INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP 22290-270

M444 Mattos, Diogo Oliveira Paiva

ROSA^{+}: Uma Extensão do Modelo ROSA com Suporte a Regras e Inferência / Diogo Oliveira Paiva Mattos – Rio de Janeiro: Instituto Militar de Engenharia, 2006.
168p.: il., graf., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2005.

1. Ontologias. 2. Web Semântica 3. Regras e Inferência. I. Título. II. Instituto Militar de Engenharia.

CDD 006.322

INSTITUTO MILITAR DE ENGENHARIA

DIOGO OLIVEIRA PAIVA MATTOS

ROSA^{+}: UMA EXTENSÃO DO MODELO ROSA COM SUPORTE A REGRAS E INFERÊNCIA

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia de Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia de Sistemas e Computação.

Orientador: Ana Maria de Carvalho Moura, Dr.Ing.

Co-orientador: Maria Cláudia Reis Cavalcanti, D.Sc.

Aprovada em 22 de junho de 2006 pela seguinte Banca Examinadora:

_____

Prof`ª Ana Maria de Carvalho Moura, Dr. Ing., IME

_____

Prof`ª Maria Cláudia Reis Cavalcanti, D. Sc., IME

_____

Prof. João Carlos Pereira da Silva, D. Sc., DCC/UFRJ

_____

Prof`ª Maria Luiza Machado Campos, Ph. D., DCC/UFRJ

Rio de Janeiro
2006

AGRADECIMENTOS

Às professoras Ana Maria e Yoko, pela orientação, incentivo, atenção, conhecimento e paciência que despenderam durante essa caminhada.

Aos professores João Carlos e Maria Luiza, não só por fazerem parte da banca examinadora, mas também pelo conhecimento compartilhado durante o mestrado.

A meus pais, por todo o carinho e suporte ao longo de todos estes anos, sem os quais eu não teria conseguido. A meus irmãos e a todos da família pela força e pela torcida.

Aos amigos, pelas conversas e pelos providenciais momentos de descontração.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	APPLICATION PROGRAM INTERFACE
EAD	ENSINO A DISTÂNCIA
OWL	WEB ONTOLOGY LANGUAGE
RDF	RESOURCE FRAMEWORK LANGUAGE
RDF-S	RESOURCE FRAMEWORK LANGUAGE SCHEMA
ROSA	REPOSITORY OF OBJECTS WITH SEMANTIC ACCESS
SWRL	SEMANTIC WEB RULE LANGUAGE
SWRL-FOL	SEMANTIC WEB RULE LANGUAGE – FIRST-ORDER LOGIC
XML	EXTENSIBLE MARKUP LANGUAGE

RESUMO

ROSA é um repositório de Objetos de Aprendizagem (Learning Objects - LOs) com acesso semântico, que permite a criação, armazenamento, reuso e gerenciamento de LOs. Um LO é uma coleção de material reutilizável, utilizado para dar suporte ao aprendizado através de informações em seu conjunto de metadados. Conhecimento no ROSA é representado num modelo próprio, estendido a partir do RDF, onde LOs são acessados de forma contextualizada, determinada por associações semânticas entre eles. Este artigo apresenta o ROSA$^+$, uma extensão do modelo ROSA, que visa deduzir conhecimento semântico através de propriedades de relacionamentos e de regras. Baseado na linguagem de ontologia OWL (Ontology Web Language), e na de regras SWRL (Semantic Web Rule Language), o ROSA$^+$ realiza inferências sobre uma base de dados descrita em OWL, recuperando conhecimento não explicitado por sua representação ontológica.

ABSTRACT

ROSA is a Learning Object (LO) repository system with semantic access, which enables the creation, storage, reuse and management of LOs. A LO is a collection of reusable material used to support learning, characterized by a set of metadata descriptors. Knowledge in ROSA is represented in a RDF extension model, where LOs are accessed in a contextualized way, determined by semantic associations between them. This paper presents ROSA^{+}, a ROSA model extension, that aims deducing additional semantic knowledge through relationships properties and rules. Based on the ontology language OWL (Ontology Web Language), and on the SWRL (Semantic Web Rule Language) rule language, ROSA^{+} is able to perform inferences over a database knowledge written in OWL, retrieving knowledge not explicited in its ontological representation.

1 INTRODUÇÃO

Com o advento da *Word-Wide-Web*, tornou-se possível disponibilizar informações em um espaço global. Textos, imagens, vídeos e outros meios de representação da informação estão hoje disponíveis globalmente como recursos digitais na Web. A partir dessa expansão acelerada, onde o número de páginas cresce exponencialmente a cada ano, encontrar a informação desejada e garantir sua consistência tornou-se uma tarefa cada vez mais complexa nos dias de hoje. A maioria das ferramentas de busca atuais, algumas bem conhecidas, a exemplo do Google[1] e do Yahoo[2], baseia suas pesquisas a partir de palavras chaves, não contextualizadas, não conseguindo capturar com precisão a informação almejada pelo usuário, ficando sob sua responsabilidade a tarefa de selecionar, dentre as páginas retornadas, aquelas de maior relevância.

Diante desse grave problema, surgiu a idéia da *Web Semântica* proposta por Tim Berners-Lee [BERNERS-LEE et al., 2001], que visualizou-a como uma extensão da Web atual, onde a informação tivesse significado e estrutura bem definidos através de ontologias, permitindo assim que seus conteúdos fossem interpretáveis por máquinas. O objetivo maior da *Web Semântica* concentra-se em resolver o problema de heterogeneidade da informação, extraindo suas diferenças sintáticas, estruturais e semânticas, facilitando com isso a compreensão e a recuperação de informações contidas em recursos publicados na *Web*. Na visão de Horrocks [HORROCKS, 2006], o objetivo da *Web Semântica* é facilitar o desenvolvimento de aplicações inteligentes, transformando a Web de um simples repositório de documentos ligados, a uma base de conhecimento e plataforma de aplicações distribuídas (inicitivas de buscadores já baseados em conceitos da Web Semântica, tais como o SWoogle[3] e o Intellidimension[4] começam a ser desenvolvidas, encontrando-se ainda em estágio inicial de desenvolvimento). Para tal, ontologias têm um papel fundamental: transformar e capturar conhecimento que permitirá que aplicações melhor compreendam recursos na Web, utilizando-os de forma mais inteligente .

No entanto, para que esse objetivo pudesse ser alcançado, tornou-se necessário investir na criação de uma infra-estrutura tecnológica básica, fundamentada em lógica,

[1] http://www.google.com
[2] http://search.yahoo.com/
[3] http://swoogle.umbc.edu
[4] http://www.semanticwebsearch.com/

onde fossem projetadas linguagens para expressar meta-informação. Estas deveriam ainda ser interpretáveis por máquina, de modo que ferramentas e novas arquiteturas as utilizassem como linguagem padrão [HORROCKS, 2006].

Nesse processo evolutivo e de grande transformação do mundo *Web*, foram criadas linguagens atualmente consideradas como padrão nesse ambiente, a exemplo de XML [XML, 2004], RDF/RDFS [RDF, 1999; RDFS, 2004] e OWL [DEAN et al., 2004]. XML (*Extensive Markup Language*) é a linguagem padrão para a representação de dados semi-estruturados na Web. Porém, esta linguagem por si só não é capaz de oferecer significado às estruturas dos dados que abriga. Para isso, foi desenvolvida a linguagem RDF (*Resource Description Framework*), que conjuntamente com o RDFS (*RDF Schema*), utilizam a sintaxe XML para representar conhecimento através de triplas compostas por sujeito, predicado e objeto. No entanto, apesar desta linguagem expressar dado e metadado de maneira uniforme, e prover a estrutura almejada para os dados que representa, ela ainda não é capaz de capturar a conotação semântica do dado, tão fundamental no contexto da Web Semântica. Assim surgiu o OWL (*Web Ontology Language*), que tem por objetivo representar e descrever bases de conhecimento, também denominadas de ontologias. OWL é uma evolução da RDF: apresenta um vocabulário mais rico, onde é possível representar relações entre classes, propriedades de relacionamentos, cardinalidade, dentre outros avanços. Tanto RDF como OWL foram baseadas em lógicas descritivas e de primeira ordem, consideradas requisitos básicos para processamento em máquina.

Enquanto RDF e OWL foram estabelecidas como linguagens padrão de representação de ontologias pelo W3C, grandes esforços têm sido dispensados em prol do desenvolvimento do aumento da capacidade de inferência dessas linguagens. Nesse contexto, surgiram linguagens de regras para a *Web Semântica*, que permitiriam que um usuário pudesse declarar o seu conhecimento de maneira a ser interpretável por um agente, acarretando em pesquisas com resultados mais precisos.

Em uma ontologia OWL, não é possível, por exemplo, obter conhecimentos implícitos através de relacionamentos entre propriedades. Por exemplo, a definição de um relacionamento “tem tio” a partir da definição de relacionamentos “tem pai” e “tem irmão”. Regras poderiam inferir um fato não representado, dando uma maior expressividade ao documento [BARKMEYER, 2006].

Nesse contexto, nos últimos dois anos, algumas linguagens de regras foram submetidas ao W3C para avaliação: *Semantic Web Rule Language* (SWRL) [HORROCKS et al., 2004], *Semantic Web Rule Language - First-Order Logic* (SWRL-FOL) [PATEL-

SCHNEIDER, 2005] e *Web Rule Language* (WRL) [DE BRUIJIN et al., 2005]. O objetivo almejado pela W3C é estabelecer uma linguagem padrão para o desenvolvimento de regras na Web, de forma a tornar possível representá-las, para que posteriormente seja possível realizar inferências sobre a ontologia. De fato, esta capacidade pode ser o caminho para a solução de problemas complexos, que vão desde a integração de conhecimento relevante de diversas fontes, até a decomposição de problemas em partes, de modo a serem utilizados posteriormente por agentes distintos.

De modo a melhor explorar as potencialidades das linguagens de ontologias e de regras, permitindo a partir delas inferir conhecimento, torna-se necessário raciocinar sobre as ontologias. Tendo o RDF e o OWL estabelecidos como linguagens padrão de ontologias pelo W3C, e tendo este consórcio como meta importante concentrar esforços em uma linguagem para regras na Web Semântica, tanto a teoria quanto a prática de raciocínio na Web encontram-se em franco desenvolvimento [HORROCKS, 2006]. Assim, no núcleo do desenvolvimento da Web Semântica, encontram-se os raciocinadores para ontologias, inspirados nas pesquisas de dedução automática, inteligência artificial e lógica computacional, a exemplo do RACER [HAARSLEV e MÖLLER, 2003], Pellet [PARSIA e SIRIN, 2004], FaCT++ [FACT] e Bossam [JANG, 2005]. A idéia é que raciocinadores sejam capazes de interoperar entre SWRL e OWL, de modo que, após sua execução, as deduções obtidas sejam convertidas em fatos, e uma nova execução seja realizada. Esta iteração deveria ocorrer até que uma inconsistência seja detectada ou novos fatos não sejam mais inferidos [GOLBREICH, 2006].

Diversas aplicações podem se beneficiar de um modelo de dados no qual a representação semântica de suas associações sejam computáveis. Tais associações são relevantes, tanto na descrição dos objetos do domínio quanto no suporte às consultas efetuadas pelos usuários, tornando possível o desenvolvimento de sistemas que sejam capazes de efetuar consultas que se valham de um modelo semântico de dados.

Um exemplo desse tipo de sistema é o ROSA (*Repository of Objects with Semantic Access*) [MOURA et al., 2003; PORTO et al., 2003], que foi desenvolvido como uma extensão do modelo de dados RDF e atende a aplicações voltadas para o ensino a distância, ou *e-learning.* O ROSA é responsável por gerenciar o conteúdo, armazenar e dar suporte ao acesso a certos tipos de objetos, conhecidos como objetos de aprendizagem (*Learning Objects* - LOs). Esses LOs são formados por um conjunto de características e propriedades denominados metadados. Com o auxílio de LOs, torna-se possível a criação de cursos eletrônicos a distância, oferecendo dessa maneira um suporte inestimável a uma

área que muito tem crescido, o de Ensino a Distância (EAD). De posse de um determinado conjunto de LOs, um autor de um curso poderá estruturar todo o conteúdo didático necessário utilizado em seu processo de aprendizagem, educação ou treinamento.

No contexto da Web Semântica, ROSA é um sistema que permite expressar associações entre os LOs, de modo a prover uma contextualização entre os mesmos e, conseqüentemente, enriquecer semanticamente suas descrições. O sistema objetiva dar suporte a técnicas de processamento de consultas que irão explorar os metadados e os relacionamentos semânticos entre os LOs, auxiliando nas consultas efetuadas sobre o sistema. Assim, um usuário pode formular consultas mais expressivas em cima de um modelo que ofereça uma forma eficiente de obter todos os LOs que julgar úteis no processo de elaboração de seus cursos, por exemplo. Nesse contexto, o ROSA em muito poderia se beneficiar se a ontologia que suporta pudesse ser acrescida de regras, que o ajudariam sobremaneira no processo de dedução de novas informações.

1.1 MOTIVAÇÃO

O sistema ROSA, em sua versão atual, é baseado numa extensão do modelo de dados RDF, onde o conhecimento é expresso através de triplas sujeito-predicado-objeto. Assim, LOs são associados entre si através de relacionamentos, agregando uma certa semântica ao conhecimento. As consultas realizadas no sistema apresentam como resultados elementos que estejam de fato representados em sua base de dados ou que sejam oriundos de operações de relações de equivalência efetuadas sobre os relacionamentos ou propriedades, tais como transitividade, simetria, etc.

Entretanto, esta representação apresenta limitações. Neste aspecto, a lógica poderia funcionar como uma importante ferramenta, trazendo significado e semântica a um determinado contexto. Através de uma representação baseada em lógica, seria possível a inferência de conhecimento representado por propriedades de relacionamentos e regras, aumentando assim consideravelmente o poder das consultas.

Desta forma, a motivação do trabalho é dotar o ROSA de maior poder de inferência, o que requer a extensão do seu modelo de dados para uma abordagem mais voltada à lógica, aumentando assim sua expressividade através da representação de regras e do

raciocínio sobre estas. Desta maneira, pretende-se dotá-lo da capacidade de expressar características implícitas de uma ontologia.

1.2 OBJETIVO

Este trabalho tem por objetivo estender o modelo de dados ROSA acrescendo-o da capacidade de representar regras e propriedades de relacionamentos. Com isso, esse novo modelo, denominado nesse trabalho de ROSA^{+}, tornar-se-á capaz de tomar decisões não somente sobre fatos já instanciados, e por isso de conhecimento do sistema, como também de novos fatos inferidos pelas propriedades dos relacionamentos definidas na especificação da ontologia, e também por regras representadas na ontologia.

O presente trabalho também tem por objetivo representar o ROSA através de uma abordagem em camadas, permitindo que sejam realizadas consultas em diferentes níveis de abstração: além de consultar apenas sobre as instâncias, como ocorre na representação atual do ROSA, será possível realizar pesquisas também sobre o esquema no domínio EAD e sobre o próprio modelo de dados ROSA^{+}.

Além disso, o trabalho visa também implementar o sistema ROSA^{+} através de um protótipo, com o objetivo de validar o modelo e a arquitetura propostos. Nele, o usuário poderá criar e editar regras à sua base de dados e executar consultas sobre os variados níveis da arquitetura ROSA^{+}, onde conhecimento não explicitado pode ser inferido através de propriedades dos relacionamentos e de regras.

1.3 ORGANIZAÇÃO

Esta dissertação encontra-se organizada em sete capítulos, conforme detalhamento a seguir.

O capítulo 1 apresenta esta dissertação, situando e expondo sua motivação, justificativa e objetivos.

O capítulo 2 descreve o sistema ROSA, abordando os principais conceitos relativos ao domínio de Ensino a Distância, exemplificando seu funcionamento através de mapas conceituais e descrevendo seu modelo de dados.

O capítulo 3 apresenta um estudo sobre os conceitos e tecnologias importantes voltados à ontologias, utilizados ao longo deste trabalho. Nesse capítulo, grande ênfase é dada às linguagens utilizadas no contexto da *Web Semântica* bem como os princípios básicos da lógica descritiva, que marcam os fundamentos da linguagem de ontologia OWL e das linguagens de regras SWRL, SWRL-FOL e WRL, também estudadas neste capítulo.

O capítulo 4 apresenta a extensão do modelo ROSA, o ROSA$^+$, cujo objetivo maior é comportar a representação de regras. Também neste capítulo é apresentada a arquitetura ROSA$^+$, que permite uma representação do modelo em diferentes níveis de abstração, bem como a especificação de todos os elementos desta arquitetura.

O capítulo 5 mostra o desenvolvimento das ontologias que representam a arquitetura ROSA$^+$, servindo como base de dados para o desenvolvimento do sistema. Além disso, apresenta um breve estudo acerca de alguns editores de ontologias necessários à construção da ontologia proposta em OWL-DL, e também sobre raciocinadores, responsáveis pelo mecanismo de inferência de conhecimento a ser integrado ao sistema ROSA$^+$.

O capítulo 6 aborda aspectos de implementação do sistema ROSA$^+$, mostrando detalhes de seu desenvolvimento e funcionamento. Estes são realizados a partir de um estudo de caso, tendo como objetivo principal validar a arquitetura e a especificação propostas e apresentadas nos capítulos 4 e 5, respectivamente.

Finalmente, no Capítulo 7 são apresentadas as conclusões dessa dissertação, apontando as suas principais contribuições e algumas sugestões para trabalhos futuros.

2 O SISTEMA ROSA

Neste capítulo, será apresentado o ROSA, sistema sobre o qual o estudo desta dissertação será aplicado. Na seção 2.1 é feita uma breve abordagem sobre o conceito de *e-learning*, uma vez que a filosofia do ROSA está inserida neste conceito. A seção 2.2 exemplifica o funcionamento do sistema através de mapas conceituais, enquanto a seção 2.3 traz uma descrição do modelo de dados do ROSA antes de sua extensão para a representação de regras, assunto abordado no capítulo 4.

2.1 INTRODUÇÃO

Educação a distância é uma forma de ensino onde os aprendizes estão separados fisicamente da instituição de ensino, e cujo processo de aprendizado não depende de um estabelecimento baseado em um local pré-estabelecido. Esta forma de educação necessita de várias tecnologias educacionais para facilitar o aprendizado e os processos de comunicação entre tutores e aprendizes.

Existem inúmeros sistemas responsáveis por suportar aplicações voltadas para aprendizagem eletrônica, ou *e-learning*. Sistemas Gerenciadores de Objetos de Aprendizagem (*Learning Contents Management Systems* - LCMSs) foram desenvolvidos com o objetivo de abstrair o armazenamento e o acesso a material de *e-learning* por outras aplicações. A unidade fundamental em um modelo de dados de *e-learning* é denominada objeto de aprendizagem (*Learning Object* – LO). Um LO é uma coleção de material reutilizável, utilizado para dar suporte ao aprendizado [FRIESEN, 2001]. Um LO é identificado por um conjunto de descritores (metadados) definidos por um padrão internacional de metadados, tais como o LOM (*Learning Object Metadata*) [IEEE, 2004] e o IMS [IMS, 2005]. Estes metadados trazem informações sobre o LO, tais como: características gerais, ciclo de vida, requisitos técnicos, características educacionais, propriedade intelectual, relacionamentos com outros LOs, observações e classificação do LO.

Em um LCMS, o projeto de cursos de *e-learning* oferecem suporte a consultas para explorar seus metadados e relacionamentos semânticos, sendo bastante utilizados pela

comunidade acadêmica, com o objetivo de publicar e compartilhar os materiais de seus cursos. Neste contexto, o ROSA (*Repository of Objects with Semantic Acess*) [MOURA et al., 2003; PORTO et al., 2004] foi concebido para dar suporte aos profissionais da área educacional na fase de projeto de um curso de *e-learning*, auxiliando-os na busca de materiais didáticos que forneçam subsídios para a preparação de suas aulas ou conteúdos instrucionais.

No ROSA, usuários utilizam-se de um mapa conceitual para visualizar composições de LOs e suas associações. LOs representam entidades do mundo real, como curso, disciplina e tópico, bem como seu conteúdo físico digital, que podem estar disponíveis na forma de apresentações, documentos, etc. Os LOs se relacionam através de associações, que são predicados pré-definidos por uma ontologia. Associações permitem contextualizar melhor os LO, auxiliando os usuários durante sua pesquisa. Também representam a estrutura de um curso, informando sobre todos os relacionamentos semânticos que possuem com outros LOs. Além do mais, através de uma ontologia, outras propriedades não representadas explicitamente podem ser deduzidas sobre os LOs, tais como a simetria, a transitividade e a reflexão. Quando usada em um mapa conceitual, uma associação é representada por um termo de acordo com um determinado relacionamento ontológico.

Por contextualizar LOs, o ROSA auxilia o usuário na sua pesquisa a estas estruturas. Por exemplo, é possível que um instrutor pesquise por um material relevante para sua disciplina, cujo conteúdo dependa de um determinado tópico, ou ainda que um estudante acesse o sistema para obter informações sobre uma disciplina baseado em características específicas, tais como compreender um tema ministrado por um professor específico em um dado ano.

Ao invés de estender os mapas conceituais para diferentes vocabulários, os termos usados para nomear os LOs e associações podem ser classificados através de um tesauro [GOMES, 1990] de domínio, onde sinônimos, generalizações/especializações e termos associados são definidos. A capacidade de expressar consultas sobre mapas conceituais no ROSA com auxilio de tesauros, possibilita um mecanismo de pesquisa simples e poderoso, adequado para comunidades com muitos membros.

O ROSA possui um modelo de dados cuja estrutura principal é construída sobre a hierarquia de classes dos LOs. Além disso, ela inclui uma álgebra poderosa [COUTINHO, 2004] que torna possível não somente consulta a metadados dos LOs, bem como às associações semânticas entre os mesmos.

2.2 MAPAS CONCEITUAIS

A representação de cursos no ROSA é feita através de mapas conceituais. Eles são utilizados como artefatos para organizar e representar um conhecimento de maneira simples e prática [DÜRSTELER, 2004]. Um mapa conceitual é representado como um grafo direcionado, onde os vértices representam os LOs, e as arestas representam os relacionamentos, da mesma maneira que os predicados no RDF. Assim, este mecanismo provê ao projetista de *e-learning* uma maneira de modelar e visualizar os relacionamentos entre LOs de sua aplicação.

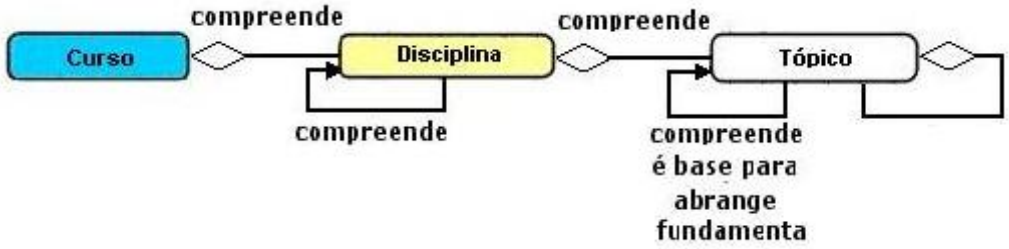

FIG. 2.1 Exemplo de Mapa Conceitual.

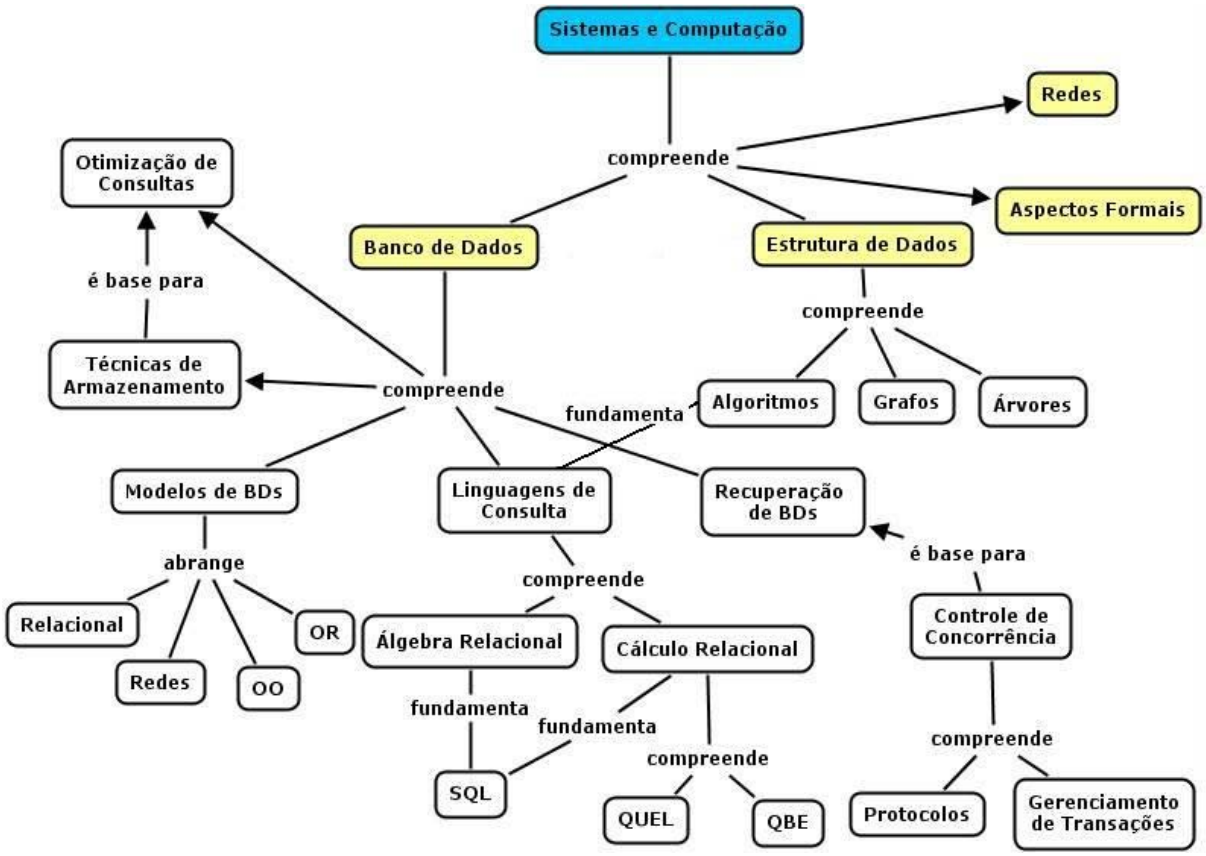

FIG. 2.2 Exemplo de Mapa Conceitual.

A figura 2.1 representa esquema descritor correspondente ao mapa conceitual parcial do programa de Mestrado em Sistemas e Computação do IME, representado na figura 2.2. As classes Curso, Disciplina e Tópico correspondem a tipos de domínio específicos de LOs, representando o esquema. De fato, este esquema modela a estrutura de composições da maioria dos programas no Brasil, onde um curso é composto por disciplinas, e estas, por tópicos, os quais podem também conter outros tópicos. Entretanto, existe a possibilidade de se expressar diferentes esquemas, tornando possível a modelagem de domínios de EAD que tenham uma estrutura diferente da representada na figura 2.1, ou até mesmo de outros domínios de conhecimento, que podem vir a ser explorados em outros trabalhos.

Neste mapa conceitual, alguns LOs estão associados às suas classes, representadas pela mesma cor. Relacionamentos entre os LOs podem representar, por exemplo, como as disciplinas de um curso específico podem ser relacionadas aos tópicos que abrangem, ou quando elas podem ser lecionadas, se antes ou depois de um determinado tópico ou disciplina, de acordo com a semântica do predicado correspondente, como por exemplo, "é base para" e "fundamenta". Assim, é possível expressar, por exemplo, que o ensino de *Álgebra Relacional* fundamenta o ensino de *Linguagem SQL*, e que o tópico *Controle de Concorrência* é base para o tópico *Recuperação de BDs*, isto é, o último não pode ser estudado antes que o primeiro seja abordado

2.2.1 RELACIONAMENTOS

Os relacionamentos em um mapa conceitual também são chamados de predicados. Eles podem ser classificados de acordo com sua semântica como tipo de agregação ou domínio específico. Os relacionamentos colocam os LOs em um contexto por associações significativas com outros LOs. Assim, os usuários podem escolher os termos para representar os predicados que lhe dêem uma melhor semântica para cada domínio. A computabilidade de cada termo é obtida classificando-os de acordo com seu tipo, para posteriormente associá-lo a propriedades (reflexão, simetria e transitividade).

No contexto de *e-learning*, relacionamentos do tipo agregação são especialmente importantes, pois expressam qual o conjunto de LOs deve ser definido quando se deseja

expressar determinado conhecimento (uma disciplina, por exemplo). Assim, predicados do tipo agregação aparecem freqüentemente nos mapas conceituais, apresentando a propriedade da transitividade, implicitamente explorada durante a execução de consultas.

2.3 MODELO DE DADOS ROSA

O modelo de dados ROSA possui uma parte estrutural e outra comportamental. A primeira é definida por LOs e relacionamentos entre os mesmos. A estrutura de um LO consiste em um conjunto de metadados do domínio de educação.

O conjunto de metadados dos LOs foi definido segundo o padrão de metadados IEEE-LOM [IEEE, 2002], e contém informações importantes sobre o LO, tais como identificadores, títulos, idiomas, autores, versão, etc. Inicialmente, o aspecto estrutural do modelo é especificado, seguido por uma álgebra que define um conjunto de operações válidas.

A figura 2.3 apresenta o modelo de dados ROSA expresso em um diagrama de classes UML. A classe *RecursoComplexo* é a raiz das classes hierárquicas, provendo uma representação comum tanto para LOs quanto para relacionamentos.

A classe *LO* é definida segundo o padrão de metadados LOM, onde cada instância é identificada por um atributo identificador único. A classe *LO* é especializada em *LO Lógico* e *LO Físico*. O primeiro representa conceitos, enquanto o segundo representa os documentos armazenados, associados a determinado *LO*. Os LOs Lógicos são classificadas de acordo com os níveis de agregação do domínio (curso, disciplina e tópico). Já os LOs Físicos correspondem a arquivos e seus metadados. Uma instância em um LO Físico é considerada atômica, o que significa que ele não pode ser composto por outros LOs, ao contrário de uma instância de LO Lógico, que pode conter uma coleção de outros LOs associados por agregação e relacionamentos semânticos.

A classe *Relacionamento* possui como instâncias todos os verbos (predicados) empregados no domínio de um mapa conceitual: *fundamenta*, *compreende*, etc. Suas instâncias são classificadas de acordo com um tipo de relacionamento. Como mencionado na seção 2.2.1, os tipos de relacionamentos podem ser de agregação ou semântico, ou seja, de um domínio específico - podendo ainda definir propriedades de equivalência, reflexão, simetria e transitividade. O sistema explora implicitamente cada propriedade quando

executa uma consulta sobre os relacionamentos. Por exemplo, uma consulta para se obter os LOs compreendidos pelo LO *Sistemas e Computação* (FIG 2.2) pode incluir em seu conjunto resultado o LO *Álgebra Relacional*, após efetuar uma navegação sobre o predicado compreende que, implicitamente, assume a propriedade transitiva sobre uma lista de relacionamentos do tipo agregação.

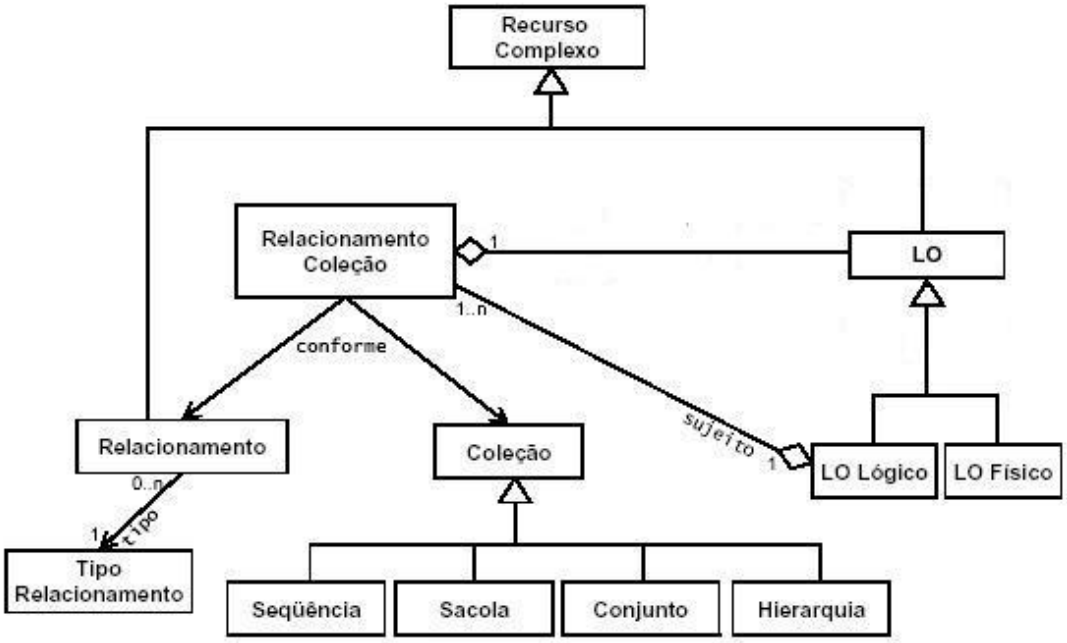

FIG. 2.3 O Modelo de dados ROSA.

Coleção é uma classe que representa coleções físicas de objetos. Uma coleção pode impor uma determinada ordenação que deve ser seguida durante a visita aos LOs aí contidos. Assim, o modelo especializa a classe coleção em: *Lista*, *Sacola*, *Conjunto* e *Hierarquia*. Conjuntos e listas representam os mesmos conceitos matemáticos correspondentes, enquanto sacolas podem conter elementos duplicados na coleção. Uma hierarquia corresponde aos LOs cujos elementos seguem uma estrutura de árvore. As classes *Conjunto* e *Sacola* possuem atributos para identificar cardinalidades máxima e mínima, definindo assim os números máximo e mínimo de elementos que podem ser extraídos da respectiva coleção.

Associações entre LOs, cuja cardinalidade é tipicamente um-para-muitos, são modeladas como *Relacionamento Coleção*, especialização das classes *Relacionamento* e *Coleção*. Estas associações conceitualmente definem triplas *t(sujeito, propriedade, objeto)*, estendendo relacionamentos n-ários, onde *sujeito* e *objeto* são do tipo *LO* e *propriedade* é

um *Relacionamento*. O esquema especifica as associações válidas através de triplas *T(classe, tipo de relacionamento, classe)*. Assim, instâncias válidas de *t* estão de acordo com *T*.

É importante observar algumas diferenças entre os modelos de dados ROSA e RDF [RDF, 1999], até mesmo através de estruturas básicas similares, como as representadas pelos relacionamentos no ROSA e por sentenças em RDF. Primeiramente, os atributos de metadados LOM são especificados como propriedades da classe *LO*, ao invés de sentenças sobre a fonte dos LOs. Assim, os valores dos atributos não são identificáveis e não são considerados objetos de primeira classe, como no caso dos literais em RDF [FRANCINCAR et al., 2003]. Agregação e relacionamentos semânticos são modelados como coleções de LOs. Esta forma é mais geral que o RDF, onde coleções se referem a uma sacola e especificam uma lista de predicados para modelar coleções, conduzindo a uma representação complexa para a avaliação de consultas. A cardinalidade dos relacionamentos no ROSA restringe a participação dos membros em coleções, que não está disponível na semântica das sentenças RDF. Por exemplo, considere os predicados mostrados na figura 2.4a, que associam *Linguagem de Consulta* a *Álgebra Relacional* e *Cálculo Relacional*. Em RDF não é possível garantir que todos os membros da coleção possam ser visitados quando alcançado o LO *Linguagem de Consulta*. Enquanto isso, o modelo proposto considera os membros da coleção como uma unidade única, como mostra a figura 2.4b, garantindo que todos os seus elementos sejam efetivamente visitados.

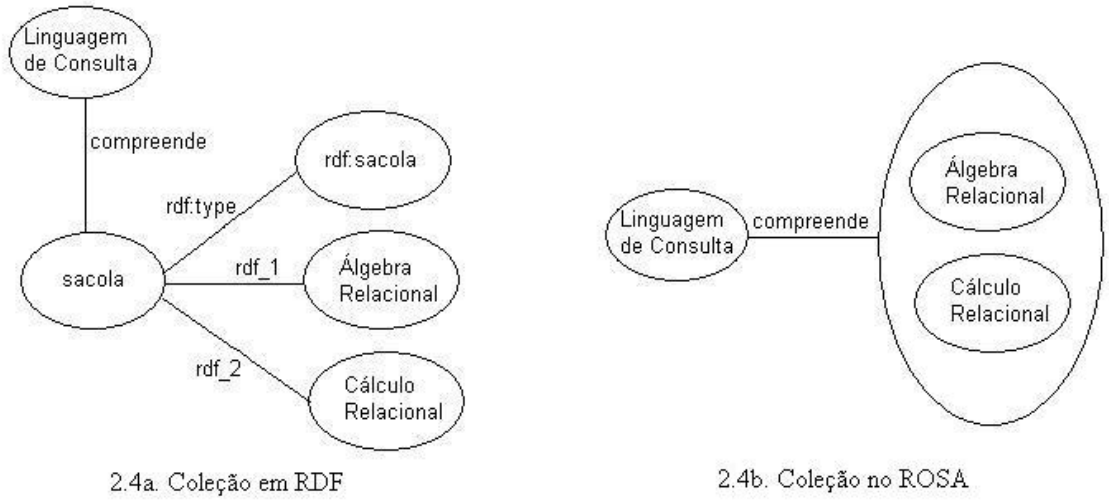

FIG 2.4. Relacionamentos do tipo Coleção.

Além disso, o modelo de dados ROSA representa relacionamentos como parte dos dados que instancia. Assim, usuários podem consultar associações e dados sem distinção. Isto diferencia o ROSA do modelo de dados OO, onde relacionamentos são partes dos esquemas de bancos de dados e não podem ser consultados.

Finalmente, o modelo de dados ROSA não trata da herança. Isto foi proposto como uma extensão do modelo [HANDRICK, 2005], no qual a programação lógica é usada para expressar mapas conceituais, herança, regras e propriedades dos relacionamentos.

2.4 EXTENSÕES DO SISTEMA ROSA

Desde sua concepção, diversos trabalhos foram feitos sobre o sistema ROSA, visando incorporar novas funcionalidades. Os *Topic Maps* [GARSHOL et al., 2003] dão um contexto aos LOs e suas associações, oferecendo visões diferenciadas com mapas conceituais simples para acomodar objetivos de cursos particulares. Fernandez [FERNANDEZ, 2004] incorporou a abordagem de *topic maps* na representação do modelo ROSA.

Em Coutinho [COUTINHO, 2004], foi proposta uma álgebra ROSA, para permitir que sejam feitas consultas sobre bases de dados ROSA, com ênfase nos metadados e relacionamentos do modelo de dados. Através desta álgebra, é possível compor expressões complexas através da composição entre expressões básicas.

Handrick [HANDRICK, 2005] estendeu o ROSA para o reconhecimento da herança, onde a programação lógica foi usada para expressar mapas conceituais e regras implementando herança e propriedades dos relacionamentos.

Foi desenvolvido também o ROSA P2P, permitindo que consultas sejam submetidas de qualquer nó pertencente à rede ROSA e processadas de acordo com estratégias específicas [BRITO, 2005], que coletam e integram resultados intermediários de diversos *peers* num resultado final, que é então enviado ao usuário. Esta funcionalidade permite ao ROSA a integração com outras instituições acadêmicas, como um repositório flexível de LOs.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o sistema ROSA, que oferece a seus usuários a possibilidade de criar e consultar objetos de aprendizagem. Este sistema é baseado em um modelo de dados de LOs, e possui uma álgebra para suportar consultas sobre LOs e navegação através de um mapa conceitual.

O modelo de dados ROSA estende o modelo de dados RDF, com algumas vantagens: simplifica a representação de atributos, facilitando a implementação e formulação de consultas; considera relacionamentos de equivalência entre as propriedades, usados implicitamente nas consultas; modela relacionamentos como coleções n-árias ordenadas, restringindo o acesso a LOs em uma coleção através de cardinalidades máxima e mínima.

De modo a conferir ao sistema ROSA maior poder de expressividade, compatível com o grau de evolução das linguagens de representação de ontologias, a linguagem OWL (*Web Ontology Language*) [DEAN et al., 2004], recomendada atualmente pelo W3C como linguagem para desenvolvimento de ontologias na *Web*, aparece como forte candidata para dar prosseguimento às pesquisas no ambiente ROSA em questão.

Assim, a próxima etapa do desenvolvimento do sistema ROSA diz respeito à representação de seu modelo em OWL, bem como a utilização de linguagens de regras, como SWRL [HORROCKS et al., 2004] e WRL [BRUIJIN et al., 2005] para prover descoberta de conhecimento através de raciocínio sobre regras na base de dados ROSA. De forma a melhor descrever os conceitos fundamentais das linguagens de manipulação de ontologias e de regras aqui mencionadas, serão apresentados no próximo capítulo seus conceitos básicos, fundamentando melhor os passos desenvolvidos no contexto dessa dissertação.

3 ONTOLOGIAS NA REPRESENTAÇÃO DO CONHECIMENTO

O objetivo deste capítulo é realizar um apanhado geral sobre os conceitos e tecnologias que fundamentam o estudo de ontologias, com o objetivo de representar conhecimento. Para tanto, após a introdução à Web Semântica, a seção 3.2 apresenta a arquitetura para as linguagens de desenvolvimento no contexto da Web Semântica; a seção 3.3 introduz conceitos de lógica descritiva, cuja teoria é de grande importância no desenvolvimento das linguagens da Web Semântica; a seção 3.4 descreve a linguagem para a descrição e representação de ontologias, a OWL, já recomendada pelo W3C como linguagem oficial para o desenvolvimento de ontologias; e, finalmente, a seção 3.5 faz uma análise das principais linguagens de regras utilizadas na Web Semântica, com o objetivo de aumentar o poder de expressividade de uma ontologia.

3.1 A WEB SEMÂNTICA

A maioria do conteúdo da *Web* hoje em dia é projetado para a leitura humana e não para que programas de computador possam operá-los de forma significativa. Computadores podem habilmente analisar páginas *Web* para processamento de *layout* e rotinas, mas, em geral, não apresentam nenhuma maneira confiável para, por exemplo, processar a semântica e identificar palavras homônimas [DAVIES et al., 2003].

A Web Semântica surgiu então como uma extensão da *Web* atual, onde a informação recebe um significado bem definido, possibilitando que computadores e pessoas trabalhem em cooperação através de metadados [BERNERS-LEE et al., 2001]. A Web Semântica vislumbra a idéia de que dados na Web sejam definidos e ligados entre si, sendo assim processáveis por máquina, na automação, integração e reuso de aplicações. Para isso, computadores necessitam ter acesso a coleções estruturadas de informações (dados e metadados) e de um conjunto de regras de inferência que ajudem no processo de dedução automática.

Para atingir este objetivo, dados devem ser descritos de forma que seu significado seja interpretável por computadores. O êxito da Web Semântica [BERNERS-LEE, 2001] necessita de:

- desenvolvimento de linguagens que sejam capazes de expressar metainformações. Estas linguagens devem ser interpretáveis por máquinas, para assim poderem ser utilizadas em documentos e desenvolvimento de terminologias e disponibilizadas na Web;
- desenvolvimento de ferramentas e novas arquiteturas que utilizem estas linguagens e terminologias adequadas para prover suporte a busca, manutenção e apresentação das fontes de informações;
- desenvolvimento de aplicações que contenham um novo nível de serviços para usuários da Web Semântica.

As ontologias oferecem uma maneira apropriada para dar semântica a informações heterogêneas de diversas fontes. O modelo de um domínio expressado em uma ontologia pode ser considerado uma estrutura uniforme que dá semântica e uma representação comum à informação.

3.2 LINGUAGENS DA WEB SEMÂNTICA

Um dos principais aspectos da arquitetura da Web Semântica é o conjunto de linguagens que a compõem. Berners-Lee [BERNERS-LEE, 2005] propôs uma arquitetura inicial, onde cada tipo de linguagem representa um nível da pirâmide, conforme mostra a figura 3.1.

Nesta arquitetura, as linguagens de marcação estão na base da pirâmide. HTML, XML, XHTML e XML Schema, além dos padrões Unicode[5], para conjunto de caracteres, e Universal Resourse Identifier[6] (URI), para identificação e referência universal de recursos.

Na camada seguinte, estão as linguagens no nível de dados e de esquema. São documentos auto-descritivos. Nela estão as linguagens RDF (Resource Description Framework) [RDF, 1999] e RDF Schema (RDFS) [RDFS, 2004]. RDF é uma linguagem

[5] http://www.unicode.org
[2] http://www.w3.org/Addressing/URL/URI_Overview.html

usada para a definição e uso de descrições de metadados, enquanto a RDFS, estende as funcionalidades da RDF para as primitivas básicas de modelagem de ontologias, como hierarquia de classes, restrições de domínio e abrangência de propriedades.

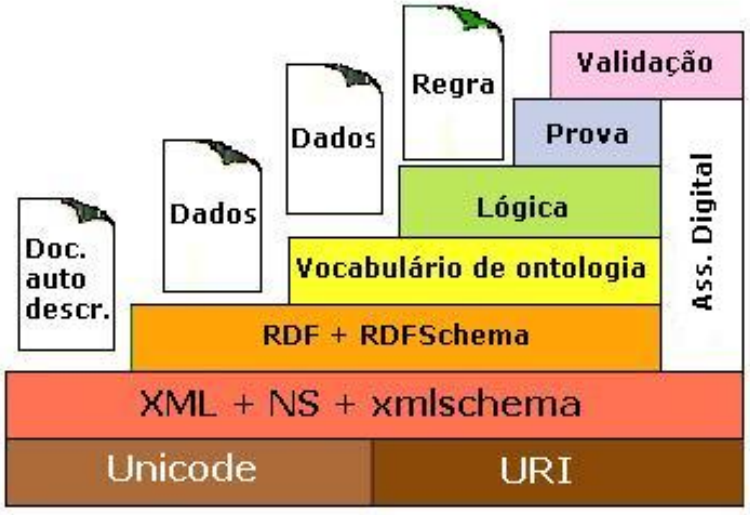

FIG. 3.1 Arquitetura de linguagens da Web Semântica.

O nível seguinte na hierarquia proposta por Berners-Lee é o das linguagens de ontologias, que têm como objetivo prover termos com significados bem definidos através do uso de ontologias, para que a camada seguinte possa utilizar as bases de conhecimento fornecidas na resolução de problemas por ela requisitados. Nesta camada, está localizada a OWL [DEAN et al., 2004].

Acima delas está a camada lógica, onde se podem deduzir informações, permitindo que a partir delas seja possível deduzir também inferências das definições e relacionamentos dos termos. Nela encontram-se as linguagens de regras para ontologias, tais como a SWRL [HORROCKS et al., 2004], a SWRL-FOL [PATEL-SCHNEIDER, 2005] e a WRL [DE BRUIJIN et al., 2005].

Assim como em um sistema distribuído, a Web Semântica pressupõe a idéia de que "qualquer pessoa pode declarar qualquer coisa". Desta forma, é possível declarar alguma sentença, que pode acabar entrando em conflito com outra já declarada anteriormente. Assim, é necessário que, a partir de uma sentença deduzida pelo sistema, seja possível recuperar todos os passos gerados até a obtenção daquela sentença (camada de prova), de

tal forma que esta seja validada (camada de validação) através da verificação da integridade da prova [LU et al., 2002].

Entretanto, esta arquitetura supõe que as linguagens das camadas inferiores na hierarquia sejam sempre plenamente compatíveis com as das camadas superiores. Uma alternativa para esta arquitetura foi proposta por alguns grupos de estudo de linguagens de regras. Kifer et. al [KIFER, 2005] apresentam uma proposta, considerada mais realista, por permitir que múltiplas tecnologias se localizem lado a lado na mesma camada, conforme mostra a figura 3.2. A camada de lógica é apoiada pela linguagem OWL e por regras, enquanto na camada de prova pode-se utilizar tanto regras baseadas em programação em lógica quanto regras de cunho puramente de negócio para gerar o passo a passo das cláusulas avaliadas no processo de geração do resultado final.

SparQL[7] é uma linguagem de consulta padrão para RDF, que oferece a usuários e desenvolvedores uma forma padrão para se escrever e se obter resultados ao longo das várias camadas de arquitetura.

Assim, se porventura uma tecnologia se tornar obsoleta, uma nova tecnologia pode ser adicionada à arquitetura, sem a necessidade de ser totalmente compatível com a antiga.

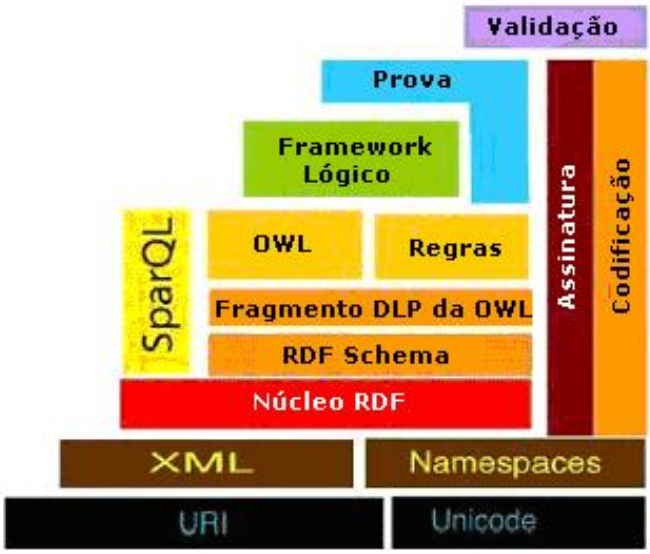

FIG. 3.2 Nova proposta para arquitetura de linguagens da Web Semântica.

Limitações podem ser observadas na arquitetura proposta para a Web Semântica, especialmente no tocante à expressão de regras. Regras são uma necessidade antiga da

[7] http://www.w3.org/TR/rdf-sparql-query/

Web Semântica. Com a definição da OWL na base da camada de ontologias da arquitetura, regras passaram a ser um importante foco de estudo.

Esta linha de raciocínio é contrária à defendida por Horrocks et al. [HORROCKS et al., 2005], que defendem a arquitetura atual sem o acúmulo de tecnologias na mesma camada, por considerar que uma maximização das compatibilidades entre as linguagens existentes, especialmente RDF e OWL, irão trazer mais benefícios para o desenvolvimento da Web Semântica do que dispersar os estudos com o desenvolvimento de novas linguagens.

Nas seções seguintes, serão abordados conceitos de lógica descritiva, que fundamentam tanto a linguagem de ontologia OWL como as linguagens de regras atualmente em desenvolvimento. A primeira será utilizada no ROSA$^+$ para a construção da ontologia que representará o sistema em todos os seus níveis de abstração; já o estudo das segundas fundamenta a escolha de uma delas para a representação de regras no ROSA$^+$.

3.3 LÓGICA DESCRITIVA COMO SUPORTE À REPRESENTAÇÃO DO CONHECIMENTO E INFERÊNCIA

A proposta da Web Semântica prevê que seu conteúdo seja provido de um embasamento conceitual (as ontologias), permitindo que os conceitos representados possam ser interpretados por máquinas. Linguagens da Web Semântica como RDF e RDFSchema apresentavam uma sintaxe básica. Por ser uma linguagem baseada em Lógica Descritiva, a OWL trouxe uma maior representatividade para a representação de ontologias, tornando-se por isso recomendação oficial do W3C[8].

As Lógicas Descritivas (Description Logics – DL) [NARDI e BRACHMAN, 2003] são formalismos para representação de conhecimento baseados na lógica. Uma linguagem de DL pode ser vista como a base de um sistema de representação do conhecimento capaz de representar suas estruturas, acessando e raciocinando sobre estas estruturas.

As DLs são consideradas uma das mais importantes famílias de representação formal do conhecimento. Representam o conhecimento de determinado domínio de aplicação,

[8] World Wide Web Consortium: http://www.w3c.org

definindo primeiramente os conceitos relevantes do domínio, para só então usar estes conceitos para especificar propriedades de indivíduos que ocorrem no domínio.

As origens da DL estão nas pesquisas sobre representação do conhecimento da década de 70. Enquanto uma linha estudava um formalismo baseado na lógica, na qual a representação do mundo era feita através de predicados, a outra estudava uma representação não baseada em lógica, mas de outras formas, como redes semânticas e *frames*, que, por serem métodos específicos de determinadas áreas, não podiam ser utilizados em qualquer domínio.

Enquanto os sistemas baseados em lógica, que utilizavam a lógica de primeira ordem, eram mecanismos poderosos e genéricos de representação, os sistemas baseados em redes se mostravam mais eficientes na prática. Faltava-lhes semântica, e isto se constituiu no maior desafio das pesquisas da época.

Para solucionar este problema, surgiu a idéia de exploração de estruturas hierárquicas, dando-se semântica aos *frames* através de uma lógica de primeira ordem. Os elementos básicos da representação eram caracterizados como predicados unários e binários, na qual os primeiros denotavam o conjunto de indivíduos, e os segundos, os relacionamentos entre estes indivíduos.

Mas, embora a lógica fosse a base natural para especificar um significado para estas estruturas, *frames* e redes semânticas necessitavam de apenas uma parte dos mecanismos da lógica de primeira ordem. Além disso, diferentes características das linguagens de representação poderiam levar a diferentes fragmentos da lógica de primeira ordem.

Uma importante conseqüência disso foi o reconhecimento de que a forma típica de raciocínio usado em estruturas baseadas em representação poderia ser acoplada à técnica de raciocínio específico, sem necessariamente necessitar de provas de teoremas da lógica de primeira ordem. Raciocinando em diferentes fragmentos da lógica de primeira ordem, pode-se chegar a diferentes problemas computacionais de diferentes complexidades. Assim, iniciaram-se as pesquisas na área da Lógica Descritiva, com a idéia de deixar claro que a linguagem representativa foi usada para estabelecer a terminologia básica adotada no domínio do modelo.

3.3.1 REDES SEMÂNTICAS E LÓGICA DESCRITIVA

Como visto na seção anterior, as origens da DL estão nas redes semânticas [QUILLIAN, 1967], cuja representação se dá através de uma rede, onde os nós representam conceitos, como conjuntos ou classes de indivíduos, e os vértices representam relacionamentos que fazem ligações entre as classes.

Um exemplo de rede semântica é dado na figura 3.3; esta exibe uma taxonomia para representar conhecimento sobre conceitos como pessoa, mãe, filho, etc., representando também a generalização/especialização dos conceitos envolvidos. Por exemplo, o vértice que liga as classes *Mãe* e *Pais* indicam que todas as mães são pais de alguém. Este relacionamento é tipicamente conhecido como do tipo "é um".

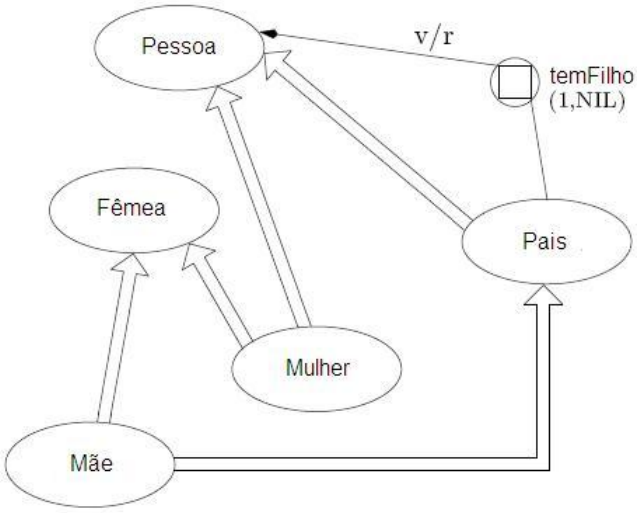

FIG. 3.3 Exemplo de rede semântica.

Os relacionamentos do tipo "é um" definem herança entre classes, fazendo com que uma classe mais específica herde as características de outra mais genérica. A DL tem capacidade de representar outros tipos de relacionamentos entre conceitos, indo além dos relacionamentos do tipo "é um". No exemplo da figura 3.3, o conceito *Pais* é ligado, através de um vértice, à classe *temFilho*. Esta classe, que também é chamada de papel, funciona como uma regra capaz de definir uma restrição de valor (representado na figura pela legenda *v/r*) e de representar a limitação dos tipos de objetos que podem suprir essa regra.

Ela também possui uma expressão de restrição, (1,NIL), que representa, respectivamente, a quantidade mínima e máxima de filhos que a regra impõe (onde o NIL representa um número infinito). Dessa maneira, pode-se observar que a regra afirma que,

se alguém for *pais*, essa pessoa deve possuir pelo menos um filho, e todas os seus filhos também devem ser *pessoas*.

Relacionamentos deste tipo apresentam também a herança entre conceitos e seus subconceitos. Por exemplo, na figura, *mãe* é um conceito que herda características de *pais*. Desta maneira, ela também irá herdar o relacionamento com a propriedade *temFilho*.

Devido a este fato, pode-se observar que podem existir várias relações implícitas entre os conceitos. Os sistemas de representação do conhecimento devem ser capazes de encontrar esses relacionamentos implícitos no modelo. Assim, os sistemas chegam ao que é chamado de inferência, sempre analisando as propriedades que envolvem a rede semântica. Porém, podem acontecer casos em que existam relações bastante complexas entre os conceitos, e isso irá trazer certa dificuldade em precisar quais os tipos de relacionamentos podem ser computados, e de que maneira esses relacionamentos podem ser analisados de forma que não tragam resultados falhos.

3.3.2 LÓGICA DESCRITIVA

Construídos com base nas idéias descritas na seção anterior, vários sistemas de base de conhecimento foram propostos, surgindo a necessidade de criação de um modelo para tratamento de inferência. A linguagem aqui descrita foi introduzida para representar a sintaxe das estruturas propostas. Para a sintaxe da DL, é utilizada uma linguagem abstrata que se parece com outros formalismos lógicos. A idéia básica da construção é utilizar dois alfabetos disjuntos de símbolos que são usados para denotar conceitos atômicos, designados por símbolos de predicados unários; e papéis atômicos, designados por símbolos de predicados binários, que são usados para designar relacionamentos entre conceitos [NARDI e BRACHMAN, 2003].

Além desse formalismo, foram definidos vários símbolos para representação de diferentes tipos de construtores. Por exemplo, interseção de conceitos, representada pelo símbolo ⊓, que é usada para restringir o conjunto de indivíduos sob consideração a apenas aos conceitos indicados. A representação se apresenta da maneira A ⊓ B. Na DL, a expressão de conceito denota todos os indivíduos que satisfazem as propriedades da expressão. Para a lógica de primeira ordem, a expressão acima pode ser vista como: A(x)

Λ B(x), onde A(x) corresponde a todos os indivíduos que satisfazem o conceito de A, e B(x), os que satisfazem B.

Uma das funcionalidades mais importantes da DL reside nos construtores para estabelecer relacionamentos entre conceitos. Os básicos são restrições de valores. Por exemplo, uma restrição de valor, da forma $\forall$R.C, que requer que todos os indivíduos que sejam relacionados através de R com algum conceito sejam descritos sobre o conceito C.

Já os conceitos dão uma interpretação da teoria dos conjuntos: um conceito é interpretado como conjuntos de indivíduos, e papéis são interpretados como conjuntos de pares de indivíduos. O domínio da interpretação pode ser escolhido arbitrariamente, e pode ser infinito. A não-finitude do domínio e a definição de um mundo aberto são características da DL no que diz respeito a linguagens de modelagem desenvolvidas no estudo de bancos de dados. Um exemplo disso pode ser visto pela simbologia apresentada anteriormente: A $\sqcap$ B, $\forall$R.C.

Ainda no exemplo da figura 3.3, temos os termos *Fêmea*, *Pessoa* e *Mulher* como conceitos atômicos e o termo *temFilho* como relacionamento. Usando os operadores interseção, união e complemento de conceitos, pode-se descrever conceitos como os de "pessoas que não são do sexo feminino" ou de "indivíduos que são do sexo masculino ou feminino", através das expressões Pessoa $\sqcap \neg$Fêmea e Fêmea $\sqcup$ Macho.

Existem ainda as restrições de relacionamentos, que podem ser de duas formas: através de quantificadores (existenciais ou universais) ou de números. A maioria das linguagens oferece as restrições de quantificadores, que permitem, por exemplo, as representações do conceito de "indivíduos que têm uma filha do sexo feminino", que se daria através de um quantificador existencial ($\exists$, cuja sintaxe seria $\exists$R.C, e a semântica, $\{a \in \Delta^I \mid \exists b.\ (a,b) \in R^I\}$), com a representação $\exists$temFilho.Fêmea. Outro exemplo seria o de "indivíduos cujas filhas são todas do sexo feminino", através de um quantificador universal ($\forall$, cuja sintaxe seria $\forall$R.C, e a semântica, $\{a \in \Delta^I \mid \forall b.\ (a,b) \in R^I \rightarrow b \in C^I\}$), com a expressão $\forall$temFilho.Fêmea".

Desta maneira, os quantificadores universal e existencial caracterizam os conceitos. Por exemplo, $\forall$temFilho.Pessoa $\sqcap$ $\exists$temFilho.Fêmea representa todos os indivíduos que têm todos os seus filhos pessoas e possuem pelo menos um filho do sexo feminino (a semântica de $(C \sqcap D)^I$ é $C^I \cap D^I$).

Outro tipo de restrição de relacionamento são as restrições de números, que definem a cardinalidade de um relacionamento. Um exemplo seria o conceito temFilho) $\sqcup$ (5<

temFilho) (com semanticas $(>=n\ R)^I = \{a \in \Delta^I \mid |\{b|\ (a,b) \in R^I\}| >= n\}$ e $(<n\ R)^I = \{a \in \Delta^I \mid |\{b|\ (a,b) \in R^I\}| < n\}$, respectivamente), que representa os indivíduos que possuem pelo menos dois filhos e menos que cinco filhos. As restrições de números são vistas como uma característica peculiar da lógica descritiva, que pode ser encontrada em algumas linguagens de modelagem para Banco de Dados.

Além dos construtores de conceitos apresentados até agora, a DL também provê construtores de relacionamentos, que estabelecem hierarquias de relacionamentos. Entretanto, o uso dessas expressões é geralmente limitado a expressar relacionamentos entre conceitos.

A expressão de regras pode se tornar poderosa se fizermos a interseção de várias delas, o que necessita definir um formalismo. O conceito temFilho ⊓ ∀temFilho.Fêmea, por exemplo, possui uma similaridade com o relacionamento *temFilha*, já que ambos indicam que qualquer que seja o filho de um indivíduo ele deve ser do sexo feminino.

Assim, é possível unir essa regra a outra e se formar um outro conceito mais completo, como o de todas as mulheres que possuem no máximo 2 filhas, que poderia ser representado da seguinte maneira: "Mulher <=2 (temFilho ⊓ ∀temFilho.Fêmea)".

As tabelas 3.1 e 3.2 a seguir serão apresentadas algumas das proposições presente na DL, bem como algumas regras de equivalência.

TAB. 3.1 Proposições equivalentes.

A	Conceitos Primitivos
R	Papéis
¬	Complemento
C ⊔ D	Conjunção
C ⊓ D	Disjunção
∀R.C	Quantificador Universal
∃R.C	Quantificador Existencial

TAB. 3.2 Proposições mais simples da Lógica Descritiva.

∃R.T	∃R
¬ (C ⊓ D)	¬C ⊔ ¬D
¬ (C ⊔ D)	¬C ⊓ ¬D
¬(∀R.C)	∃R. ¬C
¬ (∃R.C)	∀R. ¬C

3.3.3 RACIOCINANDO EM LÓGICA DESCRITIVA

A subsunção (ou “subjulgamento”) é representada na forma C $\sqsubseteq$ D, verificando se o primeiro conceito sempre representa um subconceito do segundo conceito declarado. A inferência, dentro da lógica descritiva, tem o objetivo de dizer se dados dois conceitos C e D, um "subjulga" o outro.

Estes procedimentos de inferência na DL foram, em grande parte, influenciados pelas redes semânticas, onde seus nós representam os conceitos, e os vértices, as regras da rede. Muitos algoritmos foram criados para trabalhar com inferência nas redes semânticas, dentre eles os algoritmos de subsumption. Estes têm como dado de entrada dois conceitos que são transformados inicialmente em um grafo direcionado, e a idéia é descobrir se existe algum outro grafo que pode ser embutido no primeiro. Caso fosse possível, significaria que tal grafo corresponde ao conceito mais genérico. Esse método é conhecido como comparação estrutural [LIPKIS, 1982].

Em DLs, há uma distinção entre as chamadas TBox (*terminological box*) e as ABox (*assertional box*). A TBox representa conhecimento intencional, expressando conhecimentos gerais sobre o domínio de problemas (como relacionamentos entre conceitos), enquanto a ABox contém expressa conhecimentos extensíveis, que trazem um conhecimento específico sobre os indivíduos do domínio (como relacionamentos entre indivíduos e conceitos). Por exemplo, a sentença “toda mulher é uma pessoa” pertence à TBox, enquanto “Maria é uma mulher” é uma sentença contida na ABox. Juntas, a TBox e a ABox constituem uma base de conhecimento [NARDI e BRACHMAN, 2003].

Para a descrição da sintaxe DL, utiliza-se uma notação abstrata. A letra *A* representa um conceito atômico, *R* e *S* papéis atômicos e *C* e *D* descrições de conceitos. Linguagens de descrição distinguem-se pelos construtores que oferecem. A linguagem *AL* (linguagem atributiva) [SCHMIDT-SCHAUß e SMOLKA, 1991] é uma linguagem mínima, que fundamenta uma família de representação de conhecimento. Nesta linguagem básica, descrições de conceitos são formadas de acordo com os seguintes construtores:

C, D → A | (conceito atômico)
T | (conceito universal)
⊥ | (conceito *bottom*)
¬A | (negação)
C ⊓ D | (interseção)
∃R.C | (restrição de valor)
∀R.T | (quantificador existencial)

A lógica descritiva *AL* pode ser estendida para outras mais representativas, como *SHIQ*, *SHOIQ*, *SHIN* e *SHON* [BAADER e WERNER, 2003]:

- *SHIQ*: estende a *AL* para hierarquias de papéis ($R \sqsubseteq S$), propriedades inversa (R^-) e transitiva ($R \in R_+$) e restrições de quantidade qualificadas (>= nR.C e <= nR.C) [HORROCKS et. al, 1999];
- *SHOIQ*: estende a *AL* para hierarquias de papéis, propriedades inversa e transitiva, restrições de quantidade qualificadas e declaração de nominais;
- *SHIN*: estende a *AL* para hierarquias de papéis, propriedades inversa e transitiva e restrições de quantidade não qualificadas (>= nR e <= nR);
- *SHON*: estende a *AL* para hierarquias de papéis, propriedade transitiva, declaração de nominais e restrições de quantidade não qualificadas;
- *SHOIN*: estende a *AL* para hierarquias de papéis, propriedade transitiva, inversa, declaração de nominais e restrições de quantidade não qualificadas.

A DL pode ser aplicada para a representação de conhecimento, desenvolvendo bases de conhecimento (assunto da próxima seção) e ferramentas capazes de gerar inferência em cima dessa base (a ser abordado no capítulo 5), respectivamente através de ontologias e raciocinadores de lógica descritiva.

3.4 OWL

OWL (Web Ontology Language) [DEAN et al., 2004] é uma linguagem de marcação voltada para o desenvolvimento e o compartilhamento de ontologias na Web. Derivada da linguagem DAML+OIL [CONNOLLY et al., 2001], ela foi projetada para ser utilizada em aplicações que necessitem processar o conteúdo de informações, ao invés de apenas apresentar a informação para humanos.

Neste contexto, a OWL é capaz de representar explicitamente ontologias, ou seja, o significado de um vocabulário de termos e os relacionamentos entre estes termos. Por ser

uma evolução das linguagens XML[9], RDF e RDF-S[10], OWL vai além delas, pois, por ter maior poder para expressar significado e semântica, apresenta maior capacidade de representar a interpretação da máquina sobre o conteúdo da Web [DING e FOO, 2002]. Juntamente com a habilidade da linguagem XML em definir esquemas com *tags* personalizadas e da flexibilidade da abordagem da linguagem RDF em representar dados é que se vislumbra a construção de uma Web Semântica, na qual a informação é dada com significado explícito, tornando mais fácil para máquinas processar e integrar a informação disponível na Web de forma automática [DEAN et al., 2004].

A linguagem OWL apresenta três sublinguagens, cuja expressividade aumenta progressivamente: OWL Lite, OWL-DL e OWL Full. A primeira suporta as necessidades primárias do usuário: classificação hierárquica e restrições simples. Permite apenas valores de cardinalidades 0 ou 1. A segunda, também chamada *OWL Description Logics*, oferece uma máxima expressividade, enquanto mantém a completude computacional (existe a garantia de que todas as conclusões sejam computadas) e é capaz de realizar deduções, garantindo que todas as operações computacionais sejam terminadas em um tempo finito. A OWL DL inclui todas as construções da linguagem OWL, porém podem ser usadas apenas sob certas restrições (por exemplo, enquanto uma classe é uma subclasse de outra classe, ela não pode ser uma instância de uma outra classe). Finalmente, a terceira é direcionada para usuários que queiram uma máxima expressividade e liberdade sintática da RDF, mas sem garantias computacionais para sistemas de dedução automática. De fato, ela permite o uso do vocabulário completo, tanto da OWL quanto da RDF.

OWL é um componente de atividade da Web Semântica, localizado na camada de ontologia na arquitetura definida por Berners-Lee (figura 3.1), que tem como objetivo tornar os recursos da Web mais acessíveis para processos automatizados, permitindo adicionar informação sobre os recursos que descrevem conteúdo na Web. Como a Web Semântica é inerentemente distribuída, a OWL deve permitir que informações sejam reunidas de diferentes fontes.

A linguagem OWL-DL define:

- classes e subclasses, o que torna possível expressar hierarquias;
- propriedades e subpropriedades, que definem relacionamentos entre indivíduos;

[9] http://www.w3.org/XML/
[10] http://www.w3.org/RDF/

- as propriedades funcional, inversa, funcional inversa, transitiva e simétrica, que dão maior riqueza semântica às ontologias explorando relacionamentos não descritos na ontologia, mas implicitamente pressupostos através destas propriedades;
- restrições às propriedades, que definem uma faixa de propriedades em contextos específicos, que determina que todos (ou pelo menos um) elementos associados por um relacionamento pertencem a determinada classe; ou ainda de cardinalidade, que determina a quantidade de elementos que determinada propriedade pode agregar.

Uma discussão mais detalhada sobre as particularidades da OWL pode ser encontrada em Mattos et al. [MATTOS et al., 2005].

3.5 LINGUAGENS DE REGRAS PARA ONTOLOGIAS

Como visto na seção 3.4, através da OWL é possível definir classes, propriedades e instâncias, além de prover informações sobre estas instâncias. Entretanto, a linguagem apresenta limitações para definir relacionamentos entre propriedades. De fato, como a OWL não possui um construtor para realizar composições, torna-se impossível capturar relacionamentos entre propriedades, como no exemplo de um relacionamento entre a composição das propriedades "pai" e "irmão", gerando a propriedade "tio". Além disso, a OWL apresenta limitações também na garantia da decidibilidade no processamento das sublinguagens OWL DL e OWL Lite [HORROCKS et al, 2004].

De forma a suprir algumas dessas limitações, Kifer et al. [KIFER et al., 2005] e Horrocks et al. [HORROCKS et al., 2005] propuseram a incorporação de regras à linguagem, provendo-a de maior robustez.

Para a representação de regras ao longo desta seção, será utilizada uma forma relativamente informal, representada da seguinte maneira:

$$\texttt{antecedente} \rightarrow \texttt{consequente}$$

onde tanto `antecedente` quanto `consequente` são expressos por conjunções de átomos representados por $a_1 \wedge \ldots \wedge a_n$. Variáveis são representadas através da convenção padrão, com um sinal de interrogação antecedendo-a (por exemplo, `?a`).

Assim, através de regras, é possível definir:

- composição de propriedades, onde, através de uma propriedade, é possível definir que o irmão do pai de um indivíduo é seu tio;

```
temPai(?a,?b) ∧ temIrmao(?b,?c) → temTio(?a,?c)
```

- herança sobre indivíduos, que possibilita definir que a doença de uma parte do corpo é uma doença do corpo.

```
parte_de(?a,?b) ∧ doença(?b,?c) → doença(?a,?c)
```

Dentre as linguagens submetidas ao W3C para se tornarem recomendações oficiais, estão a SWRL, a SWRL-FOL e a WRL. Elas trazem à OWL construtores de regras simples, através dos quais é possível declarar que uma afirmativa implica em outra.

3.5.1 SWRL

A SWRL (Semantic Web Rule Language) [HORROCKS et al, 2004] é uma linguagem de regras [LLOYD, 1987] para a Web Semântica, baseada em uma combinação da OWL DL e OWL Lite (sublinguagens da OWL) com a RuleML (sublinguagem da Rule Markup Language). Ela propõe uma sintaxe abstrata em alto nível para definir cláusulas de Horn[11] tanto em OWL DL quanto em OWL Lite, oferecendo assim uma maneira formal de se representar regras em ontologias OWL.

Uma ontologia OWL, em sua sintaxe abstrata, é formada por uma seqüência de axiomas e fatos. Axiomas podem ser de vários tipos, como *subClass* ou *equivalentClass*. A SWRL propõe estendê-los para axiomas de regra.

```
axioma ::= regra
```

Um axioma de regra consiste de um antecedente (corpo) e um conseqüente (cabeça), onde cada um deles consiste de um conjunto de átomos. A um axioma de regras pode também ser atribuída uma referência URI (Uniform Resource Identifier), que funciona como identificadora da regra. Seu significado pode ser definido como: se as condições

[11] Conjunto de predicados unidos por conjunções e que implicam em um único predicado.

especificadas no antecedente ocorrerem, então as condições especificadas no conseqüente também ocorrerão, conforme definido a seguir.

```
regra ::= 'Implica(' [referenciaURI] {anotaçao} antecedente consequente
')'
antecedente ::= 'Antecedente(' { atomo } ')'
consequente ::= 'Consequente(' { atomo } ')'
```

Tanto o antecedente quanto o conseqüente consistem de zero ou mais átomos. Um antecedente vazio é tratado como uma verdade trivial (satisfeito sob qualquer condição), razão pela qual o conseqüente será satisfeito em qualquer interpretação. Já um conseqüente vazio é tratado como uma falsidade trivial (não é satisfeito em qualquer interpretação) e, portanto, o antecedente não deverá ser satisfeito em qualquer interpretação. Átomos múltiplos são considerados uma conjunção.

Nas regras, os átomos podem ser das formas C(x), P(x,y), sameAs(x,y), differentFrom(x,y) ou uma função built-in (r, x...), onde C é uma descrição em OWL ou um valor de restrição, P é uma propriedade em OWL, r é uma relação built-in (predicados que permitem ao usuário definir métodos para serem utilizados nas regras) e x e y podem ser variáveis, indivíduos OWL ou valores OWL. Em um átomo C(x), x deve ser uma instância da classe ou um valor de restrição. Em um átomo P(x,y), x se relaciona a y através de uma propriedade P. Em um átomo sameAs(x,y), x e y são interpretados como o mesmo objeto, enquanto em differentFrom(x,y), eles são objetos diferentes. Um átomo é definido tal como:

```
atomo ::= descriçao '(' i-objeto ')'
        | dataRange '(' d-objeto ')'
        | individualvaluedPropertyID '(' i-objeto i-objeto ')'
        | datavaluedPropertyID '(' i-objeto d-objeto ')'
        | sameAs '(' i-objeto i-objeto ')'
        | differentFrom '(' i-objeto i-objeto ')'
        | builtIn '(' builtinID { d-objeto } ')'
builtinID ::= URIreference
```

Átomos podem se referir a indivíduos, literais de dados, variáveis representando indivíduos (i-variáveis) ou dados (d-variáveis). Variáveis são tratadas como quantificadores universais, com seu escopo limitado a uma dada regra. Apenas variáveis que ocorram no antecedente de uma regra podem ocorrer no conseqüente, conforme definido a seguir.

```
i-objeto ::= i-variavel | IDindividuo
d-objeto ::= d-variavel | literal
```

```
i-variavel ::= 'I-variavel(' referenciaURI ')'
d-variavel ::= 'D-variavel(' referenciaURI ')'
```

A semântica do modelo teórico para a linguagem de regras SWRL é uma extensão da semântica para OWL DL. A idéia básica é definida por ligações, extensões das interpretações OWL que também mapeiam variáveis para elementos do domínio. Uma regra é satisfeita por uma interpretação se e somente se toda ligação que satisfizer o antecedente satisfizer também o conseqüente. Assim, uma interpretação satisfaz uma ontologia se e somente se satisfizer todo axioma (incluindo regras) e fatos na ontologia.

Considerando-se:

- R: um conjunto de recursos;
- LV $\subseteq$ R: um conjunto de literais;
- EC: um mapeamento de classes e tipos de dados para subconjuntos de R e LV, respectivamente;
- ER: um mapeamento das propriedades para relações binárias;
- L: um mapeamento de tipos e literais para elementos de LV;
- S: um mapeamento de nomes de indivíduos para elementos de EC (owl:Thing).

tem-se que um átomo é satisfeito por uma interpretação sobre as condições dadas na Tabela de Condições de Interpretação (TAB 3.3), onde:

- C: uma descrição OWL DL;
- D: um valor de dado OWL DL;
- P: uma propriedade cujo valor é um indivíduo;
- Q: uma propriedade cujo valor é um dado;
- F: uma relação built-in;
- x e y: indivíduos;
- z: uma variável ou um valor de dado.

Um uso simples destas regras pode ser a afirmação de que a combinação das propriedades `temPai` e `temIrmao` implicam na propriedade `temTio`. Em uma linguagem

formal, como visto anteriormente, esta regra seria escrita conforme mostra a seguinte regra:

```
temPai(?a,?b) ∧ temIrmao(?b,?c) → temTio(?a,?c)
```

Na sintaxe abstrata, esta mesma regra seria escrita da forma a seguir:

```
Implica(Antecedente(temPai(I-variavel(a) I-variavel(b))
                    temIrmao(I-variavel(b) I-variavel(c)))
        Consequente(temTio(I-variavel(a) I-variavel(c))))
```

TAB. 3.3 Tabela de condições de interpretação da SWRL.

Átomo	**Condição na Interpretação**
C(x)	$S(x) \in EC(C)$
D(z)	$S(z) \in EC(D)$
P(x,y)	$<S(x),S(y)> \in ER(P)$
Q(x,z)	$<S(x),L(z)> \in ER(Q)$
sameAs(x,y)	$S(x) = S(y)$
differentFrom(x,y)	$S(x) \neq S(y)$
builtIn(r,z_1,...,z_n)	$<S(z_1),...,S(z_n)> \in D(f)$

Regras também podem ser expressas em XML, tendo por base a RuleML[12], sendo assim uma extensão da sintaxe OWL DL XML. Além disso, regras também podem ser definidas estendendo-se a linguagem RDF XML [RDF, 2004].

Para demonstrar regras formalmente em OWL, a SWRL apresenta uma sintaxe XML, baseada nas já existentes OWL XML Presentation Syntax [HORI et al., 2003] e RuleML. Através desta extensão, é possível:

- utilizar classes OWL como predicados de regras;
- alterar regras e axiomas da ontologia com facilidade.

Assim, ontologias podem incluir, além dos axiomas OWL, axiomas de regras e declarações de variáveis. A nova sintaxe é definida através do prefixo `swrlx`.

[12] http://www.ruleml.org/

Declarações de variáveis são afirmações sobre variáveis, indicando que uma dada URI deve ser utilizada na ontologia como variável, podendo-se, opcionalmente, adicionar comentários. Por exemplo:

```
<swrl:Variable swrl:name=”a” />
```

define que a URI `a` deve ser considerada uma variável.

Axiomas de regras são similares às subclasses OWL. Possuem um componente antecedente (`swrl:antecedent`) e um conseqüente (`swrl:consequent`). Tanto o antecedente quanto o conseqüente de uma regra são listas de átomos, podendo ser considerados como uma conjunção destes átomos. Átomos, por sua vez, podem ser formados por predicados unários (classes) ou binários (propriedades), igualdades e desigualdades.

Um átomo classe é descrito por um nome de indivíduo ou um nome de variável, podendo também apresentar uma descrição complexa utilizando combinações booleanas, restrições, etc. Por exemplo, um átomo que consiste de um nome de classe (no caso, Pessoa) é definido da seguinte forma:

```
<swrlx:classAtom>
  <owlx:Class owlx:name="Pessoa" />
  <ruleml:var>a</ruleml:var>
</swrlx:classAtom>
```

Já um átomo apresentando uma composição, onde os elementos da classe `“Pessoa”` possuem pelo menos um dos pais pertencentes à classe `”Medico”`, é definido da seguinte maneira:

```
<swrlx:classAtom>
  <owlx:IntersectionOf>
    <owlx:Class owlx:name="Pessoa" />
    <owlx:ObjectRestriction owlx:property="temPai">
      <owlx:someValuesFrom owlx:class="Medico" />
    </owlx:ObjectRestriction>
  </owlx:IntersectionOf>
  <ruleml:var>b</ruleml:var>
</swrlx:classAtom>
```

Os átomos propriedade consistem de um nome de propriedade e dois elementos que podem ser nomes de indivíduos, nomes de variáveis ou valores de dados. Nos casos onde o segundo elemento é o nome de um indivíduo, a propriedade deve ser do tipo *individual-valued*, como no seguinte exemplo, onde se define que o segundo elemento é um indivíduo – no caso, `”Joao”`:

```
<swrlx:individualPropertyAtom  swrlx:property="temPai">
  <ruleml:var>a</ruleml:var>
  <owlx:Individual owlx:name="Joao" />
</swrlx:individualPropertyAtom>
```

Já nos casos em que o segundo elemento for o valor de um dado, a propriedade deve ser do tipo *data-valued*. No exemplo a seguir, o segundo elemento é um valor de dado, o inteiro "4":

```
<swrlx:datavaluedPropertyAtom  swrlx:property="serie">
  <ruleml:var>b</ruleml:var>
  <owlx:DataValue owlx:datatype="&xsd;int">4</owlx:DataValue>
</swrlx:datavaluedPropertyAtom>
```

Existem também casos de átomos de indivíduos iguais, que definem a igualdade entre conjuntos de indivíduos e nomes de variáveis. No exemplo a seguir, definimos que as variáveis `a` e `b` e os indivíduos `Lattes` e `Celso Lattes` referem-se ao mesmo indivíduo.

```
<swrlx:sameIndividualAtom>
  <ruleml:var>a</ruleml:var>
  <ruleml:var>b</ruleml:var>
  <owlx:Individual owlx:name="Lattes" />
  <owlx:Individual owlx:name="Celso_Lattes" />
</swrlx:sameIndividualAtom>
```

Os átomos de indivíduos diferentes são representados de maneira análoga.

A partir destas definições, é possível definir a regra:

```
temPai(?a,?b) ∧ temIrmao(?b,?c) → temTio(?a,?c)
```

da seguinte maneira:

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#exemplo1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="temPai">
      <ruleml:var>a</ruleml:var>
      <ruleml:var>b</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="temIrmao">
      <ruleml:var>b</ruleml:var>
      <ruleml:var>c</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="temTio">
```

```
      <ruleml:var>a</ruleml:var>
      <ruleml:var>c</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Neste exemplo, o antecedente (ou corpo) da regra é definido através da *tag* `<ruleml:_body>`, enquanto o conseqüente (ou cabeça) é definido pela *tag* `<ruleml:_head>`.

Entretanto, a integração entre OWL DL e regras que a SWRL propõe é obtida simplesmente com o uso de conceitos e papéis em regras como átomos unários e binários, respectivamente. Além disso, cada uma destas extensões pode levar à indecidibilidade em um caso onde se queira verificar a consistência de uma base de conhecimento através de um programa validador. Assim, consultas sobre estas bases de conhecimento seriam indecidíveis. Motik et al. [2004] propõem uma solução para este problema da SWRL restringindo as regras a expressões seguras.

3.5.2 SWRL-FOL

A SWRL-FOL (Semantic Web Rule Language First-Order Logic) [PATEL-SCHNEIDER, 2005] estende a SWRL, incluindo um axioma para fórmulas de primeira ordem sobre predicados unários e binários. Uma sintaxe de alto nível é definida para estender a sintaxe abstrata OWL, adicionando a ela declarações que podem conter conectivos lógicos padrão, tais como negação e disjunção. Isso permite a adição da expressão de certos formalismos às ontologias ou bases de conhecimento.

Assim, na SWRL-FOL são permitidas proposições (conjunção e disjunção entre átomos, negação de um átomo, implicação e a equivalência entre dois átomos ou conjunto de átomos) e utilização dos quantificadores universal e existencial.

Uma ontologia SWRL, em sua sintaxe abstrata, contém uma seqüência de axiomas e fatos, incluindo regras SWRL. A SWRL-FOL estende a SWRL com axiomas de declarações contendo sentenças de primeira ordem.

```
axioma ::= declaraçao
```

Declarações podem ser identificadas com uma referência URI através de um identificador. Estas declaram sentenças de primeira ordem como fórmulas sem variáveis livres. Como as variáveis são definidas usando-se quantificadores de tipo, isso significa que cada variável em uma declaração deve ser definida sobre um tipo.

O axioma a seguir define uma declaração de forma abstrata.

```
declaraçao ::= 'Declaraçao(' [ referenciaURI ] {anotaçao} foformula
{foformula} ')'
```

As fórmulas de primeira ordem (representadas por `foformula`) nas declarações são ajustadas na maneira usual de lógica de primeira ordem, com algumas extensões, como a que permite que conjunções ou disjunções tenham qualquer número de operadores. As fórmulas atômicas são semelhantes aos átomos no SWRL, como vistas a seguir:

```
foformula ::= atomo
            | 'e(' { foformula } ')'
            | 'ou(' { foformula } ')'
            | 'neg(' foformula ')'
            | 'implica(' foformula foformula ')'
            | 'equivale(' foformula foformula ')'
            | 'paratodo(' variavel { variavel } foformula ')'
            | 'existe(' variavel { variavel } foformula ')'
```

Os construtores `paratodo` e `existe` possuem tipos de variáveis que restringem a possibilidade de mapeamento de uma variável, que podem ser descrições ou restrições de valor de dado (`dataRange`), conforme mostrado a seguir.

```
variavel ::= 'I-variavel(' referenciaURI descriçao ')'
           | 'D-variavel(' referenciaURI dataRange ')'
```

Assim como na SWRL, a semântica do modelo teórico para a SWRL-FOL é uma extensão da semântica para OWL DL. Também existem as ligações, que mapeiam variáveis para elementos do domínio.

A semântica assim obtida é uma semântica padrão da lógica de primeira ordem. A única ressalva é a separação da semântica de literais, tipos de dados ou outros elementos sem tipo por outras entidades. Desta forma, a lógica obtida é uma lógica de duas classificações, onde uma classificação (literal) é construída sobre predicados (predicados de tipos de dados e outras construções).

Assertivas são interpretadas usando-se ligações da SWRL. Uma associação é definida por B(I), onde: I é uma interpretação abstrata OWL, estendendo I tal que S

mapeie i-variáveis para elementos de EC (owl:thing) e L mapeia d-variáveis para elementos LV. Uma fórmula é satisfeita por uma associação (escrita da forma B(I) ⇒ F) sob as condições dadas na Tabela de Condições de Interpretação.

TAB. 3.4 Tabela de condições de interpretação da SWRL-FOL.

Átomo	Condição na Interpretação
and(C_1 ... C_n)	B(I) ⇒ C_i, para cada C_i
or(C_1 ... C_n)	B(I) ⇒ C_i, para algum C_i
neg(C)	B(I) não satisfaz C
implies(C_1 C_2)	B(I) ⇒ or(neg(C_1) C2)
equivalent(C_1 C_2)	B(I) ⇒ and(implies(C_1 C_2) implies(C_2 C_1))
forall(V_1 ... V_n C)	B'(I) ⇒ C, para todas as associações B' que sejam as mesmas que B exceto por V_1 ...V_n
exists(V_1 ... V_n C)	B'(I) ⇒ C, para alguma associação B' que seja a mesma que B exceto por V_1 ... V_n

Assim, a mesma regra "temTio" apresentada em SWRL no item anterior:

```
temPai(?a,?b) ∧ temIrmao(?b,?c) → temTio(?a,?c)
```

teria sua asserção definida em XML da seguinte maneira:

```
<Assertion owlx:name="Exemplo">
  <Implies>
    <And>
      <swrlx:individualPropertyAtom  swrlx:property="temPai">
        <ruleml:Var>a</ruleml:Var>
        <ruleml:Var>b</ruleml:Var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom  swrlx:property="temIrmao">
        <ruleml:Var>b</ruleml:Var>
        <ruleml:Var>c</ruleml:Var>
      </swrlx:individualPropertyAtom>
    </And>
      <swrlx:individualPropertyAtom  swrlx:property="temTio">
        <ruleml:Var>a</ruleml:Var>
        <ruleml:Var>c</ruleml:Var>
      </swrlx:individualPropertyAtom>
  </Implies>
</Assertion>
```

Na tabela de interpretações, pode-se observar que o átomo implies(C_1 C_2) é interpretado como B(I) ⇒ or (neg (C_1) C2). Nesta regra, é declarado que, se um indivíduo "b" for pai de "a" e irmão de "c", então "a" relaciona-se através do relacionamento "temTio" com o indivíduo "c".

Esta maior representatividade da SWRL-FOL traz, contudo, um custo: a perda de desempenho de processabilidade por máquina em alguns casos. Entretanto, falta ainda à linguagem a possibilidade de representar predicados n-arios e funções. Outro problema potencial para a extensão desta linguagem é o fato de sua adoção ser condicionada a uma licença da Lucent Technologies, empregadora do autor da proposta.

3.5.3 WRL

A WRL (Web Rule Language) [DE BRUIJIN et al., 2005], derivada do componente para ontologias da linguagem para modelagem de serviços Web WSML (WSML), é uma linguagem de ontologias baseada em regras. Seus elementos básicos consistem de conceitos, relacionamentos, indivíduos e axiomas.

Assim como a SWRL, a WRL pode ser expressa em XML, com base na RuleML. Espera-se que futuramente WRL, WSRL, SWSL e RuleML sejam convertidas para uma única sintaxe em XML.

A sintaxe da WRL consiste de duas partes principais: a sintaxe conceitual e a sintaxe de expressão lógica. A sintaxe conceitual é usada para a modelagem de ontologias, e as expressões lógicas para refinar estas definições com o uso de regras.

A WRL e a OWL são duas linguagens de ontologias complementares, que dão suporte a diferentes casos na Web Semântica. Um ponto forte da OWL é a processabilidade através de raciocinadores que suas ontologias possuem, enquanto a WRL é focada em conjuntos para a checagem da consistência de seus dados (de acordo com suas restrições) e para a especificação e processamento de regras definidas.

Sendo assim, a regra:

```
temPai(?a,?b) ∧ temIrmao(?b,?c) → temTio(?a,?c)
```

seria representada na sintaxe XML da WRL da seguinte maneira:

```
<wrl xmlns="http://www.wsml.org/wsml/wrl-syntax#"
   xmlns:ruleml="http://www.ruleml.org/0.89/xsd"
   variant="http://www.wsml.org/wsml/wrl-syntax/wrl-core">
 <ontology name="http://www.example.org/ex1">
 <Implies kind="fo">
  <body>
    <And>
```

```
      <relation name="http://www.example.org/ex1#temPai">
        <var>a</var>
        <var>b</var>
      </relation>
      <relation name="http://www.example.org/ex1#temIrmao">
        <var>b</var>
        <var>c</var>
      </relation>
    </And>
  </body>
  <head>
    <relation name="http://www.example.org/ex1#temTio">
      <var>a</var>
      <var>c</var>
    </relation>
  </head>
 </Implies>
 </ontology>
</wrl>
```

Neste exemplo, os relacionamentos "temPai" e "temIrmao" são definidos no corpo da regra, relacionando respectivamente as variáveis "a" com "b" e "b" com "c". Na cabeça da regra, como conseqüência destas condições, temos o relacionamento "temTio", relacionando as variáveis "a" e "c".

A WRL apresenta três variantes, denominadas: Core, Flight e Full. A WRL-Core define características básicas de interoperabilidade entre a WRL e a OWL. A WRL-Flight é uma linguagem de regras baseada em um subconjunto de Datalog, a linguagem F-Logic. Já a WRL-Full é uma linguagem de regras com símbolos de funções e negação sobre uma semântica bem definida. Cada uma destas linguagens possui uma semântica própria.

3.5.3.1 WRL-CORE

Uma base de conhecimento WRL-Core é uma coleção de fórmulas escritas na linguagem de expressão lógica WRL, que é o resultado da aplicação de um pré-processamento sobre uma ontologia WRL-Core. Define-se a semântica da WRL-Core através do mapeamento para Datalog utilizando-se a função de mapeamento π.

A tabela 3.5 a seguir apresenta a semântica WRL-Core através de um mapeamento direto para Datalog. Nesta tabela, id é um identificador, dt é um identificador de tipo de dado e X pode ser uma variável ou um identificador. Cada ocorrência de x e y representa a introdução de novas variáveis.

TAB. 3.5 Semântica da WRL-Core.

WRL	Datalog
π(*head* **impliedBy** *body*.)	π(*head*) ← π(*body*)
π(*head* **implies** *body*.)	π(*body*) ← π(*head*)
π(*head* **equivalent** *body*.)	π(*head*) ← π(*body*) π(*body*) ← π(*head*)
π(*lexpr* **or** *rexpr*)	π(*lexpr*) ∨ π(*rexpr*)
π(*lexpr* **and** *rexpr*)	π(*lexpr*) ∧ π(*rexpr*)
π(*X1* **memberOf** *id2*)	*id2*(*X1*)
π(*id1* **subConceptOf** *id2*)	*id2*(x) ← *id1*(x)
π(*X1*[*id2* **hasValue** *X2*])	*id2*(*X1*,*X2*)
π(*id1*[*id2* **impliesType** *id3*])	*id3*(y) ← *id1*(x) ∧ *id2*(x,y)
π(*id1*[*id2* **ofType** *dt*])	← *id1*(x) ∧ *id2*(x,y) ∧ not *dt*(y)
π($p(X_1,...,X_n)$)	$p(X_1,...,X_n)$

3.5.3.2 WRL-FLIGHT

A semântica da WRL-Flight é definida através do mapeamento do fragmento Datalog da F-Logic [KIFER et. al, 1995] usando a função de mapeamento π. Nesta tradução para F-Logic, utiliza-se a seguinte notação:

- *A*:*B* denota que *A* é um membro da classe *B*.
- *A*::*B* denota que *A* é uma subclasse de *B*.
- *A*[*B* ->> *C*] denota que o objeto *A* tem o valor *C* para o atributo *B*.
- *A*[*B* =>> *C*] denota que a classe *A* tem um atributo *B* restrito a *C*.

Conceitos, instâncias e atributos são interpretados como objetos na F-Logic. São necessárias regras auxiliares para garantir a correta interpretação das sentenças de F-Logic traduzidas.

A semântica da WRL-Flight é apresentada na tabela a seguir. Nela, X define uma variável ou identificador, '=' é o operador de unificação e '!=' é o símbolo de não-igualdade. O símbolo '$\leftarrow_{LT}$' é uma implicação de Lloyd-Topor [LLOYD e TOPOR, 1984], que é eliminada pelas fórmulas indicadas na tabela 3.6.

TAB. 3.6 Semântica da WRL-Flight.

WRL	F-Logic
π(**!-** *body*.)	← π(*body*)
π(*head* **:-** *body*.)	π(*head*) ← π(*body*)
π(*lexpr* **impliedBy** *rexpr*.)	π(*lexpr*) ←$_{LT}$ π(*rexpr*)
π(*lexpr* **implies** *rexpr*.)	π(*rexpr*) ←$_{LT}$ π(*lexpr*)
π(*lexpr* **equivalent** *rexpr*.)	(π(*rexpr*) ←$_{LT}$ π(*lexpr*)) ∧ (π(*lexpr*) ←$_{LT}$ π(*rexpr*))
π(*lexpr* **or** *rexpr*)	π(*lexpr*) ∨ π(*rexpr*)
π(*lexpr* **and** *rexpr*)	π(*lexpr*) ∧ π(*rexpr*)
π(**naf** *expr*)	not π(*expr*)
π(*X1* **memberOf** *X2*)	*X1*:*X2*
π(*X1* **subConceptOf** *X2*)	*X1*::*X2*
π(*X1*[*X2* **hasValue** *X3*])	*X1*[*X2* ->> *X3*]
π(*X1*[*X2* **ofType** *X3*])	*X1*[*X2* =>> *X3*]
π(*X1*[*X2* **impliesType** *X3*])	_impliestype(*X1*,*X2*,*X3*)
π($p(X_1,...,X_n)$)	$p(X_1,...,X_n)$
π($X_1 = X_2$)	$X_1 = X_2$
π(X_1 != X_2)	X_1 != X_2

3.5.3.3 WRL-FULL

A semântica da WRL-Full é definida da mesma maneira que a da WRL-Flight. A única diferença é que a semântica da WRL-Full não é definida através de um mapeamento para Datalog, mas sim através de um mapeamento para a Programação Lógica completa, com símbolos de função e permitindo regras não-seguras.

Uma base de conhecimento WRL-Full é uma coleção de fórmulas escritas na linguagem de expressão lógica, que é resultado do pré-processamento de uma ontologia WRL-Full. A linguagem é definida através de um mapeamento do fragmento das cláusulas de Horn da F-Logic utilizando a função de mapeamento π, conforme pode ser visto na tabela 3.7.

3.5.4 COMPARAÇÃO ENTRE SWRL E WRL

Embora tenham abordagens semelhantes, as linguagens SWRL e WRL possuem enfoques diferentes. A WRL é baseada em linguagens de regras padrão, construída com base em Programação Lógica e Bancos de Dados Dedutivos e, sintaticamente, restringe-se a cláusulas de Horn, adicionando negações não atômicas.

TAB. 3.7 Semântica da WRL-Full.

WRL	F-Logic
π(**!-** *body*.)	← π(*body*)
π(*head* **:-** *body*.)	π(*head*) ← π(*body*)
π(*lexpr* **impliedBy** *rexpr*.)	π(*lexpr*) ←$_{LT}$ π(*rexpr*)
π(*lexpr* **implies** *rexpr*.)	π(*rexpr*) ←$_{LT}$ π(*lexpr*)
π(*lexpr* **equivalent** *rexpr*.)	(π(*rexpr*) ←$_{LT}$ π(*lexpr*)) ∧ (π(*lexpr*) ←$_{LT}$ π(*rexpr*))
π(*lexpr* **or** *rexpr*)	π(*lexpr*) ∨ π(*rexpr*)
π(*lexpr* **and** *rexpr*)	π(*lexpr*) ∧ π(*rexpr*)
π(**exists** *X1,...,Xn* (*expr*))	∃ *X1,...,Xn* (π(*expr*))
π(**forall** *X1,...,Xn* (*expr*))	∀ *X1,...,Xn* (π(*expr*))
π(**naf** *expr*)	not π(*expr*)
π(*X1* **memberOf** *X2*)	*X1*:*X2*
π(*X1* **subConceptOf** *X2*)	*X1*::*X2*
π(*X1*[*X2* **hasValue** *X3*])	*X1*[*X2* ->> *X3*]
π(*X1*[*X2* **ofType** *X3*])	*X1*[*X2* =>> *X3*]
π(*X1*[*X2* **impliesType** *X3*])	impliestype(*X1*,*X2*,*X3*)
π($p(X_1,...,X_n)$)	$p(X_1,...,X_n)$
π($X_1 = X_2$)	$X_1 = X_2$
π(X_1 != X_2)	X_1 != X_2

Já a SWRL é uma extensão da OWL DL, que adiciona características importantes ao padrão. Porém não é uma linguagem de regras padrão, já que a OWL DL não é uma linguagem voltada para regras, e apenas um subconjunto pode ser traduzido para regras.

Uma linguagem não pode ser considerada subconjunto da outra. A SWRL, por ser baseada na Web Semântica, assume semânticas de mundo aberto, enquanto a WRL assume o mundo fechado associado a Datalog. Assim, o envolvimento em um mundo OWL/RDF é diferente do WRL, e, portanto a SWRL não pode ser vista como um subconjunto da WRL. Se não é mundo aberto, não é Web Semântica, pois viola as afirmações da base onde provem RDF [RDF, 1999], RDFS [RDFS, 2004] e OWL [DEAN et al., 2004].

Um exemplo disso seria a seguinte tupla RDF:

```
<#maria> <#temFilho> <#pedro>
```

Se fosse perguntado se Maria tem apenas um filho, a resposta poderia ser "não" em uma semântica de mundo aberto RDF, mas seria "sim" em uma semântica de Datalog de mundo fechado, já que nesta hipótese só é considerado verdadeiro tudo o que existe explicitamente na base de dados.

A tabela 3.8 a seguir faz uma comparação entre os aspectos principais de cada linguagem.

TAB. 3.8 Comparação entre as Linguagens de Representação de Regras para Ontologias.

	SWRL	**WRL**
Baseada em	Cláusulas de Horn	Programação Lógica
Relação com OWL	Extensão da OWL DL	Possui um subconjunto em comum com a OWL DLP
Origem	OWL DL e RuleML	WSML
Negação por falha	Sim, inclusive na cabeça da regra	Apenas na WRL-Flight e WRL-Full (exceto na cabeça da regra)
Quantificadores	Sim, inclusive na cabeça da regra	Apenas na WRL-Full (exceto na cabeça da regra)
Hipótese	Mundo aberto	Mundo fechado

A SWRL tem como principal atrativo o fato de se valer de tecnologias já disponíveis, como RDF e OWL para se estabelecer, enquanto a WRL, como proposto em [KIFER et al., 2005], faz parte de um novo paradigma, onde as novas tecnologias da Web Semântica não precisam necessariamente ser totalmente compatíveis com as já estabelecidas.

Por estar em um estágio de desenvolvimento mais avançado, a utilização da SWRL é, a curto prazo, mais benéfica. Afinal, esta é baseada em uma linguagem já estabelecida (a OWL) e inclusive já apresenta *plug-ins* para a inserção de regras em ontologias através de ferramentas editoras de ontologias, como o Protégé [NOY et al., 2000]. Entretanto, sua dependência da OWL potencializa eventuais problemas, que a WRL visa reduzir com sua proposta de menor dependência da OWL. Este é possivelmente o caminho a ser adotado pela Web Semântica futuramente.

3.6 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo abordar algumas das arquiteturas para Web Semântica e linguagens atualmente existentes para a construção de ontologias com suporte a regras, além de apresentar uma análise sucinta sobre Lógicas Descritivas e as linguagens da Web Semântica que ela fundamenta: a OWL e as linguagens de regras.

Dentre as linguagens de regras, nenhuma ainda foi adotada como recomendação oficial pelo W3C. Por ser desenvolvida há mais tempo, a SWRL já conta com ferramentas de edição e de raciocínio. Por isso, ela desponta em um primeiro momento como a tecnologia mais indicada a ser adotada neste trabalho.

No ROSA, LOs representam conteúdos instrucionais, acrescidos de um conjunto de propriedades e um conjunto de associações (ou predicados) que expressam os relacionamentos que um LO tem com outros LOs. Entretanto, o ROSA não é capaz de declarar regras, fato que aumentaria muito o poder de expressividade na representação dos objetos armazenados no sistema. Uma maneira de fazer com que o ROSA expanda sua representatividade seria através da utilização de uma linguagem de ontologia apta a representar regras. No entanto, essa funcionalidade requer a extensão do metamodelo ROSA, aqui denominado ROSA^{+}, que, inserido numa visão de arquitetura de três camadas, permite o acesso aos seus objetos em níveis distintos de abstração. A arquitetura e modelos propostos são temas do próximo capítulo.

4 EXTENSÃO DO MODELO DE DADOS ROSA PARA REPRESENTAÇÃO DE REGRAS

Neste capítulo será apresentada a extensão do modelo ROSA visando a representação de regras nesse sistema. A seção 4.1 descreve a arquitetura do ROSA$^+$ através de uma abordagem em camadas, que permite representar e visualizar o sistema através de seus modelos em vários níveis. Já a seção 4.2 apresenta a descrição do modelo de dados ROSA estendido, onde algumas classes foram adicionadas para contemplar a representação de regras. A partir da arquitetura definida e feita então para cada nível de abstração, a especificação de suas classes, relacionamentos e regras apresentada na seção 4.3.

4.1 ARQUITETURA ROSA$^+$

Um sistema de gerenciamento de conhecimento deve ser desenvolvido sobre modelos que representem todas as formas de conhecimento. Um modelo de dados deve ser capaz de representar tanto o aspecto estático de um conhecimento (objetos e relacionamentos) quanto o aspecto dinâmico (processos e procedimentos). Além disso, deve incluir também a possibilidade de representar regras e restrições, aumentando o poder de expressividade do modelo [GERBÉ e KERHERVE, 1998].

Para contemplar esses três aspectos de um modelo, é necessário representar níveis de conhecimento distintos, que melhor se expressam através de três diferentes níveis: modelos, metamodelos e meta-metamodelos.

O modelo corresponde ao uso de representações do conhecimento baseadas em sua função principal, ou seja, armazenar e validar conhecimento e comportamento, simular ou aplicar comportamentos e armazenar restrições.

Já o metamodelo corresponde ao uso de modelos não para armazenar conhecimento, mas para prover informação sobre os modelos. Este nível explica as estruturas, comportamentos e restrições.

E, finalmente, o meta-metamodelo corresponde ao uso de modelos para fornecer informações sobre os metamodelos. Esta informação é usada para converter, comparar e

integrar modelos, comportamentos e restrições, ou ainda validar a consistência de um conjunto de restrições.

Atualmente, já existe um consenso entre trabalhos e arquiteturas específicas que envolvem modelagem e metamodelagem, resultando numa arquitetura padrão que tem sido largamente adotada pela comunidade de desenvolvimento [UML, 2005] [OMG, 1996] [MOF, 2002]. Esta corresponde a uma arquitetura de quatro níveis, a saber: dados, modelo, metamodelo e meta-metamodelo, conforme ilustrada na figura 4.1.

- Meta-metamodelo: corresponde ao nível mais abstrato da arquitetura, representando a linguagem de definição do metamodelo, de forma a prover informações sobre este. Define os conceitos que formam a base da representação de todos os outros modelos, além dele próprio. Estas informações convertem, comparam e integram metamodelos, comportamentos e restrições, ou checam a consistência de um conjunto de restrições. Alguns exemplos destes conceitos são as MetaClasses, MetaAtributos e MetaOperações da UML; MetaEntidades com (meta)atributos e MetaRelacionamentos com (meta)atributos do CDIF; e conceito, relacionamento conceitual e grafo dos grafos conceituais;
- Metamodelo: é uma instância do meta-metamodelo, e define os formalismos ou linguagem de representação do modelo, representando estruturas, comportamentos e restrições. Alguns exemplos são as classes, atributos e operações da UML; entidades, atributos e relacionamentos do formalismo entidade-relacionamento; e tipos de conceito e tipos de relacionamento dos grafos conceituais;
- Modelo: é uma instância do metamodelo que define a própria representação da linguagem do domínio em questão. Corresponde ao uso de esquemas que armazenam e validam conhecimento, simulam ou aplicam comportamentos e armazenam restrições. Exemplos deste nível seriam empregado, organização e cargo;
- Dados: são instâncias do modelo, correspondendo aos objetos do mundo real que estão sendo descritos. Alguns exemplos seriam “José Silva”, “IME” e “Professor”.

Assim, um metamodelo descreve um modelo e ao mesmo tempo é descrito por um meta-metamodelo. Os conceitos definidos em um modelo são dependentes dos usados no

metamodelo. Um metamodelo pode definir conceitos como classe, objeto, herança, método, etc.

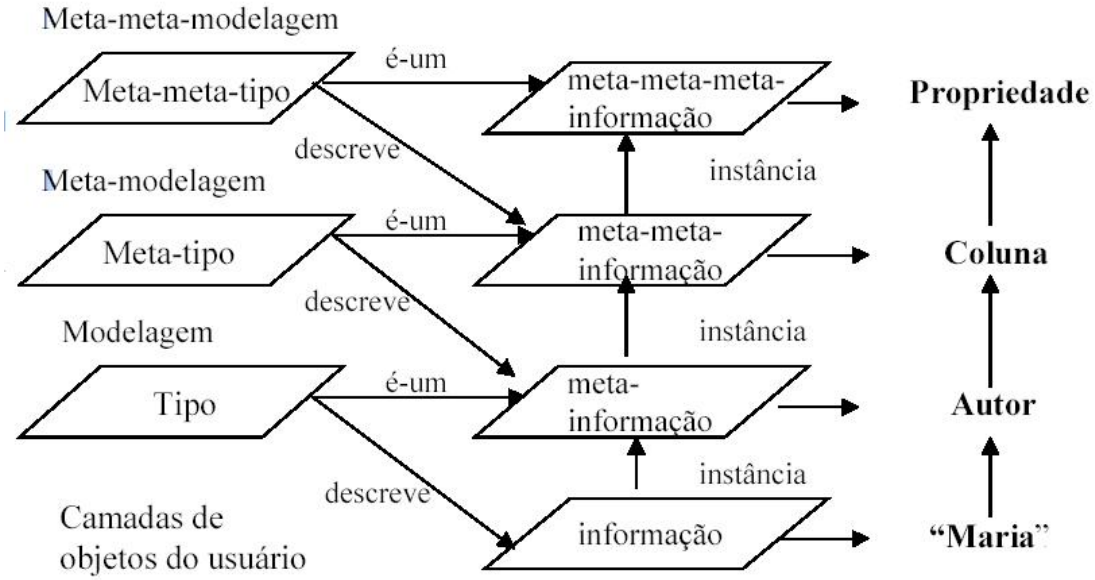

FIG. 4.1 Arquitetura de Modelagem Multi-Camadas.

A arquitetura MOF (Meta-Object Facility) [MOF, 2002], por exemplo, é baseada neste conceito de camadas. Nela, uma entidade em um modelo é definida por seu metamodelo correspondente. Um metamodelo restringe um modelo, já que este contém todos os conceitos que podem ser utilizados no modelo. Assim, cada nível sempre representa um esquema, enquanto o nível inferior a este representa suas instâncias.

A arquitetura ROSA$^+$ é baseada na idéia dessa arquitetura multicamadas. Entretanto, ao contrário da arquitetura MOF, onde cada uma das camadas representa o esquema que define e restringe a camada inferior, na arquitetura ROSA$^+$ não necessariamente a definição de um nível encontra-se em seu nível imediatamente superior. Podemos ver esta arquitetura na figura 4.2 a seguir.

A primeira camada é denominada mapa conceitual. Nela, estão os dados do domínio da aplicação ROSA: LOs e relacionamentos que constituem as bases de dados do sistema. (figura 4.2). Por exemplo, temos nessa camada os dados que constituem um mapa conceitual do ROSA, com LOs (*Sistemas e Computação*, *Banco de Dados*, *Redes*, etc.) e relacionamentos (*compreende*, *fundamenta*) que associam estes LOs.

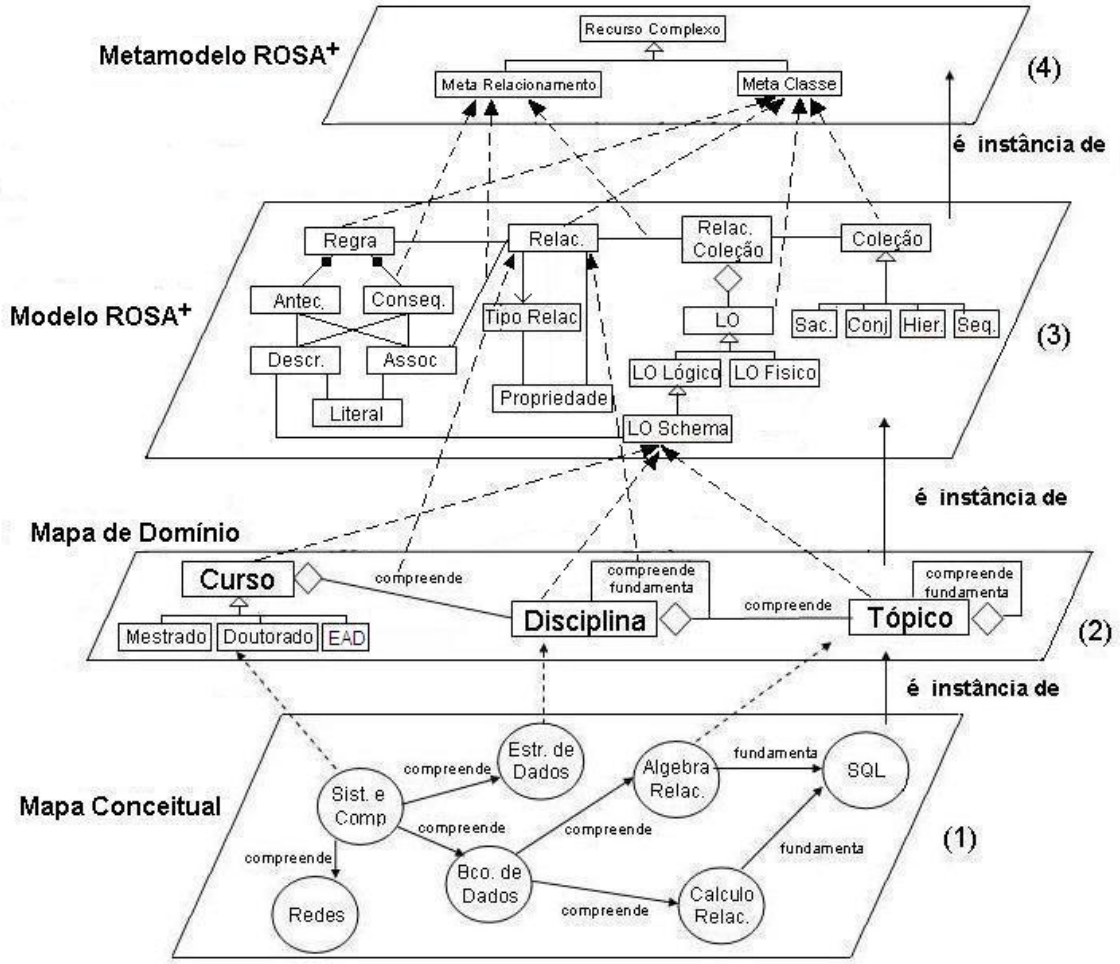

FIG. 4.2 Arquitetura de Modelagem ROSA^{+}.

Na segunda camada encontra-se o mapa do domínio, ou seja, o esquema do mapa conceitual representado na camada anterior. Este esquema é característico de cada domínio de EAD, e pode ser adaptado conforme as necessidades de cada instituição de ensino. Pode-se considerar que os elementos representados na camada inferior são instâncias desta. Fazendo uma analogia com um banco de dados relacional, seria como se cada LO do nível das instâncias pertencesse a uma tabela cujas colunas fossem uma classe do nível do modelo. Assim, o LO *Sistemas e Computação* da camada de dados é uma instância do LO *Curso* da camada de modelo. Pelo mesmo raciocínio, *Banco de Dados* é instância do LO *Disciplina* e *SQL*, de *Tópico*.

A terceira camada representa o modelo ROSA^{+}, que define os formalismos do sistema, como *LO*, *Relacionamento* e *Regra*. Estas classes compõem a base de dados do ROSA, representadas nas duas camadas inferiores. Cabe ressaltar aqui que, embora a

camada do modelo represente o esquema do nível de instâncias, os elementos de ambas as camadas são LOs, como será melhor detalhado na seção seguinte.

A quarta camada apresenta o metamodelo ROSA$^+$, que contém duas classes: *Meta LO* e *Meta Relacionamento*. As instâncias de *Meta LOs* são todas as classes da camada de metamodelo da arquitetura (LO, Relacionamento, Regra, etc.), enquanto as instâncias de *Meta Relacionamento* são os relacionamentos da UML que descrevem o modelo ROSA$^+$ descrito na terceira camada (associação, agregação, etc.).

Esta abordagem em camadas tem a vantagem de apresentar uma visão modularizada da arquitetura ROSA$^+$, onde cada nível de representação corresponde a uma camada abstrata dessa arquitetura. Assim, tem-se uma maior flexibilidade do modelo ROSA, permitindo maior expressividade e facilitando a manipulação das diversas entidades envolvidas.

4.2 MODELO DE DADOS ROSA$^+$

Em sua representação atual, o Metamodelo de Dados ROSA contempla a expressão de LOs e relacionamentos, bem como seus tipos. Entretanto, nele não é possível representar regras, o que aumentaria muito seu poder de expressividade. Para que o ROSA seja capaz de expressar regras, é necessária uma extensão do modelo, visando a representação de estruturas que sejam capazes de representar as novas funcionalidades no sistema.

Uma maneira de fazer com que o Modelo ROSA faça o uso de regras seria através da utilização de uma abordagem utilizando as linguagens de ontologias OWL (Ontology Web Language) e de regras SWRL (Semantic Web Rule Language), que são mais ricas e expressivas que a linguagem RDF. Através delas, é possível também representar propriedades como transitividade e inversão.

Assim sendo, no modelo de dados proposto, denominado ROSA$^+$, foram incorporadas novas entidades. Uma delas é a classe *LO Schema*, que representa o domínio ao qual os LOs pertencem. Em um dado mapa conceitual, por exemplo, o LO *Banco de Dados* seria um LO Lógico, e seria instância do LO Schema *Disciplina*. Cabe ressaltar que todo LO Schema é também um LO Lógico, apresentando as mesmas características deste.

O que os diferencia é apenas a função do LO Schema que define o esquema ao qual os demais LOs Lógicos pertencem. LOs Schema englobam apenas elementos da camada de instâncias, enquanto LOs Lógicos englobam tanto elementos da camada de instâncias quanto da de modelo.

Conforme visto no capítulo 3, cada regra possui um antecedente e um conseqüente, e estes, por sua vez, são definidos através de associações e descrições. Uma associação é descrita na forma P(x,y), onde x deve se relacionar a y através de uma propriedade P. Já uma descrição é da forma C(x), onde x deve ser uma instância da classe. Assim, por exemplo, uma regra:

`compreende(?a,?b) ∧ disciplina(?a) ∧ topico(?b) → possuiEmEmenta(?a,?b)`

possui uma associação entre dois literais através de um relacionamento (`compreende(?a,?b)`) e duas descrições de literais (`disciplina(?a)` e `topico(?b)`).

Assim, foram acrescentadas ao modelo as classes *Regra*, *Antecedente*, *Conseqüente*, *Associação*, *Descrição* e *Literal*. Estas novas classes podem ser vistas na FIG 4.3.

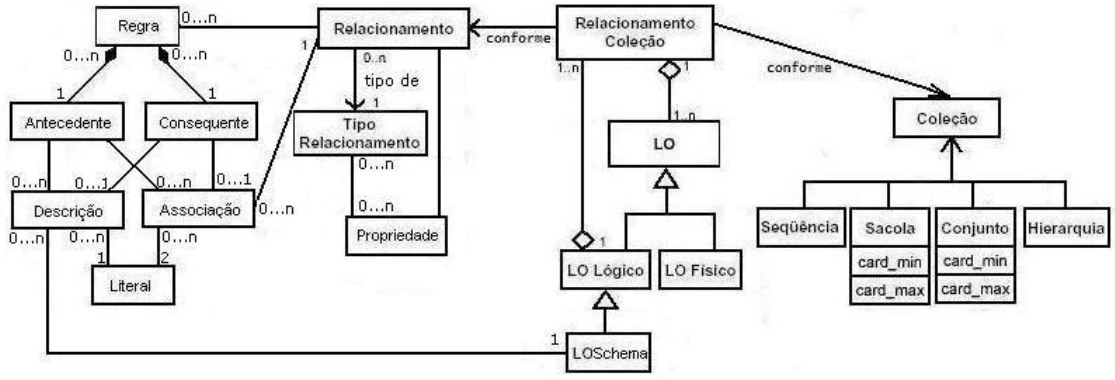

FIG. 4.3 Modelo de Dados ROSA^{+}.

Conforme pode ser visto na expressão que define o relacionamento de "ementa", uma regra é composta por um antecedente e um conseqüente. Um antecedente, por sua vez, é composto de descrições (`disciplina(?a),`) e associações (`compreende(?a,?b)`), enquanto um conseqüente é composto apenas de uma associação (`possuiEmEmenta(?a,?b), topico(?b)`) ou descrição. Uma descrição envolve uma instância de *LO Schema* e uma de *Literal*. Já para se compor uma associação, é preciso uma instância de *Relacionamento* (`compreende`) e duas de *Literal* (`?a` e `?b`).

Outra mudança que o modelo ROSA$^+$ apresenta em relação ao modelo ROSA é uma definição mais detalhada da classe *Tipo de Relacionamento* e a criação da classe *Propriedade*. O ROSA armazena uma grande quantidade de relacionamentos, muitos deles com o mesmo significado semântico. A classe *Tipo de Relacionamento* representa um tipo ao qual pertencem todos os relacionamentos de mesmo significado semântico Uma lista com todos os relacionamentos e tipos de relacionamentos foi elaborada e pode ser vista na Tabela 4.1. Já a classe *Propriedade* representa as propriedades que um tipo de relacionamento pode conter (transitividade, reflexão, simetria), que são implicitamente exploradas durante a execução de consultas.

Além disso, este modelo de dados suprime a classe *Recurso Complexo*, que se encontra agora em um nível superior, na camada do metamodelo ROSA$^+$, conforme visto na seção anterior (Figura 4.2).

4.3 ESPECIFICAÇÃO DA ARQUITETURA ROSA$^+$

Como visto na seção anterior, ROSA$^+$ é uma arquitetura definida em quatro camadas. Esta foi concebida com a finalidade de expressar melhor o conhecimento sobre diferentes níveis de abstração, com base na arquitetura utilizada no MOF.

Nesta seção, cada camada será especificada conforme apresentado na figura 4.2. O objetivo é mostrar como os objetos definidos em cada nível da arquitetura (representada na forma de triplas sujeito-relacionamento-predicado) são relacionados. Optou-se também por representar as instâncias desses objetos, sabendo-se que estas pertencem à camada inferior da arquitetura. Essa representação é feita na forma de representação binária, no formato de duplas classe-instância. Essa especificação visa a construção posterior da ontologia ROSA$^+$, que permitirá o acesso a cada um desses objetos segundo essas diversas perspectivas. Nesta seção será apresentada uma visão parcial dessa instanciação, cuja descrição completa encontra-se no Apêndice I.

TAB. 4.1 Relacionamentos e Tipos de Relacionamentos.

Tipo Relacionamento	agregação	ordenação	criação	equivalência	influência	permissão	necessidade	implicação	utilização
Relacsionamentos	compreende tem inclui enlaça abarca abrange possui envolve ehFormadoPor ehCompostoPor	fundamenta ehBasePara ehCondicaoPara	gera cria produz desenvolve	equivale ehMesmaQue seIgualaA seEquiparaA	influi inspira	permite admite licencia aceita deixa	necessita precisaDe carece demanda	implica aponta origina determina deriva resulta provê define especifica estabelece	utiliza usa usufrui aproveita
Relacionamentos inversos	ehParteDe ehCompreendidoPor ehAbrangidoPor ehPossuidoPor ehEnvolvidoPor constitui compõe ehComponenteDe forma ehMembroDe ehTidoPor ehEnlacadoPor ehAbracadoPor	ehFundamentadoPor ehBaseDe	ehGeradoPor ehCriadoPor ehProduzuidoPor ehDesenvolvidoPor	seDistingueDe seDifereDe seDiferenciaDe ehDistintoDe	ehInfluenciadoPor ehInspiradoPor	ehPermitidoPor ehAdmitidoPor ehLicenciadoPor ehAceitoPor	ehNecessarioPara ehPrecisoPara	ehImplicadoPor ehOriginadoPor ehDeterminadoPor ehDerivadoPor ehResultadoDe ehDefinidoPor ehEspecificadoPor ehEstabelecidoPor	ehUtilizadoPor ehUsufruidoPor ehAproveitadoPor

4.3.2 CAMADA 1 – MAPA CONCEITUAL

Nesta camada estão representadas, através de relacionamentos binários, as instâncias que compõem o mapa conceitual. Estas são de fato instâncias das classes do esquema ROSA que serão definidas na camada de domínio.

Compreende(Sistemas e Computação,Estrutura de Dados)
Compreende(Sistemas e Computação,Redes)
Compreende(Sistemas e Computação,Banco de Dados)
Compreende(Banco de Dados,Álgebra Relacional)
Compreende(Banco de Dados,Cálculo Relacional)
Fundamenta(Álgebra Relacional,SQL)
Fundamenta(Cálculo Relacional,SQL)

Alguns relacionamentos binários não são especificados, mas são inferidos através de regras, a exemplo de:

EhPreRequisitoDe(Estrutura de Dados,Banco de Dados)
PossuiEmPrograma(Sistemas e Computação, Estrutura de Dados)
PossuiEmEmenta(Estrutura de Dados,Algoritmos)

Estes relacionamentos não são declarados na especificação, mas obtidos através de inferência sobre as regras. Dessa forma tem-se que:

- pré requisito: determina que uma disciplina será pré requisito de outra sempre que ela contiver um tópico que fundamente outro pertencente à outra disciplina. Essa regra é expressa através de:
  ehPreRequisitoDe(a,b) ← disciplina(a), disciplina(b), tópico(c), tópico(d), compreende(a,c), compreende(b,d), fundamenta(c,d)

- ementa: quando uma disciplina compreender determinado tópico, infere-se que ela o tem em sua ementa. Essa regra é expressa através de:
  possuiEmEmenta(a,b) ← disciplina(a), tópico(c), compreende(a,c)

- programa: quando um curso compreender determinada disciplina, infere-se que ele o tem em seu programa. Essa regra é expressa através de:
  possuiEmPrograma(e,a) ← curso(e),disciplina(a), compreende(e,a)

4.3.2 CAMADA 2 – MAPA DE DOMÍNIO

Nesta camada estão representados os relacionamentos entre as classes do esquema, que definem as associações possíveis de serem representadas na camada de mapa conceitual.

Compreende(Curso,Disciplina)
DependeDe(Disciplina,Disciplina)
DependeDe(Curso,Curso)
Compreende(Tópico,Tópico)
Fundamenta(Tópico,Tópico)
IsA(Doutorado,Curso)
IsA(Mestrado,Curso)
IsA(EAD,Curso)

Na versão anterior do ROSA, o esquema do mapa conceitual estava representado na forma de atributos das instâncias. Assim, expressava-se que um LO qualquer, por exemplo, *Banco de Dados*, era do tipo *Disciplina* através de um atributo definido em RDF. Os relacionamentos entre *Disciplina* e *Tópico* não eram representados até então.

A arquitetura ROSA$^+$ expressa estes tipos de LOs através de mais um nível de abstração, aqui denominada camada de mapa de domínio. As classes desta segunda camada estão também representadas como LOs (através da especialização LO Schema, como será visto na próxima subseção). Os objetos da primeira camada (Sistemas e Computação, Estrutura de Dados, etc.) são instâncias desta segunda, onde estão os LOs que representam o domínio de ensino a distância.

Curso(Sistemas e Computação)
Disciplina(Estrutura de Dados)
Disciplina(Redes)
Disciplina(Banco de Dados)
Tópico(SQL)
Tópico(Álgebra Relacional)
Tópico(Cálculo Relacional)

4.3.3 CAMADA 3 – MODELO DE DADOS ROSA$^+$

Nesta camada está representada o modelo ROSA$^+$ visto na seção 4.2. A camada anterior da arquitetura pode ser visto como instância desta.

Nela, temos a definição de LO Schema, que é uma especialização de um LO Lógico que visa instanciar o esquema do domínio. As instâncias do ROSA$^+$ (os LOs que representam os cursos, disciplinas e tópicos) são da classe *LO Lógico*. Não trataremos aqui os LOs Físicos, que representam os arquivos associados aos LOs Lógicos (apostilas, planilhas, apresentações, etc.). Essa hierarquia é representada por:

IsA(LOSchema,LOLogico)
IsA(LOLogico,LO)
IsA(LOFisico,LO)

Assim, as classes definidas na camada anterior passam a ser de fato, instâncias de LOSschema, enquanto os LOs definidos na camada de mapa conceitual são instâncias de LOLogico, conforme mostrado a seguir:

LOSchema(Curso)
LOSchema (Disciplina)
LOSchema (Tópico)

LOLogico (Sistemas e Computação)
LOLogico (Estrutura de Dados)
LOLogico (Redes)
LOLogico (Banco de Dados)
LOLogico (SQL)
LOLogico (Álgebra Relacional)
LOLogico (Cálculo Relacional)

As classes nesta camada relacionam-se entre si através dos relacionamentos definidos pela UML: associação, agregação, herança, conformidade e tipo.

Agrega(LOLogico, relacionamentoColeção)
Agrega(relacionamentoColeção,LO)

TipoDe(Relacionamento,TipoRelacionamento)

IsA(Sacola,Coleçao)
IsA(Hierarquia,Coleçao)
IsA(Seqüência,Coleçao)
IsA(Conjunto,Coleçao)

Tem(Propriedade,TipoRelacionamento)
Tem(Regra,RelacionamentoColeçao)

Conforme(RelacionamentoColeçao,Relacionamento)
Conforme(RelacionamentoColeçao,Coleção)

Também é nesta camada da arquitetura onde são definidas as associações entre os relacionamentos (ou predicados, através dos quais os LOs se associam), e seus tipos, que representam agrupamentos de relacionamentos com o mesmo significado semântico. Assim, o relacionamento definido anteriormente expresso por: TipoDe(Relacionamento,TipoRelacionamento), classifica predicados ou relacionamentos segundo uma dada categoria. A título de exemplo, instanciamos apenas alguns predicados de cada categoria de associação (agregação, equivalência, ordenação, etc), conforme apresentados na tabela 4.1.

TipoDe(compreende, agregação)
TipoDe(fundamenta, fundamentação)
TipoDe(gera, criação)
TipoDe(equivale, equivalência)
TipoDe(influi, influência)
TipoDe (permite,permissão)
TipoDe (precisaDe,necessidade)
TipoDe (determina, implicação)
TipoDe (utiliza, utilização)

O modelo também define as instanciações dos tipos de relacionamentos do domínio. A classificação completa dos relacionamentos e seus tipos pode ser vista também na Tabela 4.1.

TipoRelacionamento(agregação)
TipoRelacionamento(fundamentação)
TipoRelacionamento(criação)
TipoRelacionamento(equivalência)
TipoRelacionamento(influência)
TipoRelacionamento(permissão)
TipoRelacionamento(necessidade)
TipoRelacionamento(implicação)
TipoRelacionamento(utilização)

A seguir são apresentadas algumas instâncias da classe Relacionamento, que aparecerão nas camadas anteriores já descritas.

Relacionamento(compreende)
Relacionamento(tem)
Relacionamento(inclui)
...

Relacionamento(fundamenta)

Relacionamento(é base para)
...

Relacionamento(gera)
Relacionamento(cria)
...

Relacionamento(equivale)
Relacionamento(se assemelha a)
...

Relacionamento(influi)
Relacionamento(inspira)
...

Relacionamento(necessita)
Relacionamento(precisa de)
...

Relacionamento(implica)
Relacionamento(determina)
...

Relacionamento(é compreendido por)
Relacionamento(é abrangido por)
...

Relacionamento(é fundamentado por)
Relacionamento(tem como pré requisito)
...

Relacionamento(é gerado por)
Relacionamento(é criado por)
...

Relacionamento(se distingue de)
Relacionamento(se difere de)
...

Relacionamento(é influenciado por)
Relacionamento(é inspirado por)

Relacionamento(é permitido por)
Relacionamento(é admitido por)
...

Relacionamento(é necessário para)
Relacionamento(é preciso para)

Relacionamento(é implicado por)

Relacionamento(é originado por)
...

Os relacionamentos e os tipos possuem propriedades, tais como transitividade, simetria, inversão e equivalência (same as), que lhes conferem maior expressividade semântica.

propriedade(transitividade)
propriedade(simetria)
propriedade(inversão)
propriedade(equivalência)

A propriedade de equivalência, por exemplo, determina que dois relacionamentos são considerados sinônimos, a exemplo de:

same as(compreende,abrange)
same as(compreende,é formado por)
same as(compreende,tem)
same as(fundamenta,é base para)
same as(fundamenta,é condição para)
same as(gera,cria)
same as(gera,produz)
same as(equivale,se assemelha a)
same as(equivale,se compara a)

A propriedade inversa determina o relacionamento inverso de cada predicado, enquanto as propriedades transitiva e simétrica têm como instâncias os predicados que estão de acordo com essas propriedades. A seguir listamos alguns exemplos:

inversa(compreende, é compreendido por)
inversa(fundamenta, é fundamentado por)
inversa(gera,é gerado por)
inversa(equivale,se distingue de)
inversa(influi,é influenciado por)
inversa(permite,é permitido por)
inversa(necessita,é necessário para)
....
transitiva(agregação)
transitiva(fundamentação)

transitiva(influência)
transitiva(necessidade)
...
simetrica(equivalência)

No modelo ROSA$^+$ também estão expressas as classes que constituem as regras, através das quais é possível inferir conhecimento a partir de implicações que determinam outras. Elas são compostas por antecedente e conseqüente, que por sua vez são compostos por associações e descrições.

Tem(Regra,Antecedente)
Tem(Regra,Conseqüente)
Tem(Antecedente,Associação)
Tem(Antecedente,Descrição)
Tem(Conseqüente,Associação)
Tem(Conseqüente,Descrição)
Tem(Associação,Literal)
Tem(Associação,Relacionamento)
Tem(Descrição,Literal)
Tem(Descrição,LOSchema)

Para exemplificar, instanciaremos a seguir os componentes da regra "ehPrerequisitoDe", considerando os literais, associações e descrições a seguir:

Literal(a)
Literal(b)
Literal(c)
Literal(d)

Associação(compreende(a,c))
Associação(compreende(b,d))
Associação(fundamenta(c,d))
Associação(ehPreRequisitoDe(a,b))

Descrição(disciplina(a))
Descrição(disciplina(b))
Descrição(tópico(c))
Descrição(tópico(d))

Antecedente(disciplina(a) ^ disciplina(b) ^ tópico(c) ^ tópico(d) ^ compreende(a,c) ^ compreende(b,d) ^ fundamenta(c,d))

Conseqüente(ehPreRequisitoDe(a,b))

Também nesta camada é mostrado como as coleções de LOs (LO_1,LO_2 ... LO_n) podem ser compostas.

Hierarquia (LO_1,LO_2) = $LO_2 \subseteq LO_1$
Sacola (LO_1,LO_2 ... LO_n) = { LO_i … LO_j }
Sequência (LO_1,LO_2 ... LO_n) = { LO_i … LO_j } Π (i < j)

Conjunto (LO_1,LO_2 ... LO_n) = { LO_i … LO_j } Π (i ≠ j)

4.3.4 CAMADA 4 – METAMODELO ROSA$^+$

Esta camada corresponde ao nível de metadados, onde está inserida a classe *RecursoComplexo*, definida como uma abstração para generalizar MetaClasses e MetaRelacionamentos, representando desta maneira os elementos utilizados na arquitetura.

Isa(MetaClasse, RecursoComplexo)
Isa(MetaRelacionamento, RecursoComplexo)

A classe *MetaClasse* define as classes do nível do modelo ROSA$^+$, a exemplo de LOs, Regras, Relacionamentos, etc.; e a classe *MetaRelacionamento*, tem como instâncias os relacionamentos da camada anterior, conforme mostrado a seguir.

MetaClasse(Relacionamento)
MetaClasse(TipoRelacionamento)
MetaClasse(LO)
MetaClasse(RelacionamentoColeção)
MetaClasse(Propriedade)
MetaClasse(Coleção)
MetaClasse(Regra)
MetaClasse(Antecedente)
MetaClasse(Conseqüente)
MetaClasse(Associação)
MetaClasse(Descrição)
MetaClasse(Literal)

MetaRelacionamento(IsA)
MetaRelacionamento(Agrega)
MetaRelacionamento(TipoDe)
MetaRelacionamento(Tem)
MetaRelacionamento(Conforme)

4.4 CONSIDERAÇÕES FINAIS

O ROSA$^+$ é um modelo estendido a partir do modelo ROSA que visa incorporar a representação de regras. Foi construído num contexto de arquitetura em quatro camadas, sobre o qual é possível trabalhar em variados níveis de abstração, a saber: instância, esquema e modelo, permitindo uma melhor compreensão de cada aspecto conceitual nos diferentes tipos de abstração. O ROSA$^+$ contempla a representação de regras, permitindo também identificar relacionamentos implícitos através do processamento de propriedades aplicadas diretamente aos LOs durante a execução de consultas.

Esta representação tem como objetivo modelar o sistema ROSA$^+$ de maneira consistente, visando a sua implementação, que será vista no capítulo seguinte.

5 ESPECIFICAÇÃO DO SISTEMA ROSA+

Este capítulo visa apresentar a etapa de especificação do ROSA+, sistema de consulta a LOs baseado na arquitetura descrita no capítulo anterior, cujo objetivo é inferir conhecimento a partir de propriedades de relacionamentos e regras. Estas etapas compreendem as fases de representação e especificação das ontologias definidas a partir da arquitetura ROSA+ em uma linguagem de ontologia. Para isso, foi necessário fazer um levantamento das tecnologias disponíveis. Assim, na seção 5.1 será mostrado um estudo sobre as ferramentas para edição de ontologias, com destaque para o Protégé, ferramenta adotada neste trabalho. A seção 5.2 trata das adaptações realizadas para que a arquitetura possa ser representada na linguagem OWL-DL. Na seção 5.3 serão mostrados detalhes de construção da ontologia a partir de trechos da ontologia em OWL-DL e SWRL. Alguns dos raciocinadores de lógica descritiva utilizados para inferir conhecimento de ontologias serão abordados na seção 5.4. E, por fim, a seção 5.5 descreve os demais módulos do sistema ROSA+.

5.1 FERRAMENTAS PARA MANIPULAÇÃO DE ONTOLOGIAS

O primeiro passo no desenvolvimento do ROSA+ foi a construção da ontologia para representar sua arquitetura em camadas. Para auxiliar este desenvolvimento, foi realizado um estudo sobre as ferramentas manipuladoras de ontologias disponíveis, com prioridade para aquelas disponíveis em software livre.

Um grande número de ferramentas foi desenvolvido [CORCHO et al, 2003] nos últimos anos, dentre as quais algumas merecem destaque: KAON [MÄEDCHE et al, 2000], WebODE [CORCHO et al, 2002] e Protégé [NOY et al, 2000]. Além destas, também é possível citar outros editores, tais como o OilEd[13], WebOnto[14], Ontolingua[15] e OntoSaurus[16], utilizados para edição de ontologias em linguagens específicas, tais

[13] http://img.cs.man.ac.uk/oil/
[14] http://kmi.open.ac.uk/projects/webonto/
[15] http://www.ksl.stanford.edu/software/ontolingua/
[16] http://www.isi.edu/isd/ontosaurus.html

como OIL, OCML, Ontolingua e LOOM, respectivamente. Estes não são editores completos de ontologias e tampouco usam bancos de dados para armazenar as ontologias.

Através destas ferramentas editoras, é possível criar e manipular ontologias, verificar inconsistências destas, compartilhar e reutilizar dados que estejam em diferentes ontologias, podendo-se representá-las através de várias linguagens. Assim, é possível que o usuário se concentre no desenvolvimento da ontologia sem se preocupar com detalhes específicos da linguagem de ontologia sobre a qual está trabalhando.

Em Mattos [MATTOS et al., 2005], foi realizado um estudo de três das principais ferramentas de manipulação de ontologias: Protégé, KAON e WebODE, onde se concluiu que o Protégé seria a ferramenta mais adequada para o desenvolvimento do trabalho por apresentar as seguintes características:

- arquitetura de *plug-ins*, onde novos serviços podem ser facilmente incorporados ao sistema, justificando o fato do sistema possuir uma base de usuários maior que a de seus concorrentes, sendo atualmente o editor de ontologias disponível mais difundido;
- interface gráfica estável e de fácil navegação, oferecendo uma excelente configurabilidade através de seus formulários, tornando assim fácil a manipulação de ontologias de forma visual;
- funcionalidades de edição, documentação, importação e exportação de ontologias em XML, OWL, SWRL e outras linguagens;
- oferece uma infra-estrutura totalmente em código aberto, podendo suas APIs[17] serem reutilizadas para o desenvolvimento de outras aplicações de edição de ontologias personalizadas.

Devido a essas características, e principalmente por fornecer *plug-ins* para as linguagens adotadas no trabalho (OWL e SWRL), a ferramenta Protégé foi adotada no desenvolvimento do ROSA^{+}. O *plug-in* OWL estende a plataforma Protégé para implementação em OWL DL. Assim, o usuário pode facilmente navegar pelas hierarquias de classes, visualizá-las e criar novas classes e propriedades, podendo relacionar propriedades entre estas classes. Já o *plug-in* SWRL permite a criação de

[17] *Application Program Interface*

regras nesta linguagem de maneira simples, e, além disso, por ser uma ferramenta de código aberto, suas APIs serão úteis no desenvolvimento do sistema ROSA$^+$, como será visto no próximo capítulo.

5.2 ADAPTAÇÃO DA ARQUITETURA ROSA$^+$ PARA A LINGUAGEM OWL

Definida a arquitetura ROSA$^+$ (seção 4.1), torna-se necessário representá-la em uma linguagem de ontologia, para que assim ela possa servir como base de dados para o sistema ROSA$^+$.

Como visto no capítulo 3, a OWL é uma linguagem capaz de representar explicitamente ontologias, ou seja, o significado de um vocabulário de termos e os relacionamentos entre estes termos. Por ser uma evolução das linguagens RDF e DAML-OIL [CONNOLLY et al., 2001], ela é capaz de processar o conteúdo de informações, ao invés de apenas apresentar a informação para humanos. Por vantagens como essa, foi adotada pelo W3C como a linguagem recomendada para representar a camada de ontologias na arquitetura da Web Semântica proposta por Berners-Lee (figura 3.1).

Assim, parece natural a escolha da linguagem OWL para representar a arquitetura definida neste trabalho, merecendo destaque especial no título desta proposta: ROSA$^+$.

Dentre as três sublinguagens da OWL, a adotada foi a OWL-DL, por apresentar maior capacidade de representação que a OWL-Lite, e, ao contrário da OWL-Full, é uma linguagem processável por raciocinadores de lógica descritiva.

Dentre as quatro camadas da arquitetura ROSA$^+$, optou-se por não explicitar na ontologia o modelo ROSA$^+$, já que este apresenta apenas elementos abstratos que permitem a definição do modelo ROSA$^+$, não sendo relevante para o sistema inferir algum conhecimento através de consulta sobre este nível.

Assim, a representação da arquitetura ROSA$^+$ em uma ontologia significa representar seus três primeiros níveis (mapa conceitual, mapa do domínio e modelo ROSA$^+$) em um único documento. Entretanto, uma limitação da linguagem OWL-DL impede esta representação: nela, não é possível expressar um elemento como classe e

instância ao mesmo tempo. Por exemplo, o elemento *Disciplina* é uma classe no mapa do domínio, e ao mesmo tempo, é uma instância no modelo ROSA^{+}. Assim, a representação desta característica não é possível na linguagem OWL-DL. Por outro lado, apesar dessa característica ser possível na OWL-Full, sua adoção seria inviável, visto que os raciocinadores disponíveis atualmente não são capazes de inferir conhecimento sobre suas funcionalidades.

Para suprir esta deficiência da linguagem OWL-DL, a solução proposta foi a criação de duas ontologias: a primeira delas abordando apenas o mapa conceitual e o mapa do domínio, onde o primeiro seria instância do segundo (figura 5.1); e a segunda ontologia envolvendo os elementos do mapa do domínio e do modelo ROSA^{+}, onde também o primeiro é representado como instância do segundo (figura 5.2).

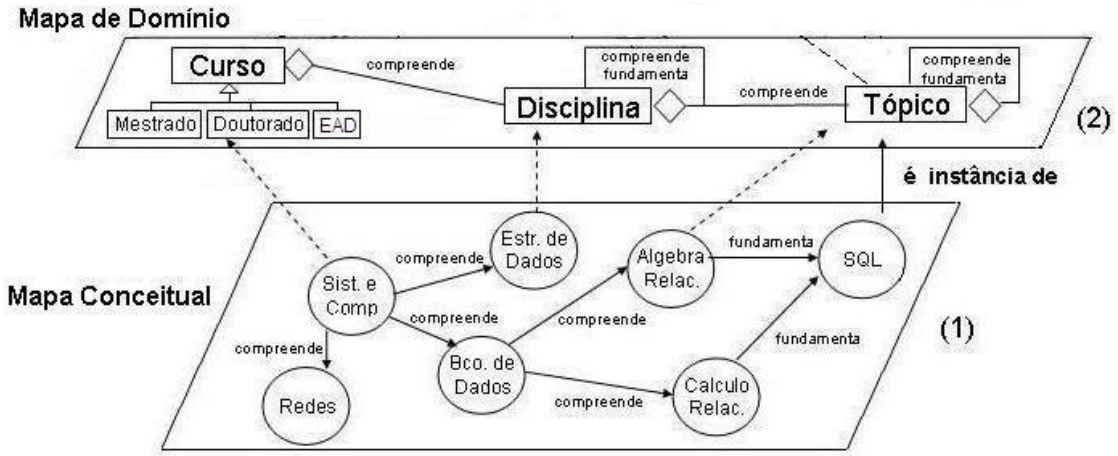

FIG. 5.1 Ontologia Mapa Conceitual/Mapa Domínio da arquitetura ROSA^{+}.

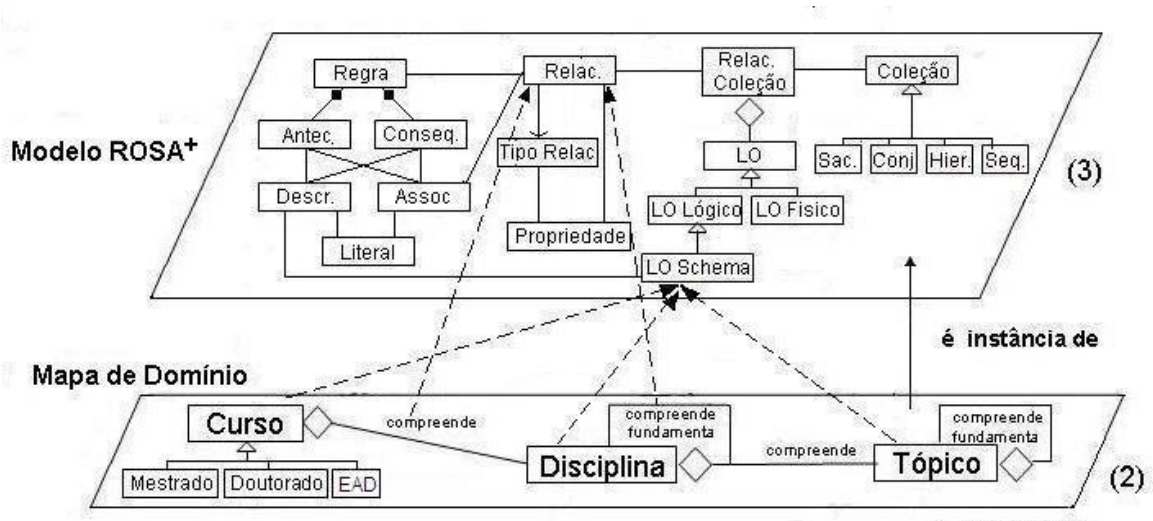

FIG. 5.2 Ontologia Mapa Domínio/Modelo ROSA$^+$ da arquitetura ROSA$^+$.

Denominaremos neste trabalho a primeira ontologia como *Ontologia Mapa Conceitual/Mapa Domínio* (figura 5.1), enquanto a segunda será chamada de *Ontologia Mapa Domínio/Modelo ROSA*$^+$ (figura 5.2). A partir destas duas ontologias, torna-se possível representar *Disciplina*, por exemplo, tanto como instância de *LOSchema* presente no modelo, quanto como a própria classe *Disciplina* do nível de mapa de domínio, que tem como instâncias os LOs *Banco de Dados*, *Estrutura de Dados* e *Redes*. A representação de cada uma destas ontologias será abordada com detalhes na próxima seção.

5.3 ONTOLOGIAS ROSA$^+$

Esta seção tem por objetivo mostrar detalhes da representação da arquitetura ROSA$^+$ em suas duas ontologias, utilizando para isso as OWL-DL e SWRL.

Para essa representação, foi necessário utilizar classes abstratas para tornar possível a representação de seus conceitos na linguagem OWL-DL.

A subseção 5.3.1 abordará detalhes da Ontologia Mapa Conceitual/Mapa Domínio, enquanto a subseção 5.3.2 mostrará a Ontologia Mapa Domínio/Modelo ROSA$^+$. A representação integral das duas ontologias encontra-se nos Apêndices 9.2 e 9.3.

5.3.1 ONTOLOGIA MAPA CONCEITUAL/MAPA DOMÍNIO

A primeira definição abordada é a de classe, ou seja, os conceitos que constituem a ontologia. Na Ontologia Mapa Conceitual/Mapa Domínio, *Curso*, *Disciplina* e *Tópico* estão representadas como subclasses de uma classe abstrata *LO*, que não possui instâncias, sendo apenas utilizada para definir os domínios dos relacionamentos. A classe *Disciplina*, por exemplo, representa uma classe do nível de modelo, e é representada em OWL da seguinte forma:

```
<owl:Class rdf:ID="Disciplina">
  <rdfs:subClassOf>
```

```
      <owl:Class rdf:ID="LO"/>
    </rdfs:subClassOf>
  </owl:Class>
```

Como definido na arquitetura, na camada de mapa conceitual encontram-se as instâncias das categorias representadas no modelo. Nesta ontologia, a classe *Disciplina*, por exemplo, possui *Banco de Dados* como instância, o que é representado na ontologia conforme visto a seguir:

```
<Disciplina rdf:ID="Banco_de_Dados"/>
```

É possível representar hierarquias de classes. A classe *Curso*, por exemplo, possui três subclasses (*EAD, Mestrado* e *Doutorado*):

```
<owl:Class rdf:ID="Mestrado">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Curso"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Doutorado">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Curso"/>
  </rdfs:subClassOf>
</owl:Class>
```

Os relacionamentos são representados no nível de modelo. As restrições de *domain* e *range* presentes na linguagem OWL definem, respectivamente, a classe (LO) e a quais classes ela podem ser associadas através do relacionamento (LO). O exemplo a seguir mostra que qualquer LO pode se relacionar através do relacionamento *fundamenta* a qualquer outro LO, independentemente da subclasse a qual pertença.

```
<owl:ObjectProperty rdf:ID="fundamenta">
  <rdfs:range rdf:resource="#LO"/>
  <rdfs:domain rdf:resource="#LO"/>
</owl:ObjectProperty>
```

Uma vez definidos os relacionamentos expressos no nível de modelo, estes podem ser aplicados às instâncias, relacionando LOs entre si, como no exemplo a seguir, onde a disciplina *Banco de Dados* compreende o tópico *Álgebra Relacional*.

```
<Disciplina rdf:ID="Algoritmo">
  <fundamenta>
    <Topico rdf:ID="Linguagem_Consulta"/>
  </fundamenta>
</Disciplina>
```

Conforme visto no capítulo anterior (tabela 4.1), os relacionamentos de mesmo significado semântico estão agrupados em tipos, possuindo as mesmas propriedades, que podem ser: equivalência, transitividade, simetria e inversão. No exemplo a seguir, o relacionamento *compreende* é definido como transitivo e inverso ao relacionamento *ehCompreendidoPor*.

```
     <owl:ObjectProperty rdf:about="#compreende">
       <rdfs:range rdf:resource="#LO"/>
       <owl:inverseOf>
         <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
       </owl:inverseOf>
       <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
       <rdfs:domain rdf:resource="#LO"/>
     </owl:ObjectProperty>
```

A representação de tipo de relacionamento só ocorre na camada de modelo ROSA$^+$, ou seja, só será declarada na Ontologia Mapa Domínio/Modelo ROSA$^+$. Entretanto, os relacionamentos estão presentes na ontologia atual. Para agrupá-los em seus tipos, fazendo com que compartilhem as mesmas propriedades, todos os relacionamentos de um tipo são declarados como equivalentes.

```
<owl:ObjectProperty rdf:ID="possui">
  </owl:equivalentProperty>
    <owl:ObjectProperty rdf:ID="tipoAgregacao"/>
  </owl:equivalentProperty>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="abrange"/>
  <owl:equivalentProperty>
    <owl:ObjectProperty rdf:ID="tipoAgregacao"/>
  </owl:equivalentProperty>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="compreende"/>
  <owl:equivalentProperty>
    <owl:ObjectProperty rdf:ID="tipoAgregacao"/>
  </owl:equivalentProperty>
</owl:ObjectProperty>
```

Na Ontologia Mapa Conceitual/Mapa Domínio é onde são representadas as regras, já que as mesmas serão aplicadas para inferir conhecimento sobre a base de instâncias do ROSA$^+$. Será utilizada como exemplo desta representação a regra *Pré-Requisito*, descrita no capítulo anterior:

```
ehPreRequisito(?a,?b) ← Disciplina(?a) ∧ Disciplina(?b) ∧ Topico(?x)
      ∧ Topico(?y) ∧ compreende(?a,?x) ∧ compreende(?b,?y) ∧
                        fundamenta(?x,?y)
```

Na representação desta regra na linguagem SWRL, uma variável é representada da seguinte maneira:

```
<swrl:Variable rdf:ID="b"/>
```

A cabeça da regra pré-requisito, com o predicado binário e as variáveis que a compõem, são representados da seguinte forma:

```
<swrl:head>
  <swrl:AtomList>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first>
      <swrl:IndividualPropertyAtom>
        <swrl:propertyPredicate rdf:resource="#ehPreRequisito"/>
        <swrl:argument2>
          <swrl:Variable rdf:ID="y"/>
        </swrl:argument2>
        <swrl:argument1>
          <swrl:Variable rdf:ID="x"/>
        </swrl:argument1>
      </swrl:IndividualPropertyAtom>
    </rdf:first>
  </swrl:AtomList>
</swrl:head>
```

O corpo tem representação semelhante, mas com a marcação *body* ao invés da *head* e com diversas associações de descrições:

```
<swrl:body>
  <swrl:AtomList>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate rdf:resource="#Topico"/>
                <swrl:argument1 rdf:resource="#x"/>
              </swrl:ClassAtom>
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:first>
                      <swrl:IndividualPropertyAtom>
                        <swrl:argument1>
                          <swrl:Variable rdf:ID="a"/>
                        </swrl:argument1>
                        <swrl:propertyPredicate rdf:resource="#compreende"/>
                        <swrl:argument2 rdf:resource="#x"/>
                      </swrl:IndividualPropertyAtom>
                    </rdf:first>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:rest>
                          <swrl:AtomList>
                            <rdf:first>
                              <swrl:IndividualPropertyAtom>
                                <swrl:argument1 rdf:resource="#x"/>
                                <swrl:argument2 rdf:resource="#y"/>
                                <swrl:propertyPredicate rdf:resource="#fundamenta"/>
                              </swrl:IndividualPropertyAtom>
                            </rdf:first>
```

```
                              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
                            </swrl:AtomList>
                          </rdf:rest>
                          <rdf:first>
                            <swrl:IndividualPropertyAtom>
                              <swrl:argument2 rdf:resource="#y"/>
                              <swrl:propertyPredicate rdf:resource="#compreende"/>
                              <swrl:argument1 rdf:resource="#b"/>
                            </swrl:IndividualPropertyAtom>
                          </rdf:first>
                        </swrl:AtomList>
                      </rdf:rest>
                    </swrl:AtomList>
                  </rdf:rest>
                  <rdf:first>
                    <swrl:ClassAtom>
                      <swrl:classPredicate rdf:resource="#Topico"/>
                      <swrl:argument1 rdf:resource="#y"/>
                    </swrl:ClassAtom>
                  </rdf:first>
                </swrl:AtomList>
              </rdf:rest>
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:ClassAtom>
              <swrl:classPredicate rdf:resource="#Disciplina"/>
              <swrl:argument1 rdf:resource="#b"/>
            </swrl:ClassAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
      <rdf:first>
        <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#Disciplina"/>
          <swrl:argument1 rdf:resource="#a"/>
        </swrl:ClassAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
```

5.3.2 ONTOLOGIA MAPA DOMÍNIO/MODELO ROSA$^+$

A Ontologia Mapa Domínio/Modelo ROSA$^+$ tem como instâncias os elementos do modelo da arquitetura ROSA$^+$, ou seja, os mesmos que, na Ontologia Mapa Conceitual/Mapa Domínio, estão representados como classes. Nesta ontologia, as classes correspondem aos elementos do metamodelo de dados ROSA$^+$.

Assim, *LOSchema* é uma subclasse de *LO Lógico*, que, por sua vez, é uma subclasse de *LO*. *LO Físico* é representado de maneira análoga a *LO Lógico*, conforme pode ser visto a seguir.

```
<owl:Class rdf:ID="LOSchema">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="LOLogico"/>
  </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="LOLogico">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="LO"/>
  </rdfs:subClassOf>
</owl:Class>
```

Os LOs pertencentes à camada de mapa do domínio são instâncias de LOs Schema, uma especialização da classe LO Lógico, onde estão representados os LOs da camada de mapa conceitual. Os exemplos a seguir ilustram esta instanciação.

```
<LOSchema rdf:ID="curso"/>
<LOSchema rdf:ID="disciplina"/>
<LOSchema rdf:ID="topico"/>
<LOLogico rdf:ID="Sistemas_e_Computacao"/>
<LOLogico rdf:ID="Banco_de_Dados"/>
<LOLogico rdf:ID="Álgebra_Relacional"/>
```

Na camada de modelo, relacionamentos não são predicados de associação entre classes. Como visto na seção 4.2, existe uma classe *Relacionamento*. LOs e relacionamentos se associam através dos meta-relacionamentos *agrega* e *é agregado por*, inversos um do outro, o que é representado da seguinte maneira.

```
<owl:Class rdf:ID="Relacionamento"/>

<owl:ObjectProperty rdf:ID="ehAgregadoPor">
  <rdfs:domain rdf:resource="#Relacionamento"/>
  <rdfs:range rdf:resource="#LO"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="agrega"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Outra classe representada nesta ontologia é *Tipo Relacionamento*. Cada relacionamento está associado a um tipo através do meta-relacionamento *tipo de*, que é inverso ao meta-relacionamento *é tipo para*, bem como suas instâncias (as propriedades *simetria* e *transitividade*), conforme pode ser visto no seguinte trecho da ontologia.

```
<owl:Class rdf:ID="TipoRelacionamento"/>
<owl:ObjectProperty rdf:ID="tipoDe">
  <rdfs:range rdf:resource="#TipoRelacionamento"/>
  <rdfs:domain rdf:resource="#Relacionamento"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="ehTipoPara"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Cabe aqui ressaltar que nem todo relacionamento pertence a um tipo. Alguns são criados para constituir a cabeça de uma regra, tendo seu significado semântico expresso pela mesma.

Propriedades podem ser atribuídas aos tipos de relacionamento. A seguir, é apresentada a definição da classe *Propriedade*, bem como sua instanciação e a representação do metarelacionamento *tem propriedade*, que indica quais propriedades pertencem a determinado tipo de relacionamento.

```
<owl:Class rdf:ID="Propriedade"/>

<owl:ObjectProperty rdf:ID="temPropriedade">
  <rdfs:domain rdf:resource="#TipoRelacionamento"/>
  <rdfs:range rdf:resource="#Propriedade"/>
</owl:ObjectProperty>

<Propriedade rdf:ID="simetria"/>
<Propriedade rdf:ID="transitividade"/>

<TipoRelacionamento rdf:ID="tipoAgregacao">
  <temPropriedade>
    <Propriedade rdf:ID="transitividade"/>
  </temPropriedade>
</TipoRelacionamento>
```

A classe *Regra* é constituída por uma composição entre as classes *Antecedente* e *Conseqüente*. Cada regra possui apenas um antecedente e um conseqüente, o que é expresso através da propriedade funcional da OWL.

```
<owl:ObjectProperty rdf:ID="temAntecedente">
  <rdfs:domain rdf:resource="#Regra"/>
  <rdfs:range rdf:resource="#Antecedente"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>

<owl:InverseFunctionalProperty rdf:ID="temConsequente">
  <rdfs:domain rdf:resource="#Regra"/>
  <rdfs:range rdf:resource="#Consequente"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:InverseFunctionalProperty>
```

Um antecedente pode possuir associações e descrições. A classe *Antecedente* se relaciona com elas, respectivamente, através dos metarelacionamentos *antecedenteTemAssociação* e *antecedenteTemDescrição*. Já o conseqüente possuirá apenas uma associação ou uma descrição, e por isso os metarelacionamentos *consequenteTemAssociação* e *consequenteTemDescrição*, que os associam com o elemento do tipo *Associação*, possuem a propriedade funcional. Os dois casos podem ser vistos a seguir:

```
<owl:Class rdf:ID="Antecedente"/>

<owl:ObjectProperty rdf:ID="antecedenteTemAssociacao">
  <rdfs:range rdf:resource="#Associacao"/>
  <rdfs:domain rdf:resource="#Antecedente"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="antecedenteTemDescricao">
  <rdfs:range rdf:resource="#Descricao"/>
  <rdfs:domain rdf:resource="#Antecedente"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Consequente"/>

<owl:FunctionalProperty rdf:ID="consequenteTemAssociacao">
  <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Associacao"/>
  <rdfs:domain rdf:resource="#Consequente"/>
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="consequenteTemDescricao">
  <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Descricao"/>
  <rdfs:domain rdf:resource="#Consequente"/>
</owl:FunctionalProperty>
```

Como visto na seção 4.2, um antecedente é constituído por descrições e associações, enquanto um conseqüente é composto por uma associação ou uma descrição. A classe *Associação* é composta por dois literais (através do metarelacionamento *associação entre*) e um relacionamento (através de *associa relacionamento*), com restrições de cardinalidade 2 e 1, respectivamente, vistas a seguir:

```
<owl:Class rdf:ID="Associacao">
  <owl:Restriction>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >2</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="associacaoEntre"/>
    </owl:onProperty>
  </owl:Restriction>
  <owl:Restriction>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="associaRelacionamento"/>
    </owl:onProperty>
  </owl:Restriction>
</owl:Class>

<owl:ObjectProperty rdf:about="#associacaoEntre">
  <rdfs:domain rdf:resource="#Associacao"/>
  <rdfs:range rdf:resource="#Literal"/>
```

```
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#associaRelacionamento">
  <rdfs:domain rdf:resource="#Associacao"/>
  <rdfs:range rdf:resource="#Relacionamento"/>
</owl:ObjectProperty>
```

A classe *Descrição* é representada de maneira análoga, sendo que a única diferença é a restrição do relacionamento *descrição de*, que apresenta cardinalidade 1, visto que uma associação relaciona dois literais, enquanto a descrição refere-se a apenas um literal. Seus metarelacionamentos são *descrição de* e *descreve LO*, que relacionam a descrição a, respectivamente, literais e LOs.

```
<owl:Class rdf:ID="Descricao">
  <owl:Restriction>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="descricaoDe"/>
    </owl:onProperty>
  </owl:Restriction>
  <owl:Restriction>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="descreveLO"/>
    </owl:onProperty>
  </owl:Restriction>
</owl:Class>

<owl:ObjectProperty rdf:about="#descricaoDe">
  <rdfs:domain rdf:resource="#Descricao"/>
  <rdfs:range rdf:resource="#Literal"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#descreveLO">
  <rdfs:domain rdf:resource="#Descricao"/>
  <rdfs:range rdf:resource="#LOLogico"/>
</owl:ObjectProperty>
```

Com todas as classes expressas, é possível instanciar a regra no nível de metamodelo. A seguir, a ilustração da regra *pré-requisito*, e de seu meta-relacionamento com o antecedente e o conseqüente que a constituem, bem como a instanciação do conseqüente da regra.

```
<Regra rdf:ID="preRequisito">
  <temConsequente>
    <Consequente rdf:ID="conseq_preRequisito">
      <consequenteTemAssociacao rdf:resource="#ehPreRequisito_a_b"/>
```

```
    </Consequente>
  </temConsequente>
  <temAntecedente rdf:resource="#antec_preRequisito"/>
</Regra>
```

A seguir, é ilustrada a maneira como é representado o antecedente da regra *pré-requisito*, com suas associações e descrições.

```
<Antecedente rdf:ID="antec_preRequisito">
  <antecedenteTemAssociacao>
    <Associacao rdf:ID="compreende_a_x">
      <associacaoEntre rdf:resource="#x"/>
      <asociaRelacionamento rdf:resource="#compreende"/>
      <associacaoEntre>
        <Literal rdf:ID="a"/>
      </associacaoEntre>
      <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >compreende(a,x)</nomeAssociacao>
    </Associacao>
  </antecedenteTemAssociacao>
  <temDescricao>
    <Descricao rdf:ID="topico_y">
      <descricaoLO rdf:resource="#topico"/>
      <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >topico(y)</nomeDescricao>
      <descritoPor>
        <Literal rdf:ID="y"/>
      </descritoPor>
    </Descricao>
  </temDescricao>
  <temDescricao>
    <Descricao rdf:ID="disciplina_b">
      <descritoPor rdf:resource="#b"/>
      <descricaoLO rdf:resource="#disciplina"/>
      <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >disciplina(b)</nomeDescricao>
    </Descricao>
  </temDescricao>
  <temDescricao>
    <Descricao rdf:ID="disciplina_a">
      <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >disciplina(a)</nomeDescricao>
      <descricaoLO rdf:resource="#disciplina"/>
      <descritoPor rdf:resource="#a"/>
    </Descricao>
  </temDescricao>
  <temDescricao rdf:resource="#topico_x"/>
  <antecedenteTemAssociacao>
    <Associacao rdf:ID="fundamenta_x_y">
      <asociaRelacionamento rdf:resource="#fundamenta"/>
      <associacaoEntre rdf:resource="#y"/>
      <associacaoEntre rdf:resource="#x"/>
      <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >fundamenta(x,y)</nomeAssociacao>
    </Associacao>
```

```
  </antecedenteTemAssociacao>
  <antecedenteTemAssociacao>
    <Associacao rdf:ID="compreende_b_y">
      <associacaoEntre rdf:resource="#y"/>
      <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >compreende(b,y)</nomeAssociacao>
      <asociaRelacionamento rdf:resource="#compreende"/>
      <associacaoEntre rdf:resource="#b"/>
    </Associacao>
  </antecedenteTemAssociacao>
</Antecedente>
```

Desta forma, temos a representação de uma regra, que na camada inferior da arquitetura é representada na linguagem SWRL.

5.4 RACIOCINANDO EM ONTOLOGIAS

Um raciocinador (ou *reasoner*) de lógica descritiva é um sistema de representação de conhecimento baseado em lógica descritiva que provê formas de manipular bases de dados e de extrair conhecimento sobre seu conteúdo [NARDI e BRACHMAN, 2003].

Para pesquisas que inferem conhecimento representado através de propriedades de relacionamentos ou de regras, como as pretendidas pelo ROSA^{+}, torna-se necessária a presença de um raciocinador de lógica descritiva em sua arquitetura.

Os raciocinadores oferecidos para a lógica descritiva provêm suporte ao desenvolvimento e manutenção de uma ontologia. Podem ser usadas implementações altamente otimizadas para a verificação e algoritmos de classificação de quadros (*tableaux*) para muitas descrições lógicas expressivas. Assim, uma ontologia expressa em OWL pode ser verificada usando-se um raciocinador de lógica descritiva. Os raciocinadores provêm funcionalidades como:

- checagem de classificação entre duas descrições de conceitos: um conceito A inclui outro B, quando o conjunto de objetos que são instâncias de B é sempre um subconjunto de objetos que são instâncias de A;
- classificação: organiza uma coleção de expressões de conceitos dentro de uma ordem parcial baseada na checagem de classificação. Isto provê uma rede simétrica de definições, dispondo-se do geral para o específico. As definições

compostas têm sua posição implicitamente determinada automaticamente. Assim, a classificação é um processo dinâmico que pode ser adicionado a uma hierarquia existente;

- satisfabilidade (ou insatisfabilidade) de conceitos: um conceito insatisfatível é aquele que não é verdadeiro para qualquer instância.

A OWL-DL é equivalente a uma descrição lógica muito expressiva, a *SHOIN(D)* [DE BRUIJIN et al., 2005]. Esta equivalência permite à OWL explorar uma parte considerável das pesquisas em descrições lógicas.

Assim, para a inferência de conhecimento sobre as ontologias que definem a arquitetura ROSA$^+$, torna-se necessária a utilização de um raciocinador de lógica descritiva.

Em Mattos [MATTOS et al., 2005], foi realizado um estudo sobre os raciocinadores de lógica descritiva RACER [HAARSLEV e MÖLLER, 2003], Pellet [PARSIA e SIRIN, 2004] e FaCT++ [FACT]. Todos eles realizam inferência sobre OWL-DL de maneira eficiente. Porém, outro raciocinador não contemplado em tal estudo, o Bossam [JANG, 2005], foi o que melhor funcionou ao raciocinar de forma satisfatória sobre regras em SWRL.

Bossam é um mecanismo de inferências para a Web Semântica baseado no algoritmo RETE[18] [FORGY, 1982]. Ele apresenta uma linguagem própria, a Buchingae [JANG, 2005], além da capacidade de raciocinar sobre linguagens da Web Semântica como XML, RDF, OWL, RuleML e SWRL. Outra vantagem do Bossam é que suas APIs o tornam uma ferramenta simples de ser integrada a aplicações Java. Estes fatores tiveram importância para que o Bossam fosse a ferramenta de raciocínio escolhida para o desenvolvimento deste trabalho.

Com a escolha de um raciocinador para inferir conhecimento sobre regras em SWRL e propriedades da OWL-DL, torna-se possível avançar para a etapa de definição e especificação do ROSA$^+$, conforme mostra a seção seguinte.

5.5 O SISTEMA ROSA$^+$

[18] Algoritmo que encontra padrões, para implementações de sistemas baseados em regras

Após realizadas as etapas de definição e representação da arquitetura ROSA$^+$ através de uma ontologia, foi realizada a próxima etapa do trabalho, que inclui a especificação dos módulos que compõem o sistema proposto.

5.5.1 FUNCIONALIDADES

O sistema ROSA$^+$ é composto pelos seguintes módulos: edição de regras e consultas.

O módulo de edição de regras permite ao usuário editar as regras que serão incorporadas à base de dados do sistema, e sobre as quais o usuário poderá realizar consultas. Ele possui as seguintes funcionalidades:

- Inserção de regras: pode-se criar uma regra, dando ao usuário a possibilidade de escolher o nome da regra, o número de LOs que a comporão, as descrições e associações que comporão o antecedente e a associação que comporá o conseqüente da regra;
- Alteração de regras: dá ao usuário a possibilidade de alterar elementos da regra, como nome, podendo também inserir e excluir novas associações e descrições;
- Exclusão de regras: pode-se excluir a regra selecionada da base de dados.

O módulo de consultas permite ao usuário realizar consultas sobre LOs, relacionamentos e regras, e nos três níveis representados pelas ontologias ROSA$^+$:

- **Consulta sobre LO**
    - **Nível do mapa conceitual**: retorna todos os LOs com os quais o LO Lógico no nível de instância selecionado pelo usuário se relaciona, seja diretamente, através de relacionamento, ou indiretamente, inferindo-se conhecimento sobre as propriedades de relacionamentos ou de regras. Retorna também o tipo do LO selecionado (a classe a qual a instância pertence).

- **Nível do mapa de domínio**: retorna informações sobre o LO Schema escolhido pelo usuário, suas instâncias, se tiver, e o tipo de cada uma.
- **Nível do modelo ROSA$^+$**: a partir das duas opções fornecidas ao usuário (LO Lógico e LO Schema), retorna suas instâncias.

- **Consulta sobre relacionamento**
  - **Nível do mapa conceitual**: exibe todas as associações entre LOs Lógicos no nível de instância que possuem o relacionamento selecionado como predicado.
  - **Nível do mapa de domínio**: retorna as associações entre LOs Schema que possuem o relacionamento selecionado como predicado.
  - **Nível do modelo ROSA$^+$**: neste nível, é possível fazer duas consultas distintas: a relacionamentos ou a tipos de relacionamentos. Selecionado um relacionamento, retorna o tipo ao qual o relacionamento pertence, suas propriedades, quais relacionamentos são sinônimos e quais são inversos. Se o usuário selecionar um tipo de relacionamento, o sistema traz como resposta os relacionamentos pertencentes ao tipo selecionado, além de suas propriedades.

- **Consulta a regras**
  - **Nível do mapa conceitual**: retorna as associações entre LOs Lógicos no nível de instâncias inferidas através de regras.
  - **Nível do mapa de domínio**: retorna os tipos de LOs que se relacionam através de alguma regra.
  - **Nível do modelo ROSA$^+$**: retorna a descrição da regra selecionada, com associações e descrições que compõem seu antecedente e conseqüente.

Na próxima seção, será abordada a especificação técnica do sistema ROSA$^+$.

5.5.2 VISÃO GERAL

A arquitetura geral do sistema ROSA$^+$ é composta por três camadas: interface com o usuário, execução lógica e persistência, conforme pode ser vista na figura 5.3.

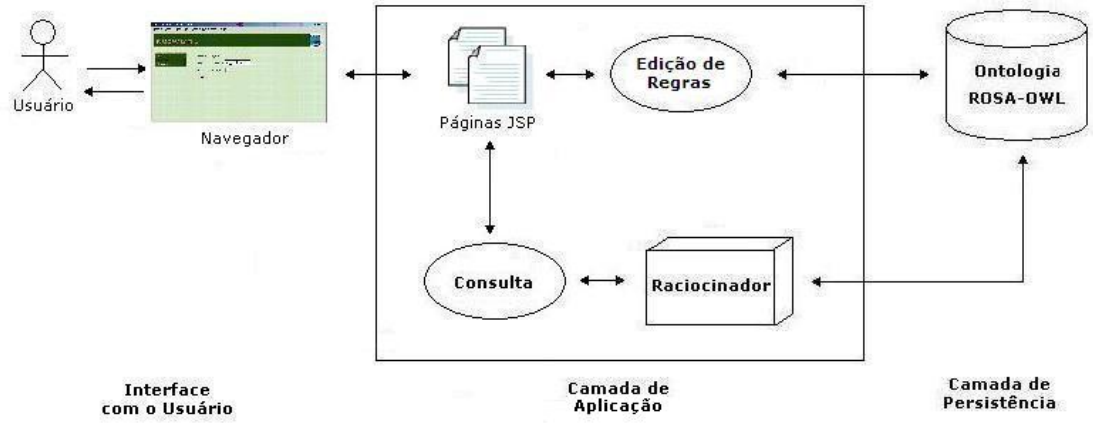

FIG. 5.3 Visão Geral do Sistema ROSA$^+$.

Inicialmente, a camada de interface representa a interação do usuário com o sistema. Através desta, o usuário está apto a interagir com o navegador, inserindo, alterando e excluindo regras na base de dados, além de poder realizar consultas que acessam dados nas três camadas da arquitetura ROSA$^+$ (mapa conceitual, mapa de domínio e modelo ROSA$^+$), sem necessitar conhecer detalhes sobre a implementação lógica do sistema.

A camada de persistência é responsável por armazenar os arquivos OWL e SWRL onde estão expressas as ontologias (Mapa Conceitual/Mapa Domínio e Mapa Domínio/Modelo ROSA+), que constituem a base de dados do sistema. Destes arquivos serão extraídos os elementos que constituem as quatro camadas da arquitetura ROSA$^+$ (figura 5.3), formando o que denominamos base de conhecimento do sistema, localizada na camada de execução lógica.

Na camada de aplicação encontra-se a classe que gera as consultas especificadas pelo usuário. A execução das consultas depende de um raciocinador, que processará a consulta sobre um arquivo da camada de persistência, retornando o resultado obtido pela inferência. Fazem parte ainda da camada de aplicação as páginas JSP, de conteúdo dinâmico, responsáveis tanto por expor ao usuário as interfaces de edição de regras e de consultas à base do sistema, quanto por exibir os resultados das consultas.

5.6 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo apresentar a especificação do sistema ROSA$^+$. Antes disso, foi realizado um estudo sumarizado sobre editores de ontologias, oriundo de Mattos et. al. [MATTOS et. al, 2005], que serviu como base para a escolha do Protégé como ferramenta utilizada para desenvolver a ontologia para representar a arquitetura proposta.

Para representar uma arquitetura de múltiplas camadas na linguagem OWL-DL, foi necessário o desmembramento do modelo em duas ontologias, devido a uma limitação importante dessa linguagem, que não permite que um objeto possa ser representado como classe e instância ao mesmo tempo.

Foi no ambiente do Protégé, em sua versão 3.1.1, onde foram implementadas, na linguagem OWL, as duas ontologias que representam a arquitetura ROSA$^+$, juntamente com regras em SWRL. Estas duas ontologias compõem a camada de persistência da arquitetura do sistema ROSA$^+$.

No entanto, as ontologias desenvolvidas no Protégé apresenta determinadas contruções que não são interpretáveis pelo raciocinador Bossam. Assim, foi necessário adaptar o código gerado pelo Protégé para o do Bossam de modo a permitir o raciocínio sobre a ontologia.

Foram abordadas também funcionalidades do sistema e aspectos técnicos para o desenvolvimento do sistema. No próximo capítulo, será apresentada a prototipação do sistema, com exemplos de edição e de consultas para ilustrar essas funcionalidades.

6 PROTOTIPAÇÃO DO SISTEMA ROSA+

Este capítulo objetiva mostrar aspectos de implementação do sistema ROSA+ a partir de um estudo de caso que permite ilustrar cada uma das funcionalidades descritas no capitulo anterior. Na seção 6.1 são apresentadas algumas características do ambiente de implementação. A seção 6.2 apresenta a descrição do estudo de caso que servirá como base para ilustrar cada uma das funcionalidades propostas e desenvolvidas no sistema. Na seção 6.3, será abordado o módulo de edição de regras, exemplificando operações efetuadas sobre regras. E na seção 6.4, algumas consultas serão mostradas para exemplificar o módulo de consultas do sistema.

6.1 IMPLEMENTAÇÃO

O ROSA+ foi desenvolvido na linguagem Java[19], versão 1.5. O ambiente de desenvolvimento utilizado foi o Eclipse[20], em sua versão 3.0, e o servidor de aplicação foi o Apache Tomcat[21] 4.0. A maquina onde foi desenvolvida a aplicação foi um PC com processador Pentium IV e 512 MB de memória.

Assim, foi possível implementar um sistema com as funções planejadas: extração, alteração e inserção de regras expressas nas linguagens OWL e SWRL; rotinas de manipulação do sistema via Web; e execução das consultas através do raciocinador Bossam.

Para auxíliar no desenvolvimento, foram utilizadas bibliotecas Java dos pacotes Protégé e Bossam. Para o módulo de edição de regras, foram utilizadas bibliotecas do pacote *edu.stanford.smi.protegex.owl* do Protégé. Através das interfaces deste pacote, é possível representar classes, relacionamentos e instâncias de uma ontologia como classes Java. Assim, torna-se possível inserir, alterar ou apagar regras, já que depois é feita a reconstrução do documento no formato OWL, realizada também com o auxílio dessa mesma biblioteca.

[19] http://java.sun.com/
[20] http://www.eclipse.org/
[21] http://tomcat.apache.org/

Para execução de consultas, o Bossam oferece duas interfaces que foram utilizadas no sistema: *IReasoner*, que provê métodos para carregar regras SWRL, e ontologias OWL, adicionando às mesmas novos fatos e regras necessários para executar inferências; e *IReasonerFactory*, que cria um raciocinador. A partir delas, pode-se realizar consultas a todos os elementos das ontologias, utilizando-se dessa base de regras.

6.2 ESTUDO DE CASO

Para validar o sistema ROSA^{+}, foi proposto um estudo de caso onde foram ilustradas as funcionalidades de edição de regras e realização de consultas.

A primeira etapa deste estudo de caso consiste em representar um dominio. Foi adotado parte do curso de Sistemas e Computação do IME neste estudo de caso, conforme ilustrado na figura 6.1 a seguir.

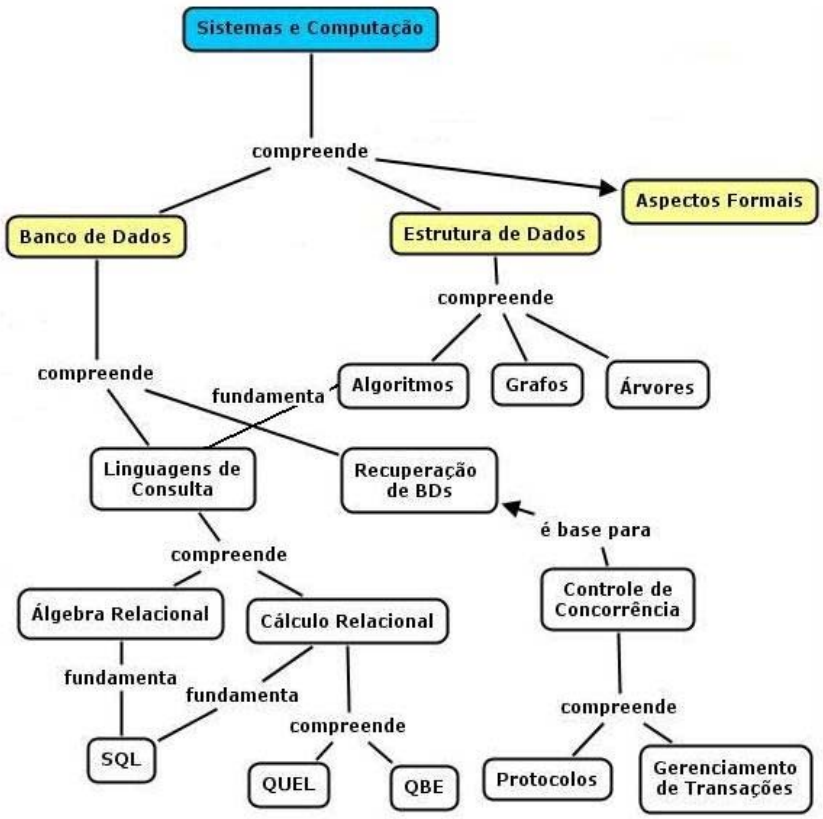

FIG. 6.1 Mapa Conceitual da Ontologia Utilizada no Estudo de Caso.

A partir deste mapa conceitual, foram instanciadas a Ontologia Mapa Conceitual/Mapa Domínio e a Ontologia Mapa Domínio/Modelo ROSA^{+} correspondentes. Estas servirão como base de dados para o estudo de caso a ser realizado no decorrer deste capítulo.

A figura mostra a tela inicial do ROSA^{+}.

FIG. 6.2 Tela Inicial do ROSA^{+}.

6.3 MÓDULO DE EDIÇÃO DE REGRAS

Conforme mencionado anteriormente, o módulo de edição de regras permite a inserção, alteração e exclusão regras da base de dados.

Ao selecionar a opção inserção de regras, após se definir o nome da regra a ser inserida e o número de LOs que será necessário para compô-la, a página JSP chama a classe *OWLProcessor*. Através dela, as instâncias das classes *Literal* e *LOSchema* da Ontologia Mapa Dominio/Modelo ROSA^{+} são armazenadas na sessão. Estes elementos

são exibidos em páginas JSP, para que sejam definidas as associações e descrições que comporão a regra. Estas serão armazenadas como objetos de classes Java do sistema. Terminada esta etapa, os elementos da regra são transformados, através da interface *SWRLFactory*, nas devidas classes, também através da classe *OWLProcessor*, são transformados novamente em arquivo OWL, tanto para serem salvos no formato SWRL (na Ontologia Mapa Conceitual/Mapa Domínio) quanto como instâncias de classes (na Ontologia Mapa Domínio/Modelo ROSA^{+}).

A alteração de regras ocorre de forma semelhante. Elementos da base de dados OWL são carregados como objetos e armazenados em sessão, e os dados sobre a regra selecionada são carregados nas páginas JSP, onde poderão ser alterados. A seguir, os objetos editados são transformados para os formatos SWRL (Ontologia Mapa Conceitual/Modelo ROSA^{+}) e OWL (Ontologia Mapa Dominio/Modelo ROSA^{+}).

Na exclusão, a regra selecionada é eliminada, ocorrendo processamento similar. Ao selecionar a regra, a classe *OWLProcessor* exclui o trecho referente à regra no formato SWRL da Ontologia Mapa Conceitual/Mapa Domínio, além de excluir também as instâncias das classes *Regra, Antecedente, Conseqüente, Associação* e *Descrição* da Ontologia Dominio/Modelo ROSA^{+}.

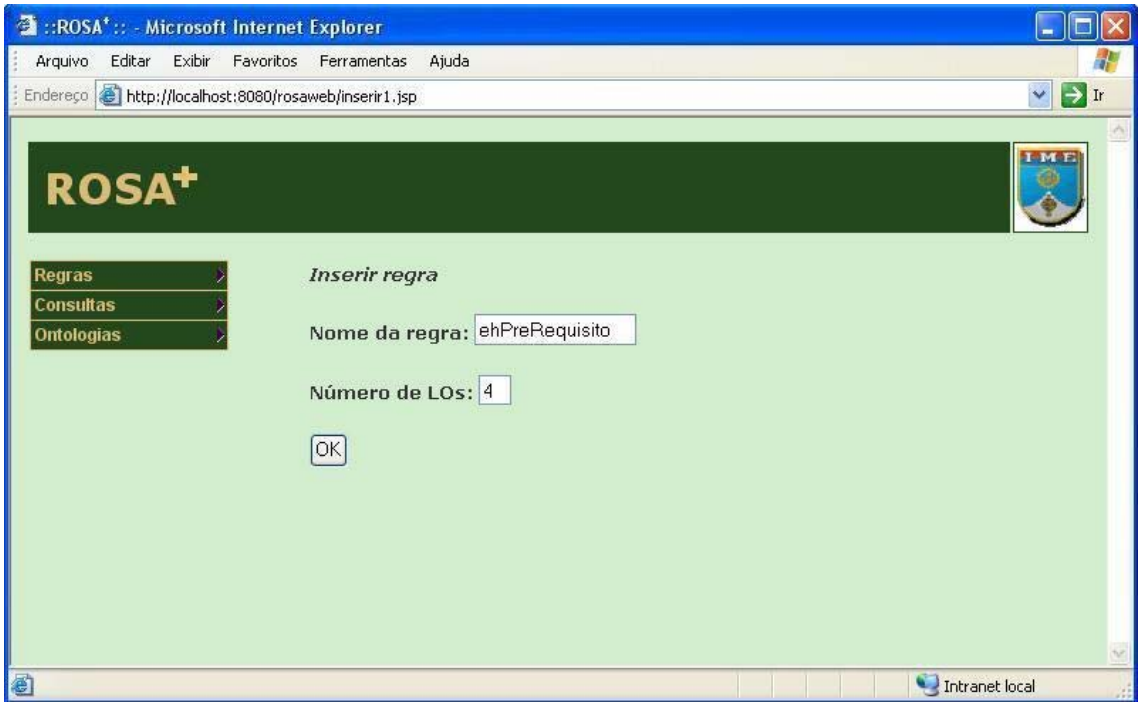

FIG. 6.3 Definição do nome de regra e número de LOs.

De forma a ilustrar a inserção de uma regra no ROSA^{+}, considere a inserção da regra *ehPreRequisito* definida no capítulo 4. Neste estudo de caso, após escolher no menu a opção "inserir regra", uma tela é exibida para que sejam preenchidos o nome da regra e o número de LOs necessários para representá-la, conforme pode ser visto na figura 6.3.

Inseridos os dados, eles são armazenados em memória. A página JSP então, através da classe *OWLProcessor*, carrega em caixas de seleção os literais e os LOs Schema definidos na Ontologia Mapa Domínio/Modelo ROSA^{+}. Será possível então definir, para cada LO da regra, o literal que o representará e a qual tipo estará associado. Na figura 6.4 é apresentada a tela do sistema que representa essas opções no exemplo de inserção da regra *ehPreRequisito*.

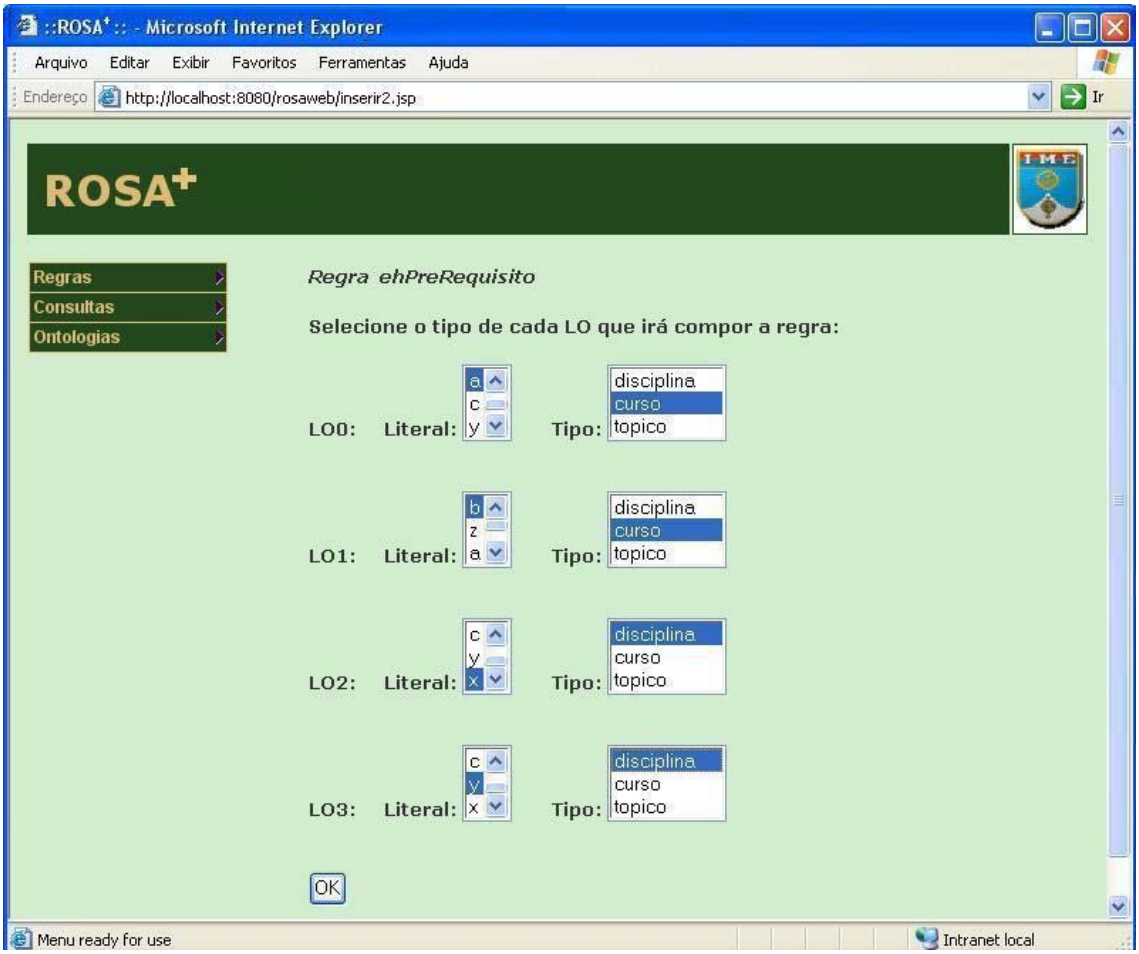

FIG. 6.4 Definição de cada LO que compõe a regra.

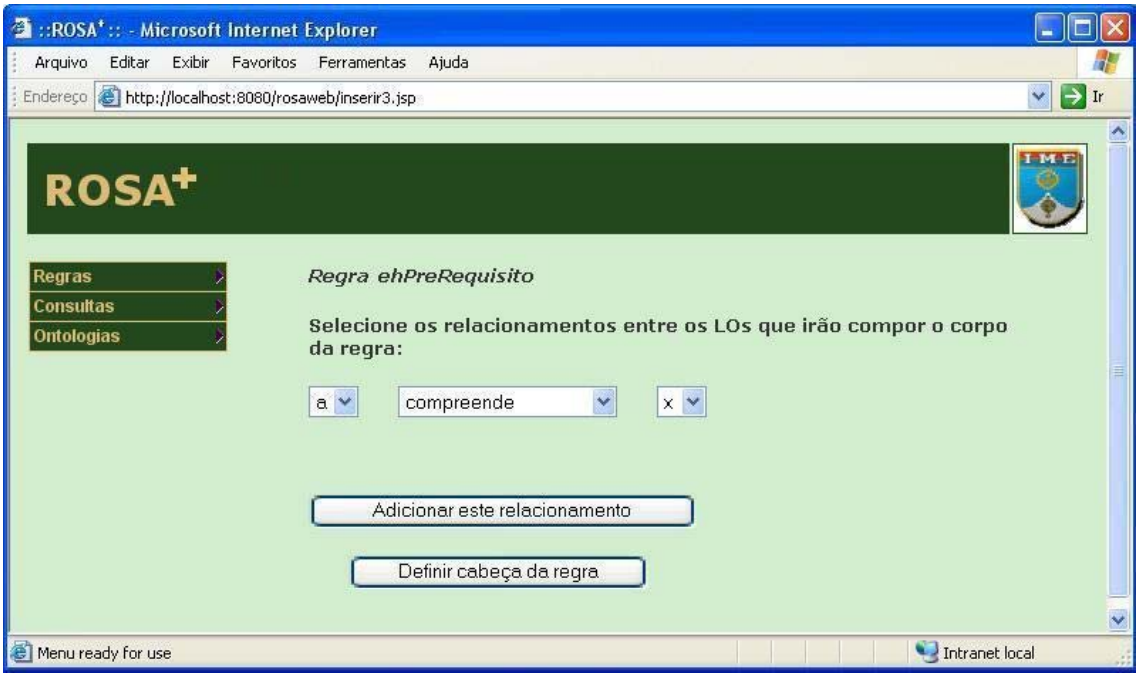

FIG. 6.5 Definição de uma associação do corpo da regra.

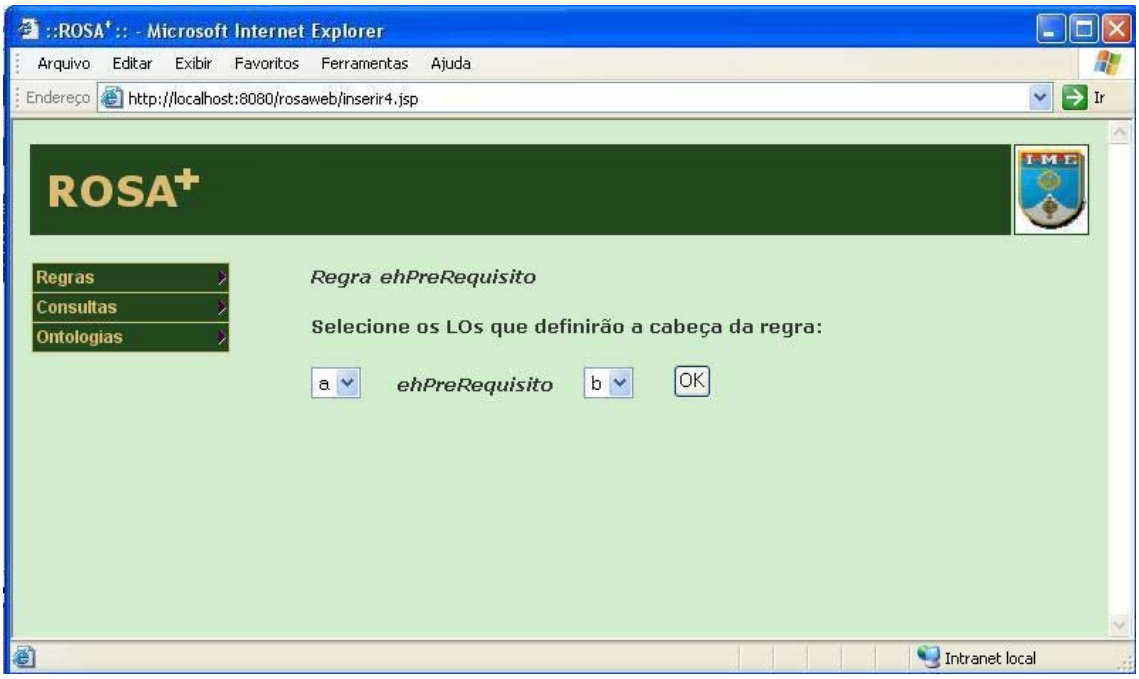

FIG. 6.6 Definição da cabeça da regra.

Os LOs definidos, que serão armazenados como objetos da classe *LOLiteral*, serão utilizados nas associações que comporão, primeiramente, o antecedente da regra. Nesse caso, todos os relacionamentos instanciados na Ontologia Mapa Domínio/Modelo ROSA$^+$ são fornecidos para que sejam associados através desses literais (figura 6.5). O conseqüente é definido de maneira semelhante, porém nele seleciona-se primeiramente

se este será uma descrição ou uma associação; para apenas posteriormente defini-la. No caso da associação, ele não seleciona um relacionamento: a associação entre os LOs é realizada através do nome da regra já definido anteriormente (figura 6.6).

Estes dados também são armazenados como objetos e em memória, para serem carregados por páginas JSP posteriormente. A classe *OWLProcessor* carrega então os dados referentes à regra definida, exibindo-os para que este confirme a regra (figura 6.7).

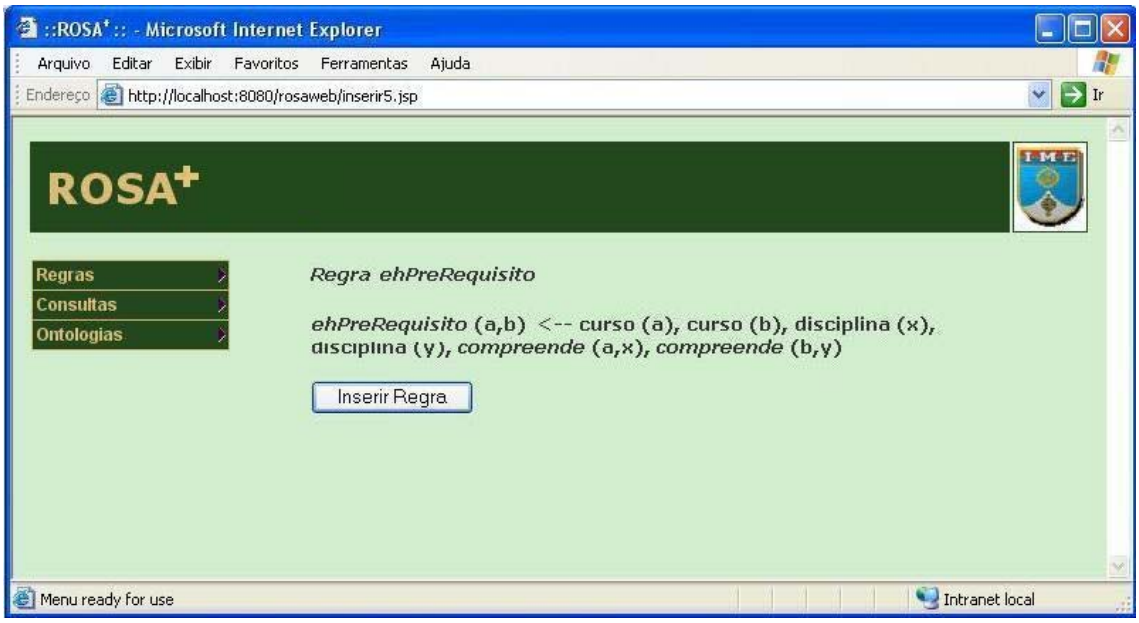

FIG. 6.7 Confirmação da regra a ser inserida.

Uma vez a inserção desta regra é confirmada, a classe *OWLProcessor* transforma os elementos definidos durante a sessão em objetos das classes do sistema referentes às classes da Ontologia Mapa Domínio/Modelo ROSA^{+}, onde eles serão salvos. Além disso, a *OWLProcessor* utiliza métodos importados da classe *SWRLFactory* para que estes mesmos elementos sejam salvos como uma regra definida na linguagem SWRL. Esta é então incorporada ao documento referente à Ontologia Mapa Conceitual/Mapa Domínio, que é por fim salvo.

A alteração de regras funciona da seguinte maneira: ao selecionar a regra a ser redefinida, a classe *OWLProcessor* carrega em memória todos os dados referentes a esta (literais, definições e associações do antecedente e do conseqüente), sendo que poderão ser alterados através das mesmas telas vistas no exemplo de inserção de regras. Depois,

as alterações realizadas e salvas em memória são incorporadas às ontologias da mesma forma que no método de inserção.

Para apagar uma regra da base de dados, deve-se apenas escolher uma entre todas as regras existentes na base de dados. A classe *OWLProcessor* chama então os métodos contidos nas classes do pacote *edu.stanford.emi.protegex.owl.swrl*, que exclui a regra selecionada do documento referente à Ontologia Mapa Conceitual/Mapa Domínio. A instância da classe *Regra* na Ontologia Mapa Domínio/Modelo ROSA$^+$ também é excluída neste processo. Os documentos são então salvos novamente, já sem a regra.

6.4 MÓDULO DE CONSULTA

Conforme visto na seção 5.5.2, sistema ROSA$^+$ apresenta um módulo de edição de consultas que permite a realização de consultas sobre LOs, relacionamentos e regras, nos três níveis representados pelas ontologias ROSA$^+$.

Após selecionado o tipo e o nível de consulta a ser realizada (figura 6.8), são carregadas nas páginas JSP as instâncias da elementos sobre qual elemento a classe *Consulta* constrói a consulta selecionada, adicionando a mesma o LO selecionado previamente. Montada a consulta, ela é realizada sobre a ontologia através do raciocinador, e o resultado é então armazenado em memória e carregado de volta à interface com o usuário pela página JSP.

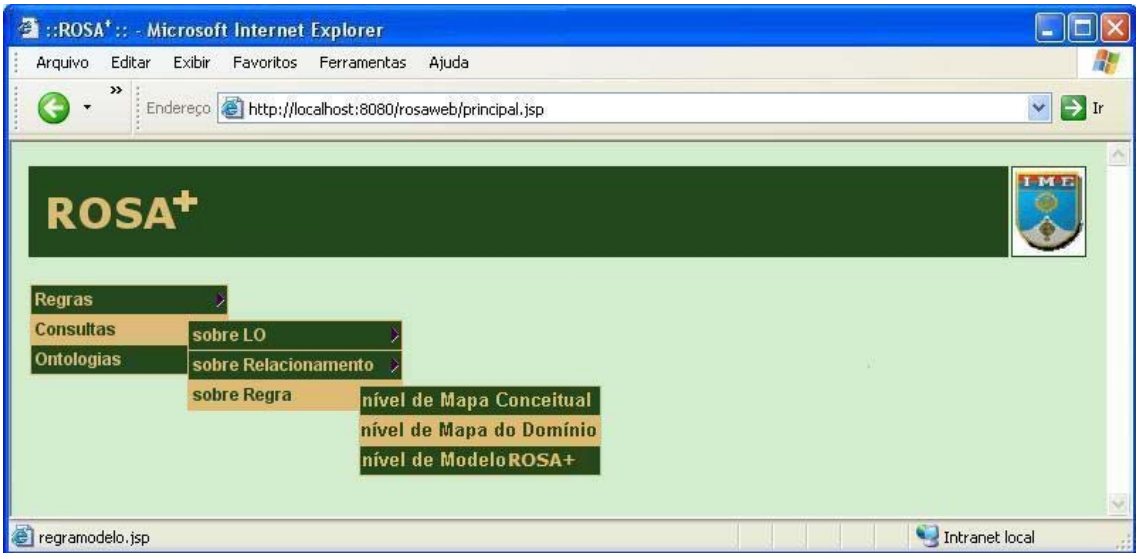

FIG. 6.8 Menu de Consultas.

As interfaces do módulo de consultas apresentam estrutura semelhante, onde, primeiramente, uma tela é exibida com as opções em uma caixa. Ao selecionar uma das opções, tem-se como resposta uma tela indicando os resultados da consulta sobre o item selecionado.

A seguir, serão listadas algumas consultas e os resultados obtidos no ROSA$^+$ sobre a base de dados definida na figura 6.1:

1) Consulta sobre LO no nível de mapa conceitual: esta consulta visa trazer como resultado tanto relacionamentos entre LOs instanciados na Ontologia Mapa Conceitual/Domínio quanto inferidos através de propriedades de relacionamentos e regras. Desta forma, uma consulta sobre o LO *Estrutura de Dados*, além de trazer como resultado que o mesmo compreende os LOs *Árvores, Grafos* e *Algoritmos*, também exibe relacionamentos não instanciados, como *ehCompreendidoPor* (figura 6.9 (a)) e *ehPreRequisito* (figura 6.9 (b)).

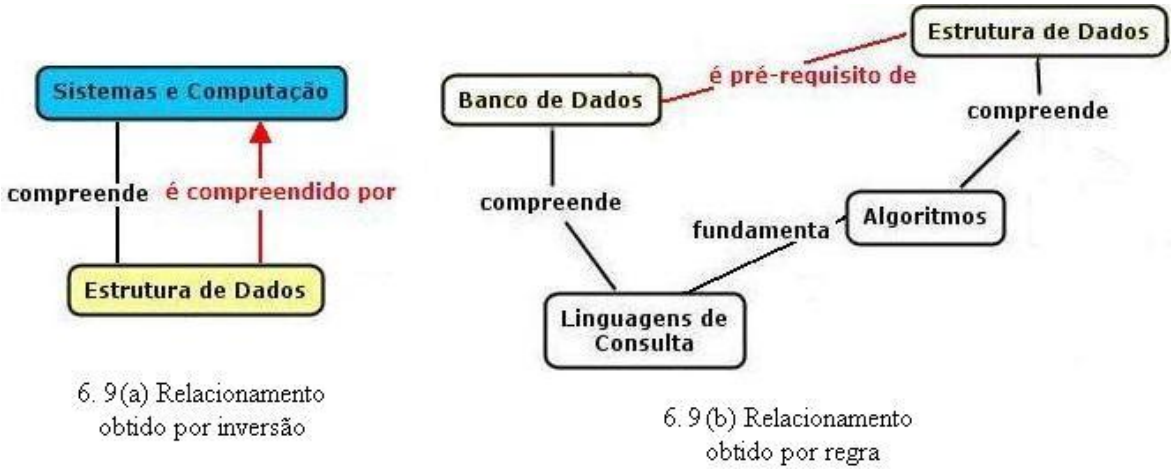

FIG. 6.9 Relacionamentos Não-Explícitos do LO *Estrutura de Dados.*

Após selecionar esta opção de consulta, qualquer LOs Lógico instanciado na Ontologia Mapa Domínio/Modelo ROSA$^+$ poderá ser selecionado, conforme pode ser visto na figura 6.10.

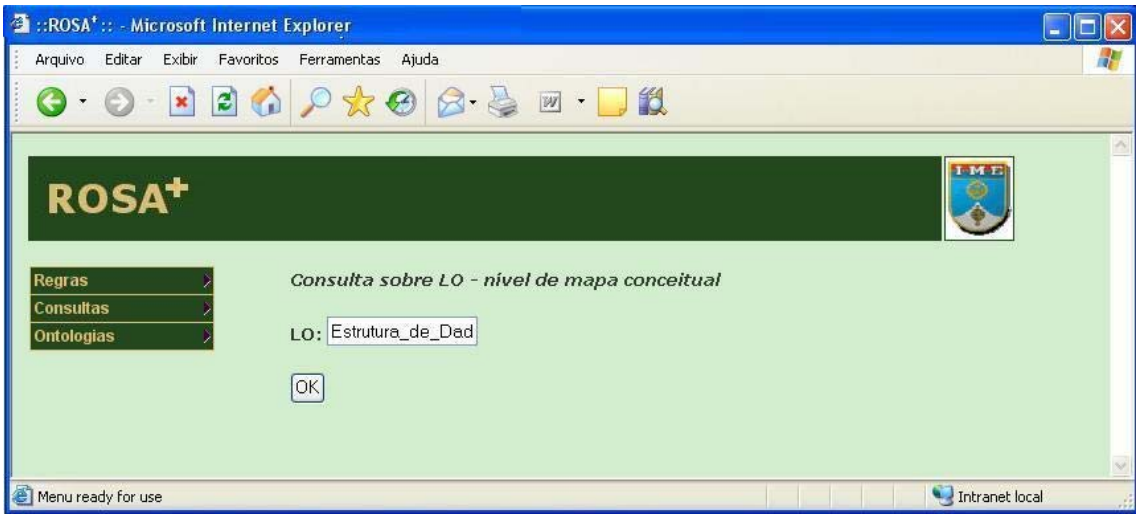

FIG. 6.10 Seleção de LO para consulta no nível de mapa conceitual.

Após o LO *Estrutura de Dados* ter sido selecionado, o sistema utiliza a classe *Reasoner*, que, através do LO selecionado, busca na Ontologia Mapa Conceitual/Mapa Domínio todos os relacionamentos que possuem o LO selecionado como domínio. A classe *Reasoner* também é responsável por construir a consulta que será feita à base de dados via raciocinador, concatenando o nome do LO selecionado a uma consulta já pronta e representada nesta mesma classe. O resultado da consulta é exibido na figura 6.11.

Assim, a consulta fornece como resultado que o LO *Estrutura de Dados*:

- compreende os tópicos *Algoritmos*, *Grafos* e *Árvores*. Estas informações foram explicitamente instanciadas na ontologia;
- todos os relacionamentos equivalentes a *compreende* (envolve, possui, etc.) são inferidos;
- é compreendido pelo curso *Sistemas e Computação*, resultado obtido por inferência. Observe que foi declarado na ontologia que *Sistemas e Computação* compreende *Estrutura de* Dados, e que o relacionamento *ehCompreendidoPor* é inverso a *compreende*;
- é pré-requisito da disciplina *Banco de Dados*, já que as condições para que este relacionamento seja inferido ocorrem, i.e., os elementos do antecedente da regra *ehPreRequisito* foram instanciados na ontologia. De fato, está instanciado na ontologia que: (1) *Estrutura de Dados* e *Banco de Dados* são disciplinas, (2)

Algoritmos e *Linguagens de Consulta* são tópicos, e (3) *Algoritmos* fundamentam *Linguagens de Consulta*;

FIG. 6.11 Resultado da consulta sobre o LO *Estrutura de Dados.*

- Tem em ementa os tópicos *Algoritmos*, *Grafos* e *Árvores* (obtido por inferência sobre a regra *ementa*).

2) Consulta sobre LO no nível de mapa do domínio: esta consulta permite listar as instâncias de um tipo, ou seja, os LOs Lógicos associados a determinado LO Schema. No mapa conceitual do ROSA$^+$, os LOs Schema definidos são *Curso*, *Disciplina* e *Tópico*.

Caso o LO Schema escolhido faça parte de uma hierarquia, é possível escolher uma de suas subclasses para realizar consulta através de métodos da classe *OWLProcessor*. Assim, ao escolher uma consulta sobre curso, por exemplo, pode-se escolher uma de suas subclasses (*mestrado, doutorado* ou *EAD*).

Desta forma, uma consulta por *curso*, no nível de *Mestrado* (figura 6.12) traria como resultado o LO *Sistemas e Computação* (figura 6.13).

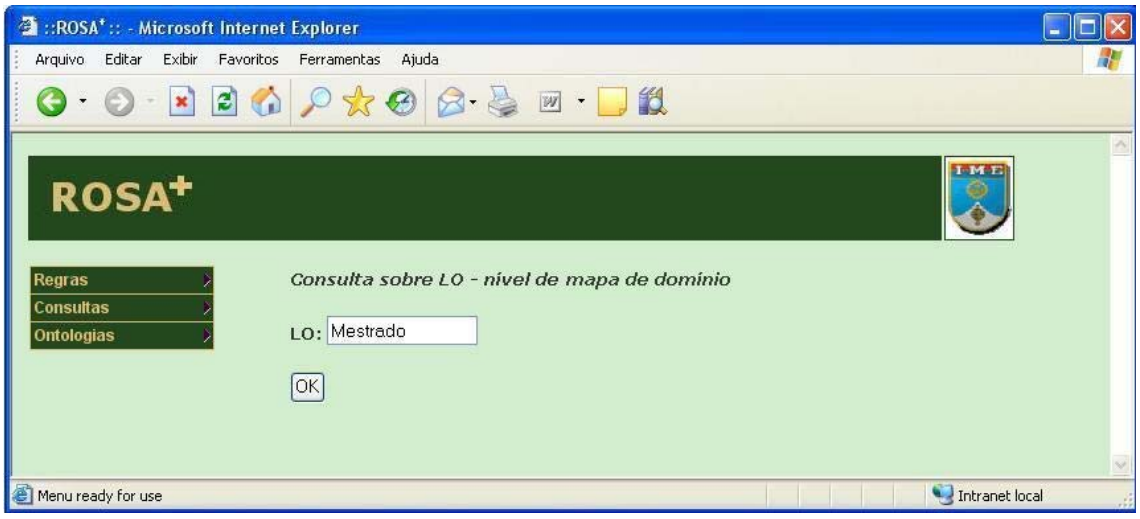

FIG. 6.12 Seleção de LO para consulta no nível de mapa do domínio.

3) Consulta sobre LO no nível de modelo Modelo ROSA$^+$: esta consulta apresenta duas opções: consultar sobre as instâncias de LOs Lógicos ou LOs Schema. Ao selecionar uma das opções, os métodos *getLOLogico* e *getLOSchema* retornam, respectivamente, todos os LOs Lógicos e LOs Schema instanciados na Ontologia Mapa Domínio/Modelo ROSA$^+$.

Por exemplo, uma consulta sobre LO Schema (figura 6.14) traria como resultados os LOs *Curso*, *Disciplina* e *Tópico* (figura 6.15). Já a consulta sobre LO Lógico

retornaria os LOs Lógicos da base de dados (*Sistemas e Computação*, *Estrutura de Dados*, *Banco de Dados*, *Linguagens de Consulta*, *SQL*, etc.).

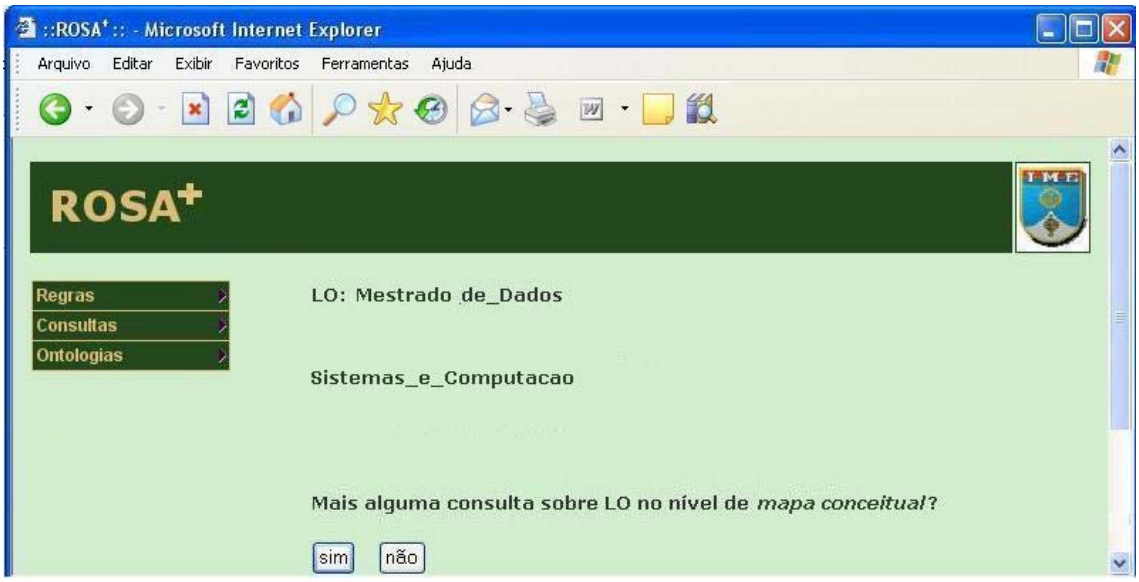

FIG. 6.13 Resultado da consulta sobre o LO Schema *Mestrado.*

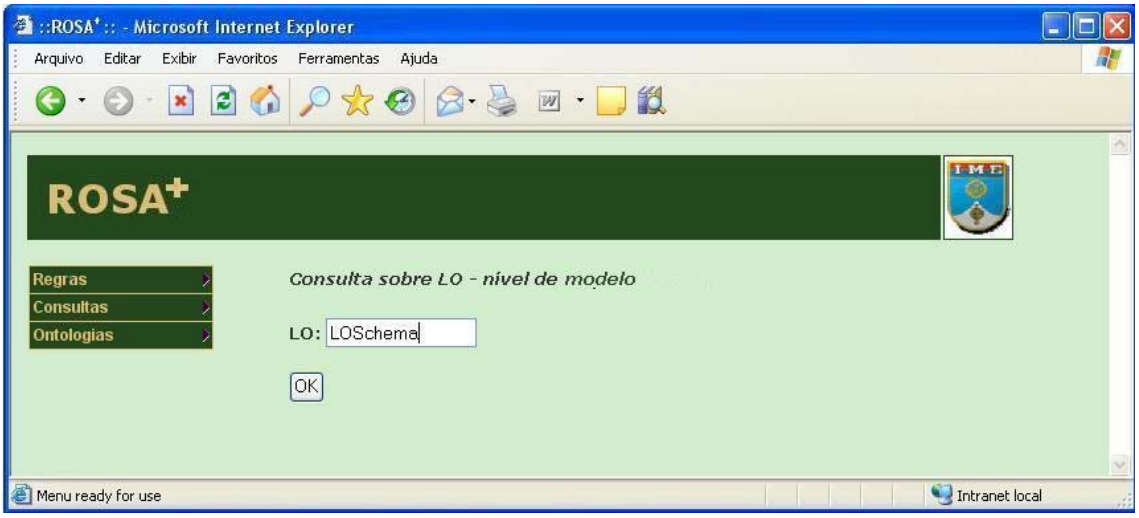

FIG. 6.14 Seleção de Tipo de LO para consulta no nível de modelo ROSA^{+}.

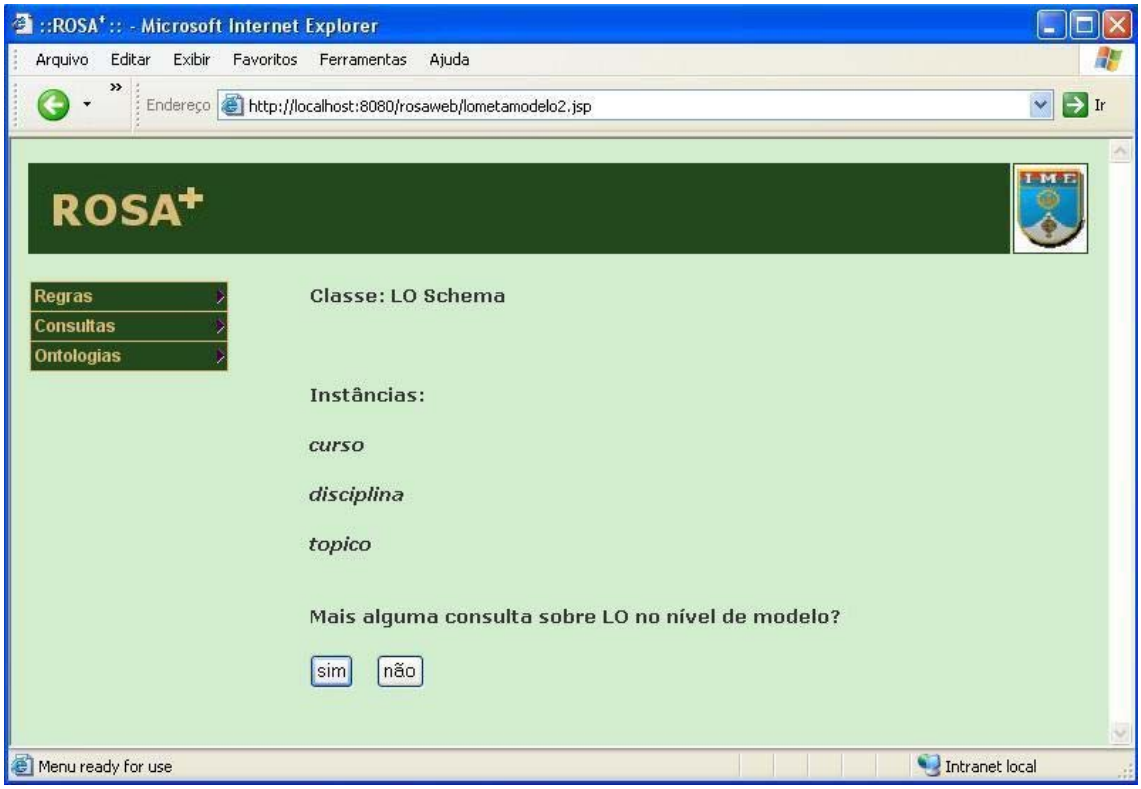

FIG. 6.15 Resultado da consulta sobre *LO Schema.*

4) Consulta sobre relacionamento no nível de mapa conceitual: esta consulta traz como resultado todas as triplas envolvendo um relacionamento selecionado. Todos os relacionamentos utilizados na Ontologia Mapa Conceitual/Domínio são disponibilizados para seleção através da classe *OWLProcessor*. Selecionado um relacionamento, a própria *OWLProcessor* retorna como resultado todas as triplas LO Lógico/Relacionamento/LO Lógico da Ontologia que tiverem esse relacionamento..

Assim, uma consulta sobre o relacionamento *fundamenta* (figura 6.16) traria como resultados (figura 6.17):

- Algoritmos *fundamenta* Linguagens de Consulta;
- Álgebra Relacional *fundamenta* SQL;
- Calculo Relacional *fundamenta* SQL;
- Controle de Concorrência *fundamenta* Recuperação de BDs (relação obtida por inferência, já que o relacionamento *é base para*, que associa estes dois LOs, é equivalente a *fundamenta*).

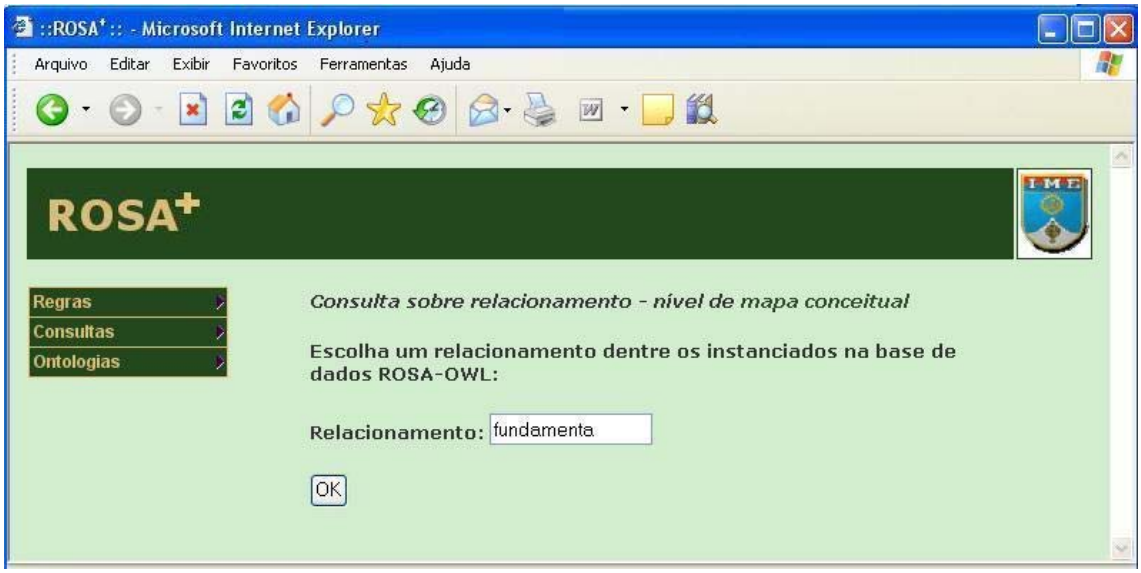

FIG. 6.16 Seleção de relacionamento para consulta no nível de mapa conceitual.

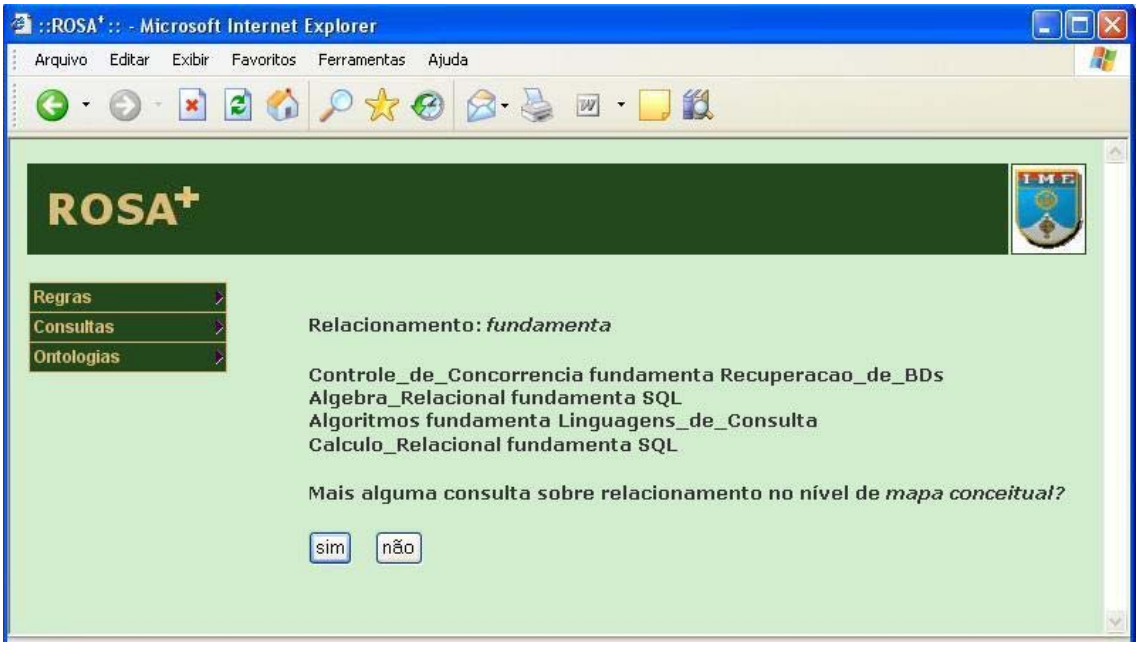

FIG. 6.17 Resultado da consulta sobre *fundamenta*.

5) Consulta sobre relacionamento no nível de mapa do domínio: esta consulta nada mais é do que uma variante da consulta anterior. Entretanto, ela traz como resultado uma tripla do tipo LO Schema/Relacionamento/LO Schema. Para isso, para cada ocorrência de um relacionamento associando dois LOs Lógicos entre as instâncias da Ontologia Mapa Conceitual/Mapa Domínio, a classe *OWLProcessor* busca o tipo do LO Lógico na mesma ontologia, ou seja, a classe à qual a instância pertence. Assim, se na consulta no nível de instância tem-se como resultado uma ocorrência de "Algoritmos

fundamenta Linguagens de Consulta", nesta o resultado seria "Disciplina *fundamenta* Tópico".

A consulta sobre o relacionamento *fundamenta* (figura 6.18) traria como resultados (figura 6.19):

- Disciplina *fundamenta* Tópico;
- Tópico *fundamenta* Tópico.

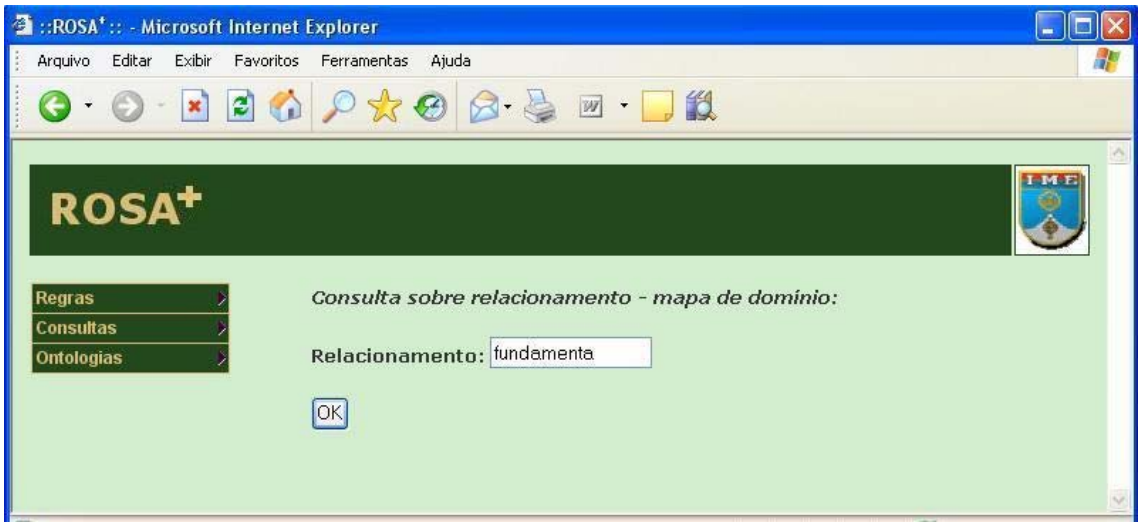

FIG. 6.18 Seleção de relacionamento para consulta no nível de mapa do domínio.

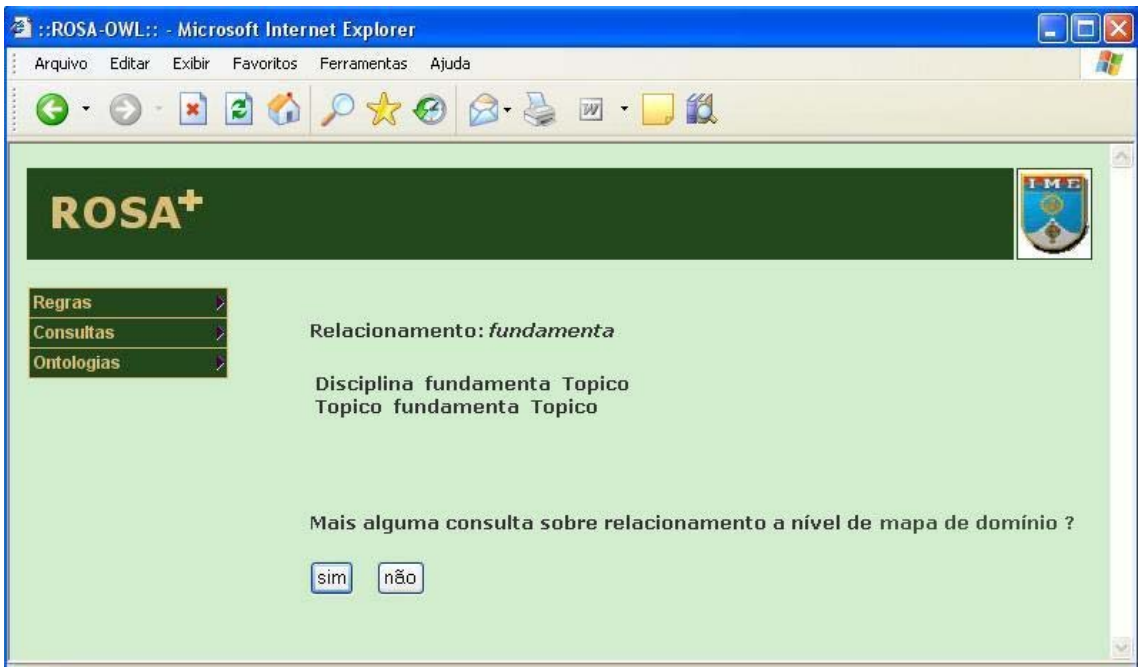

FIG. 6.19 Resultado da consulta sobre *fundamenta*.

6) Consulta sobre relacionamento no nível de Modelo ROSA$^+$: ao selecionar esta modalidade de consulta, tem-se a opção de selecionar uma consulta sobre relacionamentos ou tipos de relacionamento.

Selecionado um relacionamento (ou tipo de relacionamento), são listadas informações obtidas por instanciação (através da classe *OWLProcessor*) ou inferência (pela classe *Reasoner*) à Ontologia Mapa Domínio/Modelo ROSA$^+$: o tipo ao qual pertence (para o caso de relacionamento), as propriedades que possui, a quais relacionamentos ele é equivalente e inverso.

O resultado da consulta sobre o relacionamento *compreende* (figura 6.20) é o seguinte (figura 6.21):

- Tipo: agregação;
- Propriedades: transitiva;
- Equivalente a: *compreende, tem, inclui, enlaça, abarca, abrange, possui, envolve, ehFormadoPor, ehCompostoPor*;
- Inverso a: *ehParteDe, ehCompreendidoPor, ehAbrangidoPor, ehPossuidoPor, ehEnvolvidoPor, constitui, compõe, ehComponenteDe, forma, ehMembroDe, ehTidoPor, ehEnlaçadoPor, ehAbarcadoPor*.

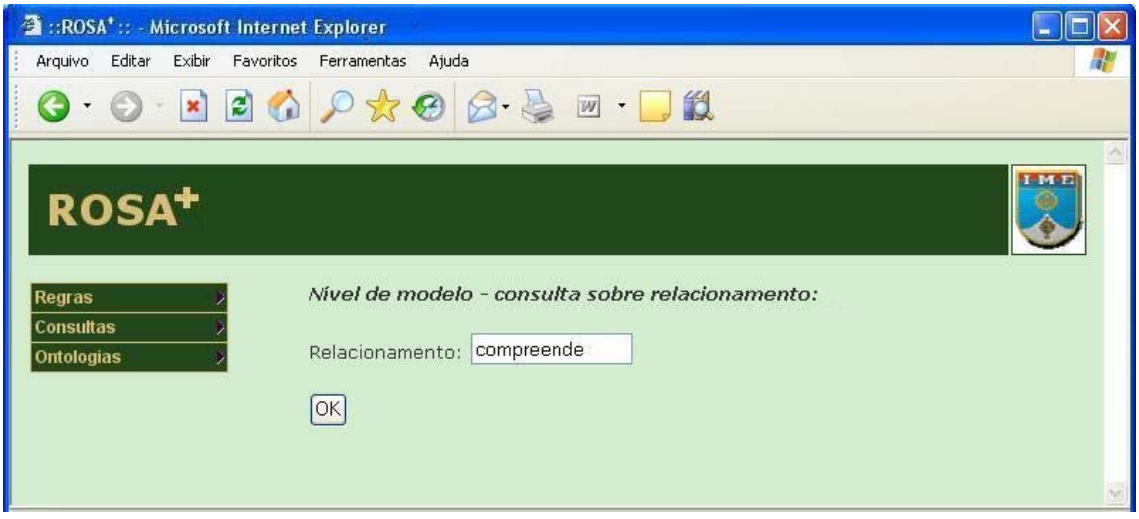

FIG. 6.20 Seleção de relacionamento para consulta no nível de modelo ROSA$^+$.

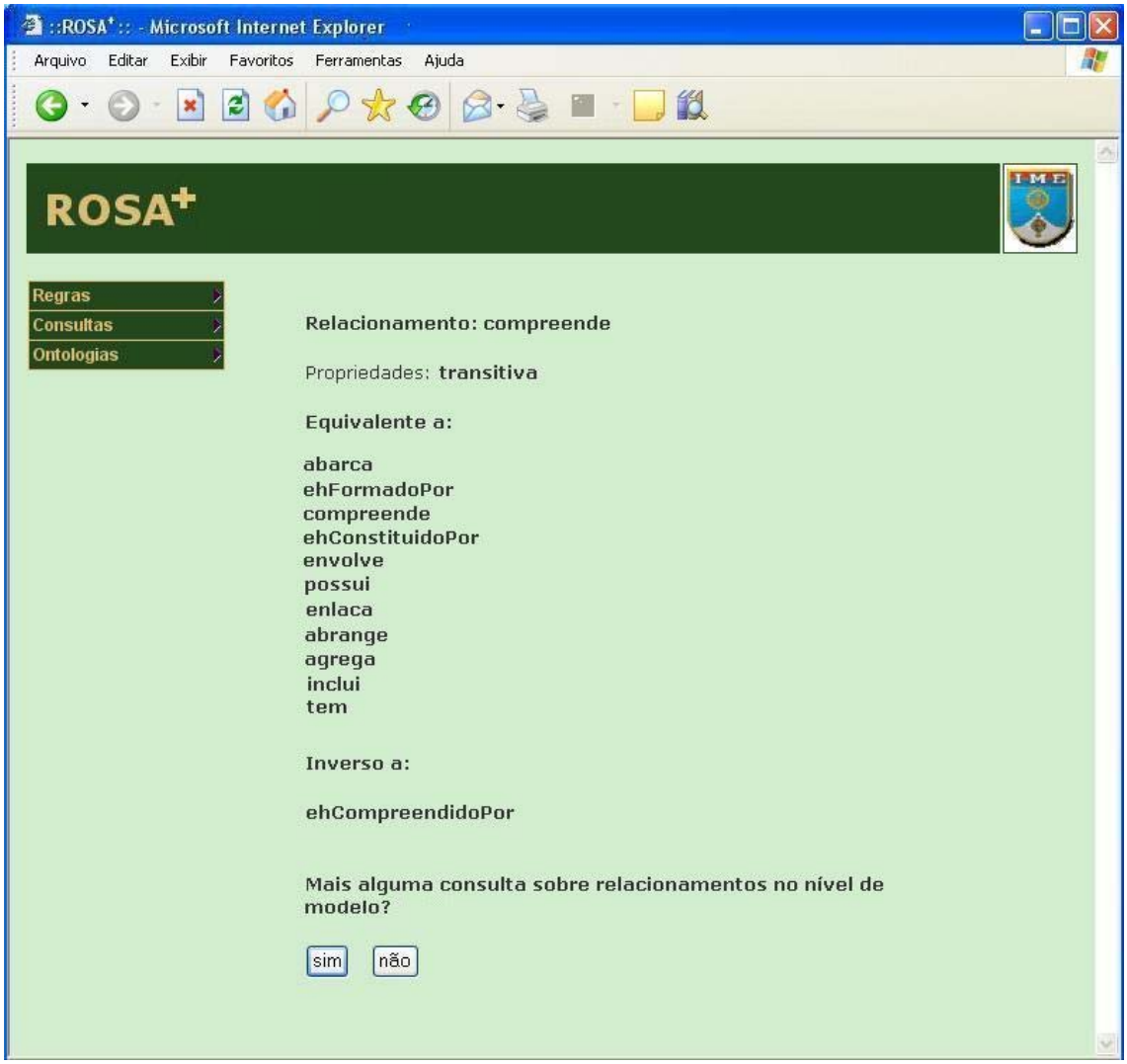

FIG. 6.21 Resultado da consulta sobre *compreende.*

7) Consulta sobre regra no nível de mapa conceitual: após disponibilizar as regras já inseridas nas ontologias, e este selecionar a opção desejada, esta consulta, utilizando a classe *Reasoner*, obtém por inferência sobre a Ontologia Mapa Conceitual/Mapa Domínio todas os relacionamentos entre LOs obtidos a partir de tal regra.

Uma consulta sobre a regra *ehPreRequisito* (figura 6.22), por exemplo, traria como resultado que "Estrutura de Dados *ehPreRequisitoDe* Banco de Dados" (figura 6.23).

FIG. 6.22 Seleção de regra para consulta no nível de mapa conceitual.

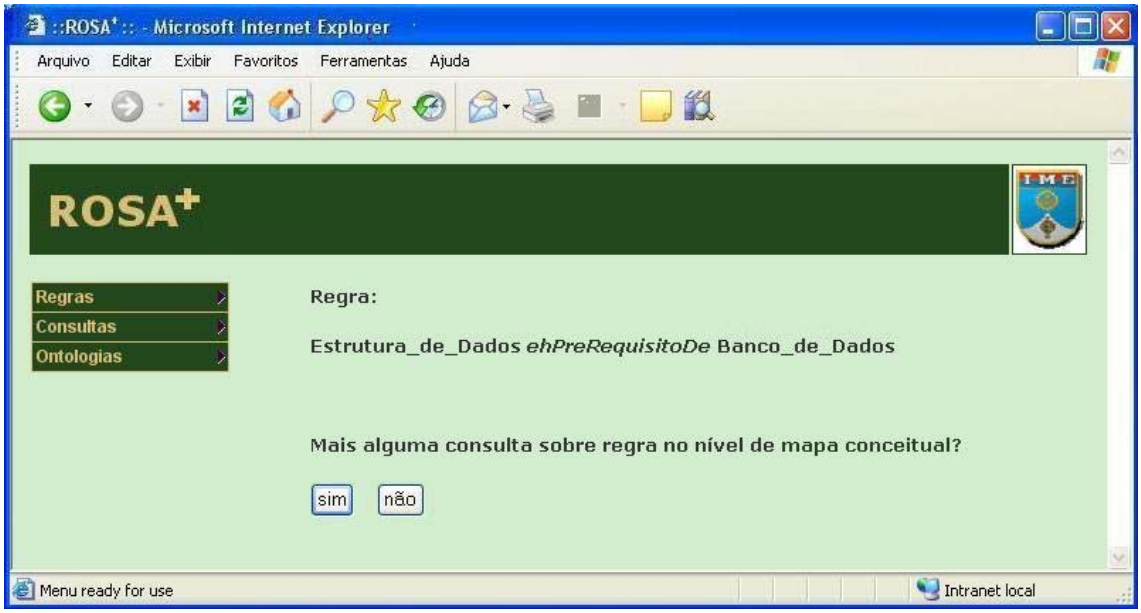

FIG. 6.23 Resultado da consulta sobre *ehPreRequisitoDe.*

8) Consulta sobre regra no nível de mapa de domínio: esta consulta é semelhante à consulta sobre relacionamento no nível de domínio, inclusive com uma resposta no mesmo formato, trazendo como retorno os LOs Schema que são associados por determinado relacionamento. A diferença é que, nesta consulta, o relacionamento entre os LOs sobre o qual a consulta é realizada não está instanciado na Ontologia Mapa Conceitual/Mapa Domínio em OWL, e sim em SWRL. O resultado desta consulta é obtido sem inferência, bastando a classe *OWLProcessor* retornar o LO Schema associado na regra às literais que constituem a cabeça da regra.

Assim, uma consulta sobre a regra *ehPreRequisitoDe* (figura 6.243) teria como resultado que "Disciplina *ehPreRequisitoDe* Disciplina" (figura 6.25).

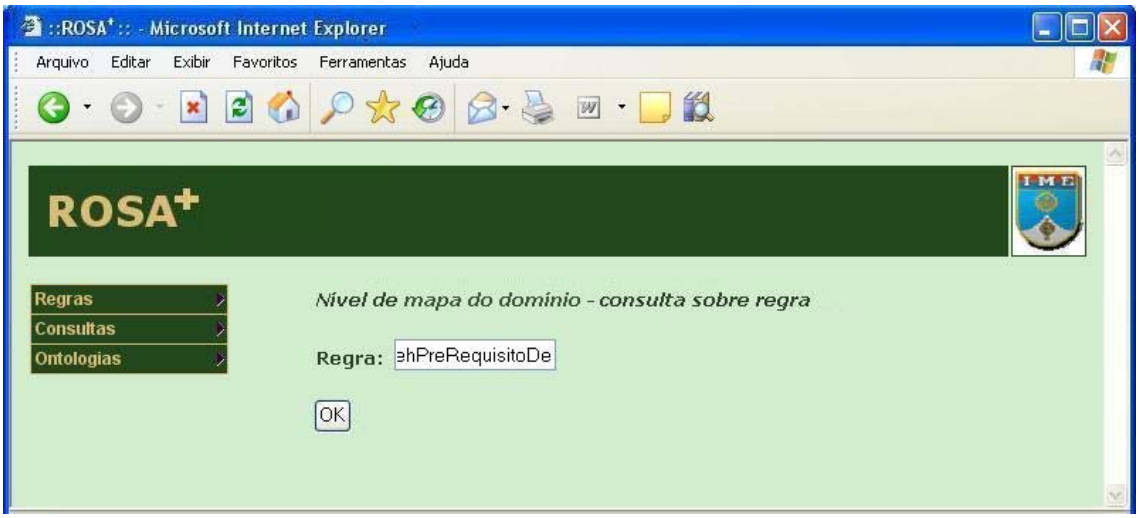

FIG. 6.24 Seleção de regra para consulta no nível de mapa de domínio.

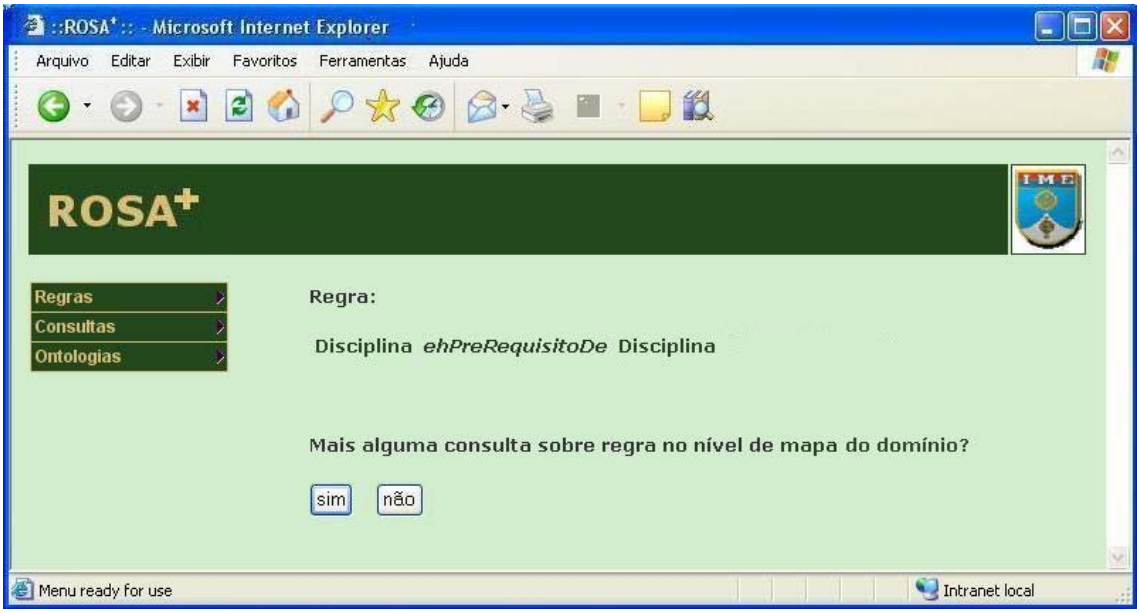

FIG. 6.25 Resultado da consulta sobre *ehPreRequisitoDe*.

9) Consulta sobre regra no nível de modelo ROSA$^+$: o objetivo desta consulta é listar, a partir de uma regra selecionada, as associações e descrições que compõem sua cabeça e corpo. Esta representação se da no formato "conseqüente ← antecedente" que vem sendo adotado ao longo deste trabalho.

Desta forma, a consulta sobre a regra *ehPreRequisito* (figura 6.26) teria como resultado "*ehPreRequisitoDe* (a,b) ← Disciplina (a), Disciplina (b), Tópico (x), Tópico (y), *compreende* (a,x), *compreende* (b,y), *fundamenta* (x,y)" (figura 6.27).

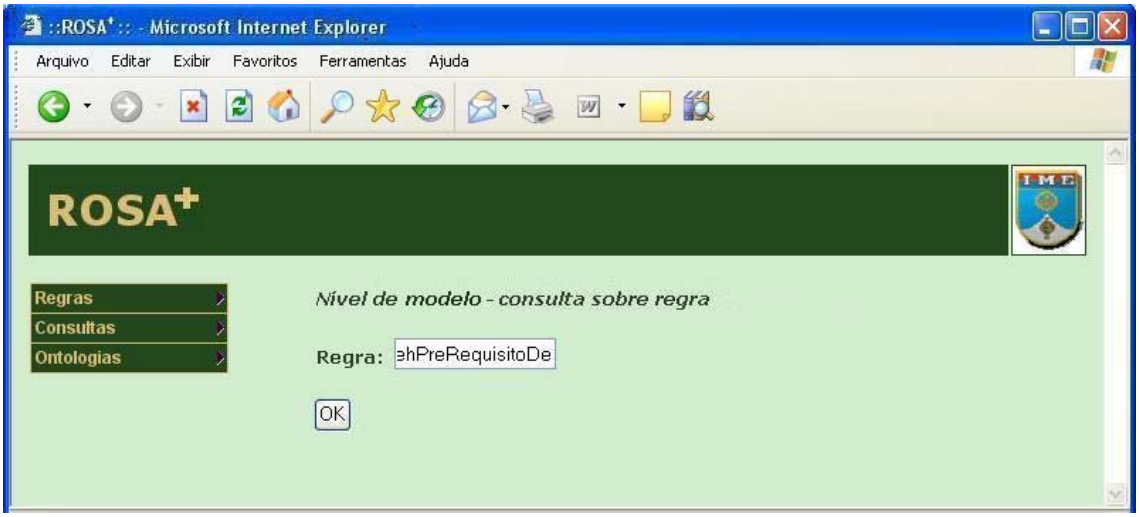

FIG. 6.26 Seleção de regra para consulta no nível de modelo.

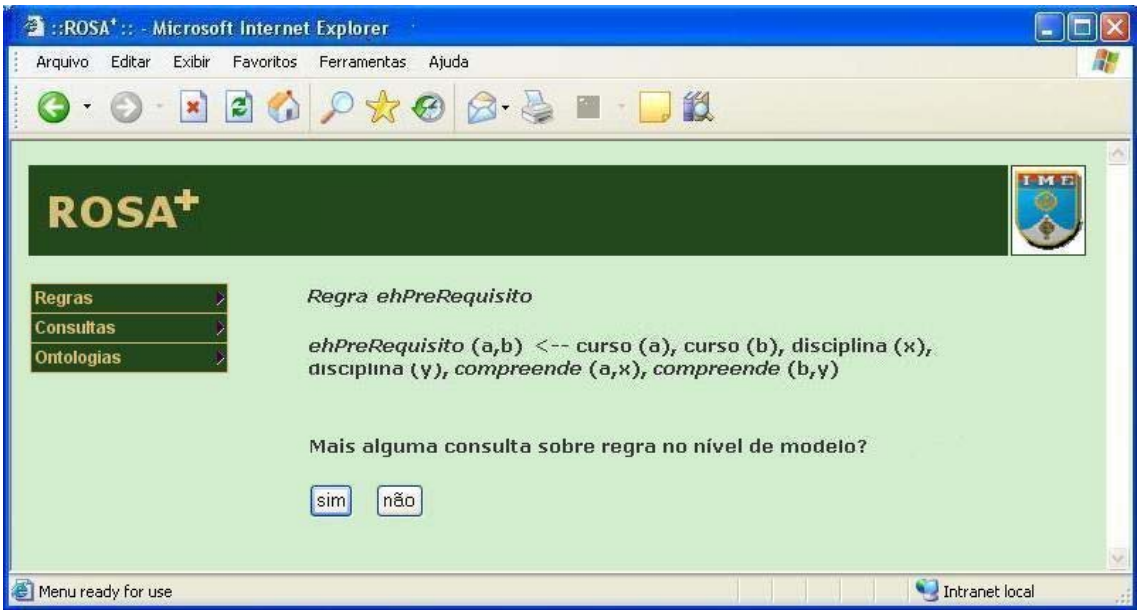

FIG. 6.27 Resultado da consulta sobre *ehPreRequisitoDe*.

6.5 TRABALHOS RELACIONADOS

Repositório é uma coleção centralizada de registros de metadados usados para descrever fontes educacionais que podem ou não estar distribuídas na Web, a exemplo dos repositórios Merlot[22], Edna[23], Dlese[24], e tantos outros. No entanto, tais repositórios armazenam e recuperam LOs por palavras chaves ou assunto, sem no entanto proverem mecanismos que permitam buscas contextualizadas ou dotadas de alguma semântica. ROSA é um sistema de repositório de LOs com acesso semântico, posicionando-se na mesma categoria que repositórios de LOs mais sofisticados, tais como CAREO [CAREO, 2002], ALOHA [ALOHA, 2002], Edutella [NEJDL et al., 2002] e WSE (Web Semântica Educativa) [GONÇALVES e CARRAPATOSO, 2006].

CAREO e ALOHA são sistemas que trabalham de maneira integrada: enquanto o primeiro auxilia o usuário a criar e gerenciar LOs, o outro permite o compartilhamento dos mesmos, funcionando como um *middleware* entre a aplicação CAREO e o usuário. No entanto, estes sistemas não provêm nenhuma facilidade dedutiva ou possibilidade de trabalhar com regras.

Edutella é outro importante repositório de LOs, que provê uma infra-estrutura *peer-to-peer* baseada na arquitetura de metadados RDF. Edutella utiliza Datalog, uma linguagem de consultas não procedural, baseada em cláusulas de Horn sem símbolos de função. Assim, é possível representar regras através de assertivas em RDF.

WSE é um projeto que propõe uma arquitetura para um sistema de pesquisa a conteúdos de aprendizagem que utiliza, além de tecnologias de Web Semântica, a tecnologia de agentes, que funciona como interface entre o usuário e uma ontologia OWL. O trabalho menciona o uso da SWRL para representação de regras, embora não descreva como isso realmente é feito.

ROSA é um sistema que possui pontos em comum com o Edutella, especialmente no que se refere à expressividade semântica, com a representação dos relacionamentos através de fatos lógicos. Os dois sistemas são baseados no modelo de dados RDF e infra-estrutura para aplicações *Peer-to-Peer* (P2P) [BRITO e MOURA, 2005]. Porém, o Edutella exige o conhecimento de uma linguagem de consulta dos usuários (RDF-QEL-i), o que é totalmente dispensável no sistema ROSA, que possui a vantagem de não exigir do usuário final nenhum tipo de conhecimento lógico e de qualquer linguagem de consulta ad-hoc específica, já que o sistema dispõe de uma interface amigável. Além do

[22] www.merlot.org
[23] www.edna.edu.au
[24] www.dlese.org

módulo de consultas, o Edutella apresenta dentre suas principais funcionalidades: i) um serviço de replicação, que provê persistência e disponibilidade dos dados; ii) serviço de mapeamento, que traduz diferentes vocabulários de metadados para permitir a interoperabilidade entre *peers*; iii) serviço de mediação, responsável pela definição de visões que permitem realizar a junção de dados de diferentes fontes de metadados; iv) e um serviço de anotações, que descreve materiais armazenados em qualquer lugar da rede Edutella. Dentre esses serviços, vale destacar que o ROSA também provê a funcionalidade ii), desenvolvida no contexto do trabalho de dissertação de Brito [BRITO, 2006].

ROSA$^+$ apresenta ainda uma abordagem baseada em linguagens mais recentes de ontologias (OWL) e regras (SWRL), além de trabalhar com objetos em diferentes níveis de abstração, através de uma arquitetura em camadas, não vislumbrada em nenhum dos sistemas aqui mencionados.

6.6 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivo mostrar detalhes da implementação e do funcionamento do sistema ROSA$^+$. Baseado na arquitetura ROSA$^+$, representada na linguagem de ontologias OWL-DL e na de regras, SWRL, foi construído um protótipo do sistema com o objetivo de validar essa proposta.

Assim, foram abordados detalhes do projeto criado e das APIs do Protégé utilizadas em cada módulo do sistema. Um dos grandes desafios do presente trabalho foi investigar as funções das APIs disponíveis do Protégé e de seus *plug-ins* para criar as funções de manipulação da ontologia desejadas, e fazer o sistema ROSA$^+$ funcionar corretamente com o raciocinador e no ambiente Protégé.

Para o desenvolvimento do sistema, foram adotadas a linguagem Java, a ferramenta para desenvolvimento Eclipse e o servidor de aplicação Apache Tomcat. Como o trabalho foi desenvolvido em mais de uma máquina, diferentes versões do Eclipse e do Tomcat foram utilizadas.

Um aspecto importante a se ressaltar na especificação foi a reutilização das APIs do Protégé e de alguns de seus *plug-ins* no desenvolvimento do sistema ROSA$^+$. O

reuso incorporou ao sistema funcionalidades do Protégé, embora tenha trazido algumas dificuldades para a implementação devido à pouca documentação disponível, principalmente sobre as APIs de manipulação da SWRL.

As consultas realizadas neste estudo de caso tiveram um tempo de resposta rápido. Entretanto, as ontologias utilizadas como base nos testes eram simples. Testes com bases de dados mais volumosas podem ser realizados em trabalhos futuros.

7 CONCLUSÃO

A cada ano, cresce de maneira exponencial o número de páginas adicionadas à Web. É importante que se aumente as fontes de conhecimento disponíveis, entretanto, paga-se um preço alto pelo aumento da dificuldade em se encontrar a informação desejada. Isso ocorre devido à falta de uma contextualização por parte das informações disponíveis na Web, problema causado pelo fato dos buscadores da Web basearem suas pesquisas apenas em palavras-chave, tornando necessária a interpretação do usuário para que a informação desejada seja encontrada.

A Web Semântica surgiu justamente como uma possível solução a necessidade maior de prover significado às informações, tornando seus dados estruturados e processáveis por máquina. Esforços vêm sendo realizados no sentido de desenvolver linguagens para a Web capazes de processar abordagens voltadas a ontologias (OWL) e regras (SWRL), com o objetivo de trazer mais significado à informação, dotando este ambiente de capacidade dedutiva.

O aumento da disponibilização de conteúdo nota-se também em outras áreas. Uma delas é a de Ensino a Distância (EAD). Com o desenvolvimento de tecnologias e a popularização da Web, cresceu o volume de dados disponível para o ensino a distância (apostilas, manuais, notas de aula, tutoriais, etc.), denominados dentro deste contexto de *Learning Objects* (LOs). Assim como na Web, a busca por LOs é um ponto crítico, sendo vista com grande interesse por empresas e instituições acadêmicas.

Assim, pode-se concluir que o universo de busca de LOs é similar ao de busca de recursos na Web. Neste contexto foi desenvolvido o ROSA (*Repository of Objects with Semantic Access*), um sistema que trata do gerenciamento de LOs e de suas associações, vistos sob a influência de padrões e conceitos providos para a Web Semântica. As associações entre LOs provêem uma contextualização dos materiais didáticos armazenados no repositório ROSA, favorecendo uma busca mais rica que leva em conta as associações semânticas entre recursos.

Uma abordagem que aumentaria ainda mais o poder semântico do sistema ROSA seria dotá-lo da capacidade de processar regras e inferência. Através destas funcionalidades, seria possível obter um conhecimento não necessariamente explicitado na ontologia, mas inferido por um raciocinador de lógica descritiva sobre uma ontologia descrita nas linguagens OWL e SWRL.

Assim, esta dissertação valeu-se do desenvolvimento destas tecnologias, visando estender o ROSA para extrair conhecimento não explicitado em sua base de dados através de inferência sobre propriedades de relacionamentos e regras.

7.1 CONTRIBUIÇÕES

Este trabalho contemplou o estudo de tecnologias da Web Semântica e a implementação de técnicas para o processamento eficiente de inferências sobre ontologias.

Dessa forma, as principais contribuições desta dissertação foram:

- Estudo comparativo das linguagens voltadas à representação de ontologias utilizadas no contexto da Web Semântica, com ênfase na linguagem OWL e nas linguagens de regras SWRL, SWRL-FOL e WRL. Adicionalmente foi feito um estudo de ferramentas de manipulação de ontologias e de raciocinadores de lógica descritiva [MATTOS, 2005];
- Extensão do modelo de dados ROSA, adicionando a este a representação de propriedades de relacionamentos e regras;
- Definição e especificação da arquitetura ROSA$^+$ em quatro camadas, representando formalmente todos os seus níveis de abstração;
- Especificação e implementação de um protótipo do sistema ROSA$^+$, permitindo a realização de inferências sobre conhecimento representado através de propriedades de relacionamentos e regras;
- Base de dados em OWL disponível para uso por outras aplicações.

Dessa maneira, o trabalho pode ser dividido sucintamente em três etapas: extensão do modelo de dados ROSA, definição da arquitetura ROSA$^+$ e implementação do sistema para validar o modelo e a arquitetura propostos.

Com a extensão do modelo ROSA, tornou-se possível a sua representação pelo modelo de dados OWL que, adicionado às regras SWRL, aumentou consideravelmente sua expressividade semântica.

A representação da arquitetura ROSA$^+$ seguindo a abordagem de camadas tornou possível realizar consultas em seus variados níveis de abstração (mapa conceitual, mapa de domínio e Modelo ROSA$^+$). Além disso, trouxe uma maior flexibilidade quanto à representação do esquema, facilitando alterações no mesmo.

A fase de implementação permitiu, portanto, a validação das idéias propostas pelo modelo e arquitetura ROSA$^+$. Na etapa de implementação do protótipo foi utilizada a ferramenta Protégé, um editor de ontologias desenvolvido em software livre, juntamente com o raciocinador de lógica descritiva Bossam, que se integra de maneira harmoniosa tanto ao Protégé quanto ao processamento da linguagem SWRL .

7.2 TRABALHOS FUTUROS

O sistema ROSA$^+$ foi desenvolvido visando a validação da extensão do modelo de dados e da arquitetura propostos neste trabalho. Assim, foram desenvolvidos apenas os módulos de inserção de regras e de consultas. Entretanto, ele não engloba todas as funcionalidades que o sistema ROSA apresenta para se consultar e manipular LOs. Sugere-se a migração de todo o sistema ROSA para a abordagem de ontologias e regras.

Outro trabalho futuro poderia estender a álgebra ROSA para que esta incorporasse a representação de regras. Assim, seria possível criar planos de execução de consultas incorporando aos planos já existentes as operações relacionadas à execução de regras. Desta maneira, seria possível melhorar os aspectos relativos à otimização de consultas, já que na versão ROSA$^+$ a interpretação de regras fica a cargo do raciocinador, não sendo possível identificar como essas são realizadas e nem tão pouco intervenir na sua forma de processamento.

Outra possibilidade de trabalho futuro consiste em se explorar mais regras dentro do domínio de EAD, justificando melhor o uso do modelo proposto. Uma possibilidade seria realizar um trabalho interdisciplinar com o apoio de um especialista de forma a ampliar o número de regras nessa área.

O fato do sistema ROSA^{+} trabalhar com duas ontologias simultaneamente abre também possibilidades de se estudar com mais detalhes a capacidade de interoperar ontologias através de um módulo do sistema que permita gerenciá-las nos seus vários níveis.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALOHA Advanced Learning Object Hub Application **The Broadcast Enabled Lifelong Learning Environment - BELLE Project.** 2002. Disponível: http://aloha.netera.ca/, [capturado em 12 abr. 2006].

ARPREZ J.C., CORCHO Oscar, FERNÁNDEZ-LOPEZ, Mariano, GÓMEZ-PÉREZ Asunción. **A Scalable Workbench for Ontological Engineering.** In: Proceedings of the First International Conferenceon Knowledge Capture (K-CAP), 2001, Victoria, B.C., Canada.

BAADER, Frank, WERNER, Nutt. **Basic Description Logics.** In: The Description Logic Handbook. Editado por BAADER, Franz, CALVANESE, Diego, MCGUINNESS, Deborah L., NARDI, Daniele, PATEL-SCHNEIDER, Peter F. 2003, Cambridge University Press, New York, NY, USA.

BARKMEYER, Ed. **Rules on the Web –** Why?. Keynote in: Reasoning on the Web (RoW2006), 15th. International World Wide Web Conference (WWW2006). 2006, Edimburg, Sweden.

BERNERS-LEE, Tim, HENDLER, James, LASSILA, Ora. **The Semantic Web**. Scientific American, 284(5):34-43. 2001.

BERNERS-LEE, Tim. **Web for Real People**. Keynote, 2005. Disponível: http://www.w3.org/2005/Talks/0511-keynote-tbl/ [capturado em 10 out. 2005].

BRITO, Gabriel A. D. D. **Integração de Objetos de Aprendizagem no Sistema ROSA-P2P.** 2005. Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, 2005.

BRITO, Gabriel A. D. D., MOURA, Ana M. C. **ROSA - P2P: a peer-to-peer system for Learning Objects integration on the web.** Proceedings of the 11th Brazilian Symposium on Multimedia and the web**,** 2005. Poços de Caldas, Minas Gerais, Brasil. p.1 - 9

CAREO Campus Alberta Repository of Educational Objects. 2002. Disponível: http://www.careo.org/index.html [capturado em 12 abr. 2006].

CORCHO, Oscar, FERNÁNDEZ-LOPEZ, Mariano, GÓMEZ-PÉREZ, Asunción. **Methodologies, Tools and Languages for Building Ontologies. Where is Their Meeting Point?** Data&Knowledge Engineeing 46, 2003, 41-64.

CONNOLLY, Dan, VAN HARMELEN, Frank, HORROCKS, Ian, MCGUINNESS, Deborah L., PATEL-SCHNEIDER, Peter F. STEIN, Lynn Andrea. **DAML+OIL Reference Description.** W3C Note, 2001. Disponível: http://www.w3.org/TR/daml+oil-reference [capturado em 01 fev. 2006].

COUTINHO, F. **Processamento de Consultas sobre o Modelo de Dados ROSA.** 2004. Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, 2004.

DAVIES, John, FENSEL, Dieter, VAN HARMELEN, Frank. **Towards the Semantic Web: Ontology-Driven Knowledge Management.** West Sussex, England : John Wiley & Sons, 2003. pp 11-31. ISBN 0470848677.

DE BRUIJIN, Jos, ANGELE, Jürgen, BOLEY, Harold, FENSEL, Dieter, HITZLER, Pascal, KIFER, Michael, KRUMMENACHER, Reto, LAUSEN, Holger, POLLERES, Axel, STUDER , Rudi. **Web Rule Language (WRL)**, W3C Member Submission, 2005. Disponível: http://www.w3.org/Submission/2005/SUBM-WRL-20050909/ [capturado em 01 abr. 2006].

DEAN, Mike, SCHREIBER, Guus, BECHHOFER, Sean, VAN HARMELEN, Frank., HENDLER, Jim, HORROCKS, Ian, MCGUINNESS, Deborah L., PATEL-SCHNEIDER, Peter F., e ANDREA Lynn. **OWL Web Ontology Language Reference**, W3C Recommendation, 2004. Disponível: http://www.w3.org/TR/2004/REC-owl-ref-20040210/ [capturado em 02 abr. 2006].

DING, Ying, FOO, Schubert. **Ontology Research and Development Part 2 - A Review of Ontology Mapping and Evolving.** 2002, *Journal of Information Science*, 28(5):375–388.

DÜRSTELER, Juan C. **Conceptual Maps.** Inf@Vis! The digital magazine of InfoVis.net, n. 141, 2004. Disponível: http://www.infovis.net/printMag.php?num=141&lang=1 [capturado em 05 out. 2005].

FACT ++. **Fast Classification of Terminologies.** Disponível: http://owl.man.ac.uk/factplusplus/ [capturado em 15 out. 2005].

FERNANDEZ, Adriana P. **Representação e Acesso a Objetos de Aprendizagem no Sistema ROSA:** Uma abordagem em Topic Maps. 2004. Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, 2004.

FORGY, C.L. **Rete: a fast algorithm for the many pattern/many object pattern match problem.** 1982. Artificial Intelligence 19, 17-37.

FRANCINCAR, Flavius, HOUBEN Geert-Jan, VDOVJAK Richard, BARNA, Peter. **RAL: an Algebra for Querying RDF.** WISE. Singapore : IEEE Computer Society, 2002. ISBN 0-7695-1766-8.

FRIESEN, Norm. **What are Educational Objects?** Interactive Learning Environments, Vol. 9, No. 3, 2001.

GARSHOL, Lars M., MOORE, Graham, **ISO 13250-2:** Topic Maps - Data Model. [online] 2003. Disponível: http://www.isotopicmaps.org/sam/ [capturado em 14 ago. 2005].

GERBÉ, Olivier, KERHERVÉ, Brigitte. **Modeling and Metamodeling Requirements for Knowledge Management**, Proc. of OOPSLA'98 Workshops, 1998, Vancouver.

GOLBREICH, Christine. **Web rules for Health Care and Life Sciences: use cases and requirements.** In: Reasoning on the Web (RoW2006), 15th. International World Wide Web Conference (WWW2006). 2006, Edimburg, Sweden.

GOMES, Hagar E. **Manual de Elaboração de Tesauros Monolingüísticos.** Brasília : Programa Nacional de Bibliotecas de Instituições de Ensino Superior,1990.

GONÇALVES, Vitor e CARRAPATOSO Eurico. **Um Sistema de Conteúdos de Aprendizagem para a Web Semântica.** XML: Aplicações e Tecnologias Associadas XATA2006, 2006, ESTG, Portalegre, Portugal, fevereiro.

HAARSLEV, Volker, MÖLLER, Ralf. **Racer:** A Core Inference Engine for the Semantic Web. In: Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA. 2003. p 27–36.

HANDRICK, Francisco T. **ROSAI:** Uma Extensão do Modelo ROSA para Tratamento de Inferência. 2005. pp. 95. Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, 2005.

HORI, Masahiro, EUZENAT, Jérôme, PATEL-SCHNEIDER, Peter F. **OWL Web Ontology Language XML Presentation Syntax.** W3C Note, 2003. Disponível: http://www.w3.org/TR/owl-xmlsyntax/ [capturado em 12 jan 2006].

HORROCKS, Ian, SATTLER, Ulrike, TOBIES, Stephan. **Practical Reasoning for Expressive Description Logics.** Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), Tbilsi, Georgia.

HORROCKS, Ian. **Ontology Reasoning:** Why and How. Keywords in: Reasoning on the Web (RoW2006), 15th. International World Wide Web Conference (WWW2006). 2006, Edimburg, Sweden.

HORROCKS, Ian, PARSIA, Ian, PATEL-SCHNEIDER, Peter F., HENDLER, James. **Semantic web architecture: Stack or two towers?** In: Francois Fages and Sylvain Soliman, editors, *Principles and Practice of Semantic Web Reasoning (PPSWR 2005)*, number 3703 in LNCS, pages 37-41. SV, 2005.

HORROCKS, Ian, PATEL-SCHNEIDER, Peter F., BOLEY, Harold, TABET, Said, GROSOF, Benjamin, DEAN, Mike. **SWRL:** A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 2004. Disponível:

http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/ [capturado em 01 abr. 2006].

IEEE Learning Technology Standards Committee. **Draft Standard for Learning Object Metadata.** Final Draft [online] 2002. Disponível: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf [capturado em 21 jan. 2004].

IMS Global Learning Consortium, Inc. **IMS Learning Resource Meta-data Specification.** [online] 2004. Disponível: http://www.imsglobal.org/metadata/index.cfm [capturado em 16 out. 2005].

JANG, Minsu. **Getting Started with Bossam Rule Engine**. [online] 2005. Disponível: http://mknows.etri.re.kr/bossam-content/gettingstarted.html [capturado 16 mar. 2006].

KIFER, Michael, DE BRUIJIN, Jos, BOLEY, Harold, FENSEL, Dieter. **A Realistic Architecture for the Semantic Web.** 2005. RULEML. Galway, Ireland, 2005. pp. 17-29.

LIPKIS, Thomas A. **A KL-ONE classifier.** In: Schomolze and Brachman, 1982, pag 128-145. BBN Research Report 4842, Bolt Beranek and Newman Inc., June.

LLOYD, J. W. **Foundations of logic programming** (second, extended edition). 1987, Springer series in symbolic computation. Springer-Verlag, New York.

LU, Shiyong, DONG, Ming, FOTOUHI, Farshad. **The Semantic Web: opportunities and challenges for next-generation Web applications**. 2002. Information Research, Vol. 7 No. 4, July.

MÄDCHE, Alexander, SCHNURR, Hans-Peter, STAAB, Steffen, STUDER, Rudi. **Representation Language-Neutral Modeling of Ontologies.** Workshop Modellieung, 2000. St. Goar, Alemanha.

MATTOS, Diogo O. P. Moura, Ana M. C. Cavalcanti, M. C. **Linguagens e Ferramentas para Criação e Manipulação de Ontologias.** Relatório Técnico n. 116/SE8/DEZ2005, 2005, IME/RJ.

MOF Meta Object Facility Specification, Version 1.4. 2002. Disponível: http://www.omg.org/docs/formal/02-04-03.pdf [capturado em 18 mar 2006].

MOTIK, Boris, SATTLER, Ulrike, STUDER, Rudi. **Query Answering for OWL-DL with Rules.** Proc. of the 3rd Int. Semantic Web Conf. (ISWC 2004), volume 3298 of Lecture Notes in Computer Science, 2004, pages 549–563, Hiroshima, Japan. Springer.

MOURA, Ana M. C., PORTO, Fábio, FERNANDES, Abílio, FERNANDEZ, Adriana P., COUTINHO, Fábio. **ROSA/e-Learning: Repositório de Objetos com Acesso Semântico para e-Learning.** 2003. Relatório técnico. Instituto Militar de Engenharia, 2003.

NARDI, Daniele, BRACHMAN, Ronald J. **An Introduction to Description Logics.** In: The Description Logic Handbook. Editado por BAADER, Franz, CALVANESE, Diego, MCGUINNESS, Deborah L., NARDI, Daniele, PATEL-SCHNEIDER, Peter F. 2003, Cambridge University Press, New York, NY, USA.

NEJDL, Wolfgang, WOLF, Boris, QU, Changtao, DECKER, Stefan, SINTEK, Michael, NAEVE, Ambjörn, NILSSON, Mikael, PALMER, Matthias, RISCH, Tore. **EDUTELLA: a P2P networking infrastructure based on RDF.** Proceedings of the 11th international conference on World Wide Web, 2002, Honolulu, Hawaii, USA, maio.

NOY, Natalya Fridman, FERGENSON, Ray. W., MUSEN, Mark A. **The Knowledge Model of Protégé-Combining Interoperability and Flexibility.** In 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW), 2000. Springer-Verlag, Juan-les-Pins, França, 17-32.

OMG. **Common Facilities RFP-5: Meta-Object Facility.** OMG TC document cf/96-05-02, 1996. Object Management Group, Junho.

PARSIA, Bijan, SIRIN, Evrin. **Pellet:** An OWL DL Reasoner. In: Proc. ISWC-2004, 2004. Disponível: http://www.mindswap.org/2003/pellet/index.shtml [capturado em 10 out. 2005].

PATEL-SCHNEIDER, Peter F. **A proposal for a SWRL Extension towards First-Order Logic.** W3C Member Submission, 2005. Disponível: http://www.w3.org/Submission/2005/SUBM-SWRL-FOL-20050411/ [capturado em 01 abr. 2006].

PORTO, Fábio, MOURA, Ana M. C., FERNANDES, Abilio, FERNANDEZ, Adriana, COUTINHO, Fabio, COUTINHO, L. e CAMPOS, G. **ROSA:** A Data Model and Query Language for e-Learning Objects I PGL de Pesquisa em Banco de Dados para e-Learning, 2003, PUC-RIO.

PORTO, Fábio, MOURA, Ana M. C., COUTINHO, Fábio. **ROSA:** a Repository of Objects with Semantic Access for e-Learning. IDEAS. Lisboa : IEEE Computer Society, 2004. pp. 486-488. ISBN 0-7695-2168-1.

QUILLIAN, M. Ross. **Word concepts: A theory and simulation of some basic semantic capabilities.** 1967, Behavioural Science 12.

RDF Resource Description Framework Model and Syntax Specification [online]. 1999. W3C Recommendation. Disponível: http://www.w3.org/TR/PR-rdf-syntax/ [capturado em 14 out. 2005].

RDFS RDF Vocabulary Description Language 1.0: RDF Schema [online]. 2004. W3C Recommendation. Disponível: http://www.w3.org/TR/PR-rdf-schema/ [capturado em 15 out. 2005].

SCHMIDT-SCHAUß, Manfred, SMOLKA, Gert. **Attributive concept descriptions with complements.** Artificial Intelligence, 48(1):1-26, 1991.

UML Unified Modeling Language: Superstructure. Version 2.0. 2005. Disponível: http://www.omg.org/docs/formal/05-07-04.pdf [capturado em 18 mar 2006].

XML Extensive Markup Language (XML) 1.0 (Third Edition) [online]. 2004. W3C Recommendation. Disponível: http://www.w3.org/TR/2004/REC-xml-20040204/ [capturado em 14 out. 2005].

9 APÊNDICES

9.1 ESPECIFICAÇÃO DA ARQUITETURA ROSA$^+$

CAMADA DE MAPA CONCEITUAL

Compreende(Sistemas e Computação,Estrutura de Dados)
Compreende(Sistemas e Computação,Redes)
Compreende(Sistemas e Computação,Banco de Dados)
Compreende(Banco de Dados,Álgebra Relacional)
Compreende(Banco de Dados,Cálculo Relacional)
Fundamenta(Álgebra Relacional,SQL)
Fundamenta(Cálculo Relacional,SQL)

ehPreRequisitoDe(a,b) ← disciplina(a), disciplina(b), tópico(c), tópico(d), compreende(a,c), compreende(b,d), fundamenta(c,d)

possuiEmEmenta(a,b) ← disciplina(a), tópico(c), compreende(a,c)

possuiEmPrograma(e,a) ← curso(e),disciplina(a), compreende(e,a)

CAMADA DE DOMÌNIO

Curso(Sistemas e Computação)
Disciplina(Estrutura de Dados)
Disciplina(Redes)
Disciplina(Banco de Dados)
Tópico(SQL)
Tópico(Álgebra Relacional)
Tópico(Cálculo Relacional)

Compreende(Curso,Disciplina)
DependeDe(Disciplina,Disciplina)
DependeDe(Curso,Curso)
Compreende(Tópico,Tópico)
Fundamenta(Tópico,Tópico)
IsA(Doutorado,Curso)
IsA(Mestrado,Curso)
IsA(EAD,Curso)

CAMADA DO MODELO ROSA$^+$

LOSchema(Curso)
LOSchema (Disciplina)
LOSchema (Tópico)

LOLogico (Sistemas e Computação)
LOLogico (Estrutura de Dados)
LOLogico (Redes)
LOLogico (Banco de Dados)
LOLogico (SQL)
LOLogico (Álgebra Relacional)
LOLogico (Cálculo Relacional)

Relacionamento(compreende)
Relacionamento(tem)
Relacionamento(inclui)
Relacionamento(enlaça)
Relacionamento(abarca)
Relacionamento(abrange)
Relacionamento(possui)
Relacionamento(envolve)
Relacionamento(ehFormadoPor)
Relacionamento(ehCompostoPor)

Relacionamento(fundamenta)
Relacionamento(ehBasePara)
Relacionamento(ehCondicaoPara)
Relacionamento(requer)

Relacionamento(gera)
Relacionamento(cria)
Relacionamento(produz)
Relacionamento(desenvolve)

Relacionamento(equivale)
Relacionamento(seAssemelhaA)
Relacionamento(seComparaA)
Relacionamento(seIgualaA)
Relacionamento(seEquiparaA)
Relacionamento(nivelaSeA)
Relacionamento(ehIgualA)

Relacionamento(influi)
Relacionamento(inspira)

Relacionamento(permite)
Relacionamento(admite)
Relacionamento(licencia)
Relacionamento(aceita)
Relacionamento(deixa)

Relacionamento(necessita)
Relacionamento(precisaDe)
Relacionamento(carece)
Relacionamento(demanda)

Relacionamento(implica)
Relacionamento(segue)
Relacionamento(aponta)
Relacionamento(origina)
Relacionamento(determina)
Relacionamento(deriva)
Relacionamento(resulta)
Relacionamento(provê)
Relacionamento(define)
Relacionamento(especifica)
Relacionamento(estabelece)

Relacionamento(ehCompreendidoPor)
Relacionamento(ehAbrangidoPor)
Relacionamento(ehPossuidoPor)
Relacionamento(ehEnvolvidoPor)
Relacionamento(constitui)
Relacionamento(compõe)
Relacionamento(ehParteDe)
Relacionamento(ehComponenteDe)
Relacionamento(forma)
Relacionamento(ehMembroDe)
Relacionamento(ehTidoPor)
Relacionamento(ehEnlacadoPor)
Relacionamento(ehAbracadoPor)

Relacionamento(ehFundamentadoPor)
Relacionamento(temComoPreRequisito)
Relacionamento(temComoBase)

Relacionamento(ehGeradoPor)
Relacionamento(ehCriadoPor)
Relacionamento(ehProduzuidoPor)
Relacionamento(ehDesenvolvidoPor)

Relacionamento(seDistingueDe)
Relacionamento(seDifereDe)
Relacionamento(seDiferenciaDe)
Relacionamento(ehDistintoDe)

Relacionamento(ehInfluenciadoPor)
Relacionamento(ehInspiradoPor)

Relacionamento(ehPermitidoPor)

Relacionamento(ehAdmitidoPor)
Relacionamento(ehLicenciadoPor)
Relacionamento(ehAceitoPor)

Relacionamento(ehNecessarioPara)
Relacionamento(ehPrecisoPara)

Relacionamento(ehImplicadoPor)
Relacionamento(ehOriginadoPor)
Relacionamento(ehDeterminadoPor)
Relacionamento(ehDerivadoPor)
Relacionamento(ehResultadoDe)
Relacionamento(ehDefinidoPor)
Relacionamento(ehEspecificadoPor)
Relacionamento(ehEstabelecidoPor)

TipoRelacionamento(agregação)
TipoRelacionamento(fundamentação)
TipoRelacionamento(criação)
TipoRelacionamento(equivalência)
TipoRelacionamento(influência)
TipoRelacionamento(permissão)
TipoRelacionamento(necessidade)
TipoRelacionamento(implicação)
TipoRelacionamento(utilização)

TipoDe(compreende, agregação)
TipoDe(fundamenta, fundamentação)
TipoDe(gera, criação)
TipoDe(equivale, equivalência)
TipoDe(influi, influência)
TipoDe (permite,permissão)
TipoDe (precisaDe,necessidade)
TipoDe (determina, implicação)
TipoDe (utiliza, utilização)

same as(compreende,abrange)
same as(compreende,é formado por)
same as(compreende,tem)
same as(compreende,inclui)
same as(compreende,enlaça)
same as(compreende,abarca)
same as(compreende,possui)
same as(compreende,envolve)
same as(compreende,é composto por)
same as(é compreendido por,é parte de)
same as(é compreendido por,é abrangido por)
same as(é compreendido por,é possuído por)
same as(é compreendido por,é envolvido por)

same as(é compreendido por,constitui)
same as(é compreendido por,compõe)
same as(é compreendido por,forma)
same as(é compreendido por,é membro de)
same as(é compreendido por,é tido por)
same as(é compreendido por,é enlaçado por)
same as(é compreendido por,é abarcado por)
inversa(compreende, é compreendido por)
transitiva(agregação)

same as(fundamenta,é base para)
same as(fundamenta,é condição para)
same as(é fundamentado por,tem como base)
inversa(fundamenta, é fundamentado por)
transitiv(fundamentação)
same as(gera,cria)
same as(gera,produz)
same as(gera,desenvolve)
same as(é gerado por,é criado por)
same as(é gerado por,é produzido por)
same as(é gerado por,é desenvolvido por)
inversa(gera,é gerado por)
same as(equivale,se equipara a)
same as(equivale,se é mesma que a)
same as(equivale,se iguala a a)
same as(se distingue de,se difere a)
same as(se distingue de,se diferencia a)
same as(se distingue de,é distinto a)
inversa(equivale,se distingue de)
simetrica(equivalência)

same as(influi,inspira)
same as(é influenciado por,é inspirado por)
inversa(influi,é influenciado por)
transitiva(influência)

same as(permite,admite)
same as(permite,licencia)
same as(permite,aceita)
same as (permite,deixa)
same as(é permitido por,é admitido por)
same as(é permitido por,é licenciado por)
same as(é permitido por,é aceito por)
inversa(permite,é permitido por)

same as(necessita,precisa de)
same as(necessita,carece)
same as(necessita,demanda)
same as(é necessário para,é preciso para)
inversa(necessita,é necessário para)

transitiva(necessidade)

same as(utiliza,usa)
same as(utiliza,usufrui)
same as(utiliza,aproveita)
same as(é utilizado por,é usufruído por)
same as(é utilizado por,é aproveitado por)
inversa(utiliza,é utilizado para)
transitiva(utilização)

same as(necessita,precisa de)
same as(necessita,carece)
same as(necessita,demanda)
same as(é necessário para,é preciso para)
inversa(necessita,é necessário para)
transitiva(necessidade)

same as(implica,aponta)
same as(implica,origina)
same as(implica,determina)
same as(implica,resulta)
same as(implica,deriva)
same as(implica,provê)
same as(implica,define)
same as(implica,especifica)
same as(implica,estabelece)
same as(é implicado por,é originado por)
same as(é implicado por,é determinado por)
same as(é implicado por,é derivado por)
same as(é implicado por,é resultado de)
same as(é implicado por,é definido por)
same as(é implicado por,é especificado por)
same as(é implicado por,é estabelecido por)
inversa(utiliza,é utilizado para)
transitiva(implicação)

Propriedade(transitividade)
Propriedade(simetria)
Propriedade(inversão)
Propriedade(equivalência)

Literal(a)
Literal(b)
Literal(c)
Literal(d)
Literal(e)

Associação(compreende(a,b))
Associação(compreende(a,c))
Associação(compreende(e,a))

Associação(compreende(b,d))
Associação(fundamenta(c,d))
Associação(ehPreRequisitoDe(a,b))
Associação(possuiEmEmenta(a,b))
Associação(possuiEmPrograma(a,b))

Descrição(curso(e))
Descrição(disciplina(a))
Descrição(disciplina(b))
Descrição(tópico(c))
Descrição(tópico(d))

Antecedente(disciplina(a) ^ disciplina(b) ^ tópico(c) ^ tópico(d) ^ compreende(a,c) ^ compreende(b,d) ^ fundamenta(c,d))
Antecedente(disciplina(a) ^ tópico(c) ^ compreende(a,c))
Antecedente(curso(e) ^ disciplina(a) ^ compreende(e,a))

Conseqüente(ehPreRequisitoDe(a,b))
Conseqüente(possuiEmEmenta(a,b))
Conseqüente(possuiEmPrograma(e,a))

Hierarquia (LO_1,LO_2) = $LO_2 \subseteq LO_1$
Sacola (LO_1,LO_2 ... LO_n) = { LO_i … LO_j }
Sequência (LO_1,LO_2 ... LO_n) = { LO_i … LO_j } Π ($i < j$)
Conjunto (LO_1,LO_2 ... LO_n) = { LO_i … LO_j } Π ($i \neq j$)

IsA(LOSchema,LOLogico)
IsA(LOLogico,LO)
IsA(LOFisico,LO)

Agrega(LOLogico, relacionamentoColeção)
Agrega(relacionamentoColeção,LO)

TipoDe(Relacionamento,TipoRelacionamento)

IsA(Sacola,Coleçao)
IsA(Hierarquia,Coleçao)
IsA(Seqüência,Coleçao)
IsA(Conjunto,Coleçao)

Tem(Propriedade,TipoRelacionamento)
Tem(Regra,RelacionamentoColeçao)

Tem(Regra,Antecedente)
Tem(Regra,Conseqüente)
Tem(Antecedente,Associação)
Tem(Antecedente,Descrição)
Tem(Conseqüente,Associação)
Tem(Conseqüente,Descrição)
Tem(Associação,Literal)

Tem(Associação,Relacionamento)
Tem(Descrição,Literal)
Tem(Descrição,LOSchema)

Conforme(RelacionamentoColeçao,Relacionamento)
Conforme(RelacionamentoColeçao,Coleção)

CAMADA DE METAMODELO

MetaRelacionamento(IsA)
MetaRelacionamento(Agrega)
MetaRelacionamento(TipoDe)
MetaRelacionamento(Tem)
MetaRelacionamento(Conforme)

MetaClasse(Relacionamento)
MetaClasse(TipoRelacionamento)
MetaClasse(LO)
MetaClasse(RelacionamentoColeção)
MetaClasse(Propriedade)
MetaClasse(Coleção)
MetaClasse(Regra)
MetaClasse(Antecedente)
MetaClasse(Conseqüente)
MetaClasse(Associação)
MetaClasse(Descrição)
MetaClasse(Literal)

isA(MetaClasse, RecursoComplexo)
isA(MetaRelacionamento, RecursoComplexo)

9.2 ONTOLOGIA MAPA CONCEITUAL/DOMÍNIO DO ESTUDO DE CASO

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:swrlbImport="http://www.daml.org/rules/proposal/swrlb.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:swrlImport="http://www.daml.org/rules/proposal/swrl.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about="">
    <owl:imports
rdf:resource="http://www.daml.org/rules/proposal/swrlb.owl"/>
    <owl:imports
rdf:resource="http://www.daml.org/rules/proposal/swrl.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="PosGraduacao">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Curso"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Disciplina">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="LO"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Graduacao">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Curso"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Curso">
    <rdfs:subClassOf rdf:resource="#LO"/>
  </owl:Class>
  <owl:Class rdf:ID="Doutorado">
    <rdfs:subClassOf rdf:resource="#PosGraduacao"/>
  </owl:Class>
  <owl:Class rdf:ID="Mestrado">
    <rdfs:subClassOf rdf:resource="#PosGraduacao"/>
  </owl:Class>
  <owl:Class rdf:ID="Topico">
    <rdfs:subClassOf rdf:resource="#LO"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="my_property"/>
  <owl:ObjectProperty rdf:ID="ehOrdenadaPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:ID="ehFundamentadaPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehEnlacadoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:ID="ehCompreendidoPor"/>
```

```
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehGeradoPor">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="gera"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#LO"/>
  <rdfs:domain rdf:resource="#LO"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="seDifereDe">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="distingue"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehComponenteDe">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="possuiEmPrograma"/>
<owl:ObjectProperty rdf:ID="aceita">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="permite"/>
  </rdfs:subPropertyOf>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="ehAceitoPor"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="resulta">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:ID="implica"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="desenvolve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#gera"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="define">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#implica"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="licencia">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="ehLicenciadoPor"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#permite"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="especifica">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#implica"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="possui">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="compreende"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="ehInfluidoPor">
  <rdfs:domain rdf:resource="#LO"/>
  <rdfs:range rdf:resource="#LO"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="influi"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="abarca">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#compreende"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehDefinidoPor">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="ehImplicadoPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="inclui">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#compreende"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehBaseDe">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#ehFundamentadaPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="usufrui">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:ID="utiliza"/>
  </rdfs:subPropertyOf>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="ehUsufruidoPor"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="abrange">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#compreende"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehMesmoQue">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:ID="equivale"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="compoe">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehAproveitadoPor">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="aproveita"/>
  </owl:inverseOf>
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:ID="ehUtilizadoPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="possuiEmEmenta">
  <owl:inverseOf rdf:resource="#possuiEmEmenta"/>
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehUsufruidoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehUtilizadoPor"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf rdf:resource="#usufrui"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehCondicaoPara">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:ID="fundamenta"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehPermitidoPor">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#permite"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#LO"/>
    <rdfs:range rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="determina">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="tem">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#compreende"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehResultadoDe">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#ehImplicadoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="produz">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#gera"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehAbrangidoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="deriva">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehParteDe">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehInfluenciadoPor">
    <rdfs:subPropertyOf rdf:resource="#ehInfluidoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehAdmitidoPor">
    <owl:inverseOf>
```

```
      <owl:ObjectProperty rdf:ID="admite"/>
    </owl:inverseOf>
    <rdfs:subPropertyOf rdf:resource="#ehPermitidoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehEnvolvidoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="enlaca">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#compreende"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehLicenciadoPor">
    <rdfs:subPropertyOf rdf:resource="#ehPermitidoPor"/>
    <owl:inverseOf rdf:resource="#licencia"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="cria">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#gera"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehPossuidoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#fundamenta">
    <rdfs:domain rdf:resource="#LO"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:about="#ehFundamentadaPor"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehImplicadoPor">
    <rdfs:domain rdf:resource="#LO"/>
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </owl:inverseOf>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehPreRequisitoDe"/>
  <owl:ObjectProperty rdf:ID="aponta">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="inspira">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#influi"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="seComparaA">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#equivale"/>
    </rdfs:subPropertyOf>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="envolve">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#compreende"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehCriadoPor">
  <rdfs:subPropertyOf rdf:resource="#ehGeradoPor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ordena">
  <rdfs:subPropertyOf rdf:resource="#fundamenta"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="prove">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#implica"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehOriginadoPor">
  <rdfs:subPropertyOf rdf:resource="#ehImplicadoPor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#equivale">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#distingue"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#LO"/>
  <rdfs:range rdf:resource="#LO"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehAgregadoPor">
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#admite">
  <owl:inverseOf rdf:resource="#ehAdmitidoPor"/>
  <rdfs:subPropertyOf>
    <owl:ObjectProperty rdf:about="#permite"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehEspecificadoPor">
  <rdfs:subPropertyOf rdf:resource="#ehImplicadoPor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehProduzidoPor">
  <rdfs:subPropertyOf rdf:resource="#ehGeradoPor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#aproveita">
  <owl:inverseOf rdf:resource="#ehAproveitadoPor"/>
  <rdfs:subPropertyOf>
    <owl:TransitiveProperty rdf:about="#utiliza"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehBasePara">
  <rdfs:subPropertyOf rdf:resource="#fundamenta"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="seIgualaA">
  <rdfs:subPropertyOf rdf:resource="#equivale"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ehDerivadoPor">
  <rdfs:subPropertyOf rdf:resource="#ehImplicadoPor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#distingue">
  <owl:inverseOf rdf:resource="#equivale"/>
```

```
    <rdfs:domain rdf:resource="#LO"/>
    <rdfs:range rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="estabelece">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#permite">
    <rdfs:domain rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#ehPermitidoPor"/>
    <rdfs:range rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehAbarcadoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehConstituidoPor">
    <rdfs:subPropertyOf>
      <owl:ObjectProperty rdf:about="#compreende"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#compreende">
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </owl:inverseOf>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehEstabelecidoPor">
    <rdfs:subPropertyOf rdf:resource="#ehImplicadoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasIngredient"/>
  <owl:ObjectProperty rdf:ID="ehUsadoPor">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehUtilizadoPor"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="usa"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="constitui">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehDeterminadoPor">
    <rdfs:subPropertyOf rdf:resource="#ehImplicadoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehDesenvolvidoPor">
    <rdfs:subPropertyOf rdf:resource="#ehGeradoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehMembroDe">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehAceitoPor">
```

```
    <rdfs:subPropertyOf rdf:resource="#ehPermitidoPor"/>
    <owl:inverseOf rdf:resource="#aceita"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="deixa">
    <rdfs:subPropertyOf rdf:resource="#permite"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehFormadoPor">
    <rdfs:subPropertyOf rdf:resource="#compreende"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="forma">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#ehCompreendidoPor"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#influi">
    <owl:inverseOf rdf:resource="#ehInfluidoPor"/>
    <rdfs:domain rdf:resource="#LO"/>
    <rdfs:range rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="origina">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#implica"/>
    </rdfs:subPropertyOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehDistintoDe">
    <rdfs:subPropertyOf rdf:resource="#distingue"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="seDiferenciaDe">
    <rdfs:subPropertyOf rdf:resource="#distingue"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#usa">
    <rdfs:subPropertyOf>
      <owl:TransitiveProperty rdf:about="#utiliza"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf rdf:resource="#ehUsadoPor"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#gera">
    <rdfs:domain rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#ehGeradoPor"/>
    <rdfs:range rdf:resource="#LO"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="dificuldade">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >facil</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:parseType="Resource">
              <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
              <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >dificil</rdf:first>
            </rdf:rest>
            <rdf:first
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >medio</rdf:first>
          </rdf:rest>
        </owl:oneOf>
```

```
      </owl:DataRange>
    </rdfs:range>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="palavraChave">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="id">
    <rdfs:domain rdf:resource="#LO"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="autor">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="titulo">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:DatatypeProperty>
  <owl:TransitiveProperty rdf:about="#implica">
    <rdfs:domain rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#ehImplicadoPor"/>
    <rdfs:range rdf:resource="#LO"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#ehFundamentadaPor">
    <rdfs:range rdf:resource="#LO"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <owl:inverseOf rdf:resource="#fundamenta"/>
    <rdfs:domain rdf:resource="#LO"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#utiliza">
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf>
      <owl:TransitiveProperty rdf:about="#ehUtilizadoPor"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#LO"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#ehCompreendidoPor">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#LO"/>
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#compreende"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:about="#ehUtilizadoPor">
    <rdfs:range rdf:resource="#LO"/>
    <rdfs:domain rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#utiliza"/>
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:ID="agrega">
    <rdfs:subPropertyOf rdf:resource="#compreende"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <Disciplina rdf:ID="Estrutura_de_Dados">
    <compreende>
      <Topico rdf:ID="Grafos">
        <ehCompreendidoPor rdf:resource="#Estrutura_de_Dados"/>
      </Topico>
    </compreende>
    <compreende>
      <Topico rdf:ID="Arvores">
        <ehCompreendidoPor rdf:resource="#Estrutura_de_Dados"/>
      </Topico>
    </compreende>
    <ehCompreendidoPor>
      <Mestrado rdf:ID="Sistemas_Computacao">
        <compreende>
          <Disciplina rdf:ID="Banco_de_Dados">
            <ehCompreendidoPor rdf:resource="#Sistemas_Computacao"/>
            <compreende>
              <Topico rdf:ID="Recuperacao_de_BDs">
                <ehFundamentadaPor>
                  <Topico rdf:ID="Controle_de_Concorrencia">
                    <fundamenta rdf:resource="#Recuperacao_de_BDs"/>
                  </Topico>
                </ehFundamentadaPor>
                <ehCompreendidoPor rdf:resource="#Banco_de_Dados"/>
              </Topico>
            </compreende>
          </Disciplina>
        </compreende>
        <compreende rdf:resource="#Estrutura_de_Dados"/>
        <compreende>
          <Disciplina rdf:ID="Aspectos_Formais">
            <ehCompreendidoPor rdf:resource="#Sistemas_Computacao"/>
          </Disciplina>
        </compreende>
      </Mestrado>
    </ehCompreendidoPor>
  </Disciplina>
  <swrl:Variable rdf:ID="b"/>
  <Topico rdf:ID="Algebra_Relacional">
    <fundamenta>
      <Topico rdf:ID="SQL">
        <ehFundamentadaPor>
          <Topico rdf:ID="Calculo_Relacional">
            <compreende>
              <Topico rdf:ID="QUEL">
                <ehCompreendidoPor
rdf:resource="#Calculo_Relacional"/>
              </Topico>
            </compreende>
            <compreende>
              <Topico rdf:ID="QBE">
                <ehCompreendidoPor
rdf:resource="#Calculo_Relacional"/>
```

```
              </Topico>
            </compreende>
            <fundamenta rdf:resource="#SQL"/>
          </Topico>
        </ehFundamentadaPor>
        <ehFundamentadaPor rdf:resource="#Algebra_Relacional"/>
      </Topico>
    </fundamenta>
  </Topico>
  <swrl:Imp rdf:ID="regra_pre_requisito">
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:ClassAtom>
                    <swrl:argument1>
                      <swrl:Variable rdf:ID="x"/>
                    </swrl:argument1>
                    <swrl:classPredicate rdf:resource="#Topico"/>
                  </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest>
                  <swrl:AtomList>
                    <rdf:rest>
                      <swrl:AtomList>
                        <rdf:first>
                          <swrl:IndividualPropertyAtom>
                            <swrl:propertyPredicate
rdf:resource="#compreende"/>
                            <swrl:argument1>
                              <swrl:Variable rdf:ID="a"/>
                            </swrl:argument1>
                            <swrl:argument2 rdf:resource="#x"/>
                          </swrl:IndividualPropertyAtom>
                        </rdf:first>
                        <rdf:rest>
                          <swrl:AtomList>
                            <rdf:first>
                              <swrl:IndividualPropertyAtom>
                                <swrl:propertyPredicate
rdf:resource="#compreende"/>
                                <swrl:argument1 rdf:resource="#b"/>
                                <swrl:argument2>
                                  <swrl:Variable rdf:ID="y"/>
                                </swrl:argument2>
                              </swrl:IndividualPropertyAtom>
                            </rdf:first>
                            <rdf:rest>
                              <swrl:AtomList>
                                <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                <rdf:first>
                                  <swrl:IndividualPropertyAtom>
                                    <swrl:argument1
rdf:resource="#x"/>
                                    <swrl:argument2
rdf:resource="#y"/>
```

```
                                        <swrl:propertyPredicate
rdf:resource="#ordena"/>
                                    </swrl:IndividualPropertyAtom>
                                  </rdf:first>
                                </swrl:AtomList>
                              </rdf:rest>
                            </swrl:AtomList>
                          </rdf:rest>
                        </swrl:AtomList>
                      </rdf:rest>
                      <rdf:first>
                        <swrl:ClassAtom>
                          <swrl:classPredicate rdf:resource="#Topico"/>
                          <swrl:argument1 rdf:resource="#y"/>
                        </swrl:ClassAtom>
                      </rdf:first>
                    </swrl:AtomList>
                  </rdf:rest>
                </swrl:AtomList>
              </rdf:rest>
              <rdf:first>
                <swrl:ClassAtom>
                  <swrl:classPredicate rdf:resource="#Disciplina"/>
                  <swrl:argument1 rdf:resource="#b"/>
                </swrl:ClassAtom>
              </rdf:first>
            </swrl:AtomList>
          </rdf:rest>
          <rdf:first>
            <swrl:ClassAtom>
              <swrl:classPredicate rdf:resource="#Disciplina"/>
              <swrl:argument1 rdf:resource="#a"/>
            </swrl:ClassAtom>
          </rdf:first>
        </swrl:AtomList>
      </swrl:body>
      <swrl:head>
        <swrl:AtomList>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:propertyPredicate rdf:resource="#ehPreRequisitoDe"/>
              <swrl:argument2 rdf:resource="#y"/>
              <swrl:argument1 rdf:resource="#x"/>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
        </swrl:AtomList>
      </swrl:head>
    </swrl:Imp>
    <swrl:Imp rdf:ID="programa">
      <swrl:body>
        <swrl:AtomList>
          <rdf:rest>
            <swrl:AtomList>
              <rdf:first>
                <swrl:ClassAtom>
                  <swrl:classPredicate rdf:resource="#Disciplina"/>
                  <swrl:argument1 rdf:resource="#y"/>
                </swrl:ClassAtom>
              </rdf:first>
```

```
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:propertyPredicate
rdf:resource="#compreende"/>
                    <swrl:argument2 rdf:resource="#y"/>
                    <swrl:argument1 rdf:resource="#x"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Curso"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#possuiEmPrograma"/>
            <swrl:argument2 rdf:resource="#y"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
  </swrl:Imp>
  <swrl:Imp rdf:ID="ementa">
    <swrl:head>
      <swrl:AtomList>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:propertyPredicate rdf:resource="#possuiEmEmenta"/>
            <swrl:argument2 rdf:resource="#y"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:classPredicate rdf:resource="#Topico"/>
                <swrl:argument1 rdf:resource="#y"/>
              </swrl:ClassAtom>
```

```
            </rdf:first>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:IndividualPropertyAtom>
                    <swrl:propertyPredicate
rdf:resource="#compreende"/>
                    <swrl:argument2 rdf:resource="#y"/>
                    <swrl:argument1 rdf:resource="#x"/>
                  </swrl:IndividualPropertyAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate rdf:resource="#Disciplina"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
  <Topico rdf:ID="Gerenciamento_de_Transacoes"/>
  <Topico rdf:ID="Protocolos"/>
  <Topico rdf:ID="Algoritmos">
    <fundamenta>
      <Topico rdf:ID="Linguagens_de_Consulta">
        <ehFundamentadaPor rdf:resource="#Algoritmos"/>
      </Topico>
    </fundamenta>
  </Topico>
</rdf:RDF>
```

9.3 ONTOLOGIA DOMÍNIO/MODELO ROSA^{+} DO ESTUDO DE CASO

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
  xml:base="http://www.owl-ontologies.com/unnamed.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Colecao"/>
  <owl:Class rdf:ID="Sacola">
    <rdfs:subClassOf rdf:resource="#Colecao"/>
  </owl:Class>
  <owl:Class rdf:ID="Conjunto">
    <rdfs:subClassOf rdf:resource="#Colecao"/>
  </owl:Class>
  <owl:Class rdf:ID="Sequencia">
    <rdfs:subClassOf rdf:resource="#Colecao"/>
  </owl:Class>
  <owl:Class rdf:ID="TipoRelacionamento"/>
  <owl:Class rdf:ID="Antecedente"/>
  <owl:Class rdf:ID="Consequente"/>
  <owl:Class rdf:ID="Hierarquia">
    <rdfs:subClassOf rdf:resource="#Colecao"/>
  </owl:Class>
  <owl:Class rdf:ID="LOSchema">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="LOLogico"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Propriedade"/>
  <owl:Class rdf:ID="LOFisico">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="LO"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Associacao">
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="associacaoEntre"/>
        </owl:onProperty>
        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >2</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Literal"/>
  <owl:Class rdf:ID="Regra"/>
  <owl:Class rdf:about="#LOLogico">
    <rdfs:subClassOf rdf:resource="#LO"/>
  </owl:Class>
  <owl:Class rdf:ID="Descricao">
```

```
    <rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="descritoPor"/>
        </owl:onProperty>
        <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Relacionamento"/>
  <owl:ObjectProperty rdf:ID="temSujeito">
    <rdfs:domain rdf:resource="#Relacionamento"/>
    <rdfs:range rdf:resource="#LO"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="seRelacionaAtravesDe"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="ehDoTipo">
    <rdfs:range rdf:resource="#TipoRelacionamento"/>
    <rdfs:domain rdf:resource="#Relacionamento"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="ehTipoDe"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temPredicado">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="ehRelacionadoPor"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Relacionamento"/>
    <rdfs:range rdf:resource="#Colecao"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temCorpo">
    <rdfs:range rdf:resource="#Antecedente"/>
    <rdfs:domain rdf:resource="#Regra"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehTipoDe">
    <rdfs:domain rdf:resource="#TipoRelacionamento"/>
    <owl:inverseOf rdf:resource="#ehDoTipo"/>
    <rdfs:range rdf:resource="#Relacionamento"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="sameAs">
    <rdfs:domain rdf:resource="#Relacionamento"/>
    <rdfs:range rdf:resource="#Relacionamento"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#associacaoEntre">
    <rdfs:domain rdf:resource="#Associacao"/>
    <rdfs:range rdf:resource="#Literal"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="inversoA">
    <rdfs:domain rdf:resource="#Relacionamento"/>
    <rdfs:range rdf:resource="#Relacionamento"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temPropriedade">
    <rdfs:domain rdf:resource="#TipoRelacionamento"/>
    <rdfs:range rdf:resource="#Propriedade"/>
```

```
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#descritoPor">
    <rdfs:range rdf:resource="#Literal"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Descricao"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temCabeca">
    <rdfs:domain rdf:resource="#Regra"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="#Consequente"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"
/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temDefinicao">
    <rdfs:domain rdf:resource="#Antecedente"/>
    <rdfs:range rdf:resource="#Descricao"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#seRelacionaAtravesDe">
    <rdfs:range rdf:resource="#Relacionamento"/>
    <rdfs:domain rdf:resource="#LO"/>
    <owl:inverseOf rdf:resource="#temSujeito"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#ehRelacionadoPor">
    <rdfs:range rdf:resource="#Relacionamento"/>
    <rdfs:domain rdf:resource="#Colecao"/>
    <owl:inverseOf rdf:resource="#temPredicado"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="possuiLO">
    <rdfs:domain rdf:resource="#Colecao"/>
    <rdfs:range rdf:resource="#LOLogico"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="corpoTemComposicao">
    <rdfs:range rdf:resource="#Associacao"/>
    <rdfs:domain rdf:resource="#Antecedente"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="temLOLogico">
    <rdfs:range rdf:resource="#LOLogico"/>
    <rdfs:domain rdf:resource="#LOSchema"/>
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:ID="nomeDescricao">
    <rdfs:domain rdf:resource="#Descricao"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="nomeAssociacao">
    <rdfs:domain rdf:resource="#Associacao"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="cabecaTemComposicao">
    <rdfs:range rdf:resource="#Associacao"/>
    <rdfs:domain rdf:resource="#Consequente"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
```

```
</owl:FunctionalProperty>
<Relacionamento rdf:ID="ehEnlacadoPor">
  <ehDoTipo>
    <TipoRelacionamento rdf:ID="agregacao">
      <ehTipoDe>
        <Relacionamento rdf:ID="ehFormadoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="possui">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehComponenteDe">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="constitui">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehAbrangidoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <temPropriedade>
        <Propriedade rdf:ID="transitiva"/>
      </temPropriedade>
      <ehTipoDe>
        <Relacionamento rdf:ID="enlaca">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="inclui">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="forma">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe rdf:resource="#ehEnlacadoPor"/>
      <ehTipoDe>
        <Relacionamento rdf:ID="abrange">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="envolve">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehPossuidoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
```

```
  </Relacionamento>
</ehTipoDe>
<ehTipoDe>
  <Relacionamento rdf:ID="tem">
    <ehDoTipo rdf:resource="#agregacao"/>
  </Relacionamento>
</ehTipoDe>
<ehTipoDe>
  <Relacionamento rdf:ID="compreende">
    <temSujeito>
      <LOSchema rdf:ID="disciplina">
        <temLOLogico>
          <LOLogico rdf:ID="Aspectos_Formais"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Estrutura_de_Dados"/>
        </temLOLogico>
        <seRelacionaAtravesDe rdf:resource="#compreende"/>
        <temLOLogico>
          <LOLogico rdf:ID="Banco_de_Dados"/>
        </temLOLogico>
      </LOSchema>
    </temSujeito>
    <temSujeito>
      <LOSchema rdf:ID="topico">
        <seRelacionaAtravesDe rdf:resource="#compreende"/>
        <temLOLogico>
          <LOLogico rdf:ID="QUEL"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Recuperacao_de_BDs"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Protocolos"/>
        </temLOLogico>
        <seRelacionaAtravesDe>
          <Relacionamento rdf:ID="fundamenta">
            <temSujeito rdf:resource="#topico"/>
            <ehDoTipo>
              <TipoRelacionamento rdf:ID="ordenacao">
                <ehTipoDe>
                  <Relacionamento rdf:ID="ehFundamentadoPor">
                    <ehDoTipo rdf:resource="#ordenacao"/>
                  </Relacionamento>
                </ehTipoDe>
                <ehTipoDe rdf:resource="#fundamenta"/>
                <ehTipoDe>
                  <Relacionamento rdf:ID="ehBasePara">
                    <ehDoTipo rdf:resource="#ordenacao"/>
                  </Relacionamento>
                </ehTipoDe>
                <ehTipoDe>
                  <Relacionamento rdf:ID="ehBaseDe">
                    <ehDoTipo rdf:resource="#ordenacao"/>
                    <temSujeito rdf:resource="#topico"/>
                  </Relacionamento>
                </ehTipoDe>
                <ehTipoDe>
                  <Relacionamento rdf:ID="ehCondicaoPara">
                    <ehDoTipo rdf:resource="#ordenacao"/>
                  </Relacionamento>
```

```
                </ehTipoDe>
              </TipoRelacionamento>
            </ehDoTipo>
          </Relacionamento>
        </seRelacionaAtravesDe>
        <temLOLogico>
          <LOLogico rdf:ID="Controle_de_Concorrencia"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="QBE"/>
        </temLOLogico>
        <seRelacionaAtravesDe rdf:resource="#ehBaseDe"/>
        <temLOLogico>
          <LOLogico rdf:ID="Arvores"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Calculo_Relacional"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Linguagens_de_Consulta"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="SQL"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Gerenciamento_de_Transacoes"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Grafos"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Algebra_Relacional"/>
        </temLOLogico>
        <temLOLogico>
          <LOLogico rdf:ID="Algoritmos"/>
        </temLOLogico>
      </LOSchema>
    </temSujeito>
    <temSujeito>
      <LOSchema rdf:ID="curso">
        <seRelacionaAtravesDe rdf:resource="#compreende"/>
        <temLOLogico>
          <LOLogico rdf:ID="Sistemas_Computacao"/>
        </temLOLogico>
      </LOSchema>
    </temSujeito>
    <inversoA>
      <Relacionamento rdf:ID="ehCompreendidoPor">
        <ehDoTipo rdf:resource="#agregacao"/>
      </Relacionamento>
    </inversoA>
    <ehDoTipo rdf:resource="#agregacao"/>
  </Relacionamento>
</ehTipoDe>
<ehTipoDe>
  <Relacionamento rdf:ID="ehMembroDe">
    <ehDoTipo rdf:resource="#agregacao"/>
  </Relacionamento>
</ehTipoDe>
<ehTipoDe>
  <Relacionamento rdf:ID="ehCompostoPor">
```

```
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehAbarcadoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="compoe">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehTidoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehEnvolvidoPor">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehParteDe">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe rdf:resource="#ehCompreendidoPor"/>
      <ehTipoDe>
        <Relacionamento rdf:ID="abarca">
          <ehDoTipo rdf:resource="#agregacao"/>
        </Relacionamento>
      </ehTipoDe>
    </TipoRelacionamento>
  </ehDoTipo>
</Relacionamento>
<Relacionamento rdf:ID="ehGeradoPor">
  <ehDoTipo>
    <TipoRelacionamento rdf:ID="criacao">
      <ehTipoDe>
        <Relacionamento rdf:ID="ehProduzidoPor">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehCriadoPor">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="desenvolve">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="gera">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
```

```
        <Relacionamento rdf:ID="cria">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="produz">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe rdf:resource="#ehGeradoPor"/>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehDesenvolvidoPor">
          <ehDoTipo rdf:resource="#criacao"/>
        </Relacionamento>
      </ehTipoDe>
    </TipoRelacionamento>
  </ehDoTipo>
</Relacionamento>
<TipoRelacionamento rdf:ID="utilizacao">
  <ehTipoDe>
    <Relacionamento rdf:ID="aproveita">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="ehUtilizadoPor">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="utiliza">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="usa">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="usufrui">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="ehAproveitadoPor">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="ehUsufruidoPor">
      <ehDoTipo rdf:resource="#utilizacao"/>
    </Relacionamento>
  </ehTipoDe>
</TipoRelacionamento>
<Relacionamento rdf:ID="seDifereDe">
  <ehDoTipo>
    <TipoRelacionamento rdf:ID="equivalencia">
      <ehTipoDe>
        <Relacionamento rdf:ID="ehMesmaQue">
          <ehDoTipo rdf:resource="#equivalencia"/>
```

```
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe rdf:resource="#seDifereDe"/>
        <ehTipoDe>
          <Relacionamento rdf:ID="seIgualaA">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="seDistingueDe">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="equivale">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="ehDistintoDe">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="seDiferenciaDe">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="seEquiparaA">
            <ehDoTipo rdf:resource="#equivalencia"/>
          </Relacionamento>
        </ehTipoDe>
      </TipoRelacionamento>
    </ehDoTipo>
  </Relacionamento>
  <Associacao rdf:ID="possuiEmEmenta_a_x">
    <associacaoEntre>
      <Literal rdf:ID="x"/>
    </associacaoEntre>
    <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >possuiEmEmenta(?a,?x)</nomeAssociacao>
    <associacaoEntre>
      <Literal rdf:ID="a"/>
    </associacaoEntre>
  </Associacao>
  <Relacionamento rdf:ID="ehPrecisoPara">
    <ehDoTipo>
      <TipoRelacionamento rdf:ID="necessidade">
        <ehTipoDe rdf:resource="#ehPrecisoPara"/>
        <ehTipoDe>
          <Relacionamento rdf:ID="precisaDe">
            <ehDoTipo rdf:resource="#necessidade"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="carece">
            <ehDoTipo rdf:resource="#necessidade"/>
          </Relacionamento>
        </ehTipoDe>
```

```
      <ehTipoDe>
        <Relacionamento rdf:ID="necessita">
          <ehDoTipo rdf:resource="#necessidade"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehNecessarioPara">
          <ehDoTipo rdf:resource="#necessidade"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="demanda">
          <ehDoTipo rdf:resource="#necessidade"/>
        </Relacionamento>
      </ehTipoDe>
    </TipoRelacionamento>
  </ehDoTipo>
</Relacionamento>
<Relacionamento rdf:ID="licencia">
  <ehDoTipo>
    <TipoRelacionamento rdf:ID="permissao">
      <ehTipoDe>
        <Relacionamento rdf:ID="aceita">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="permite">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="admite">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="deixa">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehAdmitidoPor">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehAceitoPor">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehLicenciadoPor">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehPermitidoPor">
          <ehDoTipo rdf:resource="#permissao"/>
        </Relacionamento>
      </ehTipoDe>
```

```
        <ehTipoDe rdf:resource="#licencia"/>
      </TipoRelacionamento>
    </ehDoTipo>
  </Relacionamento>
  <Antecedente rdf:ID="ant_preRequisito">
    <temDefinicao>
      <Descricao rdf:ID="disciplina_b">
        <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >disciplina(?b)</nomeDescricao>
        <descritoPor>
          <Literal rdf:ID="b"/>
        </descritoPor>
      </Descricao>
    </temDefinicao>
    <temDefinicao>
      <Descricao rdf:ID="disciplina_a">
        <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >disciplina(?a)</nomeDescricao>
        <descritoPor rdf:resource="#a"/>
      </Descricao>
    </temDefinicao>
    <corpoTemComposicao>
      <Associacao rdf:ID="compreende_b_y">
        <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >compreende(?b,?y)</nomeAssociacao>
        <associacaoEntre>
          <Literal rdf:ID="y"/>
        </associacaoEntre>
        <associacaoEntre rdf:resource="#b"/>
      </Associacao>
    </corpoTemComposicao>
    <temDefinicao>
      <Descricao rdf:ID="topico_y">
        <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >topico(?y)</nomeDescricao>
        <descritoPor rdf:resource="#y"/>
      </Descricao>
    </temDefinicao>
    <corpoTemComposicao>
      <Associacao rdf:ID="compreende_a_x">
        <associacaoEntre rdf:resource="#x"/>
        <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >compreende(?a,?x)</nomeAssociacao>
        <associacaoEntre rdf:resource="#a"/>
      </Associacao>
    </corpoTemComposicao>
    <temDefinicao>
      <Descricao rdf:ID="topico_x">
        <descritoPor rdf:resource="#x"/>
        <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >topico(?x)</nomeDescricao>
      </Descricao>
    </temDefinicao>
    <corpoTemComposicao>
      <Associacao rdf:ID="fundamenta_x_y">
```

```
        <associacaoEntre rdf:resource="#y"/>
        <associacaoEntre rdf:resource="#x"/>
        <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >fundamenta(?x,?y)</nomeAssociacao>
      </Associacao>
    </corpoTemComposicao>
  </Antecedente>
  <Relacionamento rdf:ID="ehDefinidoPor">
    <ehDoTipo>
      <TipoRelacionamento rdf:ID="implicacao">
        <ehTipoDe>
          <Relacionamento rdf:ID="ehDerivadoPor">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="determina">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="ehDeterminadoPor">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="ehEstabelecidoPor">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="estabelece">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="implica">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="especifica">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="deriva">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="ehResultadoDe">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
        <ehTipoDe>
          <Relacionamento rdf:ID="define">
            <ehDoTipo rdf:resource="#implicacao"/>
          </Relacionamento>
        </ehTipoDe>
```

```
      <ehTipoDe rdf:resource="#ehDefinidoPor"/>
      <ehTipoDe>
        <Relacionamento rdf:ID="resulta">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehOriginadoPor">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="origina">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehEspecificadoPor">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="prove">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="aponta">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
      <ehTipoDe>
        <Relacionamento rdf:ID="ehImplicadoPor">
          <ehDoTipo rdf:resource="#implicacao"/>
        </Relacionamento>
      </ehTipoDe>
    </TipoRelacionamento>
  </ehDoTipo>
</Relacionamento>
<TipoRelacionamento rdf:ID="influencia">
  <ehTipoDe>
    <Relacionamento rdf:ID="ehInspiradoPor">
      <ehDoTipo rdf:resource="#influencia"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="inspira">
      <ehDoTipo rdf:resource="#influencia"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="influi">
      <ehDoTipo rdf:resource="#influencia"/>
    </Relacionamento>
  </ehTipoDe>
  <ehTipoDe>
    <Relacionamento rdf:ID="ehInfluenciadoPor">
      <ehDoTipo rdf:resource="#influencia"/>
    </Relacionamento>
  </ehTipoDe>
</TipoRelacionamento>
```

```
  <Descricao rdf:ID="disciplina_x">
    <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >disciplina(?x)</nomeDescricao>
    <descritoPor rdf:resource="#x"/>
  </Descricao>
  <Consequente rdf:ID="con_ementa">
    <cabecaTemComposicao rdf:resource="#possuiEmEmenta_a_x"/>
  </Consequente>
  <Literal rdf:ID="z"/>
  <Associacao rdf:ID="possuiEmPrograma_a_x">
    <associacaoEntre rdf:resource="#x"/>
    <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >possuiEmEmenta(?a,?x)</nomeAssociacao>
    <associacaoEntre rdf:resource="#a"/>
  </Associacao>
  <Antecedente rdf:ID="ant_programa">
    <corpoTemComposicao rdf:resource="#compreende_a_x"/>
    <temDefinicao rdf:resource="#disciplina_x"/>
    <temDefinicao>
      <Descricao rdf:ID="curso_a">
        <descritoPor rdf:resource="#a"/>
        <nomeDescricao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >curso(?a)</nomeDescricao>
      </Descricao>
    </temDefinicao>
  </Antecedente>
  <Regra rdf:ID="programa"/>
  <Regra rdf:ID="ementa"/>
  <Literal rdf:ID="c"/>
  <Antecedente rdf:ID="ant_ementa">
    <temDefinicao rdf:resource="#topico_x"/>
    <corpoTemComposicao rdf:resource="#compreende_a_x"/>
    <temDefinicao rdf:resource="#disciplina_a"/>
  </Antecedente>
  <Consequente rdf:ID="con_programa">
    <cabecaTemComposicao rdf:resource="#possuiEmPrograma_a_x"/>
  </Consequente>
  <Regra rdf:ID="preRequisito"/>
  <Propriedade rdf:ID="simetrica"/>
  <Associacao rdf:ID="ehPreRequisitoDe_a_b">
    <associacaoEntre rdf:resource="#b"/>
    <nomeAssociacao
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ehPreRequisitoDe</nomeAssociacao>
    <associacaoEntre rdf:resource="#a"/>
  </Associacao>
  <Consequente rdf:ID="con_preRequisito">
    <cabecaTemComposicao rdf:resource="#ehPreRequisitoDe_a_b"/>
  </Consequente>
</rdf:RDF>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo