Semantic Web Reasoning using a Blackboard System

Craig McKenzie, Alun Preece, Peter Gray

University of Aberdeen

4th Workshop on Principles and Practice of Semantic Web Reasoning Budva, Montenegro, June 10-11, 2006

Overview

- Introduction
- Building Workgroups
- Blackboard Architecture
  - Traditional vs. Semantic Web approaches
  - Knowledge Sources
  - Controller
- Conclusions
- Questions and Answers

Introduction

- Logic layer of Semantic Web architecture means not only use of logic to enrich data, but also being able to ***do something*** with it.
- Reasoning is time consuming and processor intensive.
- We question the "one size fits all" approach to reasoning, and believe that a *combination* of reasoning techniques is the way forward.
- Our research interest:
  - Explore the suitability of a Blackboard System to coordinate multiple reasoning mechanisms.
- Therefore, we wish to use SW data to construct and solve a Constraint Satisfaction Problem (CSP).

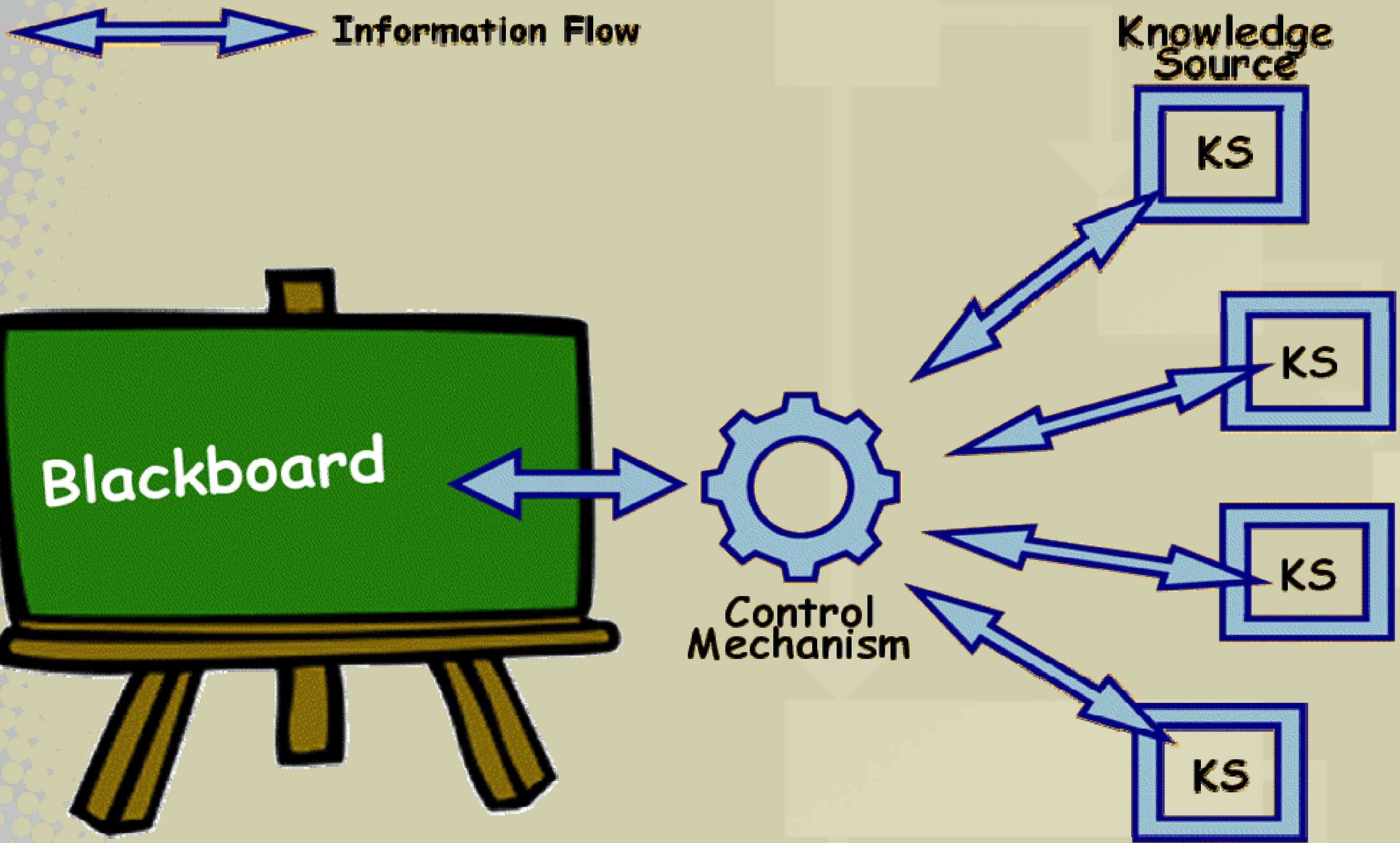
Building Workgroups

- **AKTive Workgroup Builder + Blackboard (AWB+B)** attempts to assemble one or more workgroups from a pool of known people.
  - Workgroup is a set of people, composed such that all membership restrictions (or constraints) imposed upon it have been satisfied.
    - User specifies constraints, i.e. min/max size; “ it must contain a professor”
- The problem domain is based on CS AKTive Space (also part of the AKT project)
  - Dataset describing Computing Science Staff and Researchers in UK.
  - Assumption is quality (and completeness) is not guaranteed.
- Workgroup is built by performing reasoning against the data, coordinated using a Blackboard.
  - Uses Ontology and Instance data (RDF(S), OWL); Derivation Rules (SWRL); and Constraints (CIF/SWRL).

Blackboard Systems

- Based on a metaphor whereby a group of people are all standing around a blackboard trying to solve a problem.
  - Each person has their own “expertise” and individual knowledge.
  - No individual capable of solving it on their own.
  - Solution assembled opportunistically and in incremental steps.
- Key aspects are of contributions:
  - **Coordination:** Can everyone see when a new piece of information is added to or removed from the blackboard?
  - **Control:** One piece of chalk – who gets it? Box of chalk – how stop people getting in each others way?
  - **Focus:** Is the added information relevant? Or “best-fit”?

Blackboard Components



Traditional Blackboard Systems

- In computing terms, the architecture of the Blackboard is a shared, highly *structured* Knowledge Base (KB).
  - Hierarchical structure (*Abstraction Levels).*
  - Multiple distinct hierarchies (*Panels*).
- People from the metaphor are Knowledge Sources (KS).
  - e.g. reasoners, CSP solvers, databases, Web Services, etc.
- KSs can access the Blackboard and continually check if they can make some contribution.
- Overseen by a control mechanism that monitors changes to the Blackboard and delegates actions accordingly.
  - Controller can range from being lightweight (simple transaction scheduler) to more intelligent (goal oriented).
  - Blackboard is fundamentally backward chaining.

Semantic Web Approach

- Maintains all the principles of the Traditional approach, but incorporates concepts from the Semantic Web.
  - Use of RDF means all information uses a similar syntax.
  - Communication protocols well known.
  - Abstraction Levels aligned with hierarchal structure of an Ontology (OWL Lite).
- Blackboard KB is an RDF graph allowing:
  - Easy serialisation (RDF, N3) for debugging or propagation.
  - Can be reasoned over…

The Blackboard's Reasoner...

- Blackboard generally passive, but we have added an element of intelligence to it.
  - Removes the need to make call outs to KSs that would perform the same function.
- Unfortunately, allowing the blackboard to make inferences about itself became a bottleneck…
- Simple rule based, hierarchical (class/sub-class/property only) based entailment
  - using 4 forward chaining rules.
- Custom rules perform simple class and property subsumption on both ontological definitions and instances.
  - This is based on RDFS classification but without the use of property range and domain values to improve result accuracy.

The Rules...

```
(?a rdfs:subClassOf ?b), (?b rdfs:subClassOf ?c)
                              -> (?a rdfs:subClassOf ?c)

(?x rdfs:subClassOf ?y), (?a rdf:type ?x)
                                     -> (?a rdf:type ?y)

(?a rdfs:subPropertyOf ?b), (?b rdfs:subPropertyOf ?c)
                           -> (?a rdfs:subPropertyOf ?c)

(?a ?p ?b), (?p rdfs:subPropertyOf ?q) -> (?a ?q ?b)
```

Knowledge Sources (KSs)

- KS Behaviours
- The differing types of KS:
  - Human (User Interface)
  - Instance Based
  - Schema Based
  - Rule Engine
  - CSP Solver
- Controller

KS Behaviours

- KSs represent the problem solving knowledge of the system – regarded as black boxes.
  - Can be Semantic Web Service, a RDF Datastore, DB, a CSP solver.
  - In the AWB+B we access them via Java API.
- KSs access the blackboard continually and check if they can make a contribution.
  - A pre-condition (or event trigger) indicating that they can respond to a goal already on the blackboard.
    - Response is either a solution to a goal;
    - Or division of an existing goal into sub-goals, indicating more knowledge is required.
  - An action – *what* they can add to the blackboard.
    - Facts are only ever added to the blackboard, never retracted.

Human (User Interface) KS

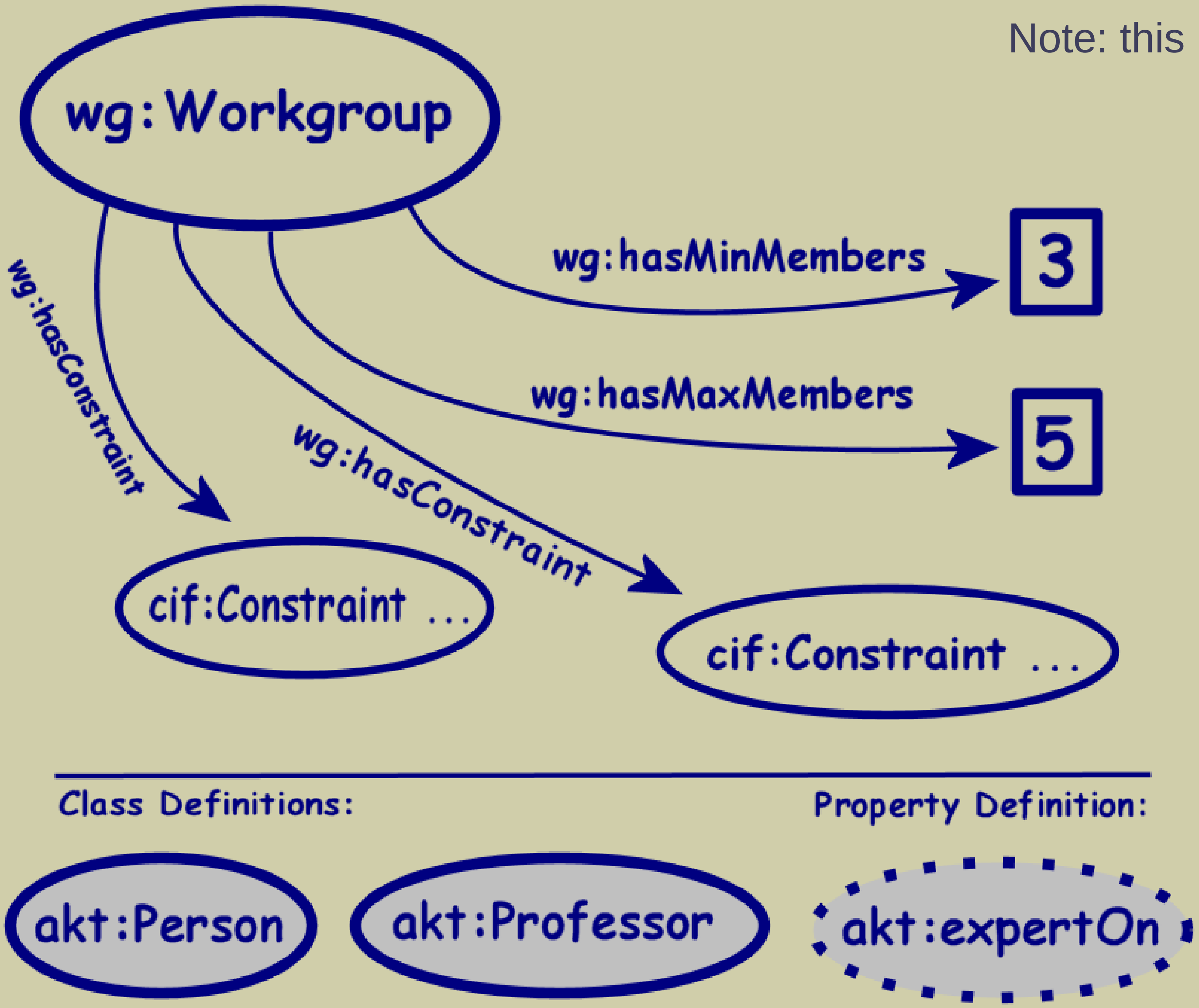
- This represents human knowledge, entered via a web interface (html form).
- Specification of problem parameters:
  - Number of workgroups to be built
  - Size of each workgroup
  - Various compositional constraints (written in CIF/SWRL and available via a URI)
- Specification of dataset URIs:
  - Ontology, RDF Data and SWRL Derivation rules
- KS transforms these into system starting goals and posts them onto the blackboard.

Example: system starting goals…

- Workgroup Properties:
  - The constraints on the group are:
    - Must contain between **3** and **5** members, of type **Person**.
    - Must contain at least 1 **Professor**.
    - Must contain an **expertOn** “*Semantic Web*”.
  - Make use of the following Derivation Rule:
    - **Person**(?p) & **authorOf**(?p, ?b) & **Book**(?b) & **hasSubject**(?b, ?s) $\Rightarrow$ **expertOn**(?p, ?s).

Blackboard Contents (Initial Goals)

Note: this is a Simplified Graph



These *Goals* are derived from **“membership of type Person**” and the 2 “must contain” constraints.

Instance Based KS

- Contains only instance data, not actual schema itself, i.e. a single RDF data file or a larger *triple store*.
  - We cannot assume that all entailments have been generated for RDF.
- KS contributes in the following ways:
  - Offers to add a **solution** to a posted sub-goal by adding instance data for classes and/or properties defined on the blackboard.
  - Offers to add a **solution** to classify any property's subject and/or object which the blackboard does not have a class definition for.

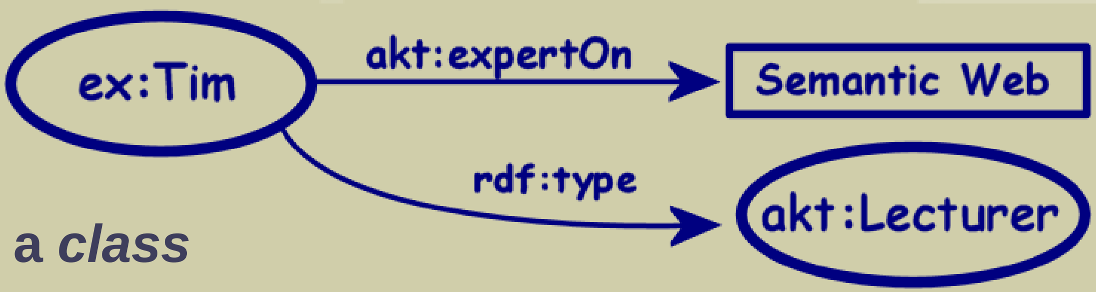
Blackboard Contents (Instance KS)

• We have the 3 potential goals (1 property and 2 classes) defined on the blackboard:

• This KS will offer a “solution” triple statement containing the property **expertOn**, i.e.

…but this gives no information about the subject **<ex:Tim>**.

• Therefore, it will also offer a classification of this:



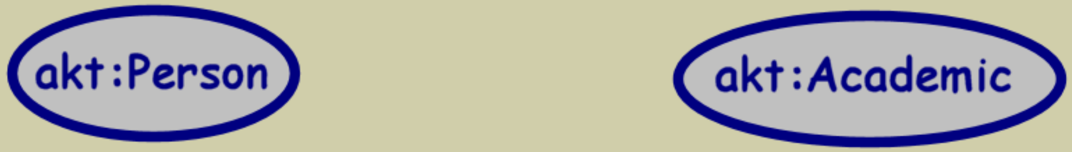
Note: this KS does not offer a *class definition* for <ont:Lecturer>

Schema Based KS

- This represents a KS that only contains ontological schema information.
  - Facilitates construction of relevant ontological parts on the blackboard.
- KS contributes in the following ways:
  - Offers to add new **sub-goals** by looking for ontological sub-classes/properties of those already defined on the blackboard.
  - Offers to add new **sub-goals** by adding `<rdfs:subClassOf>` or `<rdfs:subPropertyOf>` statements connecting those already defined on the blackboard.
  - Offers to add new **sub-goals** for any subject/object on the blackboard that does not have a class definition.

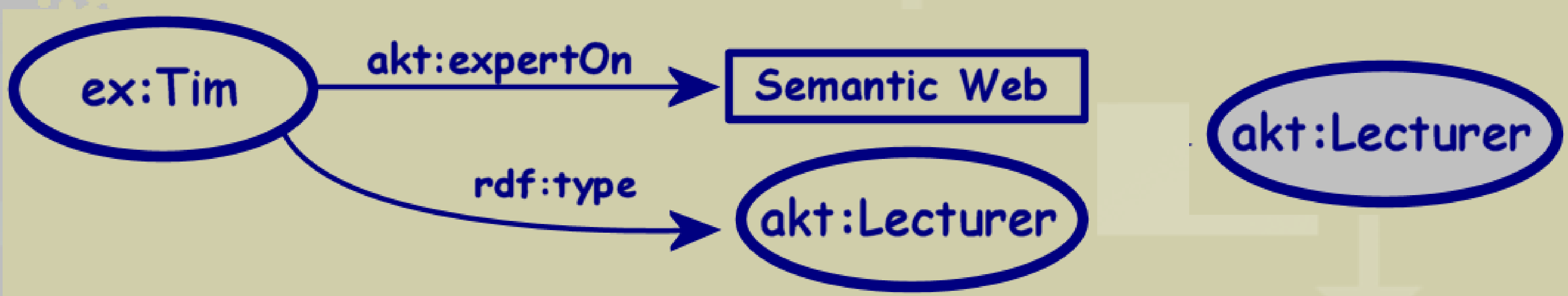
Blackboard Contents (Schema KS)

The KS would see **<akt:Person>** defined on the blackboard, and then offer to add a sub-goal by defining a sub-class Academic:



Subsequently, it would offer the sub-class *link* between these 2 classes:

Finally, from the previous contributions by the Instance KS, it would see the **<rdf:type> <akt:Lecturer>** belonging to **<ex:*Tim*>** and since it knows about this class, explicitly add the class definition to the board:



Rule Based KS

- Examines the contents of the blackboard and determines if any of the rules that it knows about are *required*.
  - A rule is required only if any of the consequents are present on the blackboard.
- KS contributes in the following ways:
  - Offers to add a **solution** by firing the rule against instances already on the blackboard and asserting the appropriate statements.
  - Offers to add new **sub-goals** by offering class/property definitions of rule antecedents not on the blackboard.
- Currently, a rule KS only contains one rule at a time.
  - This is rewritten into a SPARQL query and run against the blackboard.
  - Uses a brute force, forward chaining approach…

Blackboard Contents (Rule KS)

- Remembering our derivation rule:

```
Person(?p) & authorOf(?p, ?b) & Book(?b) &
        hasSubject(?b, ?s) ⇒  expertOn(?p, ?s).
```

- Blackboard contains **Person** class defn but not property defs for **authorOf** & **hasSubject** – these have not been defined
  - regardless of instance data, the rule is incapable of firing.

- This KS adds the sub-goals: authorOf and hasSubject.

- (Hopefully) Once other KSs have contributed instance data for the antecedents, the rule can fire and generate a **solution** instance for the expertOn property that has not been explicitly stated in a KS.

The Controller (1)

- Role of the controller is to oversee the running of the system.
  - Does not allow addition of `<owl:Thing>` and prevents the KSs modifying the blackboard directly.
- The AWB+B blackboard actually contains 2 panels:
  - Data Panel & TaskList Panel (both RDF Graphs).
- TaskList is used by the controller to store *what* information a KS can contribute based on the blackboard (Data Panel) contents.
  - Unlike the Data Panel, KSs are allowed to add **`TaskListItems`** to the TaskList panel directly.
- Once a TaskListItem has been actioned by the controller, it is removed from the TaskList –
  - this is the *only* time anything is ever deleted from the blackboard.

The Controller (2)

- All KS registered the system cycles over each one asking it to populate the TaskList panel.
  - Calls `canContribute()` method.
- Decision is made on which tasks to action
  - Calls `makeContribution()` method.
- Simple implementation of the controller
  - Action all items on the TaskList.
  - Possible to introduce a more goal oriented decision process.
- Process stops when nothing new is added after a complete cycle or if a solution to the workgroup appears on the blackboard (i.e. `wg:hasMember` properties are added to the `wg:Workgroup` instance).

Conclusions…

- Main issue is the blackboard architecture is inefficient:
  - 2 step `canContribute` and `makeContribution` process – inefficient
    - Effort involved to determine if a contribution can be made is comparable to actually making the contribution.
  - Contradictions on the Blackboard.
- However, the paradigm allows for:
  - Coordination of a mix of reasoning methods on data.
  - (Hopefully!) Only small, *relevant* subset of all the available data is ever placed on the blackboard
  - Can add/remove KSs with the only impact on the final results.
- AWB+B is still in development, so still have scope to explore:
  - Differing KS combinations; alternate Controller strategies; rule chaining; concurrency; code optimisation; etc.

the end..!

Thanks for your attention…

…any Questions?