The Modelling We Do

Enrico Motta

Modelling Primitives

Classes

Slots

Instances

Relations

Functions

Rules

Axioms

Reasoning Services

Query Answering (Ask)

Constraint Checking (Tell)

Class Modelling

- *Intensional vs Extensional Definitions*
- *Classes as Objects of Discourse*

```
(def-class person)

(def-instance enrico person)

(def-class guideline-user-type () ?x
    :iff-def (or (subclass-of
                       ?x generic-care-giver )
                     (= ?x patient)))
```

Constraints on Class Definitions

```
(def-class project (activity) ?x
   ((has-leading-organization :type organization)
    (involves-organization :type organization :min-cardinality 1)
   (has-project-leader :type person)
   (has-project-member :type person :min-cardinality 1)
   (funding-source :type organization)
   (has-web-address :type URL)
   (addresses-generic-area-of-interest :type generic-area-of-interest))

   :constraint (and (forall ?y
                           (=> (has-leading-organization ?x ?y)
                               (involves-organization ?x ?y)))
                        (forall ?y
                           (=> (has-project-leader ?x ?y)
                               (has-project-member ?x ?y)))))
```

Relations (in addition to slots)

```
(def-relation PROJECT-INVOLVES-ORGANIZATION-UNIT (?p ?u)
  "It is sufficient that somebody in unit ?u works in project ?p"
  :constraint (and (project ?p)(organization-unit ?u))
  :sufficient (and (project ?p)(organization-unit ?u)
                   (has-project-member ?p ?x)
                   (works-in-unit ?x ?u)))
```

Functions

```
(def-function filter (?l ?rel) -> ?sub-l
   "Returns all the elements in ?l which
      satisfy ?rel"
 :body (if (null ?l)
            ?l
            (if (holds ?rel (first ?l))
                (cons (first ?l)
                      (filter (rest ?l) ?rel))
                (filter (rest ?l) ?rel))))
```

Holds

```
Holds (?rel ?arg1…..?argn)
   iff
(?rel ?arg1…..?argn)
```

Functions

```
(def-function EXTENSION (?r) -> ?set
  "The extension of a relation is the set of all tuples for which the
    relation
   holds.  This is a kind of operational definition, which retrieves
    the set of all
   tuples for which the relation is predicated in the current KB.
    This function
   is restricted to defined relations only"
  :constraint (defined-relation ?r)
  :body (if (= (the-schema ?r) ?list)
          (eval-setofall ?list (cons ?r ?list))))
```

Rules are also useful

- *Used for inferences (no constraint checking)*
- *Separate from ontological definitions*
- *Allow modular extensions of definitions*

```
(def-rule rule-for-collaborating#1
  ((collaborates-or-collaborated-with ?p1 ?p2)
   if
   (or  (and (involved-in-projects ?p1 ?project)
             (or (has-project-leader ?project ?p2)
                 (has-project-member ?project ?p2)))
        (and (or (technology ?d) (document ?d))
             (has-author ?d ?p1)
             (has-author ?d ?p2)))
             (not (= ?p1 ?p2))))))
```

Axioms

Used for additional constraint checking

```
(def-axiom agrees-and-disagrees-are-mutually-inconsistent
  (forall (?a ?y)
         (not (exists (?x1 ?x2 ?z ?z2)
                      (and
                       (agrees ?x1 ?y ?z)
                       (disagrees ?x2 ?y ?z2)
                       (member ?a ?x1)
                       (member ?a ?x2))))))
```

Formulas as values

```
(def-class classification-task (goal-specification-task) ?task
  ((has-goal-expression
    (:default-value
       (kappa (?task ?sols)
           (forall ?sol
             (=> (member ?sol
                        (role-value ?task 'has-solutions))
                    (admissible-solution
                     ?sol
                     (apply-match-criterion
                      (role-value ?task 'has-match-criterion)
                      (role-value ?task 'has-observables)
                      ?sol)
                     (role-value ?task
                      'has-solution-admissibility-criterion)))))))
```

Things we would also like to have

- Comprehensive meta-level
  - — Clean way to annotate individual statements

    (def-relation criticises (?person ?statement)

- Mechanisms to define inference schemas
  - — E.g., new inheritance mechanisms for different part-of relations