Capítulo 1

Semântica e Multimídia: Uma Introdução à Inteligência Artificial Simbólica Aplicada à Multimídia

Guilherme Lima[1]
Rodrigo Costa[1]
Marcio Ferreira Moreno[1]
[1]IBM Research, Brazil

Guilherme.Lima@ibm.com,Rodrigo.Costa@ibm.com,mmoreno@br.ibm.com

Abstract

In this chapter, we give an introduction to symbolic artificial intelligence (AI) from first principles and discuss its relation and application to multimedia. We begin by defining what symbolic AI is, what distinguishes it from non-symbolic approaches, such as machine learning, and how it can used in the construction of advanced multimedia applications. We then introduce description logic (DL) and use it to discuss symbolic representation and reasoning. DL is the logical underpinning of OWL, the most successful family of ontology languages. After discussing DL, we present OWL and related Semantic Web technologies, such as RDF and SPARQL. We conclude the chapter by discussing a hybrid model for multimedia representation, called Hyperknowledge. Throughout the chapter, we strive to make references to technologies and extensions specifically designed to solve the kinds of problems that arise in multimedia representation.

Resumo

Neste capítulo apresentamos uma introdução à inteligência artificial (IA) simbólica a partir de seus princípios básicos e discutimos a sua relação com e aplicação à multimídia. Começamos definindo o que é IA simbólica, o que a distingue de abordagens não-simbólicas, como o aprendizado de máquina, e como ela pode ser utilizada para construir aplicações multimídia avançadas. Em seguida, apresentamos a lógica de descrição (DL) e a usamos para discutir a representação e o raciocínio simbólico. DL é a lógica que fundamenta OWL, a mais bem sucedida família de linguagens ontológicas. Após DL, apresentamos OWL e as tecnologias de Web Semântica relacionadas, como RDF e SPARQL. Concluímos o capítulo apresentando um modelo híbrido para representação de multimídia, chamado Hyperknowledge. Durante todo o capítulo, procuramos fazer referência às tecnologias e extensões especificamente projetadas para resolver os tipos de problemas que aparecem no contexto de representação multimídia.

1.1. Introdução

Um problema clássico em representação e compreensão de dados multimídia é o problema do *gap* (vão) semântico. Ele afirma que há uma grande distância de representação entre os sinais audiovisuais que compõem os objetos multimídia e os conceitos representados por esses sinais. Por exemplo, a cor dominante e a trajetória de movimento de um dado conjunto de pixels em um vídeo, ambas características de baixo nível do vídeo, em geral não fornecem informações suficientes sobre o *significado* desse conjunto de pixels—pelo menos não para os computadores. Mas avanços recentes em inteligência artificial (IA) estão mudando isso.

Métodos de aprendizado de máquina atuais, apoiados por vastos conjuntos de dados de treinamento, são capazes de extrapolar padrões complexos a partir de dados multimídia de baixo nível. Esses padrões são materializados em modelos treinados que podem ser usados para identificar pessoas e objetos rapidamente e com precisão razoável em imagens, trechos de áudio, e em menor escala trechos de vídeo. Porém, a identificação de pessoas e objetos em dados multimídia resolve apenas metade do prolema. Para emular a cognição humana e realmente entender uma cena—por exemplo, para determinar quem está fazendo o que e as consequências dessas ações—os computadores precisam de informações adicionais: eles precisam de conhecimento geral de mundo e de conhecimento de domínio, e também precisam da capacidade de inferir conhecimento novo a partir de conhecimento preexistente. E aí que entra a inteligência artificial simbólica.

A ideia básica de IA simbólica é descrever o mundo, suas entidades e seus relacionamentos através de uma linguagem formal—uma linguagem que pode ser convenientemente manipulada por computadores—e desenvolver algoritmos eficientes para consultar e deduzir coisas a partir dessas descrições formais. Para ilustrar o tipo de aplicação que a combinação entre IA simbólica e multimídia possibilita considere a Figura 1.1.

Suponha que essa figura nos foi apresentada e suponha que a única coisa que sabemos sobre ela é o que podemos inferir a partir da imagem. Podemos ver que ela retrata Jean-Paul Marat (assumindo que sabemos identificá-lo), um ferimento semelhante ao causado por uma punhalada, uma faca com manchas de sangue, e uma carta endereçada a Marat e assinada por Charlotte Corday (assumindo que podemos ler o conteúdo da carta). A analogia aqui é que extraímos essas informações—ou *fatos*—através do reconhecimento de padrões visuais presentes na imagem. Apesar desses fatos básicos nos permitirem realizar tarefas computacionais simples, como a classificação e busca de imagens indexadas por palavras-chave, eles não são suficientes para entendermos a imagem.

Para realmente *entendermos* o que a Figura 1.1 retrata precisamos ir além dos fatos básicos. Precisamos (1) de conhecimento geral de mundo, (2) de conhecimento específico sobre as pessoas mencionadas, e (3) da capacidade de combinar esses conhecimentos gerais e específicos com os fatos extraídos da imagem e a partir disso inferir novos fatos.

Suponha que (1), (2) e (3) nos sejam dados. A partir do nosso conhecimento geral de mundo, e talvez de uma nova análise mais aprofundada da imagem, podemos afirmar com confiança que Marat está segurando a carta e que ele possui um ferimento no tórax. Disso e da faca manchada de sangue que aparece abaixo de Marat, podemos concluir que é provável que a faca retratada seja o objeto que causou o ferimento. Como facas não são seres autônomos, podemos também concluir que alguém (talvez o próprio Marat)

Figura 1.1. *A Morte de Marat* (detalhe), por Jacques-Louis David, 1793. (WikiMedia)

esfaqueou-o no peito. Mas quem e por quê?

Para responder a essas perguntas vamos precisar de mais informações. Suponha que nos digam que Marat foi um jornalista e agitador político, e um dos líderes de uma facção política radical durante o período do Terror da Revolução Francesa (c. 1793). Suponha também que nos digam que Charlotte Corday, que assina a carta, foi uma inimiga declarada de Marat—ela o culpava por diversos assassinatos em Paris e outra cidades e o considerava uma grave ameaça à República Francesa. À luz desses novos fatos, podemos concluir que a Figura 1.1 parece retratar um assassinato político.

Ao combinarmos essa conclusão com o fato adicional de que é sabido que Charlotte Corday assassinou Jean-Paul Marat com uma faca enquanto ele estava na banheira segurando uma carta assinada por ela, podemos inferir com um alto grau de confiança que a Figura 1.1 se trata de uma representação gráfica desse incidente, isto é, do assassinato por motivações políticas de Jean-Paul Marat por Charlotte Corday.

A derivação desse último fato a partir dos padrões visuais da Figura 1.1 só foi possível porque tivemos acesso não só a fatos básicos extraídos da imagem, mas também a fatos gerais sobre o mundo (conhecimento de senso comum) e sobre os objetos e pessoas retratadas (conhecimento de domínio), e também porque pudemos de combinar todos esses fatos e fazer inferências.

Um dos principais objetivos de IA simbólica é permitir a representação e manipulação de pedaços de conhecimento por computadores de formas que se assemelhem ou emulem os tipos de manipulação realizados por pessoas—manipulações similares às considerações que nos levaram a determinar o verdadeiro significado da Figura 1.1. A combinação dessa capacidade com multimídia abre muitas possibilidades. O exemplo do assassinato de Marat é uma instância de aplicação de compreensão automática de imagens. Duas aplicações relacionadas são a compreensão de vídeo e de áudio, normalmente mais complexas por envolverem a extração de informação temporal.

Outras aplicações que misturam IA simbólica e multimídia incluem a recuperação, classificação, recomendação e inspeção semântica de dados multimídia—por exemplo, para detectar automaticamente atividades suspeitas em imagens de vigilância, gerar classificação indicativa de músicas e filmes, e identificar fatores de risco associados à doenças em imagens médicas. Poderíamos listar muitas outras aplicações interessantes, algumas das quais vamos discutir mais adiante, mas nosso foco aqui é outro. Estamos mais interessados em apresentar os princípios e tecnologias que possibilitam a concepção dessas aplicações em primeiro lugar.

Neste capítulo apresentamos uma introdução à IA simbólica a partir da perspectiva de multimídia. Começamos apresentando a lógica de descrição (*Description Logic*, ou DL) na Seção 1.2. O motivo de começarmos com a lógica de descrição é que ela nos permite discutir a representação simbólica num contexto abstrato, livre das complicações de uma tecnologia específica. Outro motivo é que a lógica de descrição é em si uma tecnologia prática: ela é a lógica que fundamenta a família mais expressiva de linguagens ontológicas, a família OWL.

Após apresentarmos a lógica de descrição, na seção seguinte (Seção 1.3), tratamos da principal manifestação de IA simbólica na Web, a chamada *Web Semântica*. Discutimos a visão da Web Semântica, as tecnologias por trás dessa visão (RDF, OWL, SPARQL, SWRL, etc.) e a relação entre essas tecnologias e as noções apresentadas na Seção 1.2. Em ambas as seções, sempre que possível motivamos a discussão com exemplos do domínio de multimídia. Também sempre que possível apresentamos ou mencionamos metodologias e extensões projetadas especificamente para resolver os tipos de problemas que aparecem no contexto de representação multimídia.

Concluímos o capítulo com a discussão de um modelo híbrido para representação multimídia (Seção 1.4). Esse modelo, chamado *Hyperknowledge* (HK), generaliza um modelo clássico de hipermídia com as noções de nós de conceitos e elos semânticos. Ao fazer isso, o Hyperknowledge permite a representação e o processamento integrados de conteúdo multimídia junto com a sua descrição semântica.

Sugestões de leitura são apresentadas no fim do capítulo (Seção 1.5).

1.2. Lógica(s) de Descrição

A palavra lógica possui muitos significados e nenhuma definição completamente satisfatória. Podemos entender lógica como um sistema para dedução de proposições a partir de outras proposições afirmadas anteriormente. Existem muitos tipos de lógicas. Uma das mais simples é a lógica proposicional, que trata de proposições construídas a partir dos conectivos proposicionais $\neg$ (não), $\wedge$ (e), $\vee$ (ou), $\rightarrow$ (se-então) e $\leftrightarrow$ (se e somente se). A lógica de primeira ordem, utilizada normalmente para formalizar proposições matemáticas, estende a lógica proposicional com as noções de variáveis, constantes, funções, predicados e quantificadores. Há ainda outras extensões, por exemplo, a lógica de segunda ordem, e lógicas que adotam uma abordagem ligeiramente diferente, como as lógicas modais e *fuzzy*.

O termo "lógica de descrição" (DL) não se refere a uma lógica específica mas sim a uma *família* de lógicas. Por esse motivo às vezes falamos das lógica*s* de descrição (no plural). As lógicas dessa família apresentam uma grande variedade entre si,

mas a maioria delas compartilha as mesmas características de modelagem—elas tratam de indivíduos, conceitos e papéis. Neste capítulo, vamos focar na lógica de descrição $\mathcal{SROIQ}$ [Horrocks et al. 2006]. A principais DLs atuais são sublinguagens de $\mathcal{SROIQ}$, que é também a base da linguagem de ontologia OWL 2 DL [W3C OWL WG 2012]. Vamos seguir a maneira usual de se apresentar uma linguagem formal. Começamos apresentando a forma (sintaxe) das proposições de $\mathcal{SROIQ}$ juntamente com o seu significado intuitivo. Mais tarde definiremos o significado exato (semântica) dessas proposições.

1.2.1. Sintaxe

Sejam N_I, N_C e N_R três conjuntos de nomes contendo, respectivamente, nomes de indivíduos, nomes de conceitos e nomes de papéis (*roles*). A ideia aqui é que indivíduos representam entidades do mundo real, conceitos representam características dessas entidades e papéis representam relações entre (duas) entidades.[1] Vamos assumir que os conjuntos N_I, N_C e N_R são disjuntos par a par, isto é, que nenhum nome ocorre ao mesmo tempo em mais de um deles. Esses três conjuntos de nomes constituem o *vocabulário* de $\mathcal{SROIQ}$. Eles contêm os blocos básicos que usaremos para construir as proposições da linguagem.

Vamos distinguir três tipos de proposições, também chamadas em DL de *axiomas*, as quais agruparemos em caixas (*boxes*) correspondentes: ABox, TBox e RBox. A última, RBox, está disponível apenas em lógicas de descrição bastante expressivas, como $\mathcal{SROIQ}$, mas todas as ontologias baseadas em DL possuem uma TBox e a maioria delas uma ABox. Juntas essas três caixas formam uma *base de conhecimento* (*knowledge base*).

A ABox contém o *conhecimento assertivo*: proposições fazem afirmações sobre indivíduos específicos. Por exemplo, a assertiva de conceito

$$\texttt{Diretor}(\texttt{kubrick})$$

afirma que o indivíduo cujo nome é `kubrick` é um diretor. De maneira similar, a assertiva de papel

$$\texttt{dirige}(\texttt{kubrick}, \texttt{space-odyssey})$$

afirma que o indivíduo cujo nome é `kubrick` dirige o indivíduo (isto é, o filme) chamado `space-odyssey`.

A TBox e a RBox contém o *conhecimento terminológico*: proposições que fazem afirmações que se aplicam a todos os indivíduos. A TBox contém proposições que tratam de conceitos e a RBox contém proposições que dizem respeito a papéis. Por exemplo, o axioma de TBox

$$\texttt{Diretor} \sqsubseteq \texttt{Pessoa}$$

afirma que o conceito chamado `Diretor` é subsumido pelo conceito chamado `Pessoa`, isto é, que todo indivíduo que é um diretor é também uma pessoa. Já o axioma de RBox

$$\texttt{dirige} \sqsubseteq \texttt{gosta}$$

[1] Caso o leitor esteja familiarizado com lógica de primeira ordem: nomes de indivíduos são constantes, nomes conceitos são predicados unários e nomes de papéis são predicados binários.

afirma que o papel chamado `dirige` é subsumido pelo papel chamado `gosta`, isto é, que dirigir um filme implica em gostar desse filme.[2]

Essa última afirmação é obviamente questionável. Talvez algum diretor ou diretora em algum lugar do mundo não goste de algum filme que ele ou ela tenha dirigido. No entanto, o ponto aqui é que as proposições em uma base de conhecimento, especialmente aquelas na TBox e RBox, formalizam uma *conceitualização* particular de um domínio. Essa conceitualização reflete a visão de mundo do seu projetista e pode conter simplificações que fazem sentido no seu contexto de uso.

Outra propriedade de uma conceitualização é que ela é independente de linguagem. A conceitualização é apenas um modelo conceitual que pode ser criado em uma linguagem de modelagem qualquer, como UML. O termo *ontologia* é utilizado para se referir a uma instanciação de uma conceitualização em uma linguagem de representação de conhecimento específica. Para os nossos propósitos, o termo ontologia significa o mesmo que base de conhecimento, isto é, um conjunto de axiomas particionados em ABox, TBox e RBox.

De volta à $\mathcal{SROIQ}$, vamos agora definir a sintaxe exata das proposições de cada uma das caixas.

RBox A RBox de $\mathcal{SROIQ}$ contém proposições que descrevem interdependências entre papéis e características de papéis. Um *papel* é uma das seguintes expressões: (1) um nome de papel $\mathtt{r}$, para algum $\mathtt{r}$ em N_R; (2) um nome de papel invertido $\mathtt{r}^-$, para algum $\mathtt{r}$ em N_R; ou (3) o papel universal u.

Um *axioma de inclusão de papel* (*role inclusion axiom*, ou RIA) é uma proposição da forma

$$r_1 \circ \cdots \circ r_n \sqsubseteq r,$$

em que $r_1, \ldots, r_n$ e r são papéis.

Considere o seguinte RIA:

$$\mathtt{paiDe} \sqsubseteq \mathtt{filhoDe}^- .$$

Esse RIA afirma que o papel chamado `paiDe` é subsumido pelo inverso do papel chamado `filhoDe`, isto é, que se a é pai de b então b é filho a, para quaisquer indivíduos a e b. De maneira similar, os RIAs

$$\mathtt{donoDe} \sqsubseteq \mathtt{cuidaDe} \quad \text{e} \quad \mathtt{donoDe} \circ \mathtt{parteDe} \sqsubseteq \mathtt{donoDe}$$

afirmam, respectivamente, que se a é dono de b então a cuida de b (propriedade implica em cuidado) e que se a é dono de b e b é parte de c então a é dono de c (propriedade da parte implica em propriedade do todo).[3] Um conjunto finito de RIAs é chamado de *hierarquia de papéis*.

[2]Podemos reescrever as duas últimas proposições em lógica de primeira ordem da seguinte forma: $\forall x(\mathtt{Diretor}(x) \rightarrow \mathtt{Pessoa}(x))$ e $\forall x \forall y(\mathtt{dirige}(x,y) \rightarrow \mathtt{gosta}(x,y))$. A notação de DL é inspirada na notação de teoria dos conjuntos e não faz uso de variáveis.

[3]Esses três RIAs podem ser formalizados em lógica de primeira ordem como $\forall x \forall y(\mathtt{paiDe}(x,y) \rightarrow \mathtt{filhoDe}(y,x))$, $\forall x \forall y(\mathtt{donoDe}(x,y) \rightarrow \mathtt{cuidaDe}(x,y))$, e $\forall x \forall y \forall z(\mathtt{donoDe}(x,y) \wedge \mathtt{parteDe}(y,z) \rightarrow \mathtt{donoDe}(x,z))$.

Uma *característica de papel* é uma proposição com uma das seguintes formas:

$$\mathsf{Sym}(r), \quad \mathsf{Asy}(r), \quad \mathsf{Tra}(r), \quad \mathsf{Ref}(r), \quad \mathsf{Irr}(r), \quad \mathsf{Dis}(r,s),$$

em que r e s são quaisquer papéis diferentes do papel universal u. Conforme veremos na Seção 1.2.2, papéis representam relações binárias e características de papéis afirmam que essas relações possuem determinadas propriedades—a saber, simetria (Sym), assimetria (Asy), transitividade (Tra), reflexividade (Ref), irreflexividade (Irr), e disjunção (Dis). Vamos definir o significado exato dessas expressões na próxima seção.

Uma *RBox* de $\mathcal{SROIQ}$ consiste de uma hierarquia de papéis juntamente com um conjunto finito de características de papéis. Uma RBox é dita *regular* se a sua hierarquia de papéis é regular.

Regularidade é uma propriedade desejável porque ela garante a decidibilidade da lógica resultante. Na prática, isso significa que qualquer tarefa de raciocínio terminará em tempo finito (não entrará num laço infinito). Não vamos definir o que exatamente é uma RBox regular, já que essa definição nos levaria para fora do escopo desse capítulo, mas ressaltamos que a regularidade é uma propriedade puramente sintática: ela é obtida restringindo-se a forma dos RIAs que ocorrem numa dada hierarquia de papéis. Veja [Horrocks et al. 2006] para mais detalhes.

TBox A TBox de $\mathcal{SROIQ}$ contém proposições que relacionam conceitos, ou mais precisamente, expressões de conceitos. Uma *expressão de conceito* é uma das seguintes expressões:

1. Um nome de conceito $\mathtt{C}$, para algum $\mathtt{C}$ em N_C.
2. O conceito *top* $\top$.
3. O conceito *bottom* $\bot$.
4. Um *nominal* $\{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$ em que $\mathtt{a}_1, \ldots, \mathtt{a}_n$ são nomes de indivíduos em N_I.
5. Uma *negação* $\neg C$ em que C é uma expressão de conceito.
6. Uma *interseção* $C \sqcap D$ em que C e D são expressões de conceitos.
7. Uma *união* $C \sqcup D$ em que C e D são expressões de conceitos.
8. Um *existencial* $\exists r.C$ em que r é um papel e C é uma expressão de conceito.
9. Um *universal* $\forall r.C$ em que r é um papel e C é uma expressão de conceito.
10. Uma *autorrestrição* $\exists r.\mathsf{Self}$ em que r é um papel (simples).
11. Uma *restrição de cardinalidade* $\geq nr.C$ em que n é um inteiro não-negativo, r é um papel (simples), e C é uma expressão de conceito.
12. Uma *restrição de cardinalidade* $\leq nr.C$ em que n é um inteiro não-negativo, r é um papel (simples), e C é uma expressão de conceito.

Observe que uma expressão de conceito, conforme definido acima, não é uma proposição—ela não faz uma afirmação que pode ser verdadeira ou falsa. Uma expressão de conceito *constrói* um novo conceito a partir de outras expressões, incluindo outras expressões de conceitos, nomes de indivíduos ou papéis. E essa construção sempre começa com conceitos atômicos, isto é, nomes de conceitos ou os conceitos *top* $\top$ e *bottom* $\bot$ (conforme as regras 1, 2 e 3 acima).

Lembre-se que um conceito representa uma determinada característica de certos indivíduos. Podemos entender um conceito C como a coleção (ou *conjunto*) de todos os indivíduos que possuem essa determinada característica. Dessa forma, o nome de conceito `Ator` pode ser entendido como o conjunto de todos os indivíduos que são atores. Similarmente, o conceito *top* $\top$ pode ser entendido como o conjunto de *todos* os indivíduos (sem qualificação) e o conceito *bottom* $\bot$ pode ser entendido como o conjunto vazio—o conjunto contendo nenhum indivíduo.

O conceito nominal $\{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$ (regra 4) representa o conjunto contendo precisamente os indivíduos chamados $\mathtt{a}_1$, ..., $\mathtt{a}_n$. Por exemplo, $\{\mathtt{kubrick}, \mathtt{scorsese}\}$ representa o conjunto cujos membros são exatamente os indivíduos chamados `kubrick` e `scorsese`.

As operações de negação, interseção e união de conceitos (regras 5, 6 e 7) nos permitem construir novos conceitos a partir de outros conceitos. Por exemplo, o conceito $\neg\mathtt{Ator}$ representa o conjunto de todos os indivíduos que não são atores. De maneira similar, os conceitos

$$\mathtt{Ator} \sqcap \mathtt{Mulher} \quad \text{e} \quad \mathtt{Ator} \sqcup \mathtt{Diretor}$$

representam, respectivamente, o conjunto dos indivíduos que são ao mesmo tempo atores e mulheres, e o conjunto dos indivíduos que são atores ou diretores (ou ambos). Observe que essas operações podem ser aplicadas a quaisquer expressões de conceitos e não apenas a nomes de conceitos. Por exemplo, o conceito complexo

$$\mathtt{Diretor} \sqcap \neg(\mathtt{Mulher} \sqcup \{\mathtt{kubrick}, \mathtt{scorsese}\})$$

representa os diretores que não são nem mulheres nem os indivíduos chamados `kubrick` ou `scorsese`.

Os quantificadores existencial e universal (regras 8 e 9) recebem um papel e um conceito e constroem um novo conceito. Por exemplo, os conceitos

$$\exists\mathtt{conhece}.\mathtt{Ator} \quad \text{e} \quad \forall\mathtt{dirige}.\{\mathtt{a\text{-}bronx\text{-}tale}, \mathtt{good\text{-}shepherd}\}$$

representam, respectivamente, o conjunto dos indivíduos que conhecem *algum* ator e o conjunto dos indivíduos que ou não dirigiram nenhum filme ou dirigiram exatamente os filmes chamados `a-bronx-tale` e `good-shepherd`. Conforme veremos na próxima seção, pela maneira como o operador $\forall$ é definido, se quisermos excluir a parte "não dirigiram nenhum filme" do significado da expressão anterior precisamos escrever

$$(\exists\mathtt{dirige}.\top) \sqcap (\forall\mathtt{dirige}.\{\mathtt{a\text{-}bronx\text{-}tale}, \mathtt{good\text{-}shepherd}\}),$$

que, a propósito, especifica o conjunto $\{\mathtt{de\text{-}niro}\}$.

O operador de autorrestrição (regra 10) recebe um papel (simples) r e constrói o conceito contendo todos os indivíduos que estão relacionados a si mesmos via r. Por

exemplo, o conceito $\exists \texttt{ama.Self}$ representa o conjunto de todos os indivíduos que amam a si mesmos. Evidentemente, esse operador só faz sentido quando aplicado a papéis que conectam o mesmo tipo de entidade, por exemplo, pessoas à pessoas, filmes a filmes, etc.

O requisito de que r é um papel *simples* na definição do operador de autorrestrição está relacionado ao problema da regularidade mencionado anteriormente. Os papéis de uma RBox $\mathcal{SROIQ}$ são classificados em simples ou não-simples dependendo da forma dos RIAs em que eles ocorrem. Informalmente, os papéis não-simples são aqueles "criados" por cadeias de papéis de comprimento maior do que um; os papéis restantes são considerados simples. Vamos assumir que os papéis r e s que ocorrem em características de papéis da forma $\mathsf{Irr}(r)$, $\mathsf{Asy}(r)$ e $\mathsf{Dis}(r,s)$ são sempre papéis simples. Veja [Horrocks et al. 2006] para mais detalhes.

Os dois construtores de conceitos $\mathcal{SROIQ}$ restantes são os operadores de restrição de cardinalidade $\geq$ e $\leq$ (regras 11 e 12). Ambos recebem um inteiro não-negativo n, um papel simples r e um conceito C, e constroem um novo conceito contendo todos os indivíduos que estão r-conectados a pelo menos/no máximo n indivíduos em C. Por exemplo, os conceitos

$$\geq 2\texttt{conhece.Ator} \quad \text{e} \quad \leq 5\texttt{dirige.}\top$$

representam, respectivamente, o conjunto dos indivíduos que conhecem dois ou mais atores e o conjunto dos indivíduos que dirigiram cinco ou menos filmes.

Agora que definimos o formato das expressões de conceito de $\mathcal{SROIQ}$ podemos definir o que é uma TBox. Uma *TBox* de $\mathcal{SROIQ}$ é um conjunto finito de *axiomas gerais de inclusão de conceitos* (*general concept inclusion axioms*, ou GCIs) em que cada GCI é uma proposição da forma

$$C \sqsubseteq D,$$

para conceitos quaisquer C e D. Assim como nos RIAs, nos GCIs o símbolo $\sqsubseteq$ representa subsunção. Por exemplo, os GCIs

$$\texttt{Ator} \sqsubseteq \texttt{Pessoa} \quad \text{e} \quad \exists\texttt{dirige.}\top \sqsubseteq \exists\texttt{conhece.Ator}$$

afirmam, respectivamente, que todo indivíduo que é um ator é também uma pessoa e que todo indivíduo que dirige algum filme conhece algum ator.[4]

ABox A ABox de $\mathcal{SROIQ}$ contém as *proposições assertivas*, isto é, as proposições que tratam de indivíduos específicos. Essas proposições possuem uma das seguintes formas:

$$C(\texttt{a}), \quad r(\texttt{a},\texttt{b}), \quad \neg r(\texttt{a},\texttt{b}), \quad \texttt{a} \approx \texttt{b}, \quad \texttt{a} \not\approx \texttt{b},$$

em que C é um conceito, r é um papel, e $\texttt{a}$ e $\texttt{b}$ são nomes de indivíduos.

Seguem alguns exemplos de proposições assertivas:

[4]Para traduzirmos um GCI para lógica de primeira ordem primeiro transformamos os conceitos que ocorrem à esquerda e à direta do símbolo $\sqsubseteq$ em fórmulas com uma única variável livre x. Em seguida, conectamos essas duas fórmulas através de uma implicação ($\rightarrow$). Por último, prefixamos à fórmula resultante o quantificador $\forall x$. Os dois últimos GCIs podem ser traduzidos como:

$$\forall x(\texttt{Ator}(x) \rightarrow \texttt{Pessoa}(x)) \quad \text{e} \quad \forall x(\exists y(\texttt{dirige}(x,y) \wedge \top(y)) \rightarrow \exists z(\texttt{conhece}(x,z) \wedge \texttt{Ator}(z))).$$

- A *assertiva de conceito* $\texttt{Ator} \sqcap \texttt{Diretor}(\texttt{de-niro})$ afirma que o indivíduo chamado `de-niro` é uma *instância* do conceito chamado $\texttt{Ator} \sqcap \texttt{Diretor}$, isto é, que esse indivíduo é ao mesmo tempo ator e diretor.
- A *assertiva de papel* $\texttt{dirige}(\texttt{kubrick}, \texttt{dr-strangelove})$ afirma que o indivíduo chamado `kubrick` está `dirige`-conectado ao indivíduo chamado `dr-strangelove`. De maneira similar, a *assertiva de papel negada* $\neg\, \texttt{dirige}(\texttt{kubrick}, \texttt{taxi-driver})$ afirma que `kubrick` não está `dirige`-conectado a `taxi-driver`.
- A *assertiva de igualdade* $\texttt{kubrick} \approx \texttt{stanley}$ e de *desigualdade* $\texttt{kubrick} \not\approx \texttt{taxi-driver}$ afirmam, respectivamente, que `kubrick` e `stanley` designam o mesmo indivíduo e que `kubrick` e `taxi-driver` não designam o mesmo indivíduo.

Base de conhecimento Uma *base de conhecimento* (*knowledge base*, ou KB) $\mathcal{SROIQ}$ consiste da união das três caixas: RBox, TBox e ABox. Uma base de conhecimento é dita regular se sua RBox é regular.

Vamos concluir esta seção sobre a sintaxe de $\mathcal{SROIQ}$ com um exemplo. A base de conhecimento abaixo contém alguns dos *fatos sobre filmes* discutidos anteriormente. Observe que a interpretação usual das proposições que ocorrem nessa base de conhecimento—aquela em que `kubrick` designa o famoso diretor—é apenas uma das muitas outras possíveis interpretações. Trataremos da noção de interpretação na próxima seção.

ABox	
$\texttt{Diretor}(\texttt{kubrick})$	"`kubrick` é um a diretor"
$\texttt{Ator} \sqcap \texttt{Diretor}(\texttt{de-niro})$	"`de-niro` é ao mesmo tempo ator e diretor"
$\geq 2\texttt{conhece}.\texttt{Ator}(\texttt{stanley})$	"`stanley` conhece ao menos dois atores"
$\texttt{dirige}(\texttt{kubrick}, \texttt{space-odyssey})$	"`kubrick` dirigiu `space-odyssey`"
$\neg\, \texttt{dirige}(\texttt{kubrick}, \texttt{taxi-driver})$	"`kubrick` não dirigiu `taxi-driver`"
$\texttt{kubrick} \approx \texttt{stanley}$	"`kubrick` e `stanley` são o mesmo indivíduo"
$\texttt{kubrick} \not\approx \texttt{de-niro}$	"`kubrick` e `de-niro` não são o mesmo indivíduo"
TBox	
$\texttt{Diretor} \sqsubseteq \exists \texttt{dirige}.\top$	"todo diretor dirige algo"
$\exists \texttt{dirige}.\top \sqsubseteq \texttt{Diretor}$	"tudo que dirige algo é um diretor"
$\texttt{Diretor} \sqsubseteq \texttt{Pessoa}$	"todo diretor é uma pessoa"
$\texttt{Diretor} \sqsubseteq \geq 1\texttt{conhece}.\texttt{Ator}$	"todo diretor conhece ao menos um ator"
$(\exists \texttt{dirige}.\top) \sqcap (\forall \texttt{dirige}.\{\texttt{a-bronx-tale}, \texttt{good-shepherd}\}) \sqsubseteq \{\texttt{de-niro}\}$	"tudo que dirige exatamente $\{\texttt{a-bronx-tale}, \texttt{good-shepherd}\}$ é um $\{\texttt{de-niro}\}$"
RBox	
$\texttt{dirige} \sqsubseteq \texttt{gosta}$	"dirigir implica em gostar"
$\texttt{dirige} \circ \texttt{atuaEm}^{-} \sqsubseteq \texttt{conhece}$	"se x dirige y em que z atua, então x conhece z"

1.2.2. Semântica

Uma base de conhecimento, conforme definimos anteriormente, é basicamente um conjunto de *strings* bem-formadas as quais chamamos de proposições ou axiomas. O significado

de cada uma dessas *strings* depende, em última instância, do significado dos nomes que ocorrem em cada uma delas. Para ver isso, considere o axioma $\texttt{Diretor}(\texttt{kubrick})$. Esse axioma será verdadeiro se interpretarmos os nomes `Diretor` e `kubrick` como "a propriedade de ser um diretor de cinema" e "Stanley Kubrick", respectivamente. Porém, se interpretarmos `Diretor` como "a propriedade de ser um número par" e `kubrick` como "o número 37", então a proposição $\texttt{Diretor}(\texttt{kubrick})$ deixa de ser verdadeira. Esse exemplo é artificial mas serve para ilustrar o nosso ponto de que o significado de uma proposição deriva da interpretação dos nomes. Vamos agora definir precisamente a noção de interpretação.

Interpretação Uma *interpretação* $\mathcal{I}$ consiste de duas coisas:

1. Um conjunto não-vazio $\Delta^{\mathcal{I}}$ chamado de *domínio* ou *universo* de discurso.
2. Uma função $\square^{\mathcal{I}}$ partindo dos conjuntos de vocabulário, N_I, N_C e N_R, tal que:[5]
   (a) Se $\mathtt{a} \in \mathsf{N}_I$ então $\mathtt{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
   (b) Se $\mathtt{C} \in \mathsf{N}_C$ então $\mathtt{C}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.
   (c) Se $\mathtt{r} \in N_R$ então $\mathtt{r}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Em outras palavras, uma interpretação $\mathcal{I}$ fixa um domínio de discurso $\Delta^{\mathcal{I}}$, que pode ser qualquer conjunto não-vazio, e mapeia os nomes de invidados em elementos do domínio, os nomes de conceitos em subconjuntos do domínio, e os nomes de papéis em relações binárias no domínio.

Vamos reservar o termo *indivíduo* para nos referirmos à contrapartida semântica de um nome de indivíduo, isto é, a um elemento do domínio. E vamos reservar os termos *extensão de conceito* e *extensão de papel* para nos referirmos, respectivamente, às contrapartidas semânticas de conceitos e papéis, isto é, às relações unárias e binárias no domínio.

Vamos agora estender a definição da função interpretação $\square^{\mathcal{I}}$ a papéis e conceitos:

$$u^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \tag{1}$$
$$(r^{-})^{\mathcal{I}} = \{\langle \delta', \delta \rangle \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}}\} \tag{2}$$
$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \tag{3}$$
$$\bot^{\mathcal{I}} = \varnothing \tag{4}$$
$$\{\mathtt{a}_1, \ldots, \mathtt{a}_n\}^{\mathcal{I}} = \{\mathtt{a}_1^{\mathcal{I}}, \ldots, \mathtt{a}_n^{\mathcal{I}}\} \tag{5}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C \tag{6}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \tag{7}$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \tag{8}$$
$$(\exists r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \text{ e } \delta' \in C^{\mathcal{I}}, \text{ para algum } \delta' \in \Delta^{\mathcal{I}}\} \tag{9}$$

[5] Observe que $x^{\mathcal{I}}$ é apenas uma forma mais compacta de escrever $\mathcal{I}(x)$ que é a notação usual de aplicação de função.

$$(\forall r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \text{se } \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \text{ então } \delta' \in C^{\mathcal{I}}, \text{ para todo } \delta' \in \Delta^{\mathcal{I}}\} \quad (10)$$

$$(\exists r.\mathsf{Self})^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta \rangle \in r^{\mathcal{I}}\} \quad (11)$$

$$(\geq n r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \#\{\delta' \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \text{ e } \delta' \in C\} \geq n\} \quad (12)$$

$$(\leq n r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \#\{\delta' \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \text{ e } \delta' \in C\} \leq n\} \quad (13)$$

Essas equações podem ser lidas da seguinte forma:

1. O *papel universal* u representa (denota) a relação que conecta quaisquer dois indivíduos do domínio.

2. O *papel invertido* r^{-} denota a mesma relação que r porém com a primeira e segunda coordenadas trocadas.

3. O *conceito top* $\top$ denota o domínio como um todo (conjunto $\Delta^{\mathcal{I}}$).

4. O *conceito bottom* $\bot$ denota o conjunto vazio ($\varnothing$).

5. O *nominal* $\{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$ denota o conjunto contendo exatamente os indivíduos denotados pelos nomes $\mathtt{a}_1$, ..., $\mathtt{a}_n$.

6. A *negação* $\neg C$ denota a diferença entre o domínio e o conjunto denotado por C, isto é, o conjunto de todos os indivíduos em $\Delta^{\mathcal{I}}$ que não estão em $C^{\mathcal{I}}$.

7. A *interseção* $C \sqcap D$ denota a interseção dos conjuntos denotados por C e D, isto é, o conjunto de todos os indivíduos que pertencem a ambos $C^{\mathcal{I}}$ e $D^{\mathcal{I}}$.

8. A *união* $C \sqcup D$ denota a união dos conjuntos denotados por C e D, isto é, o conjunto de todos os indivíduos que pertencem a ao menos um dos conjuntos $C^{\mathcal{I}}$ ou $D^{\mathcal{I}}$.

9. O *existencial* $\exists r.C$ denota o conjunto de todos os indivíduos que estão conectados via a relação $r^{\mathcal{I}}$ a algum indivíduo em $C^{\mathcal{I}}$.

10. O *universal* $\forall r.C$ denota o conjunto de todos os indivíduos a tais que se a está conectado a algum indivíduo b via $r^{\mathcal{I}}$ então b está em $C^{\mathcal{I}}$. Uma consequência do formato condicional dessa definição é que qualquer indivíduo que não esteja conectado a nenhum outro via $r^{\mathcal{I}}$ estará automaticamente no conjunto resultante (pois a implicação é vacuamente verdadeira).[6]

11. A *autorrestrição* $\exists r.\mathsf{Self}$ denota o conjunto de todos os indivíduos que estão conectados a si mesmos via a relação $r^{\mathcal{I}}$.

12. A *restrição de cardinalidade* $\geq n.rC$ denota o conjunto de todos os indivíduos que estão conectados via a relação $r^{\mathcal{I}}$ a pelo menos n indivíduos em $C^{\mathcal{I}}$.

13. A *restrição de cardinalidade* $\leq n.rC$ denota o conjunto de todos os indivíduos que estão conectados via relação $r^{\mathcal{I}}$ a no máximo n indivíduos em $C^{\mathcal{I}}$.

Vamos agora tratar do problema de definir o valor-verdade (verdadeiro ou falso) de um axioma sob uma dada interpretação.

Satisfatibilidade Uma interpretação $\mathcal{I}$ *satisfaz* (ou *é modelo de*) um axioma α, em símbolos $\mathcal{I} \vDash \alpha$, sob as seguintes condições:

1. (RIA) $\mathcal{I} \vDash r_1 \circ \cdots \circ r_n \sqsubseteq r$ sse[7] para qualquer sequência δ_0, δ_1, ..., δ_n em $\Delta^{\mathcal{I}}$:

   $$\langle \delta_0, \delta_1 \rangle \in r_1^{\mathcal{I}},\ \langle \delta_1, \delta_2 \rangle \in r_2^{\mathcal{I}},\ \ldots,\ \text{e}\ \langle \delta_{n-1}, \delta_n \rangle \in r_n^{\mathcal{I}} \quad \text{implica em} \quad \langle \delta_0, \delta_n \rangle \in r^{\mathcal{I}},$$

   ou seja, sse para qualquer caminho em $\Delta^{\mathcal{I}}$ que atravessa $r_1^{\mathcal{I}}$, ..., $r_n^{\mathcal{I}}$ (nessa ordem) há um atalho (aresta) via $r^{\mathcal{I}}$.

2. (Simetria de papel) $\mathcal{I} \vDash \mathsf{Sym}(r)$ sse $r^{\mathcal{I}}$ é uma relação simétrica, isto é, sse $\langle \delta_1, \delta_2 \rangle \in r^{\mathcal{I}}$ implica em $\langle \delta_2, \delta_1 \rangle \in r^{\mathcal{I}}$ e vice-versa, para quaisquer δ_1, δ_2 em $\Delta^{\mathcal{I}}$.

3. (Assimetria de papel) $\mathcal{I} \vDash \mathsf{Asy}(r)$ sse $r^{\mathcal{I}}$ é uma relação assimétrica, isto é, sse $\langle \delta_1, \delta_2 \rangle \in r^{\mathcal{I}}$ implica em $\langle \delta_2, \delta_1 \rangle \notin r^{\mathcal{I}}$, para quaisquer δ_1, δ_2 em $\Delta^{\mathcal{I}}$.

4. (Transitividade de papel) $\mathcal{I} \vDash \mathsf{Tra}(r)$ sse $r^{\mathcal{I}}$ é uma relação transitiva, isto é, sse $\langle \delta_1, \delta_2 \rangle \in r^{\mathcal{I}}$ e $\langle \delta_2, \delta_3 \rangle \in r^{\mathcal{I}}$ implicam em $\langle \delta_1, \delta_3 \rangle \in r^{\mathcal{I}}$, para quaisquer $\delta_1, \delta_2, \delta_3$ em $\Delta^{\mathcal{I}}$.

5. (Reflexividade de papel) $\mathcal{I} \vDash \mathsf{Ref}(r)$ sse $r^{\mathcal{I}}$ é uma relação reflexiva, isto é, sse $\langle \delta, \delta \rangle \in r^{\mathcal{I}}$, para qualquer δ em $\Delta^{\mathcal{I}}$.

6. (Irreflexividade de papel) $\mathcal{I} \vDash \mathsf{Irr}(r)$ sse $r^{\mathcal{I}}$ é uma relação irreflexiva, isto é, sse $\langle \delta, \delta \rangle \notin r^{\mathcal{I}}$, qualquer δ em $\Delta^{\mathcal{I}}$.

7. (Disjunção de papéis) $\mathcal{I} \vDash \mathsf{Dis}(r, s)$ sse $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \varnothing$, isto é, sse não existem dois indivíduos ao mesmo tempo $r^{\mathcal{I}}$-conectados e $s^{\mathcal{I}}$-conectados.

8. (GCI) $\mathcal{I} \vDash C \sqsubseteq D$ sse $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, isto é, sse qualquer indivíduo de $C^{\mathcal{I}}$ é também um indivíduo de $D^{\mathcal{I}}$.

9. (Assertiva de conceito) $\mathcal{I} \vDash C(\mathtt{a})$ sse $\mathtt{a}^{\mathcal{I}} \in C^{\mathcal{I}}$, isto é, se o indivíduo denotado por $\mathtt{a}$ pertence à extensão do conceito C.

10. (Assertiva de papel) $\mathcal{I} \vDash r(\mathtt{a}, \mathtt{b})$ sse $\langle \mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$, isto é, sse $\mathtt{a}^{\mathcal{I}}$ está conectado a $\mathtt{b}^{\mathcal{I}}$ via $r^{\mathcal{I}}$.

11. (Assertiva de papel negada) $\mathcal{I} \vDash \neg r(\mathtt{a}, \mathtt{b})$ sse $\langle \mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}} \rangle \notin r^{\mathcal{I}}$.

12. (Assertiva de igualdade) $\mathcal{I} \vDash \mathtt{a} \approx \mathtt{b}$ sse $\mathtt{a}^{\mathcal{I}} = \mathtt{b}^{\mathcal{I}}$, isto é, sse $\mathtt{a}^{\mathcal{I}}$ e $\mathtt{b}^{\mathcal{I}}$ são o mesmo elemento de $\Delta^{\mathcal{I}}$.

[6] Em lógica clássica a proposição $A \rightarrow B$ é equivalente à proposição $(\neg A) \vee B$. Dessa forma, podemos reescrever a equação (10) da seguinte maneira:

$$(\forall r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \notin r^{\mathcal{I}} \text{ ou } \delta' \in C^{\mathcal{I}}, \text{ para todo } \delta' \in \Delta^{\mathcal{I}}\}.$$

Evidentemente, qualquer δ que não esteja r-conectado a algum δ' satisfaz a condição acima.

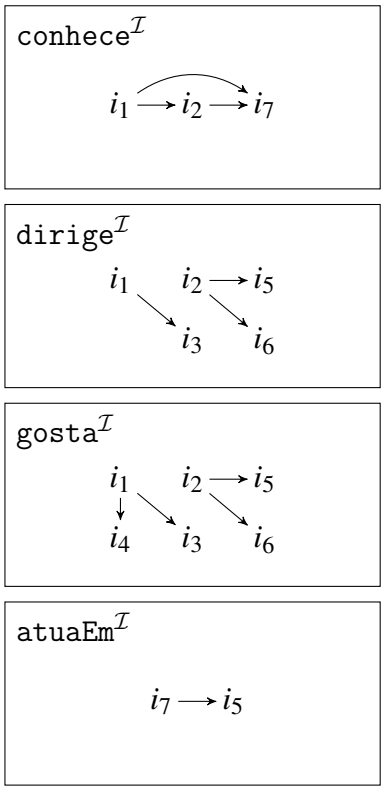
[7] Vamos usar "sse" como uma abreviação para "se e somente se".

13. (Assertiva de desigualdade) $\mathcal{I} \vDash \texttt{a} \not\approx \texttt{b}$ sse $\texttt{a}^{\mathcal{I}} \neq \texttt{b}^{\mathcal{I}}$.

Agora que definimos as condições necessárias para que uma interpretação satisfaça um dado axioma, podemos estender a noção de satisfatibilidade às bases de conhecimento (conjuntos de axiomas). Uma interpretação $\mathcal{I}$ *satisfaz* (ou *é modelo de*) uma base de conhecimento $\mathcal{KB}$, em símbolos $\mathcal{I} \vDash \mathcal{KB}$, sse $\mathcal{I} \vDash \alpha$, para qualquer axioma α em $\mathcal{KB}$. Uma base de conhecimento é dita *satisfatível* ou *consistente* se ela possui um modelo, ou seja, se existe alguma interpretação $\mathcal{I}$ que a satisfaz; caso contrário, a base de conhecimento é dita *insatisfatível* ou *inconsistente*.

Para tornar a discussão de interpretações mais concreta, considere a seguinte interpretação $\mathcal{I}$ para a KB de fatos sobre filmes apresentada no fim da seção anterior:

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{i_1, i_2, i_3, i_4, i_5, i_6, i_7\} \\
\texttt{kubrick}^{\mathcal{I}} &= i_1 \\
\texttt{de-niro}^{\mathcal{I}} &= i_2 \\
\texttt{stanley}^{\mathcal{I}} &= i_1 \\
\texttt{space-odyssey}^{\mathcal{I}} &= i_3 \\
\texttt{taxi-driver}^{\mathcal{I}} &= i_4 \\
\texttt{a-bronx-tale}^{\mathcal{I}} &= i_5 \\
\texttt{good-shepherd}^{\mathcal{I}} &= i_6 \\
\texttt{Diretor}^{\mathcal{I}} &= \{i_1, i_2\} \\
\texttt{Ator}^{\mathcal{I}} &= \{i_2, i_7\} \\
\texttt{Pessoa}^{\mathcal{I}} &= \{i_1, i_2\} \\
\texttt{conhece}^{\mathcal{I}} &= \{\langle i_1, i_2\rangle, \langle i_1, i_7\rangle, \langle i_2, i_7\rangle\} \\
\texttt{dirige}^{\mathcal{I}} &= \{\langle i_1, i_3\rangle, \langle i_2, i_5\rangle, \langle i_2, i_6\rangle\} \\
\texttt{gosta}^{\mathcal{I}} &= \{\langle i_1, i_3\rangle, \langle i_1, i_4\rangle, \langle i_2, i_5\rangle, \langle i_2, i_6\rangle\} \\
\texttt{atuaEm}^{\mathcal{I}} &= \{\langle i_7, i_5\rangle\}
\end{aligned}
$$



O domínio $\Delta^{\mathcal{I}}$ nesse caso consiste de sete indivíduos distintos, $i_1, \ldots, i_7$, e os nomes que ocorrem na KB de fatos sobre filmes são mapeados conforme as equações acima. Observe que:

- Alguns indivíduos do domínio $\Delta^{\mathcal{I}}$ podem não possuir nome correspondente: i_7 não é denotado por nenhum nome da KB de fatos sobre filmes.
- Dois nomes distintos podem designar o mesma objeto: `kubrick` e `stanley` denotam ambos o indivíduo i_1.
- Nomes de conceitos são mapeados em subconjuntos do domínio $\Delta^{\mathcal{I}}$: `Diretor`, `Ator` e `Pessoa` denotam conjuntos de indivíduos.

- Nomes de papéis são mapeados em relações binárias no domínio $\Delta^{\mathcal{I}}$: `conhece`, `dirige`, `gosta` e `atuaEm` denotam conjuntos de pares ordenados de indivíduos. Os indivíduos i_2 e i_5 estão relacionados via a relação $\texttt{dirige}^{\mathcal{I}}$ (nessa ordem) sse o par $\langle i_2, i_5 \rangle$ pertence ao conjunto $\texttt{dirige}^{\mathcal{I}}$. Relações binárias podem ser representadas convenientemente como grafos dirigidos. Os grafos correspondentes às relações $\texttt{conhece}^{\mathcal{I}}$, $\texttt{dirige}^{\mathcal{I}}$, $\texttt{gosta}^{\mathcal{I}}$ e $\texttt{atuaEm}^{\mathcal{I}}$ são apresentados nas caixas acima.

Seja $\mathcal{KB}$ a base de conhecimento de fatos sobre filmes e seja $\mathcal{I}$ a interpretação anterior. A pergunta natural a se fazer é se $\mathcal{I} \vDash \mathcal{KB}$, isto é, se a interpretação anterior satisfaz a KB. A resposta a essa pergunta será positiva apenas se $\mathcal{I} \vDash \alpha$ para todo axioma a em $\mathcal{KB}$. Vamos verificar se essa última afirmação procede.

Começamos verificando a ABox de $\mathcal{KB}$. Evidentemente, $\mathcal{I} \vDash \texttt{Diretor}(\texttt{kubrick})$, pois $\texttt{kubrick}^{\mathcal{I}} \in \texttt{Diretor}^{\mathcal{I}}$, isto é, pois $i_1 \in \{i_1, i_2\}$, conforme demandado pela definição de satisfatibilidade para assertivas de conceito. Além disso, temos que $\mathcal{I} \vDash \neg\texttt{dirige}(\texttt{kubrick}, \texttt{taxi-driver})$, pois $\langle \texttt{kubrick}^{\mathcal{I}}, \texttt{taxi-driver}^{\mathcal{I}} \rangle \notin \texttt{dirige}^{\mathcal{I}}$. Os outros axiomas da ABox de $\mathcal{KB}$ também são satisfeitos por $\mathcal{I}$, conforme o leitor pode facilmente verificar.

Vamos agora verificar a TBox de $\mathcal{KB}$. Temos que $\mathcal{I} \vDash \texttt{Diretor} \sqsubseteq \texttt{Pessoa}$, pois todo indivíduo do conjunto $\texttt{Diretor}^{\mathcal{I}}$ é também um indivíduo do conjunto $\texttt{Pessoa}^{\mathcal{I}}$. É também o caso que

$$\mathcal{I} \vDash (\exists \texttt{dirige}.\top) \sqcap (\forall \texttt{dirige}.\{\texttt{a-bronx-tale}, \texttt{good-shepherd}\}) \sqsubseteq \{\texttt{de-niro}\}$$

já que (1) $\texttt{de-niro}^{\mathcal{I}}$ está conectado via a relação $\texttt{dirige}^{\mathcal{I}}$ a algum membro de $\top^{\mathcal{I}}$ (isto é, a algum membro do domínio como um todo), (2) tudo a que $\texttt{de-niro}^{\mathcal{I}}$ está $\texttt{dirige}^{\mathcal{I}}$-conectado é um elemento do conjunto $\{\texttt{a-bronx-tale}^{\mathcal{I}}, \texttt{good-shepherd}^{\mathcal{I}}\}$ e (3) as duas últimas afirmações valem apenas para o indivíduo $\texttt{de-niro}^{\mathcal{I}}$. O restante dos axiomas da TBox de $\mathcal{KB}$ são também satisfeitos por $\mathcal{I}$. (Novamente, deixamos a verificação desses axiomas para o leitor.)

A última coisa que precisamos verificar são os axiomas da RBox de $\mathcal{KB}$. Temos que $\mathcal{I} \vDash \texttt{dirige} \sqsubseteq \texttt{gosta}$, pois todo par da relação $\texttt{dirige}^{\mathcal{I}}$ é também um par da relação $\texttt{gosta}^{\mathcal{I}}$. Uma forma de determinar se a proposição

$$\mathcal{I} \vDash \texttt{dirige} \circ \texttt{atuaEm}^{-} \sqsubseteq \texttt{conhece}$$

vale, é procurar por três indivíduos x, y e z, não necessariamente distintos, tais que $\langle x, y \rangle \in \texttt{dirige}^{\mathcal{I}}$ e $\langle z, y \rangle \in \texttt{atuaEm}^{\mathcal{I}}$ mas $\langle x, z \rangle \notin \texttt{conhece}^{\mathcal{I}}$. Se pudermos encontrar tais indivíduos então a proposição será falsa (pois acabamos de encontrar um contraexemplo). Caso não seja possível encontrar tais indivíduos, a proposição será verdadeira. No caso específico da interpretação $\mathcal{I}$, há apenas uma escolha possível que satisfaz os dois primeiros requisitos, a saber, $x = i_2$, $y = i_5$ e $z = i_7$. Observe que essa escolha também satisfaz o terceiro requisito, já que $\langle i_2, i_7 \rangle \in \texttt{conhece}^{\mathcal{I}}$. Logo, a proposição é verdadeira, isto é, o axioma $\texttt{dirige} \circ \texttt{atuaEm}^{-} \sqsubseteq \texttt{conhece}$ é satisfeito pela interpretação $\mathcal{I}$.

Como $\mathcal{I}$ satisfaz todos os axiomas em cada uma das caixas da KB de fatos sobre filmes, concluímos que a resposta à pergunta original (se $\mathcal{I} \vDash \mathcal{KB}$) é positiva. Ou seja,

a interpretação $\mathcal{I}$ satisfaz a KB de fatos sobre filmes e, consequentemente, essa KB é consistente (satisfatível).

Duas perguntas relacionadas à essa mesma KB são as seguintes: É possível encontrar uma interpretação $\mathcal{J}$ que não satisfaz a KB de fatos sobre filmes? Como podemos modificar essa KB de forma que ela se torne inconsistente (insatisfatível)?

Uma resposta simples à primeira pergunta consiste em considerar uma interpretação $\mathcal{J}$ idêntica à interpretação $\mathcal{I}$ anterior porém em que $\texttt{kubrick}^{\mathcal{J}}$ e $\texttt{stanley}^{\mathcal{J}}$ são mapeados em indivíduos distintos do domínio Δ^{J}. Evidentemente, sob essa interpretação o axioma de ABox $\texttt{kubrick} \approx \texttt{stanley}$ é falso; logo $\mathcal{J} \not\models \mathcal{KB}$.

Com respeito à segunda pergunta, para tornar a KB de fatos sobre filmes inconsistente precisamos modificá-la de forma que seja impossível construir qualquer interpretação que a satisfaça. Uma maneira trivial de fazer isso é adicionar à KB a assertiva de desigualdade $\texttt{kubrick} \not\approx \texttt{kubrick}$. Esse axioma será falso sob qualquer interpretação pois qualquer que seja a escolha de denotação para kubrick ela sempre será igual a si mesma, isto é, $\texttt{kubrick}^{\mathcal{I}} = \texttt{kubrick}^{\mathcal{I}}$. Evidentemente, existem formas menos triviais de obtermos uma KB inconsistente. (Será que o leitor consegue achar uma que envolva um axioma de TBox?)

Isso conclui a discussão sobre satisfatibilidade. O objetivo principal de uma semântica formal é definir uma *relação de consequência* que permita determinar quando um determinado axioma decorre de uma determinada base de conhecimento. Vamos agora definir essa relação de consequência em termos da relação de satisfatibilidade.

Consequência lógica Um axioma α é *consequência lógica* de (ou *é implicado logicamente por*) uma base de conhecimento $\mathcal{KB}$, em símbolos $\mathcal{KB} \models \alpha$, sse $\mathcal{I} \models \mathcal{KB}$ implica em $\mathcal{I} \models \alpha$, para qualquer interpretação $\mathcal{I}$. Isto é, sse qualquer interpretação que é modelo para a base de conhecimento $\mathcal{KB}$ é também modelo para o axioma α.[8]

Usando essa definição de consequência lógica podemos determinar precisamente quando um α decorre de um determinado conjunto de axiomas. Conforme discutimos anteriormente, algumas KBs são inconsistentes no sentido de que não existe interpretação que satisfaça todos os seus axiomas. O problema de se ter uma base de conhecimento inconsistente é que, por definição, ela implica em qualquer coisa. Isto é, se $\mathcal{KB}$ é inconsistente então a afirmação "$\mathcal{I} \models \mathcal{KB}$ implica em $\mathcal{I} \models \alpha$ para qualquer $\mathcal{I}$" é vacuamente verdadeira pois o antecedente da implicação, a saber, $\mathcal{I} \models \mathcal{KB}$, é falso. Na prática, a inconsistência de uma KB é um indicativo de erros sérios de modelagem.

Uma tarefa importante na teoria de DL é, portanto, determinar se uma determinada KB é consistente. Vamos discutir essa e outras ditas *tarefas de raciocínio* em detalhe na próxima seção. Antes disso, porém, vamos concluir essa seção com um exemplo de consequência lógica.

[8]Observe que o símbolo $\models$ possui significados diferentes dependendo do tipo de objeto que ocorre à sua esquerda. Quando esse objeto é uma interpretação, como em $\mathcal{I} \models \alpha$, o símbolo $\models$ denota a relação de satisfatibilidade. Porém, quando esse objeto é um conjunto de axiomas, como em $\mathcal{KB} \models \alpha$, o símbolo $\models$ denota a relação de consequência lógica.

Considere a seguinte proposição sobre a $\mathcal{KB}$ de fatos sobre filmes:

$$\mathcal{KB} \models \texttt{Ator} \sqsubseteq \texttt{Pessoa}.$$

Essa proposição afirma que o axioma $\texttt{Ator} \sqsubseteq \texttt{Pessoa}$ é consequência lógica do conjunto de axiomas que compõem a $\mathcal{KB}$. Essa afirmação é verdadeira? Como podemos prová-la ou refutá-la?

Vamos tentar refutá-la. O que precisamos fazer nesse caso é encontrar uma interpretação $\mathcal{I}$ que satisfaça todos os axiomas em $\mathcal{KB}$ mas que não satisfaça $\texttt{Ator} \sqsubseteq \texttt{Pessoa}$. Por exemplo, considere a interpretação $\mathcal{I}$ para $\mathcal{KB}$ discutida anteriormente, cujo domínio consiste dos indivíduos $i_1, \ldots, i_7$. Conforme verificamos há alguns parágrafos, $\mathcal{I} \models \mathcal{KB}$. Porém, não é o caso que todo ator é uma pessoa sob essa interpretação $\mathcal{I}$. Por exemplo, $i_7 \in \texttt{Ator}^{\mathcal{I}}$ mas $i_7 \notin \texttt{Pessoa}^{\mathcal{I}}$. Logo, $\mathcal{KB} \not\models \texttt{Ator} \sqsubseteq \texttt{Pessoa}$, ou seja, a $\mathcal{KB}$ de fatos sobre filmes não implica logicamente no axioma $\texttt{Ator} \sqsubseteq \texttt{Pessoa}$.

Normalmente é mais difícil provar o contrário, isto é, provar que a base de conhecimento implica logicamente em um determinado axioma α. Provar isso significa mostrar que é impossível encontrar uma interpretação que ao mesmo tempo satisfaz a base de conhecimento e falsifica α. Por exemplo, vamos mostrar que a $\mathcal{KB}$ de fatos sobre filmes implica logicamente no axioma

$$\exists \texttt{gosta}.\top(\texttt{de-niro}).$$

Em outras palavras, vamos mostrar que se assumirmos que os axiomas da $\mathcal{KB}$ são verdadeiros, somos obrigados a concluir que de-niro gosta de algo.

Suponha que exista uma interpretação $\mathcal{J}$ que satisfaça a $\mathcal{KB}$ de fato sobre filmes, mas que não satisfaça o axioma acima. Da hipótese de que $\mathcal{J} \models \mathcal{KB}$ podemos inferir que

$$\mathcal{J} \models \texttt{Ator} \sqcap \texttt{Diretor}(\texttt{de-niro}) \tag{14}$$

$$\mathcal{J} \models \texttt{Diretor} \sqsubseteq \exists \texttt{dirige}.\top \tag{15}$$

$$\mathcal{J} \models \texttt{diretor} \sqsubseteq \texttt{gosta} \tag{16}$$

Como (14), temos que o indivíduo denotado por de-niro está no conjunto $\texttt{Diretor}^{\mathcal{J}}$. Logo, como (15), esse indivíduo está $\texttt{dirige}^{\mathcal{J}}$-conectado a alguém, isto é, $\langle \texttt{de-niro}^{\mathcal{J}}, a \rangle \in \texttt{dirige}^{\mathcal{J}}$, para algum indivíduo a. Finalmente, como (16), temos que $\langle \texttt{de-niro}, a \rangle \in \texttt{gosta}^{\mathcal{J}}$, o que contradiz a nossa hipótese original de que $\mathcal{J} \not\models \exists \texttt{gosta}.\top(\texttt{de-niro})$. Portanto, somos forçados a concluir que tal $\mathcal{J}$ não pode existir. Ou seja, qualquer interpretação que satisfaça a $\mathcal{KB}$ de fatos sobre filmes também irá satisfazer $\exists \texttt{gosta}.\top(\texttt{de-niro})$, logo, $\mathcal{KB} \models \exists \texttt{gosta}.\top(\texttt{de-niro})$.

1.2.3. Tarefas de raciocínio

Uma das principais vantagens de se formalizar um corpo de conhecimento em lógica é a possibilidade de tratá-lo como um objeto tangível ao qual certas operações podem ser aplicadas. Em DL, esse objeto é a KB (conjunto de axiomas) e as operações são *tarefas de raciocínio* que visam extrair conhecimento novo a partir da KB.

Uma característica importante das tarefas de raciocínio de DL, que faz com que elas sejam ao mesmo tempo poderosas e complexas, é que essas tarefas adotam a *hipótese de*

mundo aberto (*open-world assumption*, ou OWA). Sob essa hipótese, fatos que não podem ser deduzidos a partir da KB são considerados *desconhecidos* mas não necessariamente falsos. Isso contrasta com a *hipótese de mundo fechado* (*closed-world assumption*, ou CWA), normalmente adotada em sistemas de bancos de dados, que assume que fatos que não estão no banco são falsos. Dito isso, quando necessário, é possível emular até certo ponto a CWA em DL através dos nominals. Há também DLs especializadas (autoepistêmicas, circumscriptivas) que dão suporte à CWA.

Vamos agora descrever rapidamente os principais tipos de tarefas de raciocínio de DL. Para uma descrição mais detalhada veja [Rudolph 2011, Baader et al. 2007]. Uma descrição abrangente da complexidade computacional dessas tarefas para DLs específicas pode ser encontrada em [Zolin 2019]. E uma lista de softwares que implementam algumas dessas tarefas pode ser encontrada em [OWL@Machester 2019]

Satisfatibilidade da base de conhecimento A principal tarefa de raciocínio de DL é a que determina satisfatibilidade da base de conhecimento. O objetivo dessa tarefa é decidir se uma dada KB é satisfatível, isto é, se existe uma interpretação que satisfaz todos os seus axiomas. Outras tarefas de raciocínio importantes, como a de consequência lógica, podem ser reduzidas à tarefa de satisfatibilidade.

Existem basicamente dois tipos de abordagens para verificar se uma KB é satisfatível. As abordagens *baseadas em teoria dos modelos* tentam construir um modelo para a KB ou então mostrar que tal construção é impossível. Já as abordagens *baseadas em teoria da prova* manipulam a KB sintaticamente até derivar uma contradição ou até atingir um ponto em que é garantido que uma contradição não pode ser derivada. Exemplos de métodos do primeiro tipo são os métodos baseados em autômatos de árvore e o método do tableaux. Como exemplos de métodos do segundo tipo podemos citar os métodos baseados em provas (derivações) e aqueles baseados em resolução.

Consequência lógica O objetivo da tarefa de consequência lógica é decidir se um dado axioma α é consequência lógica de uma dada KB. Esse problema se reduz diretamente ao problema de satisfatibilidade em DLs que suportam negação (como $\mathcal{SROIQ}$). Em DLs sem suporte à negação essa redução ainda é possível mas requer a simulação da negação através de alguma noção de oposição entre axiomas.

Satisfatibilidade e classificação de conceitos O objetivo da tarefa de satisfatibilidade (consistência) de conceito é decidir se um determinado conceito C é satisfatível com respeito a uma determinada base de conhecimento $\mathcal{KB}$. Isto é, se existe uma interpretação $\mathcal{I}$ que satisfaz $\mathcal{KB}$ e tal que $C^{\mathcal{I}}$ é populado (não-vazio). Observe que alguns conceitos, como $C \sqcap \neg C$, por definição não podem ser satisfeitos.

Uma tarefa relacionada é a classificação de conceitos. O objetivo dessa tarefa é decidir se os nomes de conceitos que ocorrem em uma KB podem ser organizados hierarquicamente de acordo com os seus relacionamentos de subsunção ($\sqsubseteq$). Essa organização hierárquica é normalmente um passo preliminar executados por outras tarefas de raciocínio e também ajuda na visualização de KBs contendo muitos GCIs.

Recuperação de instâncias designadas A tarefa de recuperação de instâncias designadas recebe uma base de conhecimento $\mathcal{KB}$ e um conceito C e retorna todos os indivíduos

designados a tais que $\mathcal{KB} \models C(\mathtt{a})$. Isto é, ela determina todos os nomes de indivíduos que são instâncias do conceito C em todos os modelos da $\mathcal{KB}$. Essa tarefa pode ser estendida a papéis; nesse caso a tarefa procura por pares de nomes de indivíduos $\langle \mathtt{a}, \mathtt{b} \rangle$ tais que $\mathcal{KB} \models r(\mathtt{a}, \mathtt{b})$.

Resposta a consultas O objetivo da tarefa de reposta a consultas é encontrar a solução para uma consulta q numa dada base de conhecimento $\mathcal{KB}$. A consulta q é normalmente especificada como uma proposição parcial com partes faltantes (variáveis livres) que devem ser preenchidas com nomes de indivíduos. Uma resposta à consulta q é uma solução (preenchimento específico) que torne q consequência lógica de $\mathcal{KB}$. Ou seja, uma solução é uma sequência de nomes de indivíduos que quando substituídos pelas variáveis livres de q fazem com que $\mathcal{KB} \models q$ seja verdadeiro. Uma extensão natural do problema é encontrar *todas* as possíveis respostas que satisfazem uma determinada consulta.

Indução e abdução O objetivo da tarefa de indução é, a partir dos axiomas assertivos da ABox de uma KB, gerar axiomas terminológicos de TBox e RBox correspondentes. Já o objetivo da tarefa de abdução é determinar, dadas uma base de conhecimento $\mathcal{KB}$ e um axioma α não implicado logicamente por $\mathcal{KB}$, quais axiomas (com determinadas características) precisam ser adicionados à $\mathcal{KB}$ para que $\mathcal{KB} \models \alpha$ se torne verdade. Note que diferentemente das tarefas anteriores, ambas indução e abdução são tarefas que não preservam verdade: os axiomas que elas podem ser falsificados.

1.2.4. Regras

Às vezes as tarefas de raciocínio discutidas acima não são poderosas o suficiente. Algumas aplicações precisam representar relações ou permitir derivações que vão além do que pode ser expresso e obtido em DL pura. Por exemplo, relações e derivações que envolvam computações arbitrárias, como àquelas que envolvem aritmética, ou a derivação de fatos apenas quando determinados outros fatos não podem ser derivados (inferências não-monotônicas).

Uma abordagem comum para aumentar a expressividade de uma DL é associar os axiomas da KB a um conjunto *regras* expressas em alguma linguagem de especificação de regras. Uma escolha popular de linguagem de regra são as cláusulas de Horn—um fragmento de lógica de primeira ordem que é também base para linguagens de programação lógicas, como Prolog e Datalog. Por exemplo, a linguagem SWRL (*Semantic Web Rule Language*) é essencialmente a combinação de Datalog com a linguagem de ontologia OWL. Outras combinações de DL com regras que aumentam a expressividade da DL (mas que nesse caso mantém a decidibilidade) são as linguagens DLP (*Description Logic Programs*) [Grosof et al. 2003] e DLR (*Description Logic Rules*) [Krötzsch et al. 2008]. Para mais detalhes sobre linguagens de regra veja [Hitzler et al. 2010].

1.2.5. Outras DLs

A família de DLs adota uma convenção que nos permite determinar as características de uma lógica a partir do seu nome. Essa convenção é dada pelo seguinte esquema:

$$((\mathcal{ALC} \mid \mathcal{S})\,[\mathcal{H}] \mid \mathcal{SR})\,[\mathcal{O}]\,[\mathcal{I}]\,[\mathcal{F} \mid \mathcal{N} \mid \mathcal{Q}],$$

Tabela 1.2. Características associadas às letras do nome de uma DL.

Símbolo	Características presentes/ausentes	exemplo
$\mathcal{ALC}$	Ausentes: axiomas de RBox, papel universal, papéis inversos, restrições de cardinalidade, nominals, autorrestrição.	$\neg(\texttt{Musica} \sqcup \texttt{Filme})$
$\mathcal{S}$	$\mathcal{ALC}$ com RIAs da forma $\texttt{r} \circ \texttt{r} \sqsubseteq \texttt{r}$, para r em N_R.	$\texttt{parteDe} \circ \texttt{parteDe} \sqsubseteq \texttt{parteDe}$
$\mathcal{H}$	$\mathcal{ALC}$ com RIAs da forma $r \sqsubseteq s$.	$\texttt{refilmagemDe} \sqsubseteq \texttt{baseadoEm}$
$\mathcal{SR}$	$\mathcal{ALC}$ com autorrestrição e todos os tipos de axiomas de RBox.	$\texttt{filhoDe} \circ \texttt{irmaoDe} \sqsubseteq \texttt{sobrinhoDe}$
$\mathcal{O}$	Presente: nominals.	$\{\texttt{universal}, \texttt{disney}\} \sqsubseteq \texttt{Estudio}$
$\mathcal{I}$	Presente: papéis inversos.	$\texttt{filhoDe} \sqsubseteq \texttt{paiDe}^{-}$
$\mathcal{F}$	Presente: axiomas de funcionalidade de papel (expressos como $\top \sqsubseteq \leq 1 r.\top$).	$\top \sqsubseteq \leq 1 \texttt{websiteOficial}.\top$
$\mathcal{N}$	Presente: restrições de cardinalidade da forma $\geq nr.\top$ e $\leq nr.\top$.	$\texttt{Serie} \sqsubseteq \geq 2.\texttt{possuiEpisodio}.\top$
$\mathcal{Q}$	Presente: todos os tipos de restrições de cardinalidade.	$\texttt{Serie} \sqsubseteq \geq 2.\texttt{possuiEpisodio}.\texttt{Episodio}$

em que os parênteses denotam grupos, a barra vertical representa uma escolha, e os colchetes delimitam os componentes opcionais. O significado de cada uma das letras acima é apresentado na Tabela 1.2.

Além de $\mathcal{SROIQ}$ outras lógicas de descrição populares são $\mathcal{ALC}$ e $\mathcal{SHOIQ}$. $\mathcal{ALC}$ [Schmidt-Schauß and Smolka 1991] é a menor DL com fechamento Booleano, isto é, em que os operadores Booleanos podem ser aplicados a conceitos de maneira irrestrita. $\mathcal{SHOIQ}$ [Baader et al. 2007] é uma lógica relacionada à linguagem de ontologia OWL DL (versão 1). Note que ambas $\mathcal{ALC}$ e $\mathcal{SHOIQ}$ são sublinguagens de $\mathcal{SROIQ}$.

Extensões espaciais e temporais de DL Em representação multimídia, é comum precisarmos descrever relações espaciais e temporais entre objetos. Por exemplo, durante o processo de anotar uma cena de um filme pode ser preciso especificar a posição relativa dos personagens na cena ou a sequência temporal das suas ações e diálogos.

Um formalismo simples porém expressivo para a especificação de relações espaciais 2D é o *Cálculo de Conexão de Regiões* (*Region Connection Calculus*, ou RCC8). O RCC8 [Randell et al. 1992] descreve oito tipos básicos de relações que podem se estabelecer entre duas regiões 2D: desconectadas (DC), externamente conectadas (EC), iguais (EQ), parcialmente sobrepostas (PO), parte própria tangencial (TPP), parte própria inversa (TPPi), parte própria não-tangencial (NTPP) e parte própria não-tangencial inversa (NTPPi). Essas oito relações são apresentadas na Figura 1.2. A família de lógicas de descrição espaciais $\mathcal{ALCI}_{\mathrm{RCC}}$ [Wessel 2003] contém lógicas especificamente projetadas para suportar as

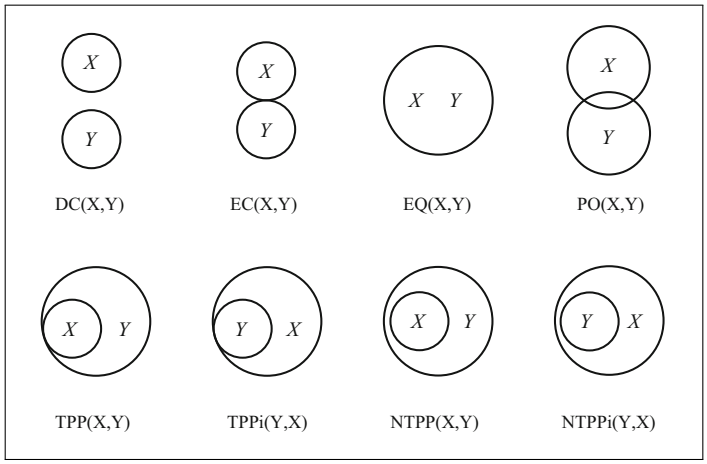


Figura 1.2. As relações do RCC8 [Sikos 2017].

relações do RCC8. Outras DLs que podem ser utilizadas para raciocínio espacial são as lógicas $\mathcal{ALC}(\mathrm{F})$ [Hudelot et al. 2015] e $\mathcal{ALC}(\mathrm{CDC})$ [Cristiani and Gabrielli 2011]. Essas últimas são propostas mais recentes que não se baseiam no RCC.

Para descrever fatos no tempo podemos utilizar uma das várias lógicas temporais disponíveis. A maioria dessas lógicas são extensões que adicionam à DL os operadores usuais de lógica temporal, isto é, operadores como $\Diamond$ (possivelmente no futuro) e $\Box$ (sempre no futuro). A lógica de descrição $\mathcal{ALC}$-LTL [Baader and Lippmann 2014] combina $\mathcal{ALC}$ com Lógica Temporal Linear (*Linear Temporal Logic*, ou LTL), que adota uma noção de tempo linear (sem ramificações) e é frequentemente utilizada na especificação de sistemas dinâmicos. Outra DL temporal é DL-CTL [Wang et al. 2014] que combina DL com *Computational Tree Logic* (CTL), uma lógica temporal com suporte a tempo não-linear (com ramificações). Há também uma extensão da lógica de descrição $\mathcal{SHIN}$ que quando usada em conjunto com OWL é chamada de tOWL [Milea et al. 2012]. tOWL permite a representação de pontos no tempo e de relações entre pontos e intervalos.

Extensões fuzzy de DL As informações extraídas de dados multimídia são frequentemente ambíguas e imprecisas. No entanto, as DLs tradicionais, por serem fragmentos de lógica de primeira ordem clássica, não cedem lugar a ambiguidade ou imprecisão—a lógica é Booleana: um dado fato é verdadeiro ou é falso. Isso significa que quando usamos as DLs tradicionais para representar dados multimídia é possível que alguma informação importante sobre incerteza seja perdida. Ou pior, podemos induzir um falso senso de certeza quando tal certeza não existe. Uma forma de amenizar esse problema é representar a incerteza na lógica.

As lógicas fuzzy são um tipo particular de lógica multi-valorada (não-Booleana) cuja semântica envolve a noção de *graus de verdade*. Os graus de verdade da lógica funcionam como possíveis "tons de cinza" entre os Booleanos verdadeiro e falso, e podem ser usados para formalizar indefinição ou imprecisão. Existem diversas extensões fuzzy de DL. Um bom ponto de partida para estudá-las é [Borgwardt and Peñaloza 2017].

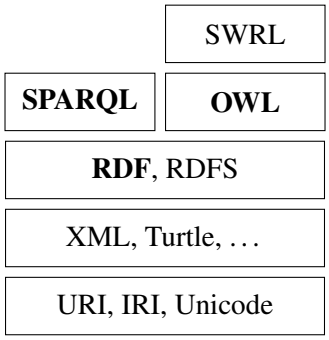


Figura 1.3. As camadas da Web Semântica.

1.3. Web Semântica

O termo *Web Semântica* diz respeito a um esforço iniciado no começo dos anos 2000 para estender a Web tradicional, então uma biblioteca global de documentos HTML interligados, para permitir a descrição e processamento semânticos desses documentos. Duas décadas depois, a Web se tornou um ambiente bem mais diverso e complexo, e o sucesso e viabilidade da visão original da Web Semântica é de certa forma contestável. O que não é contestável, porém, é o sucesso dos padrões e tecnologias desenvolvidas nesse período. Atualmente, o termo Web Semântica é mais utilizado para se referir à essas tecnologias e padrões.

Nesta seção, vamos descrever as principais tecnologias da Web Semântica. Essas tecnologias podem ser organizadas em camadas, conforme apresenta a Figura 1.3. As tecnologias que aparecem nas camadas superiores são definidas sobre aquelas que aparecem nas camadas inferiores. Vamos discutir praticamente todas as tecnologias que aparecem na figura, mas nosso foco será naquelas que aparecem em negrito, a saber, RDF, SPARQL e OWL, com ênfase em especial na última.

1.3.1. RDF

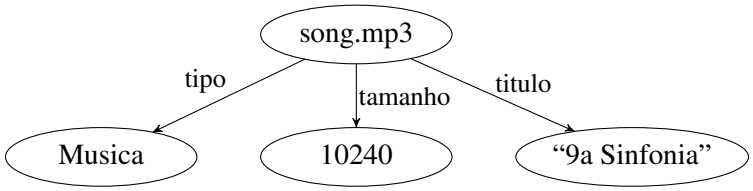
Um dos principais padrões da Web Semântica é o *Resource Description Framework* (RDF) [Wood et al. 2014]. O RDF é uma linguagem de representação baseada em grafos. Ele provê um vocabulário e construtores para a descrição de *conjuntos de triplas* que representam grafos dirigidos. Essas triplas triples chamadas de *triplas s-p-o* possuem o seguinte formato:

$$\langle sujeito, predicado, objeto \rangle.$$

As três triplas s-p-o abaixo descrevem, respectivamente, o tipo, tamanho em bytes e título associados ao arquivo de áudio “song.mp3”:

$$\langle \text{song.mp3}, \text{tipo}, \text{Musica} \rangle,\ \langle \text{song.mp3}, \text{tamanho}, 10240 \rangle,\ \langle \text{song.mp3}, \text{titulo}, \text{“9a Sinfonia”} \rangle.$$

Aqui está o grafo correspondente à essas triplas s-p-o:



A ideia é que cada tripla representa uma aresta dirigida no grafo. O sujeito da tripla determina o nó de origem da aresta, o objeto determina o nó de destino e o predicado determina o rótulo da aresta.

Evidentemente, o exemplo anterior é apenas um exemplo abstrato. Na prática, os componentes das triplas de um grafo RDF, isto é, os rótulos dos nós e arestas do grafo, devem seguir uma *sintaxe* específica. Uma sintaxe popular para RDF é a Turtle (*Terse RDF Triple Language*) [Carothers and Prudhommeaux 2014]. A sintaxe Turtle é uma alternativa à amplamente utilizada porém complicada sintaxe RDF/XML. Todos os nossos exemplos serão escritos em Turtle.

Há três tipos de nó em um grafo RDF: nós IRI, nós literais e nós em branco. Um *nó IRI* é um nó cujo rótulo é uma IRI (*Internationalized Resource Identifier*); uma IRI é uma generalização de URI com suporte à caracteres Unicode. Por exemplo, o endereço abaixo é uma IRI:

```
https://pt.wiktionary.org/wiki/maçã
```

Note os caracteres não-ASCII “ç” e “ã” acima. Em RDF, IRIs são usadas para se referir a *recursos* arbitrários, como objetos físicos, documentos, imagens, conceitos abstratos, números, *strings*, etc.

O segundo tipo de nó que ocorre num grafo RDF é o nó literal. Um *nó literal* é um nó rotulado por um literal, isto é, uma *string* que denota um valor. Na sintaxe Turtle, literais são escritos entre aspas duplas e opcionalmente acompanhados de *tags* de linguagem ou de tipo. Por exemplo, aqui estão três literais RDF (na sintaxe Turtle):

```
"Rock", "maçã"@pt, "10240"^^xsd:integer.
```

O primeiro é um literal simples, o segundo é um literal com *tag* de linguagem (Português no caso) e o terceiro é um literal com *tag* de tipo, também chamado de literal tipado. O prefixo `xsd:` que ocorre no terceiro literal denota um *namespace* apropriado que define a *tag* de tipo `integer`. Teremos mais a dizer sobre *namespaces* em instantes.

O terceiro tipo de nó que pode aparecer num grafo RDF é o nó em branco. Um *nó em branco* é um nó sem denotação específica; ele é um nó não-rotulado que não é nem nó IRI nem nó literal. Nós em branco são comumente utilizados como nós postiços (*dummies*) cujo único propósito é facilitar a codificação de estruturas complexas em grafos.

Numa tripla RDF $\langle s, p, o \rangle$ o sujeito s é necessariamente uma IRI ou nó em branco, o predicado p é necessariamente uma IRI e o objeto o é necessariamente uma IRI, literal ou nó em branco. Quando afirmamos uma determinada tripla RDF, estamos afirmando informalmente que “é válido o relacionamento indicado pelo predicado entre o recurso denotado pelo sujeito e o recurso ou valor denotado pelo objeto”.

A Figura 1.4 apresenta uma possível versão Turtle do grafo que descreve o arquivo de áudio "song.mp3" apresentado há alguns parágrafos.[9] Cada aresta do grafo (isto é, tripla do conjunto) corresponde a uma tripla no documento Turtle (linhas 5–7). Em Turtle, as partes do sujeito, predicado e objeto são separadas por espaços e cada tripla é terminada por um ponto (`.`). Observe que as IRIs aparecem entre os caracteres `<...>`.

```
1 @base <http://example.org/data/> .
2 @prefix ex: <http://example.org/#> .
3 @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
4 <song.mp3> ex:tipo ex:Musica .
5 <song.mp3> ex:tamanho "10240"^^xsd:integer .
6 <song.mp3> ex:titulo "9a Sinfonia" .
```

Figura 1.4. Representação em Turtle das triplas RDF que descrevem "song.mp3".

As diretivas que ocorrem nas linhas 1–3 da Figura 1.4 definem os espaços de nomes (*namespaces*) do documento. Isto é, elas especificam como IRIs relativas ou prefixadas devem ser interpretadas. A diretiva `@base` define a IRI base de todo o documento Turtle. Qualquer IRI relativa que ocorra no documento será interpretada como relativa à essa IRI base. Logo, por exemplo, as ocorrências de `<song.mp3>` nas linhas 4–6 da Figura 1.4 serão interpretadas como a IRI absoluta:

```
http://example.org/data/song.mp3
```

A diretiva `@prefix` introduz um novo rótulo de prefixo que pode ser utilizado para abreviar IRIs. Por exemplo, na Figura 1.4, os rótulos `ex:` e `xsd:` correspondem aos seguintes prefixos de IRIs:

```
http://example.org/#
http://www.w3.org/2001/XMLSchema#
```

Observe que IRIs prefixadas não são escritas entre `<...>`.

A sintaxe Turtle possui muitas outras facilidades. Algumas dessas são úteis para evitar repetições dentro do documento. Por exemplo, a Figura 1.5 utiliza algumas dessas facilidades adicionais para reescrever em menos linhas o documento da Figura 1.4. Observe os caracteres ponto e vírgula "`;`" nas linhas 3 e 4 da Figura 1.5. Esses caracteres são utilizados para separar pares predicado-objeto que se referem a um mesmo sujeito; nesse caso, à IRI `http://example.org/data/song.mp3`. Esse tipo de construção é chamada de *lista de predicados*. Observe também o literal tipado 10240 na linha 4. Alguns tipos comuns de literais, como inteiros e números em ponto flutuante, podem ser escritos diretamente em suas sintaxes tradicionais em Turtle.

[9] É comum se usar nós retangulares para representar literais em grafos RDF. Por se tratar de uma representação abstrata, o grafo anterior não segue essa convenção.

```
@prefix ex: <http://example.org/#> .
<http://example.org/data/song.mp3>
  ex:tipo ex:Musica ;
  ex:tamanho 10240 ;
  ex:titulo "9a Sinfonia" .
```

Figura 1.5. Exemplo da Figura 1.4 reescrito usando-se uma lista de predicados.

É importante perceber que um grafo RDF é apenas uma estrutura de dados, e que Turtle, RDF/XML e outras sintaxes para RDF são apenas formas de se codificar essa estrutura de dado em uma *string*. Ainda mais importante é perceber que os significados dos rótulos que ocorrem em um grafo RDF não são definidos pelo padrão RDF. Esses rótulos podem indicar alguma coisa para pessoa que os lê, mas para computador que entende apenas RDF esses rótulos não possuem significado—eles servem apenas para identificar e distinguir componentes específicos do grafo.

Existe um outro padrão, chamado *RDF Schema* (RDFS) [Hayes and Patel-Schneider 2014], definido sobre o RDF, que especifica um vocabulário básico (conjunto predefinido de termos) e significados associados. Esse vocabulário está disponível no seguinte *namespace*:

`http://www.w3.org/2000/01/rdf-schema#`

O RDFS provê um conjunto básico de termos que pode ser utilizado para definir termos mais específicos cujo significado é dado através da descrição de suas inter-relações com outros termos. (Note que usamos uma abordagem similar na Seção 1.2 quando introduzimos *strings* representando conceitos e papéis, como `Diretor` e `dirige`, e em seguida definimos os seus significados através de GCIs e RIAs que os relacionam com outros termos.)

O RDFS é às vezes chamado de *linguagem de ontologia leve* por permitir a definição ontologias (especificações de conceitualizações com semântica formal) através de um vocabulário pequeno cuja semântica é relativamente simples. Essas características podem ser suficientes para certas aplicações mas, em geral, aplicações mais sofisticadas demandam linguagens ontológicas mais poderosas. A escolha natural nesse caso é OWL, também uma tecnologia da Web Semântica. Vamos discutir OWL na Seção 1.3.3. Antes disso, porém, vamos apresentar SPARQL, a linguagem de consulta de RDF.

1.3.2. SPARQL

Na seção anterior mostramos um exemplo de grafo RDF contendo de quatro nós e três arestas. Quando estamos trabalhando com grafos relativamente pequenos como esse não precisamos nos preocupar com eficiência. Mas, na prática, grafos RDF raramente são pequenos. Não é incomum encontrarmos aplicações que lidam com grafos contendo milhões de nós e arestas. Para tais aplicações, ferramentas simples não são o bastante. Elas precisam de ferramentas especializadas, em particular, bancos de dados especializados, os chamados *bancos de dados de grafos* ou *bases de triplas* (*triple stores*), projetados para armazenar e processar de forma eficiente grandes volumes de dados RDF.

SPARQL (*SPARQL Protocol and RDF Query Language*) está para os bancos de dados de grafos e bases de triplas assim como SQL está para os bancos de dados

relacionais. SPARQL é uma interface declarativa para consultar e manipular o conteúdo da base/grafo [Harris and Seaborne 2013]. Há dois tipos principais de consultas SPARQL: reconhecimento de padrões em grafos e navegação em grafos. Vamos discutir ambos os tipos a seguir.

Reconhecimento de padrões em grafos Uma consulta de reconhecimento de padrão é uma consulta que busca por um dado padrão estrutural (limitado) no grafo. Em SPARQL, tais padrões estruturais são expressados como *padrões de triplas*, isto é, triplas RDF em que o sujeito, predicado e objeto podem ser variáveis (nomes começando com "`?`"). Um *padrão básico de grafo* (*Basic Graph Pattern*, ou BGP) consiste da combinação de um ou mais padrões de triplas.

Considere o grafo RDF apresentado na Figura 1.6 que descreve o filme *Taxi Driver*. Suponha que queiramos listar todos os pares de pessoas distintas que atuaram nesse filme. Podemos especificar essa consulta em SPARQL da seguinte forma:

```
1 PREFIX: <http://example.org/#>
2 SELECT ?x1 ?x2
3 WHERE {
4   ?x1 :atuaEm ?x3 . ?x1 :tipo :Pessoa .
5   ?x2 :atuaEm ?x3 . ?x1 :tipo :Pessoa .
6   ?x3 :titulo "Taxi Driver" . ?x3 :tipo :Filme .
7   FILTER (?x1 != ?x2)
8 }
```

Essa consulta expressa um BGP contendo seis padrões de triplas. Ela seleciona todos os pares de rótulos x_1 e x_2 (linha 2) com $x_1 \neq x_2$ (linha 7) tais que, para algum rótulo x_3, cada uma das seguintes triplas ocorre no grafo:

$$\langle x_1, \texttt{:atuaEm}, x_3 \rangle \quad \text{e} \quad \langle x_1, \texttt{:tipo}, \texttt{:Pessoa} \rangle \qquad \text{(linha 4)}$$
$$\langle x_2, \texttt{:atuaEm}, x_3 \rangle \quad \text{e} \quad \langle x_2, \texttt{:tipo}, \texttt{:Pessoa} \rangle \qquad \text{(linha 5)}$$
$$\langle x_3, \texttt{:titulo}, \texttt{"Taxi Driver"} \rangle \quad \text{e} \quad \langle x_3, \texttt{:tipo}, \texttt{:Filme} \rangle \qquad \text{(linha 6)}$$

Se aplicarmos essa consulta ao grafo da Figura 1.6 vamos obter o seguinte resultado:

```
1 ?x1          ?x2
2 ---------------------
3 :scorsese    :de-niro
4 :de-niro     :scorsese
```

Ou seja, o nosso *conjunto resposta* contém exatamente dois pares de rótulos: um em que x_1 é `:scorsese` e x_2 é `:de-niro` (linha 3 do resultado), e outro em que x_1 é `:de-niro` e x_2 é `:scorsese` (linha 4 do resultado).

Navegação em grafos O outro tipo comum de consulta suportado por SPARQL são as consultas de navegação em grafos. Essas são consultas que navegam na topologia do grafo percorrendo caminhos de tamanhos potencialmente arbitrários. Uma *consulta de caminho*

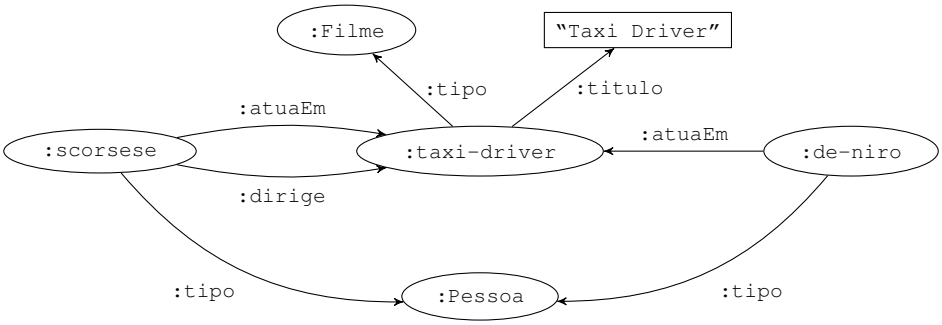


Figura 1.6. Grafo RDF descrevendo *Taxi Driver*.

é um tipo básico de consulta de navegação da forma $x \xrightarrow{\alpha} y$ que obtém todos os pares $\langle x, y \rangle$ tais que existe um caminho no grafo de x para y que satisfaz uma dada condição α.

Por exemplo, as consultas de caminho

$$x \xrightarrow{\texttt{:dirige}} y \quad \text{e} \quad x \xrightarrow{\texttt{:atuaEm} \circ \texttt{:tipo}} y$$

selecionam, respectivamente, todos os nós x e y tais que existe uma aresta com rótulo `:dirige` de x para y, e todos os nós x e y tais que existe um caminho de tamanho dois de x para y em que a primeira aresta do caminho possui rótulo `:atuaEm` e a segunda aresta possui rótulo `:tipo`. (Lembre-se de que um *caminho* em um grafo é uma sequência de arestas que conectam uma sequência de nós.)

Podemos especificar essa última consulta em SPARQL da seguinte forma:

```
SELECT ?x ?y
WHERE  ?x (:atuaEm/:tipo) ?y
```

Aqui o símbolo "`/`" denota encadeamento de rótulos de arestas e corresponde ao "$\circ$" da notação abstrata. Se aplicarmos essa consulta SPARQL ao grafo da Figura 1.6 vamos obter a resposta:

```
?x           ?y
------------------
:scorsese    :Movie
:de-niro     :Movie
```

Para um exemplo mais interessante, considere a consulta de caminho

$$x \xrightarrow{(\texttt{:atuaEm} \circ \texttt{:atuaEm}^{-})^{+}} y.$$

Essa consulta busca todos os atores que possuem uma distância de colaboração finita[10], isto é, atores x e y tais que x e y contracenam no mesmo filme (distância 0), ou x e y contracenam com o mesmo z em algum filme (distância 1), ou x contracena com algum z_1 em algum filme e y contracena com algum z_2 em algum filme e z_1 e z_2 contracenam no mesmo filme (distância 2), e assim por diante. O código SPARQL abaixo implementa essa consulta:

```
SELECT ?x ?y
WHERE  ?x (:atuaEm/^:atuaEm)+ ?y
```

Nesse caso, o símbolo "^" denota o inverso de um rótulo de aresta (ele corresponde ao símbolo "−" da notação abstrata) e o símbolo "+" denota uma ou mais repetições do padrão que o precede.

Suponha que adicionemos as seguintes triplas ao grafo da Figura 1.6:

$\langle$:de-niro, :atuaEm, :cassino$\rangle$ e $\langle$:sheron-stone, :atuaEm, :cassino$\rangle$.

Se rodarmos o SPARQL anterior sobre esse grafo estendido vamos obter o seguinte resultado:

```
?x              ?y
-----------------------------
:scorsese       :de-niro
:de-niro        :scorsese
:de-niro        :sharon-stone
:sharon-stone   :de-niro
:scorsese       :sharon-stone
:sharon-stone   :scorsese
```

SPARQL possui muitas outras funcionalidades, como grupos, opções, alternativas, filtros, etc., que podem ser combinadas com BGPs e operadores de navegação para especificar consultas complexas. Nosso objetivo aqui foi apresentar uma visão geral de SPARQL. Para uma introdução mais detalhada à linguagem sugerimos [Hitzler et al. 2010]. Além de SPARQL, há outras linguagens para consultas em grafos. Veja [Angles 2012] para uma discussão geral dos princípios por trás dessas linguagens. Para uma introdução prática aos bancos de dados de grafos recomendamos [Robinson et al. 2013].

1.3.3. OWL

Web Ontology Language (OWL) é uma família de linguagens ontológicas cuja sintaxe e semântica (formal) é padronizada [W3C OWL WG 2012]. Uma *ontologia* é um conjunto de proposições descritivas sobre um determinado domínio—proposições como àquelas de lógica de descrição. OWL é baseado em lógica de descrição mas possui muitas outras facilidades que vão além de DL pura, como por exemplo tipos de dados e funcionalidades para versionamento e anotação de ontologias. Nesta seção, vamos focar na versão 2 de OWL (chamada de OWL 2). Essa é a versão mais recente da linguagem e é compatível com a versão anterior.

Existem três sublinguagens de OWL, também chamadas de *espécies*: OWL Full, OWL DL e OWL Lite. OWL Full contém ambas OWL DL e OWL Lite, e OWL DL contém OWL Lite. O que distingue essas três sublinguagens é o seu grau de expressividade e, consequentemente, a complexidade computacional das tarefas de raciocínio associadas a cada uma delas.

[10]Quando x é o ator Kevin Bacon essa distância é conhecida como a *distância Bacon* ou *grau Bacon*. Veja: https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon

Tabela 1.3. Diferenças terminológicas entre OWL e DL.

OWL	DL
nome de classe	nome de conceito
classe	conceito
nome de propriedade de objeto	nome de papel
propriedade de objeto	papel
ontologia	base de conhecimento
axioma	axioma
vocabulário	vocabulário/assinatura

Como qualquer linguagem, para apresentar OWL vamos ter que escolher uma determinada sintaxe. No caso, vamos utilizar a sintaxe RDF/Turtle. Um documento OWL escrito nessa linguagem é também um documento RDF válido. Além de RDF, outras sintaxes comuns de OWL são a sintaxe em estilo funcional e a sintaxe Manchester. No entanto, a única sintaxe cujo suporte é mandatório para todas as ferramentas OWL 2 é a sintaxe RDF/XML [Hitzler et al. 2014].

A seguir, vamos apresentar os termos e a semântica de OWL a partir de traduções de DL. (Uma ontologia OWL 2 DL é essencialmente uma base de conhecimento $\mathcal{SROIQ}$.) Antes de fazer isso, porém, precisamos esclarecer alguns conflitos terminológicos. Por razões históricas, OWL e DL às vezes usam termos distintos para se referir ao mesmo objeto. Essas diferenças terminológicas são apresentadas na Tabela 1.3.

De $\mathcal{SROIQ}$ para OWL 2 DL A tradução de uma base de conhecimento $\mathcal{SROIQ}$ para um documento OWL consiste de três passos.[11] Primeiro, precisamos prefixar ao documento as seguintes declarações de *namespace*:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Os prefixos `rdfs:`, `rdf:` e `xsd:` são necessários pois OWL reusa os termos desses *namespaces*.

A segunda coisa que precisamos fazer é adicionar ao documento OWL declarações de tipos para os nomes de conceitos e nomes de papéis que ocorrem na KB. Ou seja, para cada nome de conceito `C` que ocorre na KB, adicionamos ao documento a tripla:

```
C rdf:type owl:Class .
```

E para cada nome de papel `r` que ocorre na KB, adicionamos a tripla:

```
r rdf:type owl:ObjectProperty .
```

[11] As traduções que apresentamos aqui foram adaptadas de [Rudolph 2011].

A Figura 1.7 apresenta o documento OWL correspondente à KB de fatos sobre filmes apresentada no final da Seção 1.2.1. As declarações de tipo nas linhas 7–13 foram geradas pelas traduções acima. Observe que por causa da diretiva `@prefix` na linha 5, IRIs com prefixo vazio, como `:Ator`, são interpretadas como sendo prefixadas pela IRI `http://example.org/movie-facts#`.

O terceiro passo da tradução de uma KB $\mathcal{SROIQ}$ para OWL é o mais complicado dos três: precisamos traduzir os axiomas que ocorrem em cada uma das caixas da KB. A complexidade, nesse caso, decorre da estrutura sintática de GCIs e RIAs que não pode ser codificada diretamente em RDF. Para reescrevermos GCIs e RIAs em RDF vamos precisar utilizar nós em branco e técnicas de codificação de listas em grafos.

A tradução exata de axiomas $\mathcal{SROIQ}$ para OWL é dada pela função $[\![\square]\!]$ abaixo:

$$
\begin{aligned}
[\![r_1 \circ \cdots \circ r_n \sqsubseteq r]\!] &= [\![r]\!]_{\mathbf{C}}\ \texttt{owl:propertyChainAxiom}\ (\ [\![r_1]\!]_{\mathbf{R}} \ldots [\![r_n]\!]_{\mathbf{R}}\ )\ . \\
[\![\mathsf{Sym}(r)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:type owl:SymmetricProperty}\ . \\
[\![\mathsf{Asy}(r)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:type owl:AsymmetricProperty}\ . \\
[\![\mathsf{Tra}(r)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:type owl:TransitiveProperty}\ . \\
[\![\mathsf{Ref}(r)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:type owl:ReflexiveProperty}\ . \\
[\![\mathsf{Irr}(r)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:type owl:IrreflexiveProperty}\ . \\
[\![\mathsf{Dis}(r,s)]\!] &= [\![r]\!]_{\mathbf{R}}\ \texttt{owl:propertyDisjointWith}\ [\![s]\!]_{\mathbf{R}}\ . \\
[\![C \sqsubseteq D]\!] &= [\![C]\!]_{\mathbf{C}}\ \texttt{rdfs:subClassOf}\ [\![D]\!]_{\mathbf{C}}\ . \\
[\![C(\mathtt{a})]\!] &= \mathtt{a}\ \texttt{rdf:type}\ [\![C]\!]_{\mathbf{C}}\ . \\
[\![\mathtt{r}(\mathtt{a},\mathtt{b})]\!] &= \mathtt{a}\ \mathtt{r}\ \mathtt{b}\ . \\
[\![\mathtt{r}^-(\mathtt{a},\mathtt{b})]\!] &= \mathtt{b}\ \mathtt{r}\ \mathtt{a}\ . \\
[\![\neg r(\mathtt{a},\mathtt{b})]\!] &= \texttt{[] rdf:type owl:NegativePropertyAssertion ;} \\
&\qquad \texttt{owl:assertionProperty}\ [\![r]\!]_{\mathbf{R}}\ \texttt{;} \\
&\qquad \texttt{owl:sourceIndividual}\ \mathtt{a}\ \texttt{;} \\
&\qquad \texttt{owl:targetIndividual}\ \mathtt{b}\ . \\
[\![\mathtt{a} \approx \mathtt{b}]\!] &= \mathtt{a}\ \texttt{owl:sameAs}\ \mathtt{b}\ . \\
[\![\mathtt{a} \not\approx \mathtt{b}]\!] &= \mathtt{a}\ \texttt{owl:differentFrom}\ \mathtt{b}\ .
\end{aligned}
$$

em que as funções auxiliares $[\![\square]\!]_{\mathbf{R}}$ e $[\![\square]\!]_{\mathbf{C}}$ são usadas para traduzir papéis e expressões de conceitos. Essas funções auxiliares são definidas da seguinte forma:

$$
\begin{aligned}
[\![u]\!]_{\mathbf{R}} &= \texttt{owl:topObjectProperty} \\
[\![\mathtt{r}]\!]_{\mathbf{R}} &= \mathtt{r} \\
[\![\mathtt{r}^-]\!]_{\mathbf{R}} &= \texttt{[owl:inverseOf :r]}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathtt{C}]\!]_{\mathbf{C}} &= \mathtt{C} \\
[\![\top]\!]_{\mathbf{C}} &= \texttt{owl:Thing} \\
[\![\bot]\!]_{\mathbf{C}} &= \texttt{owl:Nothing} \\
[\![\{\mathtt{a}_1,\ldots,\mathtt{a}_n\}]\!]_{\mathbf{C}} &= \texttt{[rdf:type owl:Class ; owl:oneOf(}\ :\mathtt{a}_1\ \ldots\ :\mathtt{a}_n\ \texttt{)]} \\
[\![\neg C]\!]_{\mathbf{C}} &= \texttt{[rdf:type owl:Class ; owl:complementnOf}\ [\![C]\!]_{\mathbf{C}}\ \texttt{]}
\end{aligned}
$$

$$[\![C_1 \sqcap \cdots \sqcap C_n]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Class ; owl:intersectionOf (} [\![C_1]\!]_{\mathbf{C}} \ldots [\![C_n]\!]_{\mathbf{C}} \texttt{)]}$$

$$[\![C_1 \sqcup \cdots \sqcup C_n]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Class ; owl:unionOf (} [\![C_1]\!]_{\mathbf{C}} \ldots [\![C_n]\!]_{\mathbf{C}} \texttt{)]}$$

$$[\![\exists r.C]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Restriction ;}$$
$$\texttt{owl:onProperty} [\![r]\!]_{\mathbf{R}} \texttt{ ; owl:someValuesFrom} [\![C]\!]_{\mathbf{C}} \texttt{]}$$

$$[\![\forall r.C]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Restriction ;}$$
$$\texttt{owl:onProperty} [\![r]\!]_{\mathbf{R}} \texttt{ ; owl:allValuesFrom} [\![C]\!]_{\mathbf{C}} \texttt{]}$$

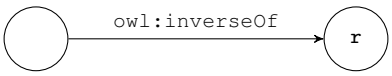
$$[\![\exists r.\mathsf{Self}]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Restriction ;}$$
$$\texttt{owl:onProperty} [\![r]\!]_{\mathbf{R}} \texttt{ ; owl:hasSelf true]}$$

$$[\![\geq n r.C]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Restriction ;}$$
$$\texttt{owl:minQualifiedCardinality} n \texttt{ ;}$$
$$\texttt{owl:onProperty} [\![r]\!]_{\mathbf{R}} \texttt{ ; owl:onClass} [\![C]\!]_{\mathbf{C}} \texttt{]}$$

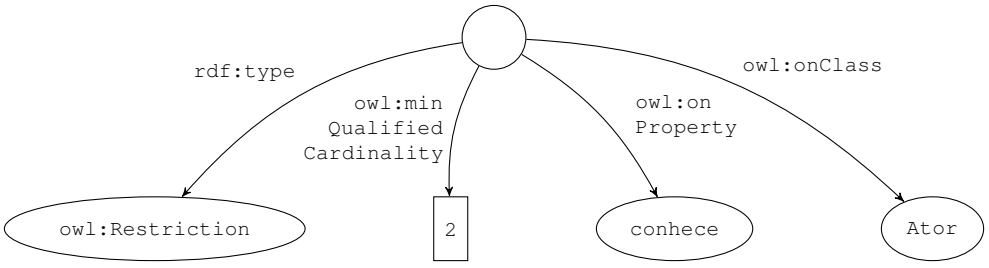
$$[\![\leq n r.C]\!]_{\mathbf{C}} = \texttt{[rdf:type owl:Restriction ;}$$
$$\texttt{owl:maxQualifiedCardinality} n \texttt{ ;}$$
$$\texttt{owl:onProperty} [\![r]\!]_{\mathbf{R}} \texttt{ ; owl:onClass} [\![C]\!]_{\mathbf{C}} \texttt{]}$$

Dois comentários sobre a tradução acima se fazem necessários. Primeiro, observe que nomes de indivíduos, nomes de conceitos e nomes de papéis são mantidos sem alteração pelas funções $[\![\square]\!]$, $[\![\square]\!]_{\mathbf{R}}$ e $[\![\square]\!]_{\mathbf{C}}$. Para simplificar, vamos assumir que esses nomes estão definidos no *namespace* de prefixo vazio.

Segundo, observe que papéis e expressões de conceitos complexas são traduzidas pelas funções $[\![\square]\!]_{\mathbf{R}}$ e $[\![\square]\!]_{\mathbf{C}}$ como subgrafos identificados por nós em branco. Na sintaxe Turtle, esses nós em branco são introduzidos por colchetes. Por exemplo, o papel $\texttt{r}^-$ é traduzido para o subgrafo



em que o nó em branco à esquerda representa o inverso de `r`. De maneira similar, o conceito complexo $\geq$2`conhece.Ator` é traduzido para o subgrafo



Novamente, o nó em branco acima representa o conceito complexo como um todo.

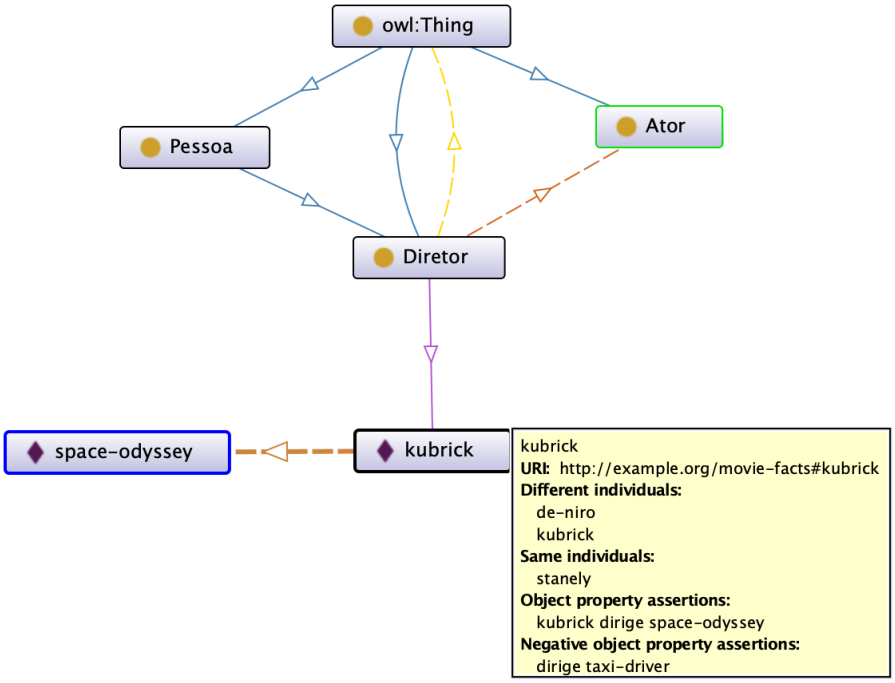
O resultado da aplicação da função $[\![\square]\!]$ aos axiomas da KB fatos sobre filmes é apresentado na Figura 1.7. Os axiomas de ABox resultantes aparecem nas linhas 16–28, os de TBox nas linhas 31–53 e os de RBox nas linhas 55–56.

```
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix :       <http://example.org/movie-facts#> .

:Ator       rdf:type    owl:Class .
:Diretor    rdf:type    owl:Class .
:Pessoa     rdf:type    owl:Class .
:atuaEm     rdf:type    owl:ObjectProperty .
:dirige     rdf:type    owl:ObjectProperty .
:conhece    rdf:type    owl:ObjectProperty .
:gosta      rdf:type    owl:ObjectProperty .

# Abox
:kubrick    rdf:type    :Diretor .
:de-niro    rdf:type    [rdf:type owl:Class ;
                         owl:intersectionOf (:Ator :Diretor)] .
:stanley    rdf:type    [rdf:type owl:Restriction ;
                         owl:minQualifiedCardinality 2 ;
                         owl:onProperty :conhece ; owl:onClass :Ator ] .
:kubrick    :dirige    :space-odyssey .
[]          rdf:type    owl:NegativePropertyAssertion ;
                        owl:assertionProperty :dirige ;
                        owl:sourceIndividual :kubrick ;
                        owl:targetIndividual :taxi-driver .
:kubrick    owl:sameAs  :stanely .
:kubrick    owl:differentFrom   :de-niro .

# TBox
:Diretor   rdfs:subClassOf     [rdf:type owl:Restriction ;
                                owl:onProperty :dirige ;
                                owl:someValuesFrom owl:Thing] .
[rdf:type owl:Restriction ;
 owl:onProperty :dirige ;
 owl:someValuesFrom owl:Thing]
           rdfs:subClassOf     :Diretor .
:Diretor   rdfs:subClassOf     :Pessoa .
:Diretor   rdfs:subClassOf     [rdf:type owl:Restriction ;
                                owl:minQualifiedCardinality 1 ;
                                owl:onProperty :conhece ;
                                owl:onClass :Ator] .
[rdf:type owl:Class ;
 owl:intersectionOf  ([rdf:type owl:Restriction ;
                       owl:onProperty :dirige ;
                       owl:someValuesFrom owl:Thing]
                       [rdf:type owl:Restriction ;
                        owl:onProperty :dirige ;
                        owl:allValuesFrom [rdf:type owl:Class ;
                                           owl:oneOf (:a-bronx-tale
                                                      :good-shepherd)]])]
           rdfs:subClassOf     [rdf:type owl:Class ;
                                owl:oneOf (:de-niro)] .
# RBox
:gosta      owl:propertyChainAxiom  (:dirige) .
:conhece    owl:propertyChainAxiom  (:dirige [owl:inverseOf :atuaEm]) .
```

Figura 1.7. OWL correspondente à KB de fatos sobre filmes da Seção 1.2.1.

Simulando os tipos adicionais de axiomas de OWL em $\mathcal{SROIQ}$ As funcionalidades de OWL utilizadas pelo algoritmo de tradução anterior são suficientes para representar qualquer axioma $\mathcal{SROIQ}$. No entanto, OWL possui muitas outras funcionalidades, incluindo tipos adicionais de axiomas lógicos que não possuem contrapartida direta em $\mathcal{SROIQ}$. Esses tipos adicionais de axiomas de OWL são úteis na prática, mas eles não adicionam expressividade extra do ponto de vista lógico. Ou seja, esses axiomas adicionais são apenas açúcares sintáticos que podem ser sempre reescritos em termos dos tipos de axiomas disponíveis em $\mathcal{SROIQ}$. Veja [Rudolph 2011] para mais detalhes.

Ferramentas para manipulação de ontologias Se carregarmos o arquivo Turtle da Figura 1.7 na ferramenta Protégé[12] e solicitarmos um grafo da ontologia vamos obter o seguinte:



Aqui os nós contendo um círculo amarelo representam conceitos (classes), os nós contendo losangos roxos representam indivíduos e as setas coloridas representam relacionamentos específicos entre nós. Por exemplo, a seta azul de `Pessoa` para `Diretor` representa o relacionamento "possui subclasse". O retângulo amarelo, contendo informações adicionais sobre `kubrick`, é o *tool-tip* apresentado pelo Protégé quando movemos o cursor do mouse sobre um nó ou aresta do grafo.

O Protégé é o editor de ontologias mais utilizado; ele suporta diversos *plugins* para visualização de ontologias, depuração e raciocínio. Além do Protégé, há muitas outras ferramentas para projetar e implementar ontologias. Para uma lista abrangente, veja [Hitzler et al. 2010] e [W3C Semantic Web Wiki 2019]

***Linked data* e grafos de conhecimento** Ao longo dos anos, o foco da visão da Web Semântica migrou da descrição de documentos para a descrição de dados em geral. O

[12]https://protege.stanford.edu/

	Vocabulário/Ontologia	Linguagem
Derivadas de descritores	Content Ontology for the TV-Anytime Content	OWL
	Content Ontology for the TV-Anytime Format	OWL
	Core Ontology for Multimedia (COMM)	OWL
	Visual Descriptor Ontology (VDO)	OWL
De propósito geral	Dublin Core	RDFS
	FOAF	OWL
	Schema.org	OWL
	SUMO	SUO-KIF
	WordNet 3.x	OWL
Especializadas	3D Modeling Language (3DMO)	OWL 2
	Audio Effects Ontology (AUFX-O)	OWL
	Linked Movie Database (LMD)	RDFS
	Multimedia Metadata Ontology (M3O)	OWL
	Ontology for Media Resources	OWL
	Video Ontology (VidOnt)	OWL

Tabela 1.4. Alguns vocabulários e ontologias para multimídia.

termo *Linked Open Data* (LOD) tem sido utilizado para se referir às bases abertas de dados relacionados (*linked data*), tais como DBpedia[13] (derivada da Wikipedia), Wikidata[14] (curada pela Fundação Wikimedia), etc. A maioria dessas bases adota as tecnologias discutidas nessa seção, porém muitas delas não são *ontologias* propriamente ditas.

Outro termo comumente associado a *linked data* é o termo *grafo de conhecimento*. Esse último é um termo geral que faz alusão ao fato de que o sistema ou banco de dados ao qual o termo se aplica utiliza grafos para representar conhecimento. Nesse sentido, um documento OWL codificado como um RDF é um grafo de conhecimento. Mas o fato de ser um grafo de conhecimento não implica em ser uma ontologia. Por exemplo, o grafo apresentado na Figura 1.6 é um grafo de conhecimento, mas certamente não é uma ontologia.

Ontologias para multimídia Há diversas ontologias para descrição de dados multimídia. Podemos classificá-las basicamente em três grupos: as derivadas de descritores, as de propósito geral e as ontologias especializadas. As ontologias do primeiro grupo são aquelas derivadas dos padrões de metadados clássicos, tais como MPEG-7, MPEG-21 e TV-Anytime. Essas ontologias são complexas, limitadas em termos de expressividade e possuem falhas de projeto que dificultam o seu uso (elas normalmente não seguem boas práticas de projeto de ontologias). As ontologias do segundo grupo tendem a ser mais maduras e bem definidas. Já as ontologias do terceiro grupo, assim como as do primeiro, também são problemáticas. Elas normalmente possuem problemas de conceitualização, por exemplo, escopo incerto ou mal definido, e costumam não utilizar termos de ontologias mais gerais (ontologias *upper level*) o que é considerado uma prática ruim.

[13]https://wiki.dbpedia.org/
[14]https://www.wikidata.org/

A Tabela 1.4 apresenta algumas ontologias que fazem parte desses três grupos. Para mais detalhes sobre essas ontologias, incluindo referências, veja [Sikos 2017].

O propósito de se utilizar uma das ontologias anteriores é, evidentemente, promover o reúso de termos e definições. Por exemplo, usando os ontologias Schema.org and Dublin Core podemos reescrever o RDF que descreve o arquivo de áudio "song.mp3" (Figura 1.5) da seguinte forma:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix schema: <http://schema.org/> .
<http://example.org/data/song.mp3> rdf:type schema:AudioObject ;
                                   schema:contentSize "10240" ;
                                   dc:title "9a Sinfonia" .
```

Observe que podemos usar *URI fragments* [Pfeiffer et al. 2012] fazer referência a partes específicas do recurso que está sendo descrito. Por exemplo, se quisermos indicar que o tom de um segmento específico de "song.mp3", digamos dos 10s aos 20s, é C Maior, podemos escrever:

```
@prefix keys: <http://purl.org/NET/c4dm/keys.owl#>
@prefix mo:   <http://purl.org/ontology/mo/>
<http://example.org/data/song.mp3#t=10,20> mo:key keys:CMajor .
```

em que `mo:key` e `keys:CMajor` são termos definidos na Music Ontology[15]. De maneira similar, podemos usar *URI fragments* fazer referencias a fragmentos espaciais de recursos especificados via `#xywh=`x`,`y`,`w`,`h. Falaremos mais sobre objetos de mídia e suas partes na seção seguinte.

1.4. Uma Abordagem Híbrida: Hyperknowledge

O *Hyperknowledge* [Moreno et al. 2017] é um modelo híbrido para representação de conhecimento que permite especificar e interligar objetos multimídia e conceitos. O Hyperknowledge estende o *Nested Context Model* (NCM) [Soares and Rodrigues 2005], um modelo hipermídia clássico, com construções que combinam funcionalidades dos domínios de hipermídia e representação de conhecimento. Para melhor entendermos a origem desse modelo, vamos discutir rapidamente as preocupações principais de ambas essas comunidades.

Nas últimas décadas a comunidade de hipermídia tem se dedicado a definição de modelos e linguagens para expressar relacionamentos entre objetos de mídia (textos, imagens, vídeos, etc.). Em linguagens como HTML, NCL, SMIL e SVG, podemos especificar como diferentes objetos interagem uns com os outros e com os usuários, mas não há como relacionar esses objetos aos conceitos que eles representam—pelo menos não diretamente. Também não podemos descrever nessas linguagens ontologias como as que discutimos nas seções anteriores. Padrões de metadados (tais como MPEG-7, MPEG-21, SMPTE, EXIF) tem sido propostos como uma solução para combinar conteúdo de mídia

[15]http://musicontology.com/

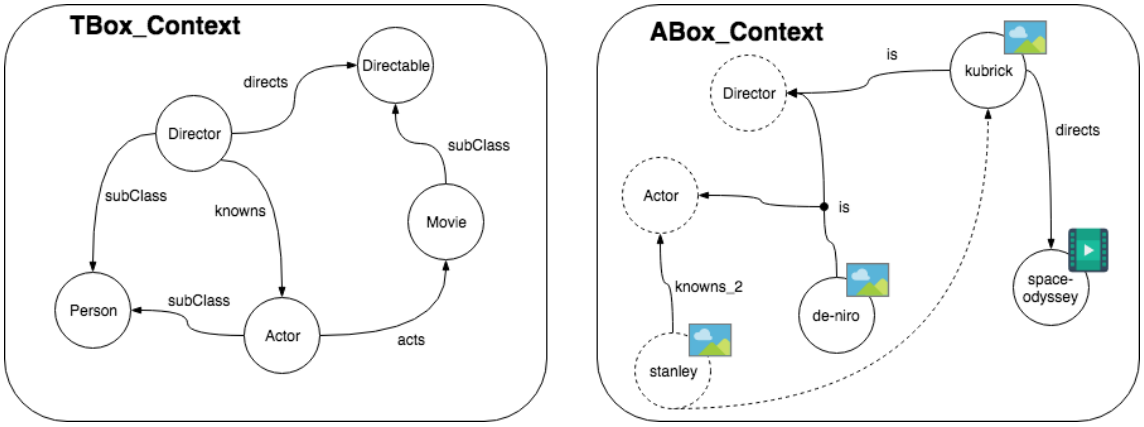


Figura 1.8. Uma base Hyperknowledge correspondente à KB de fatos sobre filmes.

com descrição semântica, mas esses padrões tendem a focar apenas em características de baixo nível do conteúdo multimídia, como codecs, taxas de bits, espaços de cores, etc.

A comunidade de representação de conhecimento, por outro lado, tem se dedicado principalmente ao desenvolvimento de modelos para representação de fatos do mundo real e de métodos para consulta e inferência de fatos a partir desses modelos. As aplicações do domínio de representação de conhecimento tendem a assumir que as bases de conhecimento são compostas apenas por fatos e normalmente desconsideram dados multimídia relacionados a esses fatos. Por exemplo, podemos usar RDF e OWL para codificar fatos em uma base de conhecimento, mas os conteúdos multimídia descritos por esses fatos são normalmente mantido em bases separadas, manipuladas via ferramentas e sistemas distintos.

O objetivo do Hyperknowledge é prover um *framework* que unifique hipermídia e representação de conhecimento e que, com isso, torne possível o desenvolvimento de aplicações multimídia *semantic-aware* avançadas e bases de conhecimento *multimedia-aware*.

1.4.1. Hyperknowledge

As principais entidades do modelo Hyperknowledge, herdadas do NCM, são os nós atômicos, composições, elos e conectores. Vamos utilizar um exemplo para introduzir essas entidades. A Figura 1.8 apresenta uma possível representação em Hyperknowledge de uma parte da KB de fatos sobre filmes apresentada no fim da Seção 1.2. Para simplificar, vamos adotar a hipótese de mundo fechado, isto é, vamos assumir que os fatos não representados na figura são falsos. (Discutimos essa hipótese e a alternativa mais geral, hipótese de mundo aberto, na Seção 1.2.3.)

Um *contexto* Hyperknowledge é um elemento contêiner utilizado para agrupar elementos relacionados (possivelmente outros contextos). A caracterização do que são elementos relacionados e a decisão de agrupá-los em contextos é uma decisão de modelagem que depende da aplicação. Na Figura 1.8, os contextos são representados pelos retângulos de bordas arredondadas. Há dois contextos na figura. O primeiro, `TBox_Context`,

contém os elementos e relacionamentos que correspondem à TBox da KB de fatos sobre filmes. O outro contexto, `ABox_Context`, contém os elementos e relacionamentos que correspondem à ABox.

Os círculos rotulados na Figura 1.8 denotam nós atômicos. Esses podem representar tanto conceitos quando conteúdo multimídia. Aqui estamos usando o termo "conceito" no mesmo sentido utilizado em DL. Isto é, um *conceito* (ou classe na terminologia OWL) representa uma característica abstrata de certos indivíduos. Na Figura 1.8, todos os nós da composição `TBox_Context`, a saber, `Pessoa`, `Diretor`, `Ator`, `Filme` e `Dirigivel`, são nós de conceito.

O outro tipo de nó atômico que ocorre na Figura 1.8 são os nós de mídia. Um *nó de mídia* é um nó contendo algum conteúdo multimídia, por exemplo, uma imagem, áudio, vídeo, texto, etc. Na figura, os nós de mídia são os nós `de-niro`, `kubrick` e `space-odyssey` que ocorrem no contexto `ABox_Context`. Os ícones à direita desses nós indicam que `de-niro` e `kubrick` são imagens e que `space-odyssey` é um vídeo.

Os nós de borda tracejada na Figura 1.8 não são nem nós de conceito nem nós de mídia; eles representam *nós de referência*. Isto é, os nós tracejados são nós que se referem a outros nós, permitindo que esses últimos possam ser reusados em outra situações (possivelmente, outros contextos). Por exemplo, os nós de conceito `Ator` e `Diretor`, definidos no contexto TBox, são reusados no contexto ABox. De maneira similar, o nó `stanley` do contexto ABox é uma referência para o nó `kubrick` do mesmo contexto. A ideia aqui é que qualquer coisa que opere sobre `stanley` estará operando na verdade sobre `kubrick`.

Esse tipo de indireção é útil em casos em que a mesma entidade está associada a diferentes nomes em contextos possivelmente diferentes. Por exemplo, o cantor e compositor Bob Dylan gravou algumas músicas sob o pseudônimo Blind Boy Grunt. Qualquer referência a Blind Boy Grunt é essencialmente uma referência a Bob Dylan. Mas a distinção entre esses dois pode ser importante em algumas aplicações.

De volta a Figura 1.8, as setas denotam elos que representam relacionamentos entre nós. Cada elo está associado a um determinado conector (não representado na figura) que especifica o tipo do elo. Um conector define uma relação abstrata e associa a essa relação um rótulo e um conjunto de restrições. Um elo podem ser visto como uma instância de seu conector, isto é, como uma realização concreta da relação abstrata definida pelo seu conector.

Na figura, o elo do contexto TBox entre `Ator` e `Pessoa` com rótulo "subClass" expressa o fato de que toda instância de `Ator` é também uma instância de `Pessoa`. Em outras palavras, esse elo codifica o axioma de TBox $\texttt{Ator} \sqsubseteq \texttt{Pessoa}$. De maneira similar, o elo "dirige" entre `Diretor` e `Dirigivel` codifica o axioma de TBox $\texttt{Diretor} \sqsubseteq \exists \texttt{dirige}.\top$.

Diferentemente de arestas de grafos RDF, que são necessariamente binárias, os elos de Hyperknowledge podem conectar mais de duas entidades. Por exemplo, na Figura 1.8 o elo "is" no contexto ABox conecta `de-niro` a ambos `Ator` e `Diretor`, e dessa forma codifica o axioma de ABox $\texttt{Ator} \sqcap \texttt{Diretor}(\texttt{de-niro})$.

O modelo Hyperknowledge possui muitas outras funcionalidades que facilitam

a manipulação de descrições semânticas em conjunto com conteúdo multimídia. Por exemplo, o modelo suporta a noção de âncoras que delimitam partes de nós de mídia [Fiorini et al. 2019]. Âncoras podem ser tanto origem quanto destino de elos e também pode ser designadas de forma independente por nós de referência. Usando a noção de âncoras, por exemplo, dada uma base Hyperknowledge (HKBase) é possível obter fragmentos específicos de nós de mídia que satisfaçam uma dada consulta semântica. Tais consultas podem ser definidas numa linguagem de consulta declarativa chamada HyQL (*Hyperknowledge Query Language*) que possui operadores nativos para comparação espacial [Moreno et al. 2018] e temporal [Moreno et al. 2018] de fragmentos.

Vamos agora discutir as ferramentas que permitem utilizar o modelo Hyperknowledge na prática.

1.4.2. Hyperknowledge na prática

Existem duas ferramentas principais para se desenvolver aplicações baseadas em Hyperknowledge: *Hyperknowledge Base* (HKBase) e *Knowledge Explorer System* (KES).

HKBase A *HKBase* é um banco de dados híbrido baseado em Hyperknowledge. Por híbrido, queremos dizer que a base armazena tanto objetos de mídia quanto fatos de um determinado domínio. Além disso, a HKBase foi especificamente projetada para utilizar o modelo Hyperknowledge como modelo de dados interno e API externa. Por razões de compatibilidade, a HKBase também suporta a importação e exportação de RDF e OWL. No caso da importação, os fatos são convertidos para Hyperknowledge antes de serem armazenados no banco de dados subjacente. A Figura 1.9 apresenta a arquitetura geral da HKBase.

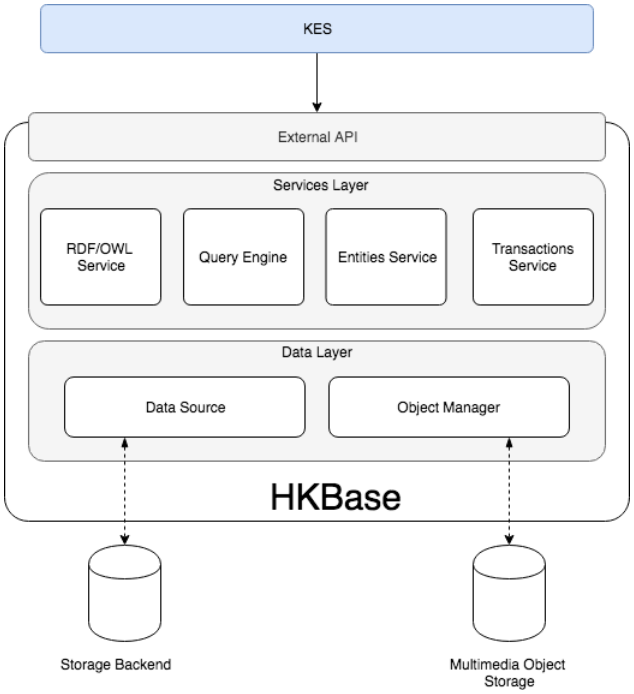


Figura 1.9. Arquitetura geral da HKBase.

A arquitetura da HKBase consiste de três camadas. A primeira delas é a API externa que expõe uma API REST para aplicações. A camada seguinte é a camada de serviços que implementa as funcionalidades principais da HKBase. Nessa camada, está o serviço de RDF/OWL que trata da conversão desses formatos de e para Hyperknowledge. O serviço de *query* trata do processamento de consultas HyQL. O serviço de entidades trata da criação, recuperação, atualização e remoção de entidades Hyperknowledge. E o serviço de transação gerencia requisições que devem executar de forma transacional.

A terceira e última camada da HKBase é a camada de dados que gerencia o acesso de baixo nível às bases de dados subjacentes. O componente *data source* é responsável pela persistência de fatos estruturados. E o componente *object manager* trata da persistência de objetos multimídia.

KES O *KES* (*Knowledge Explorer System*) [Moreno et al. 2018] é uma aplicação construída sobre a HKBase. Usando o KES, os usuários conseguem visualizar e editar o conteúdo armazenado numa base Hyperknowledge. O KES é um sistema colaborativo, isto é, múltiplos usuários podem editar simultaneamente a mesma base. Além disso, o KES possui funcionalidades para importação e manipulação de arquivos RDF e OWL.

A Figura 1.10 apresenta uma captura de tela do KES. O *canvas* principal mostra uma representação baseada em grafos do conteúdo de uma base Hyperknowledge. Além das entidades Hyperknowledge, o KES permite que o conteúdo multimídia seja manipulado diretamente. Por exemplo, a Figura 1.10 mostra que o usuário executou a consulta: "Select Goal where Goal by Neymar". (A base Hyperknowledge, nesse caso, contém fatos sobre o domínio de futebol.) Essa consulta retorna todos os gols marcados pelo jogador chamado "Neymar". Observe que o resultado apresentado pelo KES contém todas as entidades Hyperknowledge que satisfazem a consulta, isto é, todos os nós de mídia que são instâncias de `Goal` e que possuem ao menos um elo que expressa que o gol foi marcado por `Neymar`. O conteúdo desses objetos de mídia é apresentado à direta da tela.

1.5. Sugestões de Leitura

Neste capítulo, apresentamos uma visão geral de IA simbólica e discutimos a sua aplicação à multimídia. Como qualquer introdução, esta é necessariamente incompleta, especialmente por conta da abrangência dos tópicos abordados. Concluímos o capítulo com algumas sugestões de leitura que complementam aquelas apresentadas ao longo do texto.

Logica Uma introdução clássica à lógica matemática em geral é [Enderton 2001]. Já uma referência primária para lógica de descrição é [Baader et al. 2007]. Uma introdução breve porém excelente a DL a partir de $\mathcal{SROIQ}$ é apresentada em [Rudolph 2011]. Para lógicas modais, em especial lógicas temporal, um bom ponto de partida é [Goldblatt 1992]. Para uma discussão de DL e multimídia, recomendamos [Sikos 2017].

Web Semântica Uma discussão seminal da visão da Web Semântica pode ser encontrada em [Berners-Lee et al. 2001]. Uma visão geral dos seus princípios e tecnologias é apresentada em [Hitzler et al. 2010]. Especificamente sobre OWL, um bom ponto de partida é [Hitzler et al. 2014]. E para um tratamento abrangente de ontologias, veja [Staab and Studer 2009].

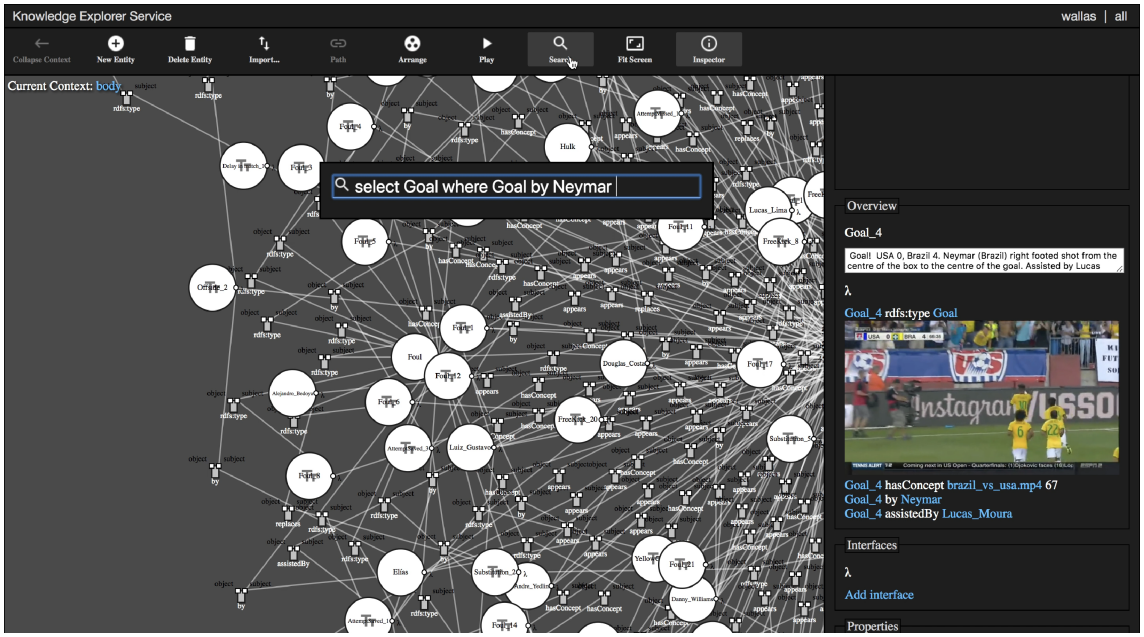


Figura 1.10. Captura de tela do KES.

Hyperknowledge A referência principal sobre Hyperknowledge é [Moreno et al. 2017]. Para uma discussão de aplicações de Hyperknowledge envolvendo interpretação de documentos e raciocínio temporal veja [Moreno et al. 2018] e [Moreno et al. 2018].

Referências

[Angles 2012] Angles, R. (2012). A comparison of current graph database models. In *2012 IEEE 28th International Conference on Data Engineering Workshops*, pages 171–177. IEEE.

[Baader et al. 2007] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2007). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition.

[Baader and Lippmann 2014] Baader, F. and Lippmann, M. (2014). Runtime verification using the temporal description logic $\mathcal{ALC}$-LTL revisited. *J. Appl. Log.*, 12(4):584–613.

[Berners-Lee et al. 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, pages 96—101.

[Borgwardt and Peñaloza 2017] Borgwardt, S. and Peñaloza, R. (2017). Fuzzy description logics – a survey. In *Scalable Uncertainty Management*, pages 31–45. Springer.

[Carothers and Prudhommeaux 2014] Carothers, G. and Prudhommeaux, E. (2014). RDF 1.1 Turtle. Recommendation, W3C. http://www.w3.org/TR/2014/REC-turtle-20140225/.

[Cristiani and Gabrielli 2011] Cristiani, M. and Gabrielli, N. (2011). Practical issues of description logics for spatial reasoning. In *AAAI Spring Symposium, Stanford University, Stanford, 21–23 March, 2011*.

[Enderton 2001] Enderton, H. B. (2001). *A Mathematical Introduction to Logic*. Academic Press, 2nd edition.

[Fiorini et al. 2019] Fiorini, S. R., dos Santos, W. S., Mesquita, R. C., Lima, G. F., and Moreno, M. F. (2019). General fragment model for information artifacts.

[Goldblatt 1992] Goldblatt, R. (1992). *Logics of Time and Computation*. CSLI, 2nd edition.

[Grosof et al. 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 48–57. ACM.

[Harris and Seaborne 2013] Harris, S. and Seaborne, A. (2013). SPARQL 1.1 query language. Recommendation, W3C. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[Hayes and Patel-Schneider 2014] Hayes, P. and Patel-Schneider, P. (2014). RDF 1.1 semantics. Recommendation, W3C. http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[Hitzler et al. 2014] Hitzler, P., Krötzsch, M., adn Peter F. Patel-Schneider, B. P., and Rudolph, S. (2014). OWL 2 Web Ontology Language primer (second edition). Recommendation, W3C. http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[Hitzler et al. 2010] Hitzler, P., Krötzsch, M., and Rudolph, S. (2010). *Foundations of Semantic Web Technologies*. Chapman & Hall.

[Horrocks et al. 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'06, pages 57–67. AAAI Press.

[Hudelot et al. 2015] Hudelot, C., , Atif, J., and Bloch, I. (2015). $\mathcal{ALC}$(F): A new description logic for spatial reasoning in images. In Agapito, L., Bronstein, M. M., and Rother, C., editors, *Computer Vision - ECCV 2014 Workshops*, pages 370–384. Springer.

[Krötzsch et al. 2008] Krötzsch, M., Rudolph, S., and Hitzler, P. (2008). Description logic rules. In *Proceedings of the 18th European Conference on Artificial Intelligence, Patras, February 2008*, pages 80–84. IOS Press.

[Milea et al. 2012] Milea, V., Frasincar, F., and Kaymak, U. (2012). tOWL: A temporal web ontology language. *IEEE Trans. Syst; Man, and Cybern., Part B (Cybernetics)*, 42(1):268–281.

[Moreno et al. 2018] Moreno, M., Santos, R., Mozart, R., Santos, W., and Cerqueira, R. (2018). Assisting seismic image interpretations with hyperknowledge. In *2018 First International Conference on Artificial Intelligence for Industries (AI4I)*, pages 48–51.

[Moreno et al. 2018] Moreno, M., Santos, R., Santos, W., Brandão, R., Carrion, P., and Cerqueira, R. (2018). Handling hyperknowledge representations through an interactive visual approach. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 139–146.

[Moreno et al. 2018] Moreno, M., Santos, R., Santos, W., Silva, R., and Cerqueira, R. (2018). Knowledge bases enrichment with temporal reasoning using hyperknowledge. In *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 125–128.

[Moreno et al. 2018] Moreno, M., Schirmer, L., Bayser, M., Brandão, R., and Cerqueira, R. (2018). Understanding documents with hyperknowledge specifications. In *Proceedings of the ACM Symposium on Document Engineering 2018*, DocEng '18, pages 41:1–41:4. ACM.

[Moreno et al. 2017] Moreno, M. F., Brandao, R., and Cerqueira, R. (2017). Extending hypermedia conceptual models to support hyperknowledge specifications. *Int. J. Semant. Comput.*, 11(1):43–64.

[OWL@Machester 2019] OWL@Machester (2019). List of reasoners. http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/.

[Pfeiffer et al. 2012] Pfeiffer, S., Mannens, E., Troncy, R., and Deursen, D. V. (2012). Media Fragments URI 1.0 (basic). W3C recommendation, W3C. http://www.w3.org/TR/2012/REC-media-frags-20120925/.

[Randell et al. 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, KR '92, pages 165–176. Morgan Kaufmann.

[Robinson et al. 2013] Robinson, I., Webber, J., and Eifrem, E. (2013). *Graph Databases*. O'Reilly.

[Rudolph 2011] Rudolph, S. (2011). *Foundations of Description Logics*, pages 76–136. Springer.

[Schmidt-Schauß and Smolka 1991] Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26.

[Sikos 2017] Sikos, L. F. (2017). *Description Logics in Multimedia Reasoning*. Springer.

[Soares and Rodrigues 2005] Soares, L. F. G. and Rodrigues, R. F. (2005). Nested Context Model 3.0 part 1: NCM core. Monographs in computer science, Informatics Department, PUC-Rio, Rio de Janeiro, Brazil.

[Staab and Studer 2009] Staab, S. and Studer, R., editors (2009). *Handbook on Ontologies*. Springer-Verlag, 2nd edition.

[W3C OWL WG 2012] W3C OWL WG (2012). OWL 2 Web Ontology Language document overview (second edition). Recommendation, W3C. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[W3C Semantic Web Wiki 2019] W3C Semantic Web Wiki (2019). Semantic web development tools. https://www.w3.org/2001/sw/wiki/Tools.

[Wang et al. 2014] Wang, Y., Chang, L., Li, F., and Gu, T. (2014). Verification of branch-time property based on dynamic description logic. In *Intelligent Information Processing VII*, pages 161–170. Springer.

[Wessel 2003] Wessel, M. (2003). Qualitative spatial reasoning with the $\mathcal{ALCI}_{\mathrm{RCC}}$ family: First results and unanswered questions. Memo 324/03, University of Hamburg. https://kogs-www.informatik.uni-hamburg.de/publikationen/pub-wessel/report7.pdf.

[Wood et al. 2014] Wood, D., Lanthaler, M., and Cyganiak, R. (2014). RDF 1.1 concepts and abstract syntax. Recommendation, W3C. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[Zolin 2019] Zolin, E. (2019). Complexity of reasoning in description logics. http://www.cs.man.ac.uk/~ezolin/dl/.