Capítulo

1

Introdução ao desenvolvimento colaborativo de regras SWRL com o SWRL Editor

João Paulo Orlando, Adriano Rívolli e Dilvan de Abreu Moreira
Dept. of Computer Science, ICMC Universidade de São Paulo – Campus de São Carlos, Caixa
Postal 668, 13560-970, São Carlos-SP, Brazil
{orlando, rivolli, dilvan}@icmc.usp.br

Abstract

This chapter presents an introduction to the development of rule sets in SWRL (Semantic Web Rule Language). The SWRL language (a W3C standard) allows the combination of rules and ontology terms (defined in OWL Web Ontology Language) to increase the expressiveness of both. The chapter presents a brief introduction to the semantic web, in which we introduce the OWL language, a W3C standard for representing and sharing ontologies over the Web. After that, the chapter shows a hands-on introduction to SWRL where we build a small ontology with a rule set. This rule set will be created and tested using the SWRL Editor, a new SWRL rule editor integrated with Web-Protégé (the Web version of the Protégé ontology editor).

Resumo

Este capítulo apresenta uma introdução ao desenvolvimento de conjuntos de regras em SWRL (Semantic Web Rule Language). A linguagem SWRL (um padrão do W3C) permite a combinação de regras e termos de ontologias (definidos em OWL Web Ontology Language) para aumentar a expressividade de ambos. O capítulo apresenta uma breve introdução sobre web semântica, na qual será abordada a linguagem OWL, uma linguagem padrão do W3C para representar e compartilhar ontologias na Web. Em seguida, é feita uma introdução prática da linguagem SWRL com a construção de uma pequena ontologia com um conjunto de regras. Para a criação desse conjunto de regras, será usado o SWRL Editor, um novo editor de regras SWRL integrado ao Web-Protégé (versão Web do editor de ontologias Protégé).

1.1. Introdução

A Web Semântica é uma maneira de explorar a associação de significados explícitos aos conteúdos de documentos presentes na Web, para que esses possam ser processados diretamente ou indiretamente por máquinas [Berners-Lee e Fischetti 2008]. Para possibilitar esse processamento, os computadores necessitam ter acesso a coleções estruturadas de informações (dados e metadados) e a conjuntos de regras de inferência sobre esses conteúdos (que ajudem no processo de dedução automática) para que seja possível o raciocínio automatizado sobre os mesmos [Berners-Lee et all 2001].

A Web Semântica renovou e aumentou o interesse em sistemas baseados em regras e seu desenvolvimento [Zacharias 2008]. A SWRL é a linguagem padrão para regras da Web Semântica, muitas áreas de estudo na computação estão usando SWRL, entre elas podemos citar: Serviços sensíveis ao contexto [Wusheng et all 2011], gestão de energia em ambientes domésticos [Rossello-Busquet et all 2011], sistemas de *e-learning* [Vesin et all 2011], gerenciamento de SLA (*Service Level Agreement*) para serviços de IPTV (*Internet Protocol Television*) [Seo et all 2011], cálculos de redes de co-autoria em redes sociais [Ahmedi et all 2011], sensores em ambientes inteligentes [Sadoun et all 2011], etc. Na área da informática biomédica, é possível citar outras aplicações de regras SWRL: Atribuição de notas a tumores [Levy et all 2009] e para classificar fenótipos de Autismo [Hassanpour et all 2011]. Como pode ser visto muitas áreas tem usado regras SWRL para facilitar a modelagem de problemas.

1.2. Ontologia e a Web Semântica

1.2.1. Ontologia

O W3C [Heflin 2004] afirma que as ontologias são vistas como a tecnologia de consolidação para a construção da Web Semântica. O termo é emprestado da Filosofia, em que uma ontologia é um relato sistemático da existência [Gruber 1993].

[Studer et all 1998] define ontologia como uma especificação **formal** e **explícita** de uma **conceituação compartilhada**. **Conceituação** se refere a um modelo abstrato de algum fenômeno no mundo, identificando os conceitos relevantes daquele fenômeno. **Explícito** significa que os conceitos utilizados e as restrições sobre seu uso são explicitamente definidos. **Formal** refere-se ao fato de que a ontologia deve ser legível pelas máquinas. **Compartilhado** refere-se à noção de que uma ontologia captura o conhecimento consensual, isto é, não é privado de algum indivíduo, mas aceito por um grupo.

Discussões e estudos aprofundados sobre a definição e conceituação do termo ontologia podem ser encontrados em [Guarino and Giaretta 1995]; [Chandrasekaran et all 1999]; [Smith et all 2001]; [Almeida and Bax 2003] e [Corazzon 2010]. Mesmo sem um consenso sobre sua definição, ontologias compartilham características comuns e impulsionam o desenvolvimento de diversos trabalhos referentes a metodologias, ferramentas, linguagens e aplicações.

Uma ontologia é especificada por meio de componentes básicos que são as **classes**, **relações**, **axiomas** e **instâncias**, além de ser expressa por meio de uma linguagem de construção [Almeida and Bax 2003].

As **classes**, o foco da maioria das ontologias, são utilizadas para descrever os conceitos de um domínio, possibilitando a organização das classes em um sistema lógico e hierárquico contendo subclasses que representam conceitos mais específicos [Noy and McGuinness 2001].

As **relações** representam o tipo de interação entre os conceitos de um domínio e as propriedades presentes nas classes e indivíduos. Os termos *slot*, *role*, propriedade e até mesmo atributo podem ser empregados como sinônimo para as relações [Noy and McGuinness 2001] [Horridge et all 2004]. As relações podem ter características próprias como serem transitivas, simétricas, ou terem uma cardinalidade definida.

Os **axiomas** são utilizados para modelar regras assumidas como verdadeiras no domínio em questão, de modo que seja possível associar o relacionamento entre os indivíduos, além de fornecer características descritivas e lógicas para os conceitos. Para [Uschold and Grüninger 1996] os axiomas são especificados para definir a semântica e significado dos termos (classes e propriedades) e sugere que a fase de definição dos axiomas (especificação da ontologia) é a mais difícil na construção de ontologias.

Por fim, os **indivíduos**, ou instâncias das classes, são utilizados para representar elementos específicos, ou seja, os próprios dados, que juntamente com a definição de uma ontologia constituem a base de conhecimento [Noy and McGuinness 2001]. Os indivíduos representam objetos do domínio de interesse [Horridge et all 2004].

Os componentes básicos de uma ontologia são definidos por meio de uma linguagem de representação. [Horridge et all 2004] aponta que diferentes linguagens para ontologias proporcionam diferentes facilidades e recursos. Alguns exemplos de linguagens que se prestam à construção de ontologias são apresentados a seguir, na Tabela 1.1. Uma listagem mais completa de linguagens de representação de ontologias pode ser encontrada em [Su and Ilebrekke 2002] e [Almeida and Bax 2003].

Tabela 1.1 – Linguagens para construção de ontologias

Linguagens	Breve descrição
DAML	*DARP Agent Markup Language* (DAML) é precursora do DAM+OIL que originou a OWL. Sua sintaxe permite que o computador possa fazer as mesmas inferências simples que os seres humanos podem fazer [Ouellet and Ogbuji 2002].
OBO	*Open Biological Ontologies* (OBO) é uma linguagem de ontologia que tem sido frequentemente utilizada para a modelagem de ontologias nas áreas biomédicas. Faz uso de uma sintaxe textual simples que foi projetada para ser compacta, legível por humanos e fácil de ser processada [Golbreic et all 2007].
OIL	Ontology Interchange Language (OIL) é base para as linguagens da Web Semântica é precursora do DAM+OIL que originou o OWL. É uma linguagem baseada em frames juntamente com as características de lógica descritiva (*Description Logic* – DL) [Fensel et all 2001].
OWL	*Web Ontology Language* (OWL) projetada para atender a necessidade de uma linguagem de representação de ontologias para a Web. Utiliza RDF para a sintaxe e lógica descritiva para a definição de regras de inferência. É o padrão adotado no W3C como linguagem da Web Semântica [McGuinness and Harmelen 2004].
RDF	*Resource Description Framework* (RDF) é uma linguagem para definição de informação na Web. Descreve os dados (significado dos termos e conceitos) em um formato que máquinas podem processar. Utiliza o *eXtensible Markup Language* (XML) e *Uniform Resource Identifier* (URI) e conjuntos de triplas (sujeito, verbo e objeto) formam as sentenças elementares [Berners-Lee et all 2001].

Por fim, as ontologias são utilizadas para promover a interoperabilidade entre sistemas, ao representarem os dados compartilhados por diversas aplicações [Uschold

and Grüninger 2004]. Ontologias são amplamente utilizadas para fins diferentes e em diferentes comunidades.

1.2.2. Web Semântica

A Web Semântica, uma extensão da Web atual, é uma representação capaz de associar significados explícitos aos conteúdos dos documentos disponíveis na Internet, sendo que sua principal meta é possibilitar que programas processem e interpretem automaticamente esses documentos [Berners-Lee et all 2001]. Para Berners-Lee, a Web Semântica deve possibilitar que computadores sejam capazes de acessar dados estruturados e de definir regras de inferências, transformando grandes volumes de dados em informação.

Seu emprego é fortemente recomendado pelo W3C, que busca desenvolver padrões, arquiteturas de metadados e linguagens para ontologias que juntos possibilitem a integração e entendimento dos dados por computadores, agregando aos mesmos significados. Sua exploração é motivada pelo potencial que tem em transformar a Internet, vista hoje como um repositório de dados, em um repositório explícito de conhecimento, disponível tanto para pessoas como para máquinas. O papel do W3C no contexto da Web atual é o desenvolvimento de padrões, recomendações e orientações com o objetivo de levar a Web ao seu potencial máximo. Além dos avanços relacionados com as aplicações da Web, o W3C tem mobilizado grandes esforços e iniciativas para o desenvolvimento de uma Web para todos, em todos os dispositivos, baseada no conhecimento, com confiança e confiabilidade [Diniz and Cecconi 2008].

A adoção das tecnologias da Web Semântica e de ontologias na representação de dados, embora não tenha sido amplamente difundida e adotada até o momento [Shadbolt et all 2006], conta com o apoio de inúmeras empresas na divulgação e desenvolvimento de soluções que fazem o uso da Web Semântica [Feigenbaum et all 2007].

A tarefa de associar significados aos dados é possível pelo uso de tecnologias como RDF e OWL. O RDF utiliza XML e URI para proporcionar uma representação minimalista do conhecimento na Web e tem como característica principal ser simples. Por outro lado, a OWL é uma tecnologia complexa e voltada para a representação de objetos que requerem grande poder de expressividade. OWL usa RDF e possibilita a criação de ontologias para representação de conhecimento [Shadbolt et all 2006].

1.2.2.1. RDF

Resource Description Framework (RDF) é um esquema para a definição de informações na Web, provendo tecnologia para expressar o significado de termos e conceitos de modo que os computadores possam facilmente processá-los [Berners-Lee et all 2001]. Sua sintaxe pode ser definida com o uso de marcações XML e URI, empregadas para especificar entidades, conceitos, propriedades e relacionamentos. [Lassila and Swick 2004] define que RDF é a tecnologia base para o processamento de metadados, provendo assim interoperabilidade entre aplicações que trocam e processam informações na Web. Para os autores, um dos objetivos do RDF é possibilitar a criação de semântica para dados baseados em XML. Contudo, o principal objetivo do desenvolvimento do RDF é permitir a descrição de recursos de qualquer domínio específico.

Definida pelo W3C, o RDF organiza o conhecimento por meio da idéia de redes semânticas, e permite a representação de conceitos, taxonomia de conceitos e relações binárias [Almeida and Bax 2003]. Em um relato histórico, [Smith and Welty 2004] afirma que o RDF foi a primeira linguagem especificada pelo W3C para representar informações semânticas na Web.

O modelo de dados em RDF é uma forma neutra de representar as expressões utilizadas para atribuir significado aos dados. O modelo básico consiste em três tipos de objetos: recursos, propriedades e afirmações[1] que são chamadas respectivamente de sujeito, verbo e objeto, formando a tripla RDF [Berners-Lee et all 2001]. A codificação do conhecimento se concretiza em conjuntos de triplas, representadas graficamente na Figura 1.1.

Figura 1.1 – Tripla RDF [Klyne et all 2004].

O sujeito compreende qualquer coisa que pode ser expressa e descrita em RDF. Alguns exemplos de recursos são: um documento HTML; uma parte de uma página Web; um elemento XML específico; um conjunto de páginas; e objetos que não podem ser acessados diretamente na Web, como um livro impresso ou uma pessoa. Os recursos são sempre nomeados por URI, pois a expressividade deste recurso possibilita que qualquer entidade possa ter seu identificador [Lassila and Swick 2004].

O verbo consiste nas características, atributos ou relacionamentos utilizados para descrever um recurso. Cada propriedade tem um significado específico, define seus valores permitidos, os tipos de recursos que podem descrever, e sua relação com outras propriedades. O valor de uma propriedade atribuída a um recurso específico é o objeto. O valor da propriedade pode ser outro recurso, especificado por um URI, ou um valor literal (simples string ou outro tipo primitivo definido em XML). Tendo como base a tripla, a linguagem RDF possui uma sintaxe abstrata que pode ser expressa em XML, Notation 3 (N3) ou graficamente, de modo que a semântica formal é definida com base nesta sintaxe abstrata. Uma completa descrição desta sintaxe é apresentada em [Klyne et all 2004].

Uma poderosa extensão do RDF é o *RDF Schema*, usado para descrição de vocabulário utilizado para descrever os relacionamentos internos de propriedades e valores, sendo considerado um mecanismo de extensão semântica para o RDF [Brickley 2004]. O *RDF Schema* prove mecanismos para descrever grupos de recursos relacionados, além do relacionamento entre estes recursos. É possível utilizar o conceito de classes, indivíduos, propriedades e subpropriedades semelhantemente ao recurso de hierarquia de classes.

Para [Feigenbaum et all 2007], o RDF é o mais fundamental bloco de construção para a Web Semântica, pois, além de poder ser utilizado para criar dados semânticos, é também utilizado como base para as linguagens de ontologia da Web Semântica. Entretanto para [Staab et all 2001], os dados em RDF são fracamente interligados, de modo que a Web Semântica necessita de técnicas mais sofisticadas.

[1] Tradução livre do o termo *statement*

1.2.2.2. OWL

A linguagem de ontologia *Web Ontology Language* (OWL) é destinada para os que desejam grande expressividade na descrição dos objetos e seus relacionamentos [Shadbolt et all 2006]. O *RDF Schema* é uma linguagem ontológica considerada leve [Staab et all 2001], pois não contém a complexidade e riqueza de outras linguagens como OWL. Todavia, OWL usa as ligações RDF e trabalha uma camada acima do mesmo [Feigenbaum et all 2007].

O W3C publicou em 2004 a linguagem de marcação semântica OWL como recomendação para representar e compartilhar ontologias na Web [Smith and Welty 2004]. Contudo, OWL foi concebida de uma revisão da linguagem DAML+OIL incorporando as lições aprendidas durante o desenvolvimento e aplicação desta [McGuinness and Harmelen 2004].

A OWL 1 provê três sublinguagens com expressividade crescente: *OWL Lite*, *OWL DL* e *OWL Full*, sendo a primeira a mais simples e a última a mais complexa. A simplicidade e complexidade referidas aqui dizem respeito ao vocabulário, recursos e capacidades de expressividade de cada sublinguagem.

OWL Lite suporta necessidades primárias, como classificação de hierarquia e construções simples, entretanto possui restrições de cardinalidade e tem um número menor de propriedades. O uso desta sublinguagem é recomendado para migração de thesauri e outras taxonomias. *OWL DL* e *OWL Full* compartilham a máxima expressividade incluindo todas as construções da linguagem OWL. A diferença entre estas sublinguagens consiste que *OWL DL* garante que a ontologia seja sempre computável e decidível enquanto que, com *OWL Full*, não se pode ter esta garantia. Isso ocorre pelo fato de *OWL DL* respeitar as restrições de definição presentes na lógica *Description Logic*, daí o sufixo DL.

[Barder and Nutt 2003] define *Description Logic* como um formalismo matemático que representa o conhecimento de um domínio de aplicação (o "mundo") primeiramente definindo os conceitos relevantes desse domínio (sua terminologia) e depois usando esses conceitos para especificar propriedades de objetos e indivíduos que ocorrem nesse domínio (a descrição do mundo). Esse formalismo é que garante que ontologias em *OWL DL* vão ser computáveis e decidíveis. A finalidade de uma linguagem de representação de conhecimento que usa *Description Logic* vai além de armazenar e definir conceitos e afirmações. Suas definições podem ser validadas (checagem de consistência) e podem conter conhecimento implícito, obtido através de inferências lógicas.

A ferramenta capaz de gerar as inferências lógicas e checar a consistência em OWL é chamada de *reasoner*. Um dos principais testes que essa ferramenta pode realizar é verificar se uma classe é uma subclasse de outra classe, de modo que isso ajuda a manter uma hierarquia correta e identificar classes inconsistentes [Horridge et all 2004]. Em OWL, uma inconsistência acontece se uma classe não pode ter instâncias.

Para [Smith and Welty 2004], uma das vantagens do uso da linguagem OWL para a construção de ontologias é que existem ferramentas de classificação. Estes tipos de ferramenta proporcionam um suporte genérico, aplicável a todos os domínios de aplicação. Uma lista de classificadores é encontrada em [Grau et all 2008].

OWL trabalha com suposições de mundo aberto, considerando que as descrições de recursos podem estar presentes em mais de um único arquivo ou escopo. Isso implica na impossibilidade de assumir que algo não existe até que isso seja explicitamente definido, ou ainda, algo não pode ser definido como verdadeiro, pelo fato de não poder ser assumido como falso, pois o conhecimento pode existir sem ter sido adicionado à base de conhecimento [Horridge et all 2004].

Proporcionando uma visão geral da OWL, [McGuinness and Harmelen 2004] apresenta a sintaxe expressa em RDF/XML. Com isso, alguns termos possuem o prefixo rdf: ou rdfs: compreendendo os mesmos termos presentes em RDF ou *RDF Schema* respectivamente.

Recentemente o W3C agregou novos recursos à linguagem OWL dando origem a uma nova recomendação, chamada OWL 2, que é uma extensão e revisão da anterior [W3C OWL Working Group 2009].

A nova linguagem é consequência de um amadurecimento e evolução da Web Semântica, e com isso, novos recurso e possibilidades foram incorporados em OWL 2. O uso de outras linguagens além do RDF/XML para definição da sintaxe, novos recursos semânticos como redes de propriedades, definição de intervalo de dados, tipos de dados mais ricos e novos tipos de propriedades são exemplos dos novos recursos incorporados.

A justificativa encontrada por [Grau et all 2008] para justificar a necessidade da evolução da OWL, é o seu sucesso. Pois, devido ao sucesso da "OWL 1" foram identificados problemas quanto ao design da linguagem, sendo alguns de natureza crítica. Problemas como as limitações de expressividade juntamente com o fato da linguagem OWL ter sua sintaxe influenciada pelo paradigma de frames, são alguns dos problemas identificados pelo autor que justificam a evolução da linguagem.

1.3. Regras SWRL

Infelizmente a expressividade de OWL nem sempre é suficiente para modelar todos os tipos de problemas. Especialmente OWL não tem suporte a cláusulas no formato de Horn (representa uma sentença de implicação). Para suprir essa deficiência, a SWRL (*Semantic Web Rule Language*) foi criada.

A SWRL é baseada em uma combinação de *OWL DL* e *OWL Lite* [O'Connor et all 2005]. Essa combinação amplia o conjunto de axiomas da linguagem OWL, mais especificamente para poder incluir cláusulas no formato de Horn [O'Connor et all 2005]. Essas regras podem ser usadas para inferir novos conhecimentos a partir de bases de conhecimento em OWL. SWRL permite que os usuários escrevam regras para raciocínio sobre os indivíduos OWL (instâncias) que podem inferir novos conhecimentos sobre esses indivíduos.

Regras em SWRL são compostas de duas partes: o antecedente (*body*) e o consequente (*head*). Cada regra é uma implicação entre o antecedente e o consequente, que pode ser entendida como: quando as condições do antecedente são verdadeiras, então as condições do consequente também são verdadeiras. Ambas as partes consistem em uma conjunção de zero ou mais átomos, não permitindo disjunções ou negação.

Os átomos, por sua vez, são formados por um predicado e um ou mais argumentos (o número e o tipo destes argumentos são determinados pelo tipo do átomo,

que por sua vez, é definido pelo predicado utilizado). Embora a especificação W3C defina sete tipos de átomos, neste capítulo foi adotada a convenção usada em [Hassanpour et all 2009], que trabalha com seis tipos de átomos, pois, SameAs e DifferenteFrom embora possuam semânticas diferentes, sintaticamente são idênticos. Na Tabela 1.2 são apresentados os tipos de átomos e suas descrições.

Tabela 1.2 – Tipos de átomos SWRL [Hassanpour et all 2009]

Tipo de átomo	Descrição
Class	O predicado corresponde a uma classe definida na ontologia e recebe um indivíduo[2] como argumento.
Object property	O predicado corresponde a uma propriedade definida na ontologia e recebe dois indivíduos como argumentos e os relaciona entre si.
Datavalued property	O predicado corresponde a uma propriedade definida na ontologia e recebe dois argumentos: um indivíduo e um valor[3] relacionando-os.
Data range	O predicado corresponde a um tipo de dado definido na ontologia e recebe um valor como argumento.
Same/different	O predicado corresponde ao termo *"same as"* ou *"different from"* e recebe dois indivíduos como argumentos definindo que estes são iguais ou diferentes respectivamente.
Built-in	O predicado corresponde a um conjunto de funções pré-definidas (comparações, funções matemáticas, lógicas, ...) ou definidas pelo usuário que recebem um ou mais argumentos e retornam verdadeiro quando estes satisfazem o predicado.

Os tipos *Class*, *Object property* e *Datavalued property* correspondem a elementos (conceitos) definidos na ontologia. Data range correspondem aos tipos de dados utilizados na ontologia. O tipo *Same*/*different* é utilizado para explicitar a igualdade ou diferença entre dois indivíduos, exigida pelo fato de OWL trabalhar com o paradigma de mundo aberto. Por fim, os Built-ins são tipos que encapsulam as mais diversas funções, podendo inclusive ser estendidos pelos usuários. Cada tipo de átomo define o número e o tipo de argumentos suportados, conforme ilustrado na Tabela 1.3.

Tabela 1.3 – Números e tipos de argumentos suportados pelos tipos de átomos

Tipo de átomo	Número de argumentos	Tipo de argument
Class	1	i-object
Object property	2	i-object, i-object
Datavalued property	2	i-object, d-object
Same/Different	2	i-object, i-object

[2] Na Tabela "Tipos de átomos SWRL" o termo indivíduo é empregado para representar: 1) um indivíduo específico definido na ontologia; ou, 2) variáveis para indivíduos, que podem ser quaisquer indivíduos definidos na ontologia. Ou seja, corresponde ao tipo I-Object.

[3] O termo valor neste caso é uma livre tradução de *data value* e representa um dado de um tipo primitivo definido na ontologia (Ex: inteiro, real, *string*). Na Tabela "Tipos de átomos SWRL", o termo valor é empregado para representar valores constantes ou variáveis para valores. Ou seja, corresponde ao tipo D-Object.

Tipo de átomo	Número de argumentos	Tipo de argument
Built-in	2 ou mais	d-object ou i-object
Data range	1	d-object

As variáveis são tratadas como quantificadores universais e possuem o escopo limitado à regra a qual pertencem. Apenas variáveis que ocorrem no antecedente podem ocorrer no consequente.

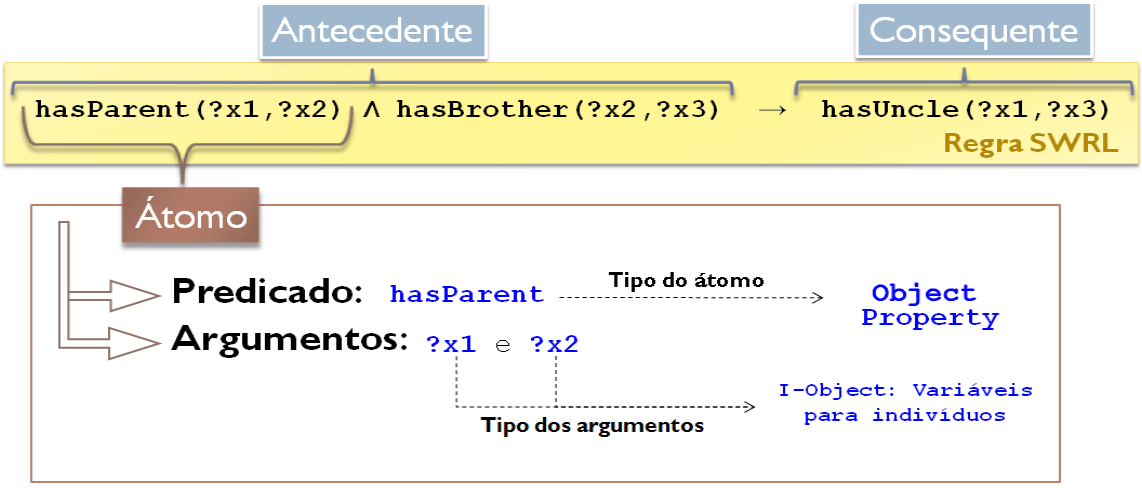

Figura 1.2 – Regra SWRL especificada no formato humano e suas partes.

Embora as regras SWRL possam ser representadas em mais de um formato, o formato de leitura humano é adotado neste capítulo. Neste formato, a seta (→) é usada para separar antecedente e consequente, o acento circunflexo (^) representa a conjunção entre os átomos e o sinal de interrogação (?) distingue as variáveis dos nomes de indivíduos. Na Figura 1.2 é apresentada uma regra SWRL, representada no formato humano, ressaltando suas partes e características.

Semanticamente, o exemplo anterior ilustra que **se** o pai/mãe de um indivíduo possui um irmão, **então** este irmão é tio do indivíduo em questão. A regra possui três átomos (todos do tipo *Object property*) sendo dois no antecedente e um no consequente (cujos predicados são: hasParent, hasBrother e hasUncle respectivamente) e faz uso de três variáveis para indivíduos: ?x1, ?x2 e ?x3. Outro exemplo, suma seria uma regra SWRL que expresse a seguinte afirmação: “uma pessoa qualquer que tenha qualquer carro será considerada um motorista”, ficaria assim:

pessoa(?x) ^ temCarro(?x, true) -> motorista(?x)

Ao executar-se a regra acima, o seu efeito seria classificar todos os indivíduos da classe pessoa que possuíssem carro como também pertencentes à classe motorista. As regras SWRL também permitem trabalhar com um indivíduo específico de uma ontologia. Por exemplo, é possível escrever uma regra diretamente para um indivíduo João, pertencente à classe pessoa, e fazer uma classificação direta deste indivíduo. Um exemplo pode ser visto na regra abaixo:

pessoa(João) ^ temCarro(João, true) -> motorista(João)

A regra acima apenas funciona para o indivíduo conhecido como João. Um dos recursos mais poderosos do SWRL é a capacidade de usuários usarem *built-ins* definidos [SWRLLanguage 2012]. *Built-in* é um predicado que pode ter um ou mais argumentos e realiza uma operação com os argumentos, avaliando e retornando verdadeiro ou falso. Por exemplo, uma regra que classifica um indivíduo da classe pessoa que seja maior de 18 anos como da classe adulto pode ser vista abaixo:

pessoa(?x) ^ temIdade(?x, ?idade) ^ swrlb:greaterThan(?idade, 18) -> adulto(?x)

SWRL permite o uso de novas bibliotecas de *built-ins* e os usuários podem definir suas próprias bibliotecas, o que torna SWRL uma linguagem muito rica em recursos de representação. Por exemplo, poderia ser criada uma biblioteca que tivesse operações de conversão de valores e busca de termos em taxonomias.

Um exemplo mais complexo do uso de SWRL é mostrado por [Hassanpour et all 2009], neste exemplo o objetivo é estabelecer relações entre duas ou mais entidade de uma ontologia. O exemplo cria uma regra que estabelece regras de condução no estado da Califórnia para menores de 18 anos:

Person(?p) ^ has_Driver_License(?p,?d) ^ issued_in_State_of(?d,?s) ^ swrlb:notEqual(?s,"CA") ^ has_Age(?p,?g) ^ swrlb:lessThan(?g,18) ^ number_of_Visiting_Days_in_CA(?p,?x) ^ swrlb:lessThan(?x,10) ^ Car(?c) ^ has_Weight_in_lbs(?c,?w) ^ swrlb:lessThan(?w,26000)

→ can_Drive(?p,?c)

Traduzindo essa regra para linguagem humana teríamos: “Qualquer indivíduo com idade inferior a 18 anos, mas que possua carta de motorista, sendo de fora da Califórnia, visitando o estado por menos de 10 dias e dirigindo um veiculo com menos de 26000 libras, pode dirigir normalmente”.

Cabe ressaltar que as regras são armazenadas na própria ontologia e o uso deste tipo de anotação permite a inferência de novos conhecimentos sobre indivíduos, uma vez que as regras correspondem a afirmações condicionais que, ao serem satisfeitas, agregam novas informações à base de conhecimento.

1.4. SWRL Editor

Nesta seção é descrita uma nova ferramenta, intitulada SWRL Editor, que foi desenvolvida nos trabalhos [Orlando 2012] e [Silva 2012]. O SWRL Editor foi desenvolvido como um plug-in para o Web-Protégé. O Web-Protégé é um editor open-source de ontologias, baseado no Protégé que funciona na Web [Tudorache et all 2008]. Ele foi desenvolvido usando o GWT (*Google Web Toolkit*). O principal motivo de se ter uma versão Web do Protégé é facilitar a colaboração entre desenvolvedores, já que não é necessário instalar programas locais. Nessa ferramenta, as ontologias estão disponíveis de forma centralizada e compartilhada. Como as ontologias estão se tornando cada vez maiores, é difícil que uma pessoa, ou um grupo pequeno de pessoas, consiga manter

grandes ontologias de forma eficaz [Tudorache et all 2008]. Daí a necessidade do desenvolvimento colaborativo de ontologias e regras.

O SWRL Editor foi integrado ao Web-Protégé e está dividido em Servidor e Cliente. O Servidor centraliza as operações com as regras, por exemplo, ontologias e regras são carregadas de seus arquivos apenas uma vez e ficam disponíveis para todos os clientes através de chamadas RPC. Além disso, a centralização das operações é um ponto chave para que a ferramenta possa funcionar colaborativamente, já que o servidor atua como nó central de distribuição das alterações feitas por cada usuário.

Já no Cliente estão as interfaces gráficas (as quais que são priorizadas nessa seção) são divididas em duas partes: Visualização (Seção 1.4.1), Filtros (Seção 1.4.2), Opções (Seção 1.4.3) e Composição (Seção 1.4.4). Para acessar o cliente do SWRL Editor é necessário escolher uma das ontologias. Na Figura 1.3 é mostrado o Web-Protégé após se ter acessado a Ontologia da Família. Nessa figura é possível ver a guia SWRL Editor (A), onde se encontra a nova ferramenta.

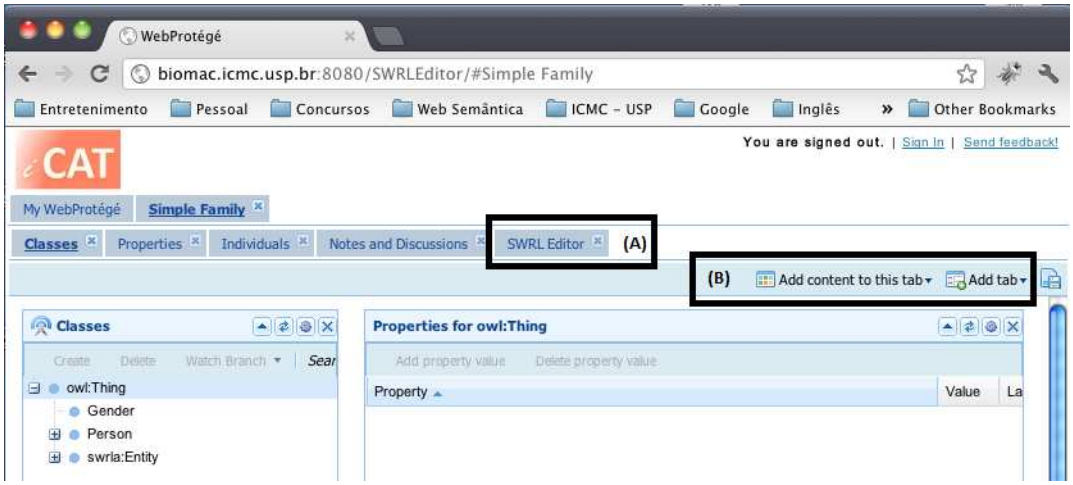

Figura 1.3 – Tela após o acesso a ontologia da família: (A) Acesso ao plug-in; (B) Acesso a todos os plug-ins não carregados automaticamente.

O Web-Protégé, após acessar a ontologia, carrega automaticamente os principais plug-ins (guias). Como não existe no Web-Protégé um plug-in para manipulação de SWRL, foi definido que o SWRL Editor é carregado automaticamente. Porém, caso seja fechado o plug-in, é possível acessá-lo novamente usando os botões Figura 1.3 (B).

1.4.1. Visualização

A página inicial do SWRL Editor é a *Visualizations* (Figura 1.4). Essa página é dividida em:

- Barra de Ferramentas (A) – Da esquerda para a direita: *Options* é o acesso as configurações do SWRL Editor; *Info* é o acesso as informações gerais do conjunto de regras; *New Rule* acessa a tela de composição para inserção de uma nova regra; *Run Rules* executa as regras no servidor. O botão Edit em *Rule Filter* acessa a tela para modificar o filtro das regras exibidas.
- Formas de Visualização dos Conjuntos de Regras (B) – Da esquerda para a direita: A guia *List* mostra as regras em uma visualização hierárquica (Seção 1.4.1.2). A guia *Text* mostra as regras usando Parafraseamento (Seção 1.4.1.3).

A guia SWRL mostra as regras usando SWRL com *Highlight* (1.4.1.1). A guia *Autism* mostra uma representação visual especial só para um conjunto especifico de regras para classificação de fenótipos de autismo [Silva 2012] e que não será abordada nesse capítulo. A guia *Groups* dispõe das técnicas para agrupamento (Seção 1.4.1.4). A guia *Decision Tree* contém as técnicas para transformar os conjuntos de regras para um formato de árvore (Seção 1.4.1.5).

- Botões para cada regra (C) – Para cada regra, nas guias *List*, *Text*, *SWRL* e *Autism*, existem botões para manipulação. Da esquerda para a direita são eles: O botão *Edit Rule* que coloca a regra em modo de edição. O botão *Duplicate and Edit* que faz uma cópia da regra e coloca a essa copia em modo de edição. O botão *Delete* excluí uma regra do conjunto. O botão *Similar Rules* mostra uma lista de regras similares usando um dos algoritmos de agrupamento.
- Barra de Status (D) – Serve como status do sistema, sendo utilizada para mostrar o numero de regras filtradas/número total de regras do conjunto.

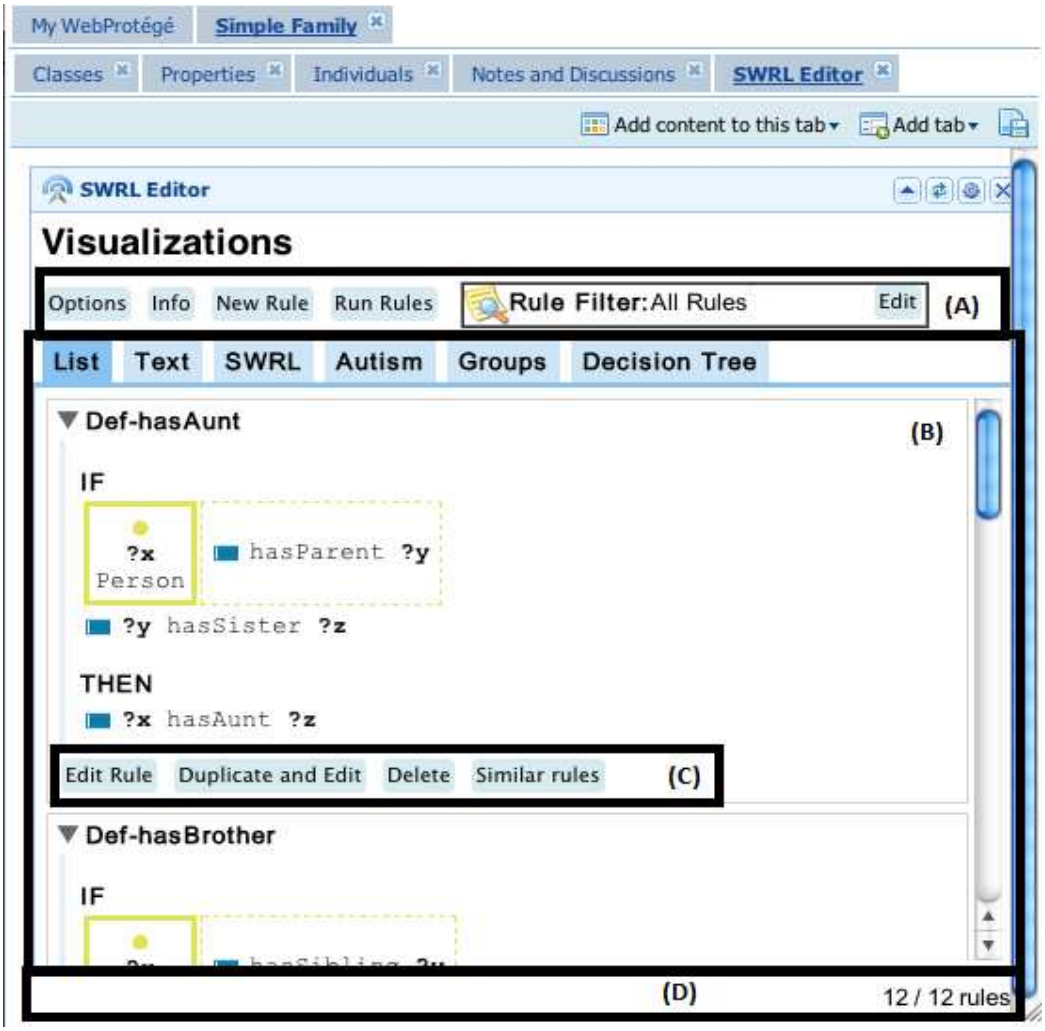

Figura 1.4 – SWRL Editor – Visualização: (A) Barra de Ferramentas; (B) Formas de Visualização dos Conjuntos de Regras; (C) Botões para cada regra; (D) Barra de Status do plug-in.

1.4.1.1. SWRL com Highlight

A representação visual *SWRL* exibe a regra sem ocultar os detalhes da sintaxe, contudo, agrega cores aos átomos e argumentos semelhantemente a recursos encontrados em alguns IDEs. Ao invés de utilizar a quebra automática no final da linha, cada átomo é apresentado em uma linha e o símbolo de implicação →, que divide o antecedente do consequente, é disposto em uma linha separada, o que torna a regra mais legível. Na Figura 1.5, a regra **def_hasAunt** é apresentada no formato original (A) e na representação com *highlight* (B).

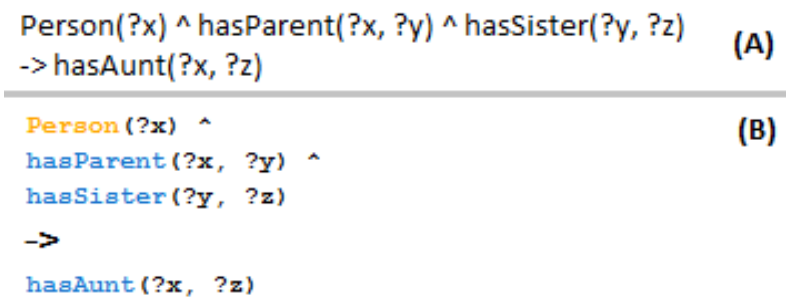

Figura 1.5 – Comparativo entre (A) uma regra apresentada no formato original e (B) a representação visual com *Highlight*.

Na Tabela 1.4 são apresentadas as cores utilizadas na geração dos *highlights* e seus significados. A representação visual destaca os tipos dos átomos, variáveis e valores literais sem sobrecarregar a apresentação da regra. Algumas das cores utilizadas foram extraídas dos símbolos utilizados pelo editor de ontologias Protégé, enquanto outras foram extraídas do editor Eclipse. O uso de uma fonte de tamanho fixo (monoespaçada) foi adotado seguindo o padrão encontrado nos editores de texto usados na programação, o que torna a leitura da regra mais organizada.

Tabela 1.4 – Cores utilizadas na representação *SWRL Highlight*

Cor	Exemplo	Descrição
Laranja	Person_Female	Representa predicados do tipo *Class*[4]
Azul	has_natural_father	Representa predicados do tipo *Object property*
Verde	has_current_age	Representa predicados do tipo *Datavalued property*
Vermelho	sqwrl:avg	Representa predicados do tipo *Built-in*
Cinza	differentFrom	Representa predicados do tipo *Same/different*
Preto em negrito	**?a**	Representa as variáveis
Roxo	32; "teste"	Representa os valores literais (Strings e números)

[4] Embora o Protégé utilize amarelo no símbolo que representa as classes, foi adotado para esta visualização o tom mais alaranjado pelo fato da cor amarela apresentar pouco contraste com fundos claros.

1.4.1.2. Visualização Hierárquica

A representação denominada visualização hierárquica busca abstrair a sintaxe da regra apresentando os átomos agrupados hierarquicamente a partir dos átomos do tipo *Class*. Assim, ao mesmo tempo em que torna a regra mais clara para apresentação, enriquece com cores e símbolos o seu significado. Os símbolos empregados na ferramenta Protégé foram reutilizados, contudo, para representar os demais elementos (*Same/Different* e *Built-ins*) foram empregados retângulos com as cores utilizadas na representação com *highlight* (Seção 1.4.1.1). Na Tabela 1.5 são apresentados exemplos dos símbolos utilizados na visualização.

Tabela 1.5 – Cores utilizadas na representação *SWRL Highlight*

Tipo	Símbolos e Cores	Exemplo
Class	Circulo amarelo	?b Person_Female
Object property	Retângulo azul	has_natural_father(?a,?f)
Datavalued property	Retângulo verde	has_current_age(?a,?b)
Built-in	Retângulo vermelho	?c > 18
Same/different	Retângulo cinza	?m same ?n ?m different ?n

O resultado dessa visualização pode ser visto na Figura 1.6, onde é apresentada a mesma regra (**def_hasAunt**) da representação com *highlight*. Embora esta representação oculte alguns detalhes da sintaxe SWRL e exiba os átomos organizadamente, seu uso se torna apropriado a desenvolvedores não técnicos em computação, apenas quando são utilizados na ontologia nomes apropriados nos labels (rdfs: Label) dos termos.

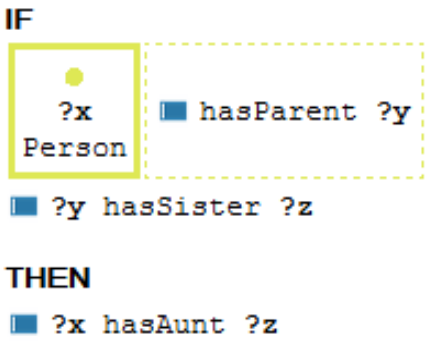

Figura 1.6 – Representação Hierárquica.

1.4.1.3. Parafraseamento

Essa representação fornece uma explicação em forma de texto da regra. É uma re-implementação de uma técnica do Axiomé [Hassanpour 2010] para gerar paráfrases em inglês. Apenas foi implementada uma mudança para essa técnica: a utilização de negrito em termos de ligação (IF, IS, AN, ...) entre os átomos de uma regra. A mudança é simples, porém torna mais fácil a leitura (Figura 1.7). Além disso, o SWRL Editor oferece a possibilidade de entrar em modo de edição a partir da representação em parafraseamento. Não edita o parafraseamento, mas acessa a edição desta regra.

```
IF
        "y" IS A Person
        AND "y" HAS Child "x"
        AND "y" HAS Child "z"

THEN
        "x" HAS Sibling "z"
```

Figura 1.7 – Parafraseamento.

1.4.1.4. Agrupamento

O agrupamento tem o objetivo de facilitar a visualização de grandes conjuntos de regras. Essa técnica facilita a localização de regras pelo usuário dentro de um conjunto de regras. O agrupamento é feito por meio de algoritmos que implementam interfaces Java e são carregados automaticamente no SWRL Editor. Atualmente, o SWRL Editor possui 3 algoritmos de agrupamento:

- *K-means* – Similaridade de átomos: Implementa o algoritmo de *K-means* considerando a similaridade dos átomos de cada regra [Silva 2012];
- *K-means* – Similaridade de predicados: Implementa o algoritmo de *K-means* considerando a similaridade dos predicados de cada átomo [Silva 2012];
- Estrutura da regra: Separa as regras em grupos por estrutura sintática das regras [Orlando 2012];

Além disso, o SWRL Editor possui sistemas de carregamento automático de algoritmos para o agrupamento. Esse sistema carrega os algoritmos de forma automática usando a classe Java: ServiceLoader. Para o ServiceLoader carregar os algoritmos é necessário que esses sejam implementados usando as interfaces Java que o SWRL Editor disponibiliza para agrupamento ou para árvore de decisão. Mais detalhes de como desenvolver novos algoritmos para o SWRL Editor estão disponíveis em [Orlando 2012].

O sistema de carregamento automático de algoritmos tornou-se uma ótima estratégia para usuários mais avançados que tenham interesse em desenvolver seus próprios algoritmos. Ele facilita essa operação já que para inserir um novo algoritmo é apenas necessário gerar um arquivo JAR com as implementações e o colocá-lo numa pasta do servidor, sem ser necessário alterar o código do SWRL Editor.

1.4.1.5. Árvore de Decisão

Da mesma forma que no agrupamento, árvores de decisão usam o sistema de carregamento automático de algoritmos e podem ser adicionados novos algoritmos. Atualmente, o SWRL Editor conta com 3 implementações de árvore de decisão:

- Ocorrência de átomos (Figura 1.8) – Essa técnica apenas cria uma árvore com base em um ranking de ocorrência de átomos, ou seja, átomos que mais ocorrem tendem a estar mais perto da raiz.
- Dependência de variáveis (Figura 1.9) – Essa técnica cria as árvores levando em conta a dependência das variáveis.

- Paráfrases (Figura 1.10) – Divide o parafraseamento (Seção 1.4.1.3) em pequenas paráfrases e com isso cria uma árvore mantendo a ordem original da paráfrase.

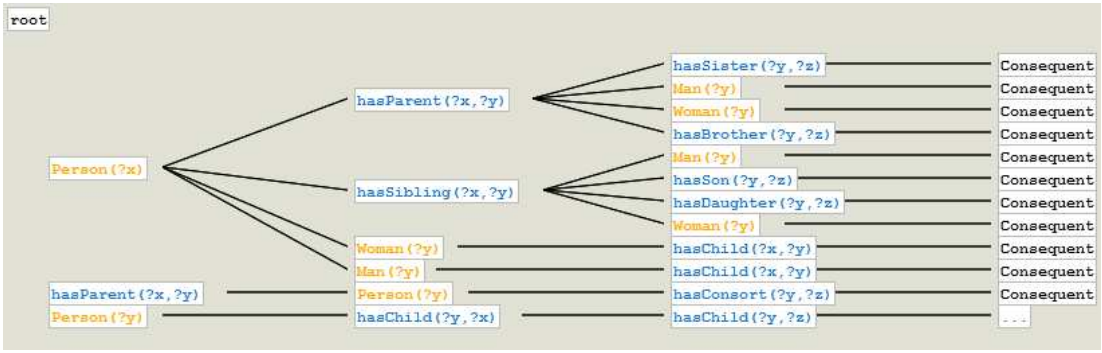

Figura 1.8 – Árvore de decisão: Ocorrência de átomos.

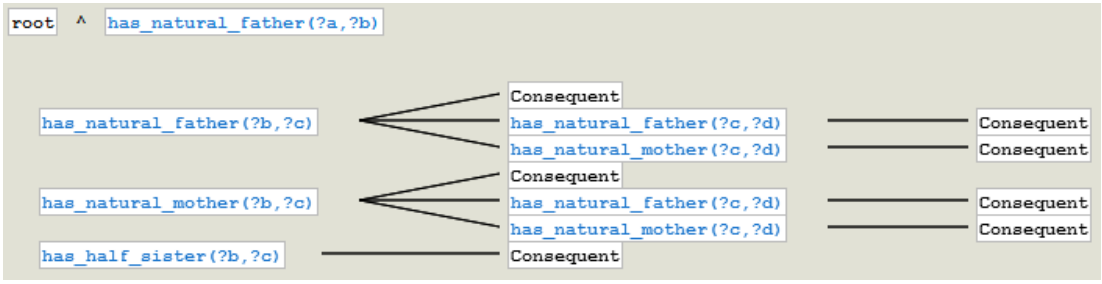

Figura 1.9 – Árvore de decisão: Dependência de variáveis.

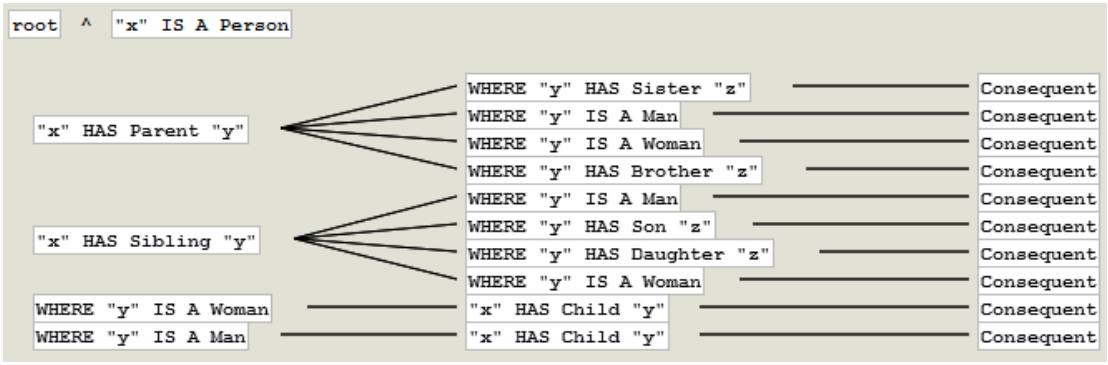

Figura 1.10 – Árvore de decisão: Paráfrases.

A funcionalidade das árvores de decisão é uma das principais contribuições do SWRL Editor, pois essa técnica nunca foi usada para regras SWRL e mostra-se como uma forma fácil de navegar em conjuntos de regras. Através dela é possível encontrar facilmente regras com antecedentes iguais ou semelhantes. Na Figura 1.11 é possível identificar que ocorrem antecedentes com mais de um consequente. De cima para baixo, na primeira ocorrência de um antecedente com dois consequentes, ao passar o mouse sobre os dois consequentes é possível perceber que eles são diferentes. Sendo assim, pode ser que não seja um erro, mas uma opção de projeto. Já na segunda ocorrência, quando são olhados os consequentes é possível identificar que eles são iguais (um erro). Como os algoritmos de árvore de decisão juntam átomos semelhantes é possível identificar os tipos de repetição que merecem atenção.

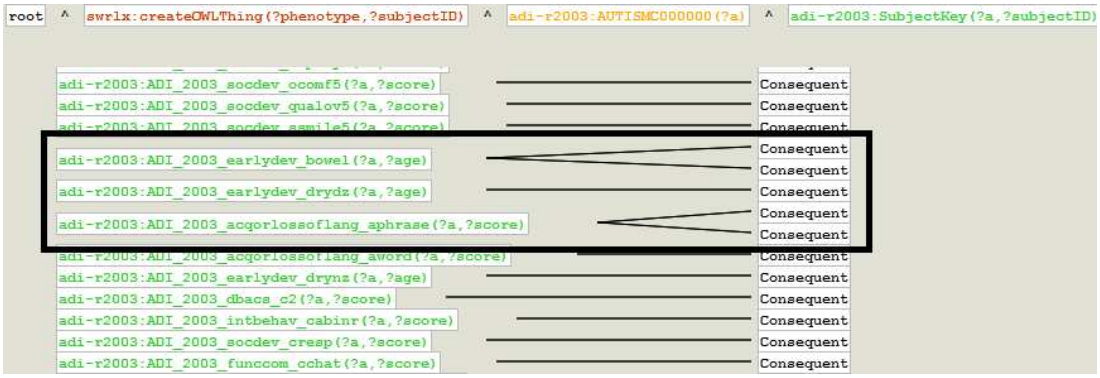

Figura 1.11 – Árvore de decisão: Identificação de regras com antecedentes iguais.

Muitas regras se diferenciam às vezes por um ou dois átomos. Então como ocorrem muitas repetições nos átomos das regras, a árvore de decisão pode ser usada para fazer o reaproveitamento de átomos. A Figura 1.12 mostra a tela da árvore de decisão, como pode ser visto na imagem, ao se clicar sobre um nodo da árvore são mostradas opções em um menu popup. Quando o nodo for um átomo do antecedente, o menu irá mostrar a opção para reaproveitar todos os átomos até a raiz, em uma nova regra. Já quando o nodo for um consequente, o menu mostrará a opção de editar essa regra.

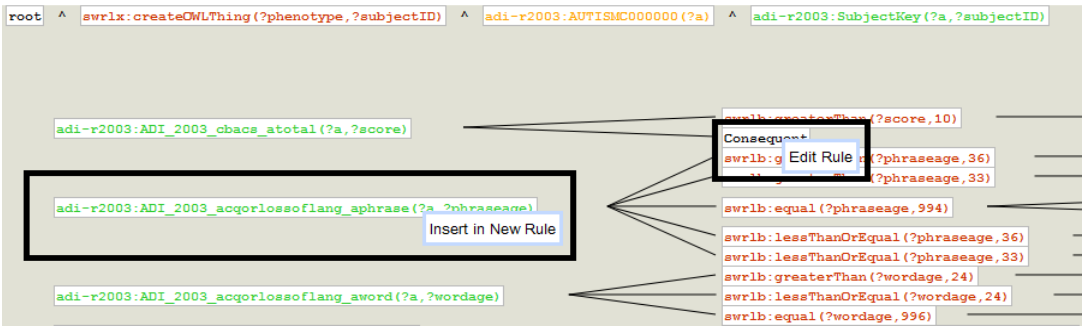

Figura 1.12 – Árvore de decisão: Botão direito sobre os nós.

1.4.2. Filtros

Como conjuntos de regras tendem a crescer, torna-se cada vez mais difícil localizar regras, então o SWRL Editor possui uma interface para filtrar regras SWRL. Nessa interface, o usuário pode usar os operadores lógicos AND, OR e NOT para criar uma expressão lógica para filtrar regras. O filtro é montado usando a interface do usuário na Figura 1.13, em que há um campo para cada operador lógico. Nesses campos, os valores de filtro são separados por espaços, expressões com espaços devem ser delimitadas por "" (exemplo "casa e carro").

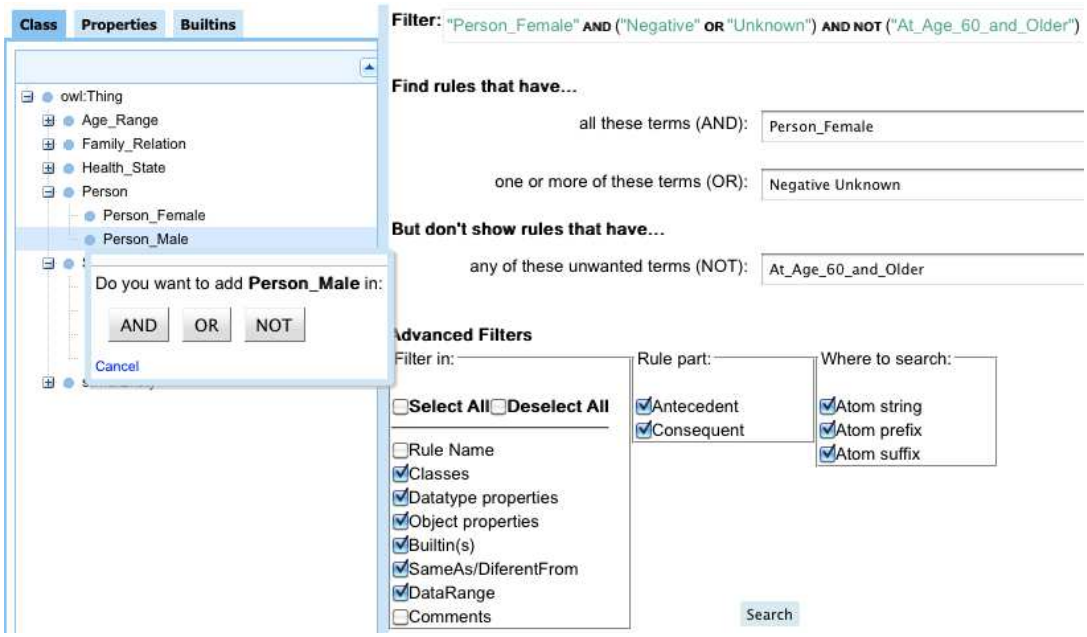

Figura 1.13 – Filtro das regras.

O filtro utiliza a expressão lógica montada (Figura 1.13, parte superior) para procurar automaticamente no rdf:ID e no rdfs:Label dos predicados de cada átomo. Além disso, a interface fornece opções avançadas para filtrar, sendo possível escolher partes de uma regra e tipos de átomos a serem filtrados. É possível escolher filtrar no Nome da regras, em *Class*, *Datatype properties*, *Object properties*, *Buitins*, *SameAs/DiferentFrom*, *DataRange*, *Comments* (Comentários dos predicados de cada átomo). Também é possível filtrar somente no antecedente ou no consequente. As opções para a filtragem ainda permitem os usuários realizem buscas nos prefixos, sufixos ou qualquer parte da string que representa cada átomo.

Na Figura 1.13, na lateral esquerda, encontram-se três guias (*Class*, *Properties* e *Bultins*) com os termos da ontologia. Essas guias são usadas para facilitar a inserção de termos da ontologia em uma busca. Como pode ser visto nessa figura, assim que o usuário clica na classe Person_Male a interface oferece a opção para escolher a qual operador lógico quer inserir.

Quando o usuário termina de montar o filtro, basta clicar no botão *Search* (localizado no lado direito parte inferior) que o filtro será executado e o seu resultado será mostrado na tela de visualização do SWRL Editor. Na Figura 1.14 é mostrada a tela de visualização com o filtro de regras modificado. Apenas as regras que obedecerem as condições desse filtro serão mostradas na interface e poderão ser usadas nas outras ferramentas.

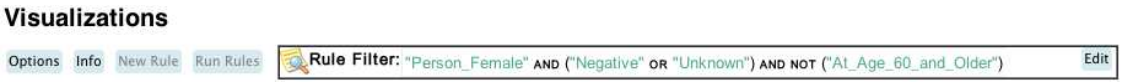

Figura 1.14 – Resultado do filtro de regras.

1.4.3. Opções

As configurações do SWRL Editor servem para tornar a ferramenta mais amigável e facilitar os processos de visualização e edição de regras. As configurações do SWRL

Editor foram divididas em três partes (Figura 1.15): *General*, *Composition* e *Visualization*. Só serão detalhadas as configurações gerais, as demais configurações são apenas detalhes de interface, já as configurações gerais servem para toda a ferramenta.

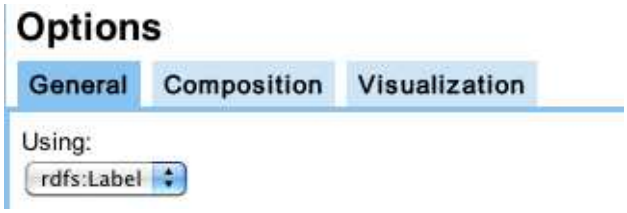

Figura 1.15 – Opções Gerais do SWRL Editor.

Na Figura 1.15 é apresentada a tela das configurações gerais, nela somente existe uma configuração para usar rdf:ID ou rdfs:Label. O rdf:ID é um identificador único para cada termo da ontologia, já o rdfs:Label é uma descrição para o termo da ontologia. Usando a ontologia do autismo [Hassanpour 2011], na Tabela 1.6 é mostrado alguns rdf:ID de termos da ontologia com o seu respectivo rdfs:Label. Fica evidente que os rdfs:Labels podem tornar mais fácil o entendimento e a edição das regras.

Tabela 1.6 – rdf:ID versus rdfs:Label

rdf:ID	rdfs:Label
adi-r2003:AUTISMC000000	*Autism Diagnostic Interview-Revised*
ados1:AUTISMC000004	*Autism Diagnostic Observation Schedule*
ados4:AUTISMC000001	*Autism Diagnostic Observation Schedule*
vabs_survey:AUTISMC000003	*Vineland Adaptive Behavior Scales*
Autism-core:AUTISMC100000	*Phenotype Record*
Autism-core:AUTISMC100002	*Quantitative value*
NIF-Invertigation:birnlex_2387	*Citation Record*
Autism-core:AUTISMC1000069	*Present*
Autism-core:AUTISMC1000070	*Repetitive stereotypical behavior*

Essa é uma importante técnica, pois permite ao usuário escolher a opção de visualizar e editar as regras usando rdfs:Labels. Porém, isso pode acarretar problemas na edição, pois um mesmo rdfs:Label pode estar em mais de um termo da ontologia. Um exemplo está na Tabela 1.6 em que os termos ados1:AUTISMC000004 e ados4:AUTISMC000001 compartilham um mesmo rdfs:Label. A ferramenta trata esse tipo de redundância, pois mesmo com a opção para usar rdfs:Label selecionada, o usuário poderá usar um rdf:ID durante a edição.

1.4.4. Composição

O processo de criação de regras é ativado no SWRL Editor quando o usuário seleciona a opção “New Rule” na visualização ou opta por editar uma regra específica. Na Figura 1.16 é apresentada a tela da ferramenta SWRL Editor no modo composição. Na lateral esquerda (A) dessa figura são exibidos os termos disponíveis na ontologia que podem ser selecionados pelos usuários (da mesma forma que nas Opções Seção 1.4.2). Quando

um deles é selecionado, uma janela é aberta com as opções de adicioná-lo como um átomo no antecedente ou consequente.

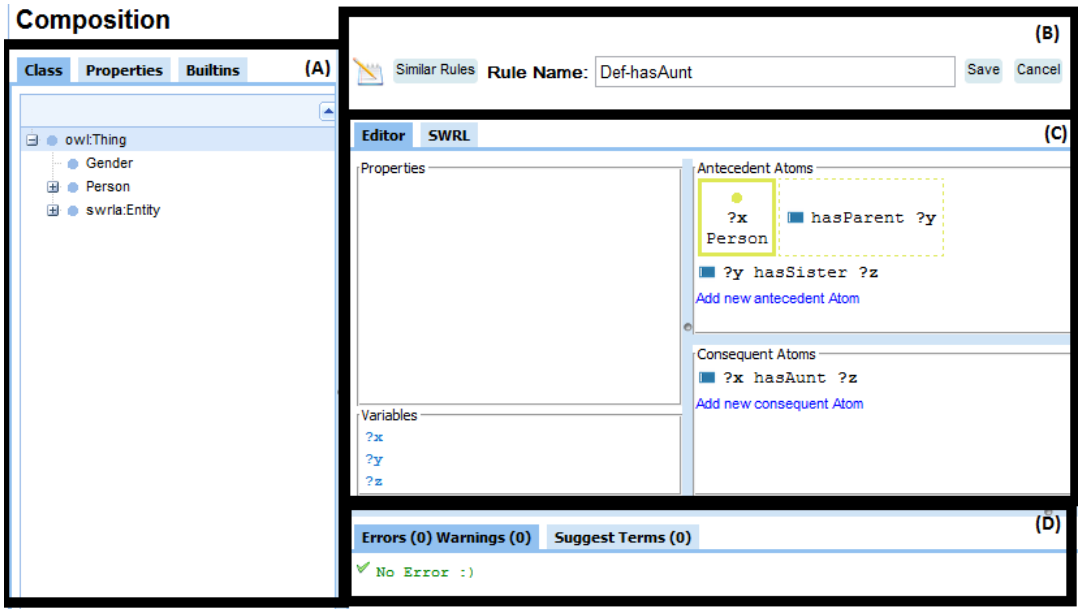

Figura 1.16 – SWRL Editor – Composição: (A) Termos disponíveis na ontologia; (B) Nome da regra e opção de salvar; (C) Editores da regra; (D) Avisos e sugestões de regras.

Na Figura 1.16 (B), é possível definir ou alterar o nome da regra. O botão salvar (*Save*) fica desativado enquanto são identificados erros na regra em desenvolvimento. Além disso, quando o desenvolvedor opta por voltar ao modo de visualização sem salvar (*Cancel*) a regra, a ferramenta exibe uma mensagem de confirmação. Ainda nessa figura em (C) é possível ver os dois modos de edição dos átomos de uma regra:

- *Editor* (ativo na Figura 1.16) – Formulário que utiliza a representação hierárquica para apresentar e organizar os átomos da regra. Divide o ambiente de desenvolvimento nas seguintes janelas de trabalho:
    - Caixa de propriedades (*Properties*), contendo as propriedades do átomo que o desenvolvedor selecionar;
    - Lista de variáveis (*Variables*) utilizadas na regra;
    - Átomos do antecedente (*Antecedent Atoms*), contendo os átomos presentes na parte antecedente da regra. Os átomos são apresentados conforme a representação hierárquica e podem ser alterados na caixa de propriedades;
    - Átomos do consequente (*Consequent Atoms*), contendo os átomos presentes na parte consequente da regra. Os átomos são apresentados conforme a representação hierárquica e podem ser alterados na caixa de propriedades;
- SWRL – Editor SWRL com *Highlight*, no qual o desenvolvedor escreve a regra utilizando a sintaxe SWRL. Os átomos e argumentos são apresentados conforme o padrão de cores adotado na visualização do SWRL com *Highlight*;

Na parte inferior da Figura 1.16 (D), são apresentadas as seguintes abas:

- Problemas (*Errors* e *Warnings*) – contem as possíveis notificações de erros e avisos para o desenvolvedor da regra;
- Termos sugeridos (*Suggest Terms*) – conterá uma lista com os termos sugeridos;

Um ponto importante, já citado, é o da ferramenta permitir que o usuário utilize os rdfs:Labels (mais fácies que os identificadores únicos rdf:ID). Porém para que não ocorram redundâncias são apresentados erros como o da Figura 1.17. Nesta Figura é mostrada a seguinte mensagem de erro: "The predicate label **Autism Diagnostic Observation Schedule** represents more than one ID: **ados4:AUTISMC000001**, **ados1:AUTISMC000004**". O usuário poderá então usar um dos dois rdf:IDs para representar o elemento e assim evitar o erro.

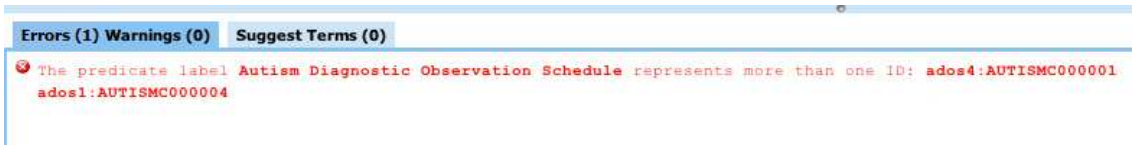

Figura 1.17 – Mensagem de erro de redundância de rdfs:Labels.

1.5. Construção de uma ontologia com regras SWRL

Essa seção descreve a criação de uma ontologia de relações familiares. A idéia de utilizar esse exemplo reside no fato de que esse domínio é de fácil entendimento por todos e pode gerar bons exemplos. Vamos dividir essa demonstração em duas etapas:

- Ontologia – Definir os termos da ontologia e suas relações:
  - Classes (Conceitos) O objetivo criar classes e subclasses, modificando a hierarquia de classes da ontologia;
  - Propriedades (Relações) As propriedades representam relacionamentos entre dois indivíduos;
- SWRL:
  - SWRL Regras de inferências;
  - Indivíduos da ontologia representam objetos no domínio de interesse (ou indivíduos de uma classe);
  - Execução – Demonstração dos resultados gerados após a execução de um conjunto de regras;

1.5.1. Construção da ontologia

Toda ontologia contém uma classe chamada owl:Thing. Conforme mencionado, as classes OWL são interpretadas como conjuntos de indivíduos (ou conjunto de objetos). A classe owl:Thing é a classe que representa o conjunto que contém todas as classes e os indivíduos, uma vez que todas as classes são subclasses de owl:Thing. Para exemplificar será apresentada nessa seção a criação de uma ontologia que representem relações parentesco entre indivíduos.

Na Figura 1.18 são definas as subclasses de owl:Thing que farão parte da ontologia de exemplo. Já nas Figuras 1.19 e 1.20 são definidas as outras classes que são subclasses de Man e Woman respectivamente.

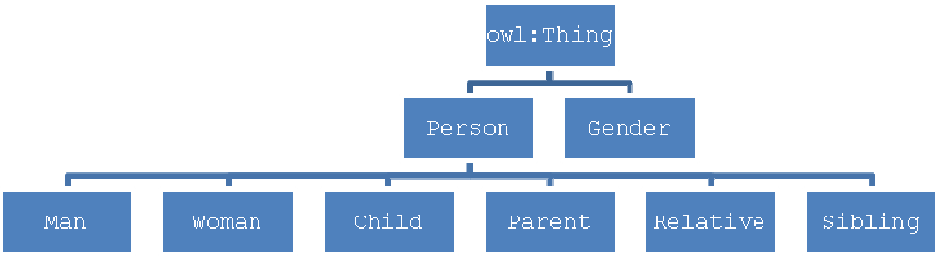

Figura 1.18 – Definição da ontologia: owl:Thing.

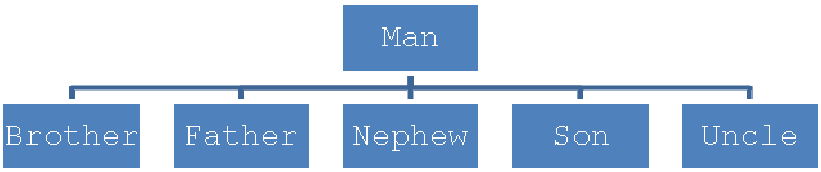

Figura 1.19 – Definição da ontologia: Man.

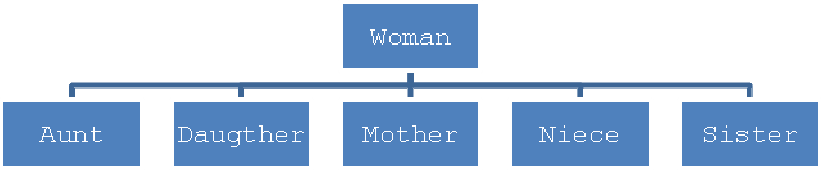

Figura 1.20 – Definição da ontologia: Woman.

Como apresentado anteriormente, esses são identificadores únicos (rdf:ID) da ontologia, ou seja, só podem existir uma vez. Porém muitas vezes, quando não existe um consenso sobre o nome ideal de um termo, é adotado um código para ser o rdf:ID e são usados rdfs:Labels para nome. Outra boa prática é usar o termo em inglês (ex: Person) e colocar rdfs:Labels para outras linguagens (seguindo o ex: Pessoa).

Com todos os termos definidos para a ontologia vamos demonstrar a criação de uma classe no Web-Protégé. Para pode editar é necessário estar logado no Web-Protégé. O Web-Protégé disponibiliza uma guia chamada “Classes” que edita as classes da ontologia. Essa guia possui por default 3 parte:

- *Classes* (Figura 1.21A): Usada exibir a hierarquia de classes;
- *Properties* (Figura 1.21B): Exibe as propriedades de uma classe;
- *Asserted Conditions* (Figura 1.21 C): Contém as restrições de uma classe;

Na Figura 1.21, para criar a classe Person é necessário selecionar a classe owl:Thing e clicar em *Create* (Figura 1.21A), irá aparecer um popup para informar o nome da nova classe. Após informar basta clicar em OK e a classe já fica disponível na hierarquia. Ao selecionar a nova classe, ela não possuirá nenhuma propriedade e possuirá uma restrição (a relação com owl:Thing). Ainda na Figura 1.21(B) foram criadas 3 propriedades de anotação para essa nova classe:

- Dois rdfs:Labels: Não tem limite de *labels*, no exemplo foi disponibilizada a tradução do termo em português e espanhol;
- Um rdfs:Comment: os comentário são interessantes para ter uma explicação do termo da ontologia;

Já nas restrições Figura 1.21(C) existem somente dois tipos que podem ser definidos:

- *Necessary*: Essa restrição pode ser lida assim: “se alguma coisa é um membro da classe então é necessário que preencha essas condições”. Na Figura 1.21 (C) é informado que para pertencer a Person o individuo deve estar em owl:Thing.
- *Necessary & Sufficient*: Essa restrição agrega mais esse fato: “se alguma coisa preenche essas condições então deve ser um membro dessa classe”. Ex: Qualquer individuo que pertença a Man ou Womem é uma Person.

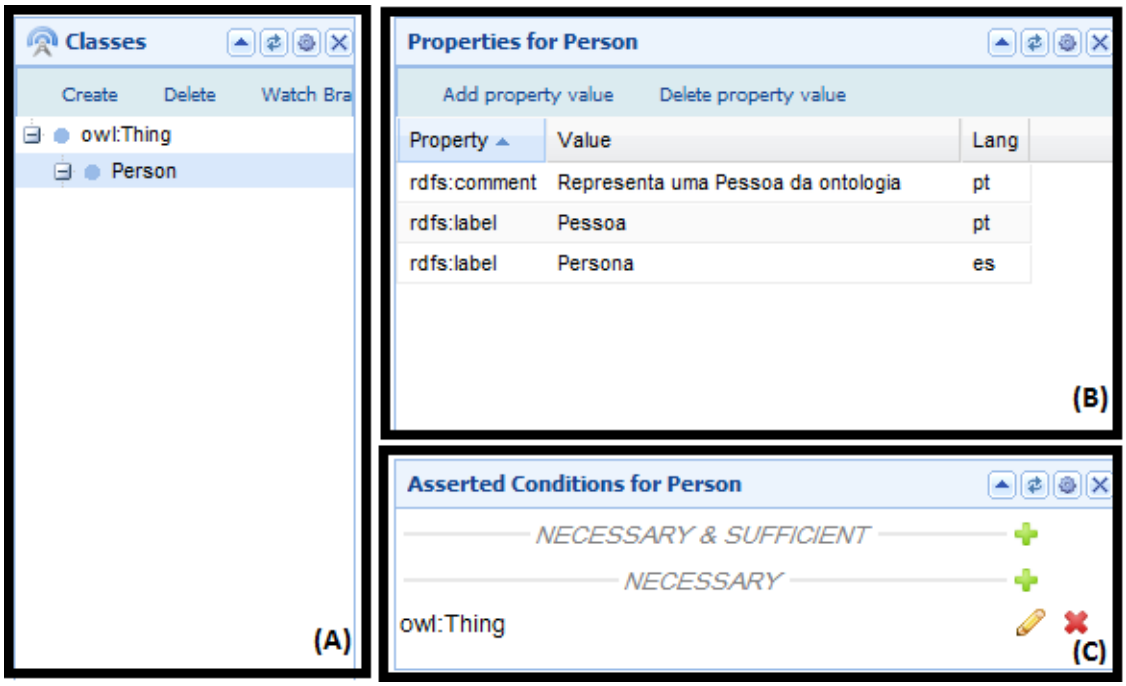

Figura 1.21 – Web-Protégé Interface para edição de classes: (A) Hierarquia de Classes; (B) Propriedades de uma classe; (C) Restrições de uma classe.

Se analisarem as Figuras 1.18, 1.19 e 1.20 pode-se perceber que, por exemplo, a classe Son criada para Man, também pertence a classe Child, porém como são identificadores únicos não podem ser inseridos duas vezes na hierarquia de classes. Então a solução é inserir uma restrição em *Necessary & Sufficient* para esses termos. No caso de Son ele já possui essa restrição para Man, então apenas basta inserir para Child também. Na tabela abaixo são apresentados os termos que necessitam de restrições:

Tabela 1.7 – Lista de restrições para estabelecer a hierarquia de classes

Classe	Tipo de Restrição	Restrição
Brother	*Necessary & Sufficient*	Sibling
Father	*Necessary & Sufficient*	Parent
Nephew	*Necessary*	Relative
Son	*Necessary & Sufficient*	Child
Uncle	*Necessary*	Relative
Aunt	*Necessary*	Relative

Classe	Tipo de Restrição	Restrição
Daugther	*Necessary & Sufficient*	Child
Mother	*Necessary & Sufficient*	Parent
Niece	*Necessary*	Relative
Sister	*Necessary & Sufficient*	Sibling

Com isso é possível montar toda a hierarquia de classes chegando ao que é apresentado na Figura 1.22.

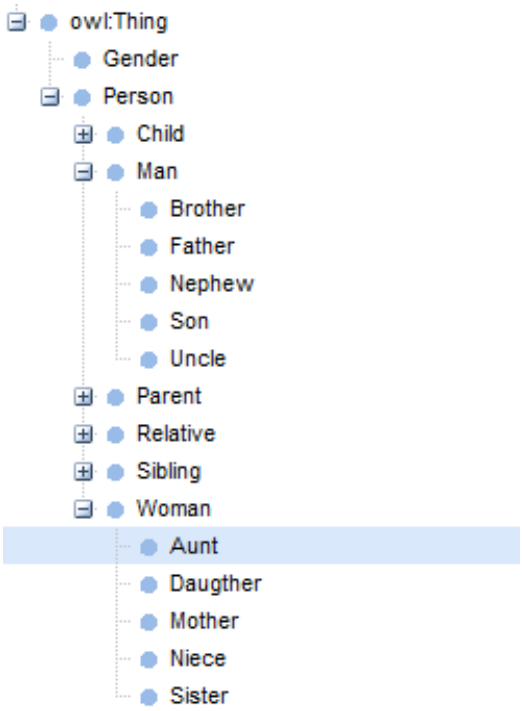

Figura 1.22 – Hierarquia Inicial de Classes.

O ideal agora é preencher todos os rdfs:Label da ontologia e disponibilizar, por exemplo, os termos da ontologia em português, porém não será apresentado neste capítulo, por não ser obrigatório. Antes de preencher as demais restrições é necessário criar as *Properties*. O Web-Protégé disponibiliza uma guia chamada “Properties” (Figura 1.23). Por default as *Properties* precisam de duas propriedades de anotação preenchidas:

- rdfs:domain: Na Figura 1.23 está selecionada a propriedade hasAunt, o domínio dessa propriedade é qualquer pessoa (Person), ou seja, qualquer pessoa pode ter uma tia;
- rdfs:range: O range já é quem pode ser classificada para aquela classe. Ex Figura 1.23: só pode ser uma tia quem for da classe Woman.

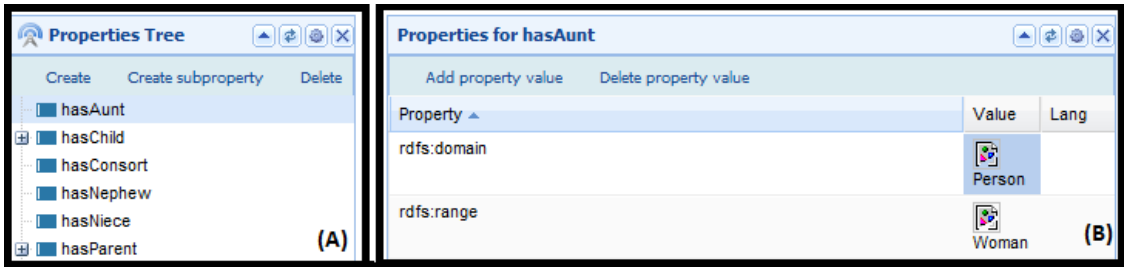

Figura 1.23 – Web-Protégé Interface para edição de propriedades: (A) Hierarquia de propriedades; (B) Propriedades de uma propriedade.

Na tabela abaixo seguem todas as propriedades da ontologia com os seus respectivos rdfs:domain e rdfs:range. Para inserir uma propriedade na Ontologia basta clicar em *Create* em (Figura 1.23 – A) e irá aparecer um popup para informar o nome da nova propriedade. Após informar basta clicar em OK e a propriedade já fica disponível. Caso queira inserir uma subpropriedade basta selecionar a propriedade pai e clicar em *Create subproperty*.

Tabela 1.8 – Lista de propriedades da ontologia

Propriedade pai	Propriedade a criar	rdfs:domain	rdfs:range
-	hasAunt	Person	Woman
-	hasChild	Person	Person
hasChild	hasDaughter	Person	Woman
hasChild	hasSon	Person	Man
-	hasConsort	Person	Person
-	hasNephew	Person	Man
-	hasNiece	Person	Woman
-	hasParent	Person	Person
hasParent	hasFather	Person	Man
hasParent	hasMother	Person	Woman
-	hasSex	Person	Gender
-	hasSibling	Person	Person
hasSibling	hasBrother	Person	Man
hasSibling	hasSister	Person	Woman
-	hasUncle	Person	Man

Ainda antes de inserir as demais restrições nas classes é necessário instanciar dois indivíduos (Female e Male) para classe Gender na guia "*Individuals*" do Web-Protégé (Figura 1.24). Para inserir um indivíduo na Ontologia é necessário primeiramente selecionar a classe referente (Figura 1.24 A). O próximo passo é clicar em *Create* em (Figura 1.24 – B) e irá aparecer um popup para informar o nome do novo indivíduo. Após informar basta clicar em OK e o indivíduo já fica disponível.

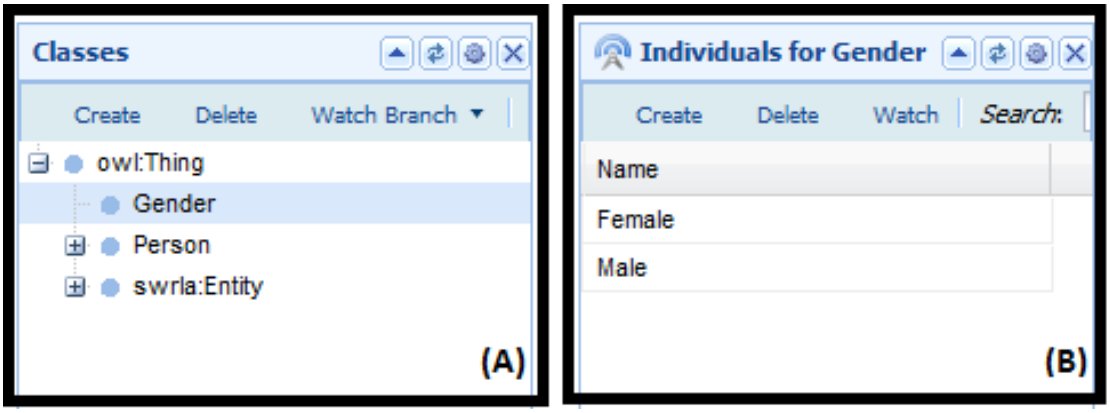

Figura 1.24 – Web-Protégé Interface para edição de indivíduos: (A) Hierarquia de classes; (B) Indivíduos de uma classe.

Após isso é possível inserir as restrições que envolvem propriedades ou indivíduos com em (Figura 1.21 C). Essas restrições podem ser vistas na Tabela abaixo.

Tabela 1.9 – Lista das demais restrições

Classe	Tipo de Restrição	Restrição
Gender	*Necessary & Sufficient*	{Female Male}
Person	*Necessary & Sufficient*	Man or Woman
Child	*Necessary & Sufficient*	Hasparent min 1
Man	*Necessary & Sufficient*	hasSex has Male
Nephew	*Necessary & Sufficient*	(hasUncle min 1) or (hasAunt min 1)
Uncle	*Necessary & Sufficient*	(hasNephew min 1) or (hasNiece min 1)
Parent	*Necessary & Sufficient*	hasChild min 1
Relative	*Necessary & Sufficient*	Child or Parent or Aunt or Nephew or Niece or Uncle or Sibling
Aunt	*Necessary & Sufficient*	(hasNephew min 1) or (hasNiece min 1)
Niece	*Necessary & Sufficient*	(hasUncle min 1) or (hasAunt min 1)
Sibling	*Necessary & Sufficient*	hasSibling min 1
Woman	*Necessary & Sufficient*	hasSex has Female

1.5.2. Construção e execução das regras SWRL

Nesta seção são apresentadas e explicadas 12 regras SWRL, as regras estão disponíveis nas Tabelas 1.10 e 1.11 abaixo:

Tabela 1.10 – Lista das demais restrições

Nº	Nome	Regra

Nº	Nome	Regra
1	Def-hasAunt	Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
2	Def-hasBrother	Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
3	Def-hasDaughter	Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
4	Def-hasFather	Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
5	Def-hasMother	Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
6	Def-hasNephew	Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
7	Def-hasNiece	Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
8	Def-hasParent	Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
9	Def-hasSibling	Person(?y) ∧ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)

Nº	Nome	Regra
10	Def-hasSister	Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
11	Def-hasSon	Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
12	Def-hasUncle	Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

Tabela 1.11 – Explicação das Regras em forma de texto

Nº	Explicação da Regra
1	Uma regra em que uma Pessoa X que tem um Pai/Mãe Y e que Y tem uma Irmã Z pode-se concluir que X tem uma Tia Z.
2	Uma regra em que uma Pessoa X que tem um Irmão/Irmã Y e Y seja Homem pode-se concluir que X tem um irmão Y.
3	Uma regra em que uma Pessoa X que tem um Filho/Filha Y e Y seja Mulher pode-se concluir que X tem uma filha Y.
4	Uma regra em que uma Pessoa X que tem um Pai/Mãe Y e Y seja homem pode-se concluir que X tem um Pai Y.
5	Uma regra em que uma Pessoa X que tem um Pai/Mãe Y e Y seja mulher pode-se concluir que X tem uma Mãe Y.
6	Uma regra em que uma Pessoa X que tem um Irmão/Irmã Y e que Y tem um filho Z pode-se concluir que X tem um sobrinho Z.
7	Uma regra em que uma Pessoa X que tem um Irmão/Irmã Y e que Y tem uma filha Z pode-se concluir que X tem uma sobrinha Z.
8	Uma regra em que uma Pessoa Y que tem um Cônjuge Z e que Y é Pai/Mãe de X pode-se concluir que Z também é Pai/Mãe X.
9	Uma regra em que uma Pessoa Y que tem um Filho/Filha X e que tem outro Filho/Filha Z pode-se concluir X tem um Irmão/Irmã Z.
10	Uma regra em que uma Pessoa X que tem um Irmão/Irmã Y e Y seja Mulher pode-se concluir que X tem uma irmã Y.
11	Uma regra em que uma Pessoa X que tem um Filho/Filha Y e Y seja Homem pode-se concluir que X tem um filho Y.
12	Uma regra em que uma Pessoa X que tem um Pai/Mãe Y e que Y tem um Irmão Z pode-se concluir que X tem um Tio Z.

Com as regras definidas, vamos criar e executar a inferência de uma regra. Para criar uma nova regra no SWRL Editor basta clicar no botão *New Rule* da tela de visualização (Figura 1.4). Após clicar, a ferramenta acessa a tela de composição (Figura 1.25). Para adicionar átomos a uma regra é possível a partir da navegação nas classes, propriedades e built-ins. Quando achar o termo, basta clicar sobre ele que o SWRL Editor exibirá um popup que serve para selecionar se o predicado será inserido no antecedente ou no consequente (Figura 1.25 A). Assim que for selecionada a opção do popup, o novo predicado é carregado para edição (Figura 1.25 B).

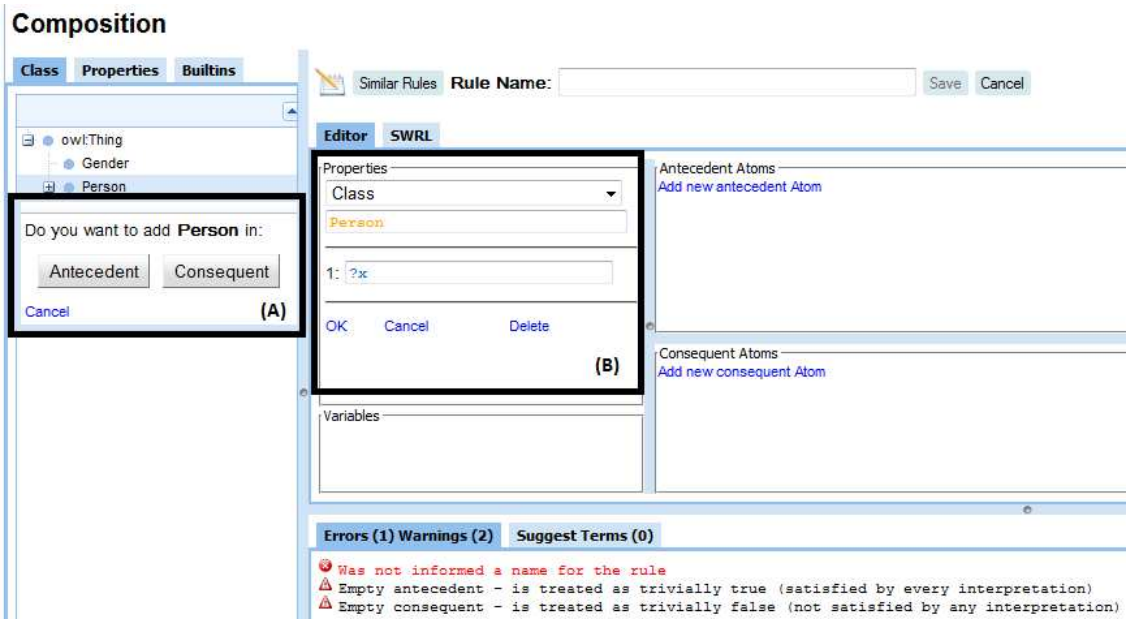

Figura 1.25 – SWRL Editor Tela de composição para nova regra: (A) Selecionar entre antecedente e consequente; (B) Átomo sendo editado.

Após informar o nome e todos os átomos da regra 1 se obtém a Figura 1.26. Na sequência é só necessário salvar a regra em "Save" e assim foi criada a primeira regra SWRL.

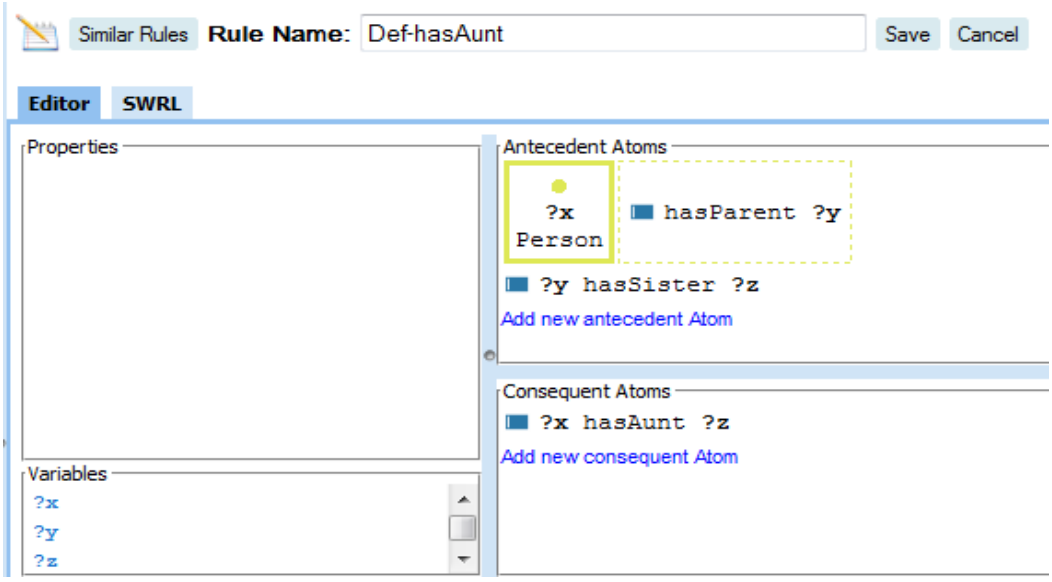

Figura 1.26 – SWRL Editor Tela de composição com uma regra preenchida.

A Regra 1 criada tem função de criar a relação de uma Pessoa ter uma Tia. Para testar regra 1 é necessário criar os seguintes indivíduos na ontologia:

- Criar o indivíduo "Filho" na classe Man.
    - Atribuir "Pai" a propriedade hasFather de "Filho";
- Criar o indivíduo "Pai" na classe Man.

    - Atribuir “Filho” a propriedade hasSon de “Pai”;
    - Atribuir “Tia” a propriedade hasSister de “Pai”;
- Criar o indivíduo “Tia” na classe Woman.
    - Atribuir “Pai” a propriedade hasBrother de “Tia”;

Agora ao executar as regras (*Run Rules* na Figura 1.4 B) será preenchida com “Tia” a propriedade hasAunt no “Filho”

1.6. Conclusão

A Web Semântica é uma maneira de explorar a associação de significados explícitos aos conteúdos de documentos presentes na Web, para que esses possam ser processados diretamente ou indiretamente por máquinas [Berners-Lee and Fischetti 2008]. Para possibilitar esse processamento, os computadores necessitam ter acesso a coleções estruturadas de informações (dados e metadados) e a conjuntos de regras de inferência sobre esses conteúdos (que ajudem no processo de dedução automática) para que seja possível o raciocínio automatizado sobre os mesmos [Berners-Lee et all 2001).

A linguagem recomendada pelo W3C para representação e compartilhamento de ontologias na Web Semântica é o OWL. Essa linguagem foi projetada para aplicações que necessitam processar o conteúdo da informação em vez de apenas apresentar informações em texto [McGuinness and Harmelen 2004]. Infelizmente a expressividade de OWL nem sempre é suficiente para modelar todos os tipos de problemas. Especialmente OWL não tem suporte a cláusulas no formato de Horn (se a → b). Para suprir essa deficiência, a SWRL (Semantic Web Rule Language) foi criada. A linguagem SWRL é muito útil para desenvolvedores de ontologias, pois faz inferências de novos conhecimentos sobre indivíduos da ontologia (OWL).

Este capítulo apresentou conceitos ligados a Web Semântica, especialmente os ligados a regras SWRL e o SWRL Editor, um editor de regras SWRL encontrado no Web-Protégé. Por último, foi mostrado um guia passo a passo para a criação de uma ontologia, com regras, que descreve relações de parentescos entre indivíduos.

1.7. Agradecimentos

Dilvan A. Moreira agradece o apoio recebido da Fapesp para apresentação do trabalho.

1.8. Referências

Ahmedi, L., Abazi-Bexheti, L. and Kadriu, A. (2011). “A Uniform Semantic Web Framework for Co-authorship Networks”. In: IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), p. 958-965. doi: 10.1109/DASC.2011.159.

Almeida, M. B. and Bax, M. P. (2003). Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. Ci. Inf., Brasília, v. 32, n. 3. doi: 10.1590/S0100-19652003000300001.

Barder, F. and Nutt, W. (2003). Basic Description Logics. In: BARDER, F. et al. The Description Logic Handbook. Cambridge University Press. Capítulo 2, p. 43-90.

Berners-Lee, T. and Fischetti, M. (2008). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. HarperSanFrancisco. ISBN: 978-0062515872.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. Scientific American, p. 34–43. Disponível em: http://www.scientificamerican.com/article.cfm?id=the-semantic-web, acesso em Jan. 2012.

Brickley, D., Guha, R. V. and Mcbride, B. (Editors). (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C. Disponível em: http://www.w3.org/TR/rdf-schema/, acesso em Jan. 2010.

Chandrasekaran, B., Josephson, J. R. and Benjamins, V. R. (1999). What Are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems, Piscataway, NJ, USA, v. 14, n. 1, p. 20-26. doi: 10.1109/5254.747902.

Corazzon, R. (2010). Theory and History of Ontology. A Resource Guide for Philosophers. Disponível em: http://www.formalontology.it/, acesso em Jan. 2010.

Diniz, V. and Cecconi, C. (2008). Padrões Web: Passado, presente e futuro, V Conferência Latino Americana de Software Livre. Disponível em: http://www.w3c.br/palestras/internet-web-jun-jul-2008/internetWeb_Out08.html, acesso em Dez. 2009.

Feigenbaum, L., Herman, I., Hongsermeier, R, Neumann, E. and Stephens, S. (2007). The Semantic Web in action, Scientific American, p. 64-71. Disponível em: http://www.ncbi.nlm.nih.gov/pubmed/18237102, acesso em Jan. 2010.

Fensel, D., Van Harmelen, F., Horrocks, I., Mcguinness, D. L. and Patel-Schneider, P. F. (2001). OIL: An Ontology Infrastructure for the Semantic Web. IEEE Intelligent Systems, v. 16, n. 2, p. 38-45. doi: 10.1109/5254.920598.

Golbreich, C., Horridge, M., Horrocks, I., Motik, B. and Shearer, R. (2007). OBO and OWL: leveraging semantic web technologies for the life sciences. In: Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference. Busan, Korea, p. 169-182.

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Scjmeoder, P. and Sattler, U. (2008). OWL 2: The next step for OWL. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Amsterdam, Netherlands, v. 6, n. 4, p. 309–322. doi: 10.1016/j.websem.2008.05.001.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowl. Acquis. London, UK, v. 5, n. 2, p. 199-220. doi: 10.1006/knac.1993.1008.

Guarino, N. and Giaretta, P. (1995). Ontologies and Knowledge Bases: Towards a Terminological Clarification. Journal Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. v. 1, n. 9, p. 25-32.

Hassanpour, S., O'connor, M. J. and Das, A. K. (2009). Exploration of SWRL Rule Bases through Visualization, Paraphrasing, and Categorization of Rules. In: Proceedings of the 2009 International Symposium on Rule Interchange and Applications (RuleML 2009), Las Vegas, Nevada, p. 246–261. doi: 10.1007/978-3-642-04985-9_23.

Hassanpour, S., O'connor, M. J. and Das, A. K. (2010). Visualizing Logical Dependencies in SWRL Rule Bases. The International RuleML Symposium on Rule Interchange and Applications, Washington, DC, p. 259-272.

Hassanpour, S., O'Connor, M. J. and Das, A. K. (2011). "Evaluation of Semantic-Based Information Retrieval Methods in the Autism Phenotype Domain". In: AMIA Annual Symposium.

Heflin, J. (2004). W3C Recommendation: OWL Web Ontology Language Use Cases and Requirements. Disponível em: www.w3.org/TR/2004/REC-webont-req-20040210. acesso Fev. 2012.

Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C. (2004). A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools edition 1.0. University Of Manchester. Disponível em: http://www.co-ode.org/resources/, acesso em Out. 2009.

Horrocks, I, Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C. Disponível em: http://www.w3.org/Submission/SWRL/, acesso em Jan. 2010.

Klyne, G., Carrol, J. J. and Mcbride, B. (Editors). (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C. Disponível em: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, acesso em Set. 2009.

Lassila, O. and Swick, R. (Editors). (2004). Resource Description Framework (RDF) model and syntax specification. W3C. Disponível em: http://www.w3.org/TR/REC-rdf-syntax/, acesso em Out. 2009.

Levy, M., O'Connor, M. J. and Rubin, D. L. (2009). "Semantic Reasoning with Image Annotations for Tumor Assessment". In: AMIA Annual Symposium.

McGuinness, D. L. and Harmelen, F. (Editors). (2004). OWL Web Ontology Language Overview. W3C. Disponível em: http://www.w3.org/TR/2004/REC-owl-features-20040210/, acesso em Jun. 2009.

Noy, N. F. and McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05.

O'connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W. and Musen, M. (2005). Supporting Rule System Interoperability on the Semantic Web with SWRL Fourth International Semantic Web Conference (ISWC2005), Galway, Ireland. Disponível em: http://bmir.stanford.edu/file_asset/index.php/1157/BMIR-2005-1080.pdf, acesso em Maio de 2012.

Orlando, J. P. (2012). Usando aplicações ricas para internet na criação de um ambiente para visualização e edição de regras SWRL. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Recuperado em 2012-08-26, de http://www.teses.usp.br/teses/disponiveis/55/55134/tde-25072012-101810/

Ouellet, R. and Ogbuji, U. (2002). Introduction to DAML: Part I. Disponível em: http://www.xml.com/pub/a/2002/01/30/daml1.html, acesso em Jan. 2010.

Rossello-Busquet, A., Brewka, L. J., Soler, J. and Dittmann, L. (2011). “OWL Ontologies and SWRL Rules Applied to Energy Management”. In: International Conference on Computer Modelling and Simulation (UKSim), p. 446-450. doi: 10.1109/UKSIM.2011.91.

Sadoun, D., Dubois, C., Ghamri-Doudane, Y. and Grau, B. (2011). “An Ontology for the Conceptualization of an Intelligent Environment and Its Operation”. In: Mexican International Conference on Artificial Intelligence (MICAI), p. 16-22. doi: 10.1109/MICAI.2011.32.

Seo, S., Kwon, A., Kang, J. and Hong, J. W. (2011). “OSLAM: Towards ontology-based SLA management for IPTV”. In: IEEE International Symposium on Integrated Network Management (IM), p. 1228-1234. doi: 10.1109/INM.2011.5990570.

Shadbolt, N., Hall, W. and Berners-Lee, T. (2006). The Semantic Web Revisited. IEEE Intelligent Systems, v. 21, n. 3, p. 96-101. doi: 10.1109/MIS.2006.62.

Silva, A. R. (2012). Aprimorando a visualização e composição de regras SWRL na Web. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Recuperado em 2012-08-26, de http://www.teses.usp.br/teses/disponiveis/55/55134/tde-27022012-142801/

Smith, B. and Welty, C. (2001). Ontology: Towards a New Synthesis. In: FOIS’01: Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, p. 3-9. doi: 10.1145/505168.505201.

Smith, M. K., Welty, C. and McGuinness, D. L. (Editors). (2004). OWL Web Ontology Language Guide. W3C. Disponível em: http://www.w3.org/TR/owl-guide/, acesso em Nov. 2008.

Staab, S., Maedche, A. and Handschuh, S. (2001). An annotation framework for the semantic web. In: Proceedings of the First Workshop on Multimedia Annotation, Tokyo, Japan, p. 30-31. doi: 10.1.1.25.910.

Studer, R., Benjamins, R. and Fensel, D. (1998). Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering, v. 25(1-2), p. 161-197. doi: 10.1.1.41.1007.

Su, X. and Ilebrekke, L. (2002). A Comparative Study of Ontology Languages and Tools. In: CAiSE'02:Proceeding of the 14th Conference on Advanced Information Systems Engineering, Toronto, Canada, v. 2348, p. 761-765. doi: 10.1007/3-540-47961-9_62.

SWRLLanguage. (2012). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Disponível em: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ, acesso em Maio 2012.

Tudorache, T., Vendetti, J. and Noy, N. F. (2008). Web-Protégé: A Lightweight OWL Ontology Editor for the Web. In: OWLED 2008: OWL: Experiences and Directions, Karlsruhe, Germany. doi: 10.1.1.142.8568.

Uschold, M. and Grüninger, M. (1996). Ontologies: principles, methods and applications. The Knowledge Engineering Review, v. 11, n. 2, p. 93-155. doi: 10.1017/S0269888900007797.

Uschold, M. and Grüninger, M. (2004). Ontologies and semantics for seamless connectivity. SIGMOD Rec, NY, USA v. 3, n. 4, p. 58-64. doi: 10.1145/1041410.1041420.

Vesin, B., Ivanovic, M., Klasnja-Milicevic, A. and Budimac, Z. (2011). “Rule-based reasoning for altering pattern navigation in Programming Tutoring System”. In: International Conference on System Theory, Control, and Computing (ICSTCC), p. 1-6.

W3C OWL Working Group. (2009). OWL 2 Web Ontology Language Document Overview. W3C. Disponível em: http://www.w3.org/TR/owl2-overview/, acesso em Dez. 2009.

Wusheng, W., Weiping, L., Zhonghai, W., Weijie, C. and Tong, M. (2011). “An Ontology-Based Context Model for Building Context-Aware Services”. In: International Conference on Intelligent Systems, Modelling and Simulation (ISMS), p. 296-299. doi: 10.1109/ISMS.2011.52.

Zacharias, V. (2008). Development and verification of rule based systems – a survey of developers. In: Rule Representation, Interchange and Reasoning on the Web: International Symposium, Orlando, Florida, USA, v. 5321, p. 6-16. doi:10.1007/978-3-540-88808-6_4.