Why do it the hard way?  
The case for Expressive Ontological Schemas for  
SNOMED  

Alan Rector and Sebastian Brandt  
School of Computer Science  
University of Manchester  
rector@cs.manchester.ac.uk  
www.co-ode.org
The Issues

► SNOMED’s schemas were designed to fit DL & hardware technologies from the late 1980s / early 1990s
  ► Many constraints are now (probably) outdated
  ► “End stage evolutionary development”

► Impose major costs in complexity and development effort
  ► Typical “query” is at least 1 page long
  ► Huge effort on “subsets” that are difficult to build and will be much more difficult to maintain
  ► Great difficulty in binding to EHRs and Messages
    • Length and complexity of the Terminfo document

► Are better schemas possible?
  ► If so, what do they require of the implementation language?
The Opportunity

- SNOMED is just the beginning of deployment
  - Very few real records are recorded with complex SNOMED expressions
  - Little software uses post-coordination or interacts with the schema in significant ways
  - The organisational changes following the formation of the IHTSDO give a major opportunity

- Change easier now than later
  - Before building massive “pregacy”
  - Before software starts to use the DL structure seriously
  - Before an even more massive combinatorial expansion of terms
    - Do we enumerate all the noun phases in medical English?
  - Although change is always hard
    - At least a major feasibility study is warranted
Goals for improved semantics

► Easier more consistent development
  ► More effective Quality Assurance
  ► Easier / more effective specification of use
    • Subsetting
    • Querying
    • EHR-Terminology - and other application - binding
    • Effective post-coordination

► A simple test for improved semantics:
  ► *Do the specs / documents / queries / subset mechanisms get shorter?*
    • Measure improvement by the length of SQL/TQL required by applications
      ‣ *By the length of the documents*
Basic proposals

► Represent all “recordable codes” uniformly as “Situations”
  ► Eliminate the separate classification of “Situations with explicit context”
    • Distinguish “kernel codes/concepts” from “recordable codes/concepts”
  ► Rationalise the context model
    • Including separating negation from the rest of context

► To represent “observables” and “findings” in such a way that the equivalences and implications can be inferred uniformly.
  ► Recognise that “finding” and “observable” are meta concepts

► To be explicit about whether each site refers to the whole or the whole or its parts

► To build SNOMED as a series of modules
  ► And make it possible to load / use only those modules needed for a particular application
  ► Development strategy: Control the dependencies between modules
The starting point:
What does a code in an EHR mean?

► Entering a code in a record makes a statement about a patient -
  ► typically that a patient has something
    • (as observed by an observer at a time & place)

► Codes represent classes of predicates
  ► Entering a code for a “Recordable entities” is saying that a patient - as understood by a observer at a time and place - belongs to the corresponding class
    • ... but patients ARE NOT diseases or procedures
      › Patients HAVE diseases and procedures
Therefore ...

► Divide codes & entities into
  ► Kernel entities / codes -
    • the diseases & acts themselves
      (and the codes that stand for them)
  ► Recordable entities/codes -
    • that the patient has a kernel entities
      (and the codes that stand for having them)
  › as observed by an observer at a given time and place
Therefore:

► “Recordable concepts” are about the patient as observed by an observer at a time and place

► We label the units of information “Patient Situations” or “Situations” for short
  • Proxies for “the patient state as understood by an observer at a given time and place”
    ‣ e.g. “I state here and now that ‘the patient has X’” or “I state here and now that ‘the patient does not have X’”

► Recordable codes represent classes of “situations”
  ► Situation THAT includes SOME Condition
    • e.g. Situation THAT includes SOME Diabetes
  ► Situation THAT NOT includes
    • e.g. Situation THAT NOT includes SOME Diabetes
Situations & Context

► Situations *encapsulate* context and *include* kernel concepts
  ► May contain any boolean combination of kernel concepts

► Provide a uniform approach to context
  ► Not a new idea
    • GALEN and PEN&PAN both used them
      › *Allow conjunctions and disjunctions of kernel concepts as well as negation*
      › *But particularly useful given an expressive DL with negation*

► All contents included in a “situation” are relative to a single root subject -
  ► the subject of that EHR instance
    • i.e. seen from the point of view of care of one patient
      › *Even if actually about some other subject*
Rationalising the Context Model

The current SNOMED context model conflates

► “Modalities”
  ‣ The patient is at risk of cardiovascular disease
  ‣ The patient has a family history of type 2 diabetes

► Subject of care
  • The fetus has a depressed heart rate

► (Relative) temporal relative markers
  ‣ The patient has a history of myocardial infarction

► Negation
  ‣ The patient does not have diabetes
Separating out the cases - 0

Simple conditions

- e.g.
  - The patient has cardiovascular disease
  - The patient has type 1 diabetes

Represent as simple inclusion in situations

- Situation THAT includes SOME Cardiovascular_disorder
- Situation THAT includes SOME Diabetes_type_1
Separating out the cases - 1

► “Modalities”
  ► e.g.
    • The patient is at risk of cardiovascular disease
    • The patient has a family history of type 2 diabetes

► “Risk of CVD” is not a kind of cardiovascular disorder
  “Family history of diabetes” is not a kind of diabetes

► Represent as concepts derived from kernel concepts
  • Situation THAT includes SOME (Risk THAT is_of SOME Cardio_vascular_disorder)
  • Situation THAT includes SOME (Family_history THAT is_of SOME Diabetes_type_2)
Separating out the cases - 2

► Subject of care

► e.g.
  • “The fetus has a depressed heart rate”
  • “The mother is depressed”

► Represent as a nested situation within a situation

  • Situation THAT has_subsituation SOME
    (Situation THAT
      has_subject SOME Fetus AND
      includes SOME Depressed_heart_rate)

  • Situation THAT has_subsituation SOME
    (Situation THAT
      has_subject SOME Mother AND
      includes SOME Depression)
Separating out the cases - 3

► Relative temporal markers
  ► e.g.
    • History of Myocardial infarct

► Two alternatives
  ► Simpler alternative:
    Represent by analogy with modalities
    • *Situation THAT includes SOME*  
      *(History THAT refers_to SOME Myocardial_infarct)*

► More expressive alternative:
  Represent by analogy with subject of care
  • *Situation THAT has_subsitutution SOME*  
    *(Situation THAT  
      has_temperal_marker SOME History AND  
      includes SOME Myocardial_infarct)*

► We have no use case requiring more expressive alternative.
  ► *Do you have any?*
Separating out the cases 4

► Negation
  ► Entities not included in the situation
  ► e.g. The patient does not have cardiovascular disease

► Represent as “NOT includes SOME X”
  • Situation THAT NOT includes SOME Cardiovascular_disease
What we organise with logic are classes of situations, negation inverts the hierarchy e.g.

- **Positive**
  - “situations that include Metabolic Disorder”
  - “situations that include Diabetes”
  - “situations that include Type 1 Diabetes”

- **Negation**
  - “situation that do not include Type 1 Diabetes”
  - “situation that does not include Diabetes”
  - “situation that does not include Metabolic Disease”
Example - classic problems just fall out
e.g. Head trauma with/without fracture
   with/without Intracranial_bleed
   subdural_hematoma
   ...

Head Injury with and without intracranial bleed, before classification

Let the classifier do the work

after classification
Aside - 1

- Ignores problem of “negative findings”
  - e.g. “Absent pedal pulses”
    - See Medinfo paper for full treatment
        - Also accessible from my ome page
True negation does require a more expressive DL but...

- Negation has proved hard to ban negation from the terminology
- Current SNOMED/HL7 guidelines state that negation should be included in the code
  - But the representation does not capture so that it can be used correctly
    - We are “lying to the logic”
      - Even complete logical algorithms will not prevent false conclusions following from false premises.
Observables and Findings: Metacategories

► Basic maxim:
All statements should convey information

► There is no point in stating a tautology -
  • Or even something that is redundant given assumed background knowledge

► Key distinction

► Some entities are always present - “observables”
  • Qualities of the organism or one of its parts or functions
    › e.g. Body temperature, white count, skin colour, etc.
      - They must be described or given a value to convey information

► Some entities are variably present - “findings”
  • Pathologies or variations in the organism or its parts or functions
    › e.g. Diabetes, stroke, third heart sound, rales, wheezes, ...
      - Saying that they are present does convey information
Encapsulation

- Often use qualitative descriptions for qualities
  - e.g. Body_temperature = “elevated”

- Sometimes we encapsulate the whole thing as a single expression
  - “Elevated body temperature”

- The “recordable entity” for both is:
  - Situation THAT incudes SOME
    (Body_temperature THAT has_level SOME Elevated_value)

- In general an “Observable + Value” = Finding
  - So the observable / finding distinction cannot be a first order distinction
    - It is not about what things “are” but how they occur
      - A “meta distinction”
Testing for Findings vs Observables

The criterion is the meta-test for tautology / redundancy

IF Situation includes SOME Kernel_X \equiv \text{Situation}

THEN Kernel_X is an “observable”
ELSE Kernel_X is a “finding”

• In a complete ontology that included all axioms of the schema
  › Situation \rightarrow includes SOME Observable_type_X
    for all observables

Consequences:

All attempts to organise a subsumption hierarchy based on the Finding/Observable distinction are doomed to failure!

• Because it is a meta-distinction
  › Requires higher order rather than first order logic
    - ... although many branches of the subsumption hierarchy consist only of findings or only of observables
Developmental Modularisation and OWL - Plug and Play development

► OWL provides powerful import mechanisms
  ▶ Because an OWL ontology is just a collection of axioms, importing just means adding in the axioms
    • Order independent.

► Need to define interfaces between modules
  ▶ Analogy to API between java packages
    • OWL equivalence and subclass axioms provide a means of defining interfaces between ontologies
      ‣ *Usually by means of a separate module - an Ontology Binding Interface*
Technical aside

Possible in any DL or other based representation

- Any language in which adding modules is “just adding axioms” and axioms are “monotonic”
  - But not exploited until recently
    - OWL has stimulated the interest but
**Parts and Wholes**

“Adapted SEP Triples”

► **Requirement**

► **Distinguish disorders and procedures on the whole from those on the whole or its parts**

► **Example 1**
   - Our usual meaning of “Heart disease” is really “Disease of the heart or any of its functional parts”
   - But “Cardiomyopathy” affects the heart muscle as a whole
     › *Usually but not always classes of diseases of the whole include diseases of the parts*
     - In general faults of a part are faults of the whole

► **Example 2**
   - An “Amputation of a finger” is a “Procedure on the Hand” but not a “Amputation of the hand”
     › *Removals act on wholes*
     - although in general classes of procedures include procedures on parts
Ironic aside

► Property-paths were originally developed to meet this requirement
  ► GALEN’s refined_along/specialised_by
  ► SNOMED’s right identities
  ► OWL’s property paths

► There is a choice to put the burden on the properties or the classes
  ► But it seems more intuitive to put it on the classes
    • Lots of “druidisms” go away
      › Inspired by SEP triples from Schulz and Hahn

► ... but property paths turn out to be powerful inference mechanisms for other purposes
Typical representation of most broad disease and procedure classes

► Typical
  ► Heart_disorder ≡
    Disorder THAT has_locus SOME
    (Heart OR is_clinical_part_of SOME Heart)

► But
  ► Cardiomyopathy ≡
    Myopathy THAT has_locus SOME Myocardium

► Typical
  ► Procedure_on_hand ≡
    Procedure THAT has_locus SOME
    (Hand OR is_part_of SOME Hand)

► But
  ► Complete_amputation_of_finger ≡
    Removal THAT has_locus SOME Finger
Does not actually require true negation

- Can be done with “Psuedo-disjunction” - $A \lor B$
- syntactic sugar for creating SEP triples only when you need them

Formally:
- A lowest primitive concept $A$ $B$ that subsumes both $A$ and $B$
- $A \rightarrow (A \lor B)$
- $B \rightarrow (A \lor B)$
- but not
- $(A \lor B) \rightarrow (A \land B)$
Summary

► A cleaner set of schemas is possible
  ► Situations allow a clean revision of context model
    ◦ Eliminate of separate axis for “Situations with explicit Context”
    ◦ Make clear distinction between “recordable codes” and “kernel codes”
    ◦ Clean separation of:
      ‣ Negation, modality, temporal markers and subject of care
        - A formalism with negation handles classic troublesome cases easily
  ► Adapted SEP triples make part-whole representations clearer

► DLs allows modularisation and defintion of Ontology Binding Interfaces
  ► A strategy for building in pieces and “pluggable” extensions

► Only selective use of OWL is needed
  ► And only negation requires anything that can’t be encoded in EL+
    ◦ Dilemma of complete inference over theory conaining known falsehoods or incomplete inference over a theory without them
  ► Choices require empirical tests at scale - e.g. 25K concepts
    ◦ The time to do it is NOW!