UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

SUZANE CARVALHO DOS SANTOS

APPONTO-PRO: um processo incremental para o aprendizado e povoamento de ontologias de aplicação

São Luís
2014

SUZANE CARVALHO DOS SANTOS

APPONTO-PRO: um processo incremental para o aprendizado e povoamento de ontologias de aplicação

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão para a obtenção do grau de Mestre em Engenharia de Eletricidade.

Orientador: Maria del Rosario Girardi

Doutora em Ciências da Computação – UFMA

São Luís

2014

Santos, Suzane Carvalho dos

APPONTO-PRO: um processo incremental para o aprendizado e povoamento de ontologias de aplicação / Suzane Carvalho dos Santos. – São Luís, 2014.

141 f.

Impresso por computador (fotocópia).

Orientador: Maria del Rosario Girardi.

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís, 2014.

1. Ontologias. 2. Aprendizagem de ontologias. 3. Povoamento de ontologias. 4. Integração de ontologias I. Titulo.

CDU 004.82

SUZANE CARVALHO DOS SANTOS

APPONTO-PRO: um processo incremental para o aprendizado e povoamento de ontologias de aplicação

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por SUZANE CARVALHO DOS SANTOS e aprovada pela comissão examinadora.

Aprovada em 18 de agosto de 2014

BANCA EXAMINADORA

Maria del Rosario Girardi (orientador)

Doutora em Ciências da Computação – UFMA

Evandro de Barros Costa

Doutor em Engenharia Elétrica – UFAL

Francisco José da Silva e Silva

Doutor em Ciências da Computação – UFMA

À minha amada mãe e inesquecível avó.

RESUMO

As ontologias são estruturas de representação de conhecimento capazes de expressar um conjunto de entidades de um dado domínio, seus relacionamentos e axiomas, sendo utilizadas pelos modernos Sistemas Baseados em Conhecimento (SBC) no processo de tomada de decisões. No entanto, a construção manual de ontologias é cara e sujeita a erros, sendo uma alternativa viável a sua construção de forma automática. Diversas técnicas e ferramentas têm sido desenvolvidas para aprender os diferentes componentes de uma ontologia a partir de fontes textuais, quais sejam conceitos, hierarquias, instâncias, relacionamentos, propriedades e axiomas. Entretanto estes elementos são, em regra, adquidiros de forma isolada. Devido à carência de abordagens que adquirem todos os elementos de uma ontologia de forma conjunta, surgiu a necessidade de desenvolver um processo que faça o reúso e a aprendizagem de cada um dos elementos de uma ontologia de forma completa. Atendendo a esta necessidade, este trabalho apresenta o Apponto-Pro, um processo incremental para o aprendizado e povoamento de ontologias de aplicação a partir de fontes de informação textuais capaz de gerar uma ontologia completa através da integração de diferentes técnicas que geram elementos da ontologia de forma isolada. O processo foi avalizado através de um estudo de caso que consistiu na construção automática da *Family_Law*, uma ontologia de aplicação no domínio do Direito da Família construída através da aplicação da ferramenta de software Apponto-ProTool, desenvolvida para dar suporte ao processo Apponto-Pro que integrou as ferramentas correspondentes as abordagens contidas no processo. Esta avaliação teve como objetivo verificar a efetividade da ontologia construída pela Apponto-ProTool em relação a uma ontologia construída manualmente por um especialista do domínio e utilizada como ontologia de referência. Para isso foi calculado o valor da medida *"precision"* para os elementos da ontologia construída utilizando a ontologia de referência. Como resultado verificou-se formalmente que em alguns casos a ontologia desenvolvida pela Apponto-ProTool tende a apresentar resultados mais adequados.

Palavras-chaves: Ontologias, Aprendizagem de Ontologias, Povoamento de Ontologias,Integração de Ontologias.

Abstract

Ontologies are knowledge representation structures capable of expressing a set of entities of a domain, their relationships and axioms that are being used by modern knowledge based systems (KBS) in the decision making process. However, manual construction of ontology is expensive and subject to errors, thus a viable alternative is the automation of this process. Several techniques and tools have been developed to learn the different components of an ontology from textual sources, named concepts, hierarchies, instances, relationships, properties and axioms. However, these elements are generally acquired in a isolated manner. Due to the lack of approaches to acquire all the elements of an ontology jointly, there is a need to develop a process to make the reuse and the learning of each of the elements of an ontology in a synergistic manner. To attend this need, this work presents Apponto-Pro, an incremental learning process for populating application ontologies from textual information sources that is capable of generating a complete ontology through the integration of different techniques to generate isolated elements of an ontology. The process was evaluated through a case study that consisted in the automatic construction of *Family_Law*, an application ontology in the field of family law developed with Apponto-ProTool, a software tool to support Apponto-Pro that integrates the approaches that compound the whole process. This evaluation aimed to determine the effectiveness of the ontology constructed with Apponto-ProTool against an ontology manually built by a domain specialist and used as reference ontology. For this reason, the "precision"was calculated for the elements of the ontology automatically generated using the reference ontology. As a result it was found that in some cases the ontology developed with Apponto-ProTool tends to present more suitable results.

Keywords: Ontologies, Ontology Learning, Ontology population, Ontology Integration.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida, pelas oportunidades que me proporcionou e por permitir que mais esta etapa seja concluída.

Agradeço a minha amada mãe, pelos conselhos, orações, palavras de incentivo, paciência em me ouvir e por ser esta mãe tão perfeita.

Agradeço a Neto pelo amor e companheirismo durante todo esse tempo.

Á professora Rosário Girardi pela oportunidade em trabalhar ao seu lado, pelo conhecimento, orientação e ensinamentos que contribuíram para a realização deste trabalho.

Agradeço a Nilson Costa pelo professor, conselheiro e principalmente pelo amigo de todas as horas, a você minha eterna gratidão. E, a amiga que encontrei Adriana Moraes pelo ser humano incrível que é.

Agradeço em especial aos amigos Fhillipi Maciel pela amizade de tantos anos, pelas incansáveis vezes que me ouviu e me ajudou a resolver problemas quase que impossíveis, a Paulo Cardoso por tantas vezes ter me atendido e que esteve disposto a ajudar e a Yanna Ketley por sempre estar presente em nossos "serões" do laboratório.

Agradeço também aos amigos Ivo Serra pelos bons conselhos, esclarecimentos e pela ajuda de sempre e a Luís Eduardo por tantas vezes me ouvir e atender aos meus pedidos.

Agradeço a Gustavo Ferreira pela força que sempre me deu, pelo amigo, companheiro, conselheiro e por tantas vezes ouvir minhas lamentações.

Agradeço à Danielly, Denise, Lucina e Danilo pelas vezes que me acolheram e me fizeram rir dos problemas que tive.

Também quero agradecer aos colegas de mestrado que durante este tempo tanto me fizeram rir, Adriana Leite, Rayane Menezes, Manoel Messias, Carla Faria, Wilian Mendes, Giulia Soares, Nailson Boaz, Rodrigo Monteiro, Fábio Vieira, Marcus

Vinícius, Dhully Andrade, Luiz Aurélio, Raquel Machado, Renato Ubaldo, Cláudio Aroucha, Arikleyton Ferreira, Ariel Soares, Marcelo Monier, Berto de Tácio, Haroldo Gomes, Antônio Oséas, Thiago Pinheiro, Allison Jorge, Milson Lima, Moisés Santos e Jonatan.

Agradeço aos colegas que fiz na turma de graduação durante o período de estágio docente, em especial a Steve Ataki, Leonardo Melo, Jean Pablo, Luciana, Débora e Núbia.

Agradeço a Alcides por todas as vezes em que atendeu as minhas solicitações.

Aos professores que encontrei durante todo mestrado e graduação, em especial a Sérgio Martins e Reinaldo de Jesus.

Aos professores Ademir Martins e Sofiane Labidi pelas cartas de recomendação.

E, a toda coordenação do mestrado, coordenadores e funcionários, pelos bons serviços oferecidos, que foram fundamentais para a conclusão deste curso.

"Apponta pra fé e rema..."'

Marcelo Camelo

Lista de Figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

As ontologias são estruturas de representação do conhecimento capazes de expressar um conjunto de entidades e seus relacionamentos, restrições e axiomas sobre um dado domínio. Pode-se observar sua aplicação na construção dos sistemas baseados em conhecimento (SBC), no processamento da linguagem natural, nos *web services*, na comunicação e bases de conhecimento de agentes de software, na recuperação e integração da informação, nas aplicações de gerenciamento de conhecimento e na *web* semântica [1]. O uso de ontologias vem recebendo uma atenção especial nos últimos anos devido à capacidade de organizar informações, principalmente no que diz respeito à representação formal de conhecimento [2] que provê benefícios como o entendimento, a comunicação, a representação e compartilhamento de informações em várias línguas bem como o reúso para a integração de diferentes fontes de dados.

Quanto ao entendimento, a ontologia pode servir como uma documentação, que as pessoas usam para entender a conceituação de um domínio de interesse; quanto à comunicação, ela pode auxiliar as pessoas na comunicação sobre um domínio de interesse livre de ambiguidades; quanto à representação de conhecimento e reúso, representa um vocabulário de consenso e especifica conhecimento de domínio de forma explícita no seu mais alto nível de abstração com enorme potencial para o reúso. E, isso facilita o entendimento e comunicação entre homem e máquina [3].

Há diferentes definições para o termo ontologias. Segundo Gruber [4], uma ontologia é *"Uma especificação formal explícita de uma conceituação compartilhada"*. Formal, quer dizer que a ontologia deve ser compreensível por máquina. Conceituação refere-se a um modelo abstrato de algum fenômeno do mundo. Explícito significa que o tipo de conceitos utilizados e as limitações do seu uso, são explicitamente definidos. Compartilhada reflete a noção de que uma ontologia captura o conhecimento consensual, isto é, não é privada de algum indivíduo, mas aceita por um grupo.

Este trabalho tem como objetivo principal o desenvolvimento e disponibilização de um processo e uma ferramenta de software que automatiza

o desenvolvimento de ontologias de aplicação a partir de fontes de informações textuais, gerando todos os elementos componentes de uma ontologia, através do reúso de técnicas e ferramentas desenvolvidas pelo Grupo de Pesquisa em Engenharia de Software e Engenharia do Conhecimento (GESEC [1]), que realizam a aprendizagem de elementos específicos da ontologia quais sejam, conceitos, hierarquias, instâncias, relacionamentos, propriedades e axiomas descritas a seguir: GAODT [5], um processo para a aquisição de relações taxonômicas de uma ontologia [6], um método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos (AFC) [7], TARNT [8], DIPPAO [3] e TAILP [9].

A GAODT proposta por Santos [5] é uma técnica orientada por objetivos para a construção de ontologias de aplicação, que provê suporte semiautomático para a formalização destes, utilizando Lógica de Primeira Ordem - LPO [10]. O processo para a aquisição de relações taxonômicas de uma ontologia proposta por Correia [6] visa a identificação de classes e relações taxonômicas aplicando processamento da linguagem natural – PLN. O método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos – AFC proposta por Galvão [7] realiza a aprendizagem automática de ontologias através da aplicação de algoritmos de AFC para aprender relacionamentos não taxonômicos. TARNT proposta por Serra [8] é um processo semiautomático que extrai relacionamentos não taxonômicos a partir de fontes de informação textuais aplicando PLN. A DIPPAO proposta por Faria [3] é um processo automático que aplica técnicas de PLN e Extração de Informação – EI para o povoamento de ontologias de aplicação. Por fim, a TAILP proposta por Leite [9] é uma técnica semiautomática que emprega Programação em Lógica Indutiva - PLI para a extração de axiomas, a partir de uma ontologia povoada.

A integração de todas estas técnicas permitiram a aprendizagem, o enriquecimento, o povoamento e a geração de novos axiomas da ontologia de maneira dinâmica e incremental, através de uma abordagem guiada por objetivos que permite a formalização dos objetivos de um sistema baseado no conhecimento.

[1] http://maae.deinf.ufma.br/index.php

1.1 Motivação

Com o crescente aumento do volume de dados disponível na *web* surgiu a necessidade de utilizar técnicas para melhor organizar o tratamento, seleção e reúso de informações relevantes assim como a semântica dos dados. Esta constante evolução de sistemas de representação de conhecimento deu origem às ontologias, esta tecnologia desponta como uma forma viável de estruturar informações esparsas disponíveis na rede. Segundo Faria & Girardi [11], uma ontologia consiste de um conjunto de conceitos e relacionamentos taxonômicos ou não taxonômicos que representam o conhecimento de um domínio específico, a esses conceitos estão relacionados as instâncias que dão valores às propriedades. Estes conceitos são organizados hierarquicamente através de relacionamentos taxonômicos.

Uma ontologia define os conceitos básicos e os relacionamentos que constituem o vocabulário de um dado domínio de conhecimento, bem como as regras para combinar termos e a definição de relações entre eles [12], sendo formalmente representada por um conjunto de classes, hierarquias, instâncias, relacionamentos, propriedades e axiomas. Técnicas e ferramentas para a aprendizagem dos diferentes componentes da ontologia vêm sendo desenvolvidas no contexto do grupo GESEC, para formalizar os diferentes elementos que constituem uma ontologia, entretanto, essas técnicas e ferramentas geram os elementos da ontologia de forma isolada, ou seja, trabalham apenas na geração de alguns dos elementos da definição de ontologias.

Atendendo a necessidade de se obter uma abordagem que trabalha na geração de uma ontologia de forma completa, propõe-se o Apponto-Pro (*Application Ontology Process*), um processo incremental para o aprendizado e povoamento de ontologias de aplicação que integra diferentes técnicas que geram elementos de uma ontologia de forma isolada, realizando a aprendizagem, o enriquecimento, o povoamento e a geração de novos axiomas em uma ontologia de maneira dinâmica e incremental, além de permitir a formalização dos objetivos de um dado sistema baseado no conhecimento. Sua aplicação é ilustrada com o desenvolvimento de uma ontologia de aplicação no domínio do Direito da Família através da ferramenta de software intitulada Apponto-ProTool (*Application Ontology Process Tool*), que dá suporte ao processo Apponto-Pro. O resultado final do processamento é uma ontologia de

aplicação composta por todos os elementos da definição formal de ontologias (C$_C$, H, I, R, P, A), atendendo ao principal objetivo deste trabalho.

1.2 Objetivos

1.2.1 Objetivo geral

Sistematizar e promover a automatização da construção de ontologias de aplicação através da integração de técnicas e ferramentas para o aprendizado e povoamento de ontologias.

1.2.2 Objetivos específicos

- Revisão do estado da arte das técnicas e ferramentas para o aprendizado e povoamento de ontologias, especialmente as técnicas que visam a construção de ontologias de aplicação desenvolvidas no contexto do Grupo de Engenharia de Software e Engenharia do Conhecimento (GESEC).
- Especificar uma metodologia e processo para a construção de ontologias de aplicação através da integração de técnicas para o aprendizado e povoamento de ontologias analisadas na meta precedente.
- Avaliar a metodologia e processo proposto através do desenvolvimento de um estudo de caso.

1.3 Metodologia de pesquisa

A metodologia utilizada no trabalho de pesquisa é listada abaixo:

1. Pesquisa bibliográfica para coletar informações sobre o estado da arte e fundamentação teórica a seguir:

- Estudo das técnicas e ferramentas para a aprendizagem e povoamento de ontologias de aplicação, em especial as abordagens GAODT [5], um processo para a aquisição de relações taxonômicas de uma ontologia [6], o método para a

aprendizagem de hierarquia de conceitos utilizando análise formal de conceitos – AFC [7], TARNT [8], DIPPAO [3] e TAILP [9] juntamente com a metodologia, ferramenta e ambiente de desenvolvimento de cada uma.

- Estudo de outras abordagens que trabalham com a construção e desenvolvimento de ontologias (*A*GROVOC [13], C&L [14] e Poronto [15]).

- Integração das técnicas que compõem o processo através de uma metodologia e processo incremental para o aprendizado e povoamento de ontologias de aplicação.

2. Especificação e desenvolvimento do processo proposto;

3. Uso da linguagem PHP [16] para a construção e desenvolvimento de uma ferramenta de software que forneça suporte ao processo;

4. Integração das ferramentas referentes às técnicas que compõem o processo dando origem à ferramenta Apponto-ProTool;

5. Uso do Protégé para a visualização da ontologia desenvolvida;

6. Estudo do domínio do Direito da Família no qual é baseada a ontologia construída;

7. Adaptação e extensão das ferramentas que compõem o processo para que possam suportar a integração proposta;

1.4 Estrutura do trabalho

Este trabalho, incluindo a introdução, está estruturado em cinco capítulos.

O segundo capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho, abordando um breve histórico das ontologias, uma descrição dos elementos da definição formal, a classificação das ontologias, a linguagem OWL [17] e o software Protégé [18], citando também a importância do uso de ontologias para descrever o conhecimento. Apresenta também as técnicas e ferramentas utilizadas para o desenvolvimento do trabalho proposto. São apresentadas ainda as abordagens do estado da arte que também trabalham na construção e desenvolvimento de ontologias.

O terceiro capítulo apresenta o Apponto-Pro, um processo incremental para o aprendizado e povoamento de ontologias de aplicação, principal proposta deste trabalho.

O quarto capítulo apresenta uma avaliação do processo com a aplicação da ferramenta de software Apponto-ProTool desenvolvida para dar suporte ao processo. Esta avaliação consistiu na construção de uma ontologia de aplicação completa no domínio do Direito da Família. Este capítulo apresenta também duas discussões realizadas, uma sobre a integração da ferramenta Apponto-ProTool e outra sobre os resultados obtidos.

No quinto capítulo são apresentadas as considerações finais do trabalho, destacando os resultados obtidos bem como a avaliação do processo, a publicação realizada e os trabalhos futuros que poderão ser desenvolvidos a partir desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho. Inicialmente é apresentada a definição formal de ontologias adotada pelo grupo GESEC, descrevendo cada um dos seus elementos. Em seguida são descritas, de maneira sucinta, as modalidades de construção de ontologias conforme seu nível de abstração, como proposto por Guarino [2] e Girardi [19].

São também apresentados a linguagem OWL para a especificação de ontologias e o software Protégé para a construção e visualização de ontologias, ambos utilizados no desenvolvimento do Apponto-Pro. Por fim, são apresentadas as técnicas e correspondentes ferramentas que foram integradas no desenvolvimento do processo proposto.

O capítulo está organizado da seguinte maneira. A seção 2.1 apresenta a definição formal de ontologias. A seção 2.2 descreve a classificação das ontologias. A seção 2.3 seguida pelas subseções 2.3.1 a 2.3.6 discorrem respectivamente, sobre as técnicas GAODT [5], um processo para a aquisição de relações taxonômicas de uma ontologia [6], aprendizagem de hierarquias de conceitos utilizando AFC [7], TARNT [8], DIPPAO [3] e TAILP [9], todas desenvolvidas no contexto do grupo GESEC.

Na seção 2.4 é apresentado o estado da arte referente a outras abordagens para a construção de ontologias. A seção 2.5 apresenta uma análise comparativa das abordagens levando em consideração quatro critérios: o produto de entrada no processo, as tecnologias utilizadas, o elemento gerado ou aprendido e o nível de automação. Finalmente, na seção 2.6 são apresentadas as considerações finais do capítulo.

2.1 Definição formal de ontologias

Formalmente uma ontologia pode ser definida como a 6-tupla [19],

O = <C, H, I, R, P, A>, onde:

$C = C_C \cup C_I$ é o conjunto de entidades do domínio sendo modelado. O conjunto C_C é formado por classes, ou seja, conceitos que representam entidades que descrevem um conjunto de objetos (por exemplo, "Mãe" $\in C_C$) enquanto que o conjunto C_I é formado por instâncias, ou seja, entidades únicas no domínio (por exemplo, "Anne Smith" $\in C_I$).

$H = \{tipo\_de(c_1,c_2) \mid c_1 \in C_C \wedge c_2 \in C_C\}$ é o conjunto de relações taxonômicas que definem a hierarquia de classes da ontologia e são denotadas por "{tipo_de(c_1, c_2)}" indicando que c_1 é uma subclasse de c_2. Um exemplo desse relacionamento é "{tipo_de(Mãe, Pessoa)}".

$I = \{é\_um(c_1,c_2) \mid c_1 \in C_I \wedge c_2 \in C_C\} \cup prop_I(c_k,valor) \mid c_k \in C_I \cup rel_k(c_1,c_2,..., c_n) \mid \forall i, c_i \in C_I$, é o conjunto de relacionamentos entre os elementos da ontologia e suas instâncias, por exemplo "é_um("Anne Smith", Mãe)", "data_de_nascimento ("Anne Smith", "12/02/1980")" e "mãe_de("Anne Smith", "Clara Smith")" são relacionamentos entre classes, relacionamentos, propriedades e suas instâncias.

$R = \{rel_k(c_1,c_2,..., c_n) \mid \forall i, c_i \in C_C\}$ é o conjunto de relacionamentos não taxonômicos de uma ontologia. Por exemplo, "mãe_de(Mãe, Filha)".

$P = \{prop_C(c_k, tipo) \mid c_k \in C_C\}$ é o conjunto de propriedades das classes de uma ontologia e seu tipo de dados básico. Por exemplo, "data_de_nascimento(Mãe, dd/mm/aaaa)".

$A = \{condition_X \implies conclusion_Y(c_1, c_2, ..., c_n) \forall_j, C_j \in C_C\}$ é um conjunto de axiomas, regras que permitem checar a consistência da ontologia e deduzir novos conhecimentos através de algum mecanismo de inferência. O termo $\{condition_X$ é dado por: $\{condition_X = \{(cond_1, cond_2, ..., cond_n) | \forall_z, cond_z \in H \cup | \cup R\}$. Por exemplo, "Mãe, Filha1, Filha2, mãe_de(Mãe, Filha1), mãe_de(Mãe, Filha2) $\implies$ irmã_de(Filha1, Filha2)" é uma regra que indica que, se duas filhas têm a mesma mãe, então as filhas são irmãs.

2.2 Classificação de ontologias

Segundo Guarino [2], as ontologias devem obedecer ao seguinte esquema de classificação: ontologias gerais, de domínio, de tarefa e de aplicação (classificadas de acordo com sua generalidade). Uma ontologia genérica descreve conceitos gerais, tais como, espaço, tempo, matéria, objeto, evento, ação, sendo independentes de domínio.

Uma ontologia de domínio reúne conceitos de um domínio particular e seus relacionamentos, definindo restrições na estrutura e conteúdo do conhecimento desse domínio, por exemplo, o domínio jurídico. Uma ontologia de tarefa expressa conceitos sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descreve o vocabulário relacionado a uma atividade ou tarefa genérica, por exemplo, as técnicas para o acesso à informação. Por fim, uma ontologia de aplicação é mais específica, pois descreve conceitos dependentes ao mesmo tempo de um domínio particular como, por exemplo, uma ontologia para identificar doenças do coração, a partir de uma ontologia de domínio de cardiologia.

Construir ontologias na ordem proposta por Guarino [2], ou seja, das de mais alto nível para as de mais baixo nível de abstração, partindo das ontologias genéricas para as de aplicação, não seria totalmente apropriado, uma vez que a forma natural de raciocínio consiste em considerar o caso concreto e depois generalizá-lo.

Uma alternativa mais adequada para a construção de ontologias é sugerida por Girardi [19], baseada em uma estruturação mais natural de raciocínio, partindo do específico para o geral. Esta construção segue o estilo *"botton-up"*, ou seja, primeiramente deve-se construir ontologias de aplicação, que são as mais específicas, para a partir delas realizar a generalização dos termos, na forma das ontologias de domínio e tarefa e por fim nas de alto nível. A Figura 2.1 apresenta a hierarquia de ontologias conforme o seu nível de abstração e a ordem de construção de acordo com as duas abordagens citadas.

O processo Apponto-Pro e sua correspondente ferramenta estão de acordo com a abordagem de construção de ontologias proposta por Girardi [19], e no contexto deste trabalho foram utilizados para a geração de uma ontologia de aplicação que poderá ser posteriormente generalizada para as de mais alto nível.

Figura 2.1: Hierarquia de ontologias e abordagens de construção *"top-down"* e *"botton-down"*

.

As ontologias também podem ser classificadas quanto a seu grau de formalidade, sendo: informal ou formal [20]. Uma ontologia informal é expressa em linguagem natural, enquanto uma ontologia formal é expressa em uma linguagem formal para definir e instanciar ontologias, como por exemplo, a OWL.

A linguagem OWL (*Web Ontology Language*) pode incluir descrições de classes e suas respectivas propriedades e relacionamentos, ela foi projetada para o uso por aplicações que precisam processar o conteúdo da informação ao invés de apenas apresentá-la aos usuários. A intenção da OWL é representar conceitos e seus relacionamentos na forma de uma ontologia [1]. A linguagem facilita a interpretação por máquinas do conteúdo da *web* em relação a XML, RDF e RDF *Schema*, por fornecer vocabulário adicional com o uso de uma semântica formal.

A linguagem pode ser classificada em três tipos de acordo com suas sublinguagens: OWL-Lite, OWL-Full e OWL-DL, onde o que as diferencia é o seu grau de expressividade, conforme ilustrado na Figura 2.2.

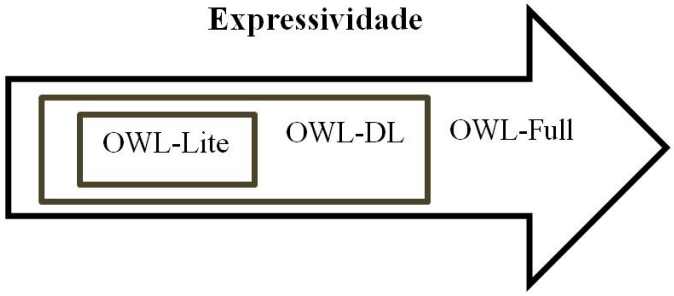


Figura 2.2: Níveis de expressividade das linguagens de especificação de ontologias

.

- OWL – *Lite* - menos expressiva e sintaticamente a mais simples, é usada quando se precisa de restrições e uma modelagem de hierarquia de classes simples, suporta aqueles usuários que necessitam principalmente de uma hierarquia de classificação e restrições simples [21].

- OWL – DL - sua expressividade está entre a OWL-Lite e a OWL-Full, é baseada em lógica de descrição *(Description Logic)* [22], possibilita computar hierarquia de classes de maneira automática, verificando sua inconsistência na linguagem ontológica.

- OWL – Full - a mais expressiva das três, é utilizada nos casos em que há necessidade de uma expressividade relativamente alta, nesta não é permitida a realização de inferências, pois permite a uma ontologia aumentar o significado do vocabulário pré-definido (RDF ou OWL).

A OWL é baseada nas linguagens OIL (*Ontology Inference Layer*, camada de inferência para ontologia) e DAML (*DARPA Agent Markup Language*, linguagem de anotação para agentes do Departamento de Defesa dos Estados Unidos) + OIL, e é atualmente uma recomendação da W3C. A linguagem vem ocupando um papel importante em um número cada vez maior de aplicações, sendo foco de pesquisa para ferramentas, técnicas de inferência e extensões de linguagens [21]. Além disso, a linguagem OWL permite o processamento e visualização de todos os elementos gerados pela maioria das ferramentas de edição de ontologias como o Protégé, uma vez que, como já citado, é o padrão adotado pela W3C. Para o desenvolvimento deste trabalho foi adotado o padrão OWL-DL, devido esta trabalhar com a lógica de descrição e ser utilizada em todas as abordagens que participam do desenvolvimento do processo poposto, além de ser o padrão mais utilizado para o desenvolvimento de ontologias [22]. A Figura 2.3, apresenta o trecho de um arquivo .owl de uma ontologia no domínio do Direito da Família. São particularmente representadas a classe *"Person"* e a hierarquia, através das subclasses *"Man"* e *"Woman"*.

Neste cenário desponta uma poderosa ferramenta para a construção e edição de ontologias, o Protégé [18]. Desenvolvida em Java pelo departamento de informática médica da Universidade de Stanford, o Protégé é uma ferramenta *open source* que dispõe de um conjunto ferramentas e uma API para a modelagem, criação, visualização e exportação de ontologias em vários formatos de representação

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:proteje="http://proteje.stanford.edu/plugins/owl/proteje"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns="http://www.owl-ontologies.com/family#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.owl-ontologies.com/family">
 <owl:Ontology rdf:about-""/>
 <owl:Class rdf:ID="Man":
   <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:iD="Woman">
    <rdfs:subClassOf rdf:resource="#Person"/>
 </owl:Class>
 <owl:ObjectProperty rdf:ID="uncle_of">
   <rdfs:range rdf:resource="#Person"/>
   <rdfs:domain rdf:resource="#Man"/>
 </owl:ObjectProperty>
 <Man rdf:ID="Man_1">
    <uncle_of>
       <Woman rdf:ID="Woman_3"/>
    </uncle_of>
 </Man>
</rdf:RDF>
```

Figura 2.3: Trecho de uma ontologia na linguagem OWL-DL

.

recomendados pela W3C tais como a OWL citada anteriormente, além de permitir a criação de bases de conhecimento. Em seu projeto original era apenas uma ferramenta de aquisição de conhecimento limitada a um sistema especialista para oncologia, o Oncocin, mas foi modernizando gradativamente para acompanhar a evolução da tecnologia de SBC [18]. Mais tarde, seus criadores optaram por disponibilizar seu código, que é desenvolvido em Java, surgindo uma arquitetura integrável a diversas aplicações.

A ferramenta tem independência de um algoritmo específico de inferência e fornece uma API de representação do conhecimento que permite a extensão do programa para necessidades específicas. Sua arquitetura é integrável a diversas aplicações via componentes que podem ser adicionados ou conectados ao sistema, sem a necessidade de redesenvolvimento. Ele possui uma interface de fácil acesso que

permite aos usuários a criação e visualização gráfica de ontologias a partir de qualquer domínio, assim como o acesso a ontologias criadas em arquivos OWL, permitindo também a alteração ou inserção de dados em arquivos OWL existentes. A Figura 2.4 ilustra uma visão da interface dessa ferramenta.

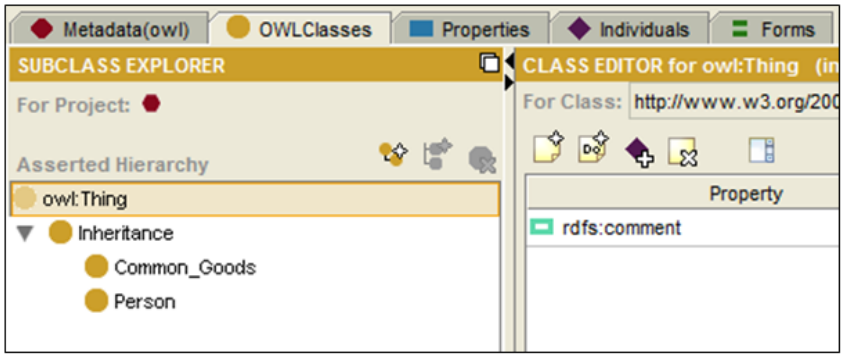


Figura 2.4: Interface do editor de ontologias Protégé

.

2.3 Técnicas e ferramentas para a aprendizagem e povoamento de ontologias

Diversas técnicas para a aprendizagem e povoamento de ontologias vêm sendo desenvolvidas, entre elas a GAODT [5] que trabalha na construção de uma ontologia de aplicação orientada por objetivos e que gera cinco elementos da ontologia (classes, hierarquias, propriedades, relacionamentos e axiomas). O processo para a aquisição de relações taxonômicas de uma ontologia [6] e o método para a aprendizagem de hierarquias de conceitos utilizando AFC [7] que trabalham na aprendizagem de relações taxonômicas de ontologias. TARNT [8] que consiste em uma técnica para a geração de relações não taxonômicas da ontologia. DIPPAO [3], um processo automático para o povoamento de ontologias de aplicação e TAILP [9], uma técnica que realiza a aprendizagem de seus axiomas utilizando PLI.

Nas seções seguintes, são descritas as técnicas e ferramentas que fazem parte da integração do trabalho proposto.

2.3.1 Técnica GAODT

GAODT *(Goal-Oriented Application Ontology Development Technique)* é uma técnica orientada por objetivos para a construção de ontologias de aplicação que traduz objetivos e fatos em linguagem natural para regras e fatos em Lógica de Primeira Ordem – LPO. Estas regras são mapeadas para uma ontologia de aplicação.

O processo inicia a partir da entrada de um conjunto de objetivos e fatos sobre um determinado domínio de aplicação, que são selecionados pelo engenheiro do conhecimento e pelo especialista do domínio e submetidos à quatro atividades: "Seleção dos Objetivos e Fatos", "Representação dos Predicados em Lógica de Primeira Ordem (LPO)", "Especificação dos Axiomas em LPO" e "Especificação/Extensão da Ontologia de Aplicação".

O desenvolvedor da ontologia de aplicação e o especialista do domínio participam da execução destas atividades. O desenvolvedor é o engenheiro do conhecimento responsável pela construção da ontologia de aplicação. O especialista do domínio é alguém que detém informações específicas daquela área de conhecimento. A Figura 2.5 mostra a visão geral da técnica GAODT [12].

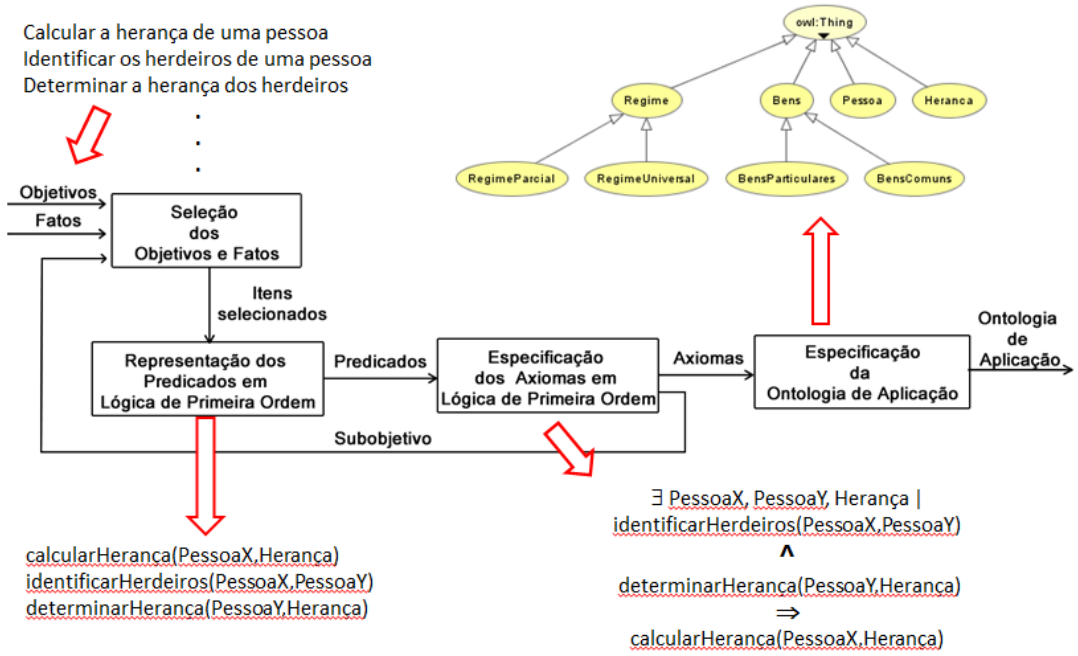


Figura 2.5: Visão geral da técnica GAODT

.

Na atividade "Seleção de Objetivos e Fatos" o especialista do domínio juntamente com o engenheiro do conhecimento elaboram a partir de um determinado

domínio de aplicação um conjunto de objetivos e fatos escritos em linguagem natural. Tal conjunto é dado para a atividade "Representação dos Predicados em Lógica de Primeira Ordem", que consiste em transformar para linguagem de máquina os objetivos e fatos inseridos em linguagem natural. Nesta fase é realizada uma interpretação dos objetivos para predicados em LPO .

Na atividade "Especificação dos Axiomas em Lógica de Primeira Ordem" os predicados são especificados em axiomas de LPO para que possa ser definida a condição/conclusão a ser atendida pelo sistema. A atividade "Especificação da Ontologia de Aplicação" permite que os objetivos inseridos e processados nas fases anteriores sejam especificados em uma ontologia de aplicação.

Para a execução das fases descritas, a técnica necessita de uma ferramenta para auxiliar no desenvolvimento das ontologias de aplicação, para isso foi desenvolvida a GAODTool *(Goal-Oriented Application Ontology Development Tool)*, uma ferramenta de software que provê suporte semiautomatizado ao processo. O resultado final é uma ontologia de aplicação composta por classes, hierarquias, propriedades, relacionamentos e axiomas, sendo uma ontologia com instâncias ausentes (elemento I) em seu produto final.

2.3.2 Um processo para a aquisição de relações taxonômicas de uma ontologia

O processo para a aquisição automática de relações taxonômicas de uma ontologia foi desenvolvido com o objetivo de demonstrar que é possível unir a utilização de padrões léxico-sintáticos e processamento da linguagem natural em um processo automático para o aprendizado de relações taxonômicas de uma ontologia, para isso aplica técnicas de PLN e padrões heurísticos. Este processo é composto por quatro fases: "Marcação", "Extração de Classes Candidatas", "Identificação de Hiperônimos e Sinônimos" e "Identificação e Representação de Relações Taxonômicas", como ilustra a Figura 2.6.

Inicialmente, na fase de "Marcação", o processo recebe um conjunto de documentos sobre determinado domínio e aplica técnicas de PLN para que o corpus seja representado e interpretado adequadamente. Esta fase tem por objetivo realizar a

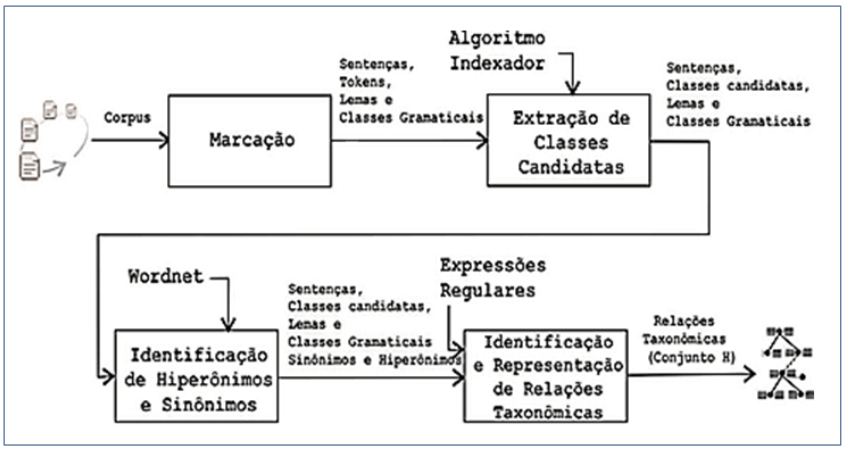


Figura 2.6: Visão geral do processo para aquisição de relações taxonômicas

.

identificação de *tokens*, sentenças, classes gramaticais e lemas, obtendo como produto um corpus marcado.

Na fase "Extração de Classes Candidatas", é dado como entrada o corpus marcado na fase anterior para que sejam separados os *tokens*, pois estes serão as possíveis classes na hierarquia de uma ontologia. ou seja, é realizada uma seleção dos substantivos concretos e abstratos que serão as classes candidatas, referente ao conjunto Cc de uma ontologia.

A fase "Identificação de Sinônimos e Hiperônimos" utiliza a base de dados do wordnet com o objetivo de identificar os sinônimos e hiperônimos dos termos que representam as classes obtidas na fase anterior.

Finalmente, na fase "Identificação e Representação de Relações Taxonômicas" são descobertas as relações a ser representadas na ontologia. Para isso, são aplicas expressões regulares para que sejam descobertas relações de hiperonímia. Estas relações são tratadas e organizadas juntamente com os sinônimos e hiperônimos identificados na fase anterior resultando em um conjunto de relações taxonômicas.

2.3.3 Um método para a aprendizagem de hierarquia de conceitos utilizando AFC

Este método permite a aprendizagem de ontologias utilizando a análise formal de conceitos. A análise formal de conceitos é um subcampo da matemática

aplicada utilizada especialmente para análise de dados e descoberta de relações. Consiste basicamente na criação automática de um contexto formal, a partir de um corpus de entrada onde algoritmos de AFC são aplicados a fim de que sejam extraídas a camada das classes e suas respectivas hierarquias da ontologia. O processo é composto por três fases: "Pré-processamento", "Extração das camadas da ontologia" e "Pós-processamento". Conforme ilustrado na Figura 2.7.

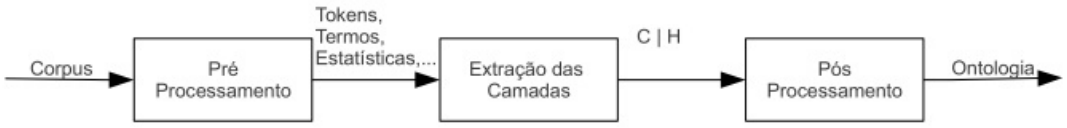


Figura 2.7: Visão geral do processo para aprendizagem de relações taxonômicas

.

O método recebe como entrada na fase de "Pré-processamento", um corpus sobre determinado domínio de aplicação. Nesta fase são aplicadas técnicas de PLN a fim de que seja realizada a tokenização, a divisão de sentenças, o *pos-tagging*, a lematização e a análise sintática. Também é realizada nesta fase a extração de conceitos candidatos e criação do contexto formal, através das abordagens de "Verbo-nome", "Verbo-nome-filtro", "Probabilidade", "Nome-nome" e "Wordnet-hipônimos", que são descritas na Tabela 2.1, gerando como produto um arquivo que representa o contexto formal.

Na fase "Extração de Camadas da Ontologia" são aplicados algoritmos de AFC sobre o contexto formal gerado na etapa anterior. O resultado é um arquivo em XML que representa uma hierarquia de conceitos formais que então é mapeada e transformada em uma hierarquia de conceitos da ontologia. Este mapeamento é realizado tanto de maneira *"top-down"* (para cada novo conceito que é encontrado os elementos da intensão deste são mapeados como conceitos da ontologia), quanto de maneira *"botton-up"* (mapeia cada elemento em um conceito e cria uma relação taxonômica entre este conceito).

Por fim na fase de "Pós-processamento", as camadas Cc e H aprendidas são representadas em um arquivo OWL correspondente à ontologia aprendida.

Tabela 2.1: Abordagem para a extração de conceitos na aprendizagem de hierarquias

Abordagem	**Objetos**	**Atributos**	**Diferencial**
Verbo-Nome	Verbos	Conceitos Candidatos	Nenhum
Verbo-Nome (filtro)	Verbos	Conceitos Candidatos	Apenas alguns tipos de relações sintáticas foram consideradas
Probabilidade	Verbos	Conceitos Candidatos	Utiliza apenas pares de (verbos, nomes) que possuem uma probabilidade de aparecerem juntos, maior que um valor de corte
Nome-Nome	Conceitos Candidatos (subconjunto)	Conceitos Candidatos (subconjunto)	O contexto formal é formado apenas por conceitos candidatos
Wordnet-Hipônimos	Hipônimos	Conceitos Candidatos	Utiliza as relações de hiponímias para construir o contexto

2.3.4 Técnica TARNT

TARNT é uma técnica para a aprendizagem de relacionamentos não taxonômicos de ontologias a partir de fontes textuais em língua inglesa. A técnica utiliza técnicas de PLN e estatísticas para extrair e sugerir ao engenheiro do conhecimento relacionamentos não taxonômicos sobre conceitos de ontologias conhecidos a priori. TARNT [8] possui seis fases: "Construção do Corpus", "Anotação do Corpus", "Extração de Relacionamentos", "Refinamento", "Avaliação do Especialista" e por fim "Atualização da Ontologia", conforme ilustrado na Figura 2.8.

Na primeira fase "Construção do Corpus" a técnica tem como entrada um corpus pronto ou, pode utilizar um elaborado pelo engenheiro do conhecimento e o especialista do domínio. A partir da obtenção deste conjunto de documentos é dado início realizada a extração dos relacionamentos candidatos.

A fase "Anotação do Corpus" tem o objetivo de marcar o corpus dado como entrada para que sejam realizadas anotações necessárias como tokenização, separação de sentenças e opcionalmente lematização, análise morfológica e *vp chunk* através do processamento da linguagem natural.

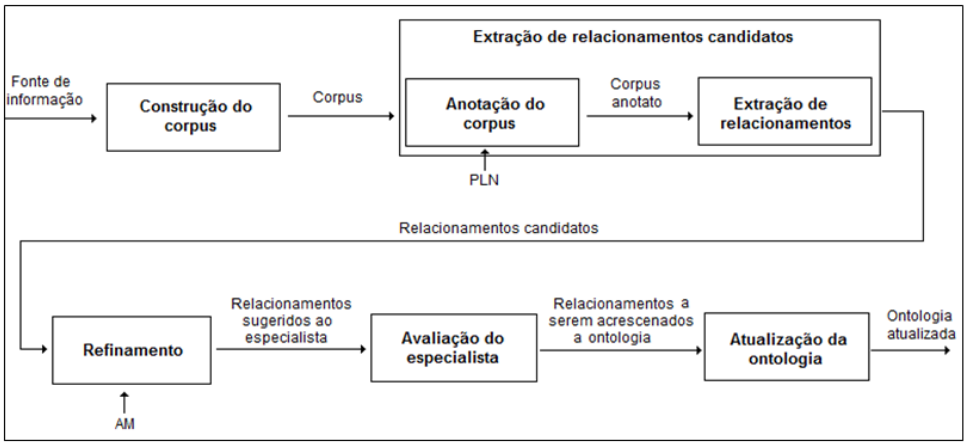


Figura 2.8: Visão geral da técnica TARNT

.

A fase "Extração dos Relacionamentos" recebe como entrada o corpus anotado na fase anterior e aplica três regras para a extração de relacionamentos: regra de sentença, regra de sentença com frase verbal e regra de apóstrofo.

Após ser realizada a extração de relacionamentos do corpus, este retorna a um conjunto de relacionamentos candidatos que é dado como entrada para a fase de "Refinamento". Nesta fase é realizada a frequência de co-ocorrências e *bag of labels*, a fim de que seja calculada a frequência dos relacionamentos candidatos e a verificação da coincidência de conceitos respectivamente.

Os relacionamentos encontrados na fase anterior são sugeridos ao especialista para que possam ser selecionados e possivelmente editados na fase de "Avaliação do Especialista". O resultado da técnica deve ser avaliado por um especialista antes que as relações sejam definitivamente adicionadas à ontologia.

Por fim, a fase "Atualização da Ontologia" tem por objetivo executar um procedimento de atualização do arquivo da ontologia com os relacionamentos não taxonômicos selecionados e possivelmente editados na fase anterior.

2.3.5 Técnica DIPPAO

A técnica DIPPAO *(Process for Domain Independent Populating Automatic Ontology)* consiste em uma abordagem automática para o povoamento de ontologias de aplicação a partir de fontes textuais (corpus [1]). Seu desenvolvimento pode ser

[1]Conjunto de documentos sobre um determinado domínio de aplicação

realizado a partir de qualquer domínio de aplicação, esta característica permite uma independência de domínio a DIPPAO.

O objetivo da técnica é realizar a identificação, extração e classificação de instâncias de relacionamentos não taxonômicos e propriedades de uma ontologia a partir de fontes textuais, utilizando técnicas de processamento da linguagem natural, reduzindo assim os custos no processo de povoamento. DIPPAO [3] é composta por três fases: "Identificação de Instâncias Candidatas", "Construção de um Classificador" e "Classificação de Instâncias", como ilustrado na Figura 2.9.

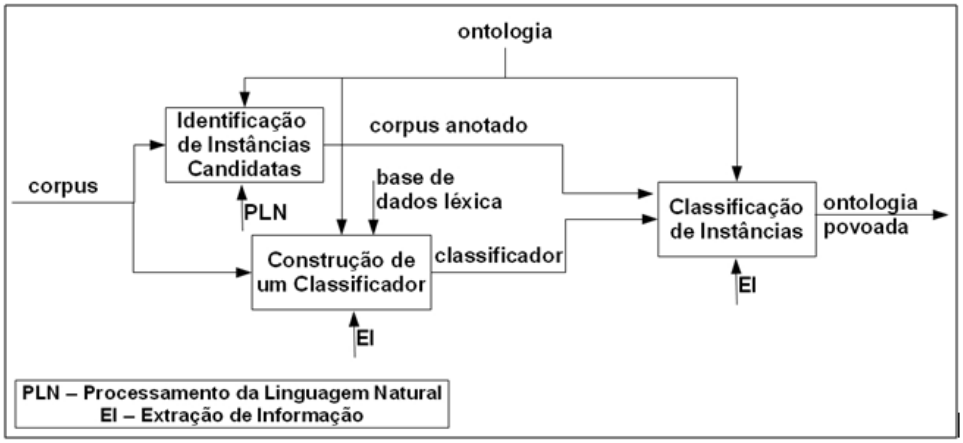


Figura 2.9: Visão geral da técnica DIPPAO

.

Seu processo inicia a partir da entrada de uma ontologia e um corpus à fase "Identificação de Instâncias Candidatas", esta aplica técnicas de PLN para identificar instâncias de relacionamentos não taxonômicos e propriedades de uma ontologia através da anotação do corpus.

A fase "Construção de um Classificador" aplica técnicas de EI para construção de um classificador, baseado em um conjunto de regras de classificação a partir de uma ontologia e, de consultas a uma base de dados léxica (Wordnet [2]), obtendo como produto um classificador.

A fase "Classificação de Instâncias" tem como entrada o classificador gerado na fase anterior, a ontologia e o corpus anotado, para que seja realizado a associação de instâncias de relacionamentos não taxonômicos e propriedades às suas respectivas classes da ontologia, gerando como produto final uma ontologia povoada.

[2]Base de dados lexical organizada por significado

Para automatizar o povoamento de ontologias, foi desenvolvida uma ferramenta denominada DIPPAOTool *(Tool for Automatic Ontology Population)* que processa o corpus em língua inglesa e povoa a ontologia automaticamente. A ontologia povoada pode ser utilizada na execução de sistemas baseados em conhecimento e na extensão de ontologias, sendo um dos motivos que levaram à integração mo processo proposto neste trabalho.

2.3.6 Técnica TAILP

TAILP é uma técnica semiautomática para a aquisição de axiomas que visa a área de aprendizagem de ontologias através da aplicação de PLI (Programação em Lógica Indutiva). Esta técnica pode ser descrita como, dada uma ontologia povoada de domínio estabelecido, tem-se como objetivo a indução de regras ou axiomas, com base nas instâncias de relações não taxonômicas da ontologia usando a PLI [23].

A técnica foi desenvolvida pelo grupo GESEC para facilitar a busca por axiomas a partir de uma ontologia previamente povoada, através de inferências indutivas sobre regras induzidas, a fim de que se tornem axiomas no contexto da aprendizagem de ontologias, permitindo um aprendizado indutivo.

O processo é composto por três fases: "Extração de Elementos da Ontologia", "Geração da Hipótese Indutiva" e "Inserção de Nova Regra na Ontologia". A Figura 2.10 ilustra a visão geral da técnica.

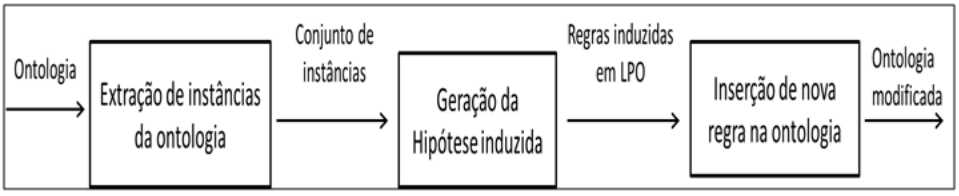


Figura 2.10: Visão geral da técnica TAILP

.

A fase "Extração de Elementos da Ontologia" tem como entrada uma ontologia povoada, para que seja extraído um conjunto de instâncias de relações e um conjunto de instâncias de classes.

A fase "Geração da Hipótese Induzida" utiliza o conjunto de instâncias extraídas na fase anterior para construir uma base de conhecimento (regras induzidas),

através de um processo de inferência indutiva. Esta base é construída com o auxílio de uma ferramenta PLI (Progol), retornando a hipóteses induzidas em LPO.

A fase "Inserção de Nova Regra na Ontologia" consiste em realizar uma avaliação das hipóteses encontradas anteriormente, para que se tornem os axiomas a ser inseridos na ontologia. O resultado deste processo é uma ontologia composta por axiomas, elemento A da definição formal.

Para o desenvolvimento destas fases foi desenvolvida uma ferramenta intitulada TAILP que semiautomatiza a construção de ontologias de aplicação. A ferramenta utiliza lógica dedutiva trabalhando a partir de um domínio específico para um geral.

Na Tabela 2.2 é apresentado um resumo comparativo das abordagens desenvolvidas no contexto do grupo GESEC, apresentando as entradas, as tecnologias utilizadas, o elemento gerado ou aprendido em cada uma bem como o seu nível de automação.

Tabela 2.2: Comparativo entre as abordagens desenvolvidas no GESEC

	Entrada	**Tecnologias Utilizadas**	**Elemento Gerado ou Aprendido**	**Nível de Automação**
GAODT	Objetivos e Fatos	LPO	C, H, R, P, A	Semiautomática
Uma técnica para a aquisição automática de relações taxonômicas	Corpus	PLN	C, H	Automática
Um método para aprendizagem de hierarquias de conceitos utilizando AFC	Corpus	AFC	C, H	Automática
TARNT	Corpus e Ontologia	PLN	R	Automática
DIPPAO	Corpus e Ontologia	PLN EI	I	Automática
TAILP	Ontologia Povoada	PLN LPO	A	Semiautomática

2.4 Estado da arte dos processos de desenvolvimento de ontologias

Esta seção apresenta outras abordagens já propostas que assim como a Apponto-Pro, fornecem suporte ao desenvolvimento de ontologias. O *framework*

AGROVOC [13], a abordagem C&L [14] e a abordagem Poronto [15], são apresentadas nas seções 2.4.1, 2.4.2 e 2.4.3 respectivamente. Na seção 2.5 é realizada uma análise comparativa entre as técnicas analisadas e na seção 2.6 são apresentadas as considerações finais do capítulo.

2.4.1 O *Framework* AGROVOC

O *framework* AGROVOC é um processo para a construção de ontologias que utiliza uma abordagem automática para aprender tanto relações taxonômicas quanto relações semânticas entre conceitos. O *framework* tem como entrada um corpus e um conjunto de conceitos no domínio da agricultura e como saída uma ontologia composta de classes, hierarquias e relações taxonômicas.

O AGROVOC é composto de quatro fases: *"Candidate Sentence Extration - Extração de Sentenças Candidatas"*, *"Clause Identification Resolving Conjunction - Identificação e Resolução de Cláusulas de Conjunção"*, *"Candidate Triple Identification - Extração da Tripla Candidata"* e *"Relation Label Suggestion - Sugestão do Rótulo da Relação"*. Estas fases são desenvolvidas utilizando a base de dados lexical Wordnet [24], que permite realizar uma análise e processamento computacional do corpus dado de entrada em linguagem natural a fim de que o resultado final do processamento obtenha resultados satisfatórios. A Figura 2.11 apresenta a visão geral do *framework* proposto.

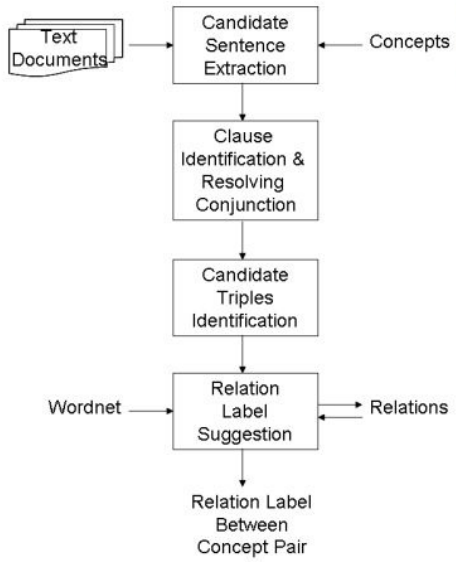


Figura 2.11: Visão geral do *framework* AGROVOC

.

O *framework* considera o conjunto de conceitos C=c_1,c_2...c_r e um conjunto de frases T=t_1,t_2...t_z identificadas a partir do corpus dado como entrada. Em posse destes dois conjuntos, é realizado o cálculo da função expressa por f: C X C $\rightarrow$ R, com o objetivo do *framework* aprender um conjunto de relações R=r_1,r_2...r_y, dando início à fase "*Candidate Sentence Extraction* - Extração de Sentenças Candidatas", ou seja, dado como entrada o corpus e o conjunto de conceitos são extraídas sentenças candidatos através de um conjunto de frases T e um conjunto de conceitos C, para isso necessita:

- Considerar (C_m,C_n) como um par de conceitos em C X C;
- Encontrar as sentenças correspondentes a T_{mn} em T para o par (C_m, C_n);
- Definir o conjunto de verbos V_{mn};
- Encontrar um conjunto de frases candidatas T_{mnc} que contribuem para o par de relacionamentos de C_m, C_n;

O processo busca relações entre dois conceitos C_m, $C_n \in$ C, para isso é necessário extrair um conjunto de sentenças $T_{mn} \in$ T, que contém tanto C_{mr}, quanto C_{nr} em suas formas plural e singular. Entretanto, um corpus pode ter um grande número de sentenças e estas podem não fazer relação entre C_m e C_n, sendo necessário a obtenção de um conjunto de sentenças candidatas T_{mnc} *C* T_{mn}, que auxilia na descoberta de relações entre pares de conceitos.

Para ajudar na descoberta de relações entre pares de conceitos, são utilizadas técnicas linguísticas que identificam um conjunto de verbos V_{mn} e que extraem todos os verbos identificados com a tag VB* (verbo base) nas sentenças. Ou seja, são consideradas todas as formas do verbo (VB), tal como VBD (verbo no pretérito), VBG (verbo no gerúndio) e VBZ (verbo na 3ª do singular). A posição do verbo deve ser entre C_{mr} e C_{nr} ou, entre C_{nr}, C_{mr}. Esta técnica permite trazer todos os verbos obtidos nestas formas a fim de que seja encontrada a frequência de cada um e contruído um conjunto de verbos V_{nf}, que considera os verbos que possuem maiores frequências.

A segunda fase *"Clause Identification Resolving Conjunction - Identificação e Resolução de Cláusulas de Conjunção"* realiza a extração das sentenças candidatas. Esta extração consiste em analisar semanticamente uma sentença a fim de que sejam

identificadas todas as cláusulas disponíveis. A partir destas cláusulas, são obtidas as informações de sujeito e predicado onde as relações podem ser determinadas a partir dos predicados identificados. Para isso, cada sentença candidata identificada em $t_{mn} \in$ T_{mnc} é analisada semanticamente para determinar a estrutura da sentença.

Esta análise semântica é realizada a partir da identificação de informações como NP (frase nominal), VP (frase verbal) e PP (frase preposicional), assim é possível identificar conjunções como *"that"*, *"which"*, etc. Por exemplo, considerando a sentença *"In the United States, all pesticides, including antimicrobial, are regulated under FIFRA"*, a análise semântica pode ser visualizada na Figura 2.12 que apresenta a árvore de análise do *framework* AGROVOC.

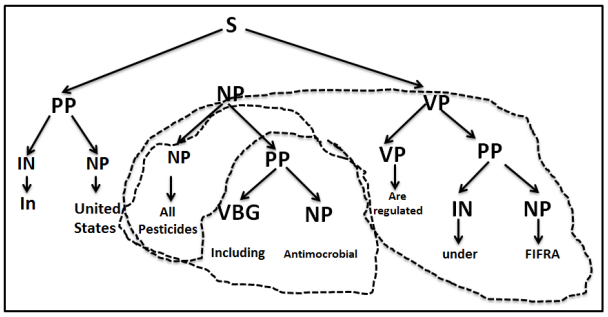


Figura 2.12: Árvore da análise semântica realizada no *framework* AGROVOC

.

Na fase *"Candidate Triples Identification* - Identificação da Tripla Candidata" as cláusulas sem objeto identificadas são ignoradas na descoberta de triplas. Já para as cláusulas em que são identificados o sujeito e o predicado a tripla (sujeito-verbo-objeto) é determinada. A Figura 2.13 mostra o exemplo das regras para a geração de cláusulas.

Rules	Clauses		
	Subject	Predicate	Object
if ((NP_1 | PP_1 NP_1 | $ADJP_1$ NP_1 | SBAR) (VP_1) (NP_2 | PP_2 NP_2 | $ADJP_2$ NP_2))	NP_1 | SBAR	VP_1 Object	NP_2 | PP_2 NP_2
if ((NP_1 | PP_1 NP_1 | $ADJP_1$ NP_1 | SBAR) (PP_2 having verb in gerund form) (NP_2 | PP_3 NP_2 | $ADJP_2$ NP_2))	NP_1 | SBAR	PP_2 Object	NP_2
if (('which') (VP_1) (NP_1 | PP_1 NP_1 | $ADJP_1$ NP_1))	previous Object	VP_1 Object	NP_1 | PP_1 NP_1

Figura 2.13: Exemplo de regras para a geração de cláusulas da AGROVOC

.

Na fase *"Relation Label Suggestion* - Sugestão do rótulo da relação" o *framework* proposto aprende um conjunto de relações (R) e mapeia, simultaneamente, qualquer r $\in$ R para os pares de conceito (C_m, C_n) e (C_n, C_m).

O resultado final é uma ontologia com classes, hierarquias e relacionamentos não taxonômicos aprendidos.

2.4.2 C&L: uma ferramenta semiautomática para a construção de ontologias utilizando engenharia de requisitos

Nesta abordagem é exposto como a engenharia de requisitos pode auxiliar no processo de criação de ontologias. Breitman [1] define que uma ontologia é um produto da engenharia de requisitos devido esta possuir um núcleo de conhecimento sobre os processos para captura, modelagem e análise de informações relevantes, contribuindo na tarefa de construção de ontologias. Com base neta ideia é apresentada a C&L, uma ferramenta de apoio à engenharia de requisitos que tem como objetivo a edição de Cenários LAL (Léxico Ampliado da Linguagem), desenvolvida para a geração semiautomática de ontologias e que tem como entrada um hiperdocumento escrito em linguagem natural que descreve os simbolos de um universo de informação (UdI) e como saída uma ontologia em um arquivo do tipo daml [3].

Esta ferramenta permite automatizar uma quantidade razoável das tarefas de geração de ontologias oferencendo sugestões ao usuário quando necessitar de intervenção. A Figura 2.14 ilustra a interface inicial do C&L utilizando o domínio de sobremesas.

Inicialmente, são listados os termos com base no LAL de acordo com o seu tipo (verbo, objeto, sujeito ou estado). Estes termos são separados em três listas: conceitos ou classes (C_c), propriedades (P) e axiomas (A), para que o LAL identifique os elementos da ontologia.

Na segunda fase, um algoritmo verifica os impactos (resposta comportamental) de cada termo identificado pelo LAL, ou seja, checa se o termo já faz parte da lista de propriedades da ontologia. Caso não faça parte (a propriedade

[3]DARPA Agent Markup Language

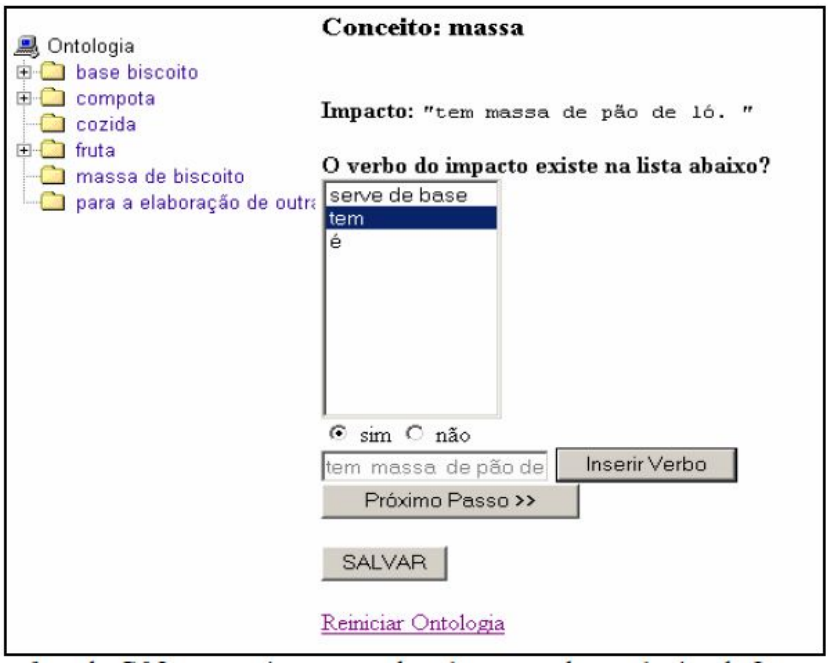


Figura 2.14: Visão geral da interface da ferramenta C&L

.

ainda não exista), é adicionada uma nova propriedade na lista. O nome da propriedade deve ser baseado no verbo utilizado para descrever o impacto (ocorrência).

Na terceira fase o algoritmo verifica a existência de termos disjuntos, checa se as classes possuem um relacionamento do tipo disjunto (por exemplo, macho e fêmea), se encontrar adiciona o *disjoint* à lista de axiomas e verifica a consistência.

Na fase seguinte, um algoritmo verifica se os termos do tipo verbo fazem parte da parte da lista de propriedades da ontologia, caso faça, adiciona uma nova propriedade na lista e verifica a consistência.

Na fase cinco, o algoritmo classifica os termos identificados utilizando uma lista de símbolos para cada termo. Para cada impacto (resposta comportamental) classificado o algoritmo tenta identificar a importância relativa do termo para a ontologia por meio de perguntas iniciadas por *"quando"*, *"onde"*, *"o quê"*, *"quem"*, *"porquê"* e verifica se estas classes são disjuntas, se for verdadeiro adiciona à lista de axiomas. Se for um termo central à ontologia, são classificados como classes (C), senão como propriedades (R), após isso é novamente verificada a consistência.

Na ultima fase, é verificado se existem conjuntos de conceitos que compartilham relações idênticas. Para cada conceito que compartilha as mesmas relações é construída uma lista de conceitos separada e realizada uma busca na

ontologia de conceitos que fazem referência a todos os membros da lista. Logo após é construída a hierarquia de conceitos onde todos os membros da lista são sub-conceitos do conceito encontrado. Novamente é verificada a consistência, o algoritmo é finalizado e todos os termos são adicionados na ontologia.

O produto final é uma ontologia no formato *daml* composta por clases, hierarquias, propriedades e axiomas. O uso desta ferramenta permite automatizar uma quantidade razoável das tarefas de geração de ontologias. A Figura 2.15 mostra um trecho do arquivo em DAML+OIL no domínio da sobremesa, utilizado para o desenvolvimento do processo.


```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <rdf:RDF xmlns:daml='http://www.daml.org/2001/03/daml+oil# xmls:doc="http://purl.org/dc/elements/1.1/"
   xmls:oiles='http://img.cs.mon.ac.uk/oil/oiles#' xmls:ref="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmls:rdf="http://www.w3.org/2000/01/rdf-schema#"xmls:xsd='http://www.w3.org/2000/10/XMLSchema#">
 - <daml:Ontology rdf:about="">
    <dc:title>sobremesa</dc:title>
    <dc:date>1-09-2003 14:54:56</dc:date>
    <dc:creator>carol</dc:creator>
    <dc:description>sobremesa</dc:description>
    <dc:subject>sobremesa</dc:subject>
    <daml:versionInfo>11</daml:versionInfo>
   </daml:Ontology>
  + <daml:Class rdf:abourt='http://pes.inf.puc-rio.br/pes03_1_1/Site/desenvolvimento/teste/sobremesa__1-09-2003_14-54-
    56.daml#apresentada fria'>
  + <daml:Class rdf:abourt='http://pes.inf.puc-rio.br/pes03_1_1/Site/desenvolvimento/teste/sobremesa__1-09-2003_14-54-
    56.daml#base biscoito'>
    <rdfs:label>base biscoito</rdfs:label>
  - <rdfs:comment>
     <![CDATA[ <a title="lexico" href="main.php?t=1&id=231">sobremesa</a> de corte cuja <a title="lexico"
     href="main.php?t=1&id=234">massa</a> é elaborada com biscoitos ao invés de farinha de trigo. ]]>
     </rdfs:comment>
    + <oiled:creationDate>
    + <oiled:creator>
    - <rdfs:subClassOf>
      - <dal:Restriction>
        <daml:onProperty rdf:resource:'http://pes.inf.puc.rio.br/pes03_1_1/Site_desenvolvimento/teste/sobremesa__1-
         09-2003_14-54-56.daml#tem"/>
       - <daml:hasClass>
          <daml:Class rdf:about='http://pes.inf.puc rio.br/pes03_1_1/Site/desenvolvimento/teste/sobremesa__1 09
           2003_14-54-56.daml#massa de biscoito"/>
          </daml:hasClass>
         </daml:Restriction>
        </rdfs:subClassOf>
       </daml:Class>
```

Figura 2.15: Trecho de um arquivo em DAML+OIL gerado pela ferramenta C&L

.

2.4.3 Poronto: ferramenta para a construção semiautomática de ontologias em português

Poronto [15] é uma ferramenta *web* semiautomática desenvolvida para construir ontologias a partir de textos em português (corpus), através de uma abordagem quantitativa no domínio da área da saúde.

Para seu desenvolvimento se faz necessário a participação de um engenheiro do conhecimento, para representar o conhecimento a ser expresso e de um especialista do domínio para transferir tal conhecimento de maneira adequada.

A ferramenta trabalha na extração de termos (conceitos) e hierarquias (H) a partir de um determinado corpus através de duas etapas: "Criação do Corpus" e "Criação da Ontologia". A fase de "Criação do Corpus" pode ser visualizada na Figura 2.16.

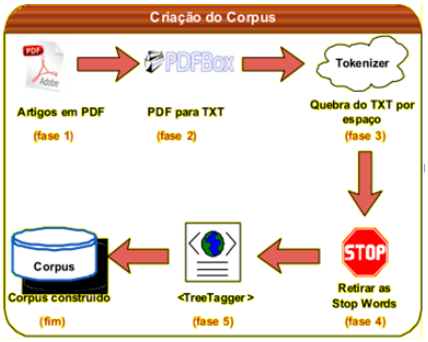


Figura 2.16: Processo de construção do corpus adotado pela Poronto proposta por Malucelli

.

A criação do corpus é composta por cinco atividades, primeiramente o usuário submete ao processo textos em formato pdf, a segunda atividade transforma os arquivos submetidos em para o formato txt (sem os marcadores padrões do pdf), a terceira atividade faz um pré-processamento dos textos através de anotações linguísticas; a quarta atividade realiza a extração das *stop words*[4] e, na última atividade os textos são processados para que sejam identificadas as *taggers*.

Ao final da criação do corpus cabe ao desenvolvedor criar a ontologia, para isso são executadas oito atividades: "Inserção do Corpus", "Extração de Termos Simples", "Extração de Termos Compostos", "Seleção de Sinônimos", "Seleção de Termos DeCS", "Lista de Termos Extraídos", "Taxonomia", "Resultado para Exportar". Estas fases são ilustradas na Figura 2.17 e detalhadas na Tabela 2.3.

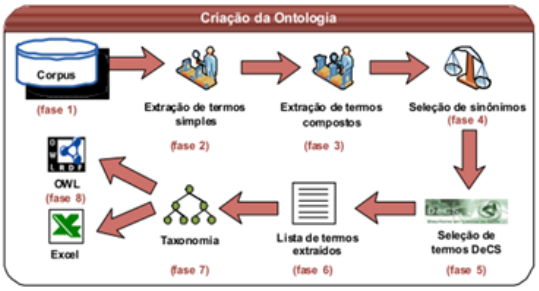


Figura 2.17: Visão geral do processo de criação da ontologia Poronto proposto por Malucelli

.

[4]Remoção de palavras que não são significativas do ponto de vista do domínio representado pela ontologia, por exemplo: preposições e artigos

Tabela 2.3: Detalhamento das fases realizadas pela ferramenta Poronto

Criação da Ontologia	
Fases	**Processamento**
Inserção do Corpus	Processamento do corpus para seleção de termos simples ou compostos, que serão processados pelo TreTagger ou através de medidas de td-idf, podendo usar ou não medidas de entropia na seleção dos termos
Extração de Termos Simples	Extração de termos simples a partir da aplicação de medidas de frequência, tf-idf e entropia.
Extração de Termos Compostos	Extração de termos compostos com base em regras expressas por sequências de tipos morfológicos.
Seleção de Sinônimos	Busca por sinônimos dos termos realizada em uma lista do OpenThesaurusPT para facilitar o critério de seleção do termo pelo usuário.
Seleção de Termos DeCS	Realização de uma pesquisa para verificar se os termos extraídos possuem correspondência na lista de Descritores em Ciência da Saúde (DeCS) (devido utilizar o domínio da saúde) e também para facilitar o critério de seleção do termo pelo usuário.
Lista de Termos Extraídos	Extração de termos simples e compostos, realizados pela ferramenta e apresentados aos usuários para verificar a existência de ou não, de sinônimos na lista de descritores. Após a apresentação, devem ser selecionados pelos usuários os termos mais relevantes a fim de que sejam inseridos na ontologia.
Taxonomia	O usuário pode optar pela organização, dos termos relevantes que foram selecionados, em uma taxonomia.
Resultado para Exportar	Possibilidade de exportação dos resultados para o formato XLS ou OWL, onde serão formalizados os termos ou conceitos (C) em uma hierarquia (H).

O resultado do processamento pode ser visualizado no editor de ontologias Protégé através da figura 2.18 que apresenta um trecho da ontologia gerada.

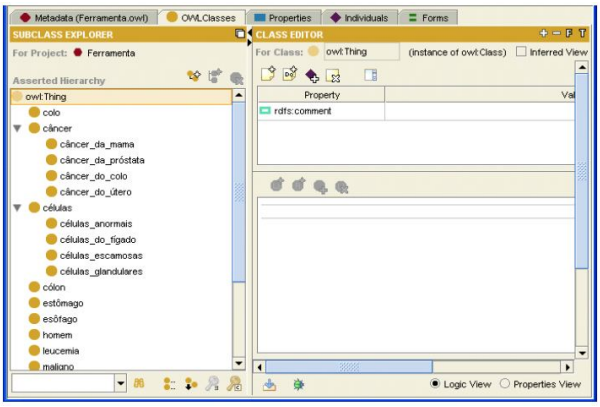


Figura 2.18: Ontologia gerada no Protégé pela ferramenta Poronto

A Tabela 2.4 apresenta um resumo comparativo das abordagens descritas neste estado da arte apresentando o produto de entrada, as tecnologias utilizadas, o elemento gerado ou aprendido e o nível de automação em cada uma delas. Na seção 2.5 é apresentada uma análise comparativa que agrupa em uma tabela as demais abordagens apresentadas neste trabalho a fim de que sejam relacionados os critérios de comparação adotados.

Tabela 2.4: Análise comparativa entre as abordagens do estado da arte

	Produto de Entrada	**Tecnologias Utilizadas**	**Elemento Gerado ou Aprendido**	**Nível de Automação**
AGROVOC	Corpus e Conceitos	WORDNET	C, H, R	Automática
C&L	Textos	LAL	C, H, R, A	Semiautomática
PORONTO	Corpus	TreeTagger	C, H	Semiautomática

2.5 Análise comparativa das abordagens de aprendizagem de ontologias

Esta seção apresenta uma análise comparativa das técnicas e ferramentas descritas nas seções 2.3 e 2.4 que descrevem respectivamente as técnicas para aprendizagem de ontologias propostas no GESEC e outras do estado da arte. Para cada abordagem da fundamentação teórica e do estado arte é apresentado uma descrição geral, incluindo seus objetivos e o resultado obtido por cada uma delas bem como é também discutido as limitações de cada abordagem. A análise comparativa realizada leva em consideração quatro critérios: o produto de entrada no processo, as tecnologias utilizadas, o elemento gerado ou aprendido e o nível de automação. A análise comparativa realizada tem o intuito de avaliar e interpretar cada processamento considerando seus resultados obtidos.

Uma ontologia de aplicação é composta de seis elementos: classes, taxonomia, relacionamentos, propriedades, axiomas e instâncias [10]. A partir da inserção de um conjunto de objetivos e fatos, a técnica GAODT [5] gera uma ontologia de aplicação composta por cinco destes elementos, porém, não trabalha na geração de

instâncias e, em uma ontologia as instâncias são consideradas como um conhecimento prévio existente nas ontologias.

O Povoamento de Ontologias [3] constitui uma abordagem para automatizar ou semiautomatizar a instanciação de propriedades e de relacionamentos não taxonômicos de classes de ontologias, este conhecimento é descoberto a partir de fontes textuais [25]. Dentre as técnicas desenvolvidas pelo grupo GESEC e as do estado da arte analisadas neste trabalho, somente a DIPPAO [3] trabalha na geração de instâncias da ontologia. Porém, DIPPAO [3] não trabalha na geração dos demais elementos que compõem a definição formal de ontologias.

As ontologias de aplicação são utilizadas pelos modernos SBC no processo de tomada de decisões, isso é possível devido ao principal elemento que compõe uma ontologia de aplicação, os axiomas [10]. Dentre as abordagens analisadas nesse trabalho, somente a GAODT [5], a TAILP [23] e o C&L [14] trabalham com a geração de axiomas durante o seu processo. No entanto, o resultado destas abordagens não contemplam a geração de uma ontologia completa.

A aquisição de relações taxonômicas a partir de fontes textuais na área de aprendizagem de ontologias seja de forma automática ou semiatutomática, é um processo que vem sendo adotado durante o desenvolvimento de diversas abordagens, tais como: um processo para a aquisição de relações taxonômicas de uma ontologia [6], um método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos – AFC [7], AGROVOC [13] e C&L [14]. Todavia, classes necessitam de relacionamentos e estas abordagens não contemplam a geração de relações não taxonômicas e nem dos demais elementos da ontologia. TARNT [8] por sua vez, trabalha na geração automática de relacionamentos não taxonômicos a partir de fontes de informações textuais, porém gera apenas o elemento R.

Boas técnicas devem considerar também o reúso de ontologias existentes, C&L [14] e a GAODT [5] são as únicas abordagens que trabalham com o reúso, sendo que o C&L [14] possui como característica particular em realção às demais, esta abordagem gera uma ontologia no formato de um arquivo do tipo *daml* (padrão mais antigo da W3C).

Por fim, a ferramenta Poronto [15] visa a extração de termos a partir de fontes textuais, estes termos podem tornar-se os conceitos e relações taxonômicas

constituintes de uma ontologia. Esta abordagem difere das outras, devido o fato de construir ontologias através da ponderação dos termos de um documento utilizando as medidas de TF-IDF e TreTagger, gerando automaticamente como produto final uma ontologia com classes (Cc) organizadas hierarquicamente (H).

A Tabela 2.5 apresenta um resumo comparativo das abordagens para a aprendizagem de ontologias discutidas conforme os critérios de comparação adotados. Também pode ser observado que as abordagens não extraem uma ontologia completa, sendo frequentemente voltadas a alguns elementos apenas, com exceção do Apponto-Pro que é a principal proposta deste trabalho. Percebe-se ainda que existem dois tipos de nível de automação: automático, quando não há intervenção do usuário, e a semiautomática, quando a intervenção do usuário é necessária em algum momento do processo.

Tabela 2.5: Análise comparativa entre as abordagens da fundamentação teórica e estado da arte

	Produto de Entrada	**Elementos gerados ou aprendidos**	**Suporte para o reuso**	**Nível de automação**
GAODT	Objetivos e Fatos	C, H, R, P, A	Sim	Semiautomática
DIPPAO	Corpus e Ontologia	C, I	Não	Automática
TAILP	Ontologia Povoada	A	Sim	Semiautomática
TARNT	Corpus	R	Não	Automática
Processo para aquisição de C e H	Corpus	C, H	Sim	Automática
Método para aquisição de H	Corpus	C, H	Sim	Automática
AGROVOC	Corpus e Conceitos	C, H, R	Não	Semiautomática
C&L	Textos	C, H, P, A	Sim	Automática
PORONTO	Corpus	C, H	Sim	Semiautomática
APPONTO PRO	Objetivos, Fatos, Corpus e Ontologia	C, H, R, I, P, A	Sim	Semiautomática

Conforme a análise realizada verifica-se que todas as abordagens realizam de forma automática ou semiautomática a construção de ontologias, para isso necessitam de um produto de entrada para que possam gerar ou aprender diferentes elementos da ontologia. As abordagens DIPPAO [3], TARNT [8], um processo para a aquisição de relações taxonômicas de uma ontologia [6], o método para a aprendizagem de hierarquia de conceitos utilizando AFC [7], AGROVOC [13], C&L [14] e PORONTO [15] utilizam como fonte de informação um corpus em seu processamento para que possam extrair as informações necessárias que permitirão construir uma ontologia.

A DIPPAO [3], utiliza tanto fontes de informações textuais quanto uma ontologia em seu processamento. Já a TAILP [23] necessita apenas de uma ontologia povoada para que possa realizar suas inferências indutivas. A GAODT [5], é a única das abordagens descritas anteriormente que utiliza, em particular, um conjunto de objetivos e fatos durante seu processamento.

O Apponto-Pro, principal proposta deste trabalho, realiza por sua vez, a integração de todas as particularidades identificadas nestas abordagens, apresentando como vantagem para o desenvolvedor, a possibilidade de empregar vários tipos de informações em um só ambiente de desenvolvimento. Uma das limitações destas abordagens é a geração de elementos da ontologia de forma isolada, enquanto que o Apponto-Pro gera uma ontologia de aplicação com todos os elementos de forma conjunta.

2.6 Considerações finais

Este capítulo apresentou a área de pesquisa de construção e aprendizagem de ontologias bem como algumas das principais técnicas e ferramentas, desenvolvidas pelo grupo GESEC, utilizadas para o desenvolvimento de ontologias e que são empregadas pelo Apponto-Pro.

Posteriormente, foram apresentadas outras abordagens desenvolvidas para a construção de ontologias que realizam a aprendizagem dos diferentes elementos da ontologia, como por exemplo o *framework* AGROVOC [13] que consiste em processo automático para a construção de ontologias que aprende tanto relações taxonômicas

quanto relações entre os conceitos. A abordagem C&L [14] que é uma ferramenta semiautomática para obtenção de classes, hierarquias, relacionamentos e axiomas de uma ontologia. A Poronto [15] é uma ferramenta para a construção semiautomática de ontologias que extrai a partir de um corpus classes e hierarquias de uma ontologia.

Por fim, com o objetivo de explicitar a contribuição feita pelo Apponto-Pro para a área de aprendizagem e povoamento de ontologias, uma avaliação qualitativa sobre todas as abordagens foi realizada considerando quatro critérios de comparação: produto de entrada, tecnologias utilizadas, elemento gerado ou aprendido e nível de automação.

3 APPONTO-PRO: UM PROCESSO INCREMENTAL PARA O APRENDIZADO E POVOAMENTO DE ONTOLOGIAS DE APLICAÇÃO

Este capítulo descreve a principal contribuição deste trabalho, o Apponto-Pro, um processo incremental para o aprendizado e povoamento de ontologias de aplicação que integra seis técnicas desenvolvidas no contexto do grupo GESEC: GAODT [5], um processo para a aquisição de relações taxonômicas de uma ontologia [6], um método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos (AFC) [7], TARNT [8], DIPPAO [3] e TAILP [9].

Este capítulo está organizado da seguinte maneira. A seção 3.1 descreve o processo Apponto-Pro. As seções 3.2 a 3.6 descrevem as fases do processo proposto e na seção 3.7 são apresentadas as considerações finais do capítulo.

3.1 Visão geral do processo

O processo Apponto-Pro tem como objetivo principal a construção de uma ontologia de aplicação completa através da integração de diferentes técnicas que geram elementos da ontologia de forma isolada. Esta integração permite a aprendizagem, o enriquecimento, o povoamento e a geração de novos axiomas da ontologia de maneira dinâmica e incremental, através de uma abordagem guiada por objetivos que permite a formalização dos objetivos de um sistema baseado no conhecimento.

A Figura 3.1 ilustra a visão geral do processo Apponto-Pro juntamente com suas seis fases: “Construção da Ontologia Base”, “Aprendizagem de Classes e Relações Taxonômicas”, “Aprendizagem de Relacionamentos não Taxonômicos”, “Povoamento de Ontologia”, “Inserção de Axiomas” e “Incrementar Ontologia”.

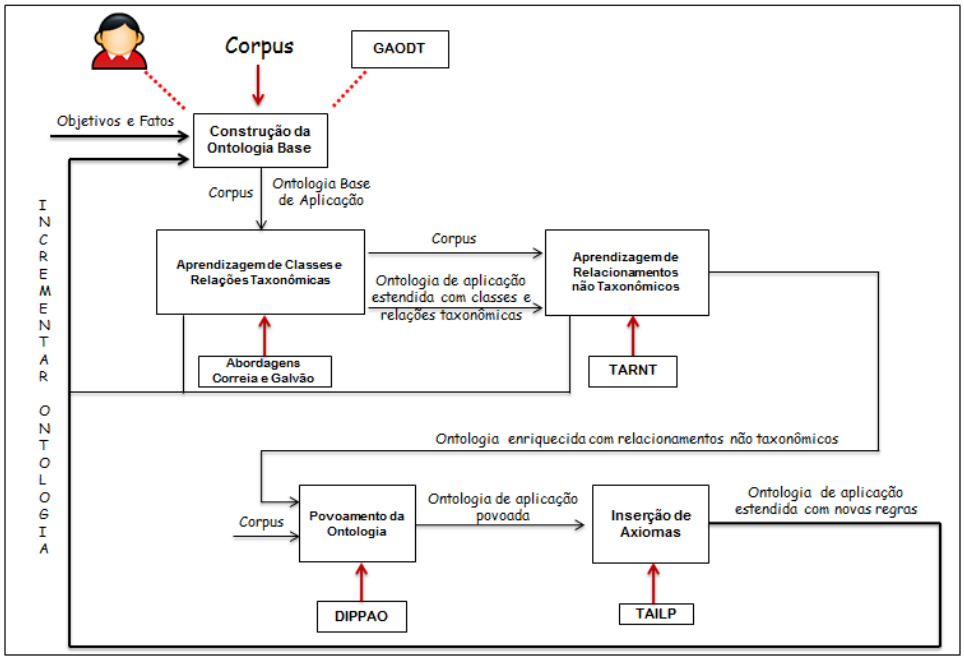


Figura 3.1: Visão geral do processo Apponto-Pro

A fase "Construção da ontologia base" tem como entrada um conjunto de objetivos e fatos de um determinado domínio de conhecimento. Os objetivos são os requisitos que o SBC pretende alcançar e os fatos são as informações que complementam esses objetivos. Por exemplo, objetivos como *"Find the father of a person"* e *"Identify the son of a person"* se sustentam de fatos como *"A person has father"* e *"A person has son"*. Para que sejam alcançados, os objetivos são representados na forma de axiomas da ontologia por meio da aplicação da técnica GAODT proposta por [5]. A tarefa de elaboração do conjunto de objetivos e fatos cabe ao engenheiro do conhecimento e ao especialista do domínio, há ainda a participação de um usuário para dar continuidade ao desenvolvimento das fases seguintes. Ao final do processamento, uma ontologia de aplicação é gerada composta por classes, hierarquias, propriedades, relacionamentos não taxonômicos e axiomas, elementos estes que guiarão todo o restante do processo.

A segunda fase do processo "Aprendizagem de Classes e Relações Taxonômicas" visa a aprendizagem de novas classes e relacionamentos taxonômicos da ontologia através das abordagens automáticas de um processo para a aquisição de relações taxonômicas de uma ontologia proposta por [6] e um método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos - AFC proposta por [7], cujo nível de intervenção humana é menor, permitindo assim

a inclusão automática de mais elementos à ontologia em desenvolvimento. A fase tem como entrada a ontologia gerada na fase anterior e um corpus, gerando como saída uma ontologia enriquecida pelos elementos C_c e H aprendidos.

A fase "Aprendizagem de Relacionamentos não Taxonômicos" tem como objetivo realizar a aprendizagem, o enriquecimento e a extensão da ontologia em desenvolvimento, desta vez com novos relacionamentos não taxonômicos. Esta fase tem como entrada a ontologia gerada na fase anterior mais um corpus que são processados através das atividades da técnica TARNT proposta por [8]. TARNT [8] aplica técnicas estatísticas e PLN para extrair novos relacionamentos não taxonômicos a partir do corpus dado como entrada, estas relações são adicionadas à ontologia em desenvolvimento. Por exemplo, para a sentença *"The young couple have two children"* identificada no corpus é gerado o relacionamento *"have(couple, child)"* através da identificação dos conceitos *"couple"* e *"child"*. O produto deste processamento é uma ontologia de aplicação estendida com novas classes e relações não taxonômicas.

Vale ressaltar, que a ontologia desenvolvida até esta fase do processo não possui instâncias, portanto a fase de "Povoamento de Ontologia" faz-se necessária para suprir tal necessidade. Esta fase recebe como entrada a ontologia desenvolvida anteriormente e um corpus a fim de que as classes que foram geradas e aprendidas possam ser instanciadas. O desenvolvimento desta fase é realizado por intermédio das atividades referentes à técnica DIPPAO proposta por [3] que aplica técnicas de processamento da linguagem natual e extração de informação para extrair automaticamente intâncias de um corpus e adicioná-las à ontologia. Por exemplo, dada a classe *"Person"* presente na ontologia, podem ser identificadas e associadas instâncias a ela como *"Stuart"* e *"John"* que encontram-se no corpus.

A fase "Inserção de Axiomas" tem o objetivo de gerar novos axiomas na ontologia, permitindo a integração das regras de LPO desenvolvidas na fase "Construção de Ontologia Base", com as regras de LPO geradas a partir desta. O produto de entrada desta fase é a ontologia povoada anteriormente para que sejam gerados novos axiomas através da aplicação da técnica TAILP proposta por [9]. Esta técnica realiza inferências indutivas através da aplicação de algoritmos de PLI na ontologia povoada, retornando a um conjunto de axiomas que são representados de maneira explícita na ontologia. Por exemplo, dada uma ontologia povoada que possui a relação *"Father_of(Man, Person)"*, as instâncias que participam desta relação

são *"Father(Stuard,Jon)"*, consequentemente o processo retorna a cláusula em LPO *"Grandfather_of(X, Y) ← Father_of(X, Z), Father_of(Z, Y)"*, esta cláusula é transformada em axioma o qual é inserido na versão final da ontologia de aplicação, que agora é composta por todos os elementos da definição formal: C_c, H, R, I, P e A. É possível incrementar esta ontologia, adicionando novos elementos em seu produto final.

As próximas seções descrevem em detalhe cada uma das fases do processo Apponto-Pro, que são ilustradas através de exemplos no domínio do Direito de Família.

3.2 Construção de ontologia base

Esta fase tem como entrada um conjunto de objetivos e fatos sobre um determinado domínio de aplicação. Para que a ontologia base seja construída, primeiramente é necessário estabelecer o objetivo principal que o sistema deseja alcançar juntamente com os objetivos específicos e o conjunto de fatos. Estes são elaborados e selecionados em consenso pelo engenheiro do conhecimento e pelo especialista do domínio a fim de que sejam submetidos de forma manual ao processo por intermédio de um usuário. A Figura 3.2 ilustra a visão geral desta fase e a Tabela 3.1 descreve um exemplo dos objetivos e fatos a ser submetidos.

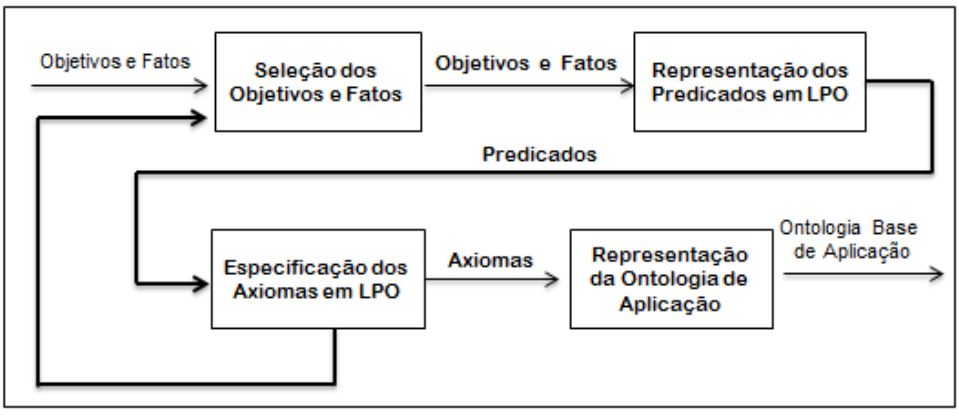


Figura 3.2: Visão geral da fase "Construção da ontologia base"

Tabela 3.1: Trecho da lista de objetivos e fatos submetidos à fase Construção de Ontologia Base

Seleção de Objetivos e Fatos	
Determine the grandfather of a person	*Goal*
Identify the father of a person	*Goal*
Identify the son of a person	*Goal*
A person has a father	*Fact*
A person has a son	*Fact*

Esta fase obedece a execução de três atividades referentes à técnica GAODT proposta em [5]: "Representação dos Predicados em Lógica de Primeira Ordem (LPO)", "Especificação dos Axiomas em LPO" e "Especificação/Extensão da Ontologia de aplicação", detalhadas nas próximas subseções.

3.2.1 Representação dos predicados em lógica de primeira ordem

Esta atividade consiste em traduzir os objetivos e fatos inseridos em linguagem natural para linguagem computacional, esta conversão é realizada através de sete subatividades: "Identificação das Entidades", "Redefinição das Entidades", "Identificação dos Relacionamentos", "Redefinição dos Relacionamentos", "Definição da Aridade", "Definição dos Predicados" e "Redefinição das Entidades do Predicado", ilustradas na Figura 3.3 e descritas nas subseções seguintes.

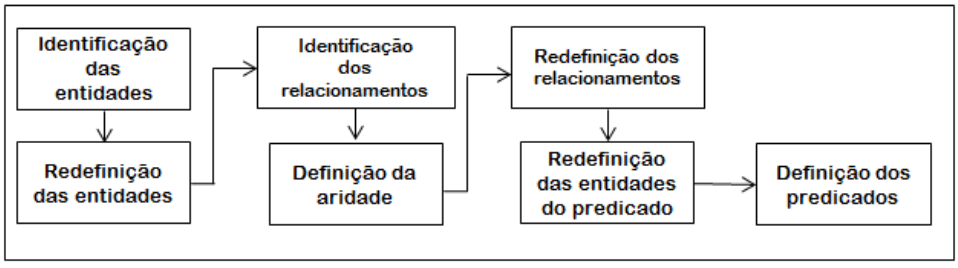


Figura 3.3: Visão geral das subatividades da "Representação dos Predicados em LPO"

a. Identificação das Entidades

Nesta subatividade são identificadas as entidades presentes nos objetivos inseridos. Essas entidades são todos os sujeitos e objetos implícitos ou não na lista

selecionada na atividade anterior. Por exemplo, para o objetivo *"Identify the grandfather of a person"* são identificadas as entidades *"Father"* e *"Person"*, como ilustrado na Tabela 3.2.

Tabela 3.2: Entradas e saídas das entidades identificadas

Itens Selecionados	**Entidades**
Determine the grandfather of a person	*Grandfather, Person*
Identify the father of a person	*Father, Person*
Identify the son of a person	*Son, Person*

b. Redefinição das Entidades

Para o desenvolvimento desta subatividade, são utilizadas as entidades que foram selecionadas na fase anterior a fim de que seja verificado se o que foi selecionado é uma entidade ou um relacionamento, ou seja, leva-se em consideração se o que foi identificado como entidade na fase anterior pode ser relacionado. Por exemplo, para o objetivo *"Identify the grandfather of a person"* foram identificadas as entidades *"Father"* e *"Person"* e redefinidas para *"Person"* e *"Person"*, assim como ilustrado na Tabela 3.3.

Tabela 3.3: Entradas e saídas das entidades redefinidas

Entidades	**Redefinição das Entidades**
Grandfather, Person	*Person,Person*
Father, Person	*Person,Person*
Son, Person	*Person,Person*

c. Identificação dos Relacionamentos

Esta subatividade considera os itens selecionados na subatividade "Identificação das Entidades", para que seja identificado o relacionamento existente entre eles. Os relacionamentos selecionados são as frases nominais identificadas nos objetivos inseridos. Por exemplo, para os itens *"Identify the grandfather of a person"* e

"Identify the son of a person", são identificados os relacionamentos *"Grandfather"* e *"Son"*. A Tabela 3.4 ilustra os relacionamentos encontrados nos itens selecionados.

Tabela 3.4: Entradas e saídas dos relacionamentos identificados

Itens Selecionados	**Relacionamentos**
Determine the grandfather of a person	*Grandfather*
Identify the father of a person	*Father*
Identify the son of a person	*Son*

d. Redefinição dos Relacionamentos

Nesta subatividade as relações encontradas na etapa anterior são redefinidas. Por exemplo, para a redefinição da entidade *"Grandfather"* será adicionado o conectivo *"of"* resultando na relação *"Grandfather_of"*. Este conectivo tem a função de permitir a busca por instâncias no corpus utilizado na fase "Povoamento de Ontologia". A redefinição pode ser vista na Tabela 3.5.

Tabela 3.5: Entradas e saídas dos relacionamentos redefinidos

Relacionamentos	**Entidades/Palavras redefinidas**
Grandfather	*Grandfather_of*
Father	*Father_of*
Son	*Son_of*

e. Definição da Aridade

Esta subatividade consiste em especificar o número de entidades encontradas a fim de que seja definido o grau de aridade. Dependendo da quantidade de entidades declaradas podem ser classificadas em aridade unária (uma entidade), binária (duas entidades) e terciária (três entidades). A Tabela 3.6 mostra uma aridade binária entre as entidades.

Tabela 3.6: Declaração da aridade entre as entidades

Relacionamentos	**Entidades**	**Aridade**
Grandfather_of	*Grandfather,Person*	2
Father_of	*Father,Person*	2
Son_of	*Son,Person*	2

f. Definição dos Predicados

Esta subatividade visa concretizar a representação das entidades e dos relacionamentos identificados na subatividade anterior, na forma de lógica de primeira ordem, ou seja, os relacionamentos e entidades encontrados são transformados em regras de LPO, conforme ilustrado na Tabela 3.7.

Tabela 3.7: Entradas e saídas dos predicados definidos

Itens selecionados	**Predicados**
Determine the grandfather of a person	*grandfather_of(person,person)*
Identify the father of a person	*Father_of(Person,Person)*
Identify the son of a person	*Son_of(Person,Person)*

g. Redefinição das Entidades do Predicado

A redefinição das entidades do predicado visa diferenciar as entidades que possuem o mesmo nome, adicionando variáveis como X, Y ou Z para avaliá-las como entidades distintas. Por exemplo, duas entidades nomeadas por *"Person"*, ao ser redefinidas passam a ser *"PersonX"*, *"PersonY"* e *"PersonZ"*. A Tabela 3.8 mostra o exemplo desta redefinição.

Tabela 3.8: Entradas e saídas dos predicados redefinidos em LPO

Predicados	**Predicados redefinidos**
Grandfather_of(person,person)	*Grandfather_of(PersonX,PersonZ*
Father_of(Person,Person)	*Father_of(PersonY,PersonZ)*
Son_of(Person,Person)	*Son_of(PersonZ,PersonY)*

3.2.2 Especificação dos axiomas em LPO

Nesta atividade é realizada a transformação dos predicados que estão em LPO para que sejam definidos em regras, permitindo que os objetivos do sistema sejam alcançados. Esta atividade é composta por quatro subatividades: "Definição da Conclusão e Condição", "Definição dos Operadores Booleanos", "Definição dos Quantificadores" e "Definição de Implicação ou Equivalência", descrita nas próximas subseções.

a. Definição da Condição e Conclusão

Esta subatividade permite a escolha do que será definido como conclusão e condição. A conclusão é o principal objetivo que o sistema deseja alcançar e a condição é o subobjetivo que alcançará o objetivo principal. Por exemplo, dados os objetivos *"Determine the grandfather of a person"*, *"Identify the father of a person"* e *"Identify the son of a person"*, é escolhido como o principal objetivo que o sistema deseja alcançar o objetivo *"Determine the grandfather of a person"*. Os demais são classificados automaticamente pelo sistema como condição. O exemplo é apresentado na Tabela 3.9.

Tabela 3.9: Definição da condição/conclusão

Condição e predicados do axioma	**Conclusão**
Father_of(Person,Person)	*Grandfather_of(PersonX,PersonZ)*
Son_of(Person,Person)	

b. Definição dos Operadores Booleanos

Para esta subatividade são definidos os operadores booleanos $(\wedge)$, $(\vee)$, $(\neg)$ que correspondem respectivamente a conjunção, disjunção e negação. O operador de negação é utilizado para os casos em que se deseja tornar inválido algum dos predicados. A conjunção é empregada para os casos em que um predicado necessitar de outro para satisfazer o objetivo, se isso não ocorrer usa-se a disjunção. Por exemplo, para alcançar o objetivo *"(grandfather_of(PersonX, PersonZ))"* é necessário satisfazer o objetivo *"(father_of(PersonY, PersonZ))"*, para isso usa-se o operador de conjunção para integrar os dois predicados. A Tabela 3.10.

Tabela 3.10: Definição do operador booleano entre os predicados

grandfather_of (PersonX, PersonZ)
^
father_of(PersonY, PersonZ)

c. Definição dos Quantificadores

Para especificar a condição da regra de inferência, são definidos os quantificadores Universal $(\forall)$ e Existencial $(\exists)$ que ligam premissas diferentes. O quantificador Universal será usado para indicar que um predicado é verdadeiro dentre todos os elementos de um determinado conjunto e o quantificador existencial é usado caso exista pelo menos um elemento verdadeiro no conjunto. Por exemplo, "*PersonX*" se refere a pelo menos uma pessoa que participa que participa da entidade familiar, sendo assim o quantificador existencial $(\exists)$ é definido.

d. Definição de Implicação ou Equivalência

Esta subatividade consiste em determinar se o axioma a ser gerado é uma implicação ou equivalência. Será uma implicação quando a condição escolhida atende o objetivo, ou seja, satisfaz a conclusão. A equivalência será definida quando há uma harmonia entre a condição e a conclusão, ou seja, uma implica na outra. Por exemplo, uma implicação pode ser ilustrada pela seguinte regra "$\exists$ *PersonX*, $\exists$ *PersonY* | *father_of(personY, personZ)* $\wedge$ *son_of(personZ, personY)* $\rightarrow$ *grandfather_of (PersonX, PersonZ)*".

3.2.3 Representação da ontologia de aplicação

Esta atividade tem por objetivo extrair os elementos que compõem os axiomas que foram especificados durante a construção da ontologia a fim de que sejam representados na ontologia. A fase consiste em seis subatividades: "Tradução de Axiomas", "Definição das Classes", "Definição dos Relacionamentos não taxonômicos", "Definição dos relacionamentos taxonômicos", "Definição das Propriedades".

a. Tradução do Axiomas

Esta subatividade consiste em traduzir as regras em LPO que foram geradas e convertê-las automaticamente para uma linguagem de marcação de regras (RuleML). Esta linguagem foi utilizada devido a sua capacidade de expressar de forma satisfatória as regras em LPO. Um exemplo de tais axiomas pode ser visualizado na Figura 3.4.

```
<Assert>
    <Rulebase mapClosure='universal'>
<Implies>
<And>
<Atom>
<Rel>father  of</Rel>
<Var>Father</Var>
<Var>Person</Var>
</Atom>
<Atom>
    <Rel> Child of</Rel>
<Var>  Child  </Var>
<Var>Person</Var>
</Atom>
</And>
<Atom>
    <Rel>graddaughter  of</Rel>
    <Var>Granddaughter</Var>
    <Var>PessoaX</Var>
</Atom>
</Implies>
    </Rulebase>

</Assert>
```

Figura 3.4: Exemplo de axioma em RuleML gerado na fase "Construção da ontologia base"

b. Definição das Classes

Esta subatividade consiste em extrair dentre as variáveis existentes, aquelas que podem ser definidas como classes. Por exemplo, o predicado *"Grandfather_of(PersonX,PersonZ)"* possui as variáveis *"PersonX"* e *"PersonY"*, ambas pertencentes à superclasse *"Person"*.

c. Definição dos Relacionamentos não Taxonômicos

Nesta subatividade são extraídos os relacionamentos não taxonômicos da ontologia a partir da lista de axiomas gerada. Por exemplo, na relação *"Grandfather_of(PersonX,PersonZ)"*, é identificado o relacionamento não taxonômico *"Grandather_of"*, definindo assim a relação existente entre *"PersonX"* e *"PersonZ"*.

d. Definição dos Relacionamentos Taxonômicos

Esta subatividade consiste em extrair os relacionamentos taxonômicos existentes entre as classes que foram definidas na subatividade *"Definição das Classes"*, isto é, define a hierarquia existente entre as classes geradas. Por exemplo, entre as classes *"Person"*, *"Father"* e *"Son"*, há uma relação hierárquica, pois *"Father"* e *"Son"* pertencem à classe *"Person"*, assim como ilustrado na Figura 3.5.

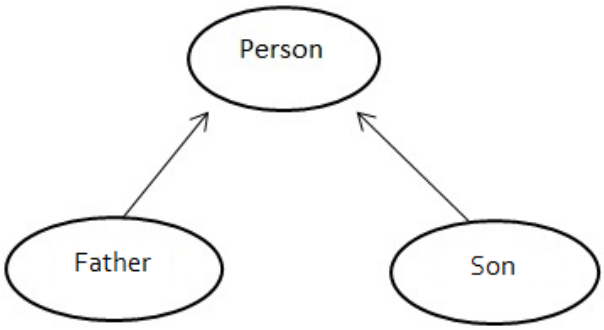


Figura 3.5: Exemplo de hierarquia de classes da ontologia no domínio do Direito da Família

e. Definição das Propriedades

Para esta subatividade leva-se em consideração as relações hierárquicas entre as classes definidas na subatividade anterior, a fim de que seja definido o domínio e a imagem pertencente a cada relacionamento. Por exemplo, na relação

"Father_of", o domínio e a imagem associado são *"Person"* e *"Person"*, como descrito na Tabela 3.11.

Tabela 3.11: Definição das propriedades da ontologia

Domínio	**Relação**	**Imagem**
Person	*Grandfather_of*	*Person*
Person	*Father_of*	*Person*

O produto final desta fase é uma ontologia no formato de um arquivo OWL contendo classes, hierarquias, relacionamentos, propriedades e axiomas. A Figura 3.6 apresenta um exemplo da ontologia construída ilustrando o produto de entrada e o produto de saída.

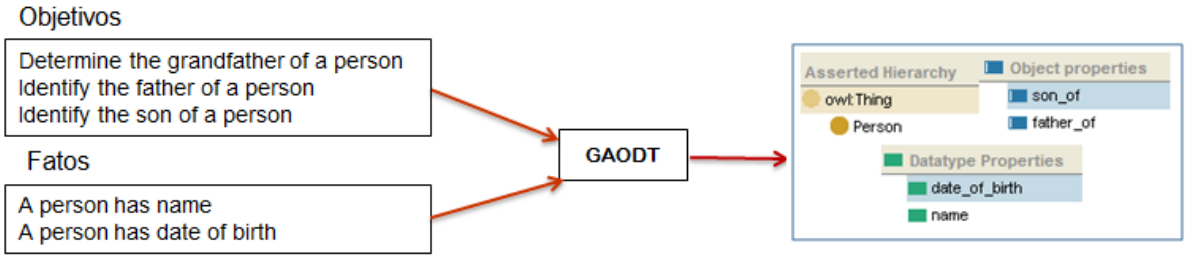


Figura 3.6: Exemplo da ontologia gerada nesta fase apresentando o produto de entrada e saída

3.3 Aprendizagem de classes e relações taxonômicas

O objetivo desta fase é aprender de maneira automática novas classes e hierarquias (C_c e H) de ontologias através da aplicação de técnicas de PLN e padrões heurísticos. Uma vez que as classes e hierarquias definidas na fase anterior são geradas a partir de um conjunto de objetivos e fatos que são fornecidos e desenvolvidos por meio de intervenção humana, esta fase visa a geração e inserção de novas classes e hierarquias à ontologia de forma automática. Para isso, tem como entrada a ontologia desenvolvida na fase precedente e um corpus de igual domínio. A Figura 3.7 ilustra a visão geral desta fase.

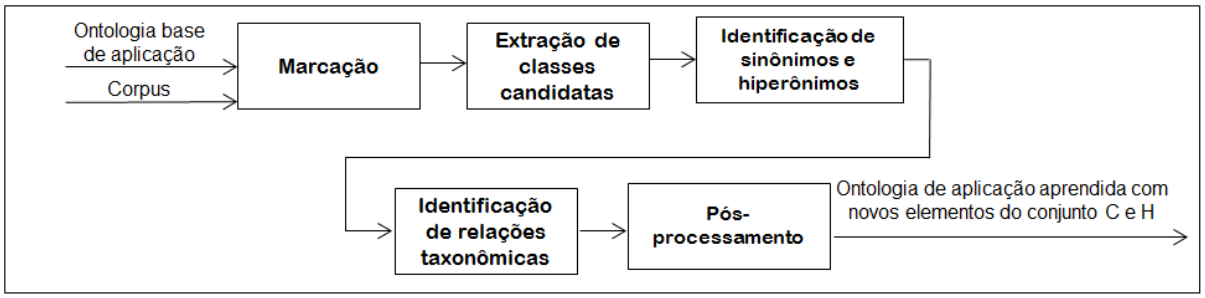


Figura 3.7: Visão geral da fase "Aprendizagem de classes e relações taxonômicas"

Para o seu desenvolvimento, conta com o suporte de um processo para aquisição de relações taxonômicas de uma ontologia propostos por [6] e um método para a aprendizagem de hierarquias de conceitos proposto por [7], baseando-se em cinco atividades: "Marcação", "Extração de Classes Candidatas", "Identificação de Hiperônimos e Sinônimos", "Identificação de Relações Taxonômicas" e "Pós-processamento", descritas nas seções seguintes.

3.3.1 Marcação

A atividade de "Marcação" tem como objetivo, processar o corpus fornecido em linguagem natural dado como entrada e transformá-lo em linguagem computacional. Para isso, são aplicadas técnicas de PLN para realizar a identificação de *tokens*, a divisão de sentenças, a lematização e a análise lexical.

A identificação de *tokens* busca identificar os *tokens* existentes no corpus. Para a divisão de sentenças os *tokens* que foram identificados são agrupados de acordo com o local em que estão localizados na sentença.

Na lematização os *tokens* são reduzidos para a sua forma básica, permindo o agrupamento de todos em um único lema. A análise lexical identifica as classes gramaticais sobre cada *token* classificando se é um verbo, um substantivo entre outros. Ao final deste processamento é gerado um corpus anotado. A Figura 3.8 ilustra um exemplo da marcação realizada.

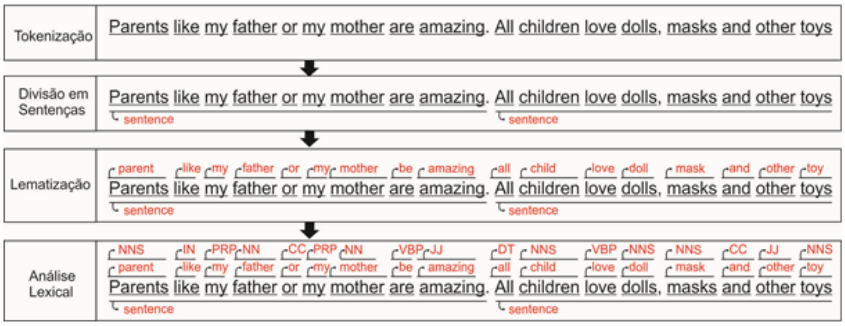


Figura 3.8: Exemplo da marcação realizada no corpus dado como entrada na fase "Marcação"

3.3.2 Extração de classes candidatas

Nesta subatividade é realizada a identificação de substantivos concretos e abstratos presentes no corpus, pois estes são considerados possíveis classes da ontologia. Na extração de *tokens* candidatos a classes, os substantivos próprios são desconsiderados sendo selecionados os substantivos concretos comuns e os abstratos, para que gerem as classes candidatas a ser adicionadas à ontologia. Por exemplo, identificada a sentença *"All children love dolls , masks and other toys"*, os substantivos *"children"*, *"doll"*, *"mask"* e *"toy"* são considerados como possíveis classes.

3.3.3 Identificação de sinônimos e hiperônimos

O objetivo desta fase é selecionar as classes que foram identificadas na atividade anterior de tal maneira que as classifica como sinônimo ou hiperônimo, para isso conta com o suporte da base de dados léxica do wordnet. Por exemplo, para as classes *"child"* e *"doll"* são identificados respectivamente, sinônimos como *"kid"* e *"dolly"* e *"juvenile"* e *"toy"* como hiperônimos.

3.3.4 Identificação de relações taxonômicas

Para a extração das relações taxonômicas são utilizados padrões heurísticos onde cada sentença identificada no corpus (fase marcação) é analisada para que seja realizado o casamento de padrões. Este casamento identifica as relações de hiponímia através da aplicação de expressões regulares sobre as sentenças, por exemplo, para a sentença *"All children love dolls, masks and other toys"* é possível observar que as classes *"dolls"* e *"mask"* são relacionados a classe *"toy"* através de uma relação de hiponímia,

ou seja, a classe *"toy"* é superclasse de *"dolss"* e *"mask"*, gerando assim um conjunto de relações taxonômicas (elemento H). Esta hierarquia é ilustrada na Figura 3.9.

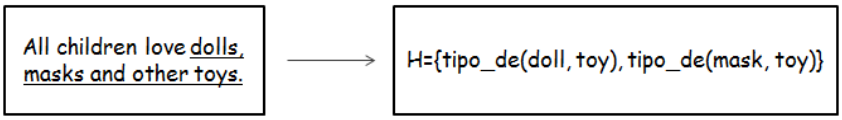


Figura 3.9: Exemplo de hierarquia extraída na identificação de relacionamentos taxonômicos"

3.3.5 Pós-processamento

Por fim, esta atividade tem o objetivo de representar em uma linguagem de especificação de ontologias as camadas C_c e H aprendidas na atividade anterior, gerarando o arquivo OWL.

3.4 Aprendizado de relações não taxonômicas

Esta fase tem o objetivo é aprender relações não taxonômicas através da aplicação de técnicas de PLN e estatísticas de maneira que realiza o enriquecimento e a extensão da ontologia em desenvolvimento. A entrada desta fase é a ontologia gerada na fase anterior e um corpus que descreve conceitos de domínio idêntico ao que vem sendo utilizado. A Figura 3.10 ilustra a visão geral desta fase.

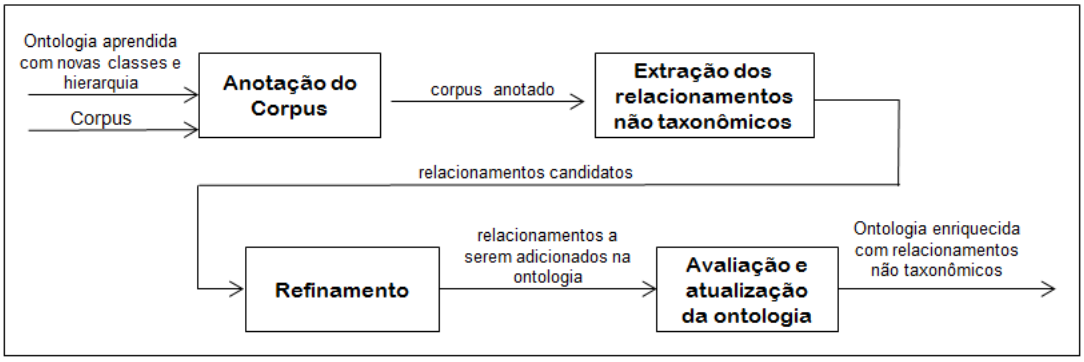


Figura 3.10: Visão geral da fase "Aprendizagem de relações não taxonômicas"

Sua execução segue quatro atividades referentes à técnica TARNT proposta por [8]: "Anotação do Corpus", "Extração de Relacionamentos", "Refinamento", "Avaliação e Atualização da Ontologia".

3.4.1 Anotação do corpus

Esta atividade tem o objetivo de marcar o corpus dado como entrada para que seja realizado anotações necessárias à extração dos relacionamentos candidatos, através da aplicação de cinco técnicas de PLN que incluem: "Tokenização", "Separação de Sentenças", "Lematização", "Análise Lexical" e *"VP Chunking"*. Estas técnicas são descritas e exemplificadas na Tabela 3.12.

Tabela 3.12: Descrição das técnicas de PLN aplicadas na técnica TARNT

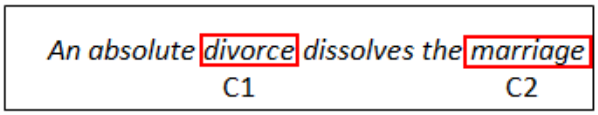
Técnicas de PLN	
Tokenização	Divisão do texto em tokens, que são unidades mais simples. Ex: An / absolute / divorce / dissolves / the / marriage
Separação de Sentenças	Divisão do texto em sentenças. Ex: An absolute divorce dissolves the marriage / Do not relly on the divorce of your spouse's lawyer./
Lematização	Tipo de normalização morfológica que deriva a forma lematizada da palavra original e incrementa o recall da busca por conceitos. Ex: walks e walking = walk
Análise Lexical	Classifica os termos em classes gramaticais.
VP Chunking	Identifica as frases verbais que são sugeridas como rótulos dos relacionamentos. Ex: An absolute divorce dissolves the marriage VP Chunking = "dissolves"

O resultado deste processamento é um corpus anotado que é dado como entrada à próxima atividade "Extração de relacionamentos".

3.4.2 Extração dos relacionamentos

Para a extração dos relacionamentos é disponibilizado ao especialista do domínio um conjunto de regras de extração: "Regra de Sentença", "Regra de Sentença com Frase Verbal", e "Regra de Apóstrofo", estas são aplicadas para extrair do corpus previamente anotado os relacionamentos candidatos.

A "Regra de Sentença" é baseada na heurística de que dois conceitos consecutivos em uma mesma sentença estão provavelmente relacionados de forma não-taxonômica. Por exemplo, para a sentença *"An absolute divorce dissolves the marriage"* é possível identificar dois conceitos *"divorce"* e *"marriage"* que retorna a tupla *(divorce, marriage)*.



De acordo com a regra de sentença aplicada, os conceitos C1 e C2 são extraídos por estarem relacionados respectivamente nas extremidades esquerda e direita do texto.

A "Regra de Sentença com Frase Verbal" é baseada na heurística de que dois conceitos consecutivos na mesma sentença com uma frase verbal entre eles estão provavelmente relacionados de forma não taxonômica, sendo a frase verbal o provável rótulo do relacionamento. Por exemplo, na sentença *"An absolute divorce dissolves the marriage"* são identificados os conceitos *"divorce"*, *"marriage"* e a frase verbal *"dissolves"*, retornando assim a tupla *(divorce,* ***disollves****, marriage)*.

A "Regra de Apóstrofo" baseia-se na heurística de que dois conceitos consecutivos com os caracteres "s" ou "'" entre eles tem grande probabilidade de estarem relacionados de forma não-taxonômica. Por exemplo, a sentença *"Do not rely on the advice of your spouse's lawyer"* retorna a tupla *(spouse, lawyer)*.

3.4.3 Refinamento

O refinamento é realizado sobre os relacionamentos candidatos identificados na atividade "Extração de Relacionamentos". Para tanto são disponibilizadas duas soluções: "Frequência de co-ocorrência" e *"Bag of labels"*. A frequência de co-ocorrência utiliza um algoritmo para filtrar tanto os relacionamentos candidatos resultantes da extração realizada com a regra de sentença quanto pela regra de apóstrofo.

Para o refinamento por *bag of labels* é utilizado para filtrar os relacionamentos candidatos obtidos da extração realizada com a regra de sentença com

frase verbal de maneira que calcula a frequência normalizada dos relacionamentos candidatos verificando a coincidência apenas dos dois conceitos, a frequência é incrementada e a frase verbal (fv) não é verificada. Para cada par de conceitos encontrado a frequência é incrementada e a "fv" é acrescentada em um repositório de rótulos – *"bag of labels"* – associado a esse par de conceitos. O resultado final deste processamento é um conjunto de relacionamentos não taxonômicos que indicam quais as classes estão relacionadas além dos rótulos que são sugeridos a cada relacionamento.

3.4.4 Avaliação e atualização da ontologia

Esta atividade tem o objetivo de permitir ao especialista selecionar e possivelmente editar os relacionamentos dentre os que foram extraídos pelas frases anteriores.Os relacionamentos selecionados são adicionados à ontologia permitindo o seu enriquecimento.

3.5 Povoamento de ontologia

Esta fase tem o objetivo de realizar o povoamento automático da ontologia a partir de fontes textuais. A entrada é a ontologia gerada na fase anterior e um corpus que descreve casos sobre o mesmo domínio de aplicação que vem sendo utilizado de maneira que supri a carência de instâncias na ontologia em desenvolvimento. A Figura 3.11 ilustra a visão geral desta fase.

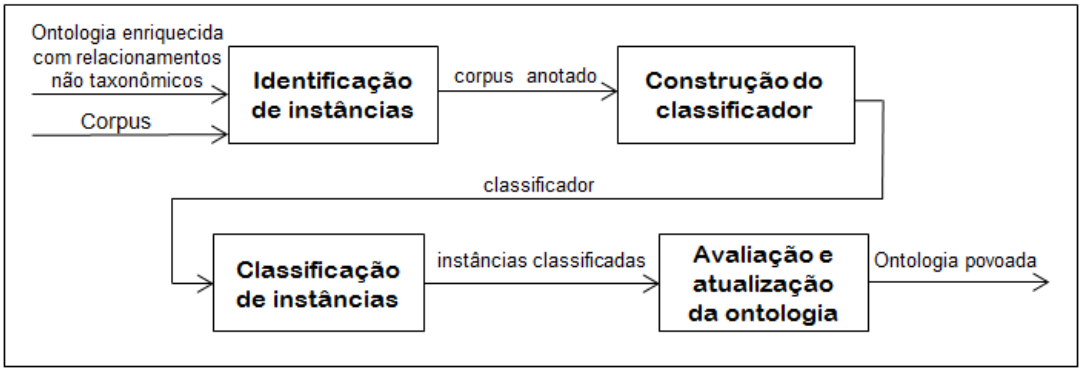


Figura 3.11: Visão geral da fase "Povoamento de ontologia"

Seu desenvolvimento obedece à execução de três atividades referentes à técnica DIPPAO proposta por [3]: "Identificação de Instâncias", "Construção do Classificador" e "Classificação de Instâncias", descritas a seguir.

3.5.1 Identificação de instâncias

Esta atividade consiste em identificar as instâncias de propriedades de relacionamentos não taxonômicos da ontologia. Para isso faz a anotação do corpus dado como entrada, através da análise: "morfo-lexical", "reconhecimento de entidade nomeadas" e "identificação de co-referências", detalhadas na Tabela 3.13.

Tabela 3.13: Análise morfo-lexical realizada na subatividade "Identificação o de instâncias"

Análise Morfo-lexical	
Tokenização	Divide o texto em tokens. Ex: Mary is married to John Mary / is / married / to / John
Normalização	Realiza a identificação de texto padrão. Ex: Data, hora, etc.
Divisão de Sentenças	Divide o texto em sentenças. Ex: Mary is married to John. / Kate is child of Mary. / John is married to Mary.
POS Tagging	Para cada token do documento é definida a categoria gramatical. Ex: Mary - substantivo Is - verbo
Reconhecimento de Entidades Nomeadas	
Identificação de Nomes	Possíveis instâncias, objetos exclusivos do mundo.
Identificação de co-referências	
Identificação de nomes ou pronomes que se referem à mesma unidade descrita previamente no texto.	

Ao final deste processamento é gerado um corpus anotado que é dado como entrada para a próxima atividade, a Figura 3.12 ilustra um trecho da anotação do corpus.

Keith (NNP) **R** (NNP) **.** (.) **married** (VBN) **with** (IN) **H** (NNP). (.) **R** (NN). (.) **They** (PRP) **married** (VBN) **in** (IN) **2004** (CD).**Keith** (NNP) **R** (NNP) **.** (.) **was** (VBD) **born** (VBN) **in** (IN) **1950** (CD). (.) **Keith** (NNP) **R** (NNP). (.) **is** (VBN) **father** (NN) **of** (IN) **Mariana** (NNP). (.) (.) **After** (IN) **Keith** (NNP) **R** (NNP). (.) **filed** (VBD) **for** (IN) **divorce** (NN) **in** (IN) **2006** (CD). (.) **H** (NNP). (.) **R** (NN). (.) **asserted** (VBD) **domestic** (JJ) **violence** (NN) **allegations** (NNS) **against** (IN) **Keith** (NNP) **R** (NNP). (.), (.) **and** (CC) **requested** (VBD) **sole** (JJ) **custody** (NN) **of** (IN) **Daughter** (NNP). (.) **Following** (VBG) **an** (DT) **investigation** (NN) **and** (CC) **a** (DT) **hearing** (NN), (.) **the** (DT) **court** (NN) **(** (()**Judge** (NNP) **Claudia** (NNP) **Silbar** (NNP)**)** ()) **denied** (VBD) **Mother** (NNP)'**s** (POS) **requests** (NNS). (.) **In** (IN) **February** (NNP) **2007** (CD), (.) **the** (DT) **court** (NN) **(** (()**Judge** (NNP) **Pollard** (NNP)**)** ()) **entered** (VBD) **an** (DT) **order** (NN) **granting** (VBG) **both** (DT) **parents** (NNS) **joint** (JJ) **legal** (JJ) **and** (CC) **physical** (JJ) **custody** (NN), (.) **and** (CC) **appointed** (VBD) **a** (DT) **child** (NN) **custody** (NN) **evaluator** (NN), (.) **who** (WP) **recommended** (VBD) **maintaining** (VBG) **the** (DT) **current** (JJ) **custody** (NN) **arrangements** (NNS) **based** (VBN) **on** (IN) **Daughter** (NNP)'**s** (POS) **parental** (JJ) **attachments** (NNS).

Figura 3.12: Trecho do corpus anotado no domínio do direito da família

3.5.2 Construção do classificador

A construção automática do classificador aplica técnica de "Extração de Informação" para especificar as regras de classificação. Esta atividade tem como entrada a ontologia e o corpus anotada na fase anterior e é realizada através de três subatividades: "Seleção de Classes, Propriedades e Relacionamentos", "Seleção de Triggers" e "Geração de Regras", descritas na próxima seção.

a. Seleção de Classes, Propriedades e Relacionamentos

Esta subatividade consiste em selecionar na ontologia o conjunto de classes (C_c), propriedades (P) e os relacionamentos não taxonômicos (R). A Tabela 3.14 ilustra a classe *"Person"*, as propriedades *"Name"* e *"Married_date"* e os relacionamentos não taxonômicos *"Wife_of"* e *"Husband_of"*.

Tabela 3.14: Exemplo de classe, propriedade e relacionamento não taxonômico identificados

Person		
Wife_of	*Instance*	*Person*
Husband_of	*Instance*	*Person*
Name		*String*
Married_date		*Date*

b. Seleção de *"Triggers"*

"Triggers" podem ser definidos como uma ou mais palavras que possuem relação entre si e que sugerem a presença de instâncias [25]. Por exemplo, a frase nominal *John* seguida pelo *"trigger" "divorced of"* sugere a presença da instância John.

Nesta subatividade, para cada sinônimo encontrado na base de dados léxica de relacionamento não taxonômico (R) ou de propriedade (P) um *"trigger"* é gerado. Por exemplo, *"Disjoin"* e *"Disassociate"* são sinônimos que correspondem aos *"triggers"* do relacionamento não taxonômico *"Divorced_of"*, da ontologia no Direito da Família.

O produto desta subatividade pode ser visualizado na Tabela 3.15 que ilustra os *"triggers"* relacionados à classe *"Person"* da ontologia do Direito da Família.

Tabela 3.15: Exemplo de triggers selecionadas

Property	**Trigger**	**Non Taxonomic Relationship**	**Trigger**
Birth_date	Born	Father	Male Parent
	Birth		Father
Divorce_date	Divorce	Divorce	Divorcement
	Disjoint		Dissociate
	Dissassociate		Divorced

c. Geração de Regras

Esta subatividade tem como objetivo especificar regras de classificação na forma: se <condição> então <conclusão>. A condição é composta por cinco predicados unidos por uma conjunção lógica, onde:

Uma frase nominal representa uma frase nominal X	John
Uma instância (I, X) representa a instância I a ser classificada na frase nominal X	John_instância da frase John
Um trigger (X, Y) representa um trigger Y da frase X	(John, divorced)
Um relacionamento sinônimo (Y, R), representa um sinônimo Y de um relacionamento não taxonômico R	(“divorced”, “divorced”)
Uma propriedade sinônimo (Y, P), representa um sinônimo Y de uma propriedade P	(“born”, “birth_date”)
Um relacionamento não taxonômico (R, C), representa um relacionamento não taxonômico R de uma classe C	(“divorced”, “Person”)
Uma propriedade (P, C, V), representa uma propriedade P de uma classe C com um valor V	Propriedade: Birth_date Classe: Person Valor: 1987

A conclusão é composta de dois predicados unidos por uma conjunção lógica, onde:

É_um (I,C) representa uma classe C onde a instância I pode ser classificada	é_um ("John_instancia", "Person")
Um relacionamento não taxonômico associação (I1, I2, R, C), representa instâncias I associadas por um relacionamento não taxonômico R de uma classe C	(“John_instância”, “A. S._instância”, “divorced”, “Person”)
Uma propriedade associação (I, P,V, C), representa uma instância I com um valor V para uma propriedade P de uma classe C	propriedade_associação (“John_instância”, “birth_year”, “1950”, “Person”)

Esta geração de regras só é possível graças às subatividades desenvolvidas anteriormente e que permtiu a identificação no corpus das frases nominais, instâncias, triggers e relacionamentos que implicam em uma conclusão. Uma exemplo de uma regra de classificação gerada através do relacionamento não-taxonômico *"married"* da classe *"Person"* pode ser visualizada na Figura 3.13.

```
noun_phrase ("John") ^ noun_phrase ("H. R.") ^
instance ("John_instance", "John") ^ instance ("H. R._instance", "H. R.") ^
trigger ("John", "married with") ^ trigger ("H. R.", "married with") ^
relationship_synonym ("married with", "married") ^
non_taxonomic_relationship ("married with", "Person") ⇒
is_a ("John_instance","Person") ^ is_a ("H. R._instance","Person") ^
non_taxonomic_relationship_association ("John_instance", "H. R._instance", "married with", "Person")
```

Figura 3.13: Exemplo de regra de classificação gerada através do relacionamento não taxonômico *"married"* referente à classe *Person*

3.5.3 Classificação de instâncias

Para a execução desta atividade tem-se como entrada o corpus anotado na atividade "Identificação de Instâncias" e o classificador gerado na atividade "Construção do Classificador". Esta atividade é composta por duas subatividades: "Associação de Instâncias" e "Instanciação". A associação de instâncias consiste em associar as instâncias encontradas às suas respectivas classes, propriedades e relacionamentos não taxonômicos. A subatividade "Instanciação"consiste em remover as instâncias duplicadas encontradas e inserí-las na ontologia de forma efetiva, resultando na ontologia finalmente povoada. A Figura 3.14 mostra um exemplo da taxonomia de uma ontologia após o povoamento e na Figura 3.15 é mostrado o exemplo de uma instância identificada.

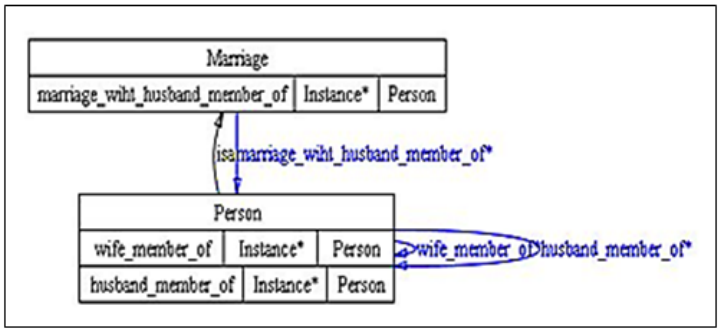


Figura 3.14: Taxonomia da ontologia gerada pela fase de povoamento

Keith R._Instance	
married	H. R._Instance
father	Mariana_Instance
birth_year	1950
divorce	2006
constitutive	2004

Figura 3.15: Exemplo de instância identificada na ontologia desenvolvida

3.6 Inserção de axiomas

O objetivo desta fase é inserir novos axiomas na ontologia em desenvolvimento de maneira que permite a integração das regras de LPO desenvolvidas na fase "Construção da Ontologia Base" com as regras de LPO geradas nesta fase. Para isso, tem como entrada a ontologia povoada anteriormente e como saída uma ontologia com novos axiomas. Esta fase obedece a execução três atividades referentes à técnica TAILP proposta por [9]: "Extração de Instâncias da Ontologia", "Geração da Hipótese Induzida" e "Inserção de Nova Regra na Ontologia", detalhadas nas próximas subseções. A Figura 3.16 ilustra a visão geral da fase.

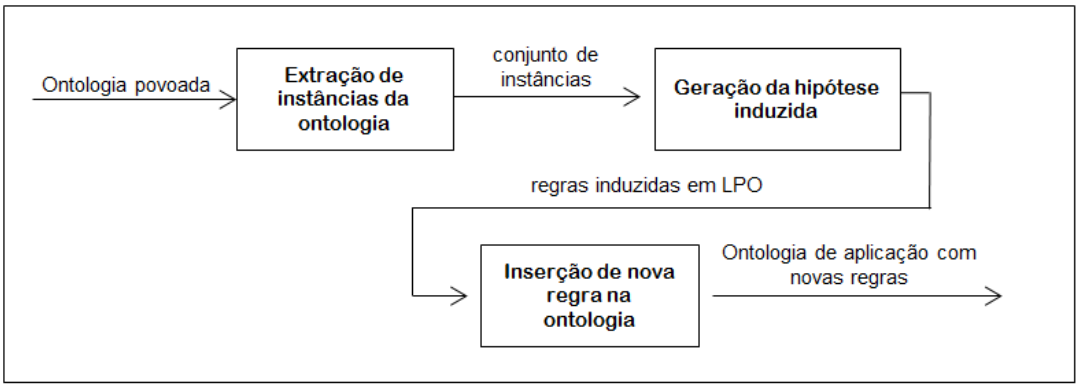


Figura 3.16: Visão geral da fase "Inserção de axiomas"

3.6.1 Extração de instâncias da ontologia

Dada a ontologia povoada na fase anterior são realizadas inferências indutivas para extrair um conjunto de instâncias I = $rel_k(c_1, c_2, ..., c_n) \in I$ de relações não taxonômicas. Este processo de indução retorna regras em LPO no formato de cláusulas de Horn invertidas no formato "a $\leftarrow$ b" (se b então a)", com o intuito de escolher a relação que virá ser a regra induzida. O conjunto de instâncias extraídas

tornam-se fatos da base de conhecimento e será utilizado pelo PLI. Por exemplo, *"mother_of(woman, person)"* representa uma relação relevante no contexto do domínio do Direito da Família que possui as instâncias *"mother_of(Anne Smith, Stuard Smith)"*.

3.6.2 Geração da hipótese induzida

Nesta atividade é construída uma base de conhecimento a partir das instâncias extraídas na atividade anterior. Esta base é utilizada no padrão de uma ferramenta PLI (Progol [1]) que realiza inferências indutivas através de algoritmos de PLI, retornando a um conjunto de cláusulas em LPO que são consideradas como hipóteses induzidas. No exemplo abaixo é ilustrado uma possível regra que pode ser extraídas de uma ontologia pertencente ao domínio do Direito da Família.

$$Grandmother\_of\ (PersonX, PersonY) \rightarrow$$

$$Mother\_of\ (PersonX, PersonZ)\ \hat{}\ Mother\_of\ (PersonZ, PersonY)$$

3.6.3 Inserção de nova regra na ontologia

Esta subatividade consiste em avaliar o conjunto de hipóteses geradas na atividade "Geração da Hipótese Induzida" a fim de que se tornem os axiomas da ontologia. Esta avaliação pode ser realizada por meios estatísticos que visam confirmar a validade da hipótese, ou através de um *"feedback"* do usuário. Para que a inserção seja realizada há a conversão da linguagem utilizada pelo sistema PLI (regras em LPO) para a linguagem de representação de ontologias, a OWL. O produto é um conjunto de axiomas que são inseridos na ontologia de maneira explícita. A Tabela 3.16 mostra um exemplo dos axiomas descobertos na ontologia.

Ao final do processamento é gerada uma ontologia de aplicação que contempla todos os elementos da definição formal de ontologias, atendendo ao principal objetivo deste trabalho.

[1]O Progol é um tipo de sistema PLI, no qual o usuário declara explicitamente o formato da cabeça e do corpo da hipótese, estas declarações são chamadas de declarações "Modes", que servem como "guia"para a construção de um grafo de refinamento

Tabela 3.16: Axiomas em SWRL gerado pela fase de "Inserção de axiomas"

Axiomas gerados pela TAILP
mother_of(?A, ?C) $\wedge$ mother_of(?C, ?B) $\rightarrow$ grandmother_of(?A, ?B)
daughter_of(?B, ?A) $\rightarrow$ mother_of(?A, ?B)
daughter_of(?B, ?A) $\rightarrow$ father_of(?A, ?B)

O processo ainda permite ao desenvolvedor, a execução de todas ou apenas uma de suas fases o que permite a adição de novos elementos à ontologia desenvolvida.

3.7 Apponto-ProTool - uma ferramenta de suporte ao processo Apponto-Pro

Para efeitos da avaliação do processo foi desenvolvida uma ferramenta de software intitulada Apponto-ProTool (*Application Ontology Process Tool*), que fornece suporte semiautomático para a construção de ontologias de aplicação e que integra três ferramentas (GAODTool [5], DIPPAOTool [3] e TAILP [9]) correspondentes às abordagens contidas no processo proposto.

A GAODTool foi desenvolvida na linguagem PHP [16] enquanto a DIPPAOTool e a TAILP foram desenvolvidas na linguagem Java [26], sendo que o PHP é o ambiente de desenvolvimento predominante da Apponto-ProTool. Foi utilizado também o XHTML [27], o CSS [28] e o JavaScript [29] para o desenvolvimento da interface. A Appont-ProTool trabalha conjuntamente ao software Protégé, pois este é responsável por executar o arquivo OWL que possui a ontologia desenvolvida. A integração realizada permitiu a construção de uma ontologia de aplicação completa, ao mesmo tempo em que satisfaz o principal objetivo do sistema. Essa característica não foi observada em nenhuma das técnicas descritas na fundamentação teórica e no estado da arte do presente trabalho.

A Apponto-ProTool é composta por quatro fases "Construção e Extensão de Ontologia Base", "Povoamento de Ontologia", "Inserção de Axiomas" e "Incrementar Ontologia". Inicialmente na fase "Construção e Extensão de Ontologia Base" é gerada

uma ontologia de aplicação com classes, hierarquias, relacionamentos, propriedades e axiomas. Esta ontologia é dada como entrada para a segunda fase "Povoamento de Ontologia" para que as classes da ontologia sejam instanciadas através do processamento de um corpus, o produto desta fase é uma ontologia de aplicação povoada. Por fim, a fase "Inserção de axiomas" recebe a ontologia povoada e tem o objetivo de inferir axiomas gerando como produto final uma ontologia completa.

A ferramenta inicia seu processo a partir de um conjunto de objetivos e fatos referentes a um determinado domínio de aplicação, em suas fases posteriores utiliza a ontologia desenvolvida e fontes de informações textuais de domínios idênticos. A formalização do processo é independente do domínio de aplicação, mas para que esta autonomia de domínio seja estabelecida, é necessária a participação do especialista do domínio e do engenheiro do conhecimento para elaborar um conjunto de objetivos e fatos semelhante ao domínio do corpus utilizado, isso permite que a construção da ontologia seja realizada de forma adequada.

A Figura 3.17 apresenta a tela inicial da Apponto-ProTool onde o usuário clica no botão "ENTRAR" para que seja dado início a construção da ontologia.

Figura 3.17: Tela inicial da ferramenta Apponto-ProTool

3.7.1 Construção e extensão da ontologia base

Inicialmente, o engenheiro do conhecimento e o especialista do domínio elaboram um conjunto de objetivos e fatos no domínio de interesse para que sejam fornecidos por meio de um usuário à primeira atividade da fase "Inserção de Objetivos

e Fatos". A Figura 3.18 ilustra a inserção do objetivo *"List the members of a family"*. A lista completa dos objetivos e fatos encontra-se no Anexo A.

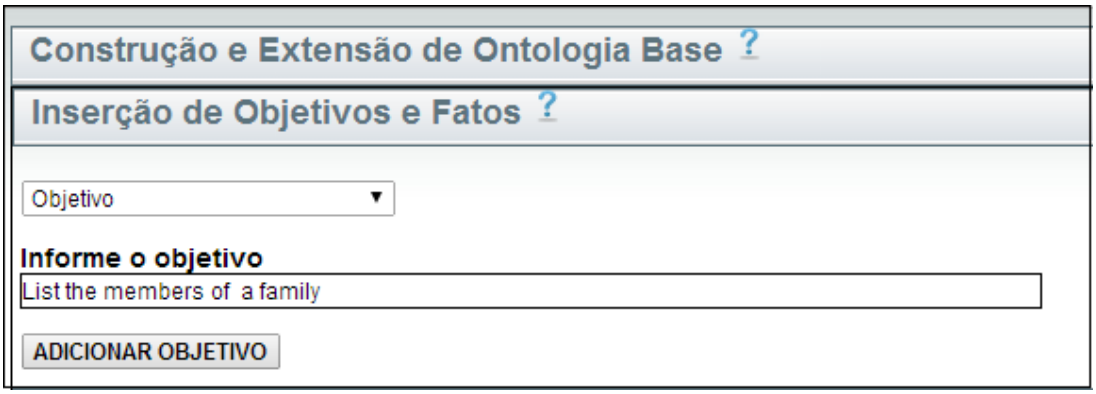


Figura 3.18: Inserção de objetivo no domínio do Direito da Família

Os objetivos e fatos são inseridos e armazenados automaticamente em uma lista na área "Seleção dos objetivos e fatos". A Figura 3.19 ilustra uma visão parcial da lista de objetivos e fatos desenvolvida.

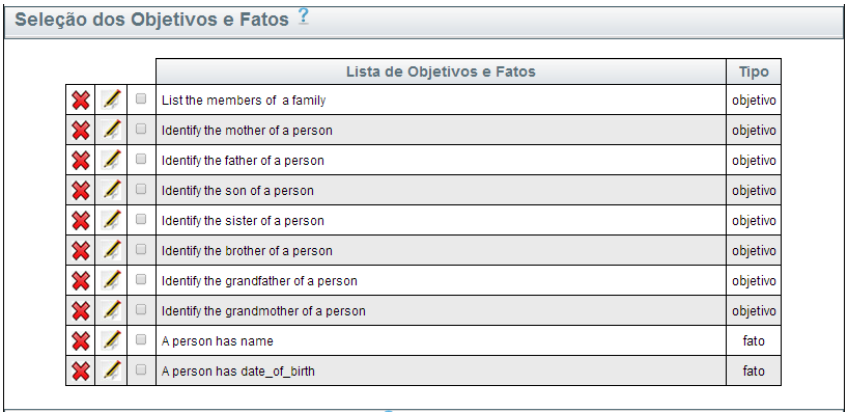


Figura 3.19: Lista de objetivos e fatos submetidos à Apponto-ProTool

O usuário pode incluir, excluir, editar e selecionar os objetivos ou fatos que deseja trabalhar através dos ícones localizados à esqueda da lista gerada. A partir desta lista o usuário pode selecionar quais serão dados como entrada para a atividade "Representação dos predicados em LPO", esta por sua vez, tem o objetivo de realizar a tradução de linguagem natural para predicados em LPO.

No contexto do Direito da Família foi definido como objetivo principal *"Identify the members of the family"* e como objetivos específicos *"Identify the father of a person"* e *"Identify the mother of a person"*. A Figura 3.20 ilustra a transformação realizada por esta atividade.

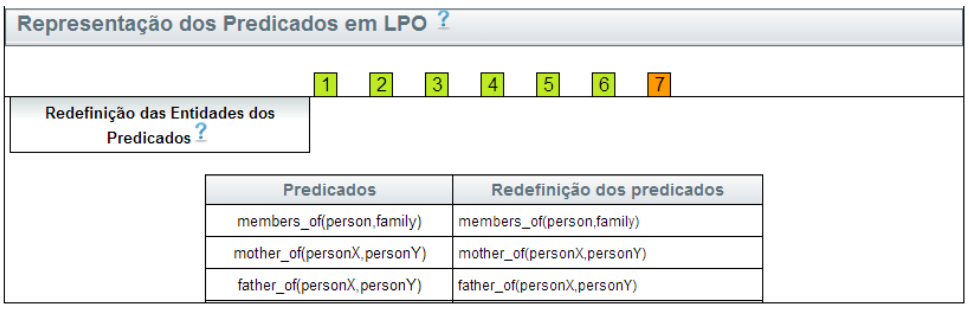


Figura 3.20: Predicados construídos em LPO

Uma vez que os objetivos e fatos são traduzidos de linguagem natural para predicados em LPO, cabe ao usuário oferecer para a atividade "Especificação dos Axiomas em LPO" os predicados que foram construídos. Neste momento é permitido ao desenvolvedor especificar as regras que permitem alcançar o principal objetivo que o sistema deseja alcançar através dos predicados que compõem a condição e a conclusão do axioma.

A Figura 3.21 apresenta a conclusão definida pelo predicado *"members_of (person,family)"* e a condição pelos predicados *"father_of (personX,personY)"*, *"mother_of (personX,personY)"*.

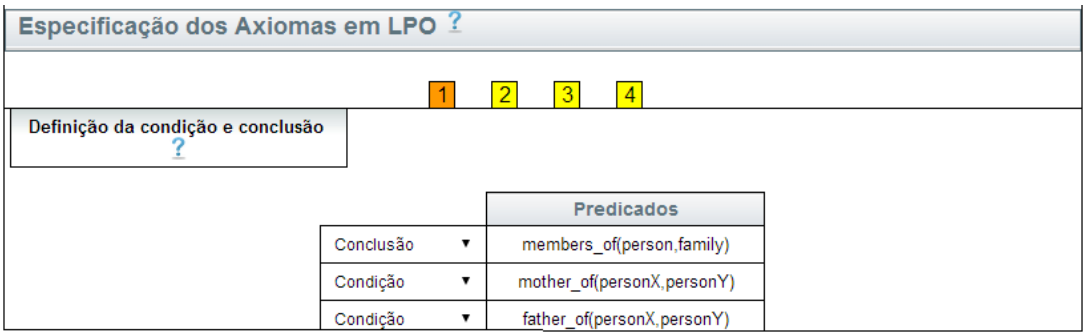


Figura 3.21: Predicados que compõem a condição/conclusão do axioma desenvolvido

Em seguida, cabe ao usuário definir os operadores booleanos que existem entre os predicados gerados, se conjunção $(\wedge)$ para os casos em que um predicado necessitar de outro para satisfazer o objetivo ou, a disjunção $(\vee)$ para os casos em que não ocorrer. Neste caso, o operador de conjunção é utilizado para unir os predicados *"members_of (person,family)"*, *"mother_of (personX,personY)"*, *"father_of (personX,personY)"*. A Figura 3.22 ilustra a definição da conjunção estabelecida entre os predicados.

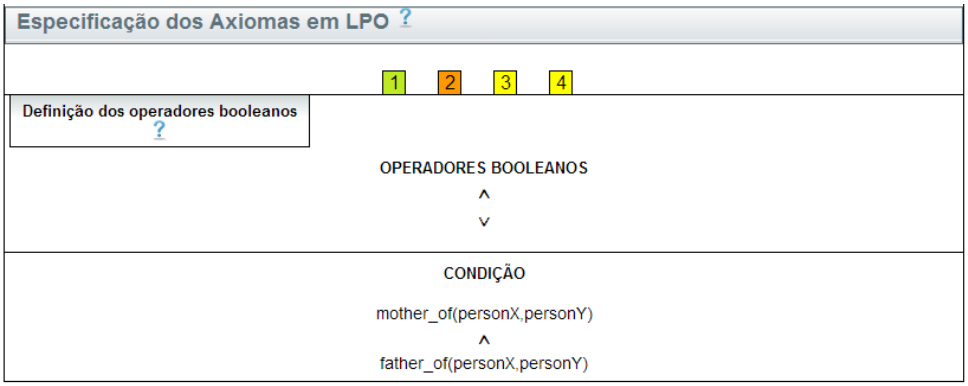


Figura 3.22: Predicados em LPO unidos pelo operador de conjunção

Logo após, é realizada a definição dos quantificadores, que pode ser existencial (∃) ou universal (∀), estes associam as entidades presentes no axioma. Para as entidades *"family"*, *"person"*, *"personX" e "personY"* é definido o quantificador existencial, conforme ilustrado na Figura 3.23.

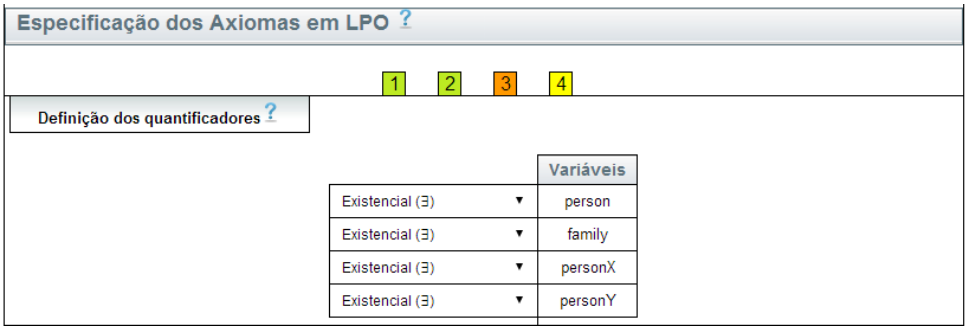


Figura 3.23: Quantificador existencial definido para cada um dos predicados

Ao término da atividade "Especificação dos axiomas em LPO", o usuário determina se o axioma a ser gerado é uma implicação ou uma equivalência. A implicação é utilizada quando a condição é satisfatória e leva a conclusão. A equivalência é aplicada nos casos em que a conclusão e a condição são proporcionais. A Figura 3.24 ilustra a implicação definida pelo usuário.

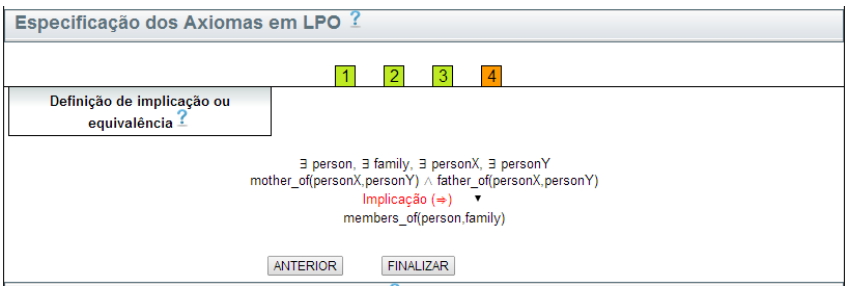


Figura 3.24: Definição da implicação do axioma desenvolvido

Após o usuário ter definido a "conclusão/condição", os "operadores booleanos", os "quantificadores" e a "implicação ou equivalência" o axioma "$\exists$ *person*, $\exists$ *family*, $\exists$ *personX*, $\exists$ *personY* | *mother_of(personX,personY)* $\wedge$ *father_of(personX,personY)* $\rightarrow$ *members_of(person,family)*" é criado e armazenado na área "Axiomas desenvolvidos", conforme ilustra a Figura 3.25. A lista completa dos axiomas encontra-se no Anexo B.

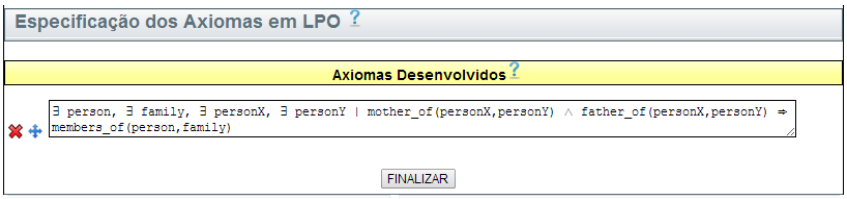


Figura 3.25: Axioma desenvolvido ao término da especificação dos axiomas em LPO

Como descrito anteriormente, o processo é incremental, sendo assim, novos axiomas podem ser inseridos, ou seja, um novo ciclo desta fase se inicia para que sejam inseridos novos elementos na ontologia.

A atividade "Representação da ontologia de aplicação" visa traduzir os axiomas desenvolvidas em uma linguagem de marcação, neste caso foi utilizado a RuleML [30] por causa de sua expressividade, ou seja, possui recursos suficientes para representar regras em LPO. Uma visão parcial dos axiomas traduzidos é apresentada na Figura 3.26. A lista completa dos axiomas encontra-se no Anexo C.

```
<Assert>
   <Rulebase mapClosure="universal">
      <Implies>
         <And>
            <Atom>
               <Rel>mother_of</Rel>
               <Var>PersonX</Var>
               <Var>PersonY</Var>
            </Atom>
            <Atom>
               <Rel>father_of</Rel>
               <Var>PersonX</Var>
               <Var>PersonY</Var>
            </Atom>
         </And>
         <Atom>
            <Rel>members_of</Rel>
            <Var>Person</Var>
            <Var>Family</Var>
         </Atom>
      </Implies>
   </Rulebase>
</Assert>
```

Figura 3.26: Visão parcial dos axiomas representados em regras RuleML

Após a tradução dos axiomas, é permitido ao usuário a definição das classes que irão compor a ontologia. A Figura 3.27 ilustra as classes especificadas.

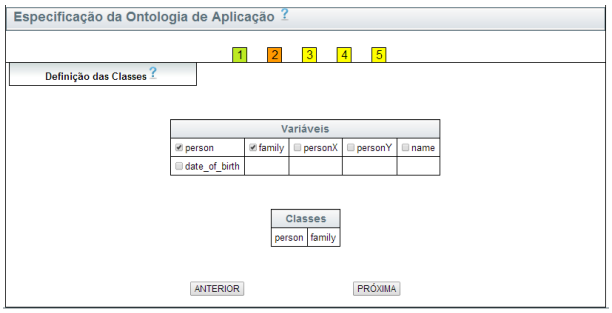


Figura 3.27: Definição das classes da ontologia

Uma vez que as classes foram identificadas, é permitido ao usuário realizar a identificação dos relacionamentos não taxonômicos. Por exemplo, a relação não taxonômica *"members_of"* define uma relação entre as classes *"person"* e *"family"* A Figura 3.28 apresenta um exemplo da definição dos relacionamentos não taxonômicos.

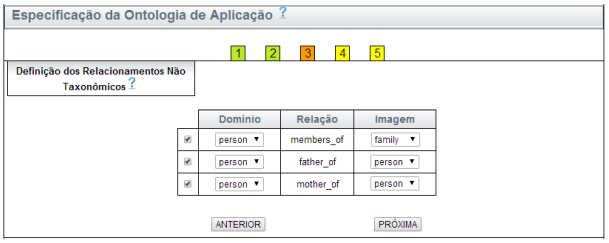


Figura 3.28: Visão parcial dos relacionamentos não taxonômicos definidos na fase "Construção da ontologia base"

Logo após, é realizada a definição dos relacionamentos taxonômicos, que são baseados nas relações hierárquicas entre as classes definidas anteriormente. Por exemplo, há uma relação hierárquica entre as classes *"family"* e *"person"*, conforme ilustra a Figura 3.29.

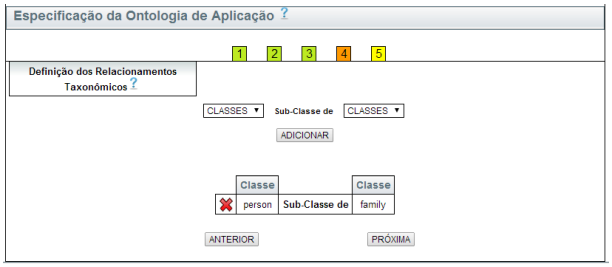


Figura 3.29: Visão parcial dos relacionamentos taxonômicos definidos entre as classes da ontologia em desenvolvimento

Por fim, são definidas as propriedades que compõem a ontologia, estas propriedades podem ser do tipo *string*, *int*, *float*, *booleano ou date*. A Figura 3.30 apresenta as propriedades *"name"* e *"date_of_birth"*, definidas pelo usuário.

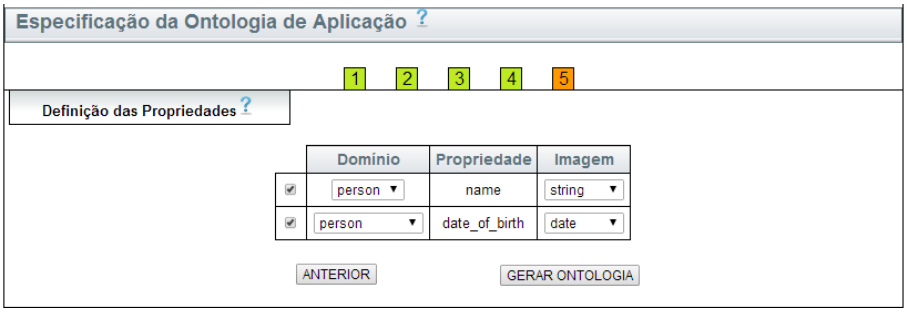


Figura 3.30: Visão parcial da definição das propriedades da ontologia

Uma vez definidas as classes, as relações taxonômicas, os axiomas, as relações não taxonômicas e as propriedades, cabe ao usuário gerar a ontologia desenvolvida que é representada em um arquivo OWL. O usuário pode realizar o download da ontologia, conforme ilustrado na Figura 3.31 e visualizá-la na interface do software Protégé.

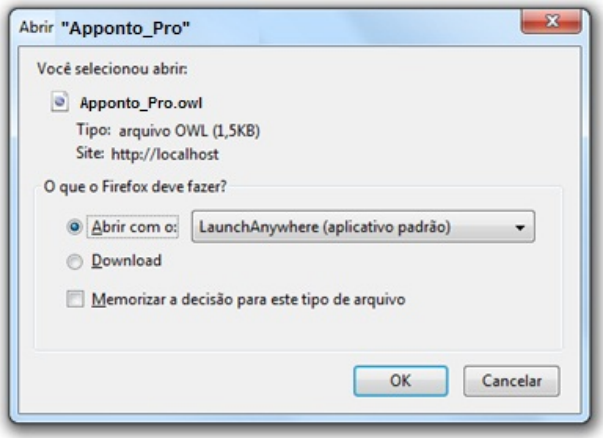


Figura 3.31: Arquivo contendo a ontologia de aplicação gerada na fase "Construção de ontologia base"

A Figura 3.32 apresenta a ontologia desenvolvida através da interface do software Potégé, onde ilustra as classes hierarquizadas, as relações não taxonômicas, as propriedades e as instâncias, no entanto os axiomas não podem ser visualizados devido estes estarem implícitos na ontologia e o Protégé não possuir um plugin para sua visualização.

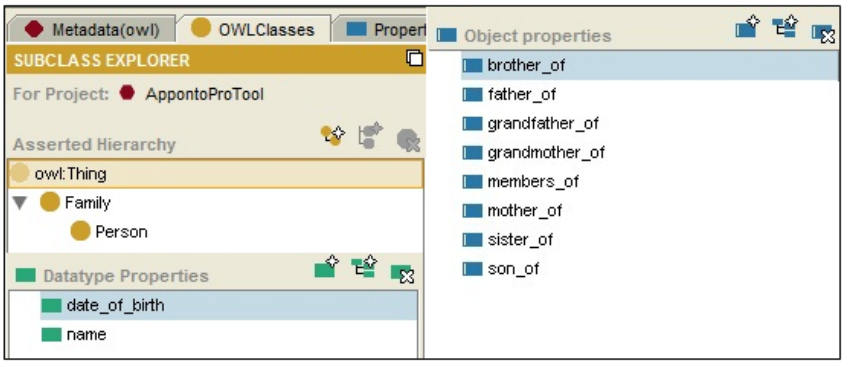


Figura 3.32: Classes, hierarquias, propriedades, relações e instâncias geradas na primeira fase da ferramenta Apponto-ProTool

Uma vez que a ontologia desenvolvida não possui instâncias, esta é dada como entrada para a próxima fase "Povoamento de Ontologia" a fim de que as classes sejam instanciadas.

3.7.2 Povoamento da ontologia

Esta fase contém dois campos de entrada, um para a seleção do corpus, que é selecionado a partir de um diretório onde estão armazenados os documentos no domínio do Direito da Família e outro para a seleção da OWL. A Figura 3.33 apresenta um trecho do corpus a ser submetido nesta fase. Na Figura 3.34 é ilustrada a visão geral da fase "Povoamento da Ontologia".

McKINSTER, J. - A father of minor children appeals from an order modifying his child-support obligation. We affirm.

FACTUAL AND PROCEDURAL BACKGROUND

Dennis Scheppers and Nancy Scheppers married in 1974. The marriage produced six children: Micah, Matthew, Amber, Joseph, Amanda, and Jennifer. When their marriage was dissolved in 1987, all six children were minors.
In July of 1998, the mother applied for and obtained an order to show cause seeking, inter alia, a modification of child support for the two minor children living with her. Following an evidentiary hearing, the trial court set the father's child support obligation at $2,991 per month. The father appeals.

CONTENTIONS

In a somewhat different order, the father contends that the trial court erred in four respects: by failing to include in the mother's gross income sums she received as the beneficiary of a life insurance policy; by failing to impute any income to the mother based upon her ability to earn; by basing the father's income upon an unreasonable work schedule; and by depriving the father of due process and equal protection of the law.

Figura 3.33: Trecho do corpus *Family_Law* a ser submetido à fase de povoamento

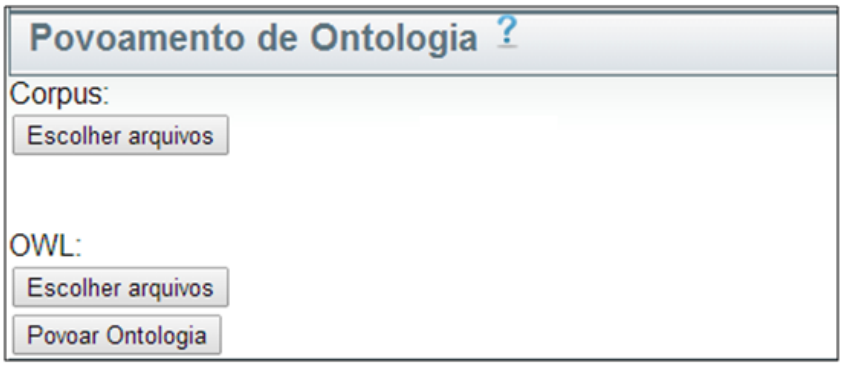


Figura 3.34: Visão geral da fase de "Povoamento da Ontologia"

O corpus e a ontologia OWL dados como entrada são processados automaticamente a fim de que sejam extraídas as instâncias a partir das fontes textuais submetidas, classificando-as como instâncias de classes da ontologia. Ao final do processamento desta fase é retornada a ontologia povoada onde é permitido ao usuário realizar o *"download"*. A Figura 3.35 ilustra um trecho da ontologia gerada, onde pode ser observado a classe *"Person"* com 105 instâncias encontradas onde uma delas chama-se "MELISSA R." que possui a data de nascimento "19.12.1988".

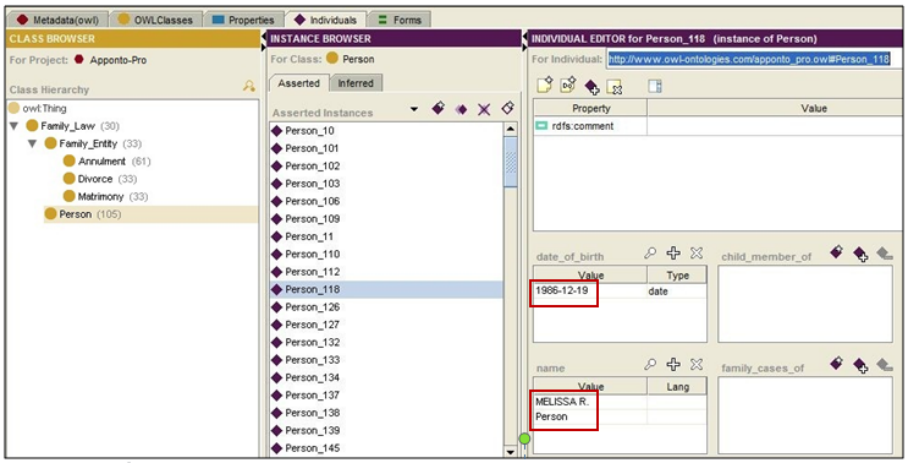


Figura 3.35: Ontologia povoada na Apponto-ProTool

A quantidade de instâncias encontradas em cada classe dependerá da quantidade de corpus inserido na fase. Esta ontologia é submetida à terceira fase da Apponto-ProTool, descrita na seção a seguir.

3.7.3 Inserção de axiomas

A interface usuário desta fase possui três campos: “Início”, “Relações” e “Resultados”. O campo “Início” permite ao usuário selecionar o arquivo OWL da ontologia em que se deseja inserir axiomas, que neste caso é a ontologia gerada na fase “Povoamento da Ontologia”, descrita anteriormente. Ao ser submetida a ontologia, compete ao usuário carrega-la através do botão “EXECUTAR” para que seja iniciado o processamento automático da ontologia extraindo todas as relações não taxonômicas encontradas. A Figura 3.36 ilustra a visão geral da interface usuário para esta fase.

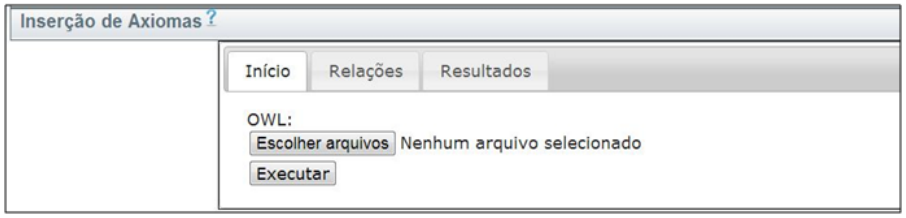


Figura 3.36: Visão geral da interface usuário da fase "Inserção de Axiomas"

Após o carregamento da ontologia, relações não taxonômicas são extraídas e retornadas ao usuário em forma de lista no campo “Relações”. É possível selecionar qual a relação se deseja trabalhar através da caixa de seleção localizado ao lado das relações. Ao clicar no botão “ENVIAR” a ferramenta inicia a extração de todas as instancias da relação selecionada. A Figura 3.37 ilustra possíveis relações identificadas nesta fase.

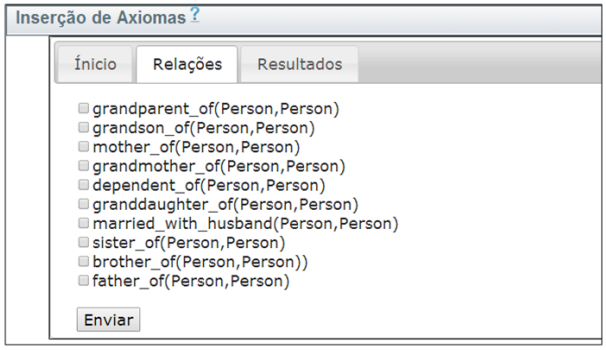


Figura 3.37: Trecho de relações não taxonômicas identificadas na ontologia

Por fim, as hipóteses inferidas anteriormente são armazenadas no campo “Resultados” a fim de que sejam transformadas em axiomas a ser incluídos na ontologia através do botão “GERAR”. A Figura 3.38 ilustra as hipóteses listadas no campo de resultados.

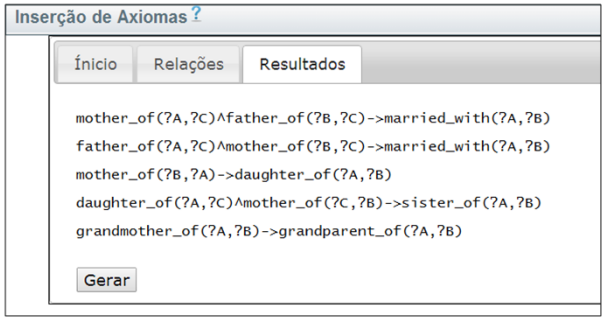


Figura 3.38: Relações selecionadas a serem transformadas em axiomas da ontologia

Os axiomas gerados podem ser visualizados através do software Protégé por meio do plugin SWRL, assim como ilustrado na Figura 3.39. A lista completa dos axiomas encontra-se no Anexo D.

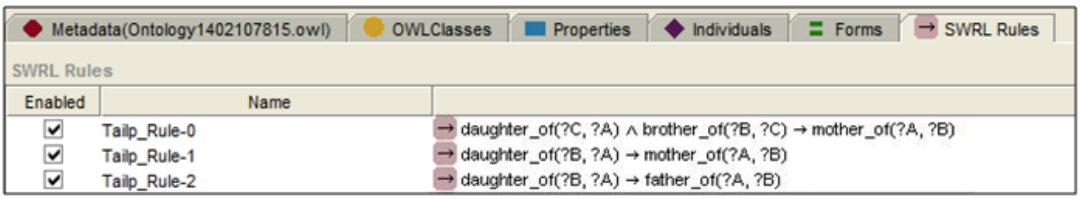


Figura 3.39: Axiomas em SWRL desenvolvidos na fase de "Inserção de Axiomas"

O resultado final do processamento é uma ontologia de aplicação com todos os elementos componentes. Esta ontologia pode ainda ser submetida à fase "Incrementar Ontologia" para que mais elementos sejam adicionados com o objetivo de extender a ontologia desenvolvida.

Este incremento pode ser realizado a partir de qualquer uma das fases da Apponto-ProTool, cabe ao desenvolvedor a decisão de incrementar ou não a ontologia desenvolvida. Um exemplo deste incremento pode ser visualizado na Figura 3.40 que apresenta as classes *"Family"* e *"Person"* geradas durante a primeira execução da ferramenta e as classes *"Family_Law"*, *"Family_entity"*, *"Civil_union"*, *"Divorce"*, *e "Matrimony"* referentes a segunda interação da ferramenta, resultando em uma ontologia incrementada.

O desenvolvimento deste processo Apponto-Pro foi realizado através de um somatório de técnicas adaptadas e integradas, uma metodologia mais um ciclo de vida que deu origem ao processo Apponto-Pro e à ferramenta case Apponto-ProTool que semiautomatiza o processo. Este ciclo de vida pode ser visualizado na Figura 3.41.

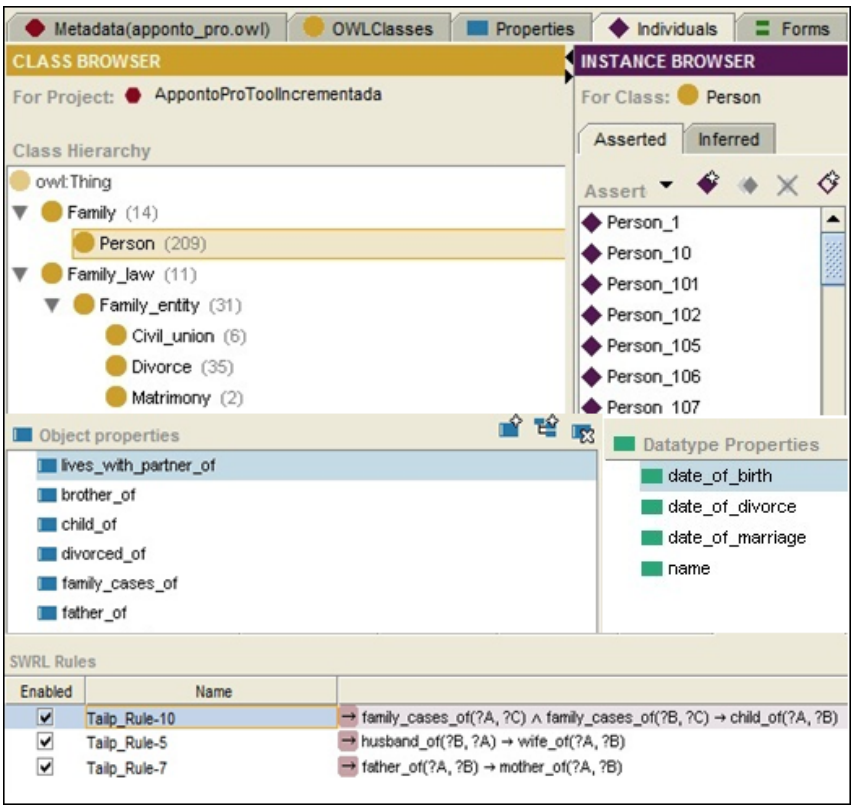


Figura 3.40: Trecho da antologia desenvolvida

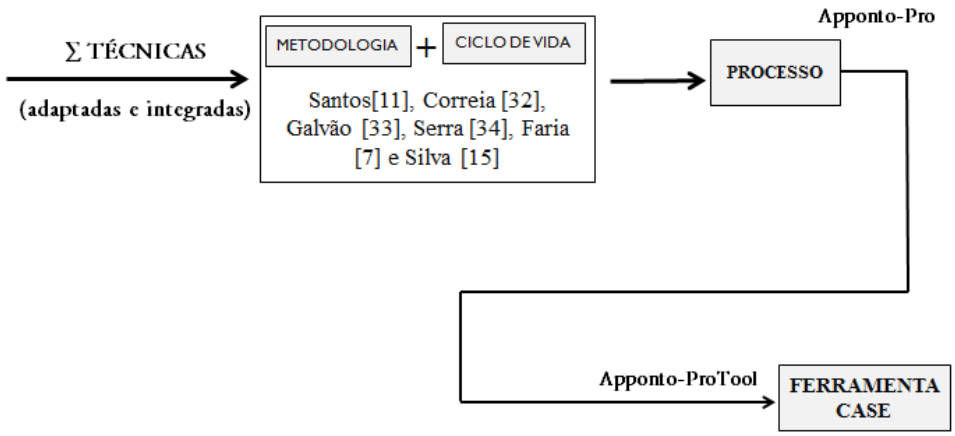


Figura 3.41: Ciclo de vida do processo Apponto-Pro

Primeiramente, foi realizado um somatório de técnicas com o objetivo de adquirir conhecimento sobre as abordagens que compõem o processo proposto. Este somatório incluiu todas as técnicas a ser adaptadas e integradas resultando numa metodologia mais um ciclo de vida que permitiu identificar as vantagens e desvantagens em cada uma das técnicas, bem como estabelecer a definição da ordem de descoberta dos elementos da ontologia para que fosse realizada a adaptação e integração adequada. Por exemplo, a GAODT [5] constrói uma ontologia de aplicação

que gera apenas cinco elementos da definição formal de ontologias (elementos Cc, H, R, P e A), obtendo em seu resultado final uma ontologia com instâncias ausentes. A técnica para a aquisição de relações taxonômicas de uma ontologia [6] e o método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos (AFC) [7] visam apenas a aprendizagem de classes e relações taxonômicas (elementos Cc, H) aprende classes e relações taxonômicas da ontologia. TARNT [8] extrai os relacionamentos não taxonômicos da ontologia (elemento R). A DIPPAO [3], gera as instâncias (elemento I) da ontologia. E, a TAILP [9], gera apenas os axiomas (elemento A) da ontologia.

A partir destas informações a ordem de integração definida anteriormente foi concretizada a partir das necessidades identificadas em cada uma das abordagens citadas. Além disso, houve a necessidade de realizar a adaptação da ontologia para que cada uma das técnicas pudessem ser integradas, estas alterações são discutidas na seção 3.8. Uma vez definida a ordem de descoberta foi desenvolvido o processo Apponto-Pro que integrou todas as abordagens analisadas no somatório de técnicas quem incluiu a metodologia e o ciclo de vida. Para provê suporte ao processo foi desenvolvida a ferramenta de software Apponto-ProTool que permitiu a construção de uma ontologia de aplicação que contempla todos os elementos da ontologia de forma conjunta atendendo ao principal objetivo deste trabalho.

3.8 Discussão sobre a Apponto-ProTool

Para o desenvolvimento da Apponto-ProTool foi definida uma ordem de integração de ferramentas onde esta, obedeceu às necessidades que foram encontradas em cada uma das abordagens utilizadas, ou seja, a integração corresponde à ordem de descoberta dos elementos da ontologia de acordo com cada ferramenta.

Estas necessidades foram identificadas a partir de uma análise sobre cada um dos produtos finais gerados pelas ferramentas, onde pôde ser constatada a ausência de elementos da definição formal em cada um dos resultados obtidos. A primeira ferramenta analisada (GAODTool [5]) trabalha na construção manual de uma ontologia de aplicação que gera classes, hierarquias, relacionamentos, propriedades e axiomas a partir de um conjunto de objetivos e fatos, porém não trabalha com a

geração de instâncias para que tenha como produto final uma ontologia completa. À vista disso, precisa do suporte de uma ferramenta relacionada ao povoamento de ontologias para que esta carência de instâncias seja suprida, logo, foi integrada a ela a ferramenta DIPPAOTool [3].

Neste ponto da integração foi constatada a solução de duas questões, uma é possibilidade de inserção de instancias em uma ontologia gerada a partir de uma ferramenta já existente e a outra é a obtenção de uma ontologia completa com todos os elementos da definição formal de ontologias. Buscando extender a ontologia desenvolvida foi integrada a Apponto-ProTool a ferramenta TAILP [9] para que novos axiomas fossem adicionados, isto permitiu a integração dos axiomas em LPO gerados pela GAODTool [5] com os axiomas gerados pela TAILP [9] gerando como produto final uma ontologia de aplicação extendida com novas regras.

No entanto, trabalhar com a integração de ferramentas já existentes não é uma tarefa simples, pois essa exige estudo e conhecimento profundo sobre cada abordagem, uma vez que mesmo utilizando a mesma linguagem de representação de ontologias (OWL), podem surgir problemas durante seu reúso, como a forma de captura e formalização do conhecimento a ser trabalhado, o reúso de conceitos previamente estabelecidos, a avaliação da implementação e até a limitação do seu desenvolvimento, devido às ferramentas utilizadas para a integração, nem sempre basearem-se em uma mesma linguagem de programação, dificultando a obtenção de resultados satisfatórios.

Na desenvolvimento da Apponto-ProTool, não foi diferente, todas as problemáticas aqui citadas foram encontradas. Para resolvê-las, foram analisadas as saídas obtidas de modo isolado em cada uma das ferramentas que compõem a Apponto-ProTool, tendo estas sofrido alterações para que pudesse haver êxito em seus resultados e o produto final obtivesse sucesso, as alterações realizadas incluíram a combinação de técnicas, a adaptação da ontologia e a integração em si.

Inicialmente, para cada ferramenta, foi analisada a forma que os dados são descritos e representados na ontologia e logo após foram analisados para verificar sua compatibilidade. Na GAODTool [5] a entrada dos dados é realizada manualmente na forma de um conjunto de objetivos e fatos, a partir desses objetivos são definidos as classes, hierarquias, relações, propriedades e axiomas. No entanto, após a análise

do processamento destas entradas, foram constatadas que a forma de descrição das relações não taxonômicas e a representação das propriedades são traçadas de maneira discorde a ontologia gerada pela DIPPAOTool [3] e, estes precisam estar descritos de forma semelhante para que possam ser integrados. Apesar da DIPPAOTool [3] não definir propriedades nem relações taxonômicas esta, realiza a identificação de instâncias a partir da aplicação de técnicas puramente linguísticas que permitem para identificar as instâncias através dos relacionamentos não-taxonômicos e propriedades de uma ontologia.

A ferramenta GAODTool [5], especifica as relações não taxonômicas levando em consideração os verbos presentes nas frases, a fim de que estes representem os relacionamentos a ser extraídos e as propriedades sejam criadas. Por exemplo, para o objetivo *"Identify the husband of a person"*, a relação extraída é *"Identify"* devido este ser o verbo encontrado na frase, porém este verbo precisa de um complemento para ter sentido, para isso é realizada uma redefinição do relacionamento, gerando assim a relação *"IdentifyHusband"*. Conforme descrito na Tabela 3.17 e visualizado na Figura 3.42 na interface do Protégé.

Tabela 3.17: Relação não taxonômica gerada pela GAODTool

Relação	Entidades	Relacionamento
Identify	*Husband, Person*	*IdentifyHusband*

Figura 3.42: Propriedades da GAODT visualizadas na interface do Portégé

Entretanto, as relações descritas desta forma impossibilitam que a ferramenta DIPPAOTool [3] realize a instanciação da ontologia pelas razões expostas a seguir. Como já citado, a DIPPAOTool [3] extrai as instâncias a partir de um corpus dado como entrada, realizando a associação das instâncias de propriedades e de relacionamentos não taxonômicos às suas respectivas classes, ou seja, associa

os nomes ou pronomes que se referem a uma mesma entidade descrita previamente no texto para que possa realizar a instânciação. Para isso, necessita gerar regras de extração e classificação de instâncias, estas regras são baseadas a partir de padrões léxico sintáticos e no conhecimento do domínio representado através da ontologia de domínio. Esta regra considera as relações hierárquicas na forma "tipo_de", por exemplo, dados os relacionamentos esposa *"wife_of"*; e marido *"husband_of"* a seguinte regra de extração e classificação é criada:

SE **NomePróprio1** e **NomePróprio2** foram casados ENTÃO

classificar **NomePróprio1** como instância do relacionamento

marido - *"husband_of"* e o **NomePróprio2** como instância

do relacionamento esposa - *"wife_of"*

Esta forma de declaração de relacionamentos não taxonômicos facilita a busca por instâncias a partir do processamento do corpus dado como entrada. Por exemplo, ao se inserir o objetivo *"Identify the husband of a person"*, a relação hierárquica extraída é *"husband_of"*. A Figura 3.43 ilustra alguns exemplos das relações não taxonômicas que a DIPPAOTool utiliza para identificar instâncias no corpus.

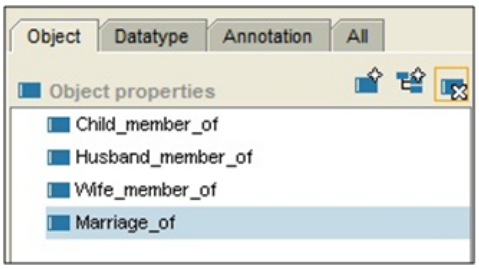


Figura 3.43: Propriedades geradas pela DippaoTool

Entretanto, outras questões ainda precisaram ser solucionadas, a primeira delas diz respeito à estrutura do arquivo OWL gerado por cada uma das ferramentas.

Para que exista a instanciação de classes na ferramenta DIPPAOTool [3] e estas possam ser representadas na OWL, é necessário que haja uma associação entre as instâncias de propriedades e de relacionamentos não taxonômicos às suas respectivas classes da ontologia. Ambas as ontologias, geram o arquivo OWL, entretanto a GAODTool [5] faz uso de construtores da OWL como o *"Union_of"* para agrupar suas relações enquanto a DIPPAOTool [3] agrupa através do *"domain"* e *"range"*.

A OWL disponibiliza três construtores que realizam combinações booleanas para manipulações das extensões de classes, chamados de conjunto de operadores. Os membros das classes construídas são especificados pelo conjunto de operadores AND, OR e NOT, usados na Lógica Descritiva (DL), a fim de que sejam usados em classes [31]. Segundo Bechhofer et al. [31], são três estes operadores:

OWL:intersectionOf: declara uma classe cuja sua extensão contém somente indivíduos que são membros da extensão de classe de todas classes descritas em uma lista, ou seja, possui duas classes com uma instância em comum. Este operador é análogo à conjunção lógica, AND. Por exemplo, a classe *"WomanAgent"* é exatamente a interseção da classe *"Agent"* e *"PersonWoman"*. A Figura 3.44 descreve o exemplo.

```
1 <owl:Class rdf:ID="WomanAgent">
2 <owl:intersectionOf rdf:parseType="Collection">
3 <owl:Class rdf:about="#Agent">
4 <owl:Class rdf:about="PersonWoman"/>
5 </owl:intersectionOf>
6 </owl:Class>
```

Figura 3.44: Trecho de código descrevendo o construtor *"intersection_of"*

A OWL:complementOf: declara uma classe cuja extensão contém exatamente os indivíduos que não pertencem à extensão da classe. Este operador é análogo a negação lógica, NOT. Um exemplo pode ser visualizado na Figura 3.45.

```
1 <owl:Class rdf:ID="Man"/>
2 <owl:complementOf>
3 <owl:Class rdf:about="#Woman">
4 </owl:complementOf>
5 </owl:Class>>
```

Figura 3.45: Trecho de código descrevendo o construtor *"complement_of"*

OWL:unionOf: declara uma classe anônima cuja extensão de classe contém indivíduos que ocorrem em pelo menos uma extensão de classe das classes descritas em uma lista. Este operador é análogo à disjunção lógica, OR.

A GAODTool agrupa suas propriedades utilizando o operador *"union_of"*. A propriedade OWL: *union_of* vincula uma classe para uma lista de descrições de

classe, ou seja, declara a uma classe anônima do tipo *"Collection"* que contém os indivíduos que ocorrem em pelo menos uma das extensões da classe descritas na lista de classes. O trecho de código descrito na Figura 3.46 representado como as propriedades são representadas na GAODTool [5].

```
1 <OWL:Class rdf:ID="IdentifyHusband"/>
2       <OWL:unionOf rdf:parseType="Collection">
3        <OWL:Class rdf:about="#PersonX">
4        <OWL:Class rdf:about="#PersonY"/>
5 </OWL:unionOf>
6 </OWL:Class>
```

Figura 3.46: Trecho de código descrevendo o construtor *"union_of"*

Porém, a ferramenta DIPPAOTool [3] utiliza outra representação para as propriedades da ontologia. Propriedades servem para descrever fatos em geral, podendo se referir a todos os membros que pertencem a uma classe [31]. Uma propriedade possui um *domain* e um *range* que ligam indivíduos de um domínio a outros indivíduos, ou seja, o domínio é a classe onde a propriedade se encontra e o range é a classe que estabelece essa relação, por isso foi utilizado apenas as relações nominais identificadas em cada objetivo inserido. Por exemplo, a propriedade *"husband_of"* possui o domain *"Marriage"* e o range *"Person"* conforme ilustrado na Figura 3.47.

```
1 <rdf:Property rdf:ID="husband_of">
2       <rdfs:domain rdf:resource="#Marriage"/>
3       <rdfs:range rdf:resource="#Person"/>
4 </rdf:Property>
```

Figura 3.47: Trecho de código da DippaolTool utilizando o *"domain"* e o *"range"*

O trecho do código acima exemplifica a relação entre a propriedade *"husband_of"* às classes *"Marriage"* e *"Person"*. A propriedade *"husband_of"* é do domínio *"Marriage"* e possui como range a classe *"Person"*. Esta é a forma que as propriedades são descritas na ferramenta DIPPAOTool [3]. Para o desenvolvimento da Apponto-ProTool foi utilizada a mesma solução da DIPPAOTool [3] e como consequência o problema do povoamento foi resolvido. Depois de realizada tais

alterações na GAODTool [5], a instanciação das classes foi efetivada, estas alterações permitiram o uso desta ontologia pela ferramenta TAILP [9] sem a necessidade de realizar mudanças em seus arquivos, devido esta receber como entrada apenas o arquivo OWL povoado.

O exemplo do resultado da integração das ferramentas pode ser visualizado na Figura 3.48, que apresenta as classes obtidas pela GAODTool e as instâncias pela DIPPAOTool [3]. A Figura 3.49 apresenta os axiomas obtidos pela ferramenta TAILP.

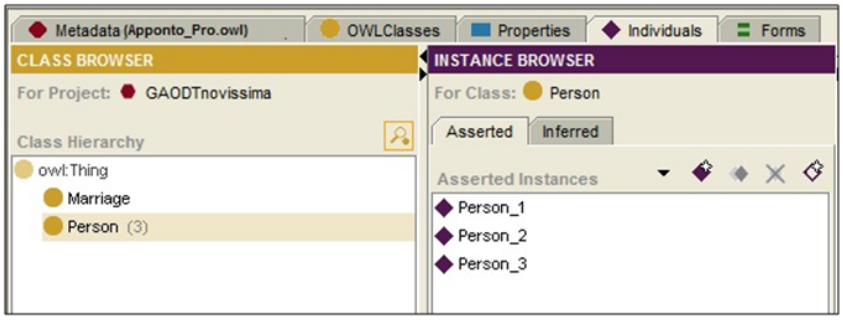


Figura 3.48: Exemplo das classes e instâncias obtidas após a integração das ferramentas GAODTool e DIPAOTool

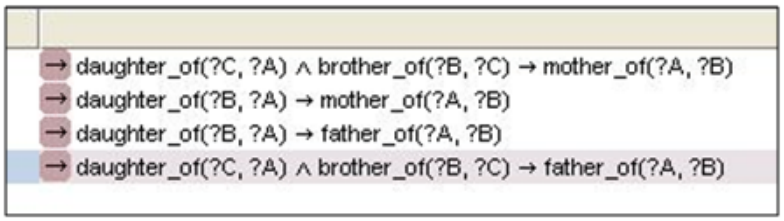


Figura 3.49: Exemplo das regras obtidas com a integração da ferramenta TAILP

A escolha da Apponto-ProTool para a construção de ontologias de aplicação possui como vantagens a capacidade do usuário desenvolver uma ontologia completa, o uso de abordagens já existentes que permitem uma melhor representação do conhecimento de um domínio, a definição do principal objetivo que o sistema pretende alcançar bem como a facilidade de manipulação da ferramenta devido possuir uma interface dinâmica e intuitiva que orienta o usuário na execução de cada uma de suas fases.

Como desvantangens podemos citar o método de entrada dos objetivos, uma vez que é um trabalho manual que consome tempo e esforços e a necessidade de um servidor web que seja executado em plataforma Windows, devido a algumas das bibliotecas utilizadas na integração das ferramentas só executarem neste sistema

opracional. Na Tabela 3.18 é apresentada a classificação da Apponto-ProTool com relação aos critérios adotados e analisados nesta discussão.

Tabela 3.18: Critérios de classificação da Apponto-ProTool

Critério	**Apponto-ProTool**
O tipo de ontologia desenvolvida	Ontologias de aplicação
A ordem na qual as ferramentas foram integradas	GAODTool DIPPAOTool TAILP
O reúso de abordagens existentes	Sim
Elementos gerados	C, H, I, R, P, A

Uma funcionalidade a ser implementada futuramente é a implementação das fases "Aprendizagem de Classes e Relações Taxonômicas" e "Aprendizagem de Relacionamentos não Taxonômicos", bem como a automação da ferramenta de maneira que facilite ainda mais o trabalho do desenvolvedor. Estas questões são vistas como uma limitação da ferramenta podendo ser implementada em trabalhos futuros.

A Tabela 3.19 apresenta as ferramentas utilizadas pela Apponto-ProTool, onde descreve de forma sucinta suas particularidades bem como as modificações realizadas em cada uma delas, descritas anteriormente, para que pudessem ser integradas na Apponto-ProTool de maneira efetiva.

Tabela 3.19: Modificações realizadas nas ferramentas utilizadas pela Apponto-ProTool

Ferramentas	Particularidade	Modificações realizadas
GAODTool	Ambiente de desenvolvimento – PHP;	Alteração no método de entrada dos relacionamentos não taxonômicos permitindo que sejam consideradas as relações tipo_de;
	Considera os verbos presentes nas frases como as relações não taxonômicas descritas;	Adaptação da ontologia permitindo que seja declarada utilizando a sintaxe domain e range;
	Uso do operador Union_of para vincular classes da ontologia a um tipo Collection;	Adaptação da ontologia para que receba novos elementos;
DIPPAOTool	Ambiente de desenvolvimento – JAVA;	Migração do ambiente de desenvolvimento JAVA para o PHP;
		Adaptação da ontologia para o ambiente de desenvolvimento;
TAILP	Ambiente de desenvolvimento - JAVA;	Migração do ambiente de desenvolvimento JAVA para o PHP;
		Adaptação da ontologia para o ambiente de desenvolvimento;

3.9 Considerações finais

Neste capítulo foi apresentado o Apponto-Pro, um processo incremental para o aprendizado e povoamento de ontologias de aplicação, sendo a principal contribuição desta pesquisa o desenvolvimento e disponibilização de um processo que integra e reúsa abordagens preexistentes desenvolvidas pelo grupo GESEC: GAODT [5], um processo para a aquisição de relações taxonômicas de uma ontologia [6], o método para a aprendizagem de hierarquia de conceitos utilizando análise formal de conceitos – AFC [7], TARNT [8], DIPPAO [3] e TAILP [9]. Todas as fases do processo foram descritas e ilustradas através de exemplos no domínio do Direito da Família.

A Apponto-Pro apresenta algumas vantagens, dentre elas podemos citar a construção de um ontologia de aplicação completa através de uma abordagem incremental para o aprendizado e povoamento de ontologias de aplicação, permitindo estender a ontologia a ser gerada, expressando melhor o conhecimento sobre um dominio de interesse. Uma desvantagem do processo e da ferramenta se dá por se

tratar de uma abordagem semiautomática, necessitando da interação do usuário em algumas etapas tornando o desenvolvimento da ontologia um tanto custoso.

Ainda neste capítulo foi apresentada a ferramenta Apponto-ProTool desenvolvida para dar suporte as atividades do processo proposto juntamente com uma discussão que descreveas alterações e implementações realizadas para a integração das ferramentas que compõem a Apponto-ProTool.

4 AVALIAÇÃO

Este capítulo tem o objetivo de apresentar o resultado do estudo de caso desenvolvido para avaliar o processo Apponto-Pro descrito no capítulo 3, bem como a efetividade de suas fases através da integração de três abordagens contidas no processo: GAODT [5], DIPPAO [3] e TAILP [9]. A integração das técnicas permitiu a geração de uma ontologia de aplicação completa, composta por todos os elementos da definição formal através da ferramenta Apponto-ProTool. A avaliação consistiu na aplicação da métrica *precision* para calcular os valores obtidos a partir dos elementos retornados na ontologia automática, desenvolvida pela Apponto-Pro, comparados com os valores obtidos a partir dos elementos retornados na ontologia construída manualmente. Também é realizada uma avaliação sobre o número de instâncias retornados pelas duas ontologias de forma quantitativamente.

NA seção 4.1 é apresenta o domínio do Direito da Família [32] em que foram baseadas as ontologia e na seção 4.2 é apresentada uma discussão e avaliação dos resultados obtidos a partir da aplicação da medida de precisão. Na seção 4.3 são apresentadas as considerações finais do capítulo.

4.1 O domínio abordado: o direito da família

O Direito da Família é o ramo do Direito Civil que especifica o complexo de normas que regulam entre outros fatos, a celebração do casamento, sua validade e os efeitos que dele resultam; as relações pessoais e econômicas da sociedade conjugal; a dissolução desta; as relações entre pais e filhos; os vínculos de parentesco e os institutos complementares da tutela, curatela e da ausência [32]. A ontologia desenvolvida neste domínio consiste em uma classe raiz chamada *"Family_Law"* da qual derivam duas grandes classes *"Family_Entity"* e *"Person"*. A primeira descreve as principais entidades familiares legalmente consideradas e a segunda discrimina os elementos pessoais constituintes de uma família. A estrutura de classes, relacionamentos não taxonômicos e propriedads da ontologia *"Family_Law"* é ilustrada na Figura 4.1.

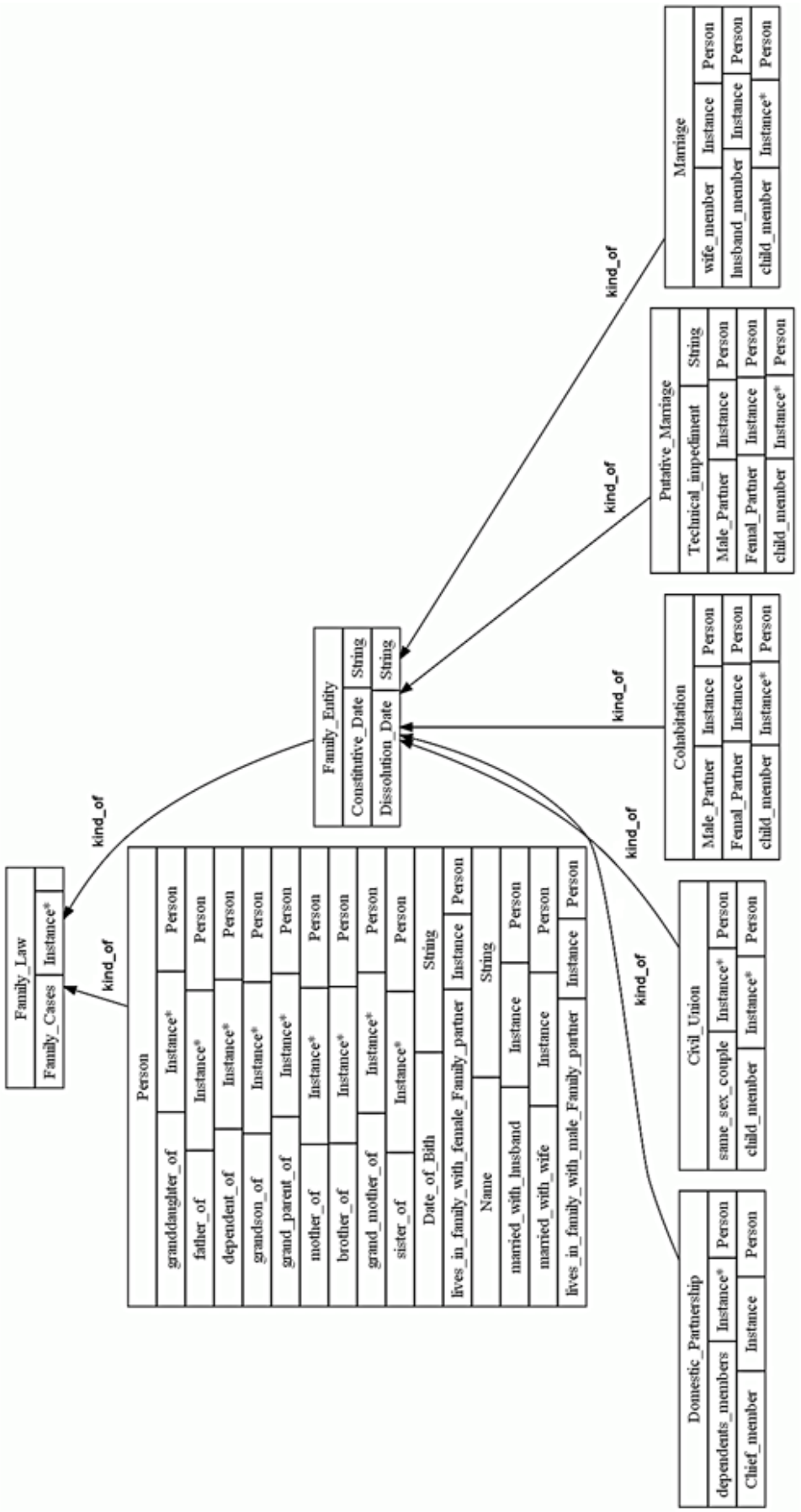


Figura 4.1: Classes, relacionamentos não taxonômicos e propriedades identificadas na ontologia de aplicação *Family_Law*

Uma entidade familiar, descrita pela classe *"Family_Entity"*, consiste de um vínculo jurídico entre duas ou mais pessoas, caracterizado obrigatoriamente por uma propriedade *"married_date"*, que indica a data de formação do vínculo e, opcionalmente, por uma propriedade *"dissolution_date"*, que indica a data de finalização do vínculo.

Há diferentes tipos de entidades familiares: *"Marriage"*, *"Cohabitation"*, *"Civil_Union"*, *"Domestic Patnership"* e *"Putative_Marriage"*, subclasses da superclasse *"Family_Entity"*. Todas as subclasses herdam as propriedades *"constitutive_date"* e *"dissolution_date"* da superclasse *"Family_Entity"*.

Um casamento, representado pela classe *"Marriage"*, é uma entidade familiar formada por esposo, esposa e, em alguns casos, um ou mais filhos. Convém ressaltar que nesta ontologia é considerado como casamento a união entre duas pessoas de sexos diferentes. Os atributos específicos da classe *"Marriage"* são os relacionamentos não taxonômicos: *"child_member"* (indica os filhos), *"husband_member"* (indica o esposo) e *"wife_member"* (indica a esposa).

A união estável representada pela classe *"Cohabitation"* é um acordo firmado entre duas pessoas, no qual ambas decidem viver juntas como se casados fossem. Os atributos específicos da classe *"Cohabitation"* são os relacionamentos não taxonômicos: *"child_member"* (indica os filhos), *"femal_partner"* (indica o parceiro feminino) e *"male_partner"* (indica o parceiro masculino).

A união civil representada pela classe *"Civil_Union"* é todo contrato civil entre duas pessoas do mesmo sexo com fins de constituir família. Os atributos específicos da classe *"Civil_Union"* são os relacionamentos não taxonômicos: *"child_members"* (indica os filhos) e *"same_sex_couple"* (indica as duas pessoas do mesmo sexo).

A parceria doméstica representada pela classe *"Domestic_Partnership"* é uma relação jurídica entre dois indivíduos que vivem juntos, mas não são unidos por casamento ou união civil, como por exemplo, um avô com seus netos. Os atributos específicos da classe *"Domestic_Partnership"* são os relacionamentos não taxonômicos: *"chief_member"* (indica membro chefe) e *"dependents_members"* (indica os membros dependentes).

O casamento putativo representado pela classe *"Putative_Marriage"* é um casamento aparentemente válido, mas legalmente inválido por um impedimento técnico, como, por exemplo, um casamento pré-existente desconhecido por uma das partes. Os atributos específicos da classe *"Putative_Marriage"* são os os relacionamentos não taxonômicos: *"child_members"* (indica os filhos), *"femal_partner"* (indica o parceiro feminino), *"male_partner"* (indica o parceiro masculino) e *"technical_impediment"* (indica o impedimento técnico).

Uma pessoa, descrita pela classe *"Person"*, é caracterizada pelas propriedades *"name"* e *"date_of_birth"*. Os relacionamentos não taxonômicos desta classe são os membros de uma família: *"father_of"*, *"mother_of"*, *"brother_of"*, *"sister_of"*, *"daugter_of"*, *"son_of"*, *"husband_of"*, *"wife_of"*, *"grandparent_of"*, *"grandmother_of"*,*"grandson_of"*,*"granddaughter_of"*, *"lives_in_family_with_female_family_partner"* e *"lives_in_family_with_male_family_partner"*.

Estas classes da ontologia *Family_Law* podem ser visualizadas graficamente na Figura 4.2.

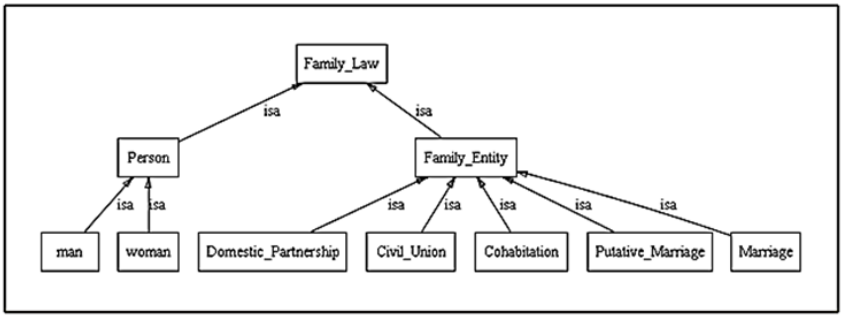


Figura 4.2: Visão geral da ontologia *Family_Law*

A seção 4.2 descreve as medidas de avaliação utilizadas juntamente com os resultados obtidos. E na seção 4.4 são discutidos os resultados encontrados através da aplicação da medida de *precision* entre a *"Family_Law"* desenvolvida manualmente pelo grupo GESEC e a *"Family_Law"* assistida pela Apponto-ProTool.

4.2 Medidas de avaliação

De acordo com Velardi et al [33] um dos objetivos da avaliação das ontologias geradas automaticamente não é apenas a comparação das diferentes

abordagens, mas a verificação da capacidade de um dado processo automático de competir com o processo totalmente humano de conceitualização de um determinado domínio.

Esta seção apresenta uma discussão sobre os resultados obtidos com a ontologia *"Family_Law"* desenvolvida pela ferramenta Apponto-ProTool em relação à ontologia *"Family_Law"* desenvolvida manualmente pelo grupo GESEC, ambas ontologias possuem o mesmo domínio de aplicação. Nesta discussão, as ontologias são avaliadas de duas formas, uma é através da aplicação da métrica precisão *precision* [34] para calcular a efetividade entre as ontologias. A segunda forma avalia as instâncias identificadas tanto pela ontologia povoada automaticamente quanto pela ontologia povoada manualmente de maneira quantitativa.

A precisão é baseada na quantidade relevante de conceitos, hierarquias, relacionamentos não taxonômicos, propriedades e axiomas retornadas pela ontologia desenvolvida automaticamente em relação a ontologia desenvolvida manualmente, ou seja, a medida de precisão reflete a quantidade de conhecimento corretamente identificados na ontologia, com relação a todo o conhecimento disponível nela. Para isso necessitam de uma ontologia que possua valores de referência, que no contexto deste trabalho é a ontologia desenvolvida manualmente.Para efeitos de discussão, as ontologias utilizadas nesta avaliação passam a ser chamadas de *Family1* quando referida a ontologia manual e *Family2* quando a ontologia automática for citada.

Segundo Dellschaft e Staab [34], a medida de precisão *precision* [34] é utilizada para comparar uma ontologia de referência, que neste caso foi a desenvolvida manualmente, com uma ontologia automática, esta última gerada pela Apponto-ProTool. Neste sentido, a adaptação da medida de precisão, não utiliza termos recuperados como se estivesse calculando a efetividade de uma técnica de recuperação de informação mas, recebe como entrada elementos que fazem parte da composição das ontologias geradas manualmente e automaticamente.

Desta forma, a aplicação da medida de precisão permite comparar a relevância dos termos contidos na *Family1* em relação aos termos contidos na *Family2*. A Figura 4.3 ilustra ontologia construída manualmente e a Figura 4.4 ontologia desenvolvida de forma automática.

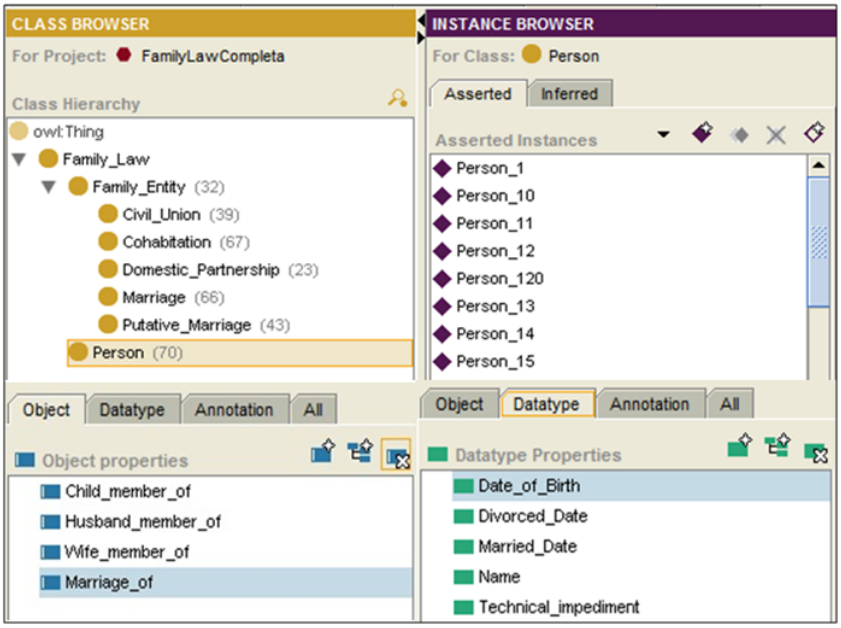


Figura 4.3: Visão parcial da ontologia *"Family_Law"* construída manualmente

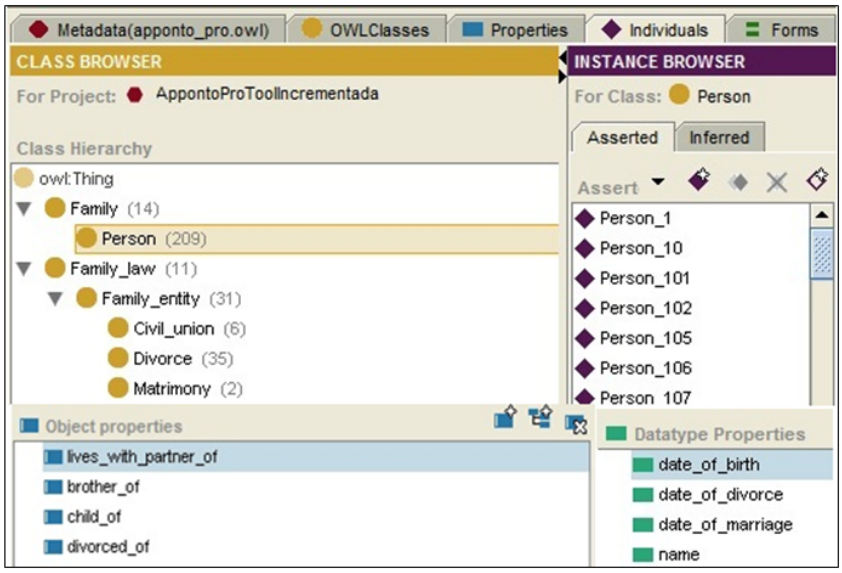


Figura 4.4: Visão parcial da ontologia *"Family_Law"* gerada automaticamente pela Apponto-ProTool

Considerando as ontologias *Family1* e *Family2*, foi calculada a precisão que considerou a quantidade de elementos relevantes entre elas através da Fórmula 4.1. Esta fórmula foi adaptada a partir dos trabalhos de Dellschaft e Staab [34] e utilizada neste contexto para calcular o valor de *precision* necessário para esta avaliação, ou que melhor atenda as necessidades do desenvolvedor.

$$\text{Precision}(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i|} \tag{4.1}$$

- Si: Lista de elementos de algum conjunto (C_c, R, P ou H) da ontologia gerada pela Apponto-ProTool.
- Sj: Lista de elementos de algum conjunto (C_c, R, P ou H) da ontologia gerada pelo especialista do domínio.

4.3 Discussão dos resultados

A quantidade de elementos identificados nas ontologias *Family1* e *Family2* são expressos na Tabela 4.1.

Tabela 4.1: Quantidade de elementos identificados nas ontologias

	C_c	**R**	**P**	**H**	**A**	**I**
Family1	21	39	8	12	20	237
Family2	7	15	5	6	12	672

Na Tabela 4.2 são demonstrados os resultados obtidos a partir do cálculo da medida de *precision*. Os valores descritos representam a precisão encontrada, de acordo com cada um dos conjuntos que compõem a ontologia.

Tabela 4.2: Valores encontrados após o cálculo da medida de *precision*

Conjuntos	**Valores de *Precision***
C_c	71%
R	73%
P	60%
H	33%
A	66%

O cálculo da medida de *precision* proposto por Dellschaft e Staab [34] foi aplicado apenas para os conjuntos C_c, H, R, P e A pois a fórmula adotada calcula os elementos em comum nas ontologias geradas manualmente e automaticamente. Para a

avaliação do conjunto de instâncias foi adotada a forma de avaliação proposta por Faria [3] que realiza uma análise comparativa para avaliar quantitativamente a ontologia povoada automaticamente e a ontologia povoada manualmente por especialistas de domínio ou engenheiros do conhecimento.

A ontologia gerada pela Apponto-ProTool obteve 71% de *precision* para o conjunto C_c, 73% para o conjunto R, 60% para o conjunto P, 33% para o conjunto H e 66% para o conjunto A.

O conjunto R obteve o maior valor de precisão, fato este já esperado, pois assim como um relacionamento pode existir entre pares de conceitos predefinidos na ontologia, por exemplo <C_1 relacionamento C_2>, vários relacionamentos também podem referenciar um mesmo conceito, por exemplo a classe *"Marriage"* possui os relacionamentos não taxonômicos *"child_members"* (indica os filhos), *"husband_member"* (indica o esposo) e *"wife_member"* (indica a esposa.

O conjunto C_c retornou o segundo maior valor de precisão encontrado que foi de 71%. Como a ontologia *Family1* é desenvolvida através de um ciclo incremental, é possível a inserção de novas classe nesta ontologia a cada interação realizada, tendo como resultado um rico conjunto C_c. O conjunto A retornou 66% de precisão, este valor está diretamente ligado a quantidade de instâncias identificadas na ontologia, pois os axiomas são identificados a partir de instâncias de relacionamentos não taxonômicos deduzidos, contudo foi observado que mesmo uma ontologia com muitas instâncias não significa que a quantidade de axiomas extraídos será extensa pois estes são gerados com a aplicação de programação em lógica indutiva, ou seja, nem todas as instâncias atendem aos requisitos de inferência para retornar axiomas. O conjunto P obteve 60% de precisão, este valor está relacionado diretamente ao conjunto C_c, pois as propriedades de classes de ontologias de mesmo domínio são, em geral semelhantes, em muitos casos os resultado divergentes se deve a questões relacionadas a nomenclatura.

O menor valor de precisão foi identificado no conjunto H, fato este já esperado, pois as ontologias foram geradas por especialistas que possuem requisitos distintos, mesmo para os casos em que as ontologias possuem um mesmo domínio de aplicação estas possuem uma quantidade de informações muito grande que ao serem organizadas nem sempre possuem hierarquias semelhantes.

Para a avaliação do conjunto de instâncias foram observados que a *Family2* obteve como resultado um rico conjunto I comparados a *Family1*, sendo um resultado já previsto, uma vez que a ontologia desenvolvida pela Apponto-ProTool extrai as instâncias a partir do processamento automático de uma grande quantidade de documentos e as instâncias contidas na *Family1* são inseridas de forma manual. A vantagem deste tipo de avaliação é que o especialista de domínio e o engenheiro de conhecimento são capazes de identificar se a ontologia povoada automaticamente retorna as instâncias de forma efetiva ou não, enquanto que uma desvantagem é que a avaliação é lenta e subjetiva. Lenta porque é realizada de forma manual, o que demanda tempo para avaliá-las, e subjetiva devido tais ontologias serem avaliadas por especialistas do domínio e engenheiros do conhecimento que possuem diferentes interesses.

Podemos concluir que os resultados obtidos após a avaliação oferecem resultados satisfatórios pois a variação dos valores retornados pelas ontologias foi muito baixo em apenas no conjunto H, enquanto que nos demais estes resultados foram relativamente satisfatórios.

4.4 Considerações finais

Neste capítulo foi apresentado a avaliação realizada para calcular a efetividade da ontologia *Family_Law* a partir da integração das ferramentas GAODTool [5], DIPPAOTool [3] e TAILP [9], que deram origem a ferramenta Apponto-ProTool. A partir desta integração pode ser constatada a construção de uma ontologia de aplicação completa por todos os elementos da definição formal de ontologias assim como proposto nos objetivos deste trabalho.

A avaliação realizada permitiu calcular a efetividade dos resultados obtidos tanto pela ontologia desenvolvida automaticamente quanto pela ontolologia desenvolvida manualmente através da aplicação da medida de *precision* para os conjuntos C_c, R, H, P e A, para o conjunto I foi realizada uma análise comparativa que avaliou quantitativamente as instâncias das ontologias.

5 CONCLUSÕES

Este trabalho apresentou as principais abordagens que trabalham na construção de ontologias, destacando aspectos positivos e as limitações encontradas, e propôs um processo incremental para o aprendizado e povoamento de ontologias de aplicação que consiste em seis fases: "Construção de Ontologia Base", "Aprendizagem de Classes e Relações Taxonômicas", "Aprendizagem de Relacionamentos não Taxonômicos", "Povoamento de Ontologia", "Inserção de Axiomas" e "Incrementar Ontologia".

Na definição da ordem de descoberta dos elementos da ontologia, o Apponto-Pro inicia seu processo com a técnica GAODT [5] devido esta formalizar e definir o principal objetivo que o sistema deseja alcançar. A segunda abordagem integrada foi a técnica para a aquisição de relações taxonômicas de uma ontologia [32] e o método para a aprendizagem de hierarquias de conceitos utilizando análise formal de conceitos - AFC [33] que realizam a identificação de relações taxonômicas de uma ontologia. A primeira através de métodos estatísticos e a segunda através da aplicação de algoritmos de AFC, isso permite o enriquecimento, a aprendizagem e a extensão da ontologia. TARNT [34] foi integrada neste ponto do processo devido precisar de um conjunto de conceitos para identificar seus relacionamentos não taxonômicos e a ontologia em desenvolvimento possui um rico conjunto de C_c e H.

Visto que neste ponto do processo a ontologia não possui o elemento I, ou seja, as classes geradas não contêm instâncias, há a necessidade de se integrar ao processo uma técnica que trabalha com o povoamento de ontologias, logo, foi integrada ao processo a DIPPAO [7], suprindo a carência do elemento I. Vale ressaltar que os axiomas gerados até este ponto da integração estão implícitos na ontologia e por isso foi integrada ao processo a técnica TAILP [15] que trabalha na geração de axiomas, estes retornados de forma explícita na ontologia. Os axiomas que esta técnica insere permite a integração das regras de LPO desenvolvidas pela GAODT [5], com as regras de LPO geradas a partir da TAILP [15], obtendo como resultado uma ontologia de aplicação completa.

A Apponto-ProTool é uma ferramenta desenvolvida que dá suporte ao processo proposto utilizando durante o seu processamento ontologias e fontes de informações textuais. A ferramenta foi desenvolvida no ambiente do PHP utilizando funcionalidades de ferramentas desenvolvidas na linguagem Java. Uma das limitações da ferramenta é o fato de trabalhar somente com fontes de informações textuais em língua inglesa, pelo fato de ter ferramentas que auxiliam a aplicação de PLI e EI desenvolvidas somente em inglês. Outra limitação é a necessidade de um servidor web que seja executado em plataforma Windows devido a algumas bibliotecas utilizadas pela ferramenta serem exclusivas da plataforma citada.

A ferramenta realiza um processamento semiautomático pois em algumas fases necessitar da intervenção do usuário. Para uma total automação da Apponto-ProTool todas as ferramentas que a compõem deveriam ser reimplemantadas e esta é uma tarefa custosa sendo de difícil implementação, uma vez que a ontologia desenvolvida é guiada por objetivos baseados a partir do conhecimento do especialista do domínio e do engenheiro do conhecimento cabendo a eles definir o principal objetivo que pretendem alcançar, além de permitir ao desenvolvedor a escolha quais as classes, relações e propriedades que guiarão todo o processo de construção da ontologia em questão.

Para avaliar o processo proposto foi desenvolvido um estudo de caso, que consistiu na construção da ontologia de aplicação *Family_Law* no domínio do Direito da Família onde sua efetividade foi calculada através da medida de precisão *precison* [34] que compara uma ontologia construída manualmente e uma ontologia automática.

Para esta avaliação foi utilizada somente a medida de precisão *precision* [34]. Uma outra medida de avaliação que poderia ser utilizada é o *recall* e a Medida-F [3]. O recall serve para indicar a proporção de elementos relevantes em resposta a uma consulta do usuário, no entanto para a ontologia em questão o usuário não deseja consultar algo e sim gerar e incrementar novos elementos na ontologia de forma que a enriqueça cada vez mais. Uma vez que o recall não é calculado a Medida-F [3] é descartada pois esta é uma medida harmônica entre o *recall* [34] e a precisão *precision* [34].

As próximas seções são descritas as constribuições e resultados desta pesquisa, a publicação realizada e os trabalhos futuros a serem desenvolvidos.

5.1 Contribuições e resultados da pesquisa

As principais contribuições desta pesquisa foram:

- Análise do estado da arte de algumas abordagens que trabalham no desenvolvimento de ontologias de aplicação.
- Uma análise comparativa das técnicas estudadas, ressaltando suas principais características bem como suas carências.
- Desenvolvimento do processo Apponto-Pro, por meio de uma abordagem incremental para o aprendizado e povoamento de ontologias de aplicação, concretizando um trabalho futuro anteriormente citado por Santos [5] e Leite [9].
- O Apponto-Pro proporciona a construção de uma ontologia de aplicação completa, característica não abordada por nenhuma das técnicas analisadas na fundamentação teórica e no estado da arte.
- Desenvolvimento da ferramenta de software Apponto-ProTool, que fornece suporte semiautomático para a construção de ontologias de aplicação, permitindo assim que o tempo de desenvolvimento de ontologias de aplicação reduza consideravelmente.
- Aplicação da Apponto-ProTool no domínio do Direito da Família para a construção da ontologia de aplicação *"Family_Law"*.

5.1.1 Publicação

A publicação **"Apponto-Pro: Um Processo Incremental para o Aprendizado e Povoamento de Ontologias**, Suzane Carvalho dos Santos e Rosario Girardi, 9ª Conferência Ibérica de Sistemas y Tecnologias de Información – CISTI, pp 167-172. Barcelona, Espanha. 18 a 21 de junho de 2014", apresentou as principais técnicas da fundamentação teórica e do estado da arte juntamente com o ciclo de desenvolvimento do processo Apponto-Pro para a construção e aprendizado de ontologias de aplicação.

5.2 Trabalhos futuros

Alguns trabalhos podem ser desenvolvidos a partir dos resultados obtidos neste:

- Implementação das fases “Aprendizagem de Classes e Relações Taxonômicas” e “Aprendizagem de Relacionamentos não Taxonômicos”, contribuindo para o enriquecimento e extensão da ontologia de maneira que represente melhor o conhecimento a cerca de um domínio.
- Automação da ferramenta Apponto-ProTool a fim de reduzir os custos e esforços durante o desenvolvimento da ontologia permitindo a aquisição automática de conhecimento.
- Aprimorar a avaliação da efetividade do processo através da construção de ontologias de aplicação em outros domínios.

Referências Bibliográficas

[1] K. K. Breitman, *Web semântica: a internet do futuro*, vol. 1. Rio de Janeiro: LTC, 2005. ISBN 85-216-1466-7.

[2] N. Guarino, *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, vol. 46. IOS press, 1998.

[3] C. G. de Faria, "Um processo independente de domínio para o povoamento automático de ontologias a partir de fontes textuais," in *Tese de Doutorado, Universidade Federal do Maranhão*, (São Luís), 2013.

[4] T. R. Gruber, "Towards Principles for the Design of Ontologies Used for Knowledge Sharing," in *Formal Ontology in Conceptual Analysis and Knowledge Representation* (N. Guarino and R. Poli, eds.), (Deventer, The Netherlands), Kluwer Academic Publishers, 1993.

[5] L. E. Santos, "Uma técnica orientada por objetivos para a construção de ontologias de aplicação," in *Dissertação, Universidade Federal do Maranhão*, (São Luís), 2012.

[6] J. Correia, "Um processo para a aquisição de relações taxonômicas de uma ontologia," in *Dissertação, Universidade Federal do Maranhão*, (São Luís), 2011.

[7] V. Galvão, "Um método para a extração de taxonomias de ontologias utilizando análise formal de conceitos," in *Monografia, Universidade Federal do Maranhão*, (São Luís), 2011.

[8] I. Serra, "Uma solução efetiva para a aprendizagem de relacionamentos não taxonômicos de ontologias," in *Tese de Doutorado, Universidade Federal do Maranhão*, (São Luís), 2014.

[9] N. B. Leite, "Aquisição automática de axiomas de ontologias utilizando a programação em lógica indutiva," in *Monografia, Universidade Federal do Maranhão*, (São Luís), 2012.

[10] L. E. Santos and R. Girardi, "Building application ontologies through knowledge system goals.," in *KEOD* (J. Filipe and J. L. G. Dietz, eds.), (Barcelona, Espanha), pp. 115–124, SciTePress, 2012.

[11] C. G. de Faria e Rosario Girardi, "Um processo semiautomático para o povoamento de ontologias a partir de fontes textuais," *Sys - Revista Brasileira de Sistemas de Informação*, 2010.

[12] N. F. Noy, D. L. McGuinness, *et al.*, *Ontology development 101: A guide to creating your first ontology*, vol. 15. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

[13] Agrovoc, "Framework para a construção de ontologias no domínio da agricultura," http://aims.fao.org/standards/agrovoc/about. Acesso em: 12 setembro 2013.

[14] C. H. Felicíssimo, B. K. K. da Silva, Lyrene Fernandes, and J. C. S. do Prado Leite, "Geração de ontologias subsidiada pela engenharia de requisitos.," in *WER* (L. E. G. Martins and X. Franch, eds.), pp. 255–269, 2003.

[15] A. M. M. Zahra, D Carvalho, "Poronto: Ferramenta para a construção semiautomática de ontologias em português," *Journal of Health Informatics*, 2013.

[16] PHP, "Php: Hypertext preprocessor. disponível em http://www.php.net," acesso em: 30 agosto 2013.

[17] S. Bechhofer, "Owl: Web ontology language," in *Encyclopedia of Database Systems* (L. LIU and M. ÖZSU, eds.), pp. 2008–2009, Springer US, 2013.

[18] Protégé, "Ontology editor and knowledge acquisition system. disponível em http://protege.stanford,edu," acesso em: 21 março 2013.

[19] R. Girardi, "Guiding ontology learning and population by knowledge system goals.," in *KEOD*, pp. 480–484, 2010.

[20] M. Uschold, M. Gruninger, M. Uschold, and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowledge Engineering Review*, vol. 11, pp. 93–136, 1996.

[21] OWL, "Web ontology language overview, disponível em http://www.w3.org/tr/owl-features," acesso em: 12 março 2013.

[22] C. d'Amato, N. Fanizzi, and F. Esposito, "A semantic similarity measure for expressive description logics," in *Proceedings of Convegno Italiano di Logica Computazionale (CILC05) 21-22 June 2005, Rome, Italy* (A. Pettorossi, ed.), 2005.

[23] N. B. Leite, "Aplicação da programação em lógica indutiva na construção automatizada de ontologias," in *Relatório Final de Iniciação Científica - PIBIC, CNPq, UFMA, DEINF*, (São Luís), 2011.

[24] C. Fellbaum, "Wordnet: An electronic lexical database," in *Cambridge: MIT Press*, 1998.

[25] C. G. de Faria and R. Girardi, "An information extraction process for semi-automatic ontology population.," in *SOCO* (E. Corchado, V. Snásel, J. Sedano, A. E. Hassanien, J. L. Calvo-Rolle, and D. Slezak, eds.), vol. 87 of *Advances in Soft Computing*, (Salamanca, Spain: Springer), pp. 319–328, Springer, 2011.

[26] J. S. at a Glance, "Linguagem de programação java. disponível em http://www.oracle.com/technetwork/java/index.html," acesso em: 25 julho, 2013.

[27] XHTML, "Extensible hypertext markup language," Disponível em $http : //www.w3schools.com/html/html_xhtml.asp.$, acesso em: 18 outubro 2013.

[28] CSS., "Cascading style sheets.," Disponível em http://www.w3schools.com/css. Acesso em: 11 agosto 2012.

[29] JAVASCRIPT, "Language language of the web," Disponível em $http : //www.w3cschools.com/js/default.asp$. Acesso em: 21 agosto 2012.

[30] B. Harold, "The rule markup language: Rdf-xml data model, xml schema hierarchy and xsl transformations," in *14th Internacional Conference on Aplications of Prolog INAP2001*, The University of tokyo, LNAI 2543, Octuber 2002.

[31] R. d. A. Falbo, C. S. d. Menezes, and A. R. Rocha, "A systematic approach for building ontologies," in *Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, IBERAMIA '98, (London, UK, UK), pp. 349–360, Springer-Verlag, 1998.

[32] C. Juridica, "Noções gerais de direito de família. disponível em $http$: $//www.centraljuridica.com/direito\_civil/noes\_gerais\_de\_direito\_de\_famlia.html$," Acessado em agosto 2013.

[33] P. Velardi, R. Navigli, A. Cucchiarelli, and F. Neri, "Evaluation of ontolearn, a methodology for automatic learning of domain ontologies," *Ontology Learning and Population*, 2005.

[34] K. Dellschaft and S. Staab, "On how to perform a gold standard based evaluation of ontology learning," (Athens, GA, USA), Springer, LNCS, November 2006.

A ANEXO: Objetivos e fatos no domínio do Direito da Família

Objetivos e Fatos	Tipo
List the members of a family	Objetivo
Identify the father of a person	Objetivo
Identify the mother of a person	Objetivo
Identify the son of a person	Objetivo
Identify the daughter of a person	Objetivo
Identify the brother of a person	Objetivo
Identify the sister of a person	Objetivo
List the grandfather of a person	Objetivo
List the grandmother of a person	Objetivo
Identify the great grandfather of a person	Objetivo
Identify the great grandmother of a person	Objetivo
List the grandparents of a person	Objetivo
List the great grandson of a person	Objetivo
List the great granddaughter of a person	Objetivo
Identify the grandson of a person	Objetivo
Identify the granddaughter of a person	Objetivo
List the uncle of a person	Objetivo
List the aunt of a person	Objetivo
Identify the cousin of a person	Objetivo
Identify the nephew of a person	Objetivo
Identify the niece of a person	Objetivo
Identify the wife of a person	Objetivo
Identify the husband of a person	Objetivo
List the parents of a person	Objetivo
Identify the father in law of a person	Objetivo
Identify the mother in law of a person	Objetivo

Identify the brother in law of a person	Objetivo
Identify the sister in law of a person	Objetivo
List the godfather of a person	Objetivo
List the godmother of a person	Objetivo
Identify the stepfather of a person	Objetivo
Identify the stepmother of a person	Objetivo
List the halfbrother of a person	Objetivo
List the halfsister of a person	Objetivo
Identify the family entity of the family law	Objetivo
Find a matrimony entity of a person	Objetivo
Find a divorce entity of a person	Objetivo
Find a civil union entity of a person	Objetivo
Find a annulment entity of a person	Objetivo
A person has name	Fato
A person has date of birth	Fato
A person has date of matrimony	Fato
A person has date of divorce	Fato
A person has date of annulment	Fato
A person has adress	Fato
A person has phone number	Fato
A person has e-mail	Fato
A person has father	Fato
A person has mother	Fato
A person has son	Fato
A person has daugther	Fato
A person has brother	Fato
A person has sister	Fato
A person has grandfather	Fato
A person has grandmother	Fato
A person has grandson	Fato
A person has wife	Fato
A person has husband	Fato

A person has parents	Fato
A person has uncle	Fato
A person has aunt	Fato
A person has children common	Fato
A person has unusual children	Fato
A person has partner	Fato
A person has descendents	Fato
A person has ascending	Fato
A person has spouse	Fato
A person cousin	Fato
A person has nephew	Fato
A person has niece	Fato

B ANEXO: Axiomas desenvolvidos no domínio do Direito da Família

$\exists$ personX, $\exists$ personY | father_of(personX,personY) $\wedge$ mother_of(personX,personY) $\rightarrow$ son_of(personX,personY)

$\exists$ personX, $\exists$ personY | brother_of(personX,personY) $\wedge$ sister_of(personX,personY) $\rightarrow$ daughter_of(personX,personY)

$\exists$ personX, $\exists$ personY | grandfather_of(personX,personY) $\wedge$ grandmother_of (personX,personY) $\rightarrow$ grandson_of(personX,personY)

$\exists$ personX, $\exists$ personY | great_grandfather_of(personX,personY) $\wedge$ great_grandmother_of(personX,personY) $\rightarrow$ grandparents_of(personX,personY)

$\exists$ personX, $\exists$ personY | uncle_of(personX,personY) $\wedge$ aunt_of(personX,personY) $\rightarrow$ cousin_of(personX,personY)

$\exists$ personX, $\exists$ personY | godfather_of(personX,personY) $\wedge$ godmother_of(personX,personY) $\rightarrow$ nephew_of(personX,personY)

$\exists$ personX, $\exists$ personY | godfather_of(personX,personY) $\wedge$ godmother_of(personX,personY) $\rightarrow$ niece_of(personX,personY)

$\exists$ personX, $\exists$ personY | father_in_law_of(personX,personY) $\wedge$ mother_in_law_of(personX,personY) $\rightarrow$ brother_in_law_of(personX,personY)

$\exists$ personX, $\exists$ personY | father_in_law_of(personX,personY) $\wedge$
mother_in_law_of(personX,personY) $\rightarrow$ sister_in_law_of(personX,personY)

$\exists$ personX, $\exists$ personY | wife_of(personX,personY) $\wedge$ husband_of(personX,personY)
$\rightarrow$ partner_of(personX,personY)

$\exists$ personX, $\exists$ personY | stepfather_of(personX,personY) $\wedge$
stepmother_of(personX,personY) $\rightarrow$ halfdaugther_of(personX,personY)

$\exists$ personX, $\exists$ personY | stepfather_of(personX,personY) $\wedge$
stepmother_of(personX,personY) $\rightarrow$ halfson_of(personX,personY)

$\exists$ personX, $\exists$ personY | halfbrother_of(personX,personY) $\wedge$
halfsister_of(personX,personY) $\rightarrow$ parents_of(personX,personY)

$\exists$ personX, $\exists$ personY | family_cases_of(family_entity,family_law) $\wedge$
family_law_of(personX,personY) $\rightarrow$ family_cases_of(personX,personY)

$\exists$ personX, $\exists$ personY | family_entity_of(personX,personY) $\wedge$
family_law_of(personX,personY) $\rightarrow$ divorced_of(personX,personY)

$\exists$ personX, $\exists$ personY | family_entity_of(personX,personY) $\wedge$
family_law_of(personX,personY) $\rightarrow$ married_with(personX,personY)

$\exists$ family_entity, $\exists$ matrimony, $\exists$ person | married_with(personX,personY) $\rightarrow$
family_cases_of(matrimony,family_entity)

$\exists$ family_entity, $\exists$ divorce, $\exists$ person | divorcied_of(personX,personY) $\rightarrow$
family_cases_of(divorce,family_entity)

$\exists$ family_entity, $\exists$ civil_union, $\exists$ person $\mid$ lives_with_partner(personX,personY) $\rightarrow$ family_cases_of(civil_union,family_entity)

$\exists$ family_entity, $\exists$ annulment, $\exists$ person $\mid$ separate_of(personX,personY) $\rightarrow$ family_cases_of(annulment,family_entity)

C ANEXO: Axiomas desenvolvidos em RuleML

<Assert>

<Rulebase mapClosure="universal»

<Implies>

<And>

<Atom>

<Rel>father_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>mother_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>son_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>daughter_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>son_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>parents_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>grandfather_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>grandmother_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>grandson_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>uncle_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>aunt_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>nephew_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>father_in_law_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>mother_in_law_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>brother_in_law_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>great_grandfather_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>great_grandmother_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>great_grandson_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>godfather_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>godmother_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

```
</Atom>

</And>

<Atom>

<Rel>godson_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</Implies>

<Implies>

<And>

<Atom>

<Rel>halfsister_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

<Atom>

<Rel>halfbrother_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>

</And>

<Atom>

<Rel>stepfather_of</Rel>

<Var>PersonX</Var>

<Var>PersonY</Var>

</Atom>
```

```
</Implies>
<Implies>
<And>
<Atom>
<Rel>husband_of</Rel>
<Var>PersonX</Var>
<Var>PersonY</Var>
</Atom>
<Atom>
<Rel>wife_of</Rel>
<Var>PersonX</Var>
<Var>PersonY</Var>
</Atom>
</And>
<Atom>
<Rel>matrimony_entity_of</Rel>
<Var>Matrimony_entity</Var>
<Var>Person</Var>
</Atom>
</Implies>
<Implies>
<And>
<Atom>
<Rel>date_of_birth</Rel>
<Var>Date_of_birth</Var>
<Var>Person</Var>
```

```
</Atom>
<Atom>
<Rel>date_of_matrimony</Rel>
<Var>Date_of_matrimony</Var>
<Var>Person</Var>
</Atom>
<Atom>
<Rel>date_of_divorce</Rel>
<Var>Date_of_divorce</Var>
<Var>Person</Var>
</Atom>
<Atom>
<Rel>date_of_annulment</Rel>
<Var>Date_of_annulment</Var>
<Var>Person</Var>
</Atom>
<Atom>
<Rel>adress</Rel>
<Var>Person</Var>
<Var>Adress</Var>
</Atom>
<Atom>
<Rel>phone_number</Rel>
<Var>Person</Var>
<Var>Phone_number</Var>
</Atom>
```

<Atom>

<Rel>email</Rel>

<Var>Person</Var>

<Var>Email</Var>

</Atom>

</And>

<Atom>

<Rel>name</Rel>

<Var>Person</Var>

<Var>Name</Var>

</Atom>

</Implies>

</Rulebase>

</Assert>

D ANEXO: Axiomas inferidos na ontologia

mother_of(?C, ?A) $\wedge$ mother_of(?C, ?B) $\rightarrow$ brother_of(?A, ?C)

father_of(?C, ?A) $\wedge$ father_of(?C, ?B) $\rightarrow$ brother_of(?A, ?B)

daughter_of(?C, ?A) $\wedge$ brother_of(?B, ?C) $\rightarrow$ mother_of(?A, ?B)

daughter_of(?B, ?A) $\rightarrow$ mother_of(?A, ?B)

daughter_of(?B, ?A) $\rightarrow$ father_of(?A, ?B)

daughter_of(?C, ?A) $\wedge$ brother_of(?B, ?C) $\rightarrow$ father_of(?A, ?B)

brother_of(A,B) $\rightarrow$ father_of(C,A)

brother_of(A,B) $\rightarrow$ father_of(C,A), grandfather_of(D,A)

brother_of(A,B) $\rightarrow$ father_of(C,A), mother_of(D,A)

brother_of(A,B) $\rightarrow$ father_of(C,A), father_of(C,B)

brother_of(A,B) $\rightarrow$ father_of(C,A), father_of(C,D)

father_of(?C, ?A) $\wedge$ father_of(?C, ?B) $\wedge$ sister_of(?A, ?B) $\rightarrow$ brother_of(?A, ?B)

father_of(?A, ?C) $\wedge$ father_of(?C, ?B) $\rightarrow$ grandfather_of(?A, ?B)

brother_of(A,B)

mother_of(?A, ?C) $\wedge$ father_of(?B, ?C) $\rightarrow$ married_with(?A, ?B)

father_of(?C, ?A) $\wedge$ mother_of(?C, ?B) $\rightarrow$ brother_of(?A, ?C)

grandmother_of(?A, ?B) $\rightarrow$ grandparent_of(?A, ?B)

daughter_of(?A, ?C) $\wedge$ mother_of(?C, ?B) $\rightarrow$ sister_of(?A, ?B)

family_cases_of(?A, ?C) $\wedge$ family_cases_of(?B, ?C) $\rightarrow$ child_of(?A, ?B)

husband_of(?B, ?A) $\rightarrow$ wife_of(?A, ?B)

father_of(?A, ?B) $\rightarrow$ mother_of(?A, ?B)