Introduction to Uncertain Datatypes in UML/OCL

In UML, logic predicates are represented by OCL expressions of type `Boolean`. This datatype represents the binary logic values `true` and `false`, which fit well with an objective view of a perfect world. In many real situations, however, we cannot be completely sure about the truth of these predicates. For example, whether a sensor is working or not, or whether it will rain tomorrow in Madrid.

For these situations, we have developed two additional datatypes (`UBoolean`, `SBoolean`) which allow expressing uncertainty about the truth of a predicate, at different levels. In this context, uncertainty can be defined as *"the quality or state that involves imperfect and/or unknown information."*

1. `UBoolean` - *Uncertain Boolean*

In previous works [1,2], we proposed an extension of UML and OCL type `Boolean`, called **`UBoolean`**, which adds to each `Boolean` value a real number in the range [0,1] that represents the likelihood that such a value is true. In other words, it allows `Boolean` values or predicates to be *partially* true. This datatype provides a *probabilistic extension* to binary logic.

Possible `UBoolean` values are, e.g., `(true, 0.99)`, or `(false, 0.75)`.

The equivalent values of `Boolean` in `UBoolean` are:

```
true  ≡ UBoolean(true, 1.0) ≡ UBoolean(false, 0.0)
false ≡ UBoolean(true, 0.0) ≡ UBoolean(false, 1.0)
```

Of course, the following expression always holds:

```
UBoolean(b,c) = UBoolean(not(b),1-c).
```

For example, `(true,0.9)=(false,0.1)`. The canonical form of `UBoolean` values is defined as `UBoolean(true,c)`.

2. UReal - Uncertain Real

In previous works, we defined another datatype, **`UReal`**, to represent *Measurement Uncertainty*. This datatype extends a `Real` value with another `Real` number that represents the dispersion (i.e., the standard deviation) of the values that could be attributed to the measurand.

Possible `UReal` values are, e.g., `(7.0, 0.02)`, or `(45.5, 3.01)`, which represent the uncertain real numbers 7.0 ± 0.02 and 45.5 ± 3.01, respectively.

Note that some aleatory uncertainty[1] happens when we compare two `UReal` numbers (for example, `3.0±0.1 < 3.1±0.1`). The result can be expressed in terms of the probability that one is, in fact, less

[1] *Aleatory uncertainty* refers to the inherent and irreducible variability associated with the physical system under consideration or its environment.

than the other. In this case, such a probability is 0.383.[2] This is why, in our proposal, comparison operations (such as "<", "<=" or "=") between `UReal` values, return `UBoolean` values.

3. SBoolean - Subjective Boolean

3.1 Motivation

Another type of uncertainty (called *Epistemic Uncertainty*) refers to the potential inaccuracy or vagueness due to the lack of knowledge that a subject has about a topic. In particular, *Belief Uncertainty* is a kind of epistemic uncertainty that occurs when a belief agent (e.g, a human being) is unsure about the truth of a statement.[3] This uncertainty is directly related to *trust*.

Although some authors propose the use of `UBoolean` values to represent such a subjective uncertainty, a fundamental limitation of probabilistic logic, in which type `UBoolean` is based, is the inability to take into account the modeler's level of confidence on the assigned probability, or being able to handle the situation where the modeler is not able to assign probabilities to a predicate due to lack of knowledge.

For example, when the modeler has total ignorance about some statement X, it might be preferable to say "I don't know" rather than assigning X a confidence of 0.5, because a confidence of 0.5 would mean than X and `not(X)` are equally likely, which does not represent ignorance since it is already quite informative. Moreover, forcing a modeler to set probabilities with little or no confidence could lead to unreliable conclusions. Therefore, we need to count on a formalism that distinguishes between the degrees of ***belief***, ***disbelief*** and ***uncertainty*** that someone holds about a predicate, since they are different aspects.

3.2 Subjective logic

Subjective logic is a type of probabilistic logic that explicitly takes *uncertainty* and *trust* into account. Subjective opinions express beliefs about the truth of propositions under degrees of uncertainty.

Expressions in subjective logic are called ***opinions***, and are defined by quadruples (b,d,u,a), where:

- *b* represents the degree of ***belief*** that the agent has about the statement
- *d* represents the degree of ***disbelief***
- *u* represents the ***uncertainty*** that the agent expressing the opinion has about the statement, i.e., the degree of trust
- *a* is the (objective) ***prior probability*** assigned to the statement (also called "***base rate***").

The following properties must always hold:

- b + d + u = 1
- b, d, u, a ∈ [0,1].

[2] Information about the propagation method/calculus available in references [1,2].
[3] In our modeling context, statements are defined in terms of Boolean predicates, which are normally expressed by OCL `Boolean` values or expressions.

For example, the quadruple *(0.7, 0.1, 0.2, 0.5)* represents an opinion of an agent A about a given statement *X*, meaning that the agent has a degree of belief of 0.7 that the statement is true, 0.1 that it is false, and is 0.2 uncertain about it -- given that the prior probability assigned to the statement is 0.5. This means that, initially, the statement *X* was equally likely to be true or false (hence the prior probability of 0.5), but the agent bets are 70% for the statement, 10% against it, and is 20% unsure about its truth. Here, the prior probability "a" normally corresponds to the initial (objective) confidence assigned to the statement, whilst (b,d,u) expresses the *subjective* opinion of the belief agent about it.

Opinions can be represented on an equilateral triangle using Barycentric coordinates as shown in Figure 1 below. A point inside the triangle represents the (b,d,u) triple for a statement *X*. Vertices at the bottom represent *absolute* opinions, and the vertex at the top represents the *vacuous* opinion (u=1). *Dogmatic* opinions belong to the baseline (u=0) and correspond to probabilities. The base rate "a", or prior probability, is shown along the baseline, too.

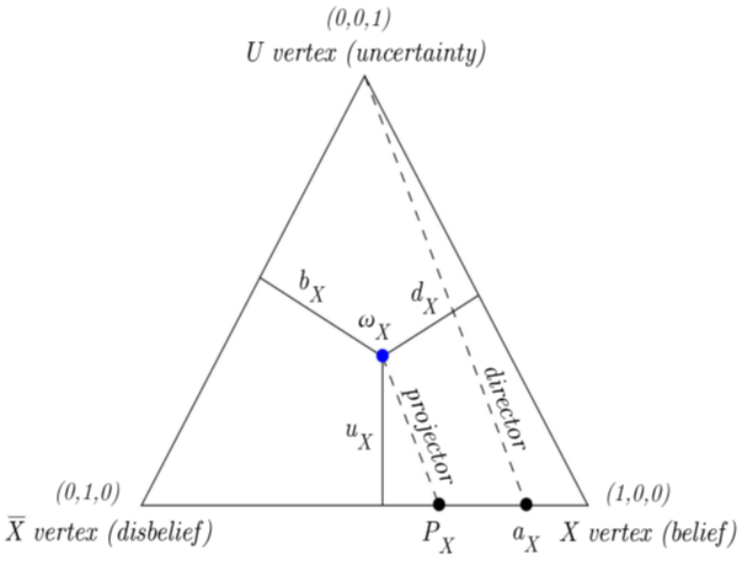

Figure 1. Graphical Representation of a subjective opinion

A key operation, called `projection()`, calculates the probability *P* of statement *X* to be true, using the prior probability "a" as the base rate and modifying it according to the subjective opinion of the agent, according to her degrees of belief, disbelief, and uncertainty — given by the triplet `(b,d,u)`. The projection "`P`" of a `SBoolean` value `X=(b,d,u,a)` is calculated as `P = b + a*u`. For example, if `X=(0.7, 0.1, 0.2, 0.5)`, then we have that `X.projection() = 0.8`.

This means that, given an *initial* probability (confidence) of `0.5` for statement *X*, and degrees of belief, disbelief, and uncertainty of `0.7`, `0.1`, and `0.2`, respectively, by a belief agent A, the confidence of "X" by agent A becomes `0.8`. In other words, we are able to represent that agent A has the confidence of 0.8 on statement *X* based on her subjective opinion, despite the fact that a priori, the statement had a probability to be true of 0.5.

We have extended the OCL and UML languages by declaring a new type, `SBoolean`, which enables the representation and management of opinions in subjective logic. The benefits are two-fold. First, opinions can be expressed in software models, enriching the current representation of subjective belief (or trust) on the logic predicates stated in a software model. Second, this information can be managed

in a transparent manner by the type system, providing a useful mechanism for reasoning about imprecise knowledge which can be supported by OCL tools.

Logic operators (`and`, `or`, `not`, `implies`, `equivalent`, etc.) are also defined for opinions, generalizing those of binary and probabilistic logic, as well as the behavior of operations on collections (`forAll`, `exists`, `select`, `collect`, `includes`, etc.) when they contain `SBoolean` values.

The lifting of basic `Boolean values` into `SBoolean` are:

```
true  ≡ SBoolean(1, 0, 0, 1)
false ≡ SBoolean(0, 1, 0, 0)
```

`UBoolean` values can also be lifted to `SBoolean` ones, as follows:

```
UBoolean(true, c) ≡ SBoolean(c, 1-c, 0, c)
```

References

[1] Loli Burgueño, Manuel F. Bertoa, Nathalie Moreno, and Antonio Vallecillo. Expressing confidence in models and in model transformation elements. In Proc. of MODELS'18, pp. 57–66, ACM, 2018. https://doi.org/10.1145/3239372.3239394.

[2] Manuel F. Bertoa, Loli Burgueño, Nathalie Moreno, and Antonio Vallecillo. Incorporating measurement uncertainty into OCL/UMLprimitive datatypes. Software and Systems Modeling 19(5):1163-1189 (2020). https://doi.org/10.1007/s10270-019-00741-0