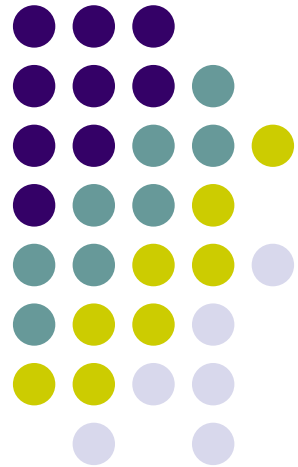
Combining XML querying with ontology reasoning: Xcerpt and DIG

Wlodek Drabent

Artur Wilk



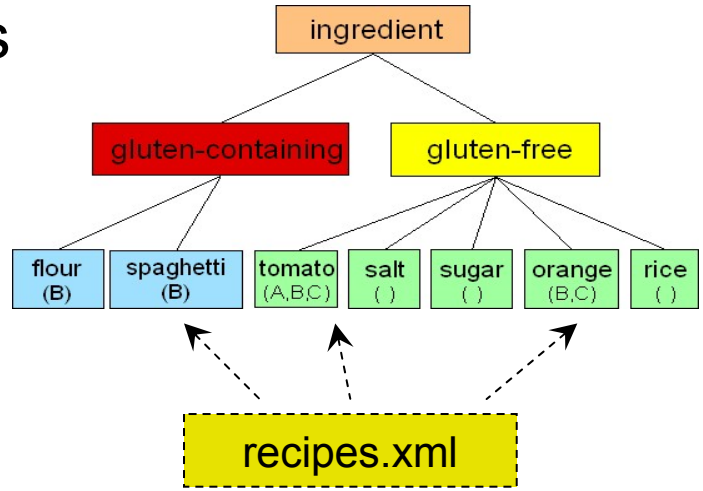
Ontology and Rule Integration Workshop

Athens, GA

November 2006

The problem

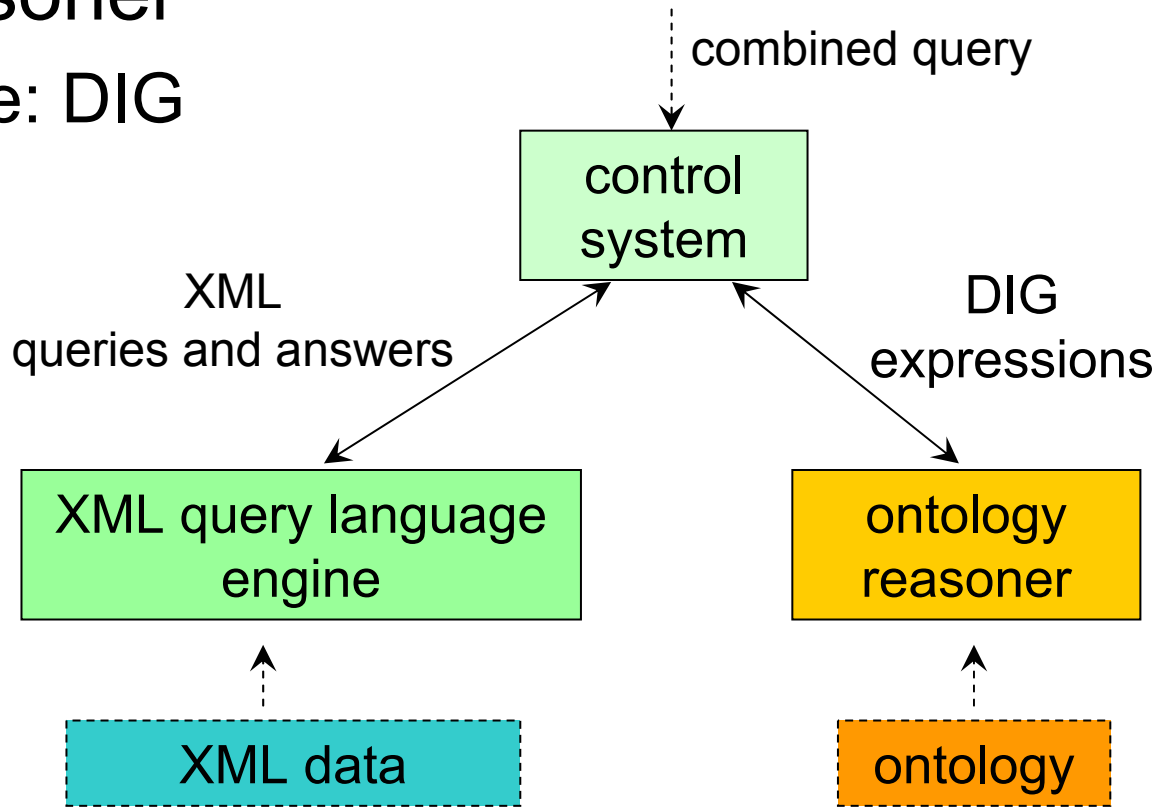
- Combining XML queries with ontology queries
- Example
  - XML document containing recipes
  - Ontology
    - classifies ingredients
    - relates ingredients with vitamins
  - Queries
    - *Find gluten free recipes* *(answer filtering)*
    - *List recipes together with vitamins they contain*
      (data enhancement)



- **Goal:** *a hybrid system answering combined queries*

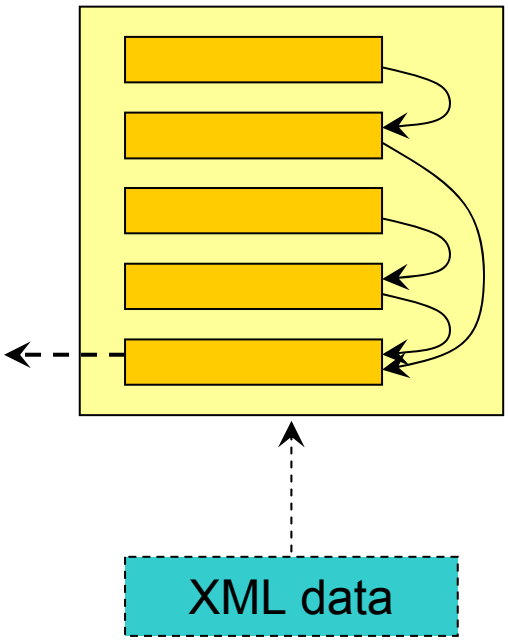
The approach

- XML query language
- Ontology reasoner
  - XML interface: DIG

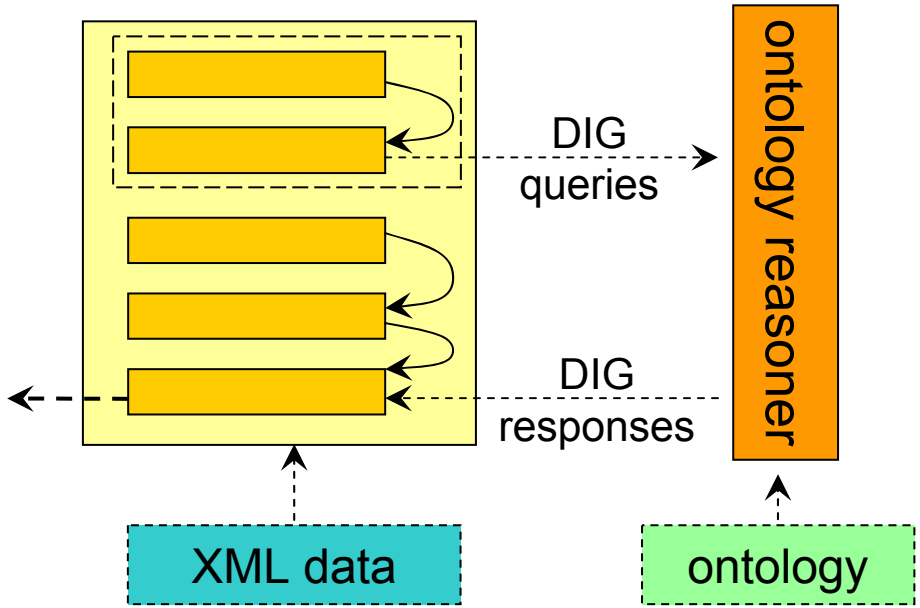


The approach (*cont.*)

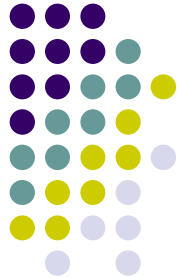
- ordinary XML query programs



- extended XML query programs
  - handled by a control system which
    - communicates with reasoner
    - calls XML query engine to execute (parts of) programs

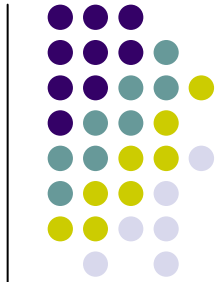


The approach: instantiation



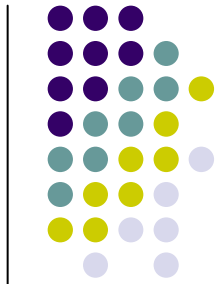
- XML query language
  - **Xcerpt**
    - simple structure of programs: rules
    - simple (fixpoint) semantics
- Ontology reasoner
  - any supporting DIG e.g. Racer
- Control system
  - Extended Xcerpt

Outline



- Preliminaries
  - *Xcerpt*
  - *DIG interface*
- *Extended Xcerpt*
  - *syntax and semantics*
  - *program examples*
- Conclusions

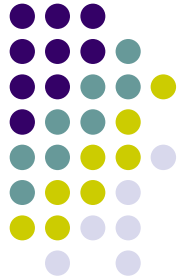
Xcerpt - Introduction



Xcerpt – query and transformation language for XML [Schaffert *et al*., 2004]

- inspired by logic programming
- uses pattern matching instead of path navigation

Xcerpt: *Core concepts*

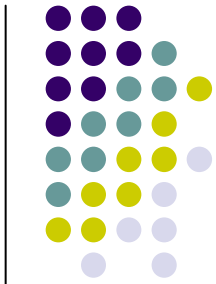


- **data terms**
  - model XML documents

    <CD> <title> *Stop* </title> </CD> CD[title["*Stop*"]]
- **query terms**
  - patterns used to match data terms
    - successful matching results in variable bindings (answer substitutions)

    CD[title [X]] *matches* CD[title["*Stop*"]] ⇨ { X / "*Stop*" }
- **construct terms**
  - used to build data terms (by applying answer substitutions)

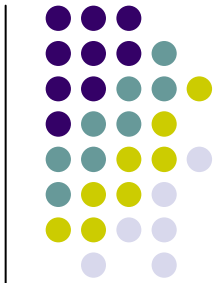
Xcerpt programs



- consist of query rules $c \leftarrow Q$
  - the body *Q*
    - used to extract XML data
    - consists of query terms
      - connected by *and*, *or* …
      - possibly associated with external resources
  - the head *c*
    - a construct term
    - used to build new XML data
  - Xcerpt syntax

```
CONSTRUCT          GOAL
  c                  c
FROM               FROM
  Q                  Q
END                END
```

Xcerpt query rules - example



Data term:

catalogue[cd[title["Empire Burlesque“], artist["Bob Dylan“], year["1985“]],
cd[title["Hide your heart“], artist["Bonnie Tyler“], year["1988“]],
cd[title["Stop“], artist["Sam Brown“], year["1988“]]]

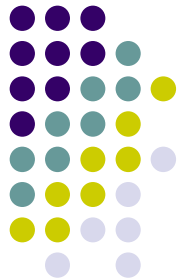
---

title [TITLE] ← *desc* cd[[title[TITLE], year["1988“]]]

Answers: { TITLE / "Hide your heart" }, { TITLE / ”Stop" }

Result: title ["Hide your heart"]
title ["Stop”]

DIG interface



- an XML interface to Description Logics systems
  - by the **D**L **I**mplementation **G**roup [S. Bechhofer]
- XML encoded messages (statements)
  - **Tell**: managing the knowledge base
  - **Ask**: querying the knowledge base
  - **Response**: replying to the queries

```
<children>
        <catom name="gluten-containing"/>
</children>
```

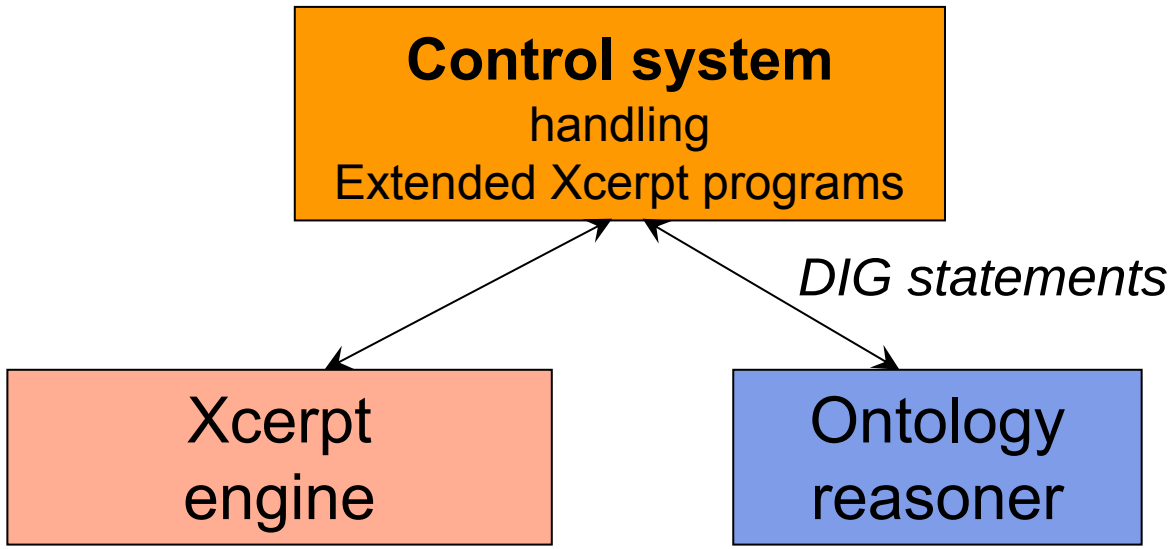
Ask

```
<conceptSet>
   <synonyms>  <catom name="flour"/>    </synonyms>
   <synonyms>  <catom name="spaghetti"/>  </synonyms>
</conceptSet>
```

Response

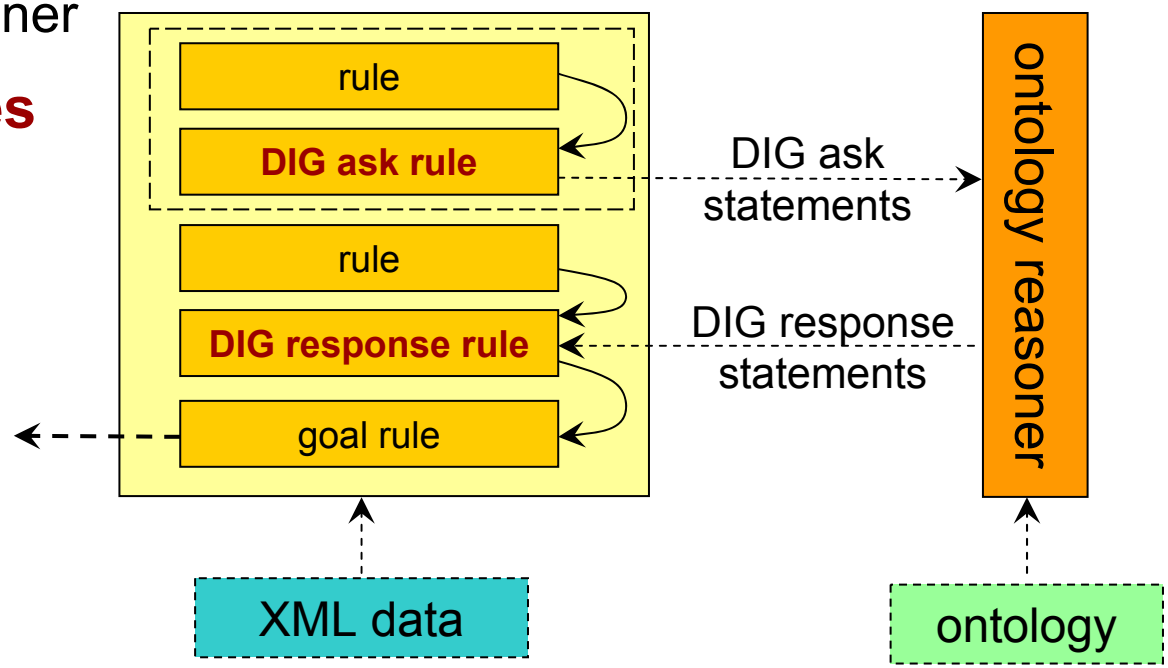
Extended Xcerpt

- Xcerpt + ontology reasoner interface
- communication with a reasoner by DIG

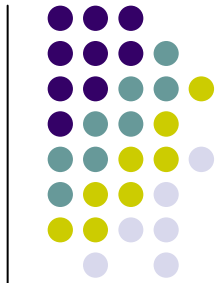


Extended Xcerpt programs

- adding to Xcerpt
  - **DIG ask rules**
    - produce intermediate results (ontology queries) to be sent to reasoner
  - **DIG response rules**
    - query reasoner responses



Extended Xcerpt: *syntax*



- the same as Xcerpt syntax
- some Xcerpt rules distinguished as DIG rules
  - **DIG ask rules** $\#l\ [a,c] \leftarrow \ldots$
    - produce data terms $\#l\ [a_i,c_i]$

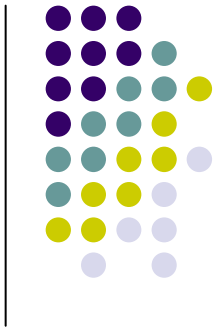
      a_i - DIG ask statement

      c_i - a context (to pass data associated with the ask statement)
  - **DIG response rules** $\ldots \leftarrow \ldots \#l\ [q_a,q_c] \ldots$

    q_a – query term matching DIG responses

    q_c – query term matching the context
- rule chaining based on ordinary rule dependence
  - restriction: no DIG rule depends on itself

Extended Xcerpt: *operational semantics*



Evaluate successively the rules of a program:

To obtain results of

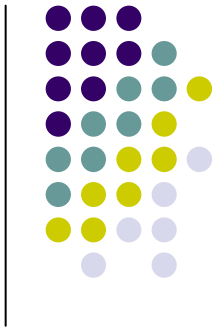
- a DIG ask rule …… $\#l\ [a,c] \leftarrow \ldots$
  - evaluate it in the standard way to obtain data terms…… $\#l\ [a_1,c_1]\ \ldots\ \#l\ [a_n,c_n]$
  - send DIG ask statements…… $a_1, \ldots, a_n$

    to the reasoner

    Ontology reasoner

    to obtain replies…… $r_1, \ldots, r_n$
  - build facts …… $\#l\ [r_1,c_1]\ \ldots\ \#l\ [r_n,c_n]$
- any other Xcerpt rule (including a DIG response rule)
  - use the standard Xcerpt evaluation method

Answer filtering – example



Query: Find recipes with ingredients containing gluten

```
GOAL
  bad-recipes[ all  var R ]
FROM
  #gluten[
     true[[ ]],
     name[  var R  ] ]
END

CONSTRUCT
  #gluten[
    subsumes[
       catom[ attr{ name["gluten-containing"] } ]
       catom[ attr{ name[ var N ] } ] ],
    name[ var R ] ]
FROM
  in[ resource[ "file:recipes.xml" ],
      desc recipe[
         name[ var R ],
         desc ingr[ name[ var N ] ] ] ]
END
```

DIG response rule

DIG ask rule

recipes.xml:

```
recipes[
   recipe[
      name[ "Recipe1" ],
      ingredients[   ingr[ name[ "sugar" ] ],
                     ingr[ name[ "orange" ] ] ] ],
   recipe[
      name[ "Recipe2" ],
      ingredients[  ingr[ name[ "flour" ] ],
                    ingr[ name[ "salt" ] ] ] ]
```

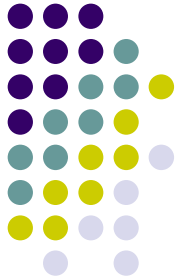
Ask statements and their contexts:

Answer substitutions (for the ask rule):

Factoid reasoner answers:

Reasoner answers and the context:

Ontology information retrieval - example



Query: Which vitamins are in each recipe?

```
GOAL
  vit-recipes[ all recipe[ var R, all var V ] ]
FROM
 #vitamins[
   conceptSet [[
      synonyms[[ catom[attr{ name [ var V ] } ] ]] ]],
   name[ var R ] ]
END
CONSTRUCT
#vitamins[ conceptSet [attr{ id[“1”] }], name[ “Recipe1”]]
END
CONSTRUCT
#vitamins[
  children[
    some[
      ratom[ attr{ name[ "contained_in" ] } ],
      catom[ attr{ name[ var N ] } ] ] ],
   name[ var R ] ]
FROM
  in[ resource[ "file:recipes.xml" ]
    desc recipe[
      name[ var R ],
      desc ingr[ name[ var N ] ] ] ]
END
CONSTRUCT
#vitamins[ conceptSet [attr{ id[“4”]}],
  name[ “Recipe2” ] ]
END
```

Ontology:

```
contained_in (“B”, “flour”)
contained_in (“B”, “orange”)
contained_in (“C”, “orange”)
```

recipes.xml:

```
recipes[
  recipe[
    name[ "Recipe1" ],
    ingredients[ ingr[ name[ "sugar" ] ],
                 ingr[ name[ "orange" ] ],
                 ingr[ name[ "salt" ] ] ] ]
  recipe[
    name[ "Recipe2" ],
    ingredients[ ingr[ name[ "flour" ] ],
                 ingr[ name[ "salt" ] ] ] ]
```

Assertions and their contexts:

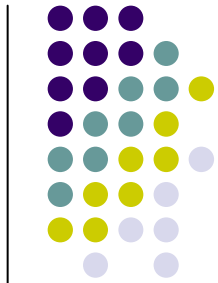
```
children[ attr{ id[“1”], some[                    name[ “Recipe1” ]
    ratom[ attr{ name["contained_in"] } ],
    catom[ attr{ name[ “sugar” ] } ] ]
children[ attr{ id[“2”], some[                    name[ “Recipe1” ]
    ratom[ attr{ name["contained_in"] } ],
    catom[ attr{ name[ “orange” ] } ] ] ]
children[ attr{ id[“3”], some[                    name[ “Recipe2” ]
    ratom[ attr{ name["contained_in"] } ],
    catom[ attr{ name[ “flour” ] } ] ] ]
children[ attr{ id[“4”], some[                    name[ “Recipe2” ]
    ratom[ attr{ name["contained_in"] } ],
    catom[ attr{ name[ “salt” ] } ] ] ]
```

result: vit-recipes [recipe [“Recipe1”, “B”], “C”],
recipe [“Recipe2”, “B”]]

Facts with reasoner answers:

```
#vitamins[ conceptSet [attr{ id[“1”]}], name[ “Recipe1” ]
#vitamins[ conceptSet [attr{ id[“2”] }], synonyms[
   catom[attr{ name [ “B” ] } ] ] , synonyms[
   catom[attr{ name [ “C” ] } ] ] ], name[ “Recipe1” ] ]
#vitamins[ conceptSet [attr{ id[“3”], synonyms[
   catom[ attr{ name [ “B” ] } ] ] ], name[ “Recipe2” ] ]
#vitamins[ conceptSet [attr{ id[“4”]}], name[ “Recipe2” ]
```

Conclusions



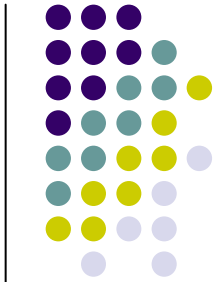
- Extension of Xcerpt allowing to query an ontology
  - communication with a reasoner with DIG interface
  - no restrictions on Xcerpt queries and DIG ask statements
  - hybrid approach
    - reusing existing systems: Xcerpt, an ontology reasoner
    - no modification of Xcerpt needed
  - a prototype implementation http://www.ida.liu.se/digxcerpt/

- Future work
  - higher level language
    - query rules $c \leftarrow O,Q$ compiled into Extended Xcerpt queries

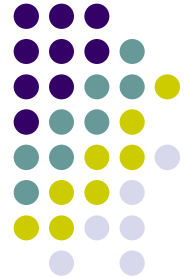
      O – ontology query

      Q – XML query

Related work



- Datalog + DL with logical semantics
  - not applicable to Xcerpt + OWL
- Hybrid framework with fixpoint semantics (Assmann *et al, 2006*)
  - ontology reasoning after rule reasoning
  - Boolean ontology queries
    - treated like constraints
- Our approach
  - ontology reasoning interleaved with rule reasoning
  - arbitrary ontology queries



Thank you!