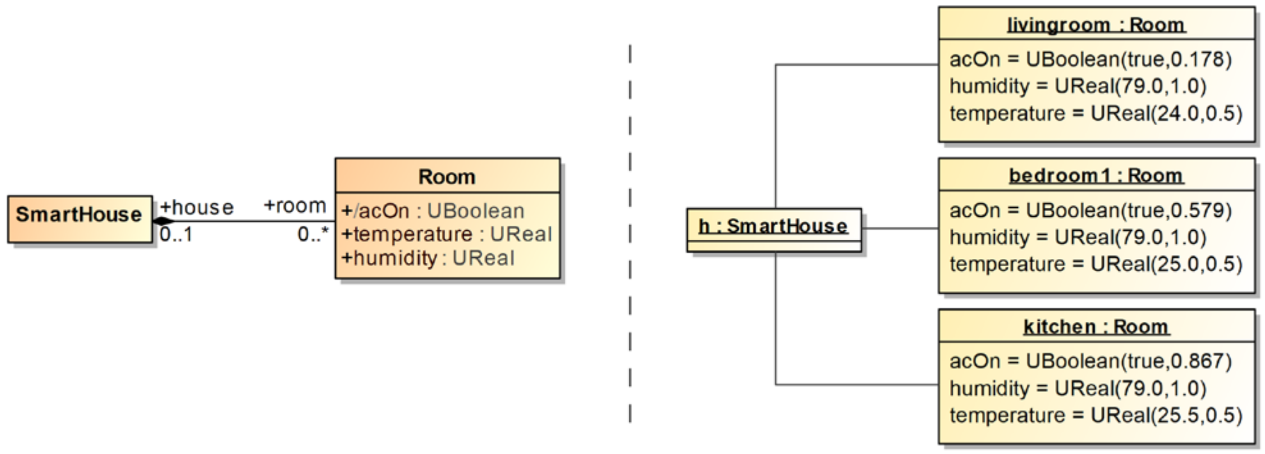
The AC system case study

1. Introduction

Let us suppose a smart house whose rooms are equipped with sensors that measure their temperature and humidity. According to their manufacturers' information, the accuracy of the temperature and humidity sensors are ± 0.5 degrees and ± 1%, respectively. A control device in the room decides whether the AC system should be turned on or off depending on the current values of the sensors measurements, using the following expression:

```
    acOn : UBoolean derive = self.temperature >= 25.0 or self.humidity >= 80.0      (1)
```

The diagrams below show the conceptual model of the system (left) and an object model that describes a house and three of its rooms (right). Note the use of the UML extended datatypes `UReal` and `UBoolean` that represent Real and Boolean values enriched with uncertainty [Bertoa et al, 2020]. The extended type system takes care of propagating the corresponding uncertainty though the OCL expressions and operations, and in particular the values of slot **`acOn`** are both derived using the OCL expression (1). Assuming a threshold of 0.5, in view of the object model, the AC system of the **`kitchen`** room should be switched on while that of the AC of the **`living room`** should be turned off.



SmartHouse
+house
0..1
+room
0..*
Room
+/acOn : UBoolean
+temperature : UReal
+humidity : UReal
h : SmartHouse
livingroom : Room
acOn = UBoolean(true,0.178)
humidity = UReal(79.0,1.0)
temperature = UReal(24.0,0.5)
bedroom1 : Room
acOn = UBoolean(true,0.579)
humidity = UReal(79.0,1.0)
temperature = UReal(25.0,0.5)
kitchen : Room
acOn = UBoolean(true,0.867)
humidity = UReal(79.0,1.0)
temperature = UReal(25.5,0.5)

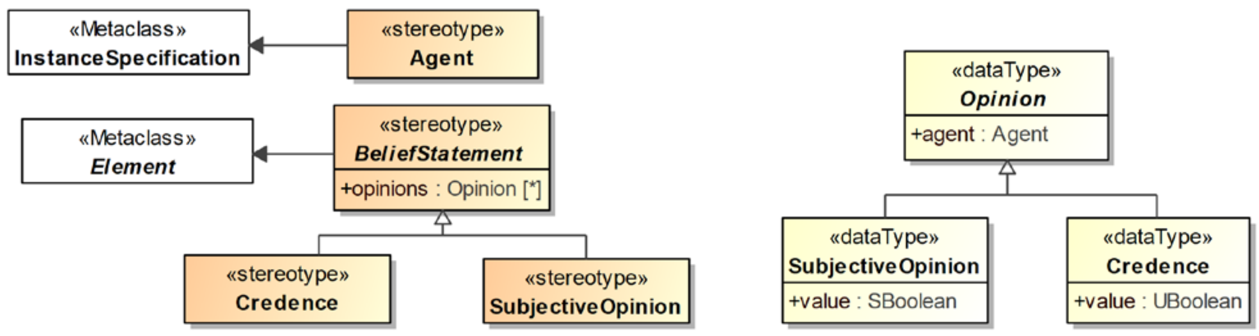
In many situations, the agents of such systems also associate subjective uncertainty with that of the sources. For example, one of the house occupants, Bob, does not trust the temperature readings of the kitchen because he knows that the sensor is close to the oven and therefore the measurements can be fairly unreliable. So Bob would add some subjective uncertainty to the objective results of the sensor’s measurements. Of course, this type of subjective uncertainty assigned to the same sensor by different agents can vary, depending on their personal history, experiences and beliefs (e.g., their individual level of trust in the machine manufacturer). Thus, other occupants may have different opinions, or trust, on the sensors' readings and consequently on their decisions to switch the AC system of a room on or off. Thus, the value of the derived attribute **`acOn`** should be somehow adjusted depending on the agent that evaluates the attribute, taking into account the subjective information added atop the objective probability provided by the initial model.

To further complicate matters, in most systems the agents do not work in isolation, but exchange information and interoperate with each other to achieve their goals. Think, for example, of agents driving unmanned vehicles in a city, or the avatars of human beings carrying out collaborative tasks such as games, performing robotic surgery or even software modeling. They must be able to combine their subjective opinions to reach a consensus on, for example, the next action to perform. In our example, if Ada, Bob, and Cam are occupants of the house and they hold different beliefs about the reliability of the temperature and humidity sensors, how to reach a consensus about turning the AC system on?

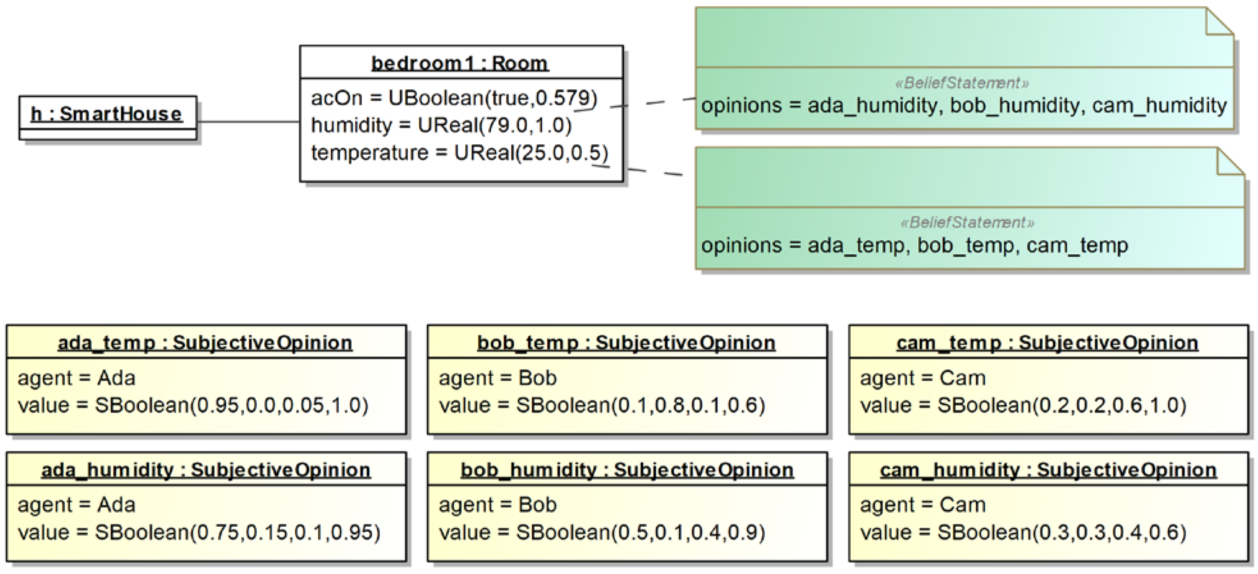
Let us see how to addresses these issues, in particular the representation of the subjective opinions of individual agents about specific model elements (objects, links or attribute values), the combination of these opinions with the objective uncertainty already associated to some of these elements, and how to merge all these uncertainties and beliefs to make combined decisions.

2. Enriching the model with subjective opinions

To add subjective opinions to the model elements we will use the Belief Uncertainty (BU) UML Profile, shown below.



Using this profile, an example of how to represent the opinions of the three house occupants (Ada, Bob and Cam) about the sensors in room **bedroom1** is shown in the following object diagram.


h : SmartHouse
bedroom1 : Room
acOn = UBoolean(true,0.579)
humidity = UReal(79.0,1.0)
temperature = UReal(25.0,0.5)
«BeliefStatement»
opinions = ada_humidity, bob_humidity, cam_humidity
«BeliefStatement»
opinions = ada_temp, bob_temp, cam_temp
ada_temp : SubjectiveOpinion
agent = Ada
value = SBoolean(0.95,0.0,0.05,1.0)
bob_temp : SubjectiveOpinion
agent = Bob
value = SBoolean(0.1,0.8,0.1,0.6)
cam_temp : SubjectiveOpinion
agent = Cam
value = SBoolean(0.2,0.2,0.6,1.0)
ada_humidity : SubjectiveOpinion
agent = Ada
value = SBoolean(0.75,0.15,0.1,0.95)
bob_humidity : SubjectiveOpinion
agent = Bob
value = SBoolean(0.5,0.1,0.4,0.9)
cam_humidity : SubjectiveOpinion
agent = Cam
value = SBoolean(0.3,0.3,0.4,0.6)

We can see how the three occupants have different subjective opinions on the values of the **temperature** and **humidity** slots of this object: Ada is quite confident in her opinions, Bob does not believe in the temperature sensor and partially trust on the humidity readings. Cam is not sure about the sensors either. The fact that agents hold opinions on the operands of the expression imply a derived opinion on the value of the derived attribute, which is calculated by propagating the opinions on the operands through the expression operations. This will be shown below when we specify how opinions are propagated, using in this case the operation **acOn_opinionOf(a:Agent)**.

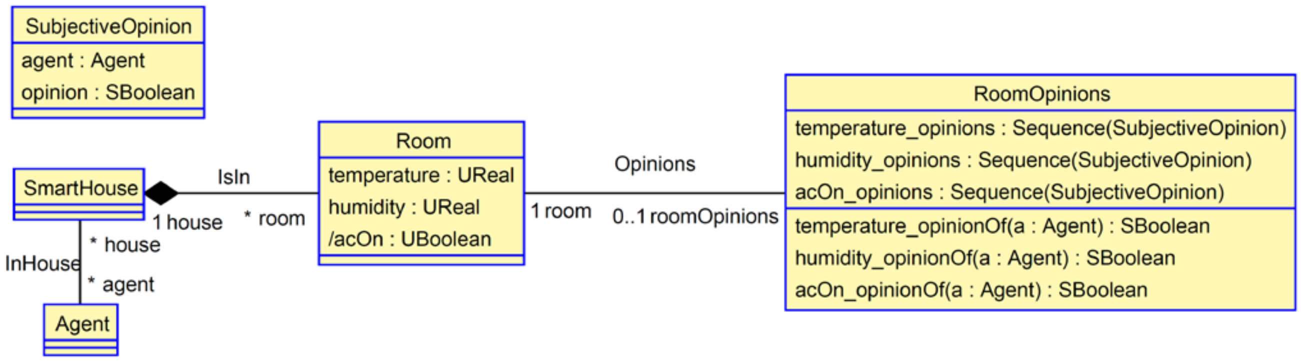
To operationalize the profile and the models enriched with subjective opinions we have defined an alternative representation that does not require the profile. We simply define the corresponding Opinions in UML, and define a set of opinions and query operation for each slot stereotyped as <<BeliefStatement>> or its subtypes.

With this, a new class, **RoomOpinions** is created to capture the opinions of the agents about the values of the attributes of class **Room**:

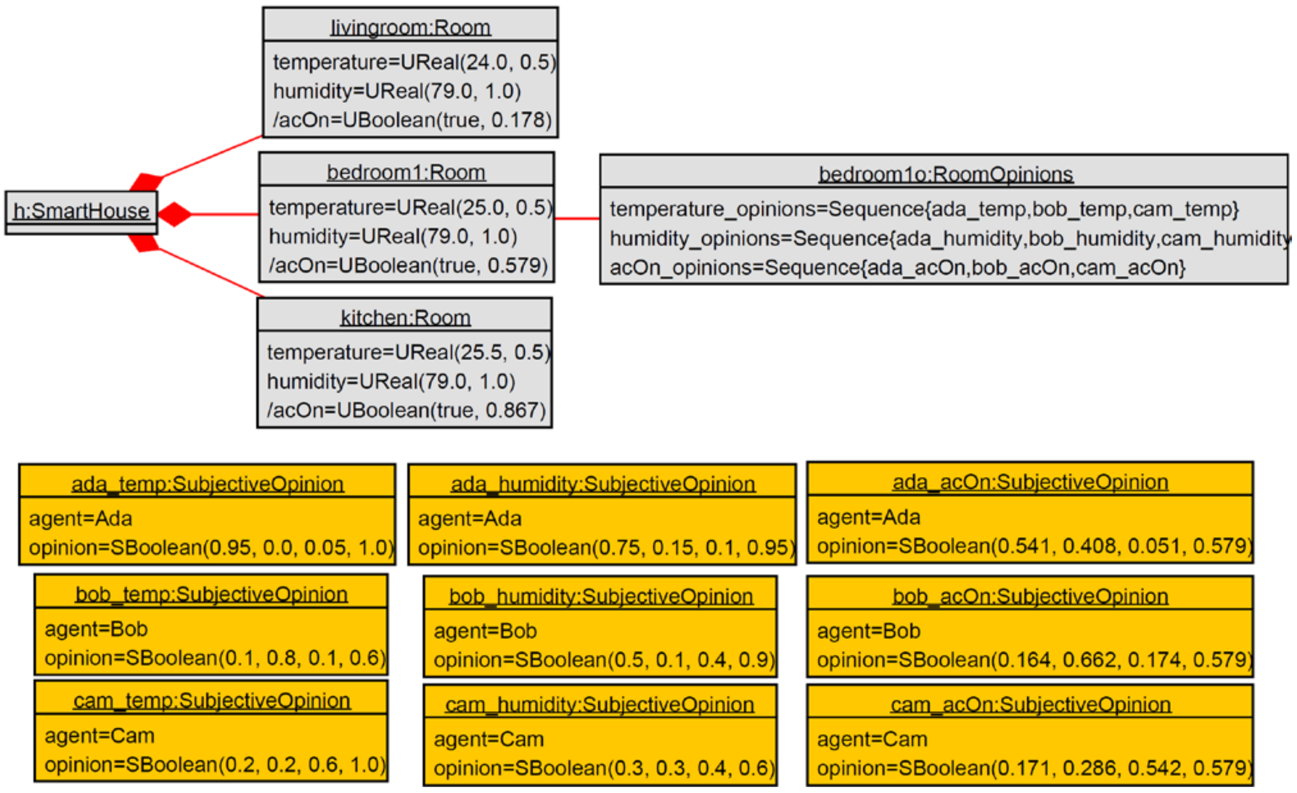
```
class RoomOpinions
  attributes
     temperature_opinions :  Sequence(SubjectiveOpinion) init: Sequence{}
     humidity_opinions :     Sequence(SubjectiveOpinion) init: Sequence{}
     acOn_opinions :         Sequence(SubjectiveOpinion) init: Sequence{}
```

```
 operations
    temperature_opinionOf (a : Agent) : SBoolean =
        let agentOpinion : Sequence(SubjectiveOpinion) = self.temperature_opinions->select(t|t.agent=a) in
        if agentOpinion=null or agentOpinion->isEmpty then SBoolean(1,0,0,1)
        else agentOpinion->collect(opinion)->last()
        endif
    humidity_opinionOf (a : Agent) : SBoolean =
        let agentOpinion : Sequence(SubjectiveOpinion) = self.humidity_opinions->select(t|t.agent=a) in
        if agentOpinion=null or agentOpinion->isEmpty then SBoolean(1,0,0,1)
        else agentOpinion->collect(opinion)->last()
        endif
    acOn_opinionOf (a : Agent) : SBoolean =
        self.temperature_opinionOf(a).applyOn(self.room.temperature >= 25.0)
         or self.humidity_opinionOf(a).applyOn(self.room.humidity >= 80.0)
end
```

The complete model is now the following:


SubjectiveOpinion
agent : Agent
opinion : SBoolean
SmartHouse
IsIn
1 house
* room
* house
InHouse
* agent
Agent
Room
temperature : UReal
humidity : UReal
/acOn : UBoolean
Opinions
1 room
0..1 roomOpinions
RoomOpinions
temperature_opinions : Sequence(SubjectiveOpinion)
humidity_opinions : Sequence(SubjectiveOpinion)
acOn_opinions : Sequence(SubjectiveOpinion)
temperature_opinionOf(a : Agent) : SBoolean
humidity_opinionOf(a : Agent) : SBoolean
acOn_opinionOf(a : Agent) : SBoolean

With this, the following object model shows the opinions of the three occupants about the sensors in room **bedroom1**, using the OCL operationalization instead of the UML Profile.


livingroom:Room
temperature=UReal(24.0, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.178)
h:SmartHouse
bedroom1:Room
temperature=UReal(25.0, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.579)
bedroom1o:RoomOpinions
temperature_opinions=Sequence{ada_temp,bob_temp,cam_temp}
humidity_opinions=Sequence{ada_humidity,bob_humidity,cam_humidity
acOn_opinions=Sequence{ada_acOn,bob_acOn,cam_acOn}
kitchen:Room
temperature=UReal(25.5, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.867)
ada_temp:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.95, 0.0, 0.05, 1.0)
ada_humidity:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.75, 0.15, 0.1, 0.95)
ada_acOn:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.541, 0.408, 0.051, 0.579)
bob_temp:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.1, 0.8, 0.1, 0.6)
bob_humidity:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.5, 0.1, 0.4, 0.9)
bob_acOn:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.164, 0.662, 0.174, 0.579)
cam_temp:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.2, 0.2, 0.6, 1.0)
cam_humidity:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.3, 0.3, 0.4, 0.6)
cam_acOn:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.171, 0.286, 0.542, 0.579)

Both representations are analogous, although the second one can be employed by the tool USE to seamlessly propagate the uncertainty associated to the opinions, since this tool implements the datatypes enriched with uncertainty as well as the Subjective logic datatype and operations [Muñoz et al, 2020]. In fact, it now contains the opinions of the three agents about the value of the **acOn** derived attribute, which have been computed using the operation **acOn_opinionOf(a:Agent)**.

```
use> ?bedroom1o.acOn_opinionOf(Ada)
-> SBoolean(0.540758, 0.4078, 0.051442, 0.579328) : SBoolean
use> ?bedroom1o.acOn_opinionOf(Bob)
-> SBoolean(0.16413, 0.662232, 0.173638, 0.579328) : SBoolean
use> ?bedroom1o.acOn_opinionOf(Cam)
-> SBoolean(0.171395, 0.286116, 0.542489, 0.579328) : SBoolean
```

Thus, Ada thinks that the AC should be turned on, while Bob and Cam think otherwise. But now we have more information about these beliefs, because we now know that Ada and Bob are quite sure of their opinions, but Cam is not (her degree of uncertainty is 0.54). Note how the three opinions have the same base rate, precisely the objective probability that was initially obtained using expression (1), which serves as prior probability for the agents' opinions.

The question now is what to do about the AC system. Is there a way for the three agents to come to an agreement? How to combine (i.e., "fuse" in Subjective logic terms) their opinions?

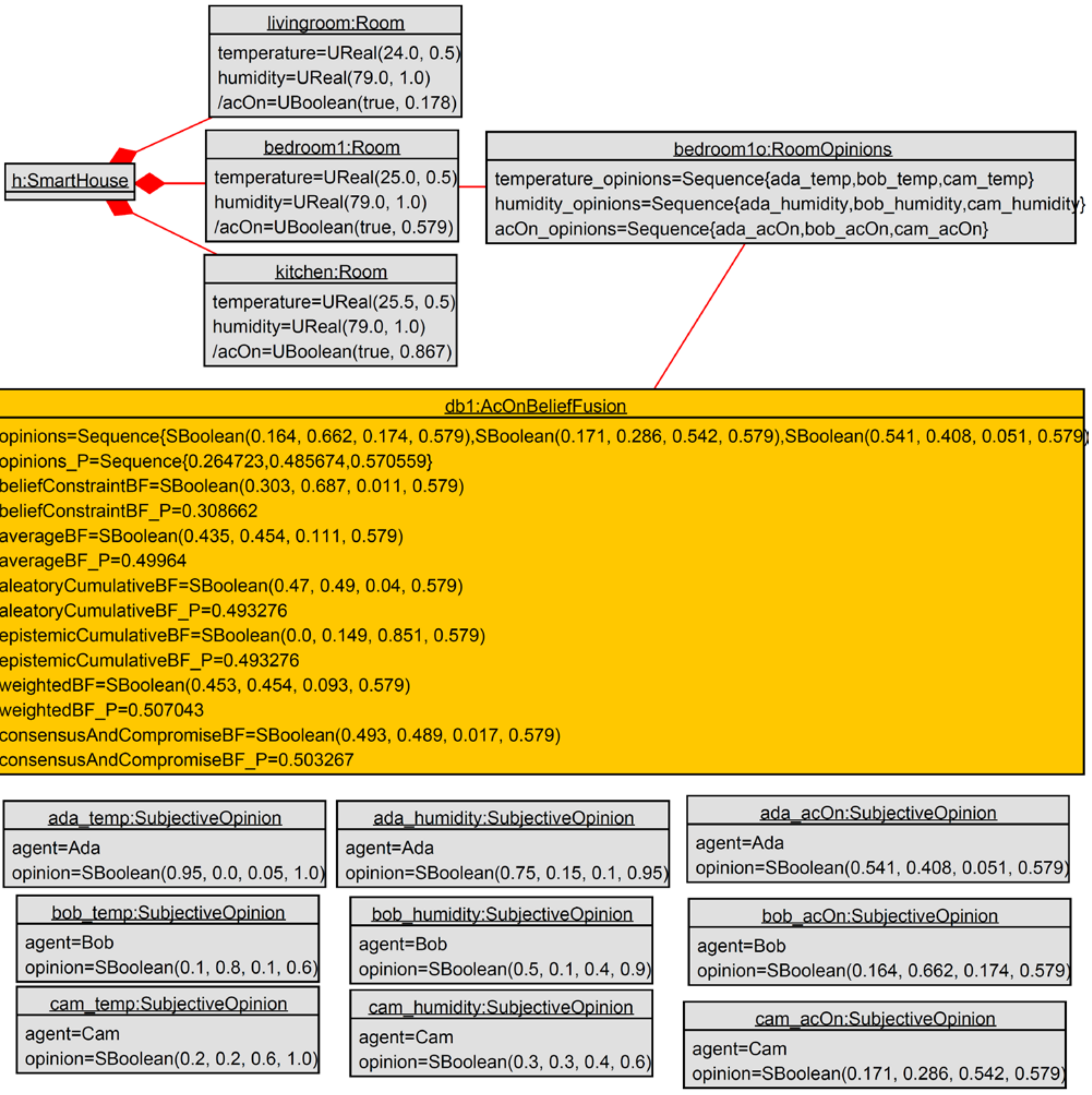
3. Fusing opinions

Our implementation of type SBoolean also supports the belief fusion operators described in [Josang, 2016]. They can be simply applied to sequences of opinions, returning a SBoolean value with the fused opinion. It is important that they return a SBoolean value because we not only want to know about the decision made, but also the degree of uncertainty associated with that uncertainty.

To compute the results of the different fusion operators, we define a new class that applies the corresponding operators to the opinions of the agents. For example, in the case of the **acOn** attribute, the following class **AcOnBeliefFusion** defines first the list of opinions that the occupants of the house hold about it, using operation **acOn_opinionOf()**, and then computes all fusion operators to that list. We also compute the projections of the opinions and of the results of the fusion for completeness (remember that the projection of a Subjective logic opinion coincides with the conversion of the opinion to probabilistic logic).

```
class AcOnBeliefFusion
attributes
    opinions : Sequence(SBoolean) derive = self.roomOpinions.room.house.agent->
        iterate(a:Agent; acc:Sequence(SBoolean)=Sequence{} |
            acc->append(self.roomOpinions.acOn_opinionOf(a)))
    opinions_P : Sequence(Real) derive = self.opinions->collect(o|o.projection())

    beliefConstraintBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
            let f:SBoolean = opinions->first() in
            let Q:Sequence(SBoolean) = opinions->excluding(f) in
            f.beliefConstraintFusion(Q)
        endif
    beliefConstraintBF_P:Real derive = beliefConstraintBF.projection()

    averageBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
            let f : SBoolean = opinions->first() in
            let Q : Sequence(SBoolean) = opinions->excluding(f) in
            f.averageBeliefFusion(Q)
        endif
    averageBF_P : Real derive = averageBF.projection()
    aleatoryCumulativeBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
              let f : SBoolean = opinions->first() in
              let Q : Sequence(SBoolean) = opinions->excluding(f) in
              f.aleatoryCumulativeBeliefFusion(Q)
            endif
    aleatoryCumulativeBF_P : Real derive = aleatoryCumulativeBF.projection()

    epistemicCumulativeBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
            let f : SBoolean = opinions->first() in
            let Q : Sequence(SBoolean) = opinions->excluding(f) in
            f.epistemicCumulativeBeliefFusion(Q)
        endif
    epistemicCumulativeBF_P : Real derive = epistemicCumulativeBF.projection()

    weightedBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
            let f : SBoolean = opinions->first() in
            let Q : Sequence(SBoolean) = opinions->excluding(f) in
            f.weightedBeliefFusion(Q)
        endif
    weightedBF_P : Real derive = weightedBF.projection()

    consensusAndCompromiseBF : SBoolean derive =
        if opinions->size=1 then opinions->first() else
            let f : SBoolean = opinions->first() in
            let Q : Sequence(SBoolean) = opinions->excluding(f) in
            f.consensusAndCompromiseFusion(Q)
        endif
    consensusAndCompromiseBF_P : Real derive = consensusAndCompromiseBF.projection()
end
```

The following object model shows the instance of that class when applied to the agents' opinions about the acOn derived attribute of room bedroom1.


livingroom:Room
temperature=UReal(24.0, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.178)

h:SmartHouse

bedroom1:Room
temperature=UReal(25.0, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.579)

bedroom1o:RoomOpinions
temperature_opinions=Sequence{ada_temp,bob_temp,cam_temp}
humidity_opinions=Sequence{ada_humidity,bob_humidity,cam_humidity}
acOn_opinions=Sequence{ada_acOn,bob_acOn,cam_acOn}

kitchen:Room
temperature=UReal(25.5, 0.5)
humidity=UReal(79.0, 1.0)
/acOn=UBoolean(true, 0.867)

db1:AcOnBeliefFusion
/opinions=Sequence{SBoolean(0.164, 0.662, 0.174, 0.579),SBoolean(0.171, 0.286, 0.542, 0.579),SBoolean(0.541, 0.408, 0.051, 0.579}
/opinions_P=Sequence{0.264723,0.485674,0.570559}
/beliefConstraintBF=SBoolean(0.303, 0.687, 0.011, 0.579)
/beliefConstraintBF_P=0.308662
/averageBF=SBoolean(0.435, 0.454, 0.111, 0.579)
/averageBF_P=0.49964
/aleatoryCumulativeBF=SBoolean(0.47, 0.49, 0.04, 0.579)
/aleatoryCumulativeBF_P=0.493276
/epistemicCumulativeBF=SBoolean(0.0, 0.149, 0.851, 0.579)
/epistemicCumulativeBF_P=0.493276
/weightedBF=SBoolean(0.453, 0.454, 0.093, 0.579)
/weightedBF_P=0.507043
/consensusAndCompromiseBF=SBoolean(0.493, 0.489, 0.017, 0.579)
/consensusAndCompromiseBF_P=0.503267

ada_temp:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.95, 0.0, 0.05, 1.0)

ada_humidity:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.75, 0.15, 0.1, 0.95)

ada_acOn:SubjectiveOpinion
agent=Ada
opinion=SBoolean(0.541, 0.408, 0.051, 0.579)

bob_temp:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.1, 0.8, 0.1, 0.6)

bob_humidity:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.5, 0.1, 0.4, 0.9)

bob_acOn:SubjectiveOpinion
agent=Bob
opinion=SBoolean(0.164, 0.662, 0.174, 0.579)

cam_temp:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.2, 0.2, 0.6, 1.0)

cam_humidity:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.3, 0.3, 0.4, 0.6)

cam_acOn:SubjectiveOpinion
agent=Cam
opinion=SBoolean(0.171, 0.286, 0.542, 0.579)

These results are summarized in Table 1 below:

Fusion operator	Fused opinion	Projection
Belief Constraint Fusion (BCF)	SBoolean(0.303, 0.687, 0.010, 0.579)	0.309
Consensus & Compromise Fusion (CCF)	SBoolean(0.494, 0.489, 0.017, 0.579)	0.503
Aleatory Cumulative Belief Fusion (ACBF)	SBoolean(0.470, 0.490, 0.040, 0.579)	0.493
Epistemic Cumulative Belief Fusion (ECBF)	SBoolean(0.000, 0.149, 0.852, 0.579)	0.493
Average Belief Fusion (ABF)	SBoolean(0.435, 0.454, 0.111, 0.579)	0.499
Weighted Belief Fusion (WBF)	SBoolean(0.453, 0.454, 0.093, 0.579)	0.507

Table 1. Results of applying the fusion operators on the opinions about the value of slot acOn of bedroom1.

Each row of Table 1 shows the belief fusion operator applied to these values, the resulting opinion, and its projected probability. Depending on the fusion operator used, the merged decision about whether the AC system of that room should be turned on or off changes. For illustration purposes, we have shown the result for each operator. Nevertheless, note that the fusion operator to apply cannot be chosen randomly but taking into account how the semantics of the different operators fit the particular situation.

When deciding what fusion operator to apply for the case of turning the AC on, our reasoning is as follows.

- The Belief Constraint Fusion (BCF) is used when the agents decide that there is no room for compromise and do not want to give in on their opinions, therefore, it cannot be applied here since there must be a final decision -- the AC must be on or off, but both things cannot happen at the same time.
- The Averaging Belief Fusion (ABF) operator assumes that the agents opinions are dependent. This is not the case here as their feeling about how warm the room is does not depend on any other fact.
- The Consensus & Compromise fusion (CCF) is suitable when agents are willing to give in on their opinions to reach agreements, which fits our situation.
- The Weighted Belief Fusion (WBF) operator also works as CCF but gives more weight to more certain opinions, unlike CCF where all opinions have the same weight even when some of them are very uncertain. In our case, it makes sense that uncertain opinions (i.e., people without strong preferences), should not have much weight. Therefore, this operation would fit even better than WBF in the AC situation.
- Regarding the cumulative fusion operators, they assume that the opinions are independent. The Aleatory one (ACBF) is suitable when giving opinions about a variable governed by a frequentist process, such as flipping a coin, which is not the case here. On the contrary, if we give an opinion about a non-frequentist variable, we will use the Epistemic Cumulative Belief Fusion (ECBF). This is an extension of the ACBF that produces the same opinion but with maximized uncertainty [Josang, 2016]. In this case, the epistemic version could be used because the agents' opinions are of this nature. However, we should take into account that when this operator finds contradictory opinions, it increases the uncertainty of the results. In fact, in the AC case, the resulting degree of uncertainty is so high (0.852) that the result is quite inconclusive and not very helpful to make a decision.

In total, four belief fusion operators are applicable in this case: BCF if the agents are stubborn and do not want to give in on their opinions; CCF if they are willing to give in to reach agreements; ABF if we want to average their opinions and each of them counts equally; or WBF if we want opinions with less certainty to count slightly less. The resulting decision will be to switch off the AC system of the room with the first and third operators, and switch it on with the other two.

References

[Josang, 2016] Audun Jøsang. 2016. "Subjective Logic - A Formalism for Reasoning Under Uncertainty." Springer. https://doi.org/10.1007/978-3-319-42337-1

[Bertoa et al, 2020] Manuel F. Bertoa, Loli Burgueño, Nathalie Moreno, and Antonio Vallecillo. "Incorporating measurement uncertainty into OCL/UML primitive datatypes." Software & System Modeling 19, 5 (2020), 1163–1189. https://doi.org/10.1007/s10270-019-00741-0

[Muñoz et al, 2020] Paula Muñoz, Loli Burgueño, Victor Ortiz, and Antonio Vallecillo. "Extending OCL with Subjective Logic." Journal of Object Technology 19, 3 (Oct. 2020), 3:1–15. https://doi.org/10.5381/jot.2020.19.3.a1

.