Using fibrations for situation identification

Simon Dobson and Juan Ye

Systems Research Group
School of Computer Science and Informatics
UCD Dublin Belfield, Dublin 4, Ireland

http://www.ucd.ie/csi

{simon.dobson,juan.ye}@ucd.ie

Overview

The chief challenge of pervasive computing is the match behaviour against user expectation

- Cross-technology, rich links between information, hooks into a range of information sources
- How do we capture and describe what the system is expected to do? How to we make these behaviours compose?

There is a strong relationship between the way behaviour adapts and the structure of the world in which that behaviour occurs

Dobson and Nixon. More principled design of pervasive computing systems. LNCS 3425. 2004

Our goal: to suggest that fibrations over graphs (or categories) provide a good analytic model

The consistency problem(s)

Making pervasive systems is complicated by the need to maintain models of the world

- Incompleteness – information will be absent
- Inaccuracy – timeliness, imprecision, lies, omissions

All inputs are *evidence* of fact, *not* facts themselves

- The vagaries of sensing, the lack of user discipline, …

Autonomous adaptation

- Match behaviour with what’s happening around the users

Open, dynamic, composable systems do not lend themselves to highly detailed semantic treatments

- Need to perform analyses and checks automatically
- Describe systems at a high level of abstraction, comparable to a business workflow

Context and situation

It is useful to draw a distinction between *context* and *situation*

Coutaz and Rey. Foundations for a theory of contextors. CADUI. 2002

- Context: who, where, what, …
- Situation: the higher-level semantics of what this means
- So we *collect* context but *react to* situation

Match behaviour against situations

- "In the following situation, provide the following services"

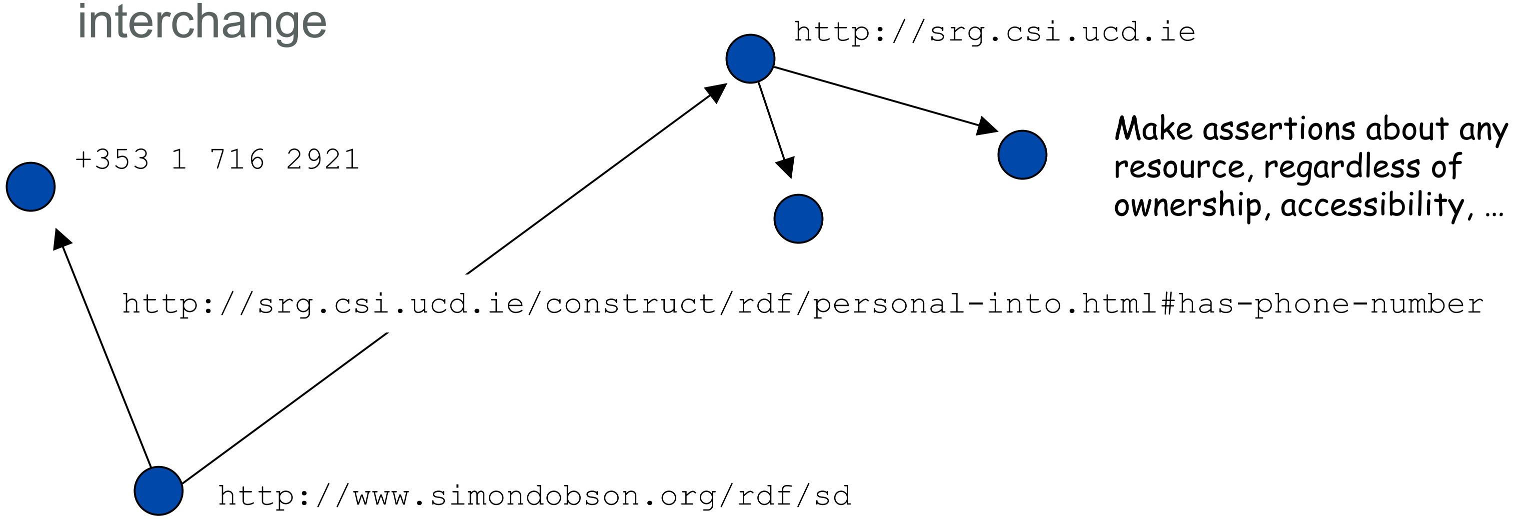
If there is a workflow associated with behavioural change, there should be a corresponding flow associated with situations

- Enrich the situation with what the user is doing
- Fixed point context…

Modeling context with RDF

W3C’s Resource Description Framework

- Traditional, well-understood, classical AI formulation of knowledge as (subject, predicate, object) triples
- ..where the first two must be, and the last may be, a URI
- Globally-unique statements about globally-unique resources
- Emerging as the standard for context representation and interchange

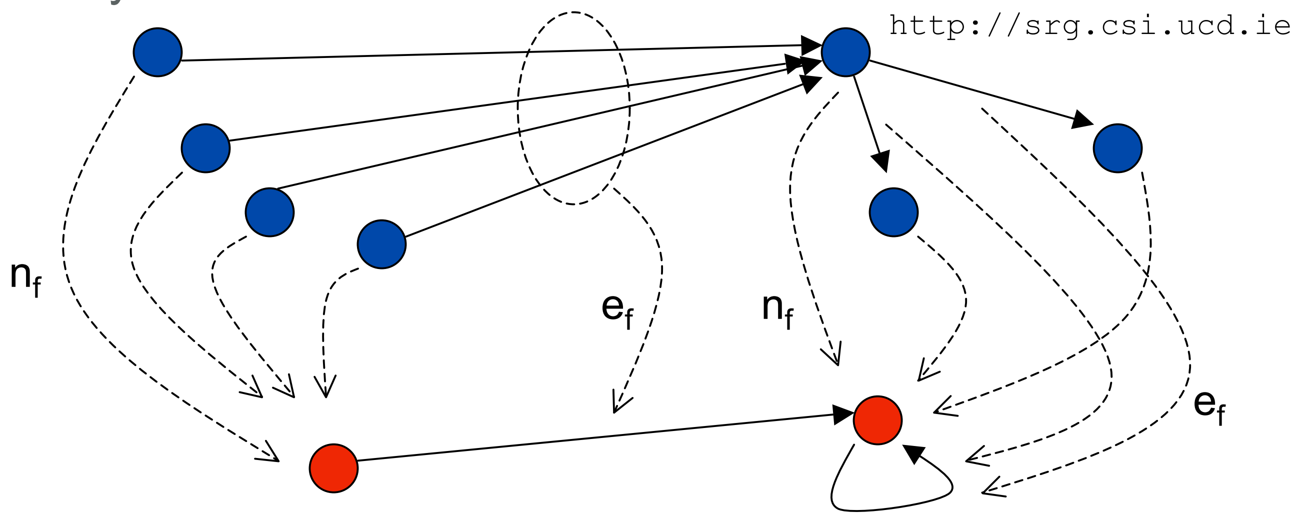


RDF graph homomorphisms

RDF models form a graph, and we can therefore consider graph-based semantic models

Graph homomorphism $f = (n_f, e_f) : A \rightarrow B$

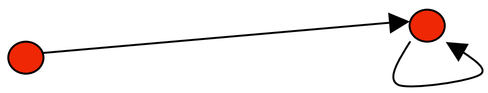
- $s_B(e_f(e)) = n_f(s_A(e))$ and similarly for edge targets
- Map nodes to nodes and edges to edges so as to preserve the adjacency structure



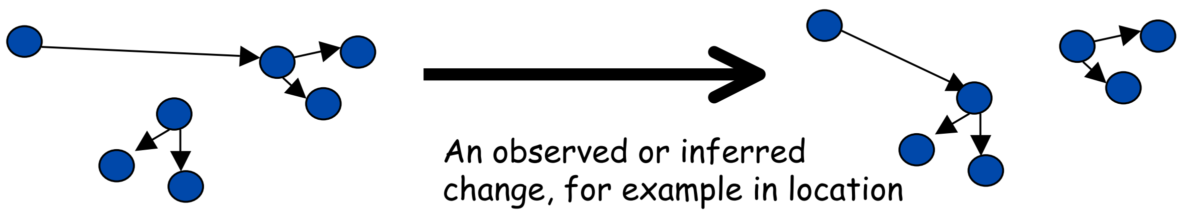
Representing situations

We can represent situations and the transitions between them as graphs too

- The workflow through the context



We move between situations by changing context, so we need to model the way context changes



We can then construct a graph homomorphism from contexts to situations

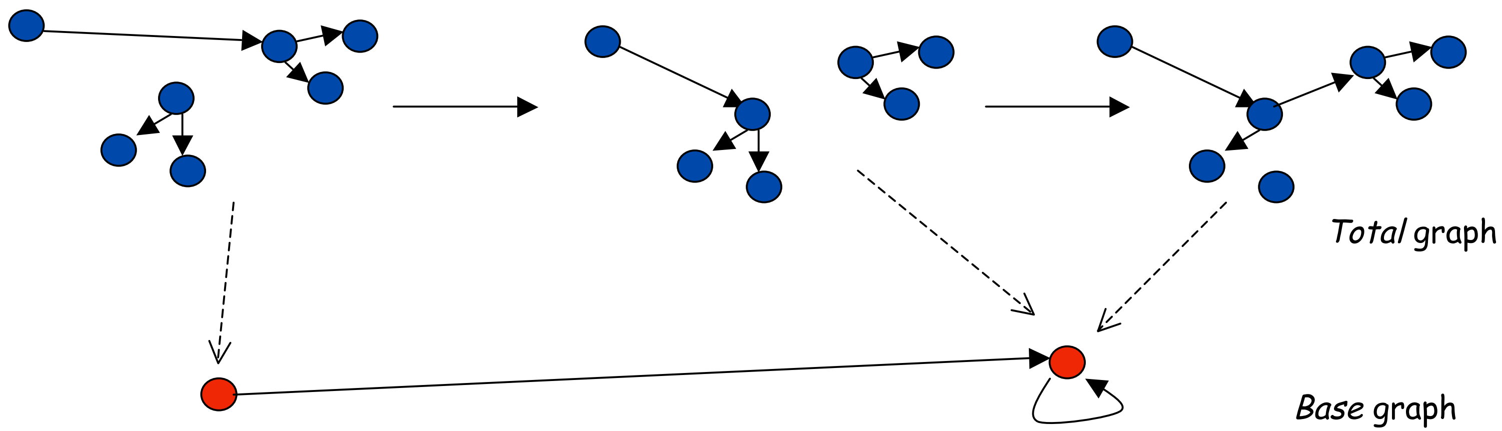
- A change of context maps to a change of situation
- The *context evolution* graph

Fibres

Given a graph A and a homomorphism f : A → B, the *fibre above b* is a sub-graph A' of A such that

- $n_f(a) = b$ for nodes a and b
- $s_B(e_f(e))$ and $t_B(e_f(e))$ for edge e are both in A'

So the fibre contains the nodes that map to b under f, together with the edges between those nodes



Fibres and situations

If we consider a context evolution graph fibred over a situation graph

- Nodes (context graphs) = those contexts that represent the system in a given situation
- Edges within fibres = those transitions (observations, inference) that do not change the situation

We may use the situation to select a behaviour, and the exact context to parameterise that behaviour

- So the contexts are equivalent but not equal: it is still worth observing the differences
- …but only a transition between fibres will lead to a "significant" change in behaviour, by changing the situation

Fibrations

A homomorphism $f : A \rightarrow B$ is a *fibration* if, for each edge e in B such that $n_f(a) = t_B(e)$, there is a unique edge e^a (the *lifting of* e *at* a) such that $e_f(e^a) = a$ and $t_A(e^a) = a$

Boldi and Vigna. Fibrations of graphs. Discrete mathematics **243**. 2002

This is a powerful connection between two graphs

- Put another way, there is at most one edge from one fibre to a particular node in another
- In the case of a context evolution graph, the mapping reduces the transitions that are observable in behaviour
- Preserves some aspects of the path through the context in the changes of behaviour

How does this help?

We believe this structure has four separate but interrelated significances:

1. Constructing a predictable, “scrutable” system involves matching situation (behaviour) transitions with contexts and their transitions
2. Fibrations compose cleanly, so we can specify different aspects of a system separately and then compose them together
3. Cases which *can’t* be modeled as fibrations may point to design issues needing to be resolved
4. Certain aspects of sensor and inference uncertainty can be handled cleanly within a fibration

We will look at these aspects within an example of a simple location-based service

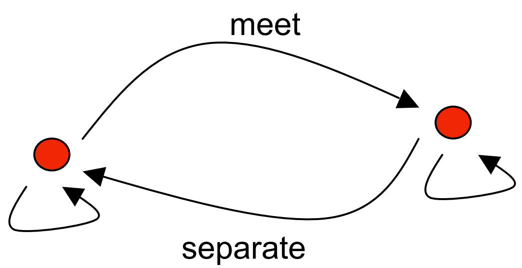
Example: location-based services

The system

- Two users a and b, being mapped to locations l_1, l_2, …, l_n by a predicate p
- Gives rise to a context evolution graph with nodes the possible combinations of locations and edges the observations between them

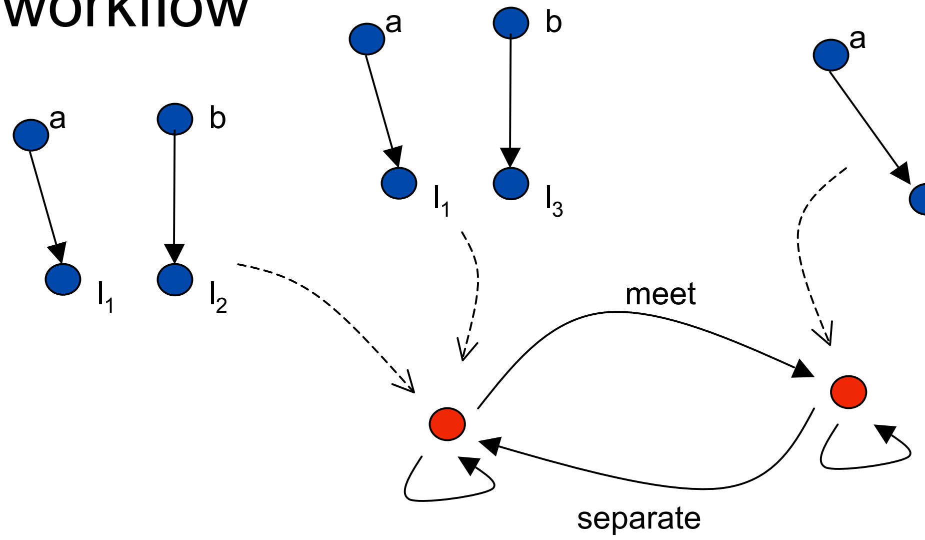
We might want to exhibit different behaviour when the users are together than when they are apart

- Situation graph with two nodes, separate and meet edges, and identity loops



Matching fibre to function

We can fibre the context evolution graph over this workflow



The "seams" in the context are identified and matched to changes in behaviour, so a user can identify a cause for a change in behaviour and predict future changes

Different contexts select the same behaviour, parameterised differently

- Match the environments to the behaviours, similarly with changes

Fibrations for design

Different contexts and situations

- Context: observe what a person is doing with their computer
- Situation: match-up actions with scenarios, such as in a meeting, travelling, …

We can construct a cross-product of the two pairs of graphs

- Actions done and the location they're done in, scenarios acted-out when apart and together

Generate a richer description from simpler ones in a well-founded way

- Lots of subtleties – overriding, overlaps – that are simple to handle algebraically but hard to match to the desired system semantics

Handling uncertainty

In realistic context-aware systems, sensed and inferred context is uncertain

- Errors, imprecisions, inaccuracies, lies, attacks, …

These manifest to some degree in the fibration form

- If an error changes a context c_1 to a context c_2, then if $n_f(c_1) = n_f(c_2)$ the error will not cause behavioural impact (although it may change the paramererisation of that behaviour)
- If the contexts are in different fibres, we will observe a difference

Smooth observations using uncertain reasoning and machine learning, correct the context *not* the interpretation as situations

Dobson, Coyle and Nixon. Hybridising events and knowledge as a basis for building autonomic systems. Journal of Trusted and Autonomic Computing. To appear.

Or capture uncertainty explicitly

Conclusions

Well-defined, scrutable response to contextual change

A mid-levels semantics for pervasive computing

- Focus on how behaviour changes, rather than on the details of what the system actually *does*

Fibrations are more commonly regarded as a category-theoretic idea, and this is the direction of our current work

- Capture the properties of various categories of context
- Construct composite contexts and behaviours using fibrations
- Use the results to guide the construction of context-aware applications wthin the Construct framework

Coyle *et alia*. Sensor fusion-based middleware for assisted living. Proc. ICOST. 2006