Why Do We Need a Meta-Level for the CRM?

Guenther Goerz
Univ. of Erlangen-Nuremberg, C.S.D. /8
goerz@informatik.uni-erlangen.de

Overview

- The Starting Point
- Abstraction, Concepts, and Tractable Inferences
- Generics and Defaults: The Semantic Dimension
- The Potential of Default Reasoning
- Implementation: Answer Set Programming
- (Still) Open Questions

The Starting Point

Categorical Documentation

"Categorical documentation"

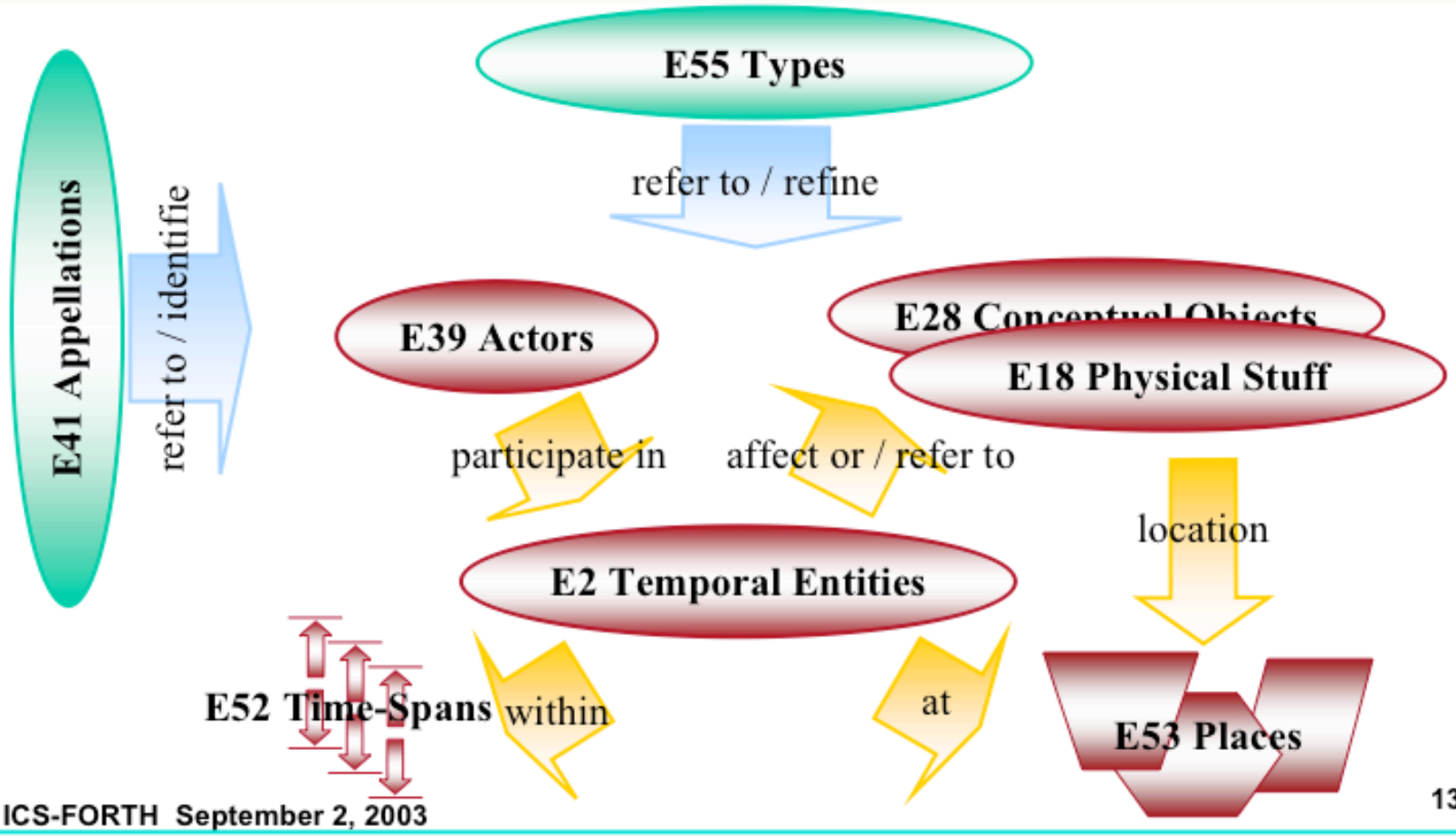
- E.g. "Stuffed Fringilla coelebs 1965-0034"

  - objects not unique, with normal production value,
    - used as example out of specific context

    - Such as most objects in Natural History, ethnological collections, many archeological objects like frequent types of pottery etc.
    - Documentation focus is "representative of its category":
      taxonomic role, deviations from prototype,
      type of context of provenance, of use;
      factual context only a statistical element for induction.
    - Classification and categorical behaviour is the information, the object and its context is only an attribute.

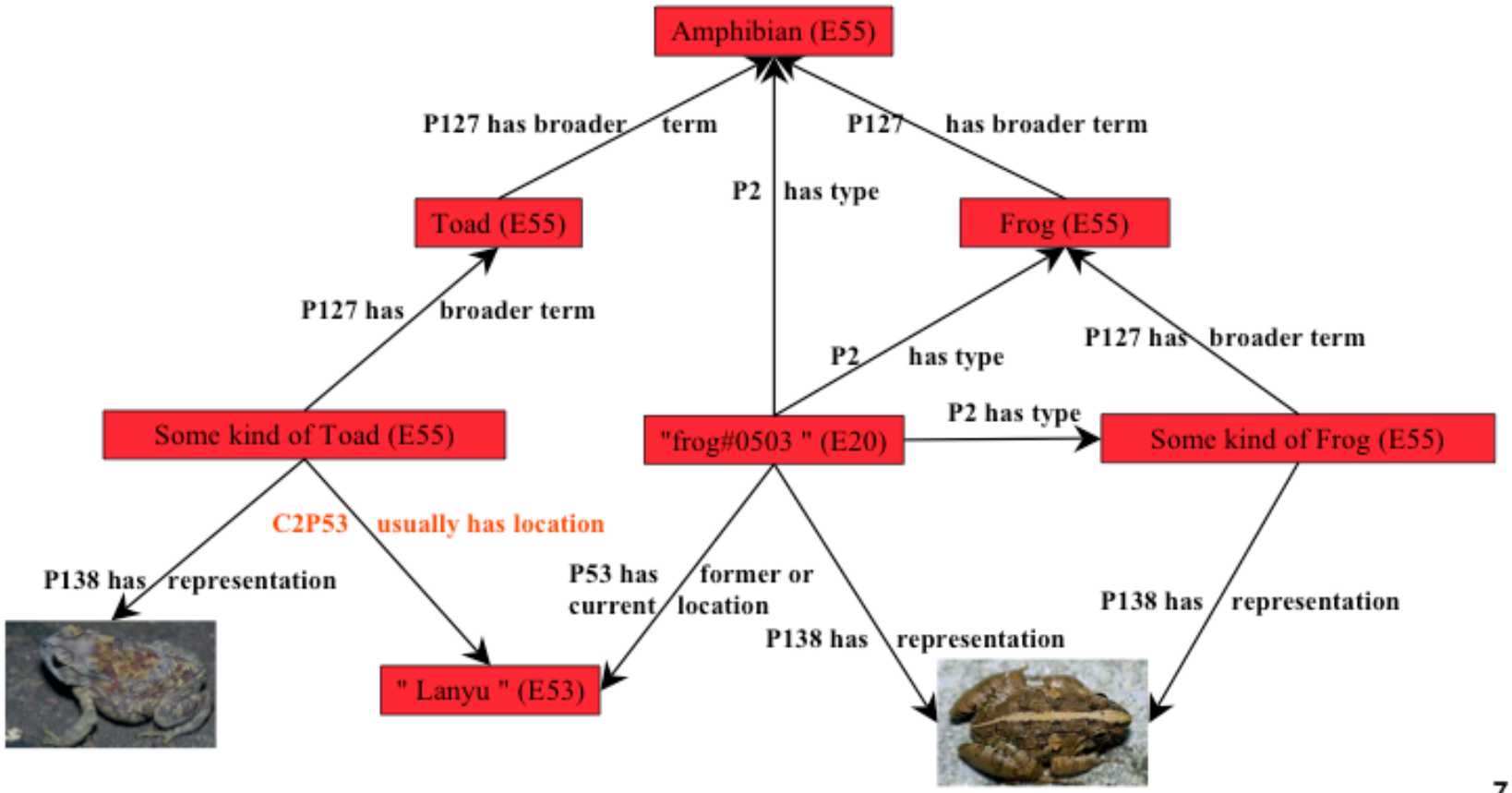
Categorical Documentation

The CIDOC CRM (ISO/CD21127) Top-Level



Categorical Documentation

Biological Example (from ChiNan University)



Categorical Documentation

Instantiation levels

Look at three kinds of knowledge elements:

- factual: "My cat – ate – my mullet" = item- relation – item
- cross-categorical: "My cat –ate - fish" = item- relation – class
- categorical: "cats-eat-fish" = class-relationship-class

- Interpretation of factual statement is unique:

  "the predicate: ate(my cat, my mullet) holds".

- Interpretation of categorical and cross-categorical statements is not unique.

Abstraction, Concepts, and Tractable Inferences

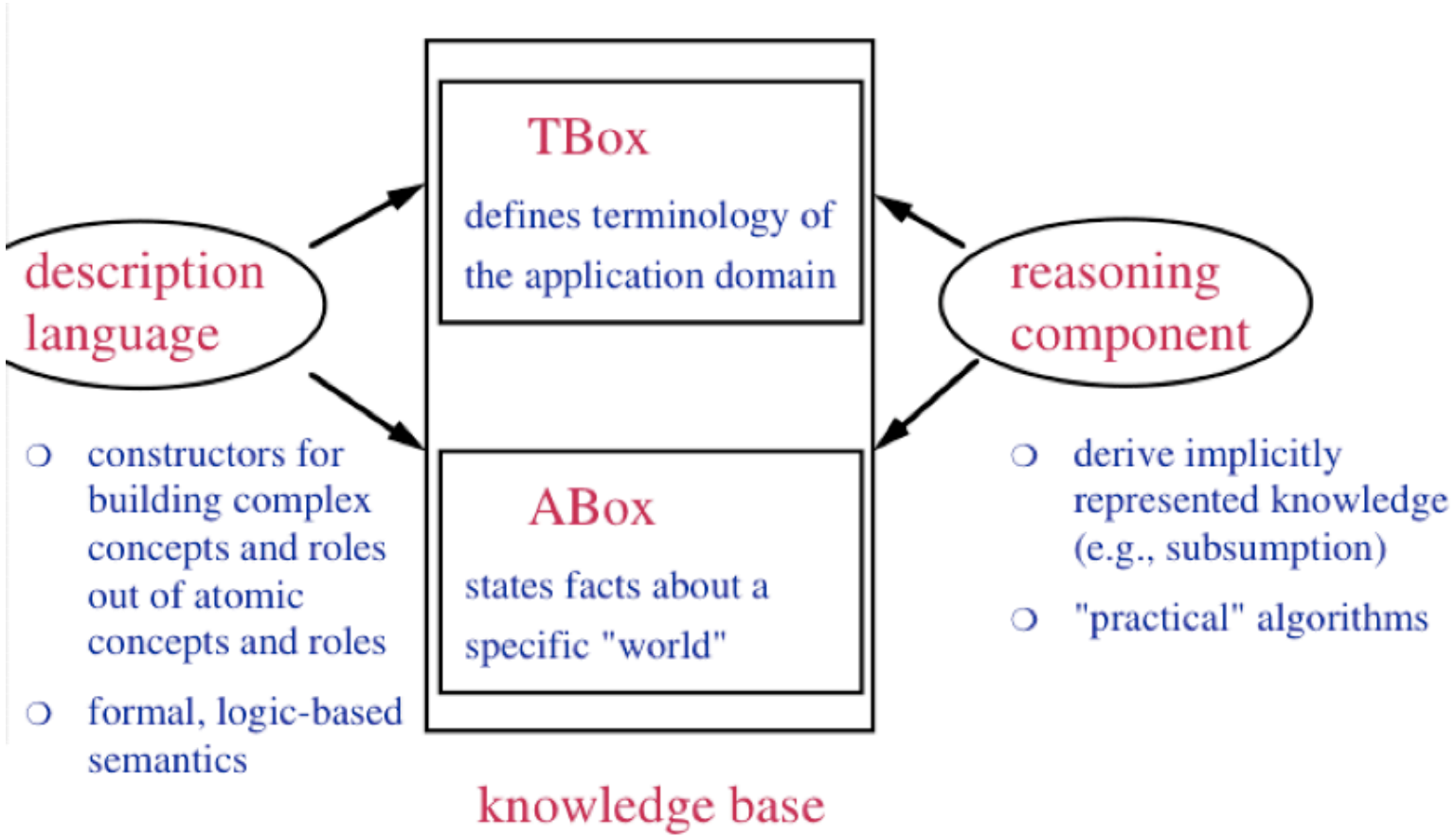
Frege: *Abstraction* as a constructive procedure:

- Build equivalence classes of objects with (positively expressed) equal properties
- Introduce the notion of *hypothetical **abstract*** objects – which have only the *common* properties – to express that certain statements are invariant w.r.t. such an equivalence relation
- Abstraction schema $A(\tilde{x}) \rightleftharpoons \forall y(x \sim y \rightarrow A(y))$ leads to a new expression for invariant statements *A* with *abstractor* α*:* $A(\alpha x)$ with "*abstract* object" αx
- E.g.: Functional abstractor λ, set abstractor $\in$

Logical Framework: Description Logics

- Decidable, efficiently implementable sub-languages of FOL (subset of $\mathcal{L}_3$)
- Suitable for KR: Representation at *predicate* level
  - Intensional level (T-Box): Definition of **concepts**
    - Concepts: inheritance hierarchy (subsumption lattice)
    - Roles: (binary) relations (hierarchy)
    - Axioms
  - Extensional level (A-Box): Assertions over **individuals** (instances; CRM: "items")
  - Open world assumption
  - *Complete* and *sound* inference procedures exist

Description Logic Systems



Description Logics

- Concept expressions:
  - Necessary and sufficient conditions
- Role-defining expressions
  - No need for predefined shortcuts
- Inferences:
  subsumption, satisfiability, consistency, instantiation
- Analytical reasoning with concepts is straightforward,
  e.g. (in informal notation)

  Person *subsumes*

  (Person with every Male Friend [who] is-a Doctor) *subsumes*

  (Person with every Friend [who] is-a

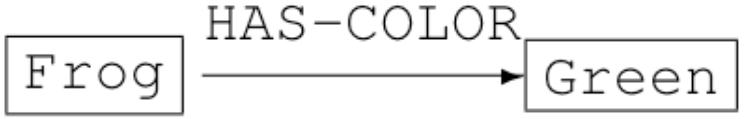
  (Doctor with a Specialty [which] is Surgery))

Categorical Documentation

Problem Statement

- Current data structures are made to organize description of facts (particulars) by providing a system of classes (nodes, tables) and relationships (attributes, links).
- No difference is made between data that are particulars and those that are universals. Inheritance of properties due to instantiation or subsumption of universals appearing as data cannot be described.
- Data in manufacturing (spare parts), ethnography, natural history and others have this problem.
- Few work in knowledge representation about metamodels and their relations to simple models.
- Missing: A theory/proposal of data structures relating particulars and universals – i.e. "cross-categorical data" in a logically well-defined way.

Ambiguities of Quantification

What is the exact meaning o 

?

- *Every frog is just green*
- *Every frog is also green*
- *Every frog is of some green*
- *There is a frog which is just green*
- *...*
- *Frogs are typically green, but there may be exceptions*

Disambiguating the Graph ...logically

- *Every frog is just green*

$$\texttt{Frog} \sqsubseteq \forall \texttt{HAS} - \texttt{COLOR}.\texttt{Green}$$

- *Every frog is also green*

$$\texttt{Frog} \sqsubseteq \exists \texttt{HAS} - \texttt{COLOR}.\texttt{Green}$$

- *There is a frog which is just green*

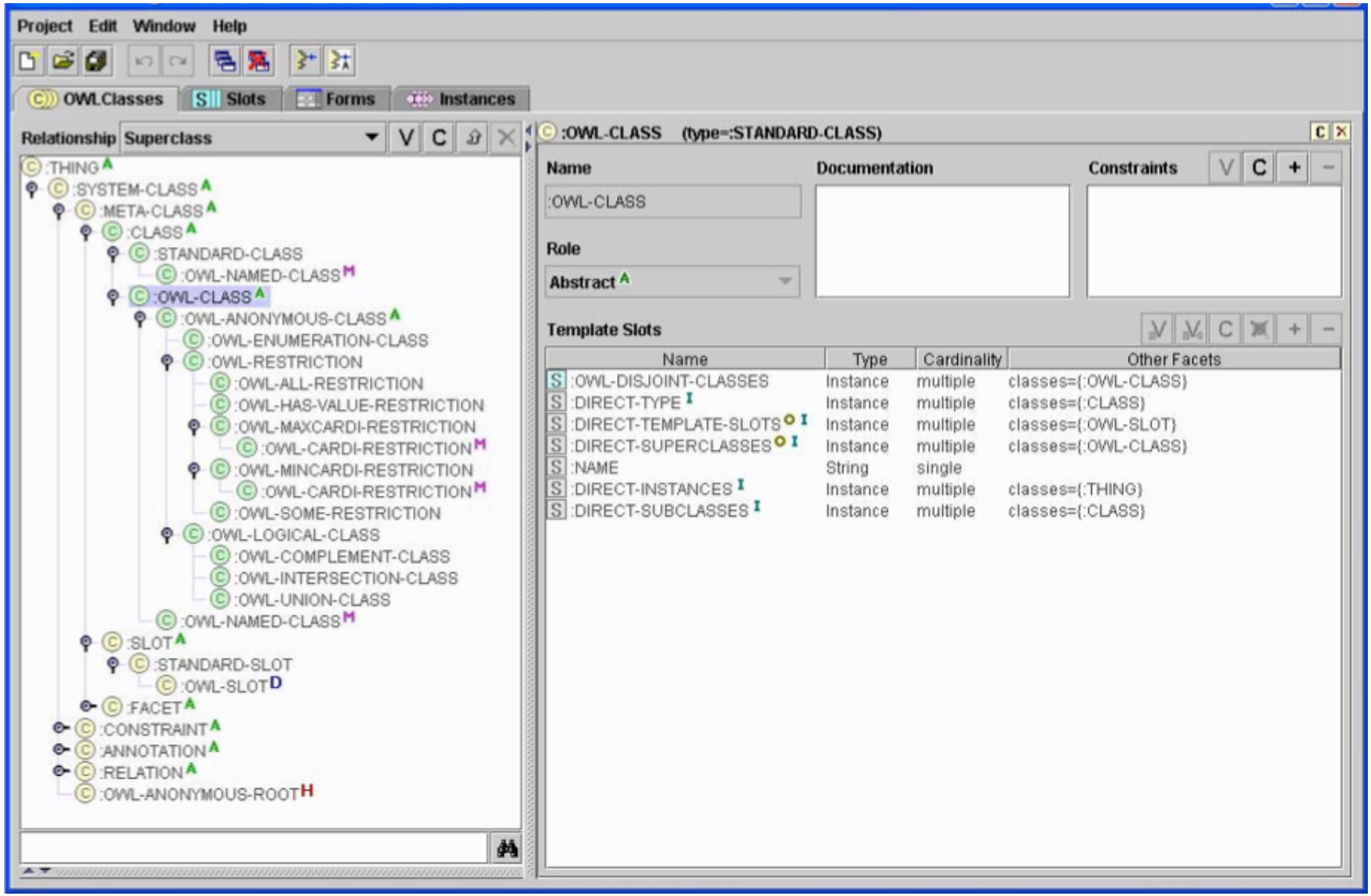
$$\texttt{Frog} \sqsubseteq \forall \texttt{HAS} - \texttt{COLOR}.\texttt{Green}$$

$$\texttt{Frog}(x), \texttt{HAS} - \texttt{COLOR}(x, y)$$

General Observations

- The meaning of most object-oriented representations can be logically very ambiguous.
- The appeal of graphical representations of object-oriented systems has led to forms of reasoning that are not covered by standard logical categories, and are not yet well understood.
- Unfortunately, it is much easier to develop some algorithm that appears to reason over structures of a certain kind than to *justify* its reasoning by explaining what the structures are expressing about the domain.

The Protégé OWL(-DL) *Meta*-Model



... a Matter of the Meta-Model !?!

Categorical Documentation

Categorical relationships in the CRM

- *E55 Type* represents a metaclass. All CRM classes can be regarded as instances of *E55 Type*. The property *P2 has type* means *instance-of.*
- *E55 Type* is related by *P127 has broader term,* meaning *IsA*.
- Important cross-categorical relationships are defined, such as: *P125 used object of type*, together with the respective factual one: *P16 used specific object.*
- *E55 Type* is also treated as simple class in the sense of a product of the human mind.
- *There are no other categorical relationships*

Universal and Prototype Views

- Combination of a prototype-based view with a conceptual ("universal") property- and class-based one?
  - In factual documentation objects are always unique, categorical documentation is about examples
  - But: uniqueness condition needs not to be given up
    - Cases where uniqueness (= monotonicity) still holds
      - allowing for variations in irrelevant properties
    - Examples where property values may be overwritten (non-monotonicity)

Categorical Documentation

Interpretation of categorical relationships

- Interpretation of categorical relationships is not unique:

  1. eat (Cat,Fish) ⇔ ∃ x:Cat,y:Fish (◊ ate(x,y)) = some cats can eat some fish
  2. eat (Cat,Fish) ⇔ ∃ x:Cat,y:Fish (ate(x,y)) = some cats have eaten some fish
  3. eat (Cat,Fish) ⇔ ∀ x:Cat ∃ y:Fish (◊ ate(x,y)) = all cats can eat some fish
  4. eat (Cat,Fish) ⇔ ∀ x:Cat,∀ y:Fish (◊ ate(x,y)) = all cats can eat all fish
  5. eat (Cat,Fish) ⇔ ∀ x:Cat (ate(x,y) ⇒ Fish(y)) = all cats can eat only fish

  * * * * * *

  7. card {x: Cat(x) ∧ ∃ y ∧ Fish(y) ∧ ate(x,y)} / card {x: Cat(x)} > 0.1 = more than 10% of all cats have eaten some fish.

- Case 1. is the normal meaning of a relationship in a schema and most generic, but normally too week. Frequently, we want to register a typical behaviour, more like case 7.

- We propose : Cat “usually eats” Fish, or Cat “typically eats” Fish.

Comments

- Separate carefully expressive means
  - ⇔ means: "is interpreted as"
  - Don't use modal operators
    - Either "can eat" = "eat" or use ***new*** predicate "can-eat"
    - Otherwise, introduce equivalent to McCarthy's modal functions like *can(.)*, if definitely required
  - Use only present tense (for the sake of simplicity)
    - Look for a generic representation of tense later on, e.g. Reichenbach's time "points" *e, s, r*
- Cross-categorial expressions in strict cases are logically transparent

Generics and Defaults: The Semantic Dimension

- Typicality is a separate issue
  - problems we looked at exist without talking about typicality
- Now: ***assumptions*** and ***exceptions***
- **Generics**: properties that hold "in general" – admitting exceptions – as opposed to universals (properties that hold over all instances)
  - Kind-referring predication ("*the frog*" or "*frogs*") vs. object predication
  - Expressing a kind of general property
  - Habituals: A regularity of action is predicated of an ordinary individual

Common Uses of Defaults

1. General statements
   - normal: under typical circumstances, *P*s are *Q*s (*frogs live on trees*)
   - prototypical: the prototypical *P* is a *Q* (*frogs are green*)
   - statistical: Most *P*s are *Q*s
2. Lack of information to the contrary
   - familiarity: if a *P* was not a *Q* you would know it
   - group confidence: All the known *P*s are known or assumed to be *Q*s
3. Conventional use
   - conversational: a *P* is a *Q* unless I tell you otherwise
   - representational: a *P* is a *Q* unless otherwise indicated (*speed limit in a city*)
4. Persistence
   - inertia: a *P* is a *Q* unless something changes it (*position of objects*)
   - time: a *P* is a *Q* if it used to be a *Q* (*color, size of objects*)

Generics and Exceptions

- What is the relationship between generic statements and explicitly quantified statements?
- Generic ("characterizing") statements are *intensional*
- Explicit statements of regularities are *extensional*, not generics
  - Examples: "mostly", "typical", "normal"
  - *Claim*: The cases we are looking at in natural history and cultural heritage documentation are extensional
    - considering primarily individuals, e.g., specimens in botanics are individuals (cf. Daston)

Generics and Exceptions (2)

- Approaches to the problem of generics:
  - Generic statements are strictly speaking false, but acceptable (exceptions!)
  - Generic statements are neither true nor false
    - Treatment as *inference rules*
    - Cannot be embedded within one another (important??)
  - Generic statements have a truth value (model-theoretic view)
    - But: How many exceptions can a generic statement tolerate?
    - There is no univocal *quantifier* which works for all generics (including vague quantifiers)

Analysis of Generic Statements

- Give some account of truth conditions (?)
- Explain the about genericity (laws) vs. quantified, extensional statements
- Use of generic statements in reasoning (...exceptions!)

$\Rightarrow$

GEN operator (Pelletier/Asher)

- Three parts: variables, restrictor, main clause
- Example: "*Frogs live in this part of Africa*"

  GEN[x](x *are frogs*; ∃y[y *is this part of Africa* & x *live in* y])

Semantics of the GEN Operator

- Candidates for interpretation
  - Relevant quantification (∀ over relevant objects)
  - Abstract objects (singular predication over abstract object)
  - Prototypes (same nature as ordinary objects)
  - Stereotypes (extension + stereotypical properties)
  - Modal conditionals (possible worlds)
  - Situation semantics (expressing constraints)
  - Default reasoning approaches
- Formal theory: interpretation as a ***conditional operator***
  - Axiomatization & model theoretic semantics… meeting all requirements – but: implementation???

Arguments pro Default Reasoning

- Assumption: The extensional view is the relevant one for CRM applications
- Is there a need to stick to the claim that generic statements are essentially truth-conditional?

vs.

- The significance of generic statements lies in their "dynamic" meaning,
  i.e. update conditions for information states
  ⇒ default inference (Veltman)

The Potential of Default Reasoning

- **Default reasoning**: If a *P(.)* is generally a *Q(.)* and *P(a)* is true, then it is reasonable to conclude that *Q(a)* is true unless there is a good reason not to.
- Generic statements like *"Birds fly"* interpreted extensionally as *"In general* [etc.], *birds fly"*.
- Proposal:
  It is sufficient to consider generic objects (*typical_bird*) as arguments for strict relations (*fly* vs. *typically_fly* [??])
- To talk about typical behaviour, introduce a generic concept for it in the first place

Default Logic[1]

- Special default rules: KB is a *default theory* consisting of two parts:
  - a set *F* of first-order sentences
  - a set *D* of default rules which specify what assumptions can be made and when
- Mechanism for specifying explicitly which sentences should be added to KB when it is consistent to do so
- Problem: Can't reason **about** defaults

[1] one of several accounts of non-monotonic reasoning

Default Rules (Reiter)

- Default inference rule:
  If *x is a bird* is true and the fact that *x flies* can be consistenly assumed, then conclude that *x flies* is true

$$\frac{bird(x) : fly(x)}{fly(x)}$$

Consequences for the Meta Level

- For generic propositions, we need only **one** operator, like **GEN**,
- which – the extensional case assumed – can be implemented in terms of default inference rules (⇒ meta level!),
- and which in turn can e.g. be generated by means of macro expansion.
- *... Counterexamples??*

A Viable Solution?

Property	Property Name	Entity – Domain	Entity - Range
CP1	is usually identified by (usually identifies)	T1 Type of CRM Entity	T41 Type of Appellation
CP5	usually consists of (usually forms part of)	T3 Type of Condition State	T3 Type of Condition State
CP7	usually takes place at (usually witnesses)	T4 Type of Period	T53 Type of Place
CP8	usually takes place on or within (usually witnesses)	T4 Type of Period	T19 Type of Physical Object
CP9	usually consists of (usually forms part of)	T4 Type of Period	T4 Type of Period
CP10	usually falls within (usually contains)	T4 Type of Period	T4 Type of Period
CP11	usually has participant (usually participates in)	T5 Type of Event	T39 Type of Actor
CP12	usually occurs in the presence of (is usually present at)	T5 Type of Event	T77 Type of Persistent Item
CP13	usually destroys (is usually destroyed by)	T6 Type of Destruction	T18 Type of Physical Stuff
CP14	is usually carried out by (usually performs)	T7 Type of Activity	T39 Type of Actor
CP15	is usually influenced by (usually influences)	T7 Type of Activity	T1 Type of CRM Entity
CP16	*usually uses type of object (is usually used for)*	T7 Type of Activity	T70 Type of Stuff
CP17	is usually motivated by (usually motivates)	T7 Type of Activity	T1 Type of CRM Entity
CP19	is usually intended use of (is		

...up to CP141

Implementation Options

- DLP: Description Logic Programming
  - Intersection of description logics and logic programming (rules)
- ASP: Answer Set Programming

  as a preprocessing module to a DL system
  - A constructive, declarative programming paradigm related to conventional logic programming, but including classical negation
  - simple and efficient model generation based on stable model semantics
  - Many specialized answer set solvers exist as smodels^{A}, dlv, cmodels, ...; cf. also XSB

Integration of Rules and Ontologies

- Allows for building rules on top of ontologies and, to a limited extent, building ontologies on top of rules
- In our case: Combination of CRM as a DL (OWL) ontology with default rules for A-Box reasoning
  - Default rules affect **only the A-Box** (extensionality assumed):
  1. Propositionalization („grounding“) of the rules over the actual A-Box, making use of T-Box relations
  2. Evaluation, i.e. model generation as a new, additional form of instance generation

A Default Rule in ASP

- $\frac{bird(x) : fly(x)}{fly(x)}$ translated to ASP

- ```
  fly(X) :- bird(X), not -fly(X).
   % penguin(X) => not fly(X)
  -fly(X) :- penguin(X).
  bird(X) :- penguin(X).
  bird(tweety).
  ```
- Answer set: `{fly(tweety), bird(tweety)}`
- Replacing the last line by `penguin(tweety).`

$\Rightarrow$ New answer set:

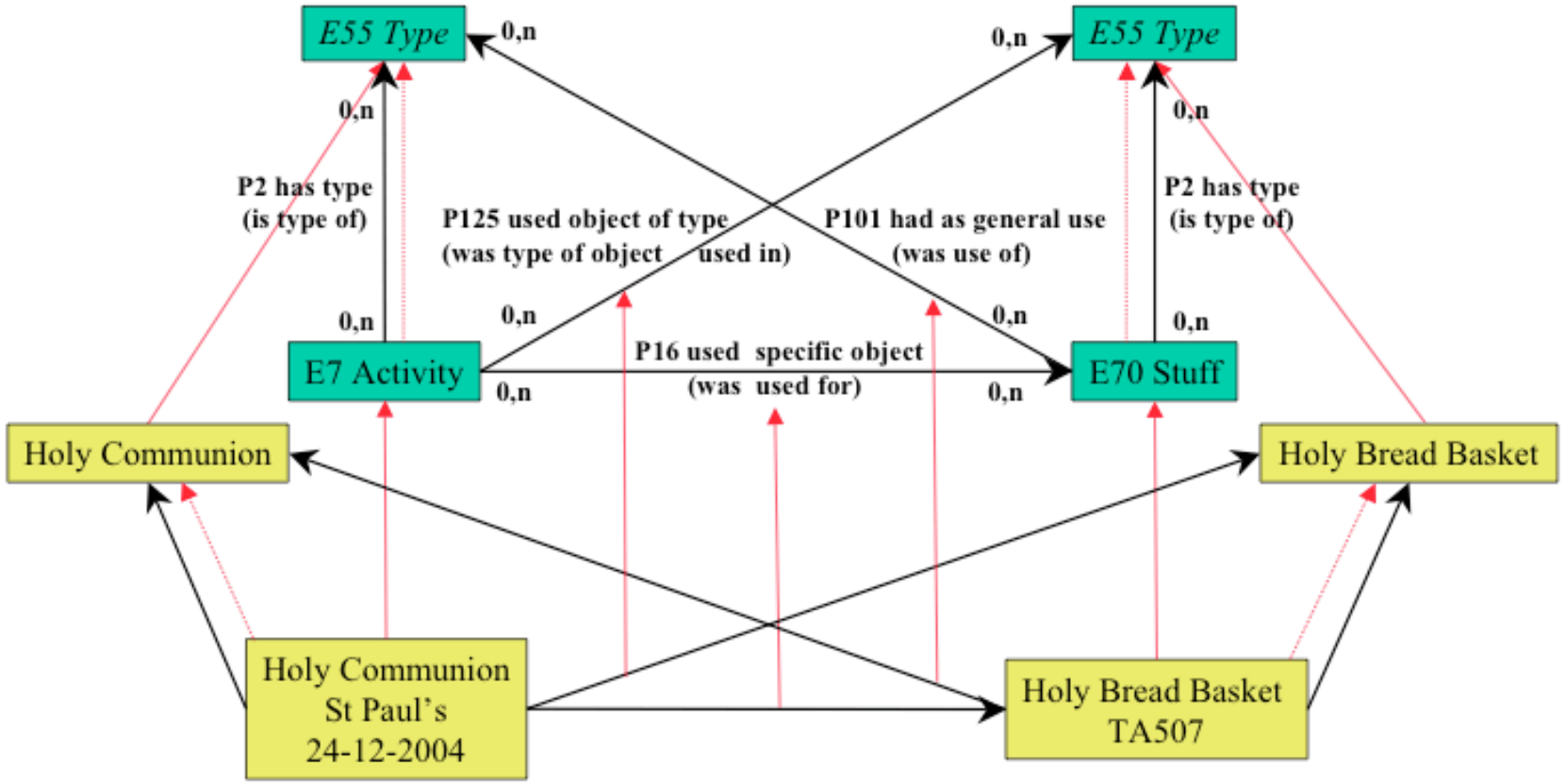
`{penguin(tweety), bird(tweety), -fly(tweety)}`
  ```

(Still) Open Questions

- What is the meaning of the following example
  - Does it suggest a graph transformation?
    ...Hard to understand: the second graph
  - What can be inferred?
    - There are two kinds of edges: “normal“ ones and “usually“ edges
    - What is the interpretation of the “usually“ relations – as opposed to the “normal“ ones?
  - Consequences for complexity?
  - Implementation?

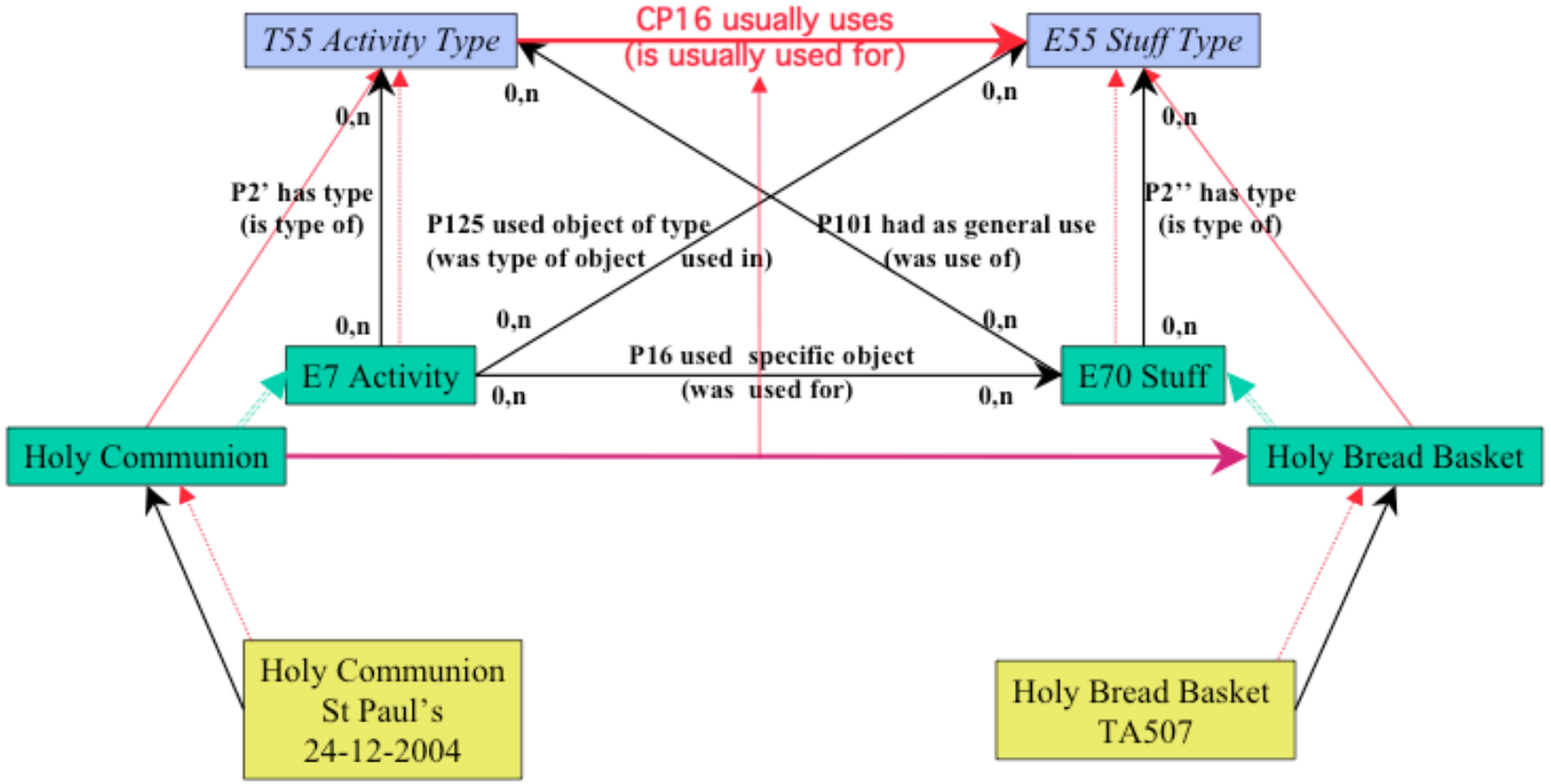
Categorical Documentation

CRM cross-categorial relationships



Categorical Documentation

New categorical relationships for the CRM



To be discussed

- Modalities
  - epistemological
  - Intentions, multi-agent scenarios
  - Separate representation layer to deal with epistemological and pragmatic questions
  - Modal logic??
    - Cf. modal functions (McCarthy)
- Time and tense
  - Dating as a classification problem?
  - General requirements for temporal reasoning?
- Statistical reasoning
  - Yet beyond scope

Formal Ontologies

- Formal Ontology
  - Standardized terminological/conceptual hierarchy
    - Concepts („is“ - intransitive, substance)
    - Relations („has“ - transitive, accidents)
  - Axioms: constraints; rules, ...
- Reference ontologies
  - Generic, universal conceptual inventory
    Representation language and fundamental distinctions
  - Foundational relations: parts & wholes (mereology),
    similarity, dependence, connection, inherence, temporal order
- Application ontologies
  - Modelling particular application domains

ASP: Logic Programs

Answer set logic programs consist of rules of four types

- Basic rules: $a \leftarrow b$
- Choice rules: $\{a\} \leftarrow b$
- Constraints: $\bot \leftarrow b$
- Aggregate rules: *see below*

Intuitive Meaning of the Answer Set

- Consider the following simple program of three rules

  a ←

  {b}← a

  c ← b

- Rule 1: a *must be* part of the solutions
- Rule 2: b *may be* part of the solution *if* a is in the solution
- Rule 3: c *must be* part of the solution *if* b is in the solution
- There are two answer sets of this program: {a}, {a b c}

Intuitive Meaning of the Answer Set (2)

- Add the following constraint to the program

  ← c

  a ←

  {b} ← a

  c ← b

- Now there is only one answer set: {a}
- The constraint "← c" weeds out {a b c}

The Idea Behind Programs with Aggregate Rules

- Consider the program

  set(a). set(b). set(c).

  twoElementsSet ← count({X,set(X)})=2.

  threeElementsSet ← count({X,set(X)})=3.

- Its only answer set is:

  {set(a) set(b) set(c) threeElementsSet}