A SPARQL Query Engine over Web Ontologies using Contextual Logic Programming

Nuno Lopes Salvador Abreu

Universidade de Évora

CENTRIA-UNL

15 de Fevereiro de 2007

Motivação e trabalho proposto

- Programação em Lógica Contextual (CxLP) e ISCO

Motivação e trabalho proposto

- Programação em Lógica Contextual (CxLP) e ISCO
- Ontologias Web

Motivação e trabalho proposto

- Programação em Lógica Contextual (CxLP) e ISCO
- Ontologias Web

Representação para ontologias em CxLP

- Integração com o ISCO e outras fontes de dados
- Permitir fazer queries internas (top-level)

Motivação e trabalho proposto

- Programação em Lógica Contextual (CxLP) e ISCO
- Ontologias Web

Representação para ontologias em CxLP

- Integração com o ISCO e outras fontes de dados
- Permitir fazer queries internas (top-level)

Com base na representação:

- Permitir interrogar a representação com SPARQL

Programação em Lógica Contextual

Contextual Logic Programming, L. Monteiro and A. Porto (1989)

- Módulos para Programação em Lógica
- módulo (unit)
- contexto (sequência de units)

Programação em Lógica Contextual

Contextual Logic Programming, L. Monteiro and A. Porto (1989)

- Módulos para Programação em Lógica
- módulo (unit)
- contexto (sequência de units)

GNU Prolog/CX

- implementação de CxLP
- apresenta argumentos de unit que se comportam como uma variável global à unit
- permitem parametrizar um contexto/unit

GNU Prolog/CX

- Units:

```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

GNU Prolog/CX

- Units:

```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

- Contextos:

```
foo(b) :> item(X).
```

GNU Prolog/CX

- Units:

```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

- Contextos:

```
foo(b) :> item(X).
```

```
X = b
```

GNU Prolog/CX

- Units:

```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

- Contextos:

```
foo(b) :> item(X).
```

```
X = b
```

```
foo(1) :> bar(a) :> item(X).
```

GNU Prolog/CX

- Units:

```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

- Contextos:

```
foo(b) :> item(X).                X = b

foo(1) :> bar(a) :> item(X).      X = a
```

GNU Prolog/CX

- Units:

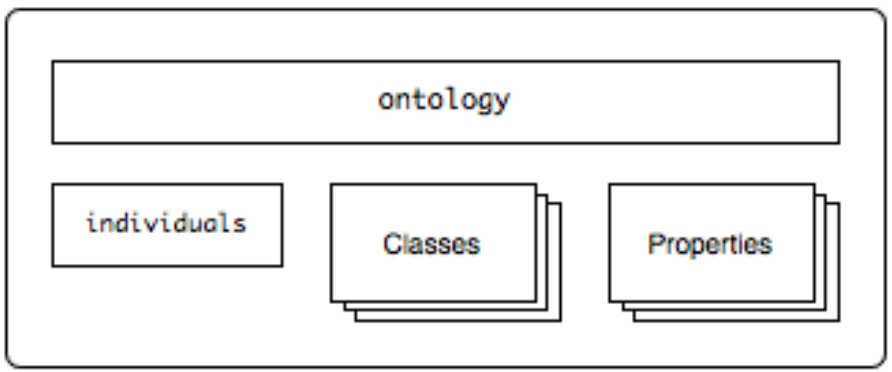
```
:- unit(foo(A)).

item(A).
```

```
:- unit(bar(B)).

item(B).
item(A) :- :^ item(A).
```

- Contextos:

```
foo(b) :> item(X).                  X = b

foo(1) :> bar(a) :> item(X).        X = a  ;
                                    X = 1
```

Representação de Ontologias



Ontologias são representadas através de units:

- uma unit que lista as classes e propriedades da ontologia;
- uma unit que representa dos individuos;
- uma unit para cada classe OWL;
- uma unit para cada propriedade

Interrogar uma ontologia

```
| ?- 'ClassName' /> item(A).
A = 'IndividualName'
```

Interrogar uma ontologia

```
| ?- 'ClassName' /> item(A).
A = 'IndividualName'
```

Para aceder ao valor das propriedades dos indivíduos:

```
| ?- 'IceWine' /> hasFlavor(F) :> hasBody(B) :>
  item(I).
B = 'Medium'
F = 'Moderate'
I = 'SelaksIceWine' ?
```

Query examples

James Bailey, François Bry, Tim Furche, and Sebastian Schaffert. Web and semantic web query languages: A survey. Reasoning Web (2005).

Query examples

James Bailey, François Bry, Tim Furche, and Sebastian Schaffert. Web and semantic web query languages: A survey. Reasoning Web (2005).

```
PREFIX   books: http://example.org/books#
PREFIX   rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT   ?essay, ?author, ?authorName, ?translator
FROM     http://example.org/books
WHERE    (?essay books:author       ?author),
         (?author books:authorName ?authorName)
OPTIONAL (?essay books:translator ?translator)
```

```
| ?- /> author(AUTHOR) :> item(ESSAY),
     /> authorName(AUTHORNAME) :> item(AUTHOR),
     /> optional(translator(TRANSLATOR)) :> item(ESSAY).
```

```
PREFIX    books: http://example.org/books#
CONSTRUCT (?x books:co-author ?y)
FROM      http://example.org/books
WHERE     (?book books:author ?x)
          (?book books:author ?y)
AND       (?x neq ?y)
```

```
| ?- /> author(X) :> item(BOOK),
     /> author(Y) :> item(BOOK),
     X \= Y,
     I = coauthor(X,Y).
```

Resposta a interrogações SPARQL

1. parser SPARQL escrito em Flex e Bison

Resposta a interrogações SPARQL

1. parser SPARQL escrito em Flex e Bison
2. gera um contexto que representa a interrogação

Resposta a interrogações SPARQL

1. parser SPARQL escrito em Flex e Bison
2. gera um contexto que representa a interrogação
3. o contexto é executado para obter os resultados

Resposta a interrogações SPARQL

1. parser SPARQL escrito em Flex e Bison
2. gera um contexto que representa a interrogação
3. o contexto é executado para obter os resultados
4. e formatado de acordo com as especificações XML do protocolo SPARQL

```
SELECT
        ?flavor ?body
WHERE {
   ?t :hasFlavor    ?flavor  .
   ?t :hasBody      ?body  .
}
```

- Variável SPARQL representada como uma variável Prolog
- estrutura do contexto parecida com a estrutura da query

```
SELECT
       ?flavor ?body
WHERE {
   ?t :hasFlavor    ?flavor  .
   ?t :hasBody      ?body  .
}
```

- Variável SPARQL representada como uma variável Prolog
- estrutura do contexto parecida com a estrutura da query

```
[where([set([
     triple(A,hasFlavor,B),
     triple(A,hasBody,C) ])
       ]),
 select([flavor=B,body=C]),
 vars([flavor=B,body=C,t=A]),
]
```

SPARQL engine

- Cada "funcionalidade" SPARQL é implementada como uma unit
- A unit `triple/3` é responsável pela instanciacão das variáveis com base na representacão da ontologia.

SPARQL engine

- Cada "funcionalidade" SPARQL é implementada como uma unit
- A unit `triple/3` é responsável pela instanciacão das variáveis com base na representacão da ontologia.

```
:- unit(triple(S, P, O)).

item :-
        /> property(P,O) :> item(S).
```

resposta XML

```
<?xml version="1.0"?>
<sparql>
  <head>
    <variable name="flavor"></variable>
    <variable name="color"></variable>
  </head>
  <results ordered="false" distinct="false">
    <result>
      <binding name="flavour">Medium</binding>
      <binding name="color">White</binding>
    </result>
  </results>
</sparql>
```

Conclusões

- Representação para uma ontologia
- "query engine" SPARQL

Conclusões

- Representação para uma ontologia
- "query engine" SPARQL

Trabalho Futuro:

- Permitir representar várias ontologias
- Melhorar o suporte para semântica do OWL
- Completar o suporte SPARQL

Perguntas?