3 – Ontologias e Representação de Conhecimento

Fonte: http://ceweb.br/livros/dados-abertos-conectados/capitulo-3/
Livro: Dados Abertos Conectados
Seiji Isotani e Ig Albert Bittencourt

Introdução

No Capítulo 2, trabalhamos o conceito de estruturação de dados e como ele é utilizado para auxiliar na descrição de Dados Conectados. Abordamos mais a fundo também o "sistema de 5 estrelas" de divulgação de dados abertos, além de apresentarmos alguns exemplos de sua utilização tanto no setor público quanto no privado. Contudo publicar um dado de maneira indiscriminada torna muito difícil o seu posterior uso. Isso ocorre porque um dado pode ser descrito, representado e interpretado de diferentes maneiras.

Para entender mais sobre o problema de representação do conhecimento, vamos discutir alguns dos problemas ao lidar com a tripla descrição-representação-interpretação de dados. Um dos problemas na descrição de dados é o vocabulário a ser utilizado para representar um conceito desejado. Por exemplo, para descrever uma pessoa que está assistindo às aulas em uma universidade, poderíamos utilizar o termo "aluno". Entretanto, existem outros, como: "estudante", "aprendiz" ou até "universitário". Além disso, no caso de uma pessoa que está lecionando aulas, poderíamos usar os termos "professor", "instrutor", "mestre" e assim por diante. Nesses exemplos, apenas dois conceitos sendo apresentados podem ser descritos por meio de uma diversidade de termos. Dessa forma, um usuário que quiser publicar dados abertos terá que escolher um conjunto de termos que descrevem adequadamente seus dados. Esse conjunto de termos é conhecido como vocabulário.

Além de escolher o vocabulário que melhor identifique um conjunto de dados, outro desafio é representar esses dados de maneira a aumentar a expressividade do dado dentro do contexto em que foi criado e reduzir a ambiguidade de sua posterior interpretação. Por exemplo, para os conceitos apresentados anteriormente, que podem ser descritos pelos termos "aluno" e "professor", existe a necessidade de dar mais riqueza de representação para que as seguintes perguntas possam ser respondidas facilmente: Quem pode ser professor? Quem pode ser aluno? Uma pessoa pode ser professor e aluno ao mesmo tempo?

Para responder a essas perguntas é preciso propor alguma forma de representação. No caso desse exemplo, uma representação seria indicar que professor e aluno são papeis (em inglês, roles) que um ator, representado pelo conceito de "pessoa", pode exercer no contexto de uma universidade. Assim deixa-se claro que o papel de professor ou aluno só existe dentro do contexto da universidade e é assumido por uma pessoa. Ademais, é possível colocar restrições nos atributos de uma pessoa para que ela possa assumir um determinado papel. Desse modo, apenas uma pessoa contratada pela universidade com o título de mestre ou doutor poderia assumir o papel de professor. Similarmente, apenas uma pessoa que finalizou o Ensino Médio e está regularmente matriculada em um curso da universidade poderia assumir o papel de aluno. Nessa situação, uma pessoa, também, poderia, ao mesmo tempo, assumir o papel de professor e aluno dentro de uma mesma universidade.

Essas restrições precisam estar claras, para que interpretações incorretas não ocorram. Caso as restrições não sejam explicitamente definidas, os dados disponibilizados podem ser interpretados erroneamente. No exemplo apresentado, os dados poderiam ser interpretados incorretamente em países em que um aluno não precisa cursar o Ensino Médio para conseguir se matricular em uma universidade (nos EUA, por exemplo).

Dessa maneira, para reduzir o problema de interpretação das representações dos dados, devem ser utilizados mecanismos e linguagens de representação/modelagem (visual ou lógica/formal) como o UML [48] (Unified Modeling Language) e o OWL (Web Ontolgy Language), que explicitam as relações (restrições e hierarquia, por exemplo) entre conceitos de maneira (semi-) formal, permitindo que tanto pessoas quanto máquinas possam compreender os conceitos que representam os dados disponibilizados.

No contexto da Web, uma das formas de representação mais robustas atualmente ocorre por meio do uso de ontologias e OWL. Dessa forma, neste trabalho, introduziremos esses conceitos para que o leitor seja capaz de entender o potencial uso das ontologias e da OWL na disponibilização de dados abertos. E, no próximo capítulo, discutiremos as técnicas de criação de ontologias provindas da área de Engenharia de Ontologias.

3.2 Ontologias

O termo "ontologia ” tem origem em um ramo da Filosofia (Metafísica), que estuda a natureza do “ser” e a “existência“. Para os filósofos, Ontologia visa explicar todas as coisas do mundo, estabelecendo sistematicamente sua linhagem conceitual. Na Ciência da Computação, o significado e finalidade desse termo s ã o (um pouco) diferentes; uma ontologia pode ser definida como um conjunto de conceitos fundamentais e suas relações, que capta como as pessoas entendem (ou interpretam) o domínio em questão e permite a representação de tal entendimento de maneira formal, compreensível por humanos e computadores (Mizoguchi, 2004).

Pesquisadores e demais especialistas na área têm diferentes entendimentos sobre o que significa uma ontologia. Nesse capítulo, centralizaremos na área da Ciência da Computação. Segundo Gruber (1993), ontologia é especificação explícita de uma conceitualização. A conceitualização refere-se ao significado de conceitos e suas relações, dado o contexto do domínio. E “ especificação ” significa uma representação formal, declarativa e explicita dos mesmos conceitos e relações. Outra definição, dada por Swartout et al. (1999), é que uma ontologia é a estrutura básica ou couraça em torno da qual uma base de conhecimento pode ser construída. Guarino (1997) também enfatiza que uma ontologia pode ser modelada para permitir o compartilhamento de conhecimento e a sua reutilização em diferentes aplicações.

Essas definições caracterizam bem o porquê de ontologias estarem atraindo recentemente muitos pesquisadores e desenvolvedores na área de Dados Abertos Conectados. Primeiro, elas fornecem uma estrutura conceitual comum sobre a qual podemos desenvolver bases de conhecimento compartilháveis e reutilizáveis. E, em segundo lugar, facilitam a interoperabilidade e a fusão das informações (mash-ups [49]), que viabilizam a criação de aplicações computacionais poderosas e mais inteligentes.

3.2.1 Composição de uma Ontologia

Em Ciência da Computação, mais especificamente no campo da Inteligência Artificial, uma ontologia é constituí da pelo seguinte (Mizoguchi, 2004; Isotani, 2009):

> 1. Um conjunto de conceitos essenciais resultantes da articulação do conhecimento básico presente em um determinado domínio. Esses conceitos podem ser representados usando um vocabulário especializado.
>
> 2. O corpo de conhecimento, que descreve o domínio utilizando os conceitos essenciais. Ele é composto por:
>
> • Uma hierarquia (classe/subclasse) resultante das relações entre conceitos *is-a*.
>
> • Um conjunto de rela çõ es importantes entre conceitos al m das relações *is-a* (por exemplo, *part-of*).
>
> • Uma axiomatização de restrições semânticas entre esses conceitos e relações.

Formalmente, podemos definir uma ontologia como um relacionamento de quatro elementos, representado por O = {C, R, I, A}, onde (Kiryakov, 2006):

> • **C** - é o conjunto de classes que representam os conceitos em um dado domínio de interesse;
>
> • **R** - é o conjunto de relações ou associações entre os conceitos do domínio;
>
> • **I** - é o conjunto de instâncias derivadas das classes, ou ainda, os exemplos de conceitos representados em uma ontologia;
>
> • **A** - é o conjunto de axiomas do dom í nio, que servem para modelar restrições e regras inerentes à s instâncias.

Esses elementos que constituem uma ontologia são fundamentais para a criação de uma estrutura que representa o conhecimento de um domínio.

Muitas vezes, durante o desenvolvimento de uma ontologia, as pessoas acabam gastando muito tempo para discutir a terminologia a ser utilizada (isto é, vocabulário), ao invés de discutir e compreender os conceitos essenciais do domínio. Gostaríamos de enfatizar que, do ponto de vista dos autores (seguindo uma perspectiva orientada a conceito), uma ontologia n ã o é um conjunto de termos interligados que formam um vocabulário (embora, algumas vezes, possa ser utilizada dessa forma). Mesmo que um termo possa ser considerado similar a um conceito, eles são diferentes em sua essência. Um termo é essencialmente um rótulo para um conceito . Quando falamos de um termo, o rótulo (palavra do vocabulário) torna-se o enfoque do problema. No entanto, para criar uma ontologia, o enfoque do problema é a definição de um conceito; e, portanto, o rótulo dado a ele passa a ter menos importância. Resumindo, uma ontologia propõe-se a representar e expressar o significado de conceitos, e não de termos (Mizoguchi, 2004; Guizzardi, 2007; Isotani, 2009). Isto ficar á mais claro no próximo capítulo, quando apresentamos a Figura 4.2, onde abordamos os diferentes significados para a palavra Bank . Ou seja, caso

os projetistas da ontologia se concentrem no termo Bank, o enfoque ser á dado de forma inadequada.

Se o leitor concorda com o ponto de vista dos autores, isto é, de que apenas a definição de um vocabulário interconectado n ã o resulta em uma (boa) ontologia, então como é criada uma ontologia? Como é que se decide a qualidade de uma ontologia?

Uma possível resposta a essas perguntas é dada por Mizoguchi (2004): " quanto mais ontológica for a ontologia, melhor ". Isso significa que uma ontologia é bem desenvolvida e tem maior qualidade quando o domínio pode ser explicado e as propriedades essenciais dos conceitos são explicitamente representadas, chegando perto da conceituação fundamental do conhecimento. Desse ponto de vista, o corpo de conhecimento que descreve o domínio é o núcleo das ontologias. Assim, ao criar uma ontologia, além da definição de conceitos e termos para identificá-los, deve-se: (a) fazer clara distinção entre as funções e conceitos básicos; (b) identificar o uso apropriado das relações, especialmente as relações *is-a* e *part-of*; (c) evitar a herança múltipla; (d) distinguir corretamente o que é atributo e o que é propriedade; e (e) realizar muitas outras decisões importantes, a fim de produzir uma boa ontologia.

No próximo capítulo abordaremos mais profundamente a metodologia utilizada para criar tais ontologias, apresentando a área de Engenharia de Ontologias .

3.2.2 Por que utilizar ontologias?

A concepçã o e uso das ontologias sempre fizeram parte da proposta da Web Semântica, conforme introduzimos no Capítulo 1. Elas est ã o no centro da arquitetura proposta por Tim Berners-Lee (Figura 7 do Capítulo 1) e, ao longo da última década (2004-2014), têm-se mostrado uma das tecnologias-chave na criação de aplicativos mais adequados para lidar com grandes quantidades de informações de maneira inteligente (Mcguinness, 2004; Horrocks, 2008).

De acordo com diversos autores, as ontologias oferecem o apoio necessário para resolver alguns dos problemas que cercam a construção de tecnologias que utilizam bases de dados com representação formal (ou bases de conhecimento – do inglês knowledge-bases), como a seguir (Mizoguchi, 2004; D'aquin and Noy, 2012; Devedzic, 2006):

> 1. Premissas básicas são deixadas implícitas nas bases de dados, com isso, impedindo a reutilização e compartilhamento do conhecimento representado.
>
> 2. Não h á modelos genéricos comuns sobre os quais possamos construir bases de dados e aplicativos de maneira simplificada.
>
> 3. Não há tecnologia viável que permita acumulação incremental dos dados (isto é, estender rapidamente a base de dados).

Na era da Web de dados, em que a informação e o conhecimento são fragmentados na rede e os recursos estão em constante evolução, o desenvolvimento de aplicativos baseados em dados abertos não pode seguir o paradigma em que as bases de dados são estáticas e criadas para um problema muito específico em um domínio restrito. Atualmente, tanto o mercado quanto a academia exigem base de dados conectados,

altamente compartilháveis, que permitam a interoperabilidade e a possibilidade de lidar com o acú mulo de conhecimento (isto é, novos dados conectados) disponível na Web .

Essa mudança de paradigma requer que os aplicativos atuais mudem de enfoque (Isotani et al., 2015; Dermeval et al., 2015):

1. Do paradigma de desenvolvimento orientado a objetos para processo orientado a dados. Geralmente, um dos principais problemas durante o desenvolvimento de aplicativos que lidam com grandes quantidades de dados de maneira inteligente tem sido como criar heurística e bases de conhecimento para guiar ou reproduzir comportamentos considerados adequados. Em outras palavras, o enfoque tem sido sobre os processos computacionais que guiam o comportamento do computador. No entanto, a partir do paradigma da Web de dados e com a crescente disseminação dos Dados Abertos Conectados, a ênfase dá-se em como utilizar os dados de maneira inteligente para auxiliar pessoas durante a tomada de decisões e na resolução de tarefas complexas. Aplicativos inteligentes, baseados em Dados Abertos Conectados, devem trazer informações e recursos adequados para as pessoas, para que possam juntas resolver os problemas e tarefas de forma mais eficiente. Para atingir tal objetivo, o processo de desenvolvimento de software deve seguir um paradigma orientado a dados. Assim os aplicativos inteligentes poderão utilizar de maneira eficaz os dados (em formato aberto e conectado) disponibilizados na Web para fornecer novas funcionalidades e conhecimentos aos usuários finais.
2. Do paradigma de desenvolvimento centrado no usuário para centrado na informação. O objetivo principal do desenvolvimento de sistemas baseados em dados não é o de melhorar como desenvolver funcionalidades para resolver um problema específico. Em vez disso, estamos interessados em ampliar a capacidade humana, criando sistemas que têm a habilidade de oferecer informações provindas da análise de grandes conjuntos de dados e, dessa forma, ajudar na resolução de problemas complexos do mundo real. Para conceber tais sistemas, existe a necessidade de compreender a semântica por trás dos Dados Conectados e os processos humanos de aquisição de informação para desenvolver adequadamente sistemas e interfaces para visualização de dados.

Além dessa mudança de enfoque no desenvolvimento de sistemas, as ontologias também oferecem diversos benefícios para publicação e consumo de dados com alta qualidade, isto é , atingindo as cinco estrelas apresentadas nos capítulos anteriores. A publicação de Dados Abertos Conectados em conjunto com ontologias exige menos esforço tanto na definição do modelo de dados quantos na integração e reuso de outras informações disponíveis na Web. Por exemplo, as ontologias não permitem que a publicação e o consumo de dados abertos ocorram de maneira inconsistente ao usar regras de inferência que possibilitam checar automaticamente a corretude das relações entre os dados e suas instâncias. Outro benefício é a possibilidade de utilizar m é todos para representação de dados tanto no nível procedural (por exemplo, utilizando SKOS - <http://www.w3.org/2004/02/skos/>) quanto no nível conceitual (por exemplo, utilizando ontologias de topo como as disponíveis em <http://suo.ieee.org/>). As ontologias também oferecem um bom balanço entre os esforços de criar um bom modelo de dados (abordado no Capítulo 4) e o valor que ele proporciona para o entendimento e consumo desses dados.

3.2.3 Tipos de Ontologias

Existem vários tipos de ontologias (Guarino, 1997) . Nesta seção, destacaremos as duas formas mais comuns de diferenciar ontologias: 1) ontologias pesadas vs. ontologias leves; 2) ontologias de domínio vs. ontologias de tarefa.

As ontologias leves (*lightweight ontologies*) são aquelas que não se preocupam em definir detalhadamente cada conceito representado. A principal ênfase das ontologias leves é definir a taxonomia que representa a relação hierárquica entre conceitos. Esse tipo de ontologia vem sendo utilizado na Web para categorizar grandes quantidades de dados, principalmente em portais como Yahoo! e AOL (Bechhofer et al., 2006; Bizer , 2009) . Ontologias pesadas ou densas (*heavyweight ontologies*) enfocam não apenas a taxonomia, mas também a representação rigorosa da semântica entre os conceitos. O desenvolvimento de ontologias pesadas requer a definição de cada conceito, a organização desses conceitos baseados em princípios bem definidos, uma definição formal da semântica entre os conceitos e suas relações, além de outras considerações. Para criar bases de conhecimento reusáveis e compartilháveis é fundamental definir ontologias pesadas.

Ontologias de domínio e de tarefa são necessárias para criar sistemas mais flexíveis e inteligentes e que possam ser aplicados em diversos domínios. A ontologia de domínio define e caracteriza o domínio no qual as tarefas ocorrem, e a ontologia de tarefa representa os processos e atividades para resolver um determinado problema abstraindo o contexto do domínio. Em outras palavras, a ontologia de domínio representa o conhecimento sobre um tópico, enquanto a ontologia de tarefa representa a habilidade de aplicar esse conhecimento para resolver problemas em diferentes situações. Essa distinção é muito importante, pois, por meio dela, torna-se possível criar sistemas e bases de conhecimento mais modulares, compartilháveis e extensíveis.

Para criar uma ontologia de tarefa, inicialmente, os passos de resolução de problemas precisam ser decompostos em ações básicas. Além disso, a estrutura hierárquica que representa as restrições para executar essas ações deve ser desenvolvida. De acordo com Mizoguchi (2004), as seguintes categorias conceituais devem ser consideradas no desenvolvimento de uma ontologia de tarefa: (a) papéis (*task roles*): refletem o papel desempenhado pelos objetos do domínio durante o processo de resolução de problemas; (b) ações (*task actions*): representam uma unidade de atividade contida no processo de resolução de problemas; e (c) os estados dos objetos e o fluxo da informação, além de papéis e tarefas, são fatores que precisam estar formalmente representados. A Figura 3.1 apresenta uma classificação de ontologias de acordo com as dimensões de expressividade, propósito e especificidade.

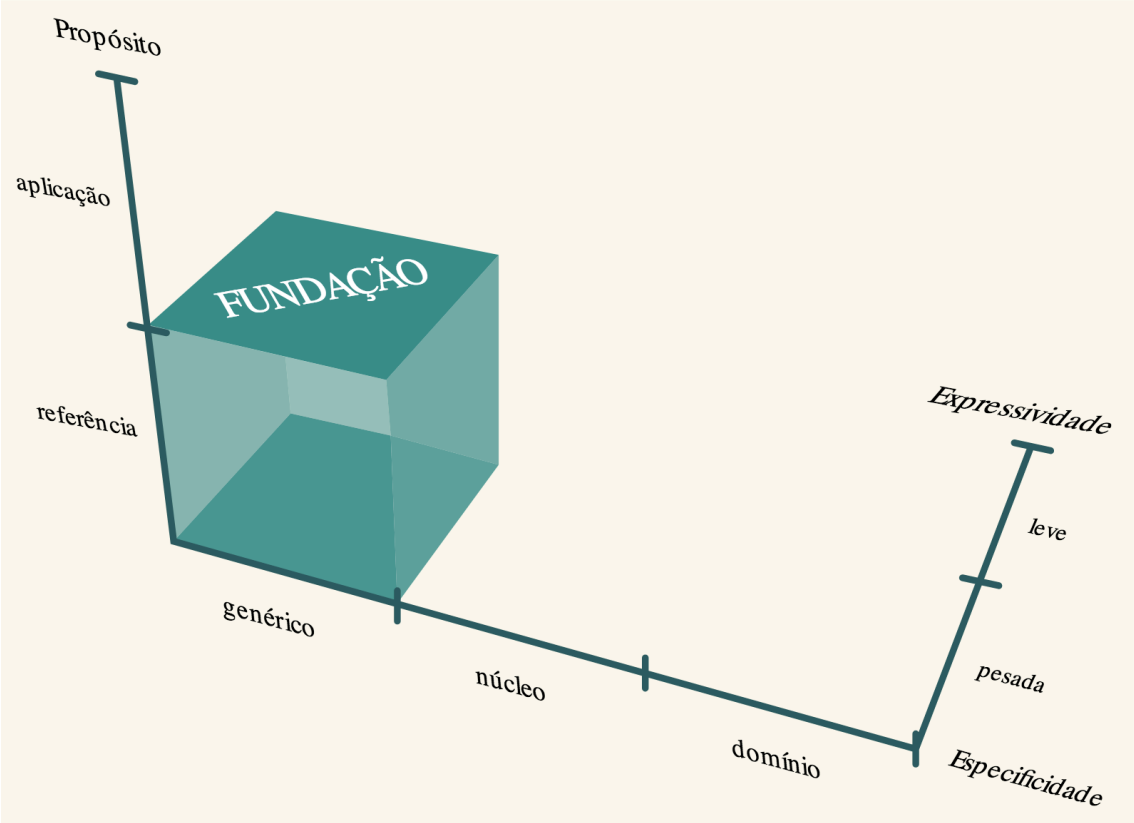


Figura 3.1 Classificação de ontologias.

3.2.4 Representação de ontologias

Basicamente existem duas maneiras de representar ontologias. A primeira delas é a representação formal e a segunda é a representação gráfica. A representação formal é usada para que as ontologias possam ser consumidas por computadores, enquanto a representação gráfica é usada para a compreensão humana. Ambas são importantes, uma vez que a falta de uma dessas representações afeta a qualidade/uso da ontologia.

Quanto a representação formal consumida por computadores, existem algumas linguagens para representação de ontologias (Patel-schneider, 2005) . Normalmente, usa-se lógica de predicados, lógica descritiva ou linguagens baseadas em "quadros" (frames). No entanto, linguagens mais expressivas (conhecidas como linguagens de descrição de ontologias) têm sido propostas (Horrocks et al., 2003) . Atualmente, as linguagens mais populares para descrever ontologias são RDF [53] /RDF-S [54] e OWL [55] .

Resource Description Framework (RDF) é a especificação proposta pelo World Wide Web Consortium (W3C) para descrever metadados. Ela permite criar triplas que contêm um nó sujeito, uma relação chamada de predicado e o nó objeto (sujeito, predicado, objeto). Mediante essa tripla, é possível indicar a relação entre dados e usá-la para representar a semântica contida neles. Por exemplo, é possível indicar a relação entre um autor e seus livros de forma que não apenas pessoas compreendam o significado dessa relação, mas que computadores também possam compreendam esta informação. RDF fornece uma maneira simples de representar triplas.

Para expressar a semântica através de triplas, também conhecidas como vocabulário RDF, é necessária a definição de tags. RDF-Schema (ou RDF-S) é utilizada para tal fim. RDF-

S é a especificação que define classes, propriedades e seus relacionamentos, que podem ser usados para descrever triplas. Isso inclui a definição de tags e sua estrutura hierárquica (taxonomia) para restringir os valores de uma tripla representada em RDF. Assim a RDF-S fornece os elementos mínimos para a descrição de ontologias [57]. Embora RDF-S possa ser usada para descrever ontologias, ela tem algumas limitações, especialmente para apoiar o raciocínio computacional dos dados disponíveis na Internet (Patel-schneider, 2005) . Assim, uma linguagem mais expressiva foi desenvolvida e é conhecida como Web Ontology Language (OWL).

A OWL também é uma linguagem desenvolvida e aprovada pelo W3C. Ela tenta satisfazer o formalismo exigido pela comunidade de Web Semântica para que programas possam compreender e responder a consultas de agentes (pessoas ou outros programas) por meio do uso de descrições ontológicas (Horrocks et al., 2003) . Atualmente, a OWL é a linguagem mais utilizada para representar ontologias formalmente, possui variantes da linguagem que lidam com a escalabilidade e expressividade das ontologias e permite que aplicações com diferentes propósitos sejam construídas (Mizoguchi, 2004) [58].

Existem muitas maneiras de representar graficamente uma ontologia, uma vez que esta é composta principalmente por conceitos e suas relações. Algumas formas comuns para representar graficamente ontologias são grafos, UML, estrutura de árvore, além de outras. Para exemplificar a diferença entre uma representação formal e uma representação gráfica, vamos definir o conceito de bicicleta (Isotani, 2009) : uma bicicleta (bicycle) é um tipo de veículo (vehicle) de transporte, ou seja, pode-se definir a relação “ bicycle is-a vehicle ”. Além disso, uma bicicleta pode ser especializada em bicicletas esportivas (sport bicycle) e bicicletas para uso na cidade (city bicycle). Finalmente, é possível identificar os atributos e propriedades da bicicleta como cor, peso, tamanho, etc. Como resultado, a Figura 3.2 mostra parte da representação da ontologia que descreve uma bicicleta utilizando OWL, enquanto a Figura 3.3 mostra a representação gráfica da uma parte mais completa da mesma ontologia utilizando uma ferramenta gráfica conhecida como HOZO.

```
<owl:Class rdf:ID="Vehicle">
    <rdfs:label>Vehicle</rdfs:label>
    <rdfs:SubClassOf rdf:resource="#Any" />
</owl:Class>
<owl:Class rdf:ID="sport_cycle">
    <rdfs:label>sport_cycle</rdfs:label>
    <rdfs:SubClassOf rdf:resource="#bicycle" />
</owl:Class>
<owl:Class rdf:ID="city_cycle">
    <rdfs:label>city_cycle</rdfs:label>
    <rdfs:SubClassOf rdf:resource="#bicycle" />
</owl:Class>
<owl:Class rdf:ID="bicycle">
    <rdfs:label>bicycle</rdfs:label>
    <rdfs:SubClassOf rdf:resource="#Vehicle" />
    <rdfs:SubClassOf>
        <owl:Restriction>
            <owl:cardinality rdf:datatype="http://www.w3.org/
            <owl:onProperty rdf:resource="#has_body_color" />
        <owl:Restriction>
    </rdfs:SubClassOf>
    <rdfs:SubClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#has_body_color" />
            <owl:allValuesFrom rdf:resource="#Color" />
```

Figura 3.2 Representação da ontologia de bicicleta em OWL.
Fonte: http://www.hozo.jp/>

Na Figura 3.3, p/o e a/o que aparecem na frente dos conceitos representam, respectivamente, as relações de *part-of* e *attribute-of* . Assim o corpo da bicicleta (frame), o pedal, o guidão e a roda são parte (*part-of*) da bicicleta. Enquanto cor e peso são atributos dela.

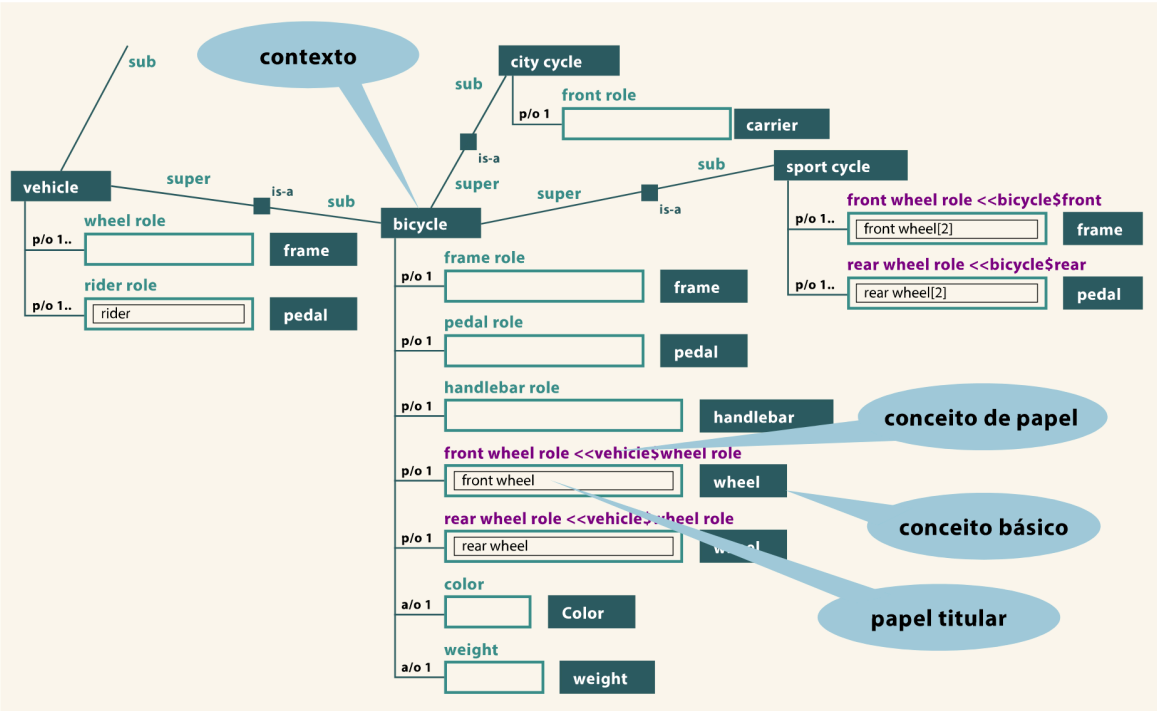


Figura 3.3. Relações de part-of e attribute-of.

O corpo do conhecimento de uma ontologia é responsável por representar o núcleo do domínio e dar significado (e propriedades) aos conceitos representados. Portanto, para

criar boas ontologias pesadas, é fundamental ter uma metodologia que auxilie sua definição e construção. Mizoguchi (2004) indica que, para criar uma boa ontologia, além da definição dos conceitos e dos termos para rotulá-los, é necessário:

1. Fazer uma distinção entre conceitos básicos e papeis ou função (roles) que um conceito pode ter dentro de um contexto. Por exemplo, o conceito professor pode ser definido como a função de uma pessoa dentro do contexto escolar (se a escola desaparecer o professor também desaparece), enquanto o conceito pessoa é um conceito básico que não depende de contexto.
2. Identificar as várias relações entre conceitos representando de forma explícita as relações *is-a* , *part-of* e *attribute-of* .
3. Evitar a herança múltipla.
4. Distinguir e definir corretamente o que é um atributo e uma propriedade de um conceito, além de realizar decisões importantes a fim de produzir uma ontologia de boa qualidade.

3.3 Linguagem de ontologias da Web

A OWL (do Inglês Ontology Web Language) é considerada a linguagem de ontologias da Web e é bastante utilizada para o desenvolvimento de aplicações baseadas na Web Semântica. Como descritas nas Figuras 1.7 e 1.8, a OWL é uma linguagem baseada nas especificações do RDF/RDF-S. Isto quer dizer que OWL, por um lado, herda as características do RDF como a estrutura baseada em triplas e a descrição de recursos com URI, e, por outro lado, herda a semântica descrita no Esquema RDF.

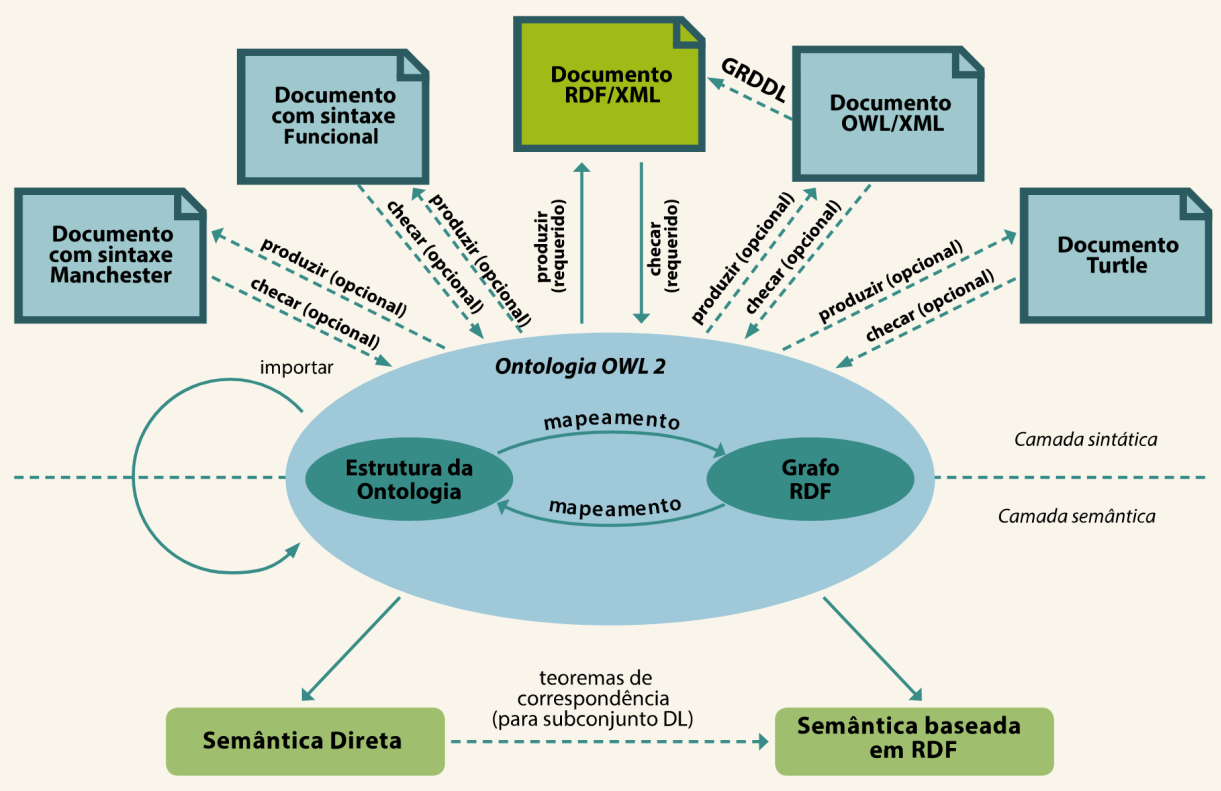
É importante destacar que há muita descrição errada sobre o que é OWL e como aplicá-la. Isto ocorre pela complexidade inerente ao termo Ontologias e pela expressividade da linguagem OWL. Destacamos abaixo três características que não são inerentes à OWL (Hitzler et al., 2012) :

1. Não é uma linguagem de programação: OWL é uma linguagem declarativa que descreve um determinado universo do discurso de forma lógica. A partir do momento que se descreve conhecimento por meio de ontologias, pode-se fazer uso de ferramentas para inferir novas informações sobre o universo de discurso. No entanto a forma que essas ferramentas fazem inferência sobre ontologias não faz parte do escopo da OWL.
2. Não é uma linguagem de esquema para conformidade sintática: não faz parte do escopo da OWL prescrever como certo documento deve ser sintaticamente estruturado. Ou seja, a Linguagem de Ontologias da Web não obriga que determinada parte do conhecimento esteja sintaticamente presente.
3. Não é um banco de dados: essa característica faz com que haja muita aplicação errada de OWL. O que se deve compreender é que os documentos OWL armazenam as informações (instâncias) e, por essa razão, faz o armazenamento dos dados. A principal diferença entre banco de dados e OWL é a semântica utilizada em cada um deles. Os bancos de dados são mundos fechados (do inglês Closed-World Assumptions), enquanto que as ontologias são mundos abertos (do inglês Open-World Assumptions). Isso implica dizer que, se determinado fato não está presente em um banco de dados, ele é considerado falso. Já em OWL ou nos mundos abertos, a implicação é diferente, pois se determinado fato não está presente, ele é considerado desconhecido, pois é possível que seja verdadeiro. Por

exemplo, se há uma determinada declaração como a descrita abaixo sobre o cidadão brasileiro Seiji, não havendo descrição de que Ig também é um cidadão brasileiro, essa pergunta é considerada falsa no mundo fechado.

Declaração: <Seiji> <é cidadão> <Brasil>
Pergunta: <Ig> <é cidadão> <Brasil>?
Resposta (mundo fechado): Não!
Resposta (mundo aberto) Não Sei!

Sob um ponto de vista didático, podemos dividir a OWL em duas camadas, sendo uma para descrever a sintaxe e outra para a semântica. A Figura 3.4 apresenta a estrutura da linguagem OWL 2 (W3C OWL Working Group, 2012) .



A Figura 3.4. Estrutura da linguagem OWL 2

As duas subseções que seguem têm por objetivo descrever tanto a camada sintática quanto a camada semântica.

3.3.1. Sintaxe da linguagem OWL

Como podemos observar na Figura 3.4, toda ontologia criada em OWL 2 possui uma estrutura sintática mandatória que é baseada em RDF/XML. Além do documento RDF/XML, é possível criar documentos OWL com mais quatro diferentes formas de serialização, de acordo com a tabela a seguir.

Tabela 3.1 Formatos de serialização Owl e seus Propósitos

Serialização	Status	Propósito

RDF/XML	Mandatória	Formato obrigatório, podendo ser reconhecido por qualquer software OWL.
OWL/XML	Opcional	Processamento simples com ferramentas XML
Sintaxe Funcional	Opcional	Visualização simples da estrutura formal da OWL
Sintaxe Manchester	Opcional	Leitura/Escrita simples de Ontologias em Lógica de Descrição
Turtle	Opcional	Leitura/Escrita simples de triplas RDF

Os formatos RDF/XML e Turtle foram apresentados no Capítulo 2. Já o formato OWL/XML respeita a sintaxe de documentos XML e possui o propósito de processar documentos OWL por meio de ferramentas de leitura/escrita de XML, como XQuery . Esse é um formato mais comum para soluções corporativas que já fazem uso de documento XML. A sintaxe funcional apresenta uma forma simples de visualizar e compreender documentos OWL, enquanto a serialização em sintaxe Manchester apresenta uma maneira simples de manipular estruturas em lógica de descrição.

Apesar de OWL possuir cinco formatos de serialização, apenas um é mandatório, e os outros devem ser usados de acordo com a necessidade do consumidor da ontologia. Os ontologistas só precisam se preocupar com um formato e, da mesma forma, que explicamos sobre os formatos de serialização RDF, não há a necessidade de criar uma descrição em diferentes formatos. Frisamos que, independentemente do formato de serialização, a linguagem OWL 2 possui uma estrutura conceitual bem-definida que é transformada para os diferentes formatos.

Não podemos esquecer que o propósito da OWL é representar conhecimento. As noções básicas de modelagem de conhecimento em OWL consideram três aspectos básicos (Hitzler et al., 2012) :

1. Entidades: elementos usados para referenciar um objeto do mundo real. Mediante as entidades, os termos primitivos de uma determinada ontologia são definidos. Por exemplo, podemos representar o grupo de todas as pessoas pela entidade <Person>. As entidades descritas em OWL são identificadas por um IRI e podem ser de diferentes tipos.
2. Expressões: combinação de entidades para formar descrições mais complexas, formadas por meio das expressões criadas pelos termos primitivos. Por exemplo, podemos representar o conjunto de todos os professores doutores mediante a conjunção das classes <Professor> e <Doctor> e criar a nova entidade <DoctorProfessor>.
3. Axiomas: as declarações básicas que uma ontologia em OWL expressa. Por meio da utilização dos axiomas, pode-se fazer inferências sobre as entidades. Por exemplo, se definimos que <Student> <SubClassOf> <Person>, pode-se concluir que uma determinada instância de estudante é também uma instância de pessoa.

Além dos elementos básicos descritos acima, não podemos esquecer que os recursos descritos em OWL possuem um IRI associado. Dessa forma, a Figura 3.5 apresenta a estrutura de ontologias descritas em OWL 2 (Bock et al., 2012) .

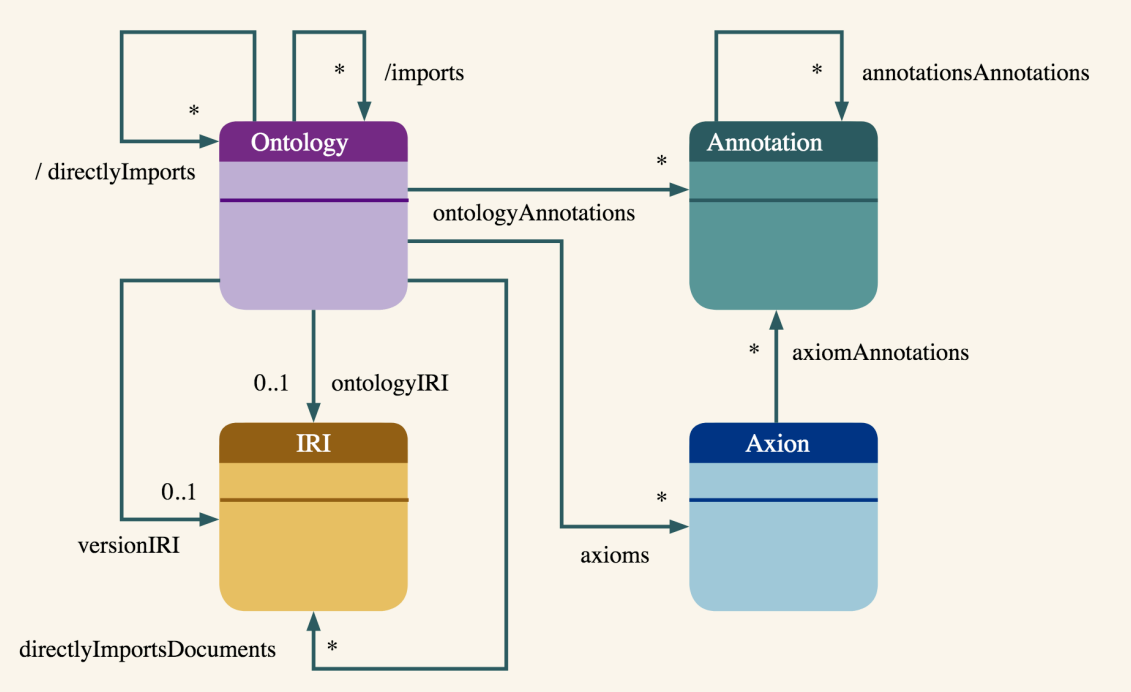


Figura 3.5. Estrutura de ontologias descritas em OWL 2.

Como apresentamos na Figura 3.5, todo documento OWL possui um conjunto de axiomas e um IRI associado. Observe que uma ontologia pode importar outras ontologias, o que caracteriza o reuso de ontologias. Além disso, uma ontologia pode possuir diferentes IRIs, com o objetivo de caracterizar versões daquelas ontologias. Outro elemento presente é Annotation, que permite que a ontologia possua anotações, de modo que a ontologia possua um conjunto de informações sobre ela (ou metainformação). Por exemplo, podemos querer identificar a organização que criou o documento, bem como os seus autores.

Uma vez explicada a estrutura das ontologias descritas em OWL 2, dissertamos sobre os elementos básicas da modelagem OWL. Da mesma forma que identificamos na Subseção 2.3.2 que um Esquema RDF também o faz, uma ontologia OWL possui classes (owl:Class), propriedades (owl:ObjectProperty e owl:DataProperty) e indivíduos ou instâncias (owl:Individual). Abaixo é apresentado um exemplo em sintaxe funcional de classe, propriedade e indivíduo.

Class (:Person)
NamedIndividual (:Mary)
NamedIndividual (:John)
ClassAssertion (:Person :Mary)
ClassAssertion (:Person :John)
ObjectProperty (:hasWife)
ObjectPropertyAssertion (:hasWife :John :Mary)

Pelo exemplo acima, temos a descrição de uma classe (<Person>), duas instâncias (<Mary> e <John>), uma propriedade (<hasWife>) e uma relação (<John> <hasWife> <Mary>). Outra característica presente em OWL que também está presente no RDF-S é a possibilidade de gerar hierarquias de classes e propriedades. As hierarquias são definidas mediante a especificação dos axiomas subclasses (owl: SubClassOf) e

subpropriedades (owl:SubObjectPropertyOf e owl:SubDataPropertyOf). O exemplo abaixo apresenta a utilização desses conceitos.

SubClassOf (:Woman :Person)
SubPropertyOf (:hasWife :hasSpouse)

De acordo com o exemplo, temos a identificação de que mulher (<Woman>) é uma subclasse de pessoa (<Person>), o que implica que qualquer instância de <Woman> é também uma instância de <Person>. Esse mesmo cenário está presente com as propriedades, nas quais indicamos no exemplo que esposa (<hasSpouse>) é subpropriedade de mulher casada [61] (<hasWife>). Esse conceito pode ser aplicado também para a utilização de sinonímias e quase-sinonímias, definindo assim propriedades equivalentes. No entanto, é importante frisar que classes, propriedades e indivíduos possuem elementos para descrever conceitos equivalentes. Por exemplo, podemos dizer que a classe pessoa (<Person>) é equivalente à classe humano (<Human>) e que as propriedades de filho e idade de cada classe são também equivalentes.

EquivalentClasses (ontA:Person ontB:Human)
EquivalentObjectProperties (ontA:hasChild ontB:child)
EquivalentDataProperties (ontA:hasAge ontB:age)

Podemos observar que nesse exemplo adicionamos os espaços de nomes ontA e ontB , pois a equivalência é comumente usada na integração/reuso de ontologias. A partir do momento que essas relações foram definidas, infere-se que as instâncias dessas entidades são equivalentes. O mesmo acontece quando temos duas instâncias diferentes, porém representando o mesmo indivíduo. Por exemplo, o ator Lima Duarte possui como nome de batismo Ariclenes Venâncio. Nesse caso, especifica-se em OWL como segue:

SameIndividual (:LimaDuarte :AriclenesVenancio)

A implicação do estabelecimento desse tipo de axioma é que tudo o que estiver relacionado com Ariclenes Venâncio está também relacionado com Lima Duarte. Esses axiomas de equivalência que foram supra apresentados não existem no RDF-S, o que já demonstra um aumento na expressividade da linguagem e consequentemente um maior poder de especificação de uma conceitualização.

Podemos além estabelecer que um indivíduo pode ser instância de diferentes classes (isto é, por meio da utilização de SubClassOf e SameIndividual), também, que os indivíduos não podem pertencer a mesma classe. Isso ocorre mediante a utilização de classes disjuntas (owl:DisjointClasses), como descrito abaixo:

DisjointClasses (:Man :Woman)

Outras noções disponíveis em OWL, que não estão disponíveis em RDF-S, são as entidades complexas que podem ser especificadas. Podemos criar novas entidades por meio da união de outras entidades. Por exemplo, abaixo criamos a entidade "pais" (< Parents>) , sendo formada pela união das entidades “mãe” (< Mother >) e “pai” (< Father >).

EquivalentClasses (

:Parent
ObjectUnionOf (:Mother :Father)
)

Estes são alguns dos conceitos que estão presentes na sintaxe de OWL e que não estão em RDF-S. Outros elementos que estão presentes em OWL 2 são descritos na tabela abaixo.

Tabela 3.2. Exemplos de outros elementos presentes na Owl 2.

Elemento	Propósito	Utilização
DifferentIndividuals	Especificar diferentes instâncias	DifferentIndividuals (:John :Mary)
ObjectIntersectionOf	Especifica nova classe através da intersecção de outras	EquivalentClasses (:Mother ObjectIntersectionOf(:Woman :Parent))
ObjectComplementOf	Especifica nova classe através da diferença entre duas classes	EquivalentClasses (:ChildlessPerson ObjectIntersectionOf (:Person :ObjectComplementOf (:Parent)))
ObjectSomeValuesFrom	Quantificador Existencial descreve que, para todos os indivíduos de uma classe, há pelo menos um indivíduo de outra classe relacionada por determinada propriedade	EquivalentClasses (:Parent ObjectSomeValuesFrom (:hasChild :Person))
ObjectAllValuesFrom	Quantificador Universal descreve uma classe de indivíduos, na qual todos os indivíduos relacionados devem ser de uma determinada classe	EquivalentClasses (:HappyPerson ObjectAllValuesFrom

		(:hasChild :HappyPerson))
ObjectHasValue	Restrição que especifica que uma classe de indivíduos se relaciona com um indivíduo em particular	EquivalentClasses (:JohnsChildren ObjectHasValues (:hasParent :John))
ObjectHasSelf	Restrição que especifica que um indivíduo se relaciona com ele mesmo	EquivalentClasses (:NarcisisticPerson ObjectHasSelf (:loves))
ObjectMaxCardinality	Restrição que descreve o número máximo de indivíduos de determinada relação	ClassAssertion (ObjectMaxCardinality (32 :hasTeams :Team) :FIFAWorldCup)
ObjectMinCardinality	Restrição que descreve o número mínimo de indivíduos de determinada relação	ClassAssertion (ObjectMinCardinality (1 :hasHostCountry :Country) :FIFAWorldCup)
ObjectExactCardinality	Restrição que descreve o número exato de indivíduos de determinada relação	ClassAssertion (ObjectExactCardinality (2 :hasTeams :Team) :FIFAWorldCupGame)
ObjectOneOf	Descreve uma classe com todas as suas instâncias	EquivalentClasses (

		:WorldCupGroupA ObjectOneOf (:Brazil :Croatia :Mexico :Camaroon))
InverseObjectProperties	Descreve que, se a classe A está relacionada com B por propriedade X, implica que a classe B está relacionada com A por propriedade Y	InverseObjectProperties (:hasParent :hasChild)
SymmetricObjectProperty	Descreve que, se a classe A está relacionada com B por propriedade X, implica que a classe B está relacionada com A da mesma forma	SymmetricObjectProperty (:hasSpouse)
ReflexiveObjectProperty	Descreve que determinada classe se relacionada através de determinada propriedade com ela mesma	ReflexiveObjectProperty (:hasRelative)
FunctionalObjectProperty	Descreve que determinado indivíduo possui apenas uma instância de determinada classe numa relação	FunctionalObjectProperty (:hasHusband)
HasKey	Restrição que especifica uma chave para determinada instância de uma classe	HasKey (:Person () (:hasSSN))

Muitas das implicações da utilização desses novos elementos são semânticas, podendo aumentar a capacidade de inferência e novas derivações a partir do que foi especificado. A próxima subseção descreve a semântica da linguagem OWL.

3.3.2 Semântica da linguagem OWL

Esta subseção tem por objetivo descrever a semântica por trás da linguagem OWL, mais especificamente abordando o poder de inferência da OWL. A Figura 3.6 apresenta como a linguagem OWL é constituída e como a lógica formal compõe a linguagem.

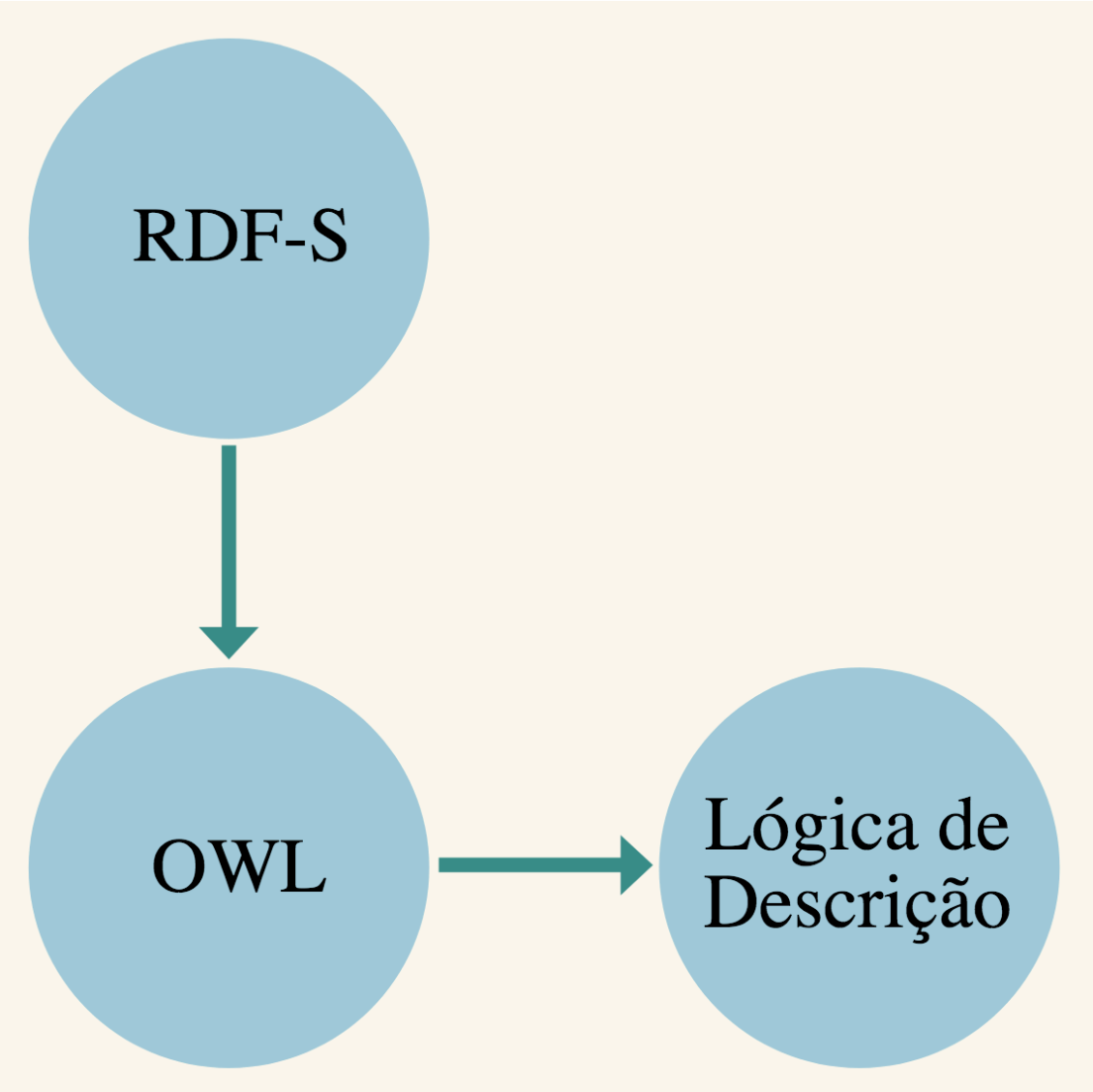


Figura 3.6 Relacionamento da Semântica Formal Na Linguagem Owl

Podemos observar que a OWL faz uso de construtores presentes na lógica de descrição. A lógica de descrição equivale a um formalismo de representação de conhecimento baseado em lógica, tendo sido remanescente das redes semânticas. Diferentemente de lógica de primeira ordem, a lógica de descrição está interessada em um procedimento que garanta que as respostas (positivas ou negativas) sejam dadas em tempo finito (Lutz et al., 2005).

De fato, como pode ser visto na Figura 3.4, há duas formas alternativas de especificar semântica em OWL:

- Semântica Direta (ou OWL 2 DL): a Semântica Direta (do inglês Direct Semantics) provê significado para as ontologias em OWL através da Lógica de Descrição. É por esta razão que a semântica direta é também associada à OWL 2 DL, onde DL tem o significado de Description Logic.
- Semântica baseada em RDF (ou OWL 2 FULL): a Semântica Baseada em RDF (do inglês RDF-Based Semantics) é uma extensão da semântica presente no Esquema RDF e é utilizada para a visualização de grafos RDF.

De um modo geral, podemos enxergar a semântica presente em OWL como OWL 2 DL ou OWL 2 FULL, onde algumas diferenças podem ser identificadas:

- OWL 2 DL possui um estilo baseada na Lógica de Descrição e a OWL 2 FULL possui um estilo baseado em Grafos RDF.
- OWL 2 DL está relacionada com a Semântica Direta enquanto OWL 2 FULL com Semântica baseada em RDF
- OWL 2 DL é uma versão mais simplificada e restrita que OWL 2 FULL.
- OWL 2 DL (Semântica Direta) é decidível (ou seja, responde a uma pergunta em tempo finito) enquanto que OWL 2 FULL (Semântica baseada em RDF) é indecidível.

É importante frisar que, na versão 1.0 da OWL, havia as propostas de OWL 1 DL e OWL 1 FULL, bem como a proposta OWL 1 Lite, que, de um modo geral, pode ser entendida como uma versão simplificada da OWL 1 DL.

No entanto novos conceitos importantes para a semântica foram adicionados para a versão 2.0 da OWL. A grande diferença está na definição de perfis para OWL 2, tendo sido adicionados principalmente pela complexidade de implementação de OWL. Com isso, os perfis foram projetados para atender a necessidades de uso e capacidade computacional e, como consequência, possuem diferentes níveis de expressividade. A Figura 3.7 apresenta os diferentes níveis de expressividade da linguagem OWL e uma comparação de suas versões (Open Semantic Framework, 2014) .

Figura 3.7. Níveis de expressividade da linguagem OWL

De acordo com a Figura 3.7, foram especificados três perfis para a versão 2.0 da OWL, sendo eles (Motik et al., 2014):

- OWL 2 EL: esse perfil é voltado para trabalhar com a família EL (Baader et al., 2005) , da lógica de descrição. Desta forma, OWL 2 EL pode ser usada para especificar classes complexas e axiomas sobre elas. Esse perfil foi projetado para sistemas de alta complexidade e necessidade de maior poder de

expressividade. Além disso, apesar da OWL 2 EL ser independente de domínio, ela foi concebida para atender a domínios com um grande número de classes e que possuam objetos estruturalmente complexos, como configuração de sistemas, inventário de produtos, estruturas biológicas e muitos domínios científicos. No entanto, esse perfil não permite a utilização de construtores como negação (owl:ObjectComplementOf) e disjunção (owl:DisjointClasses, quantificadores universais (owl:ObjectAllValuesFrom) sobre as propriedades e propriedades inversas (owl:InverseObjectProperties) . Dessa forma, a construção abaixo ("todos os filhos de uma pessoa rica são ricos") não pode ser especificada.

```
EquivalentClasses (
:RichPerson
ObjectAllValuesFrom ( :hasChild :RichPerson )
)
```

Por outro lado, a família de linguagens EL provê principalmente a utilização de quantificadores existenciais. Sendo assim, o exemplo abaixo ("os pais têm pelo menos um filho") é permitido.

```
EquivalentClasses (
:Parent
ObjectSomeValuesFrom ( :hasChild :Person )
)
```

- OWL 2 QL: esse perfil é voltado para o trabalho com bancos de dados relacionais tradicionais (isto é, que fazem uso da linguagem SQL). Como consequência, trabalhar com OWL 2 QL permite à equipe os benefícios providos pelos Sistemas Gerenciadores de Bancos de Dados Relacionais, como uma implementação robusta e características multiusuários. Ou seja, o OWL 2 QL é um perfil independente de domínio, que foi projetado para permitir a especificação de esquemas de bancos de dados e integração via consultas. Esse perfil pertence à família DL-Lite , podendo, dessa forma, reescrever consultas em SQL (Calvanese et al., 2007) . Nesse perfil, além dos construtores de negação e disjunção, não é permitida a utilização de quantificadores existenciais em classes complexas (apenas de classes simples), bem como axiomas para encadeamento de propriedades e igualdade. Dessa forma, a construção abaixo ("toda pessoa possui um pai que é mulher") não pode ser especificada, pois mulher, que é um dos parentes (sendo parente como um dos pais), é uma classe complexa.

```
EquivalentClasses (
:WomenParent
ObjectSomeValuesFrom ( :hasParent :Person )
)
```

Assim, o exemplo abaixo ("toda pessoa possui um pai") é permitido.

```
EquivalentClasses (
:Parent
ObjectSomeValuesFrom ( :hasParent :Person )
```

)

- OWL 2 RL: esse perfil é voltado para aplicações que necessitem de raciocínio de forma escalável. Além disso, esse perfil suporta tanto a semântica direta quanto a semântica baseada em RDF. Dessa forma, OWL 2 RL é o perfil ideal para o enriquecimento de dados especificados e conectados via RDF, pois pode ser implementado usando famílias de linguagens de regras (Grosof et al., 2003) , como por meio da utilização de RIF (do inglês Rule Interchange Format). Conclui-se então que, para aplicações voltadas para Dados Abertos Conectados, o perfil OWL 2 RL é mais adequado. Ou seja, diferentemente dos outros dois perfis, este perfil é mais adequado quando os projetistas já possuem uma modelagem em RDF e estão trabalhando com RDF. No entanto OWL 2 RL não permite declarações em que a existência de um indivíduo implica na existência de outro. Dessa forma, a construção abaixo (“toda pessoa possui um pai”) não pode ser especificada, pois a existência de uma pessoa implica na existência de um pai.

EquivalentClasses (
:Person
ObjectSomeValuesFrom (:hasParent :Parent))
Por outro lado, o exemplo abaixo (“a propriedade tio é definida pela relação entre o pai e o irmão do pai”) é permitido.
SubPropertyOf (
ObjectPropertyChain (:hasFather :hasBrother)
:hasUncle
)

Por fim, a Figura 3.8 apresenta parte da semântica baseada em RDF, que pode ser usada no perfil OWL 2 RL (Schneider, 2012) .

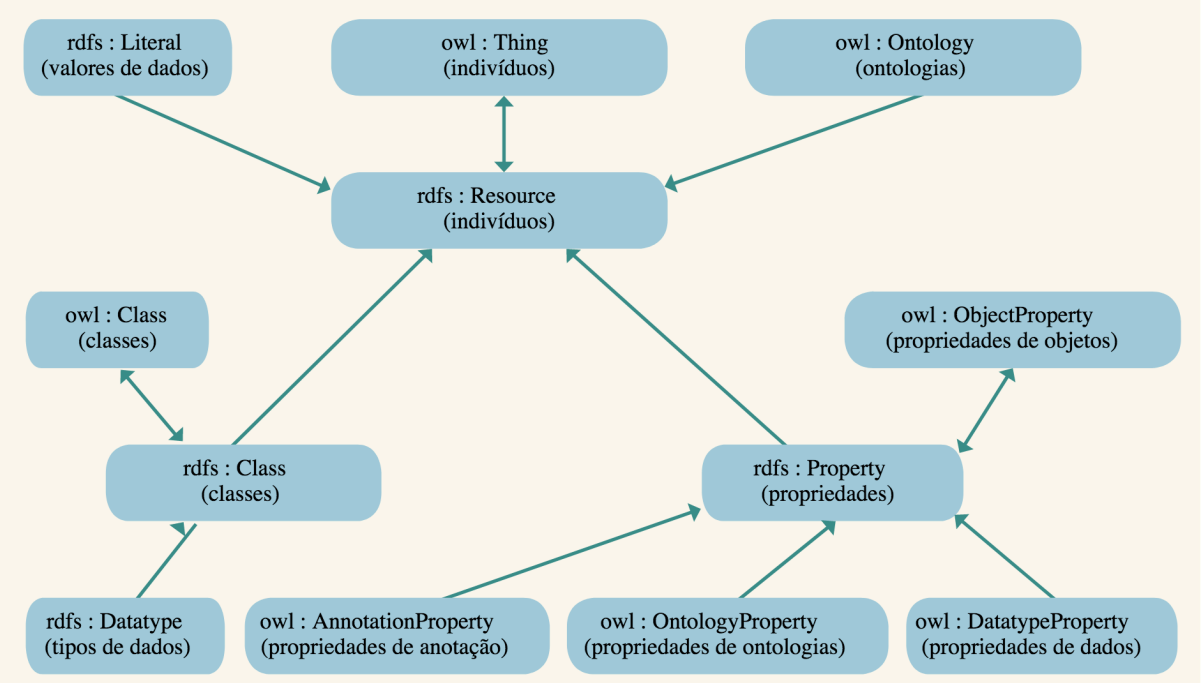


Figura 3.8. Semântica baseada em RDF.

4. Considerações Finais

O principal objetivo deste capítulo foi oferecer ao leitor uma visão geral sobre ontologias e representação de conhecimento na Web. Este capítulo foi organizado por meio de exemplos com o intuito de trazer para o leitor, pouco acostumado com o conceito de ontologias, uma perspectiva incremental de descrição semântica e de forma simplificada. Foi também intenção deste capítulo apresentar as linguagens utilizadas no contexto da Web para a descrição semântica através de ontologias. Esperamos que as seguintes mensagens tenham sido passadas:

- Compreensão sobre o conceito de ontologias e sua importância para a Web;
- Conhecimento sobre os diferentes tipos de ontologias e padrões da Web para descrição semântica;
- Entendimento sobre as características e diferenças presentes em RDF, RDF-S e OWL;
- Compreensão sobre a aplicação de OWL e RDF-S.