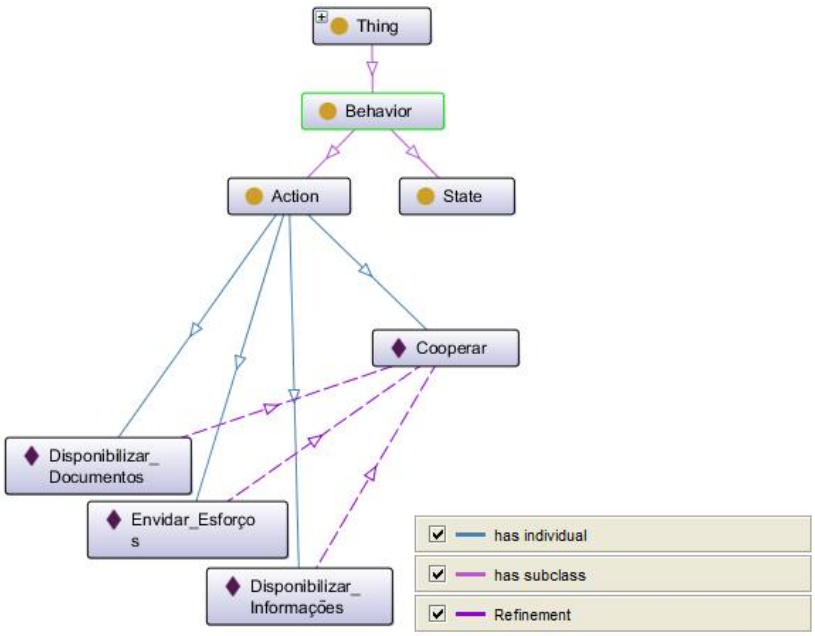

Figura 43: Relacionamento *Refinement* no contrato.

Existem um conjunto de estados e ações conhecidos como *Atos de fechamento*. Esse conjunto de comportamentos seguem uma ordem cronológica e levam ao cumprimento do

acordo estipulado no contrato de compra e venda de ações. Os *Atos de fechamento* ocorrem na a organização *Brasil* da seguinte forma:

- A vendedora transfere as ações (*transferir ações*) para a compradora – atingindo o estado de *ações transferidas*.
- A compradora transfere os valores de compra (*transferir valores de compra*) – atingindo o estado de *valores transferidos*.
- A compradora e vendedora celebram o contrato (*celebrar contrato*) – atingindo o estado *contrato celebrado*.

Os *Atos de fechamento* aqui apresentados foram simplificados, resumindo-se apenas em três ações e três estados atingidos devido a execução de tais ações. É importante lembrar que essas ações são dependentes, seguindo a ordem cronológica apresenta na lista de marcadores acima e que os estados devem ser atingidos, afim de que as ações tenham condição de serem executadas, conforme definido na Seção 3.3.3.2. As ações e estados que apresentam o relacionamento **Dependency** definidos entre elas são apresentados conforme a Figura 44.

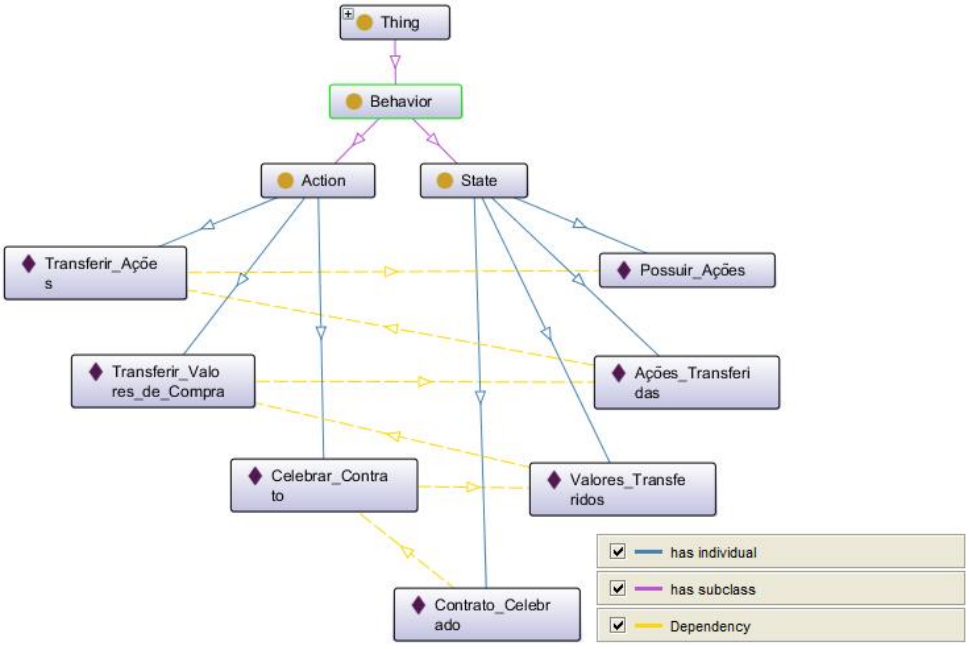

Figura 44: Relacionamento *Dependency* no contrato.

É possível definir o relacionamento **Composition** entre as ações e estados listados acima com a ação *Atos de fechamento*. No entanto, preferimos por não realizar tal relacionamento, definindo o relacionamento de composição em ações que não apresentam relacionamentos definidos até agora.

Seguindo ainda as ações que envolvem os *Atos de Fechamento*, a ação *celebrar contrato* é uma ação *todo* e que possui ações parte, tais como *concordar contrato* e *assinar contrato*. As duas ações *parte* devem ser executadas para que as metas da ação *inteira* sejam executadas, devido o relacionamento **Composition** entre as ações (Figura 45).

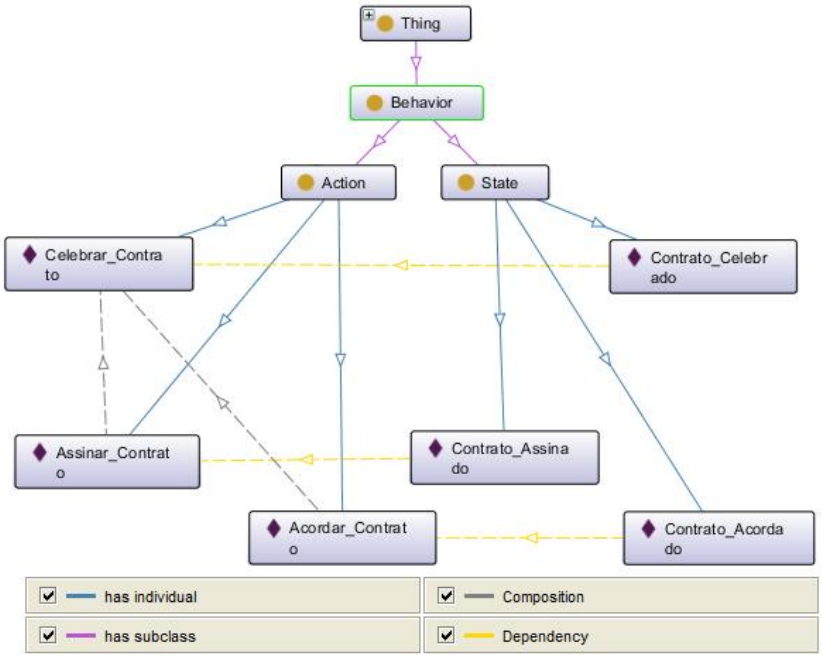

Figura 45: Relacionamento *Composition* no contrato.

É possível afirmar que uma ação *parte* de uma ação *todo* leva a um estado *parte* de um estado *todo*. O estado *Contrato Celebrado* é um estado *todo*, efeito da ação *todo Celebrar Contrato*. Assim, não é necessário definir o relacionamento **Composition** entre estados, a menos que um estado possua estados *parte* e o relacionamento **Dependency** não seja definido com ações. Note que na Figura 45 o estado *Contrato Celebrado* não apresenta um relacionamento **Composition** com os estados *parte Contrato Acordado* e *Contrato Assinado*. No entanto, devido ao fato de que ações *todo* levam ao estado *todo* e que ações *parte* levam a estados *parte*, é possível afirmar que o relacionamento **Composition** está definido entre *Contrato Celebrado* (estado *todo*), *Contrato Acordado* (estado *parte*) e *Contrato Assinado* (estado *parte*).

Para representar o relacionamento **Orthogonality**, buscou-se por esses termos que incitem ações. Assim, é possível levantar um conjunto de ações que são ortogonais. A ação *concordar*, por exemplo, é ortogonal a *discordar*, i.e., são ações opostas e que não podem ser executadas em um mesmo período de tempo. Um exemplo de estados ortogonais é *acordado* (estado efeito da ação *concordar*) é ortogonal ao *discordado* (estado efeito da ação *discordar*).

Como já mencionado nesta dissertação, ações ortogonais levam a estados ortogonais. Com isso, os estados não precisam ser representados, pois conhecendo o ciclo de ações, é possível identificar quais estados são ortogonais.

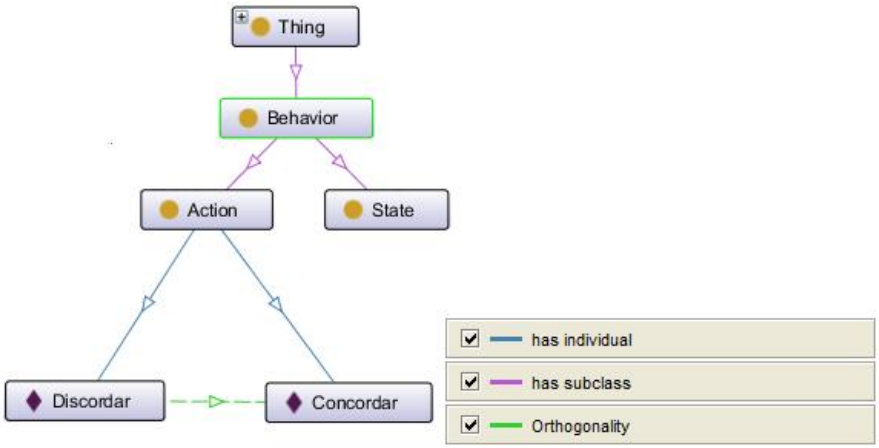

Figura 46: Relacionamento *Orthogonality* no contrato.

5.2 AS NORMAS CONTIDAS NO CONTRATO

Um contrato contém um conjunto de normas (regras) que devem ser seguidas, para que o comum acordo seja alcançado. No contrato de compra e venda em questão, dezenas de normas foram aplicadas, restringindo o comportamento das entidades envolvidas. O número de normas descritas no contrato é elevado. Devido a isso, reduziu-se esse número para um conjunto de normas que contemplem os tipos de conflitos normativos descritos nesta dissertação. Neste Capítulo são apresentadas as normas extraídas do contrato de compra e venda de ações.

O contrato diz que o *Interveniente* deverá *concordar* com os termos aplicados no contrato, no ato de assinatura do mesmo. Assim, a norma é uma *Obrigação*, aplicada sobre o papel *Interveniente* e no contexto *EUA*, regendo o comportamento *concordar*, conforme abaixo:

```
N = {NormN1, OBLIGED, EUA, Interveniente, Concordar,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

O contrato também afirma que o *interveniente* deve *cooperar* para o bom firmamento do contrato, não estipulando data limite para isso. Desse modo, afirma-se que a norma se aplica sobre o contexto *Nova York*, obrigando todas as entidades que desempenham o papel *interveniente* de *cooperar* em todo o período de tempo (norma permanentemente ativa).

```
N = {NormN2, OBLIGED, Nova York, Interveniente, Cooperar, _, _}
```

A entidade *Brasil* deve *cooperar* com as *Autoridades Governamentais* para que o contrato seja celebrado no bom cumprimento da lei. A *NormN3* é aplicada sobre o contexto

América do Sul, obrigando o *Brasil* a *Cooperar* no âmbito do contrato, não especificando os períodos de ativação e desativação da norma.

```
N = {NormN3, OBLIGED, América do Sul, Brasil, Cooperar, _, _}
```

O contrato permite que o *Comprador* assine o contrato no dia acordado para sua celebração do mesmo (desde que todos os requisitos tenham sido cumpridos). Assim, existe uma *permissão* para *Celebrar o Contrato*, definido no contexto *América do Sul*, no dia 09 de novembro de 2005.

```
N = {NormN4, PERMITTED, América do Sul, Comprador, Celebrar Contrato,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

No contrato, a primeira exigência sobre a entidade *vendedora* é garantir a existência das ações no ato da celebração do contrato, ou seja, no dia 09 de novembro de 2005.

```
N = {NormN5, OBLIGED, Brasil, Vendedor, Possuir Ações,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

A *NormN6* (é aplicada sobre o *Vendedor*, afirmando que a entidade deve *transferir suas ações* no dia 09 de novembro de 2005.

```
N = {NormN6, OBLIGED, Brasil, Vendedor, Transferir Ações,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

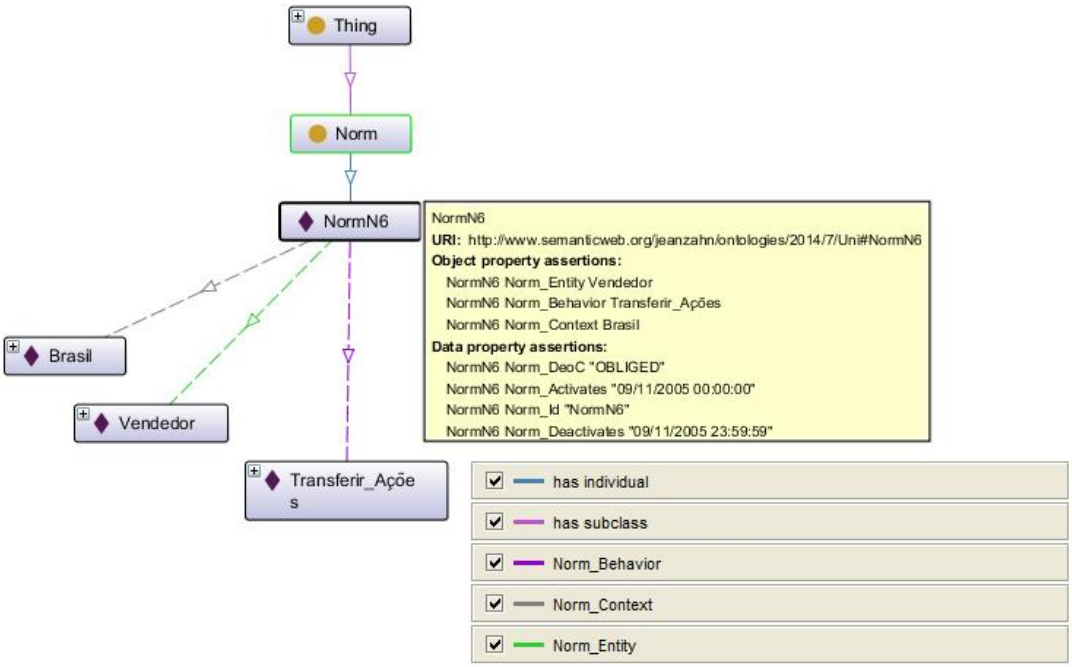

Figura 47: Um exemplo de norma do contrato, a NormN6.

As normas até aqui apresentadas são apenas algumas das inúmeras existentes no contrato. Buscou-se por normas simples e que não envolvessem conceitos jurídicos complexos. A proposta é apresentar um caso de conflito para cada tipo de conflito normativo exposto nesse trabalho.

5.3 INSERINDO NOVAS NORMAS

Como é de se esperar em um contrato de compra e venda consolidado, conflitos entre as normas definidas em tal contrato é inexistente. Em nenhum momento, o cumprimento de uma norma acarreta na violação de outra norma. No entanto, ao inserir novas normas em um contexto, deve-se atentar para o surgimento de conflitos normativos. Assim, nesta Seção será inserido um conjunto de normas com o intuito de avaliar a proposta de verificação de conflitos normativos indiretos apresentada nesta dissertação.

A primeira norma inserida no domínio da aplicação é a *NormN7*, proibindo a *Entidade C* de *disponibilizar informações* na data de celebração do contrato. *Disponibilizar informações* é tornar público toda e qualquer informação que comprometa ou que seja necessário para a celebração contratual.

```
N = {NormN7, OBLIGED, Nova York, Entidade C, Discordar,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

A *NormN8* proíbe que informações sejam disponibilizadas pelo estado de *Nova York*, do período em que o contrato foi assinado em diante.

```
N = {NormN8, FORBIDDEN, América, Nova York, Disponibilizar Informações,
                                                    09/11/2005 00:00:00, _}
```

A norma 9 (*NormN9*) proíbe, no contexto de *Brasil*, o estado do *Rio de Janeiro cooperar* para que o contrato seja celebrado, sendo a norma válida para a data de celebração do contrato.

```
N = {NormN9, FORBIDDEN, Brasil, Rio de Janeiro, Cooperar,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

A *NormN10* é uma proibição aplicada no contexto de *América do Sul*, restringindo a *Entidade B* de *Acordar o Contrato* entre as datas do dia 21 de junho de 2003 a 30/03/2009.

```
N = {NormN10, FORBIDDEN, América do Sul, Entidade B, Acordar Contrato,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

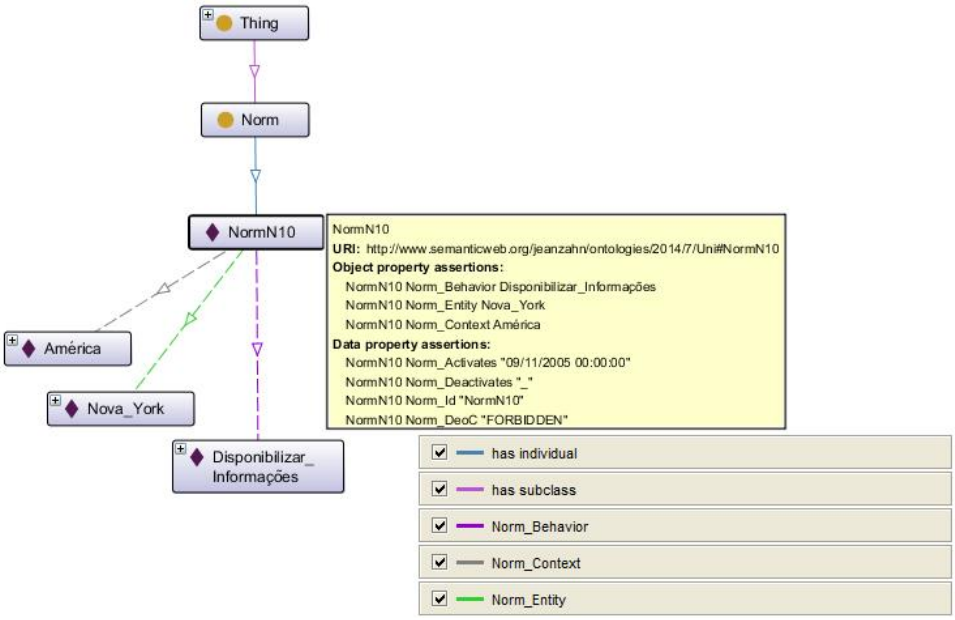

Figura 48: Um exemplo de norma inserida no domínio, a NormN10.

Uma proibição aplicada sobre a *Entidade A* é definida na *NormN11*. A norma informa que no contexto de *Rio de Janeiro*, tal entidade está proibida de possuir ações na data de celebração do contrato.

```
N = {NormN11, FORBIDDEN, Rio de Janeiro, Entidade A, Possuir Ações,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

A *NormN12* permite que a *Entidade A* possua ações judiciais pendentes, numa tentativa de acelerar o processo de compra e venda das ações. Essa norma é aplicada sobre todo o contexto de *América do Sul* e é válida somente para a data de celebração do contrato.

```
N = {NormN12, FORBIDDEN, América do Sul, Vendedor, Ações Transferidas,
                                    09/11/2005 00:00:00, 09/11/2005 23:59:59}
```

Na Seção seguinte, será apresentado o *Conflict Checker* agindo sobre o conjunto de ações expostas até aqui.

5.4 O CONFLICT CHECKER E A EXISTÊNCIA DE CONFLITO

Nesta Seção serão apresentados os casos de conflitos normativos detectados pelo *Conflict Checker*. As normas definidas no contrato foram extraídas e inseridas na ontologia. Do mesmo modo, as normas geradas aleatoriamente também foram inseridas na ontologia e, posteriormente, analisadas através do *Conflict Checker*.

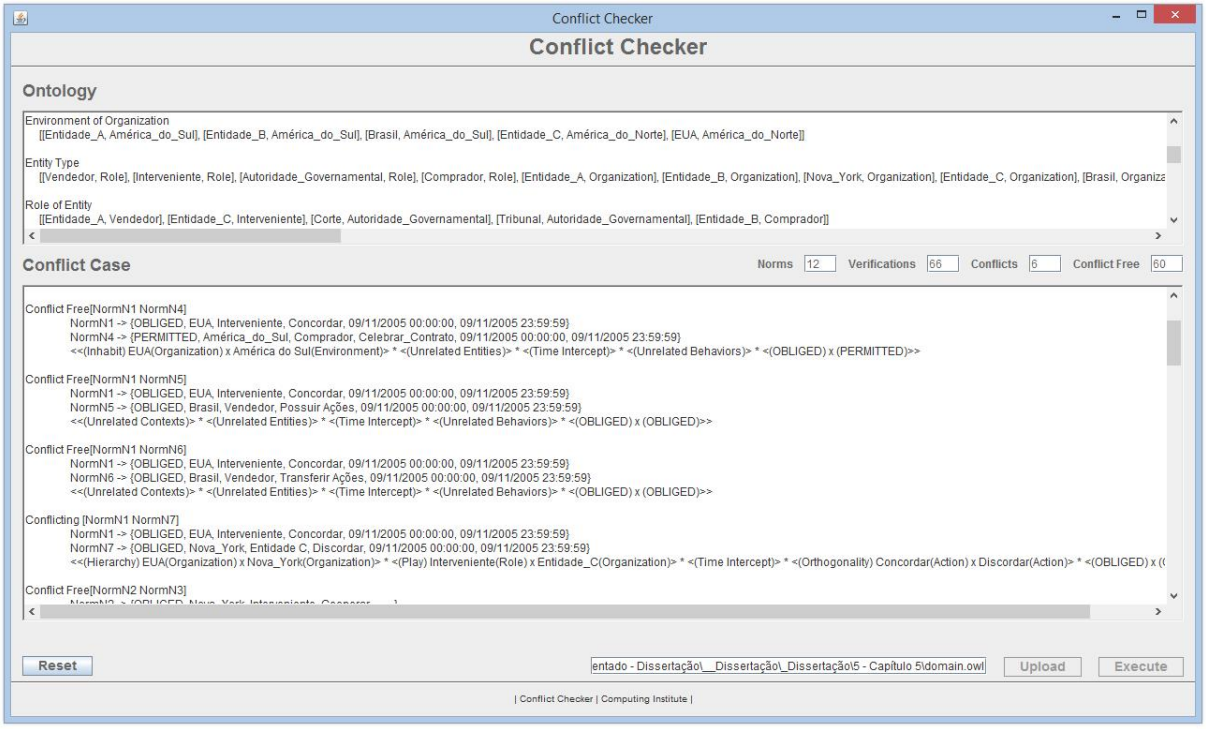

Figura 49: O resultado das verificações do *Conflict Checker*.

O estudo de caso foi constituído de um total de 12 normas, sendo 6 delas extraídas do contrato de compra e venda de ações, e 6 geradas para a verificação de conflitos normativos utilizando o *Conflict Checker*.

Conforme a Figura 49, o *Conflict Checker* realizou 66 verificações de casos de conflitos, sendo que 6 pares normativos apresentaram conflito e 60 não apresentaram conflito. Esses números demonstram que os conflitos existentes foram oriundos das 6 normas inseridas para demonstrar a aplicabilidade da abordagem.

Diante dos resultados, é importante compreender os motivos que levaram um par normativo a estar em conflito. Para isso, analisaremos os 6 pares normativos conflitantes. O primeiro conflito detectado pelo *Conflict Checker* ocorreu entre as normas *NormN1* e *NormN7*, conforme abaixo:

```
Conflicting [NormN1 NormN7]
      NormN1 -> {OBLIGED, EUA, Interveniente, Concordar, 09/11/2005
00:00:00, 09/11/2005 23:59:59}

      NormN7 -> {OBLIGED, Nova_York, Entidade C, Discordar, 09/11/2005
00:00:00, 09/11/2005 23:59:59}

<<(Hierarchy) EUA(Organization) x Nova_York(Organization)> *
 <(Play) Interveniente(Role) x Entidade_C(Organization)> *
 <(Time Intercept)> *
 <(Orthogonality) Concordar(Action) x Discordar(Action)> *
 <(OBLIGED) x (OBLIGED)>>
```

De acordo com o *Conflict Checker*, a norma *NormN1* e a *NormN7* apresentam conflito pois os elementos de ambas as normas se relacionam. Os contextos estão relacionados pelo relacionamento **Hierarchy** e, de acordo com a regra de propagação, a norma definida no contexto de uma organização, tal norma se aplica a todas as sub-organizações de tal organização, conforme a Definição 3 do Capítulo 3. A *Entidade C* (organização) desempenha o papel contido na *NormaN1* e, devido a regra de propagação da norma relativa ao relacionamento **Play** (Seção 3.3.2.1), a norma que se aplica sobre um papel, ela se propaga sobre todas as entidades que desempenham esse papel (Definição 7). Os tempos de ativação e desativação de cada norma são relacionados, conforme sinalizado pelo *Conflict Checker* como `Time Intercept`. O relacionamento **Orthogonality** (Seção 3.3.3.4, Definição 23) é o tipo de relacionamento existente entre as ações e, devido os conceitos deônticos das normas serem obrigações, o conflito é detectado. Note que os elementos das normas são diferentes (contextos diferentes, entidades diferentes e comportamentos diferentes), mas através da obtenção do conhecimento do domínio (o contexto *Nova York* é um subcontexto do contexto *EUA*, a *Entidade C* desempenha o papel de *Interveniente* e *Concordar* e *Discordar* são ações ortogonais) os conflitos indiretos podem ser detectados.

A norma *NormN2* apresenta um conflito com a norma *NormN8*. Os contextos envolvidos no par normativo são relacionados através do relacionamento **Inhabit**, pois quando uma norma é aplicada sobre um ambiente, ela se propaga sobre os agentes e organizações que habitam tal ambiente (Definição 5). As entidades estão relacionadas pelo relacionamento **Ownership**, já que o *Interveniente* é um papel desempenhado na organização *Nova York*. Desse modo, a norma que se aplica sobre a organização *Nova York* se propaga sobre todos os papéis desempenhados em tal organização (Definição 11). O relacionamento entre os comportamentos é o **Refinement** (Seção 3.3.3.1, Definição 13), que afirma que uma super ação (*cooperar*) é obrigatória e sua respectiva sub ação (*disponibilizar informações*) é proibitiva, um conflito é detectado. Note que nenhuma norma permitindo a entidade de executar qualquer outra sub ação foi aplicada. O período de tempo possui intersecção, pois quando o período de tempo em uma norma não é definido, isso significa que tal norma está ativa permanentemente (Seção 3.3.4).

```
Conflicting [NormN2 NormN8]
      NormN2 -> {OBLIGED, Nova_York, Interveniente, Cooperar, _, _}

      NormN8 -> {FORBIDDEN, América, Nova_York, Disponibilizar_Informações,
09/11/2005 00:00:00, _}

<<(Inhabit) Nova_York(Organization) x América(Environment)> *
 <(Ownership) Interveniente(Role) x Nova_York(Organization)> *
 <(Time Intercept)> *
 <(Refinement) Cooperar(Action) x Disponibilizar_Informações(Action)> *
 <(OBLIGED) x (FORBIDDEN)>>
```

As normas *NormN3* e *NormN9* apresentam conflito devido ao relacionamento **Inhabit** estar presente entre os contextos definidos na norma. O contexto *Brasil* habita no contexto *América do Sul* e, segundo a regra de propagação da norma (Definição 5), quando uma norma se aplica sobre um ambiente, tal norma se aplica sobre todas os agentes (ou organizações) que habitam tal ambiente. O relacionamento entre as entidades é o **Hierarchy** e, segundo a regra de propagação da norma para tal relacionamento, quando a norma se aplica sobre uma organização, ela se propaga a todas as sub organizações de tal organização (Definição 9). Os períodos de ativação e desativação das normas apresentam relacionamento, considerando que a *NormN3* é permanentemente ativa. Os comportamentos regidos pela norma são os mesmo e, devido as modalidades das normas serem opostas (i.e., obrigação x proibição), o conflito é definido entre tais normas.

```
Conflicting [NormN3 NormN9]
      NormN3 -> {OBLIGED, América_do_Sul, Brasil, Cooperar, _, _}

      NormN9 -> {FORBIDDEN, Brasil, Rio_de_Janeiro, Cooperar, 09/11/2005
00:00:00, 09/11/2005 23:59:59}

<<(Inhabit) América_do_Sul(Environment) x Brasil(Organization)> *
 <(Hierarchy) Brasil(Organization) x Rio_de_Janeiro(Organization)> *
 <(Time Intercept)> *
 <(Same Behavior)> *
 <(OBLIGED) x (FORBIDDEN)>>
```

As normas *NormN4* e *NormN10* apresentam conflito. As normas são aplicadas sobre o mesmo contexto, o ambiente *América do Sul*. O relacionamento **Play** é definido entre as entidades e, conforme a regra de propagação da norma (Definição 7), quando a norma se aplica sobre um determinado papel, tal norma se aplica sobre todos os agentes (ou organizações) que desempenham tal papel. Os períodos de tempo de ativação e desativação da norma são exatamente os mesmos, havendo então interceptação entre eles. O relacionamento **Composition** é definido entre os comportamentos. A ação *todo Celebrar Contrato* está sendo obrigatória e a ação *parte* de tal ação *todo* está sendo proibida. De acordo com a regra de

propagação da norma para o relacionamento **Composition** (Definição 20), tais normas apresentam um conflito entre elas.

```
Conflicting [NormN4 NormN10]
      NormN4 -> {PERMITTED, América_do_Sul, Comprador, Celebrar_Contrato,
09/11/2005 00:00:00, 09/11/2005 23:59:59}

      NormN10 -> {FORBIDDEN, América_do_Sul, Entidade B, Acordar_Contrato,
09/11/2005 00:00:00, 09/11/2005 23:59:59}

<<(Same Contexts)> *
 <(Play) Comprador(Role) x Entidade_B(Organization)> *
 <(Time Intercept)> *
 <(Composition) Celebrar_Contrato(Action) x Acordar_Contrato(Action)> *
 <(PERMITTED) x (FORBIDDEN)>>
```

Um conflito ocorre entre as normas *NormN5* e *NormN11*. O relacionamento **Hierarchy** está presente entre os contextos. De acordo com a regra de propagação da norma para o relacionamento **Hierarchy**, a norma que se aplica ao contexto de uma organização, se aplica sobre todos os sub contextos de tal organização (Definição 3). Novamente, o relacionamento **Play** é definido entre duas entidades. Quando a norma se aplica sobre um papel, tal norma se aplica sobre todos os agentes (ou organizações) que desempenham tal papel (Definição 8). Os períodos de ativação e desativação da norma são relacionados. Devido os comportamentos definidos nas normas serem os mesmos e as modalidades das normas serem opostas (i.e., obrigação e proibição), o conflito entre as normas é afirmado.

```
Conflicting [NormN5 NormN11]
      NormN5 -> {OBLIGED, Brasil, Vendedor, Possuir Ações, 09/11/2005
00:00:00, 09/11/2005 23:59:59}

      NormN11 -> {FORBIDDEN, Rio de Janeiro, Entidade A, Possuir Ações,
09/11/2005 00:00:00, 09/11/2005 23:59:59}

<<(Hierarchy) Brasil(Organization) x Rio de Janeiro(Organization)> *
 <(Play) Vendendor(Role) x Entidade A(Organization)> *
 <(Time Intercept)> *
 <(Same Behavior)> *
 <(OBLIGED) x (FORBIDDEN)>>
```

Por fim, as normas *NormN6* e *NormN7* apresentam o último conflito normativo entre as 12 normas apresentadas nesse estudo de caso. Os contextos das normas são relacionados pelo relacionamento de **Inhabit** e, dada a propagação da norma para o relacionamento **Inhabit**, quando a norma se aplica sobre um ambiente, tal norma se aplica sobre todos os agentes (ou organizações) que habitam tal ambiente (Definição 5).

```
Conflicting [NormN6 NormN12]
      NormN6 -> {OBLIGED, Brasil, Vendedor, Transferir Ações, 09/11/2005
00:00:00, 09/11/2005 23:59:59}

      NormN12 -> {FORBIDDEN, América do Sul, Vendedor, Ações Transferidas,
09/11/2005 00:00:00, 09/11/2005 23:59:59}

<<(Inhabit) Brasil(Organization) x América do Sul(Environment)> *
 <(Same Entities)> *
 <(Time Intercept)> *
 <(Dependency) Transferir Ações(Action) x Ações Transferidas(State)> *
 <(OBLIGED) x (FORBIDDEN)>>
```

As entidades especificadas nas normas são as mesmas e os períodos de ativação e desativação da norma apresentam interceptação entre eles. Quando uma norma proíbe o alcance de um estado dependente e outra norma obriga a execução de uma ação, cujo efeito é o estado sendo proibido, tais normas apresentam conflito (Definição 18).

CAPÍTULO 6 – TRABALHOS RELACIONADOS

O problema de conflitos entre normas em SMA tem gerado inúmeras pesquisas na comunidade científica. Existem diversos trabalhos na literatura que investigam sobre a verificação de conflitos normativos em SMA. Alguns deles propõem, além da verificação de conflitos, um mecanismo para a resolução dos mesmos. Nesse Capítulo é exposto um conjunto de trabalhos relevantes sobre o tema, bem como a comparação com o trabalho até aqui apresentado.

6.1 KOLLINGBAUM ET AL. (2006)

Os autores apresentam no artigo uma abordagem de verificação de conflitos entre normas em Organizações Virtuais abertas (OV). Nesse tipo de organização, conflitos entre normas é um problema recorrente. Os agentes que habitam esse tipo de entidade devem ser capaz de resolver esse tipo de conflito para que possam se manter operacionais.

A arquitetura NoA contribui com o agente na identificação de conflitos normativos e contribui com o agente na renegociação de contratos e normas. NoA é inspirado no modelo BDI[7] clássico (WOOLDRIGE, 2002), mas que apresenta algumas características: (i) em NoA, normas são entidades de primeira classe que influenciam no raciocínio prático de um agente e (ii) uma maneira específica de deliberação chamada *informed deliberation*, que possibilita o agente a identificar e resolver conflitos de maneira eficiente. Outro aspecto importante de NoA é fazer agentes robustos contra a potencial ameaça de conflitos. Com isso, NoA garante, segundo os autores, que os agentes mantenha-se operacionais quando se deparam com um conflito normativo.

Uma característica importante dos autores desse artigo é que os mesmos fazem a diferenciação entre *conflito* e *inconsistência* (conforme apresentado na Seção 2.3.1). É comum esse tipo diferenciação ocorrer em artigos desses autores (Seção 6.2 e Seção 6.3). No trabalho, em todo o momento, os autores afirmam detectar *conflitos* entre normas, apesar dos exemplos utilizados gerarem uma *inconsistência* e não um *conflito* entre as normas (proibição x obrigação). Na abordagem dessa dissertação, não é feita a diferenciação entre *conflito* e *inconsistência* entre as normas. O que é *inconsistência* neste artigo, é considerado como *conflito* nesta dissertação.

[7] Teoria filosófica de raciocínio prático, proposta inicialmente por BRATMAN; ISRAEL; POLLACK (1988). Modelo específico para o desenvolvimento de agentes inteligentes que atende os critérios utilizados na definição de SMA cognitivos.

Os autores neste artigo não formalizam um modelo normativo que rege o comportamento das entidades. No entanto, os autores apresentam em seus exemplos uma norma com a seguinte estrutura:

$$n = \{conceito\ de\hat{o}ntico\ (entidade, a\tilde{c}\tilde{a}o)\}$$

A norma consiste de um conceito deôntico (obrigação, proibição e permissão), uma entidade e uma ação. Diferente dessa abordagem, nesta dissertação é definido um contexto (delimitando a área de atuação da norma, Seção 3.1.2) e o período em que a norma permanece ativa (Seção 3.1.5). O período de ativação e desativação da norma não é definido neste artigo, devido não ser o foco da abordagem dos autores.

Um conflito entre duas normas, neste artigo, ocorre quando o *escopo de influência* de uma norma sobrepõe o *escopo de influência* da outra. Cada norma possui o seu *escopo de influência*, tal escopo é definido nas ações do agente. Estes escopos podem interceder, podendo indicar um relacionamento de especialização (*Refinement*, Seção 3.3.3.1). Note que apenas um tipo de relacionamentos é definido entre as ações. Nesta dissertação são expostos quatro tipos de relacionamento entre ações e estados. Outra limitação do artigo é a não verificação de conflitos normativos que ocorrem quando normas do mesmo tipo são verificadas (i.e., obrigação x obrigação).

Neste artigo não é apresentado um meio de detectar conflitos normativos considerando o domínio da aplicação e suas características. Diferente disso, nossa abordagem apresenta um mecanismo capaz de receber uma ontologia, reconhecer os elementos e relacionamentos contidos nessa ontologia e determinar os casos de conflitos presentes no domínio.

Além de detectar conflitos, o artigo apresenta um mecanismo para resolver os mesmos (através da renegociação de contratos e normas). No entanto, resolver conflitos normativos não é o foco desta dissertação.

6.2 VASCONCELOS ET AL. (2007)

Neste artigo os autores apresentam uma abordagem de verificação de conflitos normativos no escopo de OV governados por normas. Esse tipo de organização define, governa e facilita na coordenação do compartilhamento de recursos e resolve problemas em sociedades de agentes. A abordagem possui como foco organizações virtuais como SMA, em que agentes humanos e de software interagem para atingir metas individuais e globais.

Assim como a abordagem da Seção 6.1, essa abordagem faz diferenciação entre *conflito* e *inconsistência*. Conforme já mencionado nesta dissertação, quando uma ação é

simultaneamente permitida e proibida, um *conflito* surge entre as normas. Quando uma ação é simultaneamente obrigatória e proibida, é definido uma *inconsistência* entre as normas.

O artigo apresenta um modelo normativo onde a norma ω é uma tupla, definida conforme abaixo:

$$\omega \{\nu, t_d, t_a, t_e\} \text{ tal qual } \nu,$$

- $\mathbf{O}_{\alpha:\rho\varphi}$ o Γ (é uma obrigação),
- $\mathbf{P}_{\alpha:\rho\varphi}$ o Γ (é uma permissão),
- $\mathbf{F}_{\alpha:\rho\varphi}$ o Γ (é uma proibição),

onde α e ρ são termos, φ o Γ é uma fórmula atômica de primeira ordem com restrições; t_d, t_a, $t_e \in \boldsymbol{IN}$ são, respectivamente, o tempo quando ν é introduzida, quando ν é ativada e quando ν expira, $t_d \leq t_a, \leq t_e$. O termo α identifica o agente da norma e o termo ρ é o papel desempenhado pelo agente. Assim, a norma utilizada pelos autores possui um conceito deôntico, uma entidade, um papel, uma ação e fatores de ativação e desativação.

O modelo normativo utilizado nesse artigo não define um contexto, ou seja, não define a área de atuação da norma. A abordagem apresentada nessa dissertação considera a definição de um contexto na norma, delimitando a área de atuação da mesma. Desse modo, uma entidade só deve cumprir com a norma se a mesma estiver presente no contexto especificado pela norma.

No artigo é utilizado o conceito de *unificação* dos termos de primeira ordem para a verificação da existência de *conflitos* e *inconsistência*. A unificação das normas permite analisar se uma norma sobrepõe a outra, através da influência que expressa uma sobre as outras. Em nossa abordagem, um conflito é detectado através da *propagação da norma* (Seção 3.3). O mecanismo de detecção de conflitos se utiliza de um método diferente do apresentado no artigo, pois a proposta desse trabalho é a verificação dos relacionamentos existentes entre os elementos definidos na norma, através da obtenção do conhecimento sobre o domínio da aplicação.

Os autores demonstram como o mecanismo de verificação de conflitos – até então capaz de verificar apenas conflitos diretos – pode ser adaptado para a detecção de conflitos indiretos. O mecanismo adaptado é capaz de detectar conflitos entre ações diferentes, através da *reescrita* da norma, mas não leva em consideração o esquema das ações (*estado como condição da ação*, *ação*, *estado como efeito da ação*). Apesar de eficiente, a ausência da consideração de estados como comportamentos das entidades demonstra ser uma limitação deste artigo. O trabalho dos

autores também não considera a existência de conflitos entre normas que são simultaneamente *obrigadas* ou *permitidas*. Esse tipo de conflito existe, conforme apresentado na Seção 3.3.3.4.

6.3 KOLLINGBAUM ET AL. (2007)

Abordagens orientadas a organizações para a formação de SMA assumem que uma comunidade de agentes formam uma Organização Virtual. Esta proposta facilita o compartilhamento de recursos e a solução de problemas entre software e/ou agentes humanos. OV são definidas por um conjunto de papéis, relacionamento entre papeis e normas descrevendo obrigações e restrições sociais para um agente que desempenha um papel. A proposta dos autores busca por detectar conflitos normativos no contexto de OV. Esse tipo de organização, quando situados em um mundo que constantemente sofre mudanças, deve sofrer adaptações para se adaptar às mudanças. Essas mudanças naturais afetam o *design* do agente, fazendo com que as posições normativas de um agente podem mudar, fazendo com que conflitos possam surgir.

Através da utilização de um cenário de *Grid Service*, este trabalho trata da ocorrência de conflitos normativos e como agentes podem se mandar operativos quando um conflito entre normas surge. Tal abordagem faz a distinção do que é um *conflito* e o que é uma *inconsistência*. Um *conflito* surge quando um agente é simultaneamente permitido e proibido de realizar uma ação. Uma *inconsistência* surge quando um agente é simultaneamente obrigado e proibido de realizar uma ação. Esse tipo de diferenciação não é considerado nesta dissertação, pois em nossa abordagem, tudo se resume em conflitos normativos.

Neste artigo, a abordagem especifica os agentes sob a *Arquitetura NoA*, devido os agentes operar com um mecanismo de *planning* reativo, onde as capacidades do agente são expressadas em um conjunto de planos especificados previamente. As normas são especificadas na linguagem de especificação da norma *NoA*, que provê construtores para especificar obrigações, proibições e permissões. O modelo normativo utilizado neste artigo segue a seguinte estrutura:

$$n = \{conceito\ deôntico\ (entidade, ação)\}$$

Assim como na abordagem da Seção 6.1, neste artigo um *conflito*/*inconsistência* ocorre quando o escopo de influência de uma norma sobrepõe o escopo de influência de outra norma.

No artigo é apresentado três tipos de conflitos: *Containment*, *Intersection* e *Indirect Conflict/Inconsistency*. O conflito *Containment* ocorre quando o escopo de uma norma está

contido no escopo de outra norma, podendo haver o relacionamento de especialização (*Refinement*) entre as normas. O conflito *Intersection* ocorre entre normas que apresentam intersecção entre os escopos de influência (não existe o relacionamento de especialização para esse caso de conflito). O conflito *Indirect Conflict/Inconsistency* ocorre entre ações são obrigatórias e os estados que são **efeitos** de tal ação são proibidas, e vice-versa (apresentamos esse tipo de conflito como *Dependency* entre os comportamentos – Seção 3.3.3.2). Diferente dessa abordagem, a única distinção entre tipos de conflitos apresentado nesta dissertação é o de conflitos diretos e indiretos. O que buscamos apresentar nesta atual abordagem foram os motivos que levaram um par normativo a apresentar conflitos entre elas. Além disso, consideramos conflitos que ocorrem entre ações, através da dependência que uma ação pode ter sobre a outra, considerando também que existe um estado que faz a ligação entre tais ações.

O ponto chave da abordagem dos autores constitui em definir *níveis de consistências*: *Strong Consistency* (todas as obrigações são consistentes), *Weak Consistency* (nem todas as obrigações apresentam consistência entre elas) e *Inconsistency* (nenhuma obrigação é consistente). Através da obtenção do nível de consistência de uma norma, o agente pode realizar a renegociação dos contratos. O objetivo da renegociação de contratos é criar ou estender um conjunto de opções ao agente. Com isso, o agente pode renegociar um conjunto de normas, estendendo ou reduzindo o escopo de influência ou sobrescrevendo proibições. Não entraremos no mérito da renegociação de contratos, pois isso apresenta características de resolução de conflitos entre normas. É importante lembrar que resolver conflitos normativos não faz parte dos objetivos desta dissertação. É possível notar que o modo de verificar conflitos normativos são distintos, bem como a forma de considerar um conflito. Outro fator que diferencial grandemente a abordagem deste artigo com a dissertação até aqui apresentada é a capacidade de verificar relacionamentos entre contextos e entidades. O artigo não apresenta um método para a verificação de relacionamentos entre entidades. De fato, o artigo está focado em apresentar um mecanismo capaz de detectar conflitos/inconsistências entre ações (e eventualmente entre os estados **efeito** de tais ações).

Finalizando o estudo sobre este artigo, os autores propõem uma técnica de rotular as ações, numa tentativa de acumular informações sobre normas conflitantes em relação com uma ação. Essa técnica propõe que informações sobre uma ação sejam armazenas em uma tupla

$$l = \{\alpha, MOTIVATORS, PROHIBITORS\}$$

onde α é a ação rotulada, $MOTIVATORS$ é um conjunto de obrigações que motiva a consideração da ação como uma candidata para a execução e $PROHIBITORS$ é um conjunto de proibições ou obrigações que conflitam com todas as obrigações contidas no conjunto de *MOTIVATORS*. Se o conjunto de *PROHIBITORS* é vazio, então é considerado que uma obrigação é, ao menos, *Weakly Consistent.* Se o conjunto de *PROHIBITORS* é vazio em todas os rótulos, onde uma obrigações ocorre no conjunto de *MOTIVATORS*, então a obrigação é *Strongly Consistent*. Isso torna possível verificar o nível de consistência de obrigações. Esta dissertação não busca conflitos normativos através dos rótulos que podem ser aplicados sobre cada uma das ações, mas sim sobre os relacionamentos que os comportamentos (estados e ações) apresentam entre si (definidos na ontologia), bem como os relacionamentos entre entidade, contexto e intersecção de tempo de ativação de uma norma.

6.4 VASCONCELOS ET AL. (2009)

Neste artigo, os autores apresentam mecanismos para a detecção e resolução de conflitos normativos. Tais mecanismos são baseados na *unificação de primeira ordem* e técnicas de resolução de restrições. Através de sofisticados algoritmos, o artigo apresenta um gerenciamento de posições normativas, que consiste na adoção e remoção de permissões, obrigações e proibições em sociedades de agentes. A preocupação dos autores está voltada em como manter as normas como sendo um meio de excluir comportamentos antissociais sem alterar o design de agentes ou restringir sua autonomia. Para isso, é formalizado um conceito prático de autoridade e modelos de conflitos que podem surgir como resultado de delegações.

O modelo normativo utilizado neste artigo é semelhante ao modelo utilizado no artigo da Seção 6.2 (este artigo é uma espécie de continuação do artigo apresentado na Seção 6.2). Uma norma ω é uma tupla, definida conforme abaixo:

$$\omega \{\nu, t_d, t_a, t_e\} \text{ tal qual } \nu,$$

- $\mathbf{O}_{\alpha:\rho\varphi}$ o Γ (é uma obrigação),
- $\mathbf{P}_{\alpha:\rho\varphi}$ o Γ (é uma permissão),
- $\mathbf{F}_{\alpha:\rho\varphi}$ o Γ (é uma proibição),

onde α e ρ são termos, φ o Γ é uma fórmula atômica de primeira ordem com restrições; $t_d, t_a, t_e \in \boldsymbol{IN}$ são, respectivamente, o tempo quando ν é introduzida, quando ν é ativada e quando ν expira, $t_d \leq t_a \leq t_e$. O termo α identifica o agente da norma e o termo ρ é o papel

desempenhado pelo agente. Assim, a norma utilizada pelos autores possui um conceito deôntico, uma entidade, um papel, uma ação e fatores de ativação e desativação.

O modelo normativo não define um contexto, ou seja, não define a área de atuação da norma. Diferente deste artigo, no modelo normativo dessa dissertação é composta por um contexto, que define e restringe a área de atuação de uma norma. Assim, a entidade deve cumprir com a norma somente se estiver presente em tal contexto. A norma também se aplica somente sobre ações, impossibilitando assim definir restrições sobre um estado. Outra grande diferença entre os modelos normativos utilizados em ambas abordagens, está relacionado com entidades. As entidades consideradas neste artigo são os agentes e o papéis, não sendo considerados organizações. As normas podem ser aplicadas sobre um grupo de agentes (organizações) e o verificador de conflitos deve ser capaz de verificar se existe algum relacionamento entre, por exemplo, normas aplicadas a um papel e normas aplicadas sobre uma organização (relacionamento **Ownership**, Seção 3.3.2.3).

Para a detecção de conflitos normativos, os autores utilizam a análise de sobreposição de escopos de influência. Assim, quando o escopo de influência de uma norma sobrepõe o escopo de influência de outra, um conflito é constatado. Em outras palavras, um conflito normativo surge, de acordo com a abordagem dos autores, quando uma ação é simultaneamente proibida e obrigada/permitida, e suas variáveis tem sobreposição de valores. Note que neste artigo, os autores não consideram a possibilidade de existência de conflitos normativos entre modalidades semelhantes (i.e., permissão x permissão, obrigação x obrigação e permissão x obrigação). Nesta dissertação, conforme apresentado na Seção 3.3.3.4, conflitos podem ocorrer entre normas que possuem modalidades semelhantes.

No artigo também é apresentado o problema de conflitos normativos indiretos, que são dependentes de domínio. No entanto, a abordagem dos autores consideram apenas conflitos que ocorrem entre ações relacionadas, não considerando assim, o relacionamento existente entre ações e estados, conforme apresentado na Figura 6, Seção 3.1.4. Essa é uma limitação importante do artigo e que a solução é expressada nesta dissertação. Verificar conflitos entre estados e ações só é possível quando consideramos as características do domínio da aplicação.

O artigo apresenta ainda, um meio de resolução dos conflitos detectados através da abordagem de verificação de conflitos. Verificar conflitos não faz parte do escopo desta dissertação, mas é uma de nossas preocupações para esforços futuros.

6.5 APHALE ET AL. (2013)

O trabalho intitulado "*Goal Directed Conflict Resolution and Policy Refinement*" apresenta os desafios de prover um suporte automatizado para identificação e resolução de conflitos lógicos e conflitos funcionais. Com isso, os autores apresentam um modelo de identificação e resolução de conflitos, focando nos conflitos que são mais críticos às metas e objetivos de uma organização.

Um *Logical Conflicts* pode surgir entre as políticas (normas). Quando a mesma ação é proibida e permitida/obrigada em um mesmo período de tempo, levando a entidade ao conflito quando cumprir com uma das políticas. Um *Functional Conflicts* ocorre entre as definições contidas nas políticas com as definições contidas nas metas da organização.

Segundo os autores, políticas são normas e podem ser classificadas em duas categorias: políticas coletivas e políticas individuais. As políticas coletivas, segundo a abordagem dos autores, são aplicadas sobre um conjunto de entidades em um grupo particular. Nesta dissertação, também é considerado normas aplicadas sobre um conjunto de entidades: normas aplicadas sobre uma *organização*. Políticas individuais são aplicadas sobre agentes ou entidades que possuem preferências particulares. Do mesmo modo, esta dissertação verifica conflitos normativos para normas aplicadas sobre uma única entidade, i.e., um agente.

O desafio dos autores é: como identificar os mais relevantes conflitos normativos em um domínio e/ou organização? Isso é difícil de resolver, devido que políticas podem entrar em conflito umas com as outras (*Logical Conflicts*), bem como com os objetivos organizacionais (*Functional Conflicts*) em muitas situações.

Isso é possível através do *Activity Prioritisation Model*. Esse modelo analisa a importância relativa de cada atividade através do peso que uma atividade possui. Essa importância (ou peso que uma atividade possui) é computada de acordo com a probabilidade de execução de cada atividade sobre o alcance de determinada meta, através do custo médio do alcance de uma meta e a probabilidade de sucesso na execução da atividade. Através desse modelo, é possível identificar as atividades mais relevantes em um *Planning*, que geram os conflitos de maior relevância e que são prioridade na abordagem dos autores. Nesta dissertação, não é feita a distinção entre conflitos que ocorrem entre as normas e conflitos que ocorrem entre normas e entre os objetivos de uma organização. Na verdade, esse tipo de diferenciação não faz parte da proposta desse trabalho.

A OWL-POLAR é usada como uma linguagem para representar as políticas na abordagem dos autores. A utilização dessa linguagem se deve a capacidade que a mesma possui

em especificar políticas em aplicações de tempo real. Para detectar quais as atividades que possuem maior prioridade para a organização (*Functional Conflicts*), é utilizado um modelo definido como "*Activity Prioritisation Model*", que visa identificar as atividades prioritárias em um determinado domínio e/ou organização.

Na proposta dessa dissertação não se encontra o ato de resolver conflitos normativos. Esse é uma das importantes diferenças entre ambos os trabalhos. Outra diferença considerável é essa dissertação não considerar *Functional Conflicts*, devido não existir a verificação de prioridade entre as atividades presentes no domínio ou em uma organização. No entanto, a proposta dos autores não expressa o interesse em verificação de conflitos que ocorrem através do relacionamento existente entre uma ação e um estado, conforme proposto nessa dissertação.

6.6 SILVA, ZAHN (2013)

O trabalho apresentado por (SILVA; ZAHN, 2013), consiste um conjunto de passos a serem dados, afim de que conflitos entre pares de normas sejam detectados. O trabalho apresenta um conjunto de algoritmos para a verificação de conflitos normativos que regulam diferentes ações, porém relacionadas.

Nesse trabalho são definidos quatro relacionamentos entre ações – *Refinement*, *Dependency*, *Composition* e *Orthogonality*. O artigo também considera os relacionamentos existentes entre contextos e entidades, estendendo o trabalho apresentado em SILVA; ZAHN (2013). Os relacionamentos entre contextos e entidades são: *Inhabit*, *Play*, *PlayIn* e *Ownership*.

Os autores apresentam uma estrutura normativa com um elevado nível de detalhamento dos elementos. Segundo os autores, uma norma *n* é composta por um conceito deôntico (*deoC*) de um conjunto {*obrigação* ou *proibição*}, um contexto *c* (pertencente a um conjunto de contextos) onde a norma é definida, uma entidade *e* (de um conjunto de entidades) que é a entidade que deverá cumprir com a norma, uma ação *a* (de um conjunto de ações) que está sendo regulada, uma condição de ativação *ac* (de um conjunto de condições) que é o fator de ativação da norma, uma condição de desativação da norma *dc* e, por fim, o status *s* da norma (*cumprida*, *violada* ou *nulo*). Assim, esse completo modelo normativo possui a seguinte estrutura:

$$n = \{deoC, c, e, a, ac, dc, s\}.$$

Note que existe uma grande semelhança entre o modelo de norma apresentada pelos autores com o modelo de norma apresentado nesta dissertação. Isso se deve ao embasamento do modelo da atual abordagem com o modelo utilizado em SILVA e ZAHN (2013) e em ZAHN

e SILVA (2014). A diferença dos modelos normativos utilizados na atual pesquisa e na pesquisa dos autores é a inserção de um identificador para a norma (*id*), a capacidade de se definir um *estado* na norma e a retirada do *status*.

A proposta dos autores se diferencia das propostas apresentadas nesta dissertação em diversos pontos. Primeiramente, a proposta de SILVA e ZAHN (2013) busca detectar conflitos normativos considerando apenas ações como comportamento regido pela norma. Nesta dissertação é considerado os relacionamentos definidos entre estados e ações, devido o relacionamento direto que um estado possui com uma ação, e vice-versa (o ciclo dos estados e ações, Seção 3.1.4). Nesta dissertação é apresentado um mecanismo completo de verificação de conflitos normativos, desde a elaboração de um modelo padrão de ontologia, bem como o desenvolvimento de um protótipo capaz de verificar a existência de conflitos partindo dessa ontologia.

CAPÍTULO 7 – CONCLUSÃO E TRABALHOS FUTUROS

Neste capítulo é apresentado o resumo da pesquisa apresentada nessa dissertação, suas contribuições, limitações e sugestões de trabalhos futuros que possam contribuir com a continuidade dessa abordagem.

7.1 RESUMO DO TRABALHO

Os SMA são sistemas computacionais compostos por agentes de software capazes de atingir objetivos através da interação. Um agente de software é uma entidade situada em um determinado ambiente e que é capaz de executar ações de maneira autônoma, afim de atingir um determinado objetivo (WOOLDRIDGE, 2009). Em outras palavras, agentes de software são programas autônomos (ou autossuficientes), sendo capazes de executar ações para atingir seus objetivos e de acordo com a percepção que o mesmo possui sobre o ambiente.

A autonomia e a heterogeneidade dos agentes de software exige que o comportamento de tais agentes sejam regulados por um sistema normativo. Sistemas normativos são compostos por um conjunto de normas (ou regras) que visam instruir o agente sobre aquilo que ele pode fazer (permissão), sobre aquilo que não pode fazer (proibição) e sobre aquilo que deve ser feito (obrigação).

No contexto de SMA, normas são o meio descritivo para governar o comportamento dos agentes, em um determinado contexto e período. De acordo com JIANG et al. (2013), as normas possuem sujeitos que devem cumprir com a mesma.

Devido ao elevado número de normas que podem ser aplicadas sobre um mesmo SMA, conflitos normativos podem surgir. Um conflito entre duas normas ocorre quando o cumprimento de uma norma resulta na violação de outra, e vice-versa. Neste trabalho, consideramos a existência de dois tipos de conflitos normativos: conflitos diretos e conflitos indiretos.

Os **conflitos diretos** surgem quando os conceitos deônticos são opostos (por exemplo, obrigação e proibição, ou permissão e proibição), quando as normas são aplicadas sobre uma mesma entidade, sobre um mesmo comportamento, em um mesmo contexto e quando os eventos de ativação e desativação das normas implicam em uma interseção temporal.

Os **conflitos indiretos** surgem quando os elementos de uma norma não são iguais aos respectivos elementos contidos em outra norma. No entanto, os elementos de ambas normas são relacionados. Se as normas forem analisadas sem considerar os relacionamentos entre os elementos, o conflito entre elas não é identificado. Porém, através de uma verificação detalhada

dos elementos que compõem as normas e dos relacionamentos existentes entre estes elementos, o conflito pode ser desvendado. A primeira diferença entre conflitos normativos diretos e conflitos normativos indiretos é que, o segundo, não busca apenas por conflitos existentes entre normas aplicadas sobre entidades, contextos e comportamentos iguais.

Diversos trabalhos têm proposto mecanismos de verificação de conflitos normativos (conforme apresentado no Capítulo 6). No entanto, esses mecanismos têm se demonstrado ineficientes devido não levarem em consideração as características do domínio da aplicação. Isso implica diretamente na capacidade de verificação de conflitos normativos indiretos.

Este trabalho propõe um mecanismo baseado em regras para a verificação de conflitos normativos diretos e indiretos, levando em consideração as características presentes no domínio da aplicação. Esse mecanismo – o *Conflict Checker* – possui três etapas fundamentais em seu funcionamento. A primeira visa receber um documento em formato OWL, contendo a representação do conhecimento de um determinado domínio. A segunda etapa tem por objetivo aplicar as regras de verificação de conflitos normativos e, por fim, a terceira etapa apresenta os resultados para os casos que apresentam conflitos normativos, bem como os resultados para os pares normativos que não apresentam conflitos.

Para o desenvolvimento do mecanismo, um conjunto de regras de verificação de conflitos normativos foi desenvolvido. Essas regras descrevem as condições para que duas normas apresentem conflitos (i.e., quando elementos contidos em um par normativo são diferentes, mas possuem algum tipo de relacionamento definido entre tais elementos).

A abordagem foi avaliada por meio de um estudo de caso elaborado a partir da extração de normas (regras) contidas em um contrato real de compra e venda de ações entre empresas de aviação civil. Tais normas foram organizadas em uma ontologia OWL e, posteriormente, utilizadas para avaliação dos resultados no *Conflict Checker*. Cada verificação de conflito normativo gera um pequeno resultado descritivo, informando os motivos pelas quais um par normativo apresenta (ou não) um conflito.

Os resultados obtidos com o estudo de caso foram importantes, devido demonstrarem a eficiência das regras de verificação de conflitos indiretos, sobretudo quando obtém-se o conhecimento sobre o domínio da aplicação. Foi possível demonstrar que, dado um conjunto de normas, juntamente com o conhecimento do domínio da aplicação (conhecimento obtido através de ontologias), o *Conflict Checker* possui a capacidade de verificar, analisar e concluir sobre a existência de conflitos entre pares normativos em SMA.

7.2 CONTRIBUIÇÕES

Esta dissertação contribui com uma proposta de verificação de conflitos normativos diretos e indiretos. O seu grande diferencial das demais abordagens é a capacidade de afirmar a existência de conflitos através da obtenção do conhecimento sobre um determinado domínio. Segue abaixo um conjunto com as principais contribuições desta pesquisa:

1. Um modelo de normas que especifica um conceito deôntico, contexto, entidade, comportamento, período de ativação e desativação.
2. A identificação e definição de relacionamentos entre contextos, entidades e comportamentos.
3. A definição da estrutura de uma ontologia OWL, contemplando o modelo de normas definido no Item 1 e os relacionamentos definidos no Item 2.
4. Definição de um conjunto de regras de detecção de conflitos normativos indiretos.
5. Uma ferramenta que possibilita a identificação de conflitos entre normas descritas na ontologia levando em consideração os relacionamentos entre os elementos do domínio.

7.3 LIMITAÇÕES

Esta dissertação apresenta algumas limitações que devem ser levadas em consideração. O problema de conflitos normativos é complexo e exige um cuidado minucioso. Abaixo estão descritas as limitações que envolvem essa pesquisa:

- **Pares normativos**: Neste trabalho, as normas são analisadas em pares. Isso pode ser ineficiente em alguns casos. I.e., quando uma norma obriga um número x de agentes a consumir um número $x - 1$ de recursos consumíveis. A definição de norma nesta abordagem não leva em consideração os objetivos/recursos ao qual a ação se aplica. Alguns conflitos só podem ser detectados quando analisamos grupos de normas, e não duplas.
- **Número simultâneo de documentos OWL**: Atualmente, o *Conflict Checker* é capaz de analisar um único documento OWL por vez. Toda vez que uma análise é finalizada, o *Conflict Checker* deve ser reiniciado (através do botão *reset*). Isso impossibilita uma eventual utilização de dois documentos OWL contendo, por exemplo, um o conjunto de normas e outro o conjunto de relacionamentos.

- **Período de ativação de uma norma**: Neste trabalho, consideramos apenas o fator data como ativação e desativação de uma norma. Existem eventos de ativação, tais como o cumprimento de uma norma, a violação de uma norma e a execução de uma ação, que não podem ser considerados nesse trabalho. O Conflict Checker é uma ferramenta para ser utilizada em tempo de design e, por tanto, não possui informação sobre a execução do sistema.

7.4 TRABALHOS FUTUROS

Esta dissertação aponta para algumas sugestões de trabalhos futuros, estendendo ainda mais a capacidade do *Conflict Checker* em verificar conflitos normativos indiretos, tornando-o cada vez mais eficiente. Abaixo, são apresentadas as perspectivas para a continuidade dessa pesquisa:

1. Conforme já mencionado, este trabalho verifica conflitos normativos de duas em duas normas. Uma importante evolução nessa pesquisa será a de verificar os conflitos normativos através da análise de um número maior de normas simultaneamente.
2. O *Conflict Checker* é capaz de verificar apenas um documento OWL por vez. Ser capaz de verificar dois ou mais arquivos OWL de maneira simultânea pode ser interessante (i.e., um documento OWL contendo as normas e um documento OWL contendo os relacionamentos). Essa capacidade de verificar dois ou mais documentos OWL simultaneamente pode ser interessante também quando existe a necessidade de comparação entre dois (ou mais) conjuntos de normas (i.e., sobre um documento OWL que especifica os relacionamentos entre os elementos normativos, quais dos conjuntos de normas apresentam menos conflitos).
3. Neste trabalho, os fator único fator de ativação e desativação considerado foi o fator data (i.e., a data de ativação e desativação, respectivamente, *12/03/2014 12:00:00* e *12/04/2014 12:00:00*). No entanto, como mencionado na Seção 2.2.1, o fator de ativação e desativação de uma norma pode ser a execução de uma ação, o cumprimento ou violação de uma norma, entre outros fatores. Em alguns casos, tais como a execução de uma ação, é necessário obter o conhecimento do sistema em tempo real. Através disso, o *Conflict Checker* poderia detectar conflitos normativos considerando outros fatores que tornam uma norma ativa, além do que já foi apresentada nesta abordagem.

4. Um número considerável de relacionamentos entre os elementos que compõem uma norma foi definido nesse trabalho. No entanto, novos relacionamentos pode surgir. A cada novo relacionamento inserido no *Conflict Checker*, torna-o mais completo e eficiente. É importante ressaltar que o *Conflict Checker* é expansível e que novos relacionamentos podem ser inseridos, conforme definido no Capítulo 4.

É importante ressaltar que a ontologia pode ser usada na definição formal de conflito, associando a identificação de inconsistência com a identificação de conflitos normativos. Apesar das limitações e da necessidade de continuidade nas pesquisas envolvendo conflitos normativos, acredita-se que essa pesquisa tenha contribuído diretamente para a solução de parte dos problemas que envolvem o tema.

REFERÊNCIAS

BECHHOFER, S. **OWL: Web ontology language**. In: Encyclopedia of Database Systems. Springer, p. 2008–2009, 2009.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. **The semantic web**. Scientific American Magazine, 2001.

BORDINI, R. H.; VIEIRA, R. **Linguagens de programação orientadas a agentes: uma introdução baseada em AgentSpeak(L)**. Revista de informática teórica e aplicada. Porto Alegre. Vol.10, n.1, p.7-38. 2003.

BRATMAN, M. E.; ISRAEL, D. J.; POLLACK, M. E. **Plans And Resource-Bounded Practical Reasoning**. [s.l: s.n.].

CHANDRASEKARAN, B.; JOSEPHSON, J. R.; BENJAMINS, V. R. **What are ontologies, and why do we need them?** IEEE Intelligent systems, v. 14, n. 1, p. 20–26, 1999.

DEUTSCH, M.; GERARD, H. B. **A study of normative and informational social influences upon individual judgment**. In: The Journal of Abnormal and Social Psychology, v. 51, n. 3, p. 629–636, 1955.

DIGNUM, V. **The role of organization in agent systems**. In: Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models, p. 1–16, 2009.

ESTEVA, M. et al. **AMELI: An Agent-Based Middleware for Electronic Institutions** Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1. 2004.

GUARINO, N.; OBERLE, D.; STEFFEN, F. **What is an Ontology?** In: a handbook on ontologies, p. 1-7, 2009

FERREIRA, A. B. DE H.; FERREIRA, M. B.; ANJOS, M. DOS. **Novo dicionário Aurélio da língua portuguesa**. Curitiba: Editora Positivo, 2009.

FIGUEIREDO, K. DA S. **Modeling and validating norms in multi-agent systems**. 137 f. Dissertação (Mestrado em Computação) - Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro. 2011.

GRUBER, T. R. **Ontology**. In: Liu, L; Özsu, M. T, editors, Encyclopedia of Database Systems. Springer-Verlag, 2008.

HORRIDGE, M.; BECHHOFER, S. **The OWL-API: A Java API for OWL ontologies**. In: Semantic Web, v. 2, n. 1, p. 11–21, 2011.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; VAN HARMELEN, F. **From SHIQ and RDF to OWL: The making of a web ontology language**. In: Web semantics: science, services and agents on the World Wide Web, v. 1, n. 1, p. 7–26, 2003.

HUHNS, M. N.; SINGH, M. P. **Multiagent systems in information-rich environments**. In: Cooperative Information Agents II Learning, Mobility and Electronic Commerce for

Information Discovery on the Internet. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 79–93, 1998.

JENNINGS, N. R. **On agent-based software engineering**. In: Artificial Intelligence, v. 117, n. 2, p. 277–296, 2000.

JIANG, J. et al. **Norm Contextualization**. In: Coordination, Organizations, Institutions, and Norms in Agent Systems VIII. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 141–157, 2013.

KOLLINGBAUM, M. J. et al. Norm Conflicts and Inconsistencies in Virtual Organisations. In: NORIEGA, P. et al. (Eds.). . **Coordination, Organizations, Institutions, and Norms in Agent Systems II**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2007. p. 245–258.

KOLLINGBAUM, M. J.; NORMAN, T. J. **Norm Adoption and Consistency in the NoA Agent Architecture**. In: Programming Multi-Agent Systems. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 169–186, 2004.

KOLLINGBAUM, M. J.; NORMAN, T. J. **Informed Deliberation During Norm-Governed Practical Reasoning**. In: Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 183–197, 2006.

LANÇA, A., SOSA, E., LOZANO-TELLO, A. **Ontology search centered in concepts**. In: Information Systems and Technologies (CISTI), 2010.

LI, T. et al. Detecting Conflicts in Legal Systems. In: MOTOMURA, Y.; BUTLER, A.; BEKKI, D. (Eds.). . **New Frontiers in Artificial Intelligence**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2013. p. 174–189.

LÓPEZ, F. L. Y. **Social Power and Norms: Impact on Agent Behaviour**. PhD thesis, University of Southampton, Department of Eletronics and Computer Science, 2003.

MEYER, J. J. C.; WIERINGA, R. J. **Deontic Logic in Computer Science: Normative System Specification**. Chichester, UK: John Wiley and Sons Ltd., 1993.

MUSEN, M. A. **Domain ontologies in software engineering: use of Protege with the EON architecture**. In: Methods of Information in Medicine-Methodik der Information in der Medizin, v. 37, n. 4, p. 540–550, 1998.

NETTO, J. O. **Mini Dicionario Juridico Universitario: Terminologia Juridica E Latim Forense**. 2. ed. EDIJUR, 2010.

OREN, N., LUCK, M., MILES, S., NORMAN, T. **An argumentation inspired heuristic for resolving normative conflict**. In: Proceedings of the International Workshop on Coordination, Organisations, Institutions and Norms in Agent Systems (COIN@AAMAS 2008), Estoril, Portugal, 2008.

PROTÉGÉ. **A free, open-source ontology editor and framework for building intelligent systems**. Disponível em: <http://protege.stanford.edu/>. Acesso em: 14 de out. de 2014. [201-?].

RUSSELL, S. **Artificial intelligence : a modern approach**. 2nd ed. ed. Upper Saddle River N.J.: Prentice Hall/Pearson Education, 2003.

RUSSELL, S.; NORVIG, P. **Atificial Intelligence: a modern approach**. In: Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, v. 25, 1995.

SILVA, V. T. DA; ZAHN, J. **Normative Conflicts that Depend on the Domain**. In: Coordination, Organizations, Institutions, and Norms in Agent Systems IX. Lecture Notes in Computer Science. Springer International Publishing, p. 311–326, 2013.

SILVA, V., GARCIA, A., BRANDAO, A., CHAVEZ, C., LUCENA, C., ALENCAR, P. **Taming Agents and Objects in Software Engineering**. In: Software Engineering for Large-Scale Multi-Agent Systems. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 1–26, 2003.

STUDER, R.; BENJAMINS, V. R.; FENSEL, D. **Knowledge engineering: principles and methods**. In: Data & knowledge engineering, v. 25, n. 1, p. 161–197, 1998.

VASCONCELOS, W.; KOLLINGBAUM, M. J.; NORMAN, T. J. **Resolving Conflict and Inconsistency in Norm-regulated Virtual Organizations**Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems. **Anais**...: AAMAS '07.New York, NY, USA: ACM, 2007Disponível em: <http://doi.acm.org/10.1145/1329125.1329236>. Acesso em: 20 fev. 2015

VASCONCELOS, W. W.; KOLLINGBAUM, M. J.; NORMAN, T. J. **Normative conflict resolution in multi-agent systems**. In: Autonomous Agents and Multi-Agent Systems, v. 19, n. 2, p. 124–152, 2009.

WEB, W. S. **W3C Semantic Web Activity**. http://www.w3.org/2001/sw/, acesso em dez. de 2014, [201-?].

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. John Wiley & Sons, 2009.

WOOLDRIGE, M. **An Introduction to MultiAgent Systems**. John Wiley \& Sons, 2002.

ZAHN, J. O.; SILVA, V. T. **On the Checking of Indirect Normative Conflicts**. In: Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações, 2014, Porto Alegre. Anais do Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações, p. 13-24, 2014.