A Smart Health Assistant via DALI Logical Agents

STEFANIA COSTANTINI, LORENZO DE LAURETIS, CLAUDIO FERRI, JESSICA GIANCOLA AND FABIO PERSIA

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

CILC 2021

Artificial Intelligence

- Articial Intelligence is a field of computer science aimed at understanding and building intelligent entities
- It is a truly universal field, as it can be applied in any intellectual context, from games such as chess to many areas of medicine.

Intelligent Agents

- In the last century, the first intelligent agents were born, but what is an agent? “An agent is anything that can be seen as a system that perceives its environment through sensors and acts on it through actuators".
- That is, an agent is composed of one or more sensors, which may be cameras, infrared sensors, ultrasonic sensors and so on, which provide a perception of the external world, and actuators, which may be, for example, motors, which allow the agent to perform actions.

DALI Language

- What can make agents ‘intelligent' is their software `core', which processes the input received and obtains the best response.
- Nowadays, there are many languages that can be used for programming Artificial Intelligence, including Python, C++ and Prolog.
- SICStus Prolog is the basis of the DALI agent-oriented language. DALI is in fact a logical language, that, similarly to Prolog, it is based on the logic programming paradigm.

Communication in DALI

- The DALI communication architecture implements the DALI/FIPA protocol, which consists of the main FIPA primitives, plus few new primitives which are peculiar to DALI.
- Primitives:
  - send message(External Event, Sender agent);
  - propose(Action, [Condition1,.., ConditionN], Sender agent);
  - accept proposal(Accepted Action, [Condition1,..., ConditionN], Sender agent);
  - reject proposal(Rejected action, [Reasonl,...., ReasonN], Agent Sender);
  - failure(Action failed, motivation(Reason), Sender agent);
  - cancel(Action to delete, Sending agent);

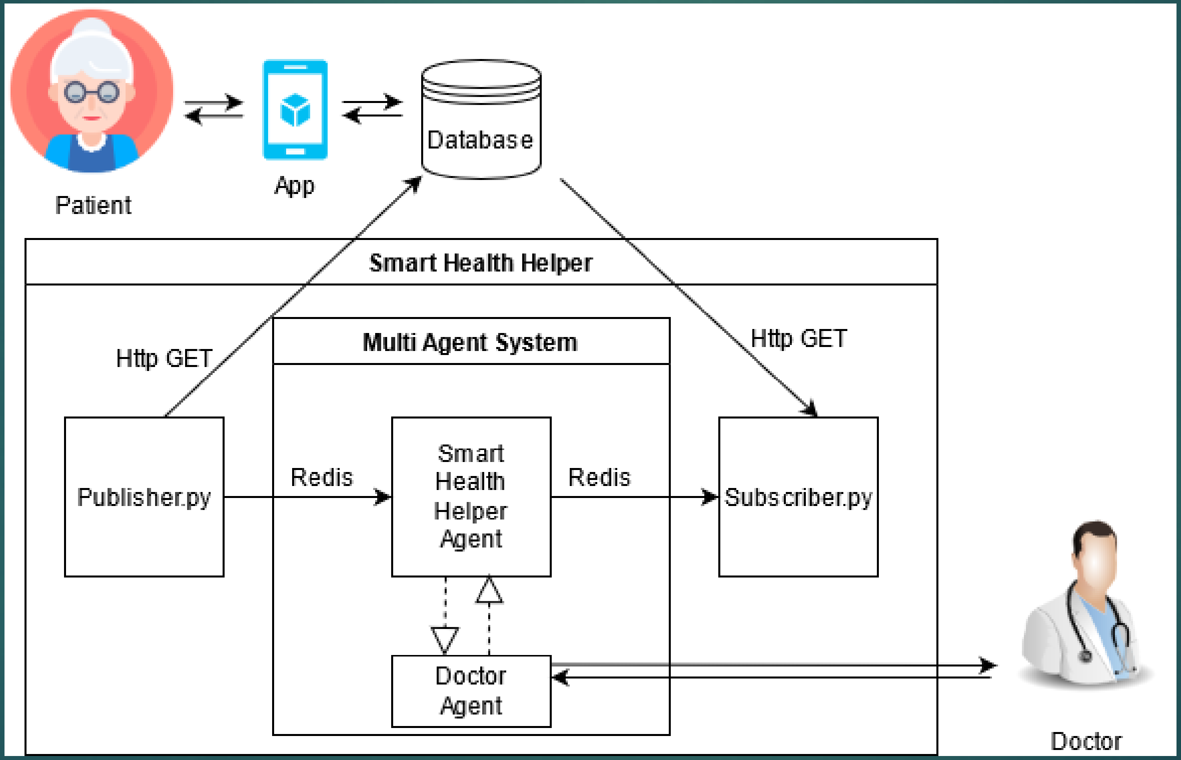
Smart Health Assistant Overview

- The goal of our project is to implement an intelligent system which, by analyzing data, it is able to recognize the patient's state of well-being or discomfort in both short and long term.
- Our system is therefore able to assess the seriousness of the situation and, if necessary, alert a **human doctor** or even call the **Emergency Service**.

Smart Health Assistant

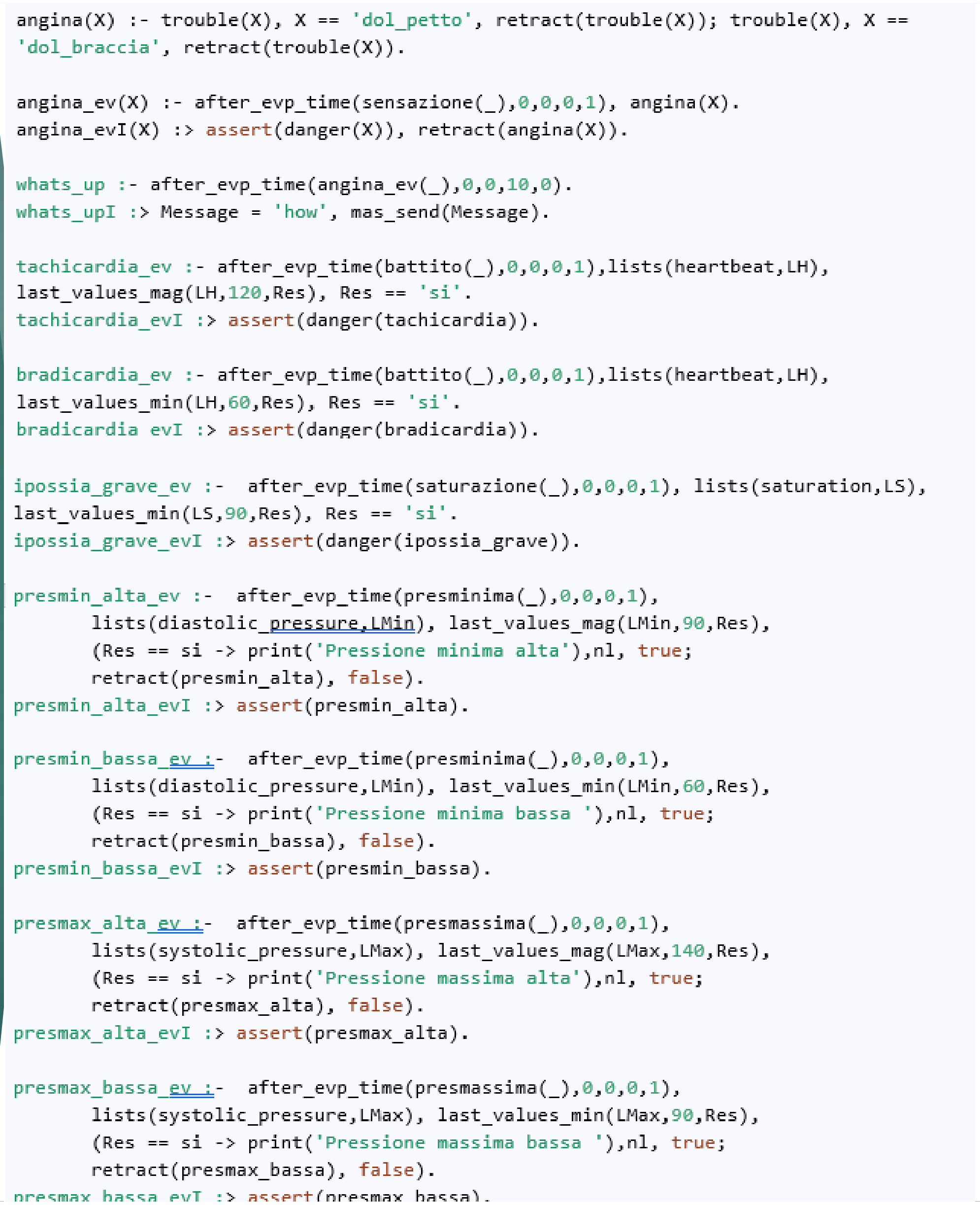
- The main agent of the system is the “SmartHealthHelper” agent, which can be seen as the patient's Personal Assistant agent, which helps the patient in the supervision of her health status.
- The system can also remind the patient not only to take the prescribed medications but also about other tasks to accomplish and aiding the patient in performing everyday activities.
- In the MAS there is also a “doctorAgent”, that is consulted if the smartHealthHelper agent detects situations that need to be evaluated. The “doctorAgent” is supposed to interact if necessary with a human doctor, so as to provide “SmartHealthHelper” with the doctor advice.
- There is also a Python module, used for communication with a Database.

System architecture



Code Sample

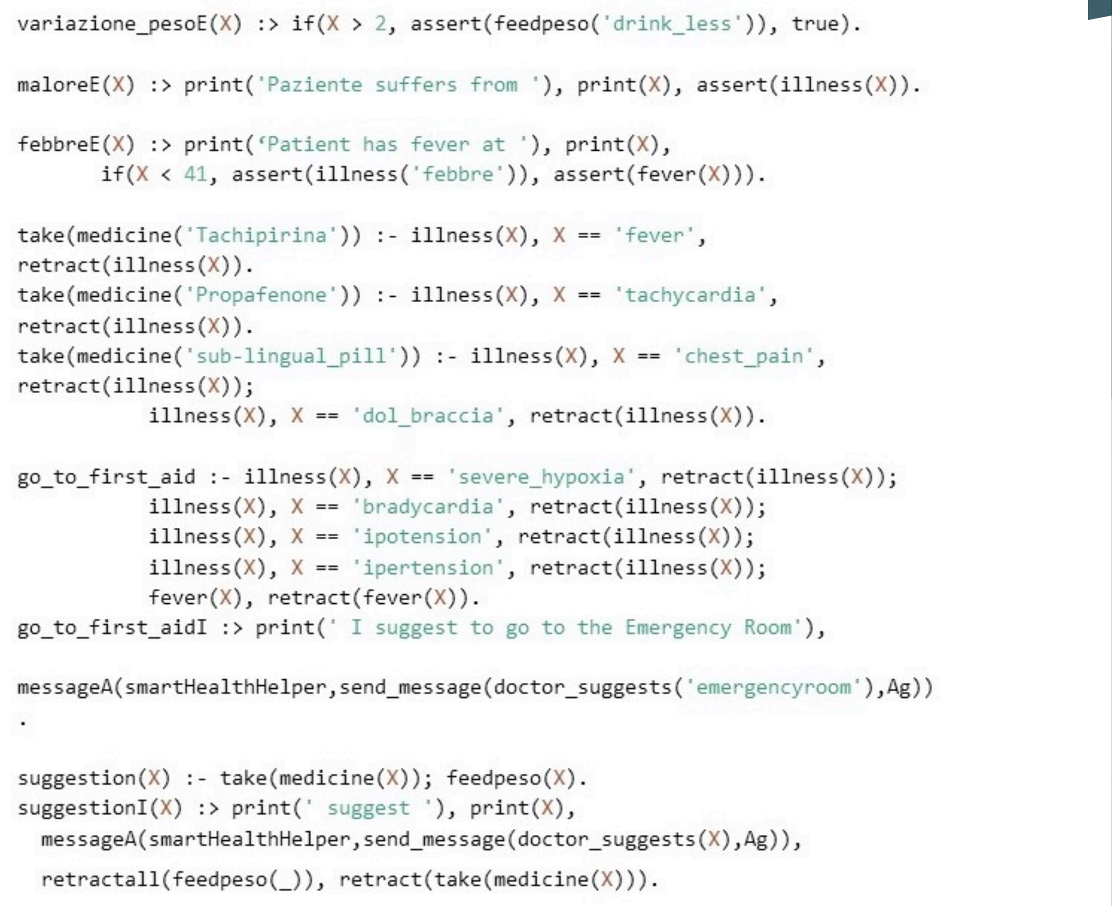
A sample of the code used to detect possible health problems



```
angina(X) :- trouble(X), X == 'dol_petto', retract(trouble(X)); trouble(X), X ==
'dol_braccia', retract(trouble(X)).

angina_ev(X) :- after_evp_time(sensazione(_),0,0,0,1), angina(X).
angina_evI(X) :> assert(danger(X)), retract(angina(X)).

whats_up :- after_evp_time(angina_ev(_),0,0,10,0).
whats_upI :> Message = 'how', mas_send(Message).

tachicardia_ev :- after_evp_time(battito(_),0,0,0,1),lists(heartbeat,LH),
last_values_mag(LH,120,Res), Res == 'si'.
tachicardia_evI :> assert(danger(tachicardia)).

bradicardia_ev :- after_evp_time(battito(_),0,0,0,1),lists(heartbeat,LH),
last_values_min(LH,60,Res), Res == 'si'.
bradicardia evI :> assert(danger(bradicardia)).

ipossia_grave_ev :-  after_evp_time(saturazione(_),0,0,0,1), lists(saturation,LS),
last_values_min(LS,90,Res), Res == 'si'.
ipossia_grave_evI :> assert(danger(ipossia_grave)).

presmin_alta_ev :-  after_evp_time(presminima(_),0,0,0,1),
       lists(diastolic_pressure,LMin), last_values_mag(LMin,90,Res),
       (Res == si -> print('Pressione minima alta'),nl, true;
       retract(presmin_alta), false).
presmin_alta_evI :> assert(presmin_alta).

presmin_bassa_ev :-  after_evp_time(presminima(_),0,0,0,1),
       lists(diastolic_pressure,LMin), last_values_min(LMin,60,Res),
       (Res == si -> print('Pressione minima bassa '),nl, true;
       retract(presmin_bassa), false).
presmin_bassa_evI :> assert(presmin_bassa).

presmax_alta_ev :-  after_evp_time(presmassima(_),0,0,0,1),
       lists(systolic_pressure,LMax), last_values_mag(LMax,140,Res),
       (Res == si -> print('Pressione massima alta'),nl, true;
       retract(presmax_alta), false).
presmax_alta_evI :> assert(presmax_alta).

presmax_bassa_ev :-  after_evp_time(presmassima(_),0,0,0,1),
       lists(systolic_pressure,LMax), last_values_min(LMax,90,Res),
       (Res == si -> print('Pressione massima bassa '),nl, true;
       retract(presmax_bassa), false).
presmax_bassa_evI :> assert(presmax_bassa).
```

DoctorAgent Code Sample

- A sample of the code used to detect possible health problems



```
variazione_pesoE(X) :> if(X > 2, assert(feedpeso('drink_less')), true).

maloreE(X) :> print('Paziente suffers from '), print(X), assert(illness(X)).

febbreE(X) :> print(‘Patient has fever at '), print(X),
       if(X < 41, assert(illness('febbre')), assert(fever(X))).

take(medicine('Tachipirina')) :- illness(X), X == 'fever',
retract(illness(X)).
take(medicine('Propafenone')) :- illness(X), X == 'tachycardia',
retract(illness(X)).
take(medicine('sub-lingual_pill')) :- illness(X), X == 'chest_pain',
retract(illness(X));
           illness(X), X == 'dol_braccia', retract(illness(X)).

go_to_first_aid :- illness(X), X == 'severe_hypoxia', retract(illness(X));
           illness(X), X == 'bradycardia', retract(illness(X));
           illness(X), X == 'ipotension', retract(illness(X));
           illness(X), X == 'ipertension', retract(illness(X));
           fever(X), retract(fever(X)).
go_to_first_aidI :> print(' I suggest to go to the Emergency Room'),

messageA(smartHealthHelper,send_message(doctor_suggests('emergencyroom'),Ag))
.

suggestion(X) :- take(medicine(X)); feedpeso(X).
suggestionI(X) :> print(' suggest '), print(X),
  messageA(smartHealthHelper,send_message(doctor_suggests(X),Ag)),
  retractall(feedpeso(_)), retract(take(medicine(X))).
```

System Tasks: Python module

- Collects patient data from the database and sends it to the MAS, and then processes the feedback received from the MAS and sends it to the database

System Tasks: SmartHealthHelper

- Analyses the patient's vital parameters (heart rate, saturation, minimum and maximum pressure, temperature, weight) taken by wearable devices or by the patient herself, returns immediate feedback on each parameter and stores them in the corresponding list.
- Every time new parameters are entered, it compares the last three values stored in the list, and detects a dangerous situation (such as tachycardia/bradycardia in case of heart rate, hypoxia in case of saturation, and hypertension/hypotension in case of pressure), if all the three values are below or exceed a given threshold. The number of last measurements to be investigated (i.e., three, in this preliminary case study) is suggested by medical doctors. In the next versions of the system, we will apply more sophisticated techniques based on temporal windows, which will specifically depend on the analyzed parameter.
- Analyzes the symptoms that the patient reports.
- In case of a dangerous situation, it contacts the doctorAgent which will provide feedback on what to do.
- For each therapy entered, i.e., a medicine to be taken every day, it sends a reminder to the user at the right time;
- Periodically, it sends a reminder for temperature, blood pressure and weight measurements.
- With every new entry of a weight value, it compares it with the previous one and returns a feedback indicating if the user has lost or taken weight, how many kilos, and also communicates it to the doctor agent.

System Tasks: doctorAgent

- Receives messages from the smartHealthHelper agent, analyzes the communicated problem, e.g., fever or tachycardia, and responds by suggesting a medication to take or, in case of serious danger, it suggests to go to the Emergency Room. It also provides feedback on when to carry out the next measurement. Such feedback can partly be provided automatically, and partly by asking a human doctor.

Conclusion and future work

- DALI language, thanks to its evolution over the years, offers great possibilities of managing complex contexts with a lot of occurring events.
- The Smart Health Assistant project, built on DALI, offers high-level logical reasoning capabilities, using very low resources, being able to run even on very low-specs machines.
- It receives a lot of raw data from the patient (such as heart rate, blood pressure, temperature, and so on..), and it is able to process such data very quickly, allowing the system to do real-time monitoring on the patient.
- If critical situation is detected, the doctor (or even the first aid services) is alerted and the user receives notifications on what he should do to prevent injuries and stay healthy.
- In the future, we would like to implement forms of complex reasoning concerning complicated medical conditions (i.e. Pneumonia, Heart Attack, and so on…).
- Also, we intend to provide our application with access to medical information systems (e.g., patient databases, medical archives).

THANKS FOR YOUR ATTENTION ☺