Research Interests

Mathieu Roger

Laboratory TIMC-IMAG- Osiris Team
Mathieu.Roger@imag.fr

My main interests are the domains of data models for Database, Programming Languages and Knowledge Bases. I am working on the OSIRIS system [Simonet et al., 94] which is a mixture of database and knowledge base. In this system I am implementing the following features:

- Inheritance as set inclusion (contrary to sub-typing)
- A view mechanism, analogous to defined concepts in DL
- Automatic instance classification
- A multi-methods mechanism (that relies on instance classification)
- A constraint language on functions and methods (that relies on instance classification)
- Constraints on transaction (i.e., modifying the state of a universe) and completion (adding coherent information about the state of the universe)

My work has followed two main directions. The first one is the comparison of the OSIRIS system with DLs [Roger et al., 00 ; Roger et al., 01]. The second one is the study of inheritance mechanisms in OO models, and especially programming languages. I have recently discovered that our data model, initiated by [Simonet 84], can be partially described through the concepts from [Guarino and Welty 01] : we partition the universe into rigid classes that supplies a global identity (these classes are called p-type in [Simonet 84]), which is not an imposed constraint onto DL schemas. As shown in [Guarino and Welty 00] this is not a strong hypothesis because every schema can be expressed that way.

I am implementing a prototype version of OSIRIS which is based on translating both schema data and instance data into the DL reasoner RACER [Haarslev and Moller 01] on which I rely for all deduction features. In the near future, I will also translate data into the object database AceDB (www.acedb.org) for persistency. Translation into DL allows our system to be more expressive, and releases me from the burden of writing a logical demonstrator. With class (or concept) and object (or instance) classification, it will be possible to express constraints on transactions [Roger et al., 02]. Classical type and sub-typing theory rely on a closed world assumption (the type is the only callable things onto its objects). I release this assumption by expressing constraints on methods, which solves problems that occur with binary methods and with mixing modification and sub-typing. In the following I rapidly describe the syntax for the Osiris data modelisation language and an example.

Abstract classes are classes that do not carry any identity. They describe only a small part of their objects' structure. The "canonical" example of an abstract class is the class THING that contains all objects: one cannot build a “thing” because one cannot decide if one already knows it or not!

```
abstract-class-definition ::
  abstract class name
    [(all|some) abstract-class-formula]
    [attribute-declarations]
    [constraints]
    [method-constraints]
   end
```

Concrete classes are the most important kind of classes; they are the equivalent in this model to the “standard” notion of class in OO models. An object is created in a unique concrete class and remains in it during its lifetime. More precisely:

1. they are **primitive** classes, in the sense of Description Logics
2. they **supply an identity**, so there exists a constructor for such a class
3. they are **rigid**, i.e., an object belonging to a concrete class will remain in it forever
4. they are pairwise disjoint.

Because of these properties, an object belongs to a unique concrete class and never changes.

```
class-definition ::
   concrete class name
     [some abstract-class-formula]
     [attribute-declarations]
     [constraints]
     [method-constraints]
     identity identity ;
   end

identity ::
  always new |
  key attribute-name (, attribute-name)*
```

A **virtual class** is a subset of a unique concrete class. It is defined by a logical constraint, including set formula over other virtual classes.

```
virtual-class-definition ::
   virtual class name
     [(all|some) classes-formula]
     [attribute-declarations]
     [method-constraints]
     [constraints]
   end
```

Example.

```
abstract class Aged
  age : int ;
  modify_age(a:int) -> Aged ;
  modify_age(a:[0..10]) -> Young ;
end

abstract class Young all Aged
  age in [0..10] ;
end

concrete class Person some Aged
  age in [0..150] ;
  name : string ;
end

virtual class Minor all Person
  age in [0..18[ ;
  change_age(a:[18..150]) -> not Minor ;
end

concrete class Car some Aged
  motor : string in {"diesel", "gasoline"} ;
  change_motor(m:string) -> Car ;
end

virtual class Diesel all Car
  motor in {"diesel"} ;
end
```

[Guarino and Welty 01] Nicolas Guarino and Christopher Welty, Identity and subsumption, in R. Green, C. A. Bean, and S. Hyon Myaeng (eds.), *The Semantics of Relationships: An Interdisciplinary Perspective*, Kluwer 2001

[Haarslev and Moller 01] Volker Haarslev and Ralf Moller, RACER System Description, IJCAR'01

[Roger et al., 00] Mathieu Roger, Ana Simonet et Michel Simonet, A Description Logics-like Model for a Knowledge and Data Management System, in DEXA 2000

[Roger et al., 01] Mathieu Roger, Ana Simonet et Michel Simonet, Object Space partitioning in a DL-like database and knowledge base management system, DEXA 2001

[Roger et al., 02] Mathieu Roger, Ana Simonet et Michel Simonet, Toward updates in Description Logics, KRDB 02

[Simonet 84] Ana Simonet, Types Abstraits et Bases de Données: formalisation du concept de partage et analyse statique de contraintes d'intégrité, PhD, University of Grenoble, 1984

[Simonet et al., 94] Ana Simonet and Michel Simonet, Objects with Views and Constraints: From Databases to Knowledge bases, OOIS'94