Extending SWRL to Express Fully-Quantified Constraints

Craig McKenzie, Peter Gray, Alun Preece

http://www.csd.abdn.ac.uk/research/akt/cif

AKT

UNIVERSITY OF ABERDEEN

In this talk...

- We argue for the utility of an expressive quantified constraint language for the SW logic layer
  - based on classical range-restricted FOL
- We develop a quantified constraint representation as an extension of the Semantic Web Rule Language (SWRL)
  - compatible with OWL as well as RDFS
- We illustrate the use of the CIF/SWRL representation in the context of a practical SW reasoning application
  - based on the CS AKTive Space demonstrator

AKT

UNIVERSITY OF ABERDEEN

Context

- An approach to knowledge fusion in open, distributed environments
- Gather relevant data from multiple network sources, along with constraints on how the data can be used
- Fuse data and constraints into a dynamically-composed constraint satisfaction problem (CSP)
- Solutions (if any) are relayed back to the query originator
- Examples:
  - e-commerce: configuring custom packages
  - e-science: coalition operations for virtual organisations

AKT

UNIVERSITY OF ABERDEEN

Semantic Web context

- Fits W3C vision of a task-oriented Web "better enabling people and computers to work in cooperation"
- RDF(S) fits our minimal requirements for data to be expressed against a semantic data model
- OWL allows far richer modelling, and also supports ontology mapping
- SWRL it is not sufficiently expressive for our needs, and we therefore propose an extension that allows the representation of fully-quantified constraints

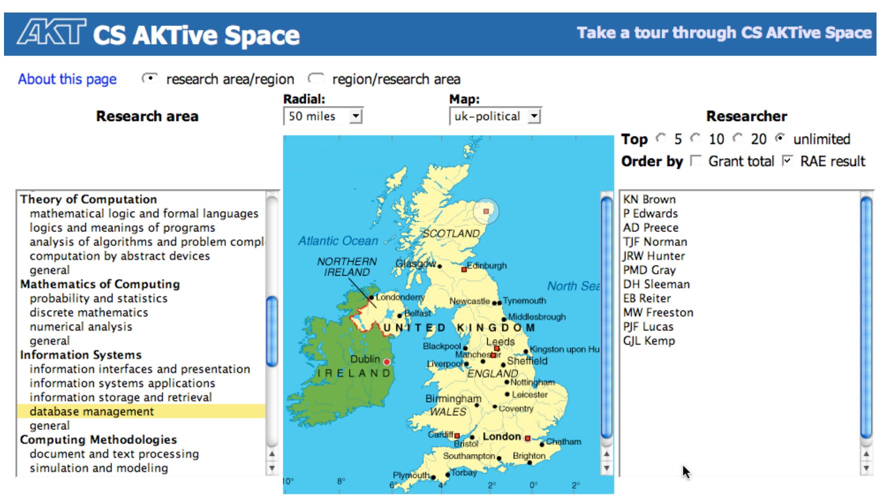
AKT

UNIVERSITY OF ABERDEEN

Application: AKTive Workgroup Builder

- Builds on CS AKTive Space demonstrator (winner of the 2003 Semantic Web Challenge)
- CAS is a large-scale repository of semantic metadata on computing science activities in the UK
- AWB uses constraints to select individuals from CAS to form working groups that satisfy particular requirements
- Could be used, for example, to form "expert panels", suggest partners for collaborative projects, or organise workshops

CS AKTive Space



Kinds of rule

- Derivation rules

```
(∀?x,?p,?s,?g) hasParent(?x,?p) ∧ hasSibling(?p,?s) ∧
  hasSex(?s,?g) ∧ (?g='male') ⇒ hasUncle(?x,?s)
```

- Rewrite rules

```
(∀?x,?y) akt:supervises(?x,?y) ⇒ foaf:knows(?x,?y)
```

- Event-condition-action rules

```
ON REGISTER-STUDENT(?s)
  WHERE supervises(?t,?s) ∧ hasGroup(?t,?g)
  DO ASSERT(hasGroup(?s,?g))
```

- Quantified constraints

```
(∀?t,?s,?g) Tutor(?t) ∧ hasStatus(?t,'research') ∧
  supervises(?t,?s) ∧ hasSubjectGrade(?s,'Computing',?g) ⇒
    (?g>60)
```

AKT UNIVERSITY OF ABERDEEN

Constraint Interchange Format (CIF)

- A CIF constraint consists of some universally quantified implications, followed by a conjunction of predicates, possibly existentially quantified

```
(∀?t) Tutor(?t) ∧ hasStatus(?t,'research') ⇒
  (∃?s,?g) supervises(?t,?s) ∧
    hasSubjectGrade(?s,'Computing',?g) ∧
    (?g>60)
```

- Why have both types of quantifier? "Sometimes readability is more important than parsimony" [*Artificial Intelligence; A Modern Approach*, Russell & Norvig, 1995]
- Choice of how to implement the constraints is left to local reasoners…

AKT UNIVERSITY OF ABERDEEN

Extending SWRL to CIF/SWRL

- Incorporates SWRL constructs where possible (striving to simplify the original CIF syntax)
- Constraints are defined as quantified implications:
  - re-use the implication structure from SWRL
  - allow for nested quantified implications within the consequent
  - innermost-nested implication will have an empty body as it is always of the form $true \Rightarrow \ldots$
- Example in the informal, human-readable syntax:

```
(∀?x∈X, ?y∈Y) p(?x,?y) ∧ Q(?x) ⇒
  (∀?z∈Z) q(?x,?z) ∧ R(?z) ⇒
    (∃?v∈V) s(?y,?v)
```

AKT

UNIVERSITY OF ABERDEEN

Abstract syntax (extended from SWRL)

constraint	::=	'Implies(' [*URIreference*] { *annotation* } *quantifiers antecedent consequent* ')'
antecedent	::=	'Antecedent(' { *atom* } ')'
consequent	::=	'Consequent(' *constraint* | { *atom* } ')'
quantifiers	::=	'Quantifiers(' { *q-atom* } ')'
q-atom	::=	*quantifier* '(' *q-var q-set* ')'
quantifier	::=	'forall' | 'exists'
q-var	::=	*i-variable*
q-set	::=	*classID*

AKT

UNIVERSITY OF ABERDEEN

Example in the abstract syntax

$$(\forall ?x \in X,\ ?y \in Y)\ p(?x,?y) \wedge Q(?x) \Rightarrow$$
$$(\forall ?z \in Z)\ q(?x,?z) \wedge R(?z) \Rightarrow$$
$$(\exists ?v \in V)\ s(?y,?v)$$

```
Implies(
    Quantifiers(forall(I-variable(x) X) forall(I-variable(y) Y))
    Antecedent(p(I-variable(x) I-variable(y)) Q(I-variable(x)))
    Consequent(
        Implies(
            Quantifiers(forall(I-variable(z) Z))
            Antecedent(
                q(I-variable(x) I-variable(z)) R(I-variable(z)))
            Consequent(
                Implies(
                    Quantifiers(exists(I-variable(v) V))
                    Antecedent()
                    Consequent(s(I-variable(y) I-variable(v)
)))))))
```

AKT

UNIVERSITY OF ABERDEEN

Sketch of CIF/SWRL RDF syntax

- New class, **cif:Constraint**; two attached properties:
  - **cif:hasQuantifiers** (range rdf:List)
  - **cif:hasImplication** (range ruleml:Imp)
- Parent class **cif:Quantifier**; two sub-classes:
  - **cif:Forall**
  - **cif:Exists**
- Two properties attached to **cif:Quantifier**:
  - **cif:var** (range rdf:Resource - URIref of SWRL variable)
  - **cif:set** (range rdf:Resource - URIref of OWL/RDFS classID)
- Note: SWRL RDF syntax allows ruleml:body to be any RDF list, so allows the nested inclusion of a **cif:Constraint**.

AKT

UNIVERSITY OF ABERDEEN

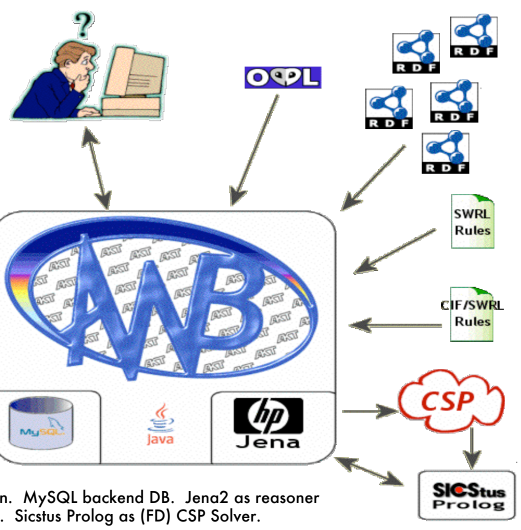
AKTive Workgroup Builder

Constructing a workgroup involves several steps:

- Defining constraints about the nature of the workgroup:
  - "generic" workgroup constraints
  - context/user-specific constraints
- Gathering (RDF) data about candidate workgroup members
- Generating entailments: ontological & from derivation rules
- Composing a CSP from the data & constraints
- Solving the CSP

AKT — UNIVERSITY OF ABERDEEN

AWB schematic



J2EE Application. MySQL backend DB. Jena2 as reasoner and RDF parser. Sicstus Prolog as (FD) CSP Solver.

AKT — UNIVERSITY OF ABERDEEN

Reasoning isn't the (only) hard part

- AWB v1 uses a cut-down version of the CS AKTive Space from ISWC 2003:
  - OWL Full **AKT Portal Ontolgy** was cut to OWL Lite (lack of reasoners)
  - 10M triples were reduced to a subset (querying limitations)
  - data was cleaned-up (errors, duplication, incompleteness)

- AWB v1 is aimed at scheduling AKT meetings:
  - data covers 5 AKT partners
  - reduced APO ignores OWL DL/Full constructs, flattens hierarchy
  - some (minor) fixes made to ontology and data

AKT UNIVERSITY OF ABERDEEN

AWB: reasoning using SWRL

- Initially, on loading candidate instances, the AWB computes OWL Lite entailments (every **Professor-in-Academia** is a **Person-in-Academia**, etc)
- Then SWRL derivation rules compute additional facts
  - Example: "if a person has an affiliation with an organisation, and that organisation has a postal address with a city then this implies that the person has a base location of the same city"

```
(∀?p,?u,?a,?c) Person(?p) ∧ Organisation(?u) ∧
  has-affiliation(?p,?u) ∧ has-postal-address(?u,?a) ∧
    address-city(?a,?c) ⇒ has-base-location(?p,?c)
```

AKT UNIVERSITY OF ABERDEEN

AWB: CIF/SWRL simple example

- Example: "every workgroup must contain at least 1 member who is a Professor'':

```
(∀?g∈Workgroup)
  (∃?p∈Professor-In-Academia)
    has-member(?g,?p)
```

- Notes
  - this kind of existentially-quantified statement can be expressed implictly in SWRL using OWL **someValuesFrom**
  - we prefer to express all quantifiers uniformly and explicitly, and leave the reasoner the option of transforming the constraints to a suitable implementation form

In RDF…

```
<cif:Constraint>
  <cif:hasQuantifiers rdf:parseType="Collection">
    <cif:Forall>
      <cif:var rdf:resource="#g"/>
      <cif:set rdf:resource="&akt;#Workgroup"/>
    </cif:Forall>
    <cif:Exists>
      <cif:var rdf:resource="#p"/>
      <cif:set rdf:resource="&akt;#Professor-In-Academia"/>
    </cif:Exists>
  </cif:hasQuantifiers>
  <cif:hasImplication>
    <swrl:Imp>
      <swrl:body rdf:parseType="Collection"/>
      <swrl:head rdf:parseType="Collection">
        <swrl:IndividualPropertyAtom>
          <swrl:classPredicate rdf:resource="&akt;#has-member"/>
          <swrl:argument1 rdf:resource="#g"/>
          <swrl:argument2 rdf:resource="#p"/>
        </swrl:IndividualPropertyAtom>
      </swrl:head>
    </swrl:Imp>
  </cif:hasImplication>
</cif:Constraint>
```

Note: the (OWL Lite) ontology URI is represented by the entity **&akt;**

AWB: CIF/SWRL 2nd example

- "Any workgroup with at least 5 members must contain people from different sites":

```
(∀?g∈Workgroup) has-size(?g,?s) ∧ (?g≥5) ⇒
  (∃?p1,?p2∈Person) has-member(?g,?p1) ∧
    has-base-location(?p1,?b1) ∧
    has-member(?g,?p2) ∧
    has-base-location(?p2,?b2) ∧ (?b1≠?b2)
```

- Notes
  - uses the (derived) property **has-base-location** from our SWRL example to indicate a person's "site"
  - interacts with previous SWRL derivation rule

Solving CIF

- Solving CIF constraints can be implemented by dynamically composing the constraints and available data instances into a CSP, code-generated for use with a particular finite domain solver
- Solvers used to date include
  - ECLiPSe (http://www.icparc.ic.ac.uk/eclipse/)
  - Sicstus Prolog FD library (http://www.sics.se/isl/sicstus/)
- 3 steps
  - form variable domains from candidate instance data
  - post constraints (translate CIF to native solver code)
  - label variables (instantiate vars such that constraints are satisfied...)

CIF's closed world assumption

- We are making a closed world assumption, at the time the finite domain CSP is composed
- This might seem contradictory to the general vision of an open world Semantic Web (and OWL DL)
- In practice, a finite number of candidate instances are always available at run-time, whether
  - gathered from a local cache (as in the current AWB)
  - acquired through some wider search (always "best-effort" on the Web)
- (We are essentially doing A-Box reasoning...)

CIF & OWL (DL)

- Our approach is not incompatible with the use of other reasoning mechanisms
- Example: OWL DL class restrictions can usefully be employed in CIF expressions to specify the domains of variables,
  - in the quantifier expressions (as the value of a **cif:set** property)
  - within the heads and bodies of the implications (unary-predicate **atoms**)
- We have yet to explore the computational complexities arising from this :-)

CIF & RDFS

- It is perfectly feasible to use CIF with only RDFS data models
- (This is true of SWRL as well, although of course SWRL has no way to handle existential quantification without OWL DL constructs)
- RDFS is relatively widely used on the current Semantic Web (Dublin Core, RSS, vCards, and FOAF are among the most widely-instantiated SW schemas)
- We feel this makes CIF immediately useful for practical SW applications

Statements about constraints

- The **URIreference** and **annotation** features from OWL allow statements to be made about constraints
- This allows provenance information to be attached
- It also allows other kinds of metadata specific to the usage of constraints (example: "strength" - hard/soft?)
- We use constraint reification in the solving process, where it becomes useful to reason about which constraints are currently satisfied
- Negotiation and argumentation can be used to soften (or in some cases harden) constraints

Conclusion

- Contributions:
  - a representation for fully-quantified constraints at the Semantic Web logic layer, as an extension to SWRL
  - a realistic test-bed application: the AKTive Workgroup Builder
- Work on multi-strategy reasoning in the AWB is ongoing:
  - Jena 2 for OWL Lite reasoning
  - trials with Jena 2, Hoolet, Jess, & Prolog for SWRL inference
  - calling SICStus Prolog FD library from Java via PrologBeans
- We aim to combine these into a practical hybrid reasoner, exploring complexity/scalability trade-offs

UNIVERSITY OF ABERDEEN