The Open Knowledge Project

Complex Knowledge Sharing in Open, Peer-to-Peer Environments

Dave Robertson
dr@inf.ed.ac.uk
http://www.openk.org

Hard Problem

An open knowledge sharing system is one which places no specific boundary on the number of people who may share knowledge and which can readily be joined by people wishing to share or discover knowledge for complex tasks. No perfectly open method of sharing knowledge can exist because of well known theoretical obstacles to maintaining common knowledge in asynchronous environments and practical problems in gaining consensus on semantics for distributed information. In all open knowledge sharing, therefore, some compromise is reached to obtain a particular form or level of openness. Traditional forms of compromise are limiting:

The Worldwide Web: is easy to join but allows only limited forms of knowledge sharing for simple tasks. Its reliance on database lookup places an upper limit on the number of people who can have their knowledge shared.

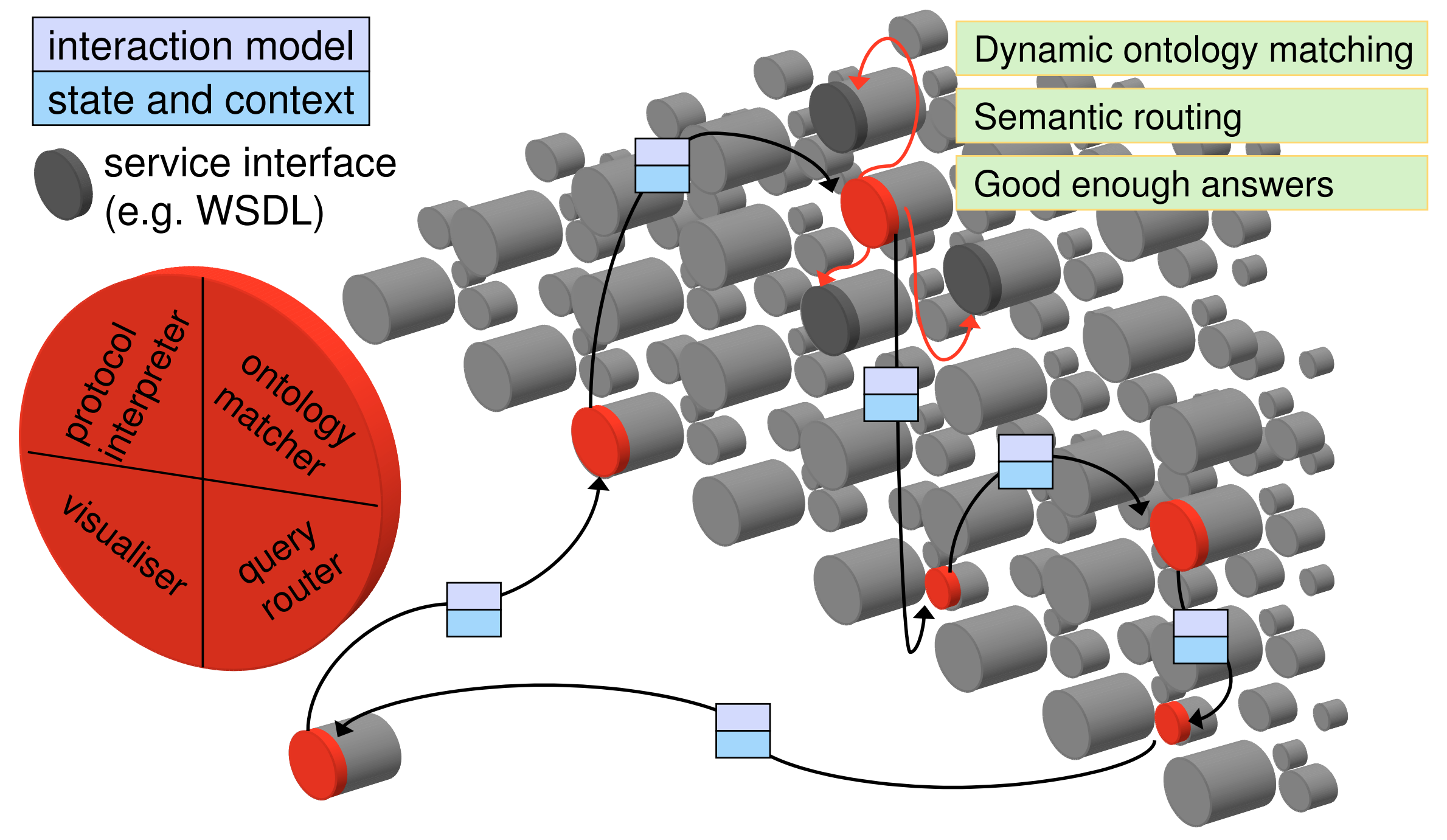
Semantic webs: are built through local markup of individual knowledge repositories. To be generally useful, local markup requires broad consensus on what terms mean, but to be confident of consensus in this absolute sense requires knowing who to consult about the ontology used.

Web services: can be tuned to complex tasks but are specific to the programs (and people) connected with those tasks. Service engineers can't connect the world's Web services together in all the combinations everyone might want but neither are individuals readily able themselves to express service composition or share knowledge about successful compositions.

Grid infrastructures: can improve performance in service delivery but suffer the same scaling problems for knowledge sharing as Web services.

Simple Solution

OpenKnowledge takes a different starting point. We begin with a strictly peer to peer architecture, in which there is no centralised database, overall consensus on ontologies, or pre-connected Web services. Rather than relying on these (non-scaleable) facilitators for knowledge sharing, we rely on models of interaction that are initiated locally by peers and shared through the peer network, simplifying discovery, ontology alignment and service composition because of the narrower interaction context provided by these models. Crucially, the interaction models themselves are the currency of knowledge transfer between peers. It turns out that by using this richer style of communication we can obtain forms of query routing; ontology matching and composition that are dynamic (so adaptive to flexible peer groups) and support complex interactions using comparatively simple computational mechanisms.



Established Theory Re-Applied

By changing the basic rules of the game for knowledge sharing, from a representation-centric to an interaction-centric view, we are able to recruit a large number of traditional forms of automated reasoning in a practical way to this global task, for example:

Process calculus is the origin of the temporal language (the Lightweight Coordination Calculus, LCC) used in the OpenKnowledge kernel. This allows us to describe interactions in a compact way and verify them pre-deployment.

Logic programming provides a design style for LCC, enabling us to teach the language rapidly to engineers. Crucially, these engineers view LCC as a programming language, for interactions.

Context logic allows us to provide "safe ontological envelopes" around deployed interactions, making commitments to ontological mappings incrementally at run time.

Meta-interpretation allows us to interpret other process control languages in LCC, so those using more traditional process languages can be brought into the LCC fold.

Probabilistic logics allow us to estimate the likelihood of success of an interaction as it unfolds, and to make choices (based on previous successes/failures) of which peers to recruit into new interactions.

Constraint relaxation: allows LCC to be deployed in applications (such as configuration or negotiation) where the interaction revolves around a space of choices to which peers commit in a flexible way.

Leads to Novel Applications

LCC is a high level language so it can be effective as a means of specifying choreography between services - the bonus then being that, since it is also an executable language, it can be used actually to choreograph these services. We have shown how this is can be done for service coordination in astronomy experiments, where the interaction becomes part of the experiment definition.

In proteomics key issues are fast data comparison, filtering and result sharing. By making it easy to devise and share procedures for performing these tasks we have been able to publish new results on comparative analyses of protein structure. This is a first step in loosening this community's reliance on centralised database curation, without losing quality or provenance.

In chaotic environments a little coordination often can go a long way. An industrial source of controlled, chaotic environments is the games industry. We have linked LCC to agents in the Unreal Tournament games environment and use this to describe and evaluate coordination strategies for complex multi-agent systems.