Document outline

1. problem statement (firm). Include both general causes and specific needs.
2. approach (firm). Use of example requirements and example transforms to identify guidance
3. requirement classification (things to represent, circumstances in which we need to distinguish absence in these representations)
4. guidance
5. catalog of transform examples

**Problem statement**

The problem of negation in knowledge representation is complex and abstract

Much work in clinical modeling follows an object-oriented paradigm where software objects, or knowledge representations, are taken to stand for business objects. A patient object, for instance, stands for a patient, and represents certain characteristics of the patient it represents; similarly, an allergy record represents an allergy, and it captures such facts as the patient whose allergy it is and the substance that triggers a reaction.

The paradigm is quite intuitive, asserting a simple correspondence between things in the world and their representations in the system. There is little confusion around the fact that the patient object isn't really the patient, but that it can usefully stand in for the patient for certain purposes. The fact that the patient object is really record of an assertion about the patient, with its own context and boundaries, is true but more complex than is usually needed, and it can usually be ignored, especially when the object is used to associate properties or "predicates" -- name, birthdate, gender -- with the patient.

In addition to these "predicate" properties, such statements also imply existential statements about the objects on which the properties are predicated: the "John Doe" record tells us not only a name and birthdate; it also asserts the patient so identified exists. And as long as the instantiation of the business object is taken to mean that the thing exists, the complexities remain latent.

When, rather than predicating properties of an object, the business object is used to make existential statements such as asserting the absence of the real-world object, things get complicated. Suddenly, the fundamental semantics of the object have changed. Rather than the presence of an allergy, for instance, the class documents the absence of allergy to the very category of substances that would otherwise be understood to be of concern. The business object is revealed as not very much like its real-world referent after all.

How does such a knowledge representation indicate that it has violated the default object-oriented semantics of presence?

The methods used to stipulate this distinction to software systems vary. In a pure object system, the assertion of absence is a completely different kind of business object from the topic, so one might choose to use entirely different classes to represent presence and absence. Alternatively, one could use the same class, but add properties to the class to change its holistic semantics. (Such an approach would work best if the class were understood as an assertion about presence or absence rather than as a representation of the real-world object.) Finally, one could add this additional semantic dimension to the set of concepts used to classify the object; e.g., to a list of substances to which one might be allergic, add both "latex" and "no allergy to latex."

Any of these patterns could support consistency across specifications, were it adopted consistently. However, each has its own difficulties, so as a result, consensus on the topic has been elusive. All of these approaches have been used in HL7 standards.

If we leave the object paradigm behind, we don't necessarily escape the problem. A common approach to capturing allergies is with a simple question, "Allergies," to which valid answers may be "latex" or "not latex," replicating the issue of an answer that seems to change the meaning of the question. The difficulty may be addressed in this case by separating the question into its constituent parts: "does the patient have any allergies" and, if the answer is "yes," a second question, "what substance(s) is the patient allergic to?" This provides unambiguous information at the time of capture, but the standard patterns for interoperability, by assembling individual questions into business objects, regenerate the issue.

In logic, the distinction has long been recognized as one of "quantification": presence and absence are not predicates of objects (like name or birthdate) but special operators that assert existential quantity. At least one project is implementing a quantified model for analysis purposes, taking the logical "existential quantifier" and representing it consistently with other quantified elements such as "how many pressure ulcers does the patient have?" The pattern is not entirely intuitive, and whether it can be adopted for modeling information in clinical processes remains to be seen, but it should support robust analytical capabilities.

A finaly wrinkle: "negation" is a handy term for capturing both assertions of things that are not present and acts that were not done, but it introduces a level of abstraction that exacerbates the problem. We will try to use more specific terms, such as "absent" and "not done," in this document. Following the idea of logical quantification, we will typically use "absent" not only for observable phenomena but also for acts not done, reserving "not done" for discussions focused solely on acts.

The problem, then, is what can standards designers to do reduce the divergence among designs and enhance consistency and predictability for implementers.

**Approach**

The purpose of this document is to ameliorate the problem to the extent possible by

1. providing a high-level catalog of the kinds of negation one is likely to see
2. identifying characteristic difficulties
3. recommending best practices for future specification designs
4. providing a catalog of translations between common existing specifications, as recommended by their sponsoring workgroups

[detail]

**Classification**

**Guidance**

**Example transforms**