Consultando documentos XML utilizando inferência

Diego Mury Gomes de Lima[1], Carla Delgado[1], Leonardo Murta[2], Vanessa Braganholo[2]

[1] Universidade Federal do Rio de Janeiro
[2] Universidade Federal Fluminense
dmglima@ufrj.br, carla@dcc.ufrj.br, {leomurta, vanessa}@ic.uff.br

Resumo. O expressivo aumento do volume de informações armazenadas em documentos XML estimula o aprimoramento de métodos capazes de obter respostas satisfatórias para consultas cada vez mais complexas. Enquanto as abordagens de consulta existentes atualmente analisam apenas as informações definidas explicitamente no documento, este trabalho apresenta uma abordagem capaz de processar conhecimento implícito através de inferências guiadas por regras derivadas automaticamente do esquema XML e por regras configuradas previamente pelo usuário. Tal fato pode proporcionar um aumento significativo nas possibilidades de consulta, permitindo fácil acesso a novas informações e possibilitando que consultas mais elaboradas obtenham os resultados esperados.

Categories and Subject Descriptors: H. Information Systems [**H.m. Miscellaneous**]: Query processing

Palavras-chave**:** Consulta em documentos XML, Inferência, Prolog.

1. INTRODUÇÃO

A linguagem XML consolidou-se rapidamente como um padrão universal para troca de informações na Web. O surgimento de grandes coleções de documentos neste formato evidencia a necessidade de se aperfeiçoar os métodos de consulta, para que sejam capazes de manipular o grande volume de dados armazenados de forma eficiente e abrangente [Kotsakis 2002].

A maioria das linguagens de consulta em documentos XML disponíveis atualmente, como por exemplo, XPath [Clark e DeRose 2010] e XQuery [Boag et al. 2010], obtêm suas respostas a partir do caminho percorrido na árvore XML. Essas abordagens navegam da raiz do documento ao elemento mais interno especificado na consulta, e aplicam os filtros definidos para processar as informações e obter o resultado. A consulta é realizada apenas sobre os dados explícitos na estrutura do documento, ignorando quaisquer informações implícitas que possam vir a ser obtidas a partir desses dados.

Estimulado pelo poder de expressividade de XML e pelo grande volume de informações representadas atualmente neste formato, a abordagem proposta neste trabalho sugere um aprimoramento nos mecanismos de consulta em documentos XML. Isso é obtido através do uso de máquinas de inferência com o objetivo de aumentar as possibilidades de consulta, viabilizando a obtenção de resultados mais elaborados. Como exemplo, um documento XML que define que João é pai de Pedro e Pedro é pai de Paulo, não possui informações suficientes para responder consultas sobre avô. Porém, uma vez que a regra que define avô como pai do pai (ou pai da mãe) é inserida pelo usuário, o sistema estaria apto a responder que João é avô de Paulo.

Para realizar inferências, precisamos traduzir os dados contidos no documento XML para alguma linguagem que nos forneça tal capacidade. As linguagens RuleML [Boley 2001], Datalog [Gallaire e Minker 1978] e Prolog [Colmerauer et al. 1973] possibilitam inferências e foram estudadas neste trabalho. O fato de Datalog não permitir termos complexos como argumento do predicado, e possuir restrições no uso de negação e recursividade, tornou-a incompatível com a abordagem proposta. O uso de RuleML também foi descartado uma vez que esta utiliza Datalog como base das inferências.

Consequentemente, optamos por adotar uma máquina de inferência Prolog para realizar a tarefa de interpretar dados implícitos e definir novas informações a partir de documentos XML.

O restante deste trabalho está organizado em três seções. A Seção 2 ilustra a proposta deste artigo e descreve o método de tradução da linguagem XML para uma que seja interpretável por uma máquina de inferência. Na Seção 3 são discutidos os trabalhos relacionados. Por fim a Seção 4 conclui o artigo apresentando as considerações finais do trabalho.

2. UM MÉTODO DE CONSULTA CAPAZ DE INFERIR DADOS

A abordagem proposta neste trabalho processa os dados contidos na estrutura do documento XML com o objetivo de inferir novas informações, possibilitando a obtenção de respostas complexas a partir de informações explícitas no documento ou deduzidas automaticamente através de regras (informações implícitas). Para que isso seja possível, esquemas que definem o vocabulário a ser seguido são acoplados aos documentos XML assim como novas regras, que ampliam as informações na base de conhecimento e estendem as possibilidades de consulta, possibilitando a inferência de novas informações.

O nosso método é dividido em três etapas: configuração das regras, geração dos fatos e consulta. A Figura 1 ilustra tais etapas e suas fases. Durante a etapa de configuração das regras, o usuário deve informar ao sistema os esquemas que validam os documentos XML que serão alvo das consultas. Estes esquemas são analisados por um tradutor de esquemas que gera regras Prolog (denominadas regras automáticas) a partir dos dados processados, além de possíveis definições de predicados. Utilizando estas definições, que são as assinaturas dos predicados Prolog, um usuário especialista pode gerar novas regras, manualmente. No final da primeira etapa, a base de conhecimento possuirá regras Prolog que foram derivadas automaticamente pelo método aqui proposto e regras definidas manualmente, a critério do engenheiro de conhecimento (usuário especialista). Tal base de conhecimento será armazenada como um novo projeto e poderá ser carregada posteriormente quando for necessário. Isso significa que, uma vez realizada a etapa de configuração, o usuário pode utilizar a etapa de consulta sempre que desejar.

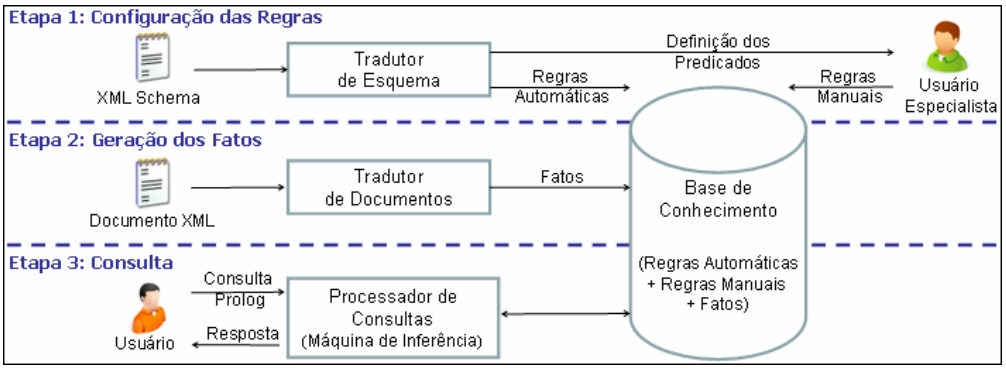

Figura 1: Etapas da abordagem proposta

A segunda etapa utiliza o tradutor de documentos proposto para derivar fatos Prolog a partir dos documentos XML. Após esta fase, todas as informações contidas no documento XML e nos esquemas estão definidas em linguagem Prolog (como fatos e regras, respectivamente), juntamente com as regras manuais. Essas informações constituem a base de conhecimento e são passíveis de inferência. Por fim, na terceira e última etapa o usuário está apto a realizar consultas, que são processadas pela máquina de inferência Prolog. Os resultados são então apresentados ao usuário, que pode submeter mais consultas uma vez que o ambiente encontra-se todo preparado.

As Seções 2.1, 2.2 e 2.3 explicam as regras de tradução das etapas 1 e 2 da Figura 1. Por questões didáticas, a tradução dos documentos XML será explicada antes da tradução dos esquemas. Essa inversão foi proposital para permitir a familiarização com a sintaxe adotada para definir os fatos Prolog antes de lidar com as regras que serão geradas. Na prática, esse processo é transparente para o usuário. Desta forma, a Seção 2.1 explica a tradução dos documentos XML para fatos Prolog (segunda etapa). As Seções 2.2 e 2.3 explicam a tradução dos esquemas XML para regras automáticas e manuais, respectivamente.

2.1 Tradução de documentos XML para Prolog

O processo de tradução gera vários fatos a partir de um único documento XML, transformando os elementos em predicados e seus conteúdos em constantes. Para relacionar predicados distintos, um identificador (uma constante Prolog) é acrescentado como parâmetro, caracterizando que há correspondência entre esses dados no documento XML (por exemplo, relação de pai/filho). Todo o processo é guiado por cinco tipos de tradução que são listadas a seguir. Um exemplo do resultado da aplicação dessas traduções é ilustrado na Figura 2.

Documento XML		**Tradução Prolog**
<cliente>	1.	cliente(id1).
<historico>	2.	historico(id1, id2).
<operacao>compra</operacao>	3.	operacao(id2, 'compra').
realizada em	4.	xml/elementoMisto(id2, 'realizada em').
<data>23/02/1990</data>	5.	data(id2, '23/02/1990').
<valor moeda="R$">586,00</valor>	6.	valor(id2, id3, '586,00').
</historico>	7.	moeda(id3, 'R$').
</cliente>		

Figura 2: Exemplo de tradução de um documento XML

Tradução da raiz do documento (Figura 2, linha 1). A raiz do documento XML deve ser tratada de forma especial, pois este é o único elemento do documento que não possui pai. Na grande maioria dos casos a raiz é um elemento composto (a não ser que se trate de um caso raro onde o documento contenha apenas um único elemento simples). Sendo assim, o resultado da tradução de raízes compostas é um fato com o nome do elemento e argumento único igual a um identificador gerado pela aplicação com o objetivo de estabelecer vínculo com seus elementos filhos.

Tradução de elementos simples sem atributo (Figura 2, linhas 3 e 5). Elementos simples sem atributos possuem o seu conteúdo textual como único filho na árvore XML, possibilitando que sua tradução possa ter como resultado um fato com nome igual ao nome do elemento e argumentos iguais ao identificador do elemento pai e o conteúdo do elemento corrente.

Tradução de elemento simples com atributo (Figura 2, linhas 6 e 7). Os atributos são considerados filhos do elemento que os contém. Um novo identificador é gerado para que os elementos filhos (atributos) possam referenciar seu pai. O resultado é composto por um fato que representa o elemento, mais um fato para cada atributo existente: o primeiro possui o nome do elemento e três argumentos: o identificador do seu elemento pai, um novo identificador gerado para representá-lo e o seu conteúdo textual. Os outros fatos possuem o nome do atributo e argumentos iguais ao identificador de seu pai (que representa o elemento que possui o atributo) e ao valor do atributo correspondente.

Tradução de elemento complexo (Figura 2, linha 2). Elementos complexos possuem outros elementos como filhos. Portanto, um novo identificador deve ser gerado para relacionar os filhos ao pai. Consequentemente, o resultado da tradução é a geração de um fato com o nome do elemento composto e dois argumentos: o identificador do seu elemento pai e um novo identificador gerado para referenciá-lo. Seus filhos são traduzidos de acordo com a regra de tradução adequada às suas características (simples, composto, misto), utilizando o identificador referente ao elemento complexo (seu elemento pai) como identificador do elemento pai.

Tradução de elemento misto (Figura 2, linha 4). Elementos mistos são traduzidos de maneira semelhante aos elementos complexos, exceto pela criação de fatos "xml/elementoMisto" para os filhos texto do elemento (uma vez que eles não possuem nomes). Essa nomenclatura foi adotada levando em consideração o fato da especificação do XML não permitir que o nome dos elementos contenha o caractere "/", impedindo, consequentemente, que tal termo seja resultado de alguma tradução que não seja o caso de elemento misto. Portanto, seus filhos do tipo texto são representados por fatos "xml/elementoMisto" com argumentos iguais ao identificador do elemento pai (o elemento complexo) e o seu conteúdo textual. Já seus outros filhos são traduzidos de acordo com a regra de tradução adequada às suas características (simples, composto, misto).

2.2 Definição das regras automáticas

Nosso método de tradução gera regras Prolog referentes aos delimitadores de grupo dos elementos complexos presentes no esquema, visto que estes definem a estrutura que os documentos XML válidos devem possuir. A seguir é apresentado como o método de tradução processa as principais definições encontradas nos esquemas e obtém as regras automáticas em linguagem Prolog.

2.2.1 Elementos complexos com filhos elementos simples

Os casos mais simples de regras automáticas são derivados de elementos complexos que possuem apenas elementos simples como filhos. Neste caso são utilizadas as regras básicas de tradução para cada um dos delimitadores de grupo: *sequence*, *choice* e *all*.

Tradução do *sequence*. Dentre os delimitadores de grupo existentes, o *sequence* é o mais utilizado [Laender et al. 2009]. Sua especificação sugere que o elemento complexo que o emprega deve seguir a estrutura definida no esquema, respeitando a ordem e cardinalidade. O método de tradução do esquema produz uma regra que representa a estrutura estabelecida pelo *sequence* analisado. A Figura 3 (lado esquerdo) mostra um trecho de um esquema XML que possui o delimitador de grupo *sequence*.

Esquema XML

```
<xs:element name="endereco" type="tEndereco"/>
<xs:complexType name="tEndereco">
  <xs:sequence>
    <xs:element name="logradouro" type="xs:string"/>
    <xs:element name="numero" type="xs:string"/>
  </xs:sequence></xs:complexType>
```

Tradução Prolog

```
endereco(ID, LOGRADOURO, NUMERO) :-
   logradouro(ID, LOGRADOURO),
   numero(ID, NUMERO).
```

Figura 3: Exemplo de ocorrência de *sequence* e sua tradução para Prolog.

A regra Prolog gerada neste caso possui cabeça referente ao elemento complexo (no exemplo da Figura 3, "endereco") com os seguintes argumentos: uma variável Prolog representando o identificador da regra "ID", responsável por vincular a regra a outros termos definidos em linguagem Prolog (fatos e regras), e uma variável para cada filho que possuir ("LOGRADOURO" e "NUMERO", no exemplo), com o objetivo de obter o valor destes através da associação do identificador da regra com os fatos (ou regras) referentes a cada um destes filhos (vale lembrar que os fatos são derivados diretamente dos documentos XML). Já o corpo da regra possui um fato referente a cada filho, com argumentos iguais ao identificador da regra (para estabelecer relação com a cabeça da regra) e à variável correspondente. A Figura 3 (lado direito) ilustra a regra automática obtida após a tradução do trecho do esquema demonstrado na Figura 3 (lado esquerdo).

Tradução do *choice*. A especificação do delimitador de grupo *choice* permite que o elemento complexo que o contém possua como filho apenas uma das opções definidas em sua estrutura. Sendo assim, o método de tradução cria uma regra para cada opção do *choice*, contemplando cada uma das possíveis estruturas válidas. A Figura 4 exibe um trecho de um esquema com ocorrência do delimitador de grupo *choice*.

Esquema XML	Tradução Prolog
<xs:element name="unidade" type="tUnidade"/> <xs:complexType name="tUnidade"> <xs:choice> <xs:element name="area" type="xs:string"/> <xs:element name="volume" type="xs:string"/> </xs:choice></xs:complexType>	unidade(ID, area, AREA) :- area(ID, AREA). unidade(ID, volume, VOLUME) :- volume(ID, VOLUME).

Figura 4: Exemplo de ocorrência de *choice* e sua tradução para Prolog.

O resultado da tradução são regras Prolog referentes às possíveis opções de estrutura que o elemento complexo pode possuir, conforme ilustrado na Figura 4, lado direito. A cabeça da regra será formada pelo elemento complexo "unidade", possuindo como argumentos uma variável representando o identificador da regra "ID", um identificador da opção (igual ao nome do elemento), responsável por indicar qual das opções foi adotada, e uma variável referente ao elemento correspondente. Na Figura 4, "area" e "volume" são respectivamente os identificadores de opção, assim como "AREA" e "VOLUME" são as variáveis. O corpo da regra possuirá o fato referente ao filho correspondente à regra com argumentos iguais ao identificador da regra (para estabelecer relação com a cabeça da regra) e à variável apropriada.

Tradução do *all.* O delimitador de grupo *all* permite que o elemento complexo contenha qualquer combinação das opções definidas em sua estrutura (em qualquer ordem), possibilitando a ocorrência de vários elementos diferentes. Durante a tradução, o elemento que possuir o delimitador de grupo se torna a cabeça da regra, que será definida com os seguintes parâmetros: um identificador da regra "ID", um identificador da opção e uma variável que representa o filho correspondente à opção escolhida. Mais uma vez, a regra irá relacionar os possíveis filhos através do seu identificador. A Figura 5 (lado esquerdo) demonstra um trecho de um esquema que possui um elemento composto *all*, enquanto a Figura 5 (lado direito) ilustra o resultado da tradução deste mesmo trecho utilizando o algoritmo proposto.

As especificações do delimitador de grupo *all* definem que ele só pode ser empregado como grupo mais externo de qualquer modelo de conteúdo, e ainda que seus filhos devem ser todos elementos. Logo, não pode haver estruturas complexas contidas ou que contenham o delimitador de grupo *all*. Outra restrição impede que os elementos filhos de *all* tenham cardinalidade maior que 1, limitando para 0 ou 1 os valores permitidos para *minOccurs* e indicando que o valor de *maxOccurs* deve ser 1.

Esquema XML	Tradução Prolog
<xs:element name="contato" type="tContato"/> <xs:complexType name="tContato"> <xs:all> <xs:element name="telefone" type="xs:string"/> <xs:element name="email" type="xs:string"/> </xs:all></xs:complexType>	contato(ID, telefone, TELEFONE) :- telefone(ID, TELEFONE). contato(ID, email, EMAIL) :- email(ID, EMAIL).

Figura 5: Exemplo de ocorrência de *all* e sua tradução para Prolog.

2.2.2 Elementos complexos com filhos elementos complexos

O método de tradução de elementos complexos com filhos complexos utiliza os mesmos princípios das regras básicas, mencionadas anteriormente, para gerar as regras automáticas. Porém, o fato dos filhos não serem mais elementos simples torna necessário alguns cuidados para que as regras geradas representem fielmente as estruturas válidas possíveis existentes nos documentos XML.

Tradução de *sequence* com filho *choice.* A tradução do delimitador de grupo *sequence* que possui um filho *choice* utiliza regras definidas através da união da tradução de ambos os grupos. O trecho de esquema representado pela Figura 6 indica que o elemento composto "cliente" possuirá sempre dois filhos, sendo que o primeiro poderá ser "cpf" ou "cnpj" e o segundo será sempre "telefone". Na prática, seria como se houvesse dois *sequences* diferentes, um contendo os filhos "cpf" e "telefone" e outro

"cnpj" e "telefone". Portanto, a tradução define duas regras distintas, uma para cada opção possível. Vale lembrar que a cabeça da regra irá conter também a opção que foi escolhida, para facilitar futuras consultas. A Figura 6, lado direito, ilustra o resultado da tradução do trecho do esquema contido na Figura 6, lado esquerdo.

Esquema XML

```
<xs:element name="cliente" type="tCliente"/>
<xs:complexType name=" tCliente ">
  <xs:sequence>
    <xs:choice>
      <xs:element name="cpf" type="xs:string"/>
      <xs:element name="cnpj" type="xs:string"/>
    </xs:choice>
    <xs:element name="telefone" type="xs:string"/>
  </xs:sequence></xs:complexType>
```

Tradução Prolog

```
cliente(ID, cpf, CPF, TELEFONE) :-
   cpf(ID, CPF),
   telefone(ID, TELEFONE).
cliente(ID, cnpj, CNPJ, TELEFONE) :-
   cnpj(ID, CNPJ),
   telefone(ID, TELEFONE).
```

Figura 6: *Sequence* com filho *choice* e sua tradução para Prolog.

Tradução de *choice* com filho *sequence.* Um elemento composto delimitado por *choice* pode possuir filhos *sequence*. Neste caso o método de tradução associa uma constante Prolog (s1, ..., sn) para cada filho *sequence*, com o objetivo de aplicar estas constantes como identificador de opção do *choice.* Este processo é necessário uma vez que não haverá elemento diretamente correspondente a este *sequence* para que seu nome seja utilizado como identificador de opção. O restante do processo é análogo à tradução do *choice* com filhos elementos simples.

O fato de não termos utilizado nenhum método de tradução de XML para Prolog implementado anteriormente [Coelho e Florido 2003, Seipel 2002, Wielemaker 2005] vem da necessidade de obtenção de fatos granulares. A informação granular permite formas de manipulação mais diversas, e aumenta as possibilidades de consultas.

Vale ressaltar que a função das regras automáticas é facilitar o processo de inserção das regras manuais através da geração automática de regras relacionadas à estrutura do documento XML. Portanto, o fato do documento a ser consultado não possuir um esquema definido não inviabiliza a utilização do método, apenas indica que possivelmente mais regras tenham que ser inseridas manualmente.

2.3 Definição das regras manuais

As regras manuais são regras mais elaboradas que não podem ser derivadas automaticamente a partir dos esquemas. A abordagem proposta permite que elas sejam inseridas por um usuário especialista durante a etapa de configuração das regras, com o objetivo de tornar possível a realização de inferências mais complexas ou obter respostas que não podem ser obtidas somente com as informações contidas no documento e esquema XML.

Como um exemplo, o esquema definido na Figura 7 não possui qualquer informação a respeito de sociedade. Porém, pode ser necessário obter tal relação entre os clientes cadastrados. Analisando a definição dos predicados gerada a partir da tradução do esquema, um usuário especialista (com conhecimento no domínio e em Prolog) pode criar regras que permitam inferir informações além das que foram explicitamente traduzidas do XML. A semântica por trás de um conjunto de tags XML (por exemplo, o fato de que dois proprietários de uma mesma empresa são sócios) não é capturada automaticamente por um esquema de tradução, mas pode ser percebida por um engenheiro de conhecimento (usuário especialista). Ele pode avaliar se ela enriquece a base e decidir incluí-la na forma de uma regra. Vale lembrar que isso ocorre somente na etapa de configuração, e este usuário pode não ser o que realizará a consulta (ele apenas irá configurar o ambiente).

A Figura 7, lado direito, ilustra as regras manuais que devem ser inseridas por um usuário especialista para tornar possível obter a relação de sociedade entre os clientes durante a etapa de consulta. Durante a etapa de consulta, o usuário obtém informações sobre a relação de sociedade entre clientes através do

termo de consulta: “socio(X, Y).”. Através de XQuery a mesma informação seria obtida através de uma consulta muito mais complexa (Figura 8).

Esquema XML

```
<xs:element name="cliente" type="tCliente"/>
<xs:complexType name="tCliente">
  <xs:sequence>
    <xs:choice>
      <xs:sequence>
        <xs:element name="razao_social" type="xs:string"/>
        <xs:element name="cnpj" type="xs:string"/>
        <xs:element name="cpfProprietario" type="xs:string"/>
      </xs:sequence>
      <xs:sequence>
        <xs:element name="nome" type="xs:string"/>
        <xs:element name="cpf" type="xs:string"/>
      </xs:sequence>
    </xs:choice></xs:sequence></xs:complexType>
```

Regra Manual Prolog

```
socio(Nome1, Nome2) :-
   nome(A, Nome1),
   cpf(A, B),
   cpfProprietario (C, B),
   cnpj(C, D),
   nome(E, Nome2),
   cpf(E, F),
   cpfProprietario (G, F),
   cnpj(G, D),
   Nome1 \= Nome2.
```

Figura 7: Exemplo de XML *Schema* e regra manual Prolog.

```
for $cliente1 in doc("teste2XML.xml")/clientes/cliente,
  $cliente2 in doc("teste2XML.xml")/clientes/cliente,
  $empresa1 in doc("teste2XML.xml")/clientes/cliente,
  $empresa2 in doc("teste2XML.xml")/clientes/cliente
where $cliente1/cpf = $empresa1/cpf_do_proprietario
  and $cliente1/cpf != $cliente2/cpf
  and $cliente2/cpf = $empresa2/cpf_do_proprietario
  and $empresa1/cnpj = $empresa2/cnpj
return  <socio>{$cliente1/nome/text()} e socio de {$cliente2/nome/text()}</socio>
```

Figura 8: Consulta XQuery sobre sociedade de clientes

O uso da Máquina Prolog proporciona a reutilização de regras com o objetivo de elaborar consultas mais complexas. A Figura 9 mostra quatro regras Prolog que tornam possível a obtenção de respostas a respeito de sociedade proveniente de herança (sócios herdeiros) no exemplo tratado. Note que as regras “socioHerdeiro” reutilizam “socio” e “herdeiro”, previamente definidas. Essa facilidade não é permitida em XQuery, tendo o usuário que definir todas as relações desejadas a cada consulta.

```
herdeiro(Herdeiro, Antecessor) :- filho(Herdeiro, Antecessor).
herdeiro(Herdeiro, Antecessor1) :- filho(Herdeiro, Antecessor2),
    herdeiro(Antecessor2, Antecessor1).
socioHerdeiro(Nome1, Nome2) :-  herdeiro(Nome1, Antecessor1), socio(Antecessor1,Nome2).
socioHerdeiro(Nome1, Nome2) :- herdeiro(Nome1, Antecessor1), herdeiro(Nome2, Antecessor2),
    socio(Antecessor1, Antecessor2).
```

Figura 9: Exemplo de regra manual socioHerdeiro

3. TRABALHOS RELACIONADOS

A integração da Programação em Lógica com a área de Banco de Dados foi estudada com a intenção de facilitar a representação e manipulação de dados. Niemi e Jarvelin [1991] propuzeram a adoção de Prolog para representar a base de conhecimento proveniente de um banco de dados relacional. Com a disseminação da Internet e o surgimento de um grande volume de dados representados em XML, nasceu o interesse de integração da Programação em Lógica com o Processamento de dados XML [Almendros-Jiménez et al. 2008, Bailey et al. 2005, Seipel 2002].

Assim como o nosso trabalho, Almendros-Jiménez et al. [2008] propõem a tradução de documentos XML em fatos, e de esquemas em regras Prolog. O objetivo do seu trabalho é implementar a linguagem XPath em Programação Lógica. Seu método, porém, traduz apenas o caminho (*path*) necessário para responder a consulta submetida, enquanto nosso trabalho traduz todo o documento, possibilita a inserção de novas regras e permite consultas em linguagem Prolog.

Coelho e Florido [2003] realizam inferência de tipos em um documento XML com a utilização de programação em lógica. Essa abordagem visa deduzir o tipo de dado dos elementos contidos na estrutura

e utiliza uma DTD para auxiliar o processo de tradução, enquanto nós adotamos o uso de XML *Schemas*. Assim como o método aqui proposto, Coelho utiliza Prolog para realizar inferências sobre os dados armazenados em XML. Porém, seu trabalho infere tipos de dados ao invés de conteúdo.

4. CONCLUSÃO

Este trabalho propõe um método de consulta em documentos XML que não se restringe somente aos seus dados explícitos. Com o auxílio de XML *Schemas* previamente carregados é possível que as regras que devem ser seguidas pelas instâncias (fatos da base de dados) também sejam consultadas e possibilitem a inferência de informações implícitas, enriquecendo o resultado das consultas. A observação das regras impostas em esquemas que validam a base de conhecimento XML pode contribuir para a criação de um conjunto de regras e fatos mais elaborado, aumentando as possibilidades de inferência.

Embora as informações iniciais estejam representadas em documentos XML, os usuários devem realizar as consultas em linguagem Prolog uma vez que todos os dados serão traduzidos para linguagem lógica com o objetivo de possibilitar a realização de inferências. Ainda que grande parte dos usuários interessados em submeter consultas a documentos XML sejam de áreas tecnológicas e possivelmente estejam familiarizados com Prolog, esse empecilho pode ser solucionado com a adoção de uma interface gráfica que gere consultas Prolog a partir de uma seleção de opções em alto nível [Costa e Braganholo 2010]. Como trabalhos futuros, pretendemos fazer experimentos com usuários com níveis de experiência diferentes para avaliar a adequação da proposta.

REFERÊNCIAS

Almendros-Jiménez, J. M., Becerra-Terón, A., Enciso-Banos, F. J., (2008), "Querying XML documents in logic programming", *Theory and Practice of Logic Programming*, v. 8, n. 03, p. 323–361.

Bailey, J., Bry, F., Furche, T., Schaffert, S., (2005), "Web and semantic web query languages: A survey". In: *Proc. of Reasoning Web, First International Summer School, LNCS 3564. Heidelberg, Germany*, p. 35–133

Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., Siméon, J., (2010). XQuery 1.0: An XML Query Language (Second Edition). Disponível em: <http://www.w3.org/TR/xquery/>. Acesso em: 7 abr 2010.

Boley, H., (2001), "The rule markup language: RDF-XML data model, XML schema hierarchy, and XSL transformations". In: *Web Knowledge Management and Decision Support at the 14th International Conference on Applications of Prolog, INAP 2001*

Clark, J., DeRose, S., (2010). XML Path Language (XPath). Disponível em: <http://www.w3.org/TR/xpath/>. Acesso em: 7 abr 2010.

Coelho, J., Florido, M., (2003), "Type-based XML processing in logic programming", *Practical Aspects of Declarative Languages*, p. 273–285.

Colmerauer, A., Kanoui, H., Pasero, R., Roussel, P., (1973), "Un systeme de communication homme-machine en francais", *Groupe de Recherche en Intelligence Artificielle, Université d'Aix- Marseilles, France*, v. 3, p. 99.

Costa, P., Braganholo, V., (2010), "Uma nova abordagem para consulta a dados de proveniência". In: *Workshop de Teses e Dissertações em Banco de Dados (WTDBD), 2010, Belo Horizonte, MG.Workshop de Teses e Dissertações em Banco de Dados (WTDBD). Porto Alegre, RS: SBC, 2010.*, p. 1-6

Gallaire, H., Minker, J., (1978), "Logic and data bases". *Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, 1977*, Advances in Data Base Theory, Plenum Press, New York.

Kotsakis, E., (2002), "Structured information retrieval in XML documents". In: *Proceedings of the 2002 ACM symposium on Applied computing*, p. 663–667

Laender, A. H. F., Moro, M. M., Nascimento, C., Martins, P., (2009), "An X-ray on Web-available XML Schemas", *ACM SIGMOD Record*, v. 38, n. 1, p. 37–42.

Niemi, T., Jarvelin, K., (1991), "Prolog-based meta-rules for relational database representation and manipulation", *Software Engineering, IEEE Transactions on*, v. 17, n. 8, p. 762–788.

Seipel, D., (2002), "Processing XML-documents in Prolog". In: *Proc. Workshop Logische Programmierung.(2002). Technische Universität Dresden, Dresden, Germany.*

Wielemaker, J., (2005), "SWI-Prolog SGML/XML Parser", *SWI, University of Amsterdam, Roetersstraat*, v. 15, p. 1018.