The vacation home of the Bowman family
(implementation sketch)

Mišo Antonič

In this work, I will try to outline a few possible solutions to some of the problems that have been proposed in the original scenario. The main focus will be on the logic reasoners and rules used in the individual ambient intelligence devices.

For the purpose of this work, let me define the word “device” as something that:

- has a unique identity
- is capable of logical reasoning using some set of rules
- may have state (such as set of formulae it knows to be true)
- may access the state of other devices

So, device may be a single machine (like a car), but can in fact also be represented by multiple machines behaving as a single device. Moreover, multiple devices can be simulated in a single machine, as long as they satisfy the provided conditions.

In context of our scenario, we will work with 4 devices:

- Family (a virtual device existing on some remote server)
- House
- Car
- Greenhouse

Now, lets look at the first part of the scenario and try to sketch its formalization. This italicized part is an exact quote from the scenario and may be skipped, as it is provided here only as a reference for our formalization attempt:

“... the House has recently found out that the family plans to stay for a weekend. It is a bit cold today, and so the House has to decide which rooms should be heated up. The Bowman’s car has just started to move and the navigation module signals that the vacation home has been set as the journey destination. The House comes to the conclusion that the vacation will really take place, just as the calendar has hinted. Then, it compares the global positions of the car and all the family members. As the oldest daughter is not in the car, the House deduces that she will probably not attend the vacation. It closes the door on her room and starts to heat up all the rest of the rooms in the House. The House then exposes all south-facing windows in order to use sun to facilitate heating.”

Let’s define the Family device further. Let Member(X) predicate be true iff the person X is member of the family. Let MemberPosition(P, X) predicate be true iff P is the exact global position of family member X. This information may be provided by GPS in each family member’s mobile phone. Let Visit(D, P) be true iff the family has scheduled visit at date D to position P in calendar.

Now, let’s look at the House device. Let CommonRoom(R) be true iff the room R does not belong to a single person, but is common for all members of the family. Let

RoomOf(X, R) be true iff room R belongs to family member X. Let “position” be House’s global position.

Finally, let’s define the Car device in a similar way. Let Journey(D, P) be true iff the Car is on its way to position P at date D. Let “position” be the Car’s global position.

We can now define a few logic rules for the Car:

CAR:

$$\frac{\text{Family.Member}(X) \wedge \text{Family.MemberPosition}(X, Y) \wedge \neg\ \text{Equal}(Y, \text{position})}{\neg\ \text{IsIn}(X)}$$

$$\frac{: \text{IsIn}(X)}{\text{IsIn}(X)}$$

These rules define the predicate IsIn(X), that is true iff a family member X is currently in the Car.

Note that the Family.Member(X) syntax is used to query other device (Family) for the value of its Member predicate. Car uses information from other device to infer new knowledge in this way. Now, the House device.

HOUSE:

$$\frac{\text{Family.Visit}(D, id) \wedge \text{Car.Journey}(D, id)}{\text{GroupVisit}(D)}$$

If the family has scheduled a visit and the car is on its way, it seems like a group visit of the house will really occur.

$$\frac{\text{Family.Member}(X) \wedge \text{GroupVisit}(D) : \text{Car.IsIn}(X)}{\text{WillVisit}(X)}$$

Lets find out which family members will really visit the house. We will presume that everybody that will visit is in the car.

$$\frac{\text{GroupVisit}(D) \wedge \text{CommonRoom}(R)}{\text{AdjustRoomTemperature}(R)}$$

We need to adjust temperature of all common rooms.

$$\frac{\text{GroupVisit}(D) \wedge \text{WillVisit}(X) \wedge \text{RoomOf}(X, R)}{\text{AdjustRoomTemperature}(R)}$$

We also need to adjust temperatures of the rooms belonging to each person that will attend the vacation.

Let predicate RoomTooHot(R) be true iff the temperature in the room R is too high. Similarly, let predicate RoomToCold(R) be true iff the temperature in room R is too low.

Then, the House device should periodically apply these rules during the whole vacation:

HOUSE:

$$\frac{\text{AdjustRoomTemperature(R)} \wedge \text{RoomTooCold(R)}}{\text{Heat! (R)}}$$

$$\frac{\text{AdjustRoomTemperature(R)} \wedge \text{RoomTooHot(R)}}{\text{Cool! (R)}}$$

It is worth noting that these two rules provide a negative feedback, thus stabilizing temperature in each room.

The exclamation point (!) is used to signify that when a formula becomes true (in this case, Heat!(R) or Cool!(R) formula), some side effect (heating or cooling of the room) will take place.

$$\frac{\text{Heat(R)} \wedge \text{SouthFacingWindows(R)}}{\text{UncoverWindows! (R)}} \qquad \frac{\text{Cool(R)} \wedge \text{SouthFacingWindows(R)}}{\text{CoverWindows! (R)}}$$

These two rules provide similar feedback loop, facilitating the sun and window covers to regulate the room temperature.

The rules we have provided for the House device so far should successfully recognize the incoming family and prepare itself by regulating temperature of the rooms. Additionally, these rules should provide stable temperature during the whole vacation by regulating the climatization and covering or uncovering windows when needed. Of course, this solution is vastly simplified and has many subtle or less subtle problems. For example, in its current form, the House may easily wake up its residents in night by covering and uncovering windows.

However, for this to work properly, the conclusions of some of the rules should be periodically deleted from the database and their rules should be periodically re-evaluated. AdjustRoomTemperature, WillVisit and GroupVisit conclusions should be deleted when the family leaves the house, and Heat, Cool, UncoverWindows and CoverWindows should be deleted and their rules re-evaluated every few minutes.

Scenario, the Greenhouse part, quote:

"It is time to check up the Greenhouse, which cares after Mr. Bowman's plant collection. Each plant has a sensor in its pot to monitor the soil moisture and the levels of all important nutrients. This way, the sensor provides valuable information for the automatic watering system. The Greenhouse has just found out that the pot sensor of a particularly precious specimen of japanese maple again signals dangerously low moisture. Based on its knowledge of this particular plant species, the Greenhouse reasons that this pot appears to dry out too often. As no other sensor in the Greenhouse exhibits abnormal behaviour, the Greenhouse comes to the conclusion that the maple's sensor must be broken. It provides this information to the House. The House informs Mr. Bowman and simultaneously plans a delivery of an entirely new sensor directly to the vacation home."

First, we define some built-in predicates for the Greenhouse device. Let Neighbour(X, Y) be true iff plant X is near plant Y. Let TooDry(X) be true iff the moisture sensor of plant X reports that soil is too dry. Let DryOrWetTooOften(X) be true iff some sensor reports data that does not seem to be at all plausible. Let ModelTooDry(X) be true iff a built-in *computer model* reports that the soil of plant X *probably should be* too dry by now.

Note that ModelTooDry predicate is a less precise version of the TooDry predicate, basing its estimates only on computer model, not the real, collected data. As you will see, the ModelTooDry predicate is used as a fallback when sensor(s) appear to be malfunctioning. Therefore it is used as some kind of emergency failsafe.

The whole watering system will then work like this:

- If the plant sensor appears to be OK, use its data to water the plant.
- If the plant sensor appears to be malfunctioning, use sensor of a nearby plant to water the plant with broken sensor.
- If there is a no neighbouring plant with functioning sensor, use a computer model to estimate the soil moisture to provide irrigation.

GREENHOUSE:

$$\frac{\text{DryOrWetTooOften}(X)}{\neg\, \text{SensorOK}(X) \wedge \text{HasBroken}(X)}$$

If some sensor does behave oddly, lets pronounce it broken.

$$\frac{\text{TooDry}(X) : \text{SensorOK}(X)}{\text{Water!}\ (X)}$$

Lets water a plant with dry soil as long as its sensor is OK.

$$\frac{\neg\ \text{SensorOK}(X) \wedge \text{Neighbour}(X, Y) \wedge \text{TooDry}(Y) : \text{SensorOK}(Y)}{\text{Water!}\ (X)}$$

If the plant's sensor is not OK, lets use its neighbour sensor (if we can) and if necessary, water the plant.

$$\frac{: \neg\ (\text{Neighbour}(X, Y) \wedge \text{SensorOK}(Y))}{*\ \text{NoNeighbour}(X)}$$

Lets assume that plant does not have a neighbour with working sensor, unless we can prove the opposite.

$$\frac{\text{NoNeighbour}(X) \wedge \text{ModelTooDry}(X) \wedge \neg\ \text{SensorOK}(X)}{\text{Water!}\ (X)}$$

If the plant's sensor is not OK and the plant has no neighbour with working sensor, lets use the computer model to estimate the soil moisture and eventually water the plant.

HOUSE:

$$\frac{\text{Greenhouse.HasBroken}(X) : \neg\ \text{Ordered}(X)}{\text{Ordered!}\ (X)}$$

The House orders broken Greenhouse parts, unless they already have been ordered.

As with the room temperature system, some Greenhouse conclusions should be periodically deleted and rules re-evaluated. The Water! conclusions should perhaps be deleted every minute and their rules re-evaluated. The SensorOK, HasBroken and Ordered! conclusions should be deleted and their rules re-evaluated when the broken sensor is replaced.

Scenario, the Light, Music & Lock part, quote:

"... Thanks to the sensors located inside each door, the House could not only decide whether to admit the visitors, but it could also automatically open the doors for the nearing family. It was dark already, so the light turned on as the family entered the hall. ... Mr. Bowman handclapped two times and said "music". As he walked to the Greenhouse, compositions of Johann Strauss accompanied him across the House."

Before we continue, we have to define a few more predicates for the House device. Let RoomPosition(R, X) be true iff global position of room R is X. Let LightWasTurnedOff(R) be true iff someone has turned off the light in the room R by pressing a button or pronouncing a voice command. Let TooDark(R) predicate be true iff the room R is too dark for humans to live in.

$$\frac{\text{RoomPosition}(R, X) \wedge \text{Family.MemberPosition}(P, Y) \wedge \text{Equal}(X, Y)}{\text{IsInRoom}(P, R)}$$

Lets find out if some family member is in the room.

$$\frac{: \neg\, \text{Occupied}(R)}{\neg\, \text{Occupied}(R)} \qquad \frac{\text{IsInRoom}(P, R)}{\text{Occupied}(R)}$$

Room is unoccupied unless there is someone in the room.

$$\frac{: \neg\, \text{Occupied}(\text{Entrance})}{\text{Lock! (Entrance)}} \qquad \frac{\text{Occupied}(\text{Entrance})}{\text{Unlock! (Entrance)}}$$

The entrance should be locked unless it is occupied by family member. Occupied entrance should be unlocked.

These rules should be periodically re-evaluated and their conclusions deleted often.

$$\frac{\text{Occupied}(R) \wedge \text{TooDark}(R) : \neg\, \text{LightWasTurnedOff}(R)}{\text{TurnOnLight! (R)}}$$

Turn on the light in the room, if it is too dark and the light was not previously explicitly turned off. This rule is also highly periodic.

The whole point of the LightWasTurnedOff predicate is essentially to allow the inhabitants of the House to sleep. People seem to prefer dark conditions for sleeping, but our system turns on the light in each room that is occupied by a family member. The LightWasTurnedOff predicate disallows the system to turn on the light if it has been explicitly turned off.

$$\frac{\neg\, \text{TooDark}(R) \wedge \text{LightWasTurnedOff}(R)}{\text{delete LightWasTurnedOff}(R)}$$

This is a pseudo rule. If it is no longed too dark in the room and the light was explicitly turned off, lets forget it was turned off. This basically reactivates the automatic light system.

$$\frac{\text{WantsMusic}(X) \wedge \text{IsInRoom}(R, X)}{\text{PlayFavouriteMusic}(R, X)}$$

If some family member wants to listen to music, the room she is in plays some of her's favourite tunes.

$$\frac{: \neg \text{PlayFavouriteMusic}(R, P)}{\text{DontPlayAnything}(R)}$$

By default, each room is silent, unless someone's favourite music is playing there. These music related rules are periodic..

Conclusion

We provided a sketch of possible solutions for some of the problems in our scenario. The solutions are quite naive and simple. In reality, there will be a lot of problems with them. For example, what happens if the family makes an unscheduled visit to the house? Or when two people listening to music come to the same room? And what about security issues? What if some family member looses his mobile phone?

It is possible that in order to solve these problems, we would have to take some of the powers from the house (like its exclusive power to lock and unlock the door). It seems to me that in reality, the integration of a system like this would have to be much gentler than the one outlined. Moreover, in some cases (like the automatic replacement order for the broken sensor), perhaps it would be better for the system to serve mainly as an advisor, leaving the final decision to humans.