2009/10

FIB UPC

Nick Veenhof and
Tomas Barton

[HOUSING REPORT]

This document is a report from Artificial Intelligence course at FIB UPC, it describes development of ontology and consequence development of knowledge based system in CLIPS with usage of ontology.

Contents

Contents 2
1. Introduction 4
2. Identification of the problem 4
   Analysis 7
   Expected Result 8
3. Conceptualization 8
   Concepts of the domain 8
   Problem and sub problem participation 9
4. Formalization 9
   Ontology 9
   Offers 10
   Services 14
   Reuse of ontology 15
5. Implementation - CLIPS Knowledge based Programming 16
   About CLIPS 16
   Strategy 18
   Added Classes/templates 18
   Messages 19
   Functions 20
   Module "initiation" 21
   Module "personal-questions" 22
   Module "house-queries" 24
   Module "output" 26
   Module "EOP" - end of program 26
   Exceptions and Trivialities 27
   Fast-forward decisions 28
   Our realties 28
   Test cases 29
6. Testing 36
7. Conclusion 38

Future work 39
8. Bibliography 39

1. Introduction

In this lab we will try to develop an expert system with usage of knowledge based engineering seen in theory classes. Firstly, we analyse the problem to see the requirements and then continue with iterative development of a complete ontology for this domain. Secondly we write a program in CLIPS for getting information from user and processing retrieved information with usage of our ontology. Finally we will try to test our system and evaluate gained results.

2. Identification of the problem

Create a system which would recommend a flat or house to people who are looking for one. There are many various criteria for choosing, starting from clear one to not very typical which people usually do not take in consideration because they would not be able to find an ideal realty. Our system should also include in results offers which fail in some criteria but the others are fulfilled.

Due to that the housing market is low; the public office for house renting of the Department of Housing wants to facilitate the process of renting a house to the citizens, so they can find the house that fits their needs the best.

This office has information about house renting all over the country. To simplify our problem we will assume that our system has only to recommend housing rentals for the fictitious city which is in our case very similar to Barcelona.

Each offer of housing rental includes the most relevant characteristics of the house:

- Rental Title
- Monthly rent in Euro
- Date of availability
- Habitable space in m^2
- Number of bedrooms and if they are single or double bedrooms.
- If the room has 1 or multiple windows.
- Specifications per room such as space and type (kitchen, normal, bathroom, ...)
- Any special equipment for a room or a house should be defined (like balcony or washing machine)
- Kind of housing: flat, duplex apartment, detached/semidetached house. In the case of at or duplex, the floor (ground level, first floor, second floor, ..., attic)
- Other characteristics as: Pets do not allowed, heating or air conditioning, if it is oriented to the sun (in the mornings, in the evenings, all day), access to common swimming pool, parking space included, with excellent views, ...

Additionally to these characteristics, the system has to use the knowledge about the city to deduce other characteristics that could be important to the clients.
if there are near services of special interest, such as:

- schools, universities, libraries
- cinemas, clubs, theatres, bars, pubs, restaurants
- green areas, hospitals, doctors
- markets, shop, supermarket/hypermarkets
- bus stops, metros, trains, trams

To be near to some of these services/areas could be annoying for some clients or desirable for others, such as nightlife areas, stadiums, etc.
To determine this kind of information we have a list of all the services/areas of the city and their coordinates[1]. We will also have the coordinates of each renting offer. The coordinates of the city are measured from an origin and they tell the relative position in two dimensions of any place in the city to that origin measured in meters. We can determine if a specific place is near, mid distance or far to another depending on if it is less than 500 meters, less than 1000 meters or more that 1000 meters respectively.

People looking to rent a house have constraints and/or preferences about the kind of space they want to rent. We also know their personal characteristics:

- The name of the person for personalization of the program
- Age of the client looking to rent (18 to 100)
- Maximum rent they want to pay or they would pay more if the place is their dream house (Hard constraint)
- Minimum rent that for the client indicates that the offer is too cheap to be true (the place is probably a dump).
- Constraints about the number/sizes of the bedrooms (double/single) according to the children in the family. Every room can inhabit 2 children + 1 extra room as playground is needed.
- If the client owns a car and/or goes to work with it
- If he brings the children to school with the car.
- Specific constraints or preferences about the distance to an specifi c service (near to schools, near to public transportation, ...). For example, bringing the children to school would exclude the fact that we need close public transport
- It the client prefers public transportation to move inside the city.
- Typology of the people looking to rent:

[1] see Conceptualization for this list

  - Alone
  - Client and partner
  - Client and partner that will have a baby in the near future
  - family with children
  - Family with special needs
  - With Friends or special cases like :
    - group of students (number)
    - just one person
- Would you like space just for yourself or someone else? [just me, partner, children, friends] From this question we can evaluate number of rooms
- If the client owns a car.
- What type of real estate you are looking for? This question can make significant difference. We can not put into results an office when clients want a house. So this condition is a hard constraint.
  - House
  - Flat
  - Room
  - Office

These characteristics determine, depending on the profile of the tenant-to-be, that some services are more important than others. So we define 5 types of areas with their own properties.

1. **Quiet**
   For people that really don't like noise outside it's important that the noise factor of the neighbourhood is not high.
2. **Centric**
   Clients that like to live in a centric area would prefer supermarkets, public transport, bars close by. They hardly mind the noise
3. **Young**
   To young people could be more important to have nightlife areas near.
4. **Residential**
   Clients that like to live residential would like to be in a more quiet neighbourhood but supermarkets close and depending on the needs also public transport
5. **Quiet but connected/outskirts**
   This location could be suitable for elderly people or elder members of the family. If there is a connection by metro or train it could be considered as a good connected.
   - Green areas should be close and noise should near 0.
   - Healthcare should be close
   - supermarket should be close

Other characteristics depend from the users profile like:

- If there are children, it could be interesting to have near or at mid distance schools.
- If they bring the children to school with the car or not for having public transport close
- To families and couples could be more important to have a hypermarket near in order to do their monthly shopping.

There are also other common sense characteristics that make a place better than other, depending on the user's choice:

- being a sunny place
- having an elevator
- better an attic than a ground level
- having as little noise as possible but as centric as possible.
- areas with more services around are better (the area has more life)
- that a detached/semi-detached house is better for larger groups of people
- that is good to have green areas not far
- areas that have no services that are considered annoying around them

A solution will be a list of the more adequate offers for a client, giving also a recommendation level:

- **Partially adequate**: Only one or two criteria fail, but it could be acceptable depending on client consideration (rent is a little bit more that the specified maximum, public transportation is at mid distance, there are not green areas at near distance, ...).
- **Adequate**: All requirements are full filled by the offer.
- **Very recommendable**: All requirements are fulfilled and it has extra characteristics that make this offer better than the rest (lower rent, more rooms, and more services near than the explicitly specified.

Analysis

A **knowledge based system** is software that attempts to provide an answer to a problem, or clarify uncertainties where normally one or more human experts would need to be consulted.

Since we start with a big amount of facts that we, by means of questioning and reasoning, gather from the client it is feasible that we can work with these fact to thin our results.
Or in the words of Wikipedia: "the creation of a so-called "knowledgebase" which uses some knowledge representation formalism to capture the **Subject Matter Expert's** (SME) knowledge."

An other analysis from Wikipedia says "Expert systems may or may not have learning components but a third common element is that once the system is developed it is proven by being placed in the same real world problem solving situation as the human SME, typically as an aid to human workers or a supplement to some information system." If we understand this correctly, this is also the goal of the program. To easily aid people in the search for their house as if they were telling a person their personal data and this person analyses the information and searches through a list of houses to show them a list of recommended houses.

Productions systems are composed of a **fact base**, **knowledge base** (or rule base) and **inference engine**. In our domain is pretty difficult to make decisions straight on one fact, for example when we have a student, we can suppose that we he would like to live close to bars. But there are also other students who like quiet location. So we need some extra information to make this presumption.

Our axioms would be about conversion of human statements to numeric representation. For example some objects could in distance:

- close – in our units less than 2 (< 500m)
- mid – 4 units (between 500 and 1000m)
- far – more than 6 (more than 1000m)

Distances should be also different if person owns a car.

Rest of facts in fact base will be filled with input from user, combination of facts will trigger execution of rules. Our goal is to make rules for all combinations of facts from which we can deduce another fact or filter some instancies.

Our knowledge should contain information about placement of all shops, pub, cinemas, stations and other points of interest in town. This database should be up-to-date to make our decisions precise.

Expected Result

We want to have an easy to use system that returns us a list of houses that matches or at least has close matches to our client's expectations.

As seen in our problem description we want to include all those questions and be able to let the user think the system understands the user's wishes.

3. Conceptualization

Concepts of the domain

To start our development we need to have certain concepts clear:

We need:

- Houses of different types
- Services of different types
- Offer that includes the price and a 'type-of-house'

We worked with these basis concepts to further extend our problem resolution.

Problems and sub problems involved in the resolution of the problem.

- A House can include several flats and a flat can include several rooms and each one of them can be added to an offer. We have to keep this in mind when we are developing the ontology.
- Also a service and a 'type-of-house' have to be situated geographically. To make this possible we can create another concept Address which describes our location with parameters such as country, province, city or even area inside a city.
- A textual address representation is not sufficient in order to know where it is situated. That is why we will need to add a coordinate class inside an address. With this field we will be able to make distance calculations.

Problem and sub problem participation

These problems described previously were found gradually when solving our project. It took several tries in order to get the (in our opinion) most feasible ontology.

4. Formalization

Our KBS should have a representation of a domain in which we are interested in, so that the system would be able to solve given problem.
A more detailed view of the ontology is included as a separate website (HTML export of protégé).

Ontology

During development of ontology we followed Ontology 101 (McGuinness) guide. We divided work to an ontology part and programming part which we done in parallel, because using created ontology in program seemed for us as a best way how to verify our work. Here we concentrate more on description of the ontology part, the programming one comes in next chapter.

Firstly we answered following questions about purpose of ontology.

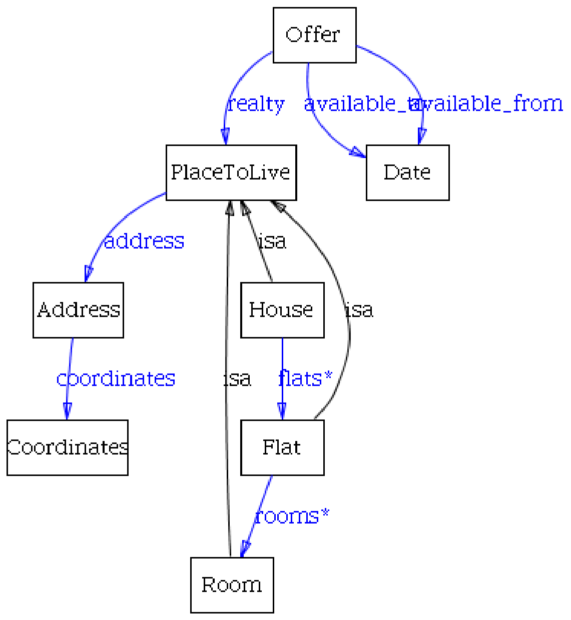
1. **What is the domain that ontology will cover?**
   Our ontology will cover domain of renting houses, flats and also domain of public services in a city. Also a brief representation of a person should be included

2. **For what we are going to use the ontology?**
   The ontology should help with searching for an ideal place to live, it should know all services in town and their location and be able to generalize the concepts from these houses and services to easily compare them later on. All the facts from the services and houses should be included in the ontology.
3. **For what types of questions the information in the ontology should provide answers?**
   Whether specific house is close or far from service, which services are noisy, the price of offers and which type of house/flat/room it contains, .. for a better understanding please read
4. **Who will use and maintain the ontology?**
   In first instance we (our team) are the only ones that will use the ontology. Since this is an assignment the question who will maintain the ontology is not relevant.

Offers

First part of our ontology is focused on estates which are usually rented.

We started with **top-down** approach; our basic concept is a PlaceToLive, which represents any place that can be use for living, so we have a house, a flat and a room. These elements are placed in hierarchy, a house can have more flats and a flat can have more rooms.



Offer – an offer is an advertisement for a realty - a house, flat, room etc. It usually has some catchy title, price, date from which is available and some address.
Date – different nations use various format of writing a date, for our system is important to know which part of date represents day, month, year, eventually also hour, minute and second.

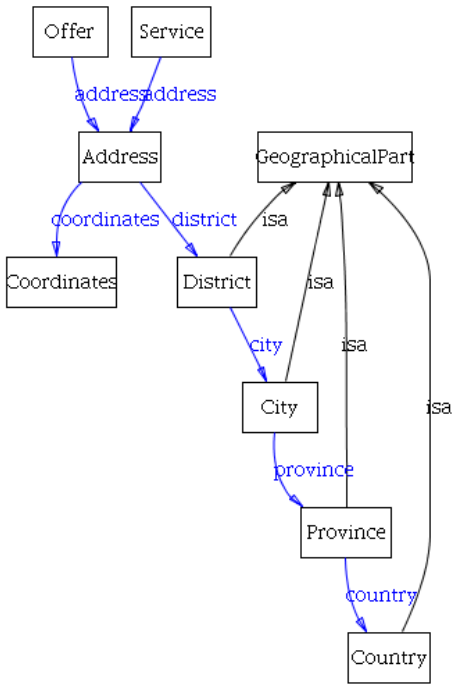
Inside an offer we can find the realty slot that contains a PlaceToLive. A PlaceToLive can be a room/flat/house. Special types of houses like terraced house, semi-detached are sub types of our

concept of house we decided to put type of house as a slot because there are no special properties or relationship to another classes.
House - there are many types of houses in different countries but generally we can say that each house contains at least one flat or apartment.
Flat - consists usually of more rooms, for us are important habitable rooms. A **Room** can be single or double, it can be kitchen, living room or sometimes all of them together.

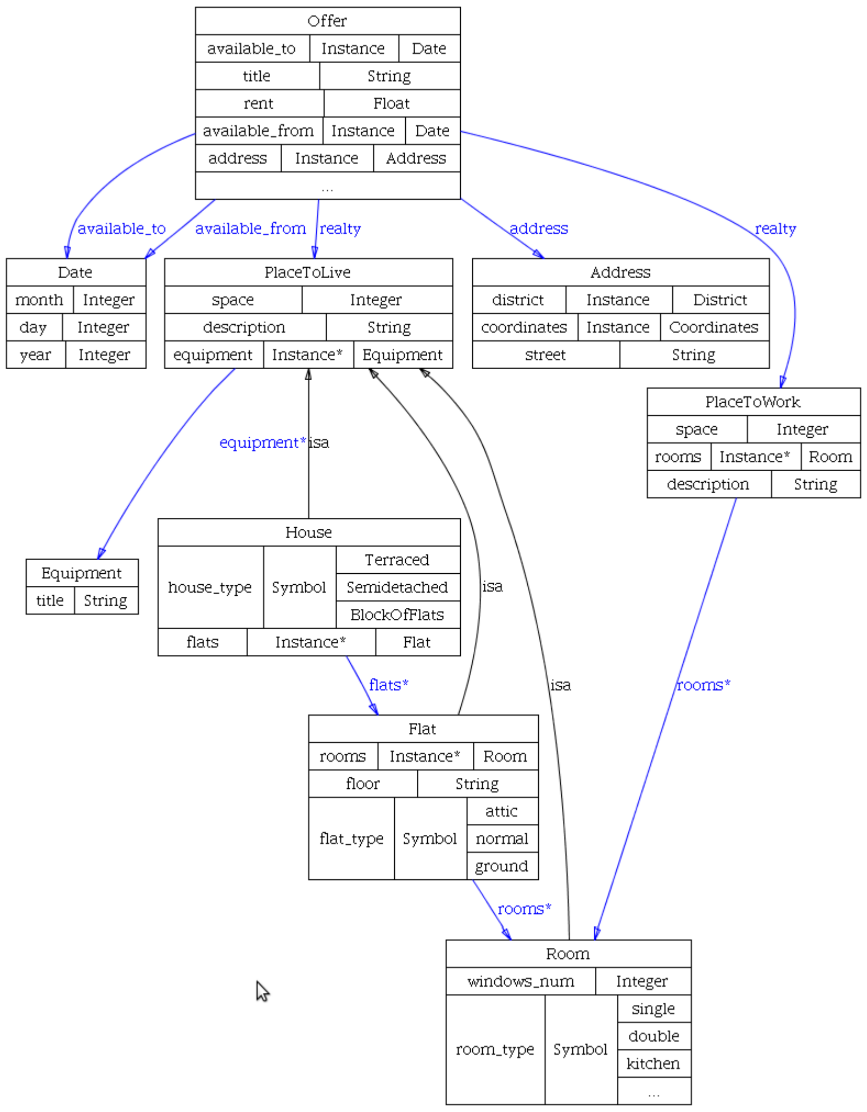
Image 3.1 : Simple Ontoviz of the offer class to get a clear overview

We discussed this concept quite a lot and switched to a **bottom-up** approach. A Province is a part of **Country** it might be a subclass of it. It is a bit tricky, because a province could be "a kind of" country but we can not say that a **City** is "kind of" country. Therefore we have to put the **District** on the same level as the **GeographicalPart**.

In concept of **Address** is enough to specify just **District** (or create a new one) and other fields like city, province and country are same for every address in district, this is guaranteed by the relation ships we created. **Coordinates** are our own implementation of a coordinate system. Location [0;0] is the centre of town, this is just for making orientation on map easier for us, it is possible to put there any other coordinate system.



For our purposes is this concept enough but if would like to answer questions like “I want to live close to a sea”, we would need not just points but also representation of area from which is counted distance. This also rises problem with choosing best two points between is distance counted. There are quite many examples of this problem in real world. We might need to know how noisy are roads which are going through town or distance from a huge park can be significantly different if we had taken in consideration just centre of park. But these problems are really beyond our task.



This concept went through some testing and we did not encounter any problems, so we continued with next iteration. Here is a more detailed diagram of our offers, enriched with other relationships and slots.

Each *PlaceToLive* can have some **Equipment**, this could be a TV, microwave, heating or whatever is considered as important for people who are looking for a house. We did not seen any meaning of dividing equipment into more categories because usually is important whether this equipment is available or not.

By all means we can determine whether internet is an ADSL and what speed, what is the size of TV and how many channels does it have. But in this phase we did not consider this parameters as crucial upon which would stand decision of renting a realty, therefore we left this for next iteration.

Image 3.2 : More detailed overview with the slots of each class

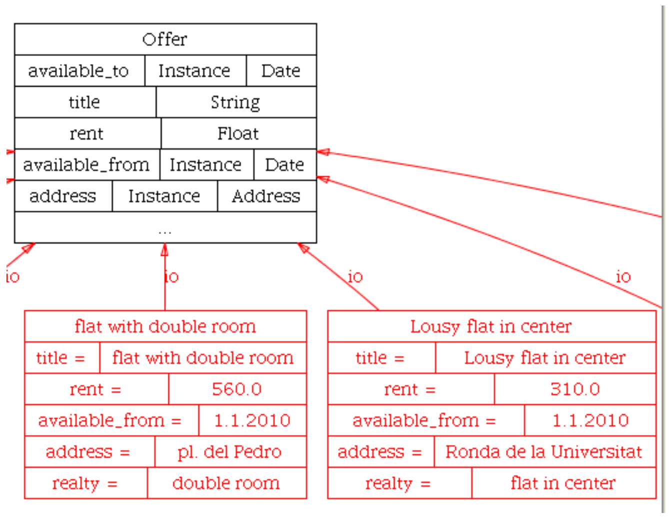
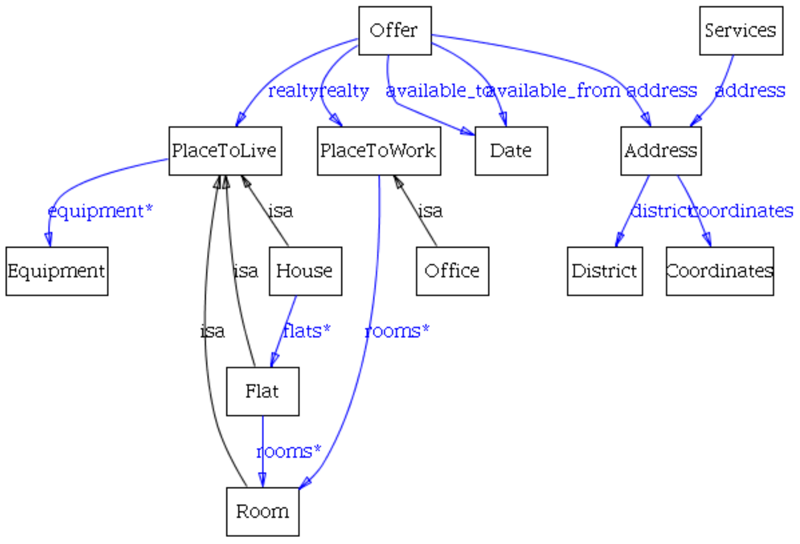

Offer
available_to | Instance | Date
title | String
rent | Float
available_from | Instance | Date
address | Instance | Address
...
io
io
io
io
flat with double room
title = | flat with double room
rent = | 560.0
available_from = | 1.1.2010
address = | pl. del Pedro
realty = | double room
Lousy flat in center
title = | Lousy flat in center
rent = | 310.0
available_from = | 1.1.2010
address = | Ronda de la Universitat
realty = | flat in center

Image 3.3 : Examples of instances of an offer

During development we had to make a few changes because we realized that the structure is not suitable for representation of the domain. Main concept stays still the same, but it is open for extension.



Services

Another part of ontology is services which are independent on the offers. In services are included supermarkets, restaurants, pub and so on. These services could be close to offers but they are not a parameter on an offer.

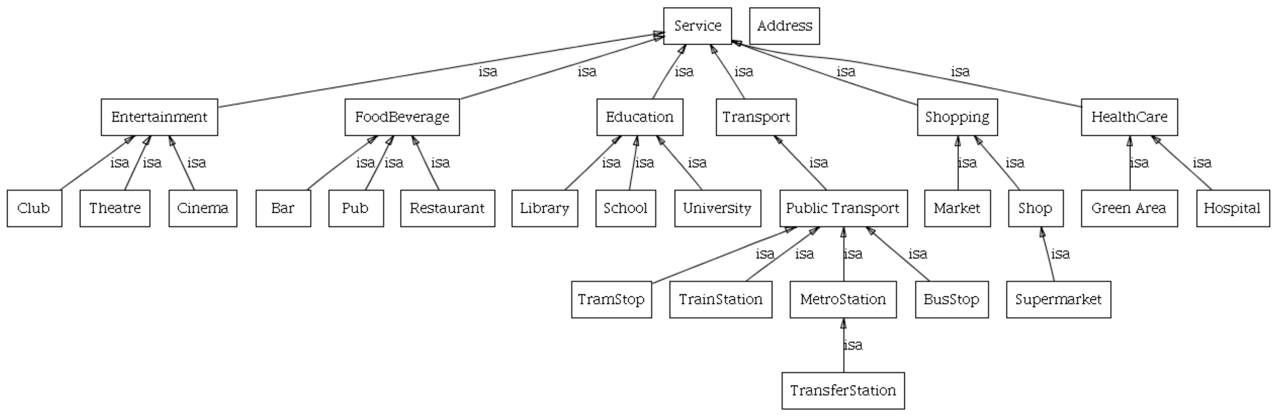


Image 3.4 : Super and subclasses of services

Every instance has certain information slots as seen in the Protégé program if the ontology is opened. An important decision that we made is to add a noise factor to every service that was added to compute later on if a person will live in a quiet or noisy neighbourhood.

We tried to group similar services into group and gather information about their characteristics. In **Transport** class we have metro station which could be important for many people because metro is fast mean of transport in big cities. Each **MetroStation** can have one line except of **TransferStation** where are at least two lines. Which line is at metro station can be important for many people because transfers can significantly prolong journey by metro. In sub-class we can change multiplicity of slot, which is definitely advantage of this language.

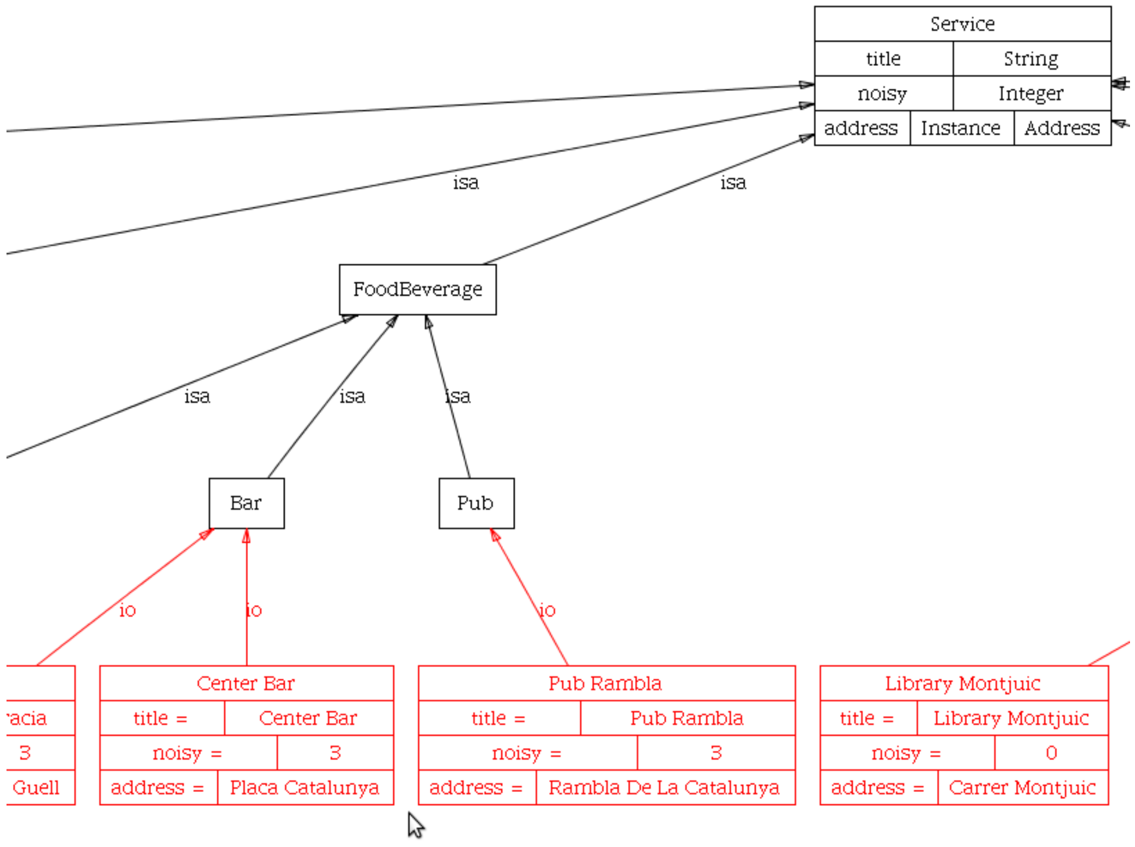


Image 3.5 : Instance examples of services

Reuse of ontology

One of basic characteristic of ontology is reuse of this knowledge. We tried to develop an ontology with universal patterns, so there are also more general concepts which we did not use at all. A good thing about the reusability of slots is that we now can create instances that have the same coordinates/address slot and also that we are able to write applications on top of this ontology that can benefit from these re-used slots.

To show an example, we created various houses and services inside our ontology. Each one is different from another but to get a clear overview on how this was done in space and time we created a graphical map. Every house or service was added with corresponding coordinates which is a reused slot in realty's and services. As explained before, this is important to know and to see to evaluate the outcome of our program and verify our noise factor system worked.

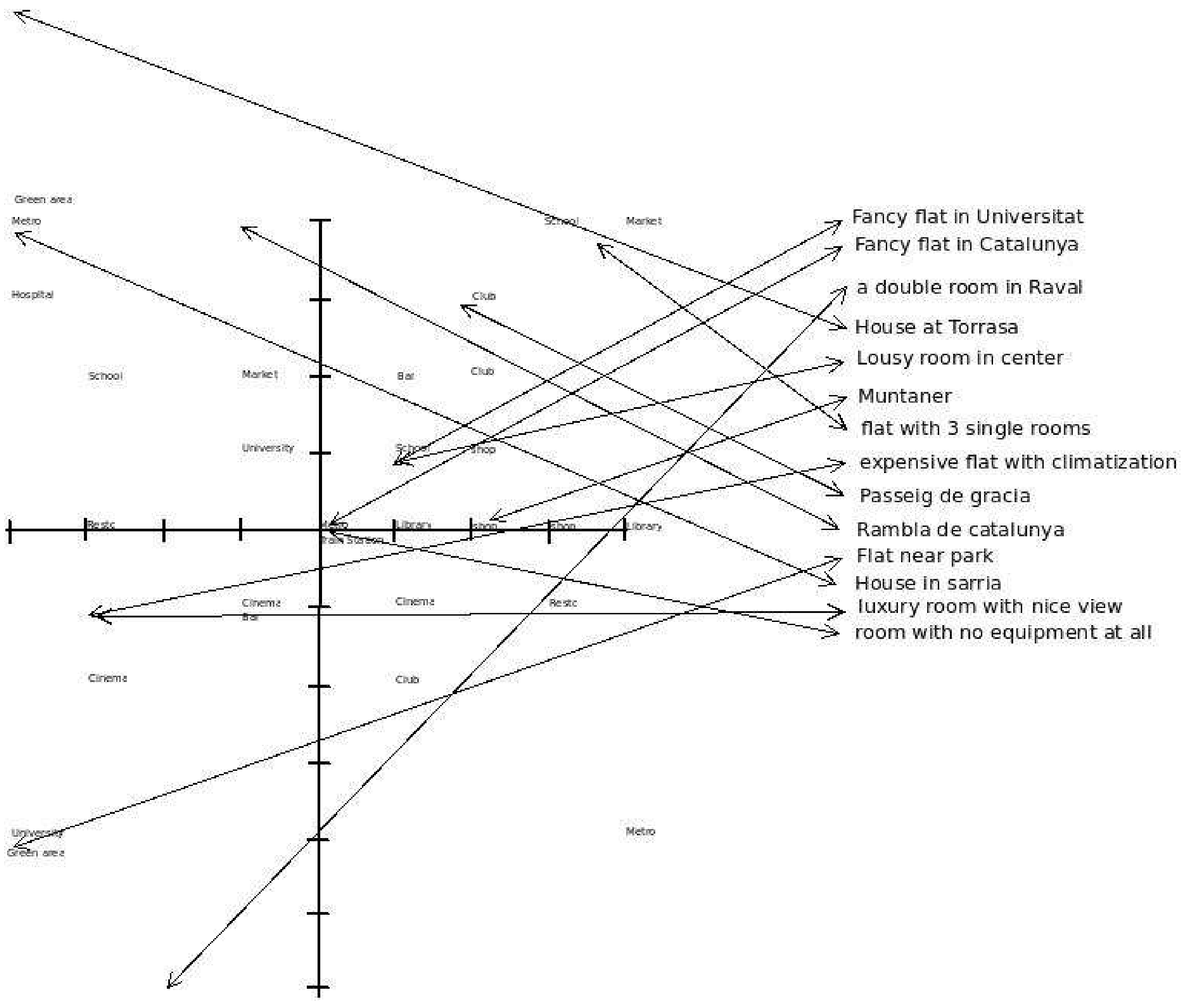


Image 3.6 : Geographical representation of the houses and services. The ones with an arrow are houses.

5. *Implementation - CLIPS Knowledge based Programming*

About CLIPS

CLIPS is a public domain software tool for building expert systems. The name is an acronym for “C Language Integrated Production System.” The first versions of CLIPS were developed starting in 1985 at NASA-Johnson Space Center until the mid 1990s when the development group's responsibilities ceased to focus on expert system technology.

CLIPS is probably the most widely used expert system tool because it is fast, efficient and free. Although it is now in the public domain, it is still updated and supported by the original author, Gary Riley. CLIPS incorporates a complete object-oriented language COOL for writing expert systems. Though it is written in C, its interface more closely resembles that of the programming language LISP. Extensions can be written in C, and CLIPS can be called from C. (Wikipedia)

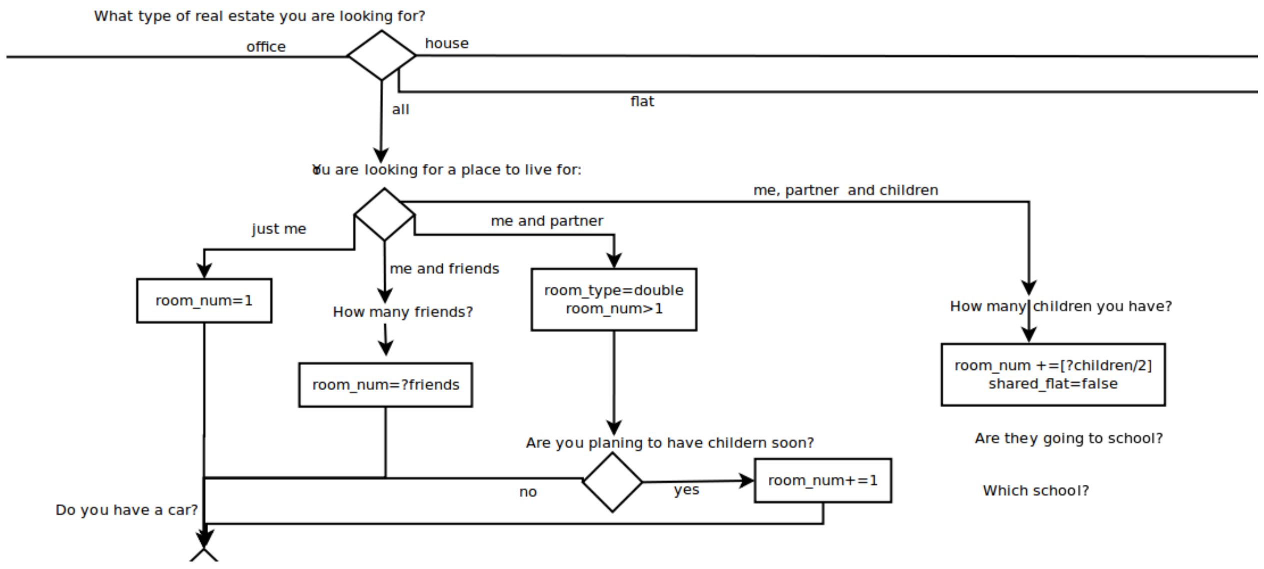
We decided to use a non-typical approach for asking question unlike any other housing web server. Our questions are not supposed to aim at very specific parameters but rather ask about client's preferences and of course to use our knowledge base. For example we do not need to ask at the beginning for preferred district in our town, we can choose the district according to other specified criteria. So, if client wants to live close to a park, we will choose districts with parks. In other words we would like to process information gained from normal human conversation and translate them into more technical way but with some boundaries for found results.

At the beginning we assume that all offers are accepted, after receiving some input from user we can apply some filtering. We decided to put some criteria as a hard one other as soft ones, this decision is sometimes a bit tricky because it can seriously affect results of system.

Hard criteria

- **price** - if user's says that he will not be paying more even for really great offer, results with higher price are filtered
- **type of realty** - it is non-sense to offer a house to person who is looking for a office, but sometimes there is not such a big difference between a room and a flat

We started with schema of question, so that some questions can be skipped or guessed by "smart" decision of KBS.



Strategy

We have chosen the "Depth" Strategy in order to solve our problem.

Snippet out of the Clips Basic Reference

Depth Strategy
newly activated rules are placed above all rules of the same salience. For example, given that fact-a activates rule-1 and rule-2 and fact-b activates rule-3 and rule-4, then if fact-a is asserted before fact-b, rule-3 and rule-4 will be above rule-1 and rule-2 on the agenda. However, the position of rule-1 relative to rule-2 and rule-3 relative to rule-4 will be arbitrary.

By choosing this algoritm we can make use of the salience property but can also choose not to. Modules are defined for having different sections in our program and not to have conflicts between different rules.

Added Classes/templates

```
(deftemplate recommendation
      (slot person)
    (slot is_final)
)
```

The deftemplate recommendation is a gathering of 1 person class (defined in the ontology) and a slot defining the state of the program.

```
(defclass Proposal
    (is-a USER)
    (role concrete)
    (slot score)
    (slot offer)
    (slot noise)
    (slot rooms)
    (slot room_diff)
    (slot is_proposed)
)
```

The proposal class is a wrapper class for an offer, and thus also includes an offer. Because we cannot define all these statical slots in our ontology - because it would make the ontology not generic anymore. In the start of the program we create new Proposal instances and step by step we fill in the scores, noise, rooms etc, everything we need during the program but was not included by the ontology.
The slot is_proposed is used as a boolean field and will be our hard-constraint filter, by default this is TRUE. This means that if during any rule an offer is rejected by a hard constraint, this value will go

on FALSE, if it is not rejected it will be offered as a possible solution to our user/client.

Messages

Having defmessages in our program was required for having a functional framework.

Print an Offer instance
```
(defmessage-handler Offer print()
```
Print a District instance
```
(defmessage-handler District print primary()
```
Print a Coordinates instance
```
(defmessage-handler Coordinates print primary()
```
Print an Address instance
```
(defmessage-handler Address print primary()
```
Print a Service instance
```
(defmessage-handler MAIN::Service print()
```
Print a Proposal instance
```
(defmessage-handler Proposal print()
```

Check if a Room instance is a double room
```
(defmessage-handler Room has-double-room primary()
```
Check if a Flat instance has a double room
```
(defmessage-handler Flat has-double-room primary()
```
Check if House instance has a double room
```
(defmessage-handler House has-double-room primary()
```
Check if Offer instance has a double room
```
(defmessage-handler Offer has-double-room primary()
```

Returns the number of habitable rooms for a Flat instance
```
(defmessage-handler Flat count-habitable-rooms primary()
```
Returns the number of habitable rooms for a Room instance
```
(defmessage-handler Room count-habitable-rooms primary()
```
Returns the number of habitable rooms for a House instance
```
(defmessage-handler House count-habitable-rooms primary()
```
Returns the number of habitable rooms for an Offer instance
```
(defmessage-handler Offer count-habitable-rooms primary()
```

Gets the type of the Realty included in the offer
```
(defmessage-handler Offer realty-type primary()
```

Check if an Offer includes a equipment
```
(defmessage-handler Offer has-equipment primary(?service)
```
Check if an PlaceToLive includes a equipment
```
(defmessage-handler PlaceToLive has-equipment primary(?service)
```

Functions

Question functions

Returns a question to the user with a yes or no answer allowed.

```
(deffunction yes-or-no (?question)
...
)
```

Checks if response is a number (float or integer)

```
(deffunction ask-number (?question ?range-start ?range-end)
  ...
)
```

Returns a question with a certain list of allowed answers. Checks if response is 1 in that list

```
(deffunction ask-question (?question $?allowed-values)
...
)
```

When answer does not fulfil given constraint, question is asked again.

Sorting proposals for listing

Sorting a list of instances is not an easy topic. After researching 3 different options and implementing 2 we choose for the 3rd one..
We tried to implement a Bubble Sort but the function gave an unlimited loop.

```
;Bubble sort!!
;(defrule bubble
;    ?recommendation <- (recommendation (is_final ?))
;    ?proposal1<-(object (is-a Proposal) (score ?score1))
;    ?proposal2<-(object (is-a Proposal) (score ?score2))
;    (test (> ?score1 ?score2))
;=>
;    (printout t "sorting")
;    (bind ?offer1 (send ?proposal1 get-offer))
;    (bind ?offer2 (send ?proposal2 get-offer))
;    (send ?proposal1 put-offer ?offer2)
;    (send ?proposal2 put-offer ?offer1)
;)
```

After trying this we reached out for a more feasable solution and use the sorting function in clips. This function is based on the behaviour of the sort function in clips. Sort allows a list of values to be sorted based on a user specified comparison function.
The function can be used as followed :

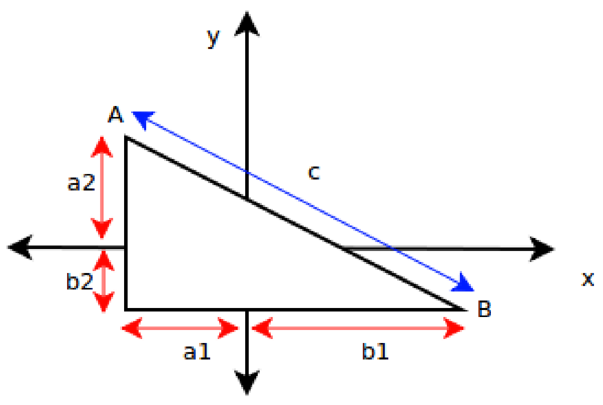
```
(sort proposal-sort ?instance1 ?instance2 ?instance3 ?instance4)
```

And will return a multifield with the instances sorted by how the proposal-sort comparision evaluates it.

```
(deffunction proposal-sort (?prop1 ?prop2)
  (< (send ?prop1 get-score) (send ?prop2 get-score))
)
```

Counting distance

This is distance in air, without consideration of street, which might not be exact enough in real world. We count Euclidean distance with following formula:



CLIPS implementation:

```
(deffunction distance (?c1 ?c2)
   (bind ?x (- (send ?c1 get-longitude) (send ?c2 get-longitude)))
   (bind ?y (- (send ?c1 get-latitude) (send ?c2 get-latitude)))
   (bind ?result (sqrt (+ (** ?x 2) (** ?y 2))))
   return ?result
)
```

Noise impact

```
asks the distance between 2 addresses and returns the noise impact factor if it is
close/medium/far
;Counts impact of noise between two Addresses and return
; 2 == close
; 1 == medium
; 0 == far
(deffunction noise-impact (?adr1 ?adr2)
...
)
```

In our case sources of noise are just "points", we did not took in consideration sources of noise like roads which can cover huge area.

Module "initiation"

In this module we define our header and go to the first module - house-queries

```
(defrule start
    (declare (salience 10))
    =>
    (printout t "------------------------------------------" crlf)
    (printout t "------ Expert system to find a house -----" crlf)
    (printout t "------------------------------------------" crlf)
    (printout t crlf)
    ;;;go to module personal-questions
    (focus personal-questions house-queries)
)
```

Module "personal-questions"

In this module we ask the user a number of questions to know his/her personal information such as age/name and also add facts based on our interpretation of the answers.

We started out with a basis of facts and extended it piece by piece so we worked according to the incremental development methodology. For example, we started with asking the name and the age but nearing a new cycle we decided to replace the question of the age with a more sophisticated question that reveals more information then just the age. For example being 21 does not reveal if a person is a student or just young.

An example is included :

```
(defmodule personal-questions "Module to know the needs of our user"
    (import MAIN ?ALL)
    (export ?ALL)
)

(defrule your-name "Find out our personal characteristics"
    (not (object (is-a Person)))
    =>
    (bind ?firstname (ask-open-question "What is your firstname"))
    (bind ?surname (ask-open-question "What is your surname"))

  ;;;create an instance of Person
    (bind ?user (make-instance user of Person))
  ;;;add this to our instance of Person
    (send ?user put-firstname ?firstname)
    (send ?user put-surname ?surname)

  ;;;insert this Person into recommendation
    (assert (recommendation (person ?user)))
  ;;;add facts that our first and surname are ok
    (assert (Person firstname ok))
    (assert (Person surname ok))
    (assert (Person complete ok))
)


;;; ASK FOR WHICH KIND OF ENVIRONMENT
;;; ASSERT FACT IF A SERVICE SHOULD BE CLOSE(TRUE) OR FAR(FALSE)
(defrule house-environment
    (not (Person type-of-environment ?))
    ?user <- (object (is-a Person))
    =>
    (bind ?type-of-environment (question "what kind of environment do you want to live
in?" quiet centric young residential outskirts))
    (if (= (str-compare ?type-of-environment "quiet") 0)
        then
        (assert (Person max-noise 3))
    (assert (Person bar FALSE))
    )
    (if (= (str-compare ?type-of-environment "centric") 0)
        then
            (assert (Person bar TRUE))
            (assert (Person public-transport TRUE))
            (assert (Person shopping TRUE))
            (assert (Person location centric))
    )
    (if (= (str-compare ?type-of-environment "young") 0)
        then
        (assert (Person bar TRUE))
    (assert (Person education TRUE))
    )
  (if (= (str-compare ?type-of-environment "residential") 0)
        then
    (assert (Person max-noise 5))
        (assert (Person shopping TRUE))
    (assert (Person foodbeverage TRUE))
    (assert (Person healthcare TRUE))

    )
  (if (= (str-compare ?type-of-environment "outskirts") 0)
        then
    (assert (Person max-noise 1))
        (assert (Person green-area TRUE))
    )
)
```

We finish by creating a proposal instance for every offer so we can add different application specific slots to it:

```
(defrule create-proposals
    (Person complete ok)
    ?offer <- (object (is-a Offer))
    =>
    ;;;initialize our system and so get all instances of offer and copy them into
    ;;;a new instance proposal
     (make-instance of Proposal
    (score 0)
    (offer ?offer)
    (is_proposed TRUE)
    (noise 0.0)
    (rooms 0.0)
    (room_diff 0)
       )
)
```

More rules are visible in the code attached to this document!

Module "house-queries"

In this module we make all the proposal instances and import the data from personal-questions, also we filter the proposals based on the facts from personal-questions.

```
(defmodule house-queries "Module that gathers information about the searched house"
    (import MAIN ?ALL)
    (import personal-questions ?ALL)
    (export ?ALL)
)
```

Used in order to add the noise points to each proposal based on how close/far a service is from that proposal. This function uses all of the specific noise methods and fields needed in order to succesfully compare noise values with eachother.
We loop through every service and every proposal instance and compare the distance from eachother, after retrieving a Close, Medium or Far we decide how much noise factor we add to that proposal.

```
;;; Loop trough all the houses and locations and give noisynesspoints
;;; if a location is close add the whole noisynesspoints
;;; if a location is medium add the half of the noise
;;; if a location is far - dont do anything
(defrule calculate-noise ""
  (declare (salience 20))
  (Person facts ok)
  ?proposal<-(object (is-a Proposal))
  ?service<-(object (is-a Service))
    =>
  (bind ?noise-weight 0.5) ;;TODO put it as a global variable
  (bind ?adr1 (send ?service get-address))
  (bind ?adr2 (send (send ?proposal get-offer) get-address))
  ;(printout t (count-distance ?adr1 ?adr2) crlf)
  ;(printout t (* ?noise-weight (noise-impact ?adr1 ?adr2)) (send ?service get-title)
crlf)
   (send ?proposal put-noise (+ (send ?proposal get-noise)
                    (* ?noise-weight (noise-impact ?adr1 ?adr2))
      ))
)
```

Example function - Filter the noisy houses if user has a fact with a specified max-noise
This is not a hard constraint but we subtract some score from the proposal noise field.

```
(defrule filter-noisy "lower score of proposal with noisy environment if user does
mind"
  (Person facts ok)
  (Proposal noisiness ok) ; must be executed after noise is calculated
  (Person max-noise ?) ; noise is set up
  ?proposal<-(object (is-a Proposal) (offer ?offer))
  ?fact <- (Person max-noise ?noise)
  =>
    (bind ?diff (- ?noise (send ?proposal get-noise)))
    (if (>= ?diff 0)
      then (if (>= ?diff 1) ;less noisy than is requested
        then  (send ?proposal put-score (+ (send ?proposal get-score) 2))
            else  (send ?proposal put-score (+ (send ?proposal get-score) 1))
           )
```

```
     else (if (>= ?diff -1)
        then  (send ?proposal put-score (- (send ?proposal get-score) 1))
          )
          (if (>= ?diff -2)
        then (send ?proposal put-score (- (send ?proposal get-score) 2))
        else (send ?proposal put-score (- (send ?proposal get-score) 3))
          )
      ;reject very noisy offers
          (if (< ?diff -5)
        then (send ?proposal put-is_proposed FALSE)
          )
      )
)
```

Calculate if the fact green-area is added and if so, add/retract points from offers close/far from a green-area

```
;;; Loop trough all the houses and locations and give points accordingly
;;; if a location is close add 2 points
;;; if a location is medium add 1 points
;;; if a location is far - dont do anything
(defrule calculate-green-area ""
  (declare (salience 20))
  (Person facts ok)
  (Person green-area ?greenarea)
  ?proposal<-(object (is-a Proposal))
  ?service<-(object (is-a FoodBeverage))  ;use the superclass to get all the transports
    =>
  (bind ?adr1 (send ?service get-address))
  (bind ?adr2 (send (send ?proposal get-offer) get-address))
  (bind ?distance (measure-distance-adr ?adr1 ?adr2))

  ;;;Define our pointsdevision
    (bind ?closepoints 2)
    (bind ?midpoints 1)
    (bind ?farpoints -2)

    (switch ?distance
     (case close then
        (if (eq ?greenarea TRUE)
          then
          (send ?proposal put-score (+ (send ?proposal get-score) ?closepoints))
          else
          (send ?proposal put-score (+ (send ?proposal get-score) ?farpoints))
        )

        ;;;if our service is close then enable the proposal again

     )
     (case mid then (send ?proposal put-score (+ (send ?proposal get-score)
?midpoints)))
     ;location is too far away
     (case far then
        (if (eq ?greenarea FALSE)
          then
          (send ?proposal put-score (+ (send ?proposal get-score) ?farpoints))
          else
          (send ?proposal put-score (+ (send ?proposal get-score) ?closepoints))
        )
     )
  )
)
```

End of the filtering functions so we know we can continue to the following module

```
;;; END OF OUR FILTERING METHODS. ADD ALL FUNCTIONS ABOVE THIS LINE
(defrule end-of-filtering
    (Person facts ok)
      ?recommendation <- (recommendation (is_final ?))
    =>
    (modify ?recommendation (is_final ok))
    (pop-focus)
)
```

more rules are visible in the code attached to this document !

Module "output"

In this module we have all the rules that take care of our output. We know that all our functions are completed and we have a full result set to output.

Also here we define what is a very recommendable offer/adequate offer and partially adequate
If an offer is between 100% - 90% of the score of the maximum score that was achieved in our proposals it deserves to be called very recommendable
If an offer is between 90% - 70% of the score of the maximum score that was achieved in our proposals it deserves to be called adequate
If an offer is below 70% of the score of the maximum score that was achieved in our proposals it deserves to be called partially adequate
Offers that were discarded because of a hard constraint are not visible and will not be accounted in this output.

```
;;;---------------------------------------------- -
;;;- define a module for the output of our program -
;;;-----------------------------------------------
(defmodule output "Module for outputting the results"
    (import MAIN ?ALL)
)
```

more rules are visible in the code attached to this document !

Module "EOP" - end of program

When everything is finished this module is executed. It is good to know this because then you are sure the program arrived in the last step.

```
(defmodule EOP "end of program"
  (import MAIN ?ALL)
)

 (defrule endprogram "final rule"
   (declare (salience 10))
   ?recommendation <- (recommendation (person ?person) (is_final finished))
    =>
  (printout t "---------------------------------------------------------------"
crlf)
  (printout t "Thank you for using our housing service" crlf)
  (printout t "-----------------------------------------------------------------"
crlf)
  (printout t crlf)
  (pop-focus)
)
```

Exceptions and Trivialities

User input

We tried to check user's input so that we would be able to handle typo mistakes and also to prevent exceptions like bad type conversion. As long as we are working just with command line interface our possibilities are very limited. We have implemented check for integer numbers with function

```
(numberp ?exp)
```

When user enter invalid value, the question is repeated and he can answer again.

Sort values

We tried to sort our values but it was not evident for applying all our instances to the same function. In the end we solved it by generating the correct command in a string and then execute this string as if it was a clips command

```
 ;;;create our multifield to sort it later
  (bind ?multifield (create$))
    (do-for-all-instances
        ((?proposal Proposal))
        (eq (send ?proposal get-is_proposed) TRUE)
        ;action
      (bind ?multifield (insert$ ?multifield 1 ?proposal))
    )

  ;;;add to the slot
  (bind ?concatenatedstring (str-cat "(sort proposal-sort " (implode$ ?multifield)
")"))
  (bind ?proposals (eval ?concatenatedstring))
```

Accessing slots

We encountered problems with accessing slots defined in abstract super-class, because there's no public getter. We solved it by adding property `(propagation inherit)` to definition of slot, this is probably not supported in Protégé when exporting it to clips.

In case of evaluating expression we did not find how to evaluate just first part of "and" and not the others when it's not necessary because first one already failed. Something like operator "&&" in Java would be nice for joining conditions.

Fast-forward decisions

In some cases we are able to fast forward certain decisions and questions. We refer back to the example of the age. When we ask a person for his/her occupation we know more then just the age. When a person says he is a student instead of asking the age directly we can say that this client needs education services close by. The same when the occupation is retired, if a person fills in age 55 we cannot say with enough certainty that he/she needs health care close by. If we don't know this information we can ask the question of the age. This way we fast forwarded a decision.

Another example of taking advantage of rules programming is a case when we are asking if partners are going to have extra room, then we should add some extra room for soon coming baby. But when we know that this couple is already retired is not very likely that they are going to have children, so we can directly skip this question. This is the way how we programmed it in CLIPS:

```
;retired person is not going to have children soon
(defrule person-living-with-partner
    (Person type-of-family ?type-of-family&partner)
    (Person occupation ?occ&~retired)
    ?user <- (object (is-a Person))
  =>
       (if (yes-or-no "are you planning to have children soon")
       then
           (assert (Person children 1))
       )
       ;;in any case we need a double room
       (assert (Person room-type double))
       (assert (Person room-num 2))
)
```

Our realties

Flats

	Title	*Rooms*	*Double room*	*Price (€)*	*Suitable for*
F1	flat near park	2	yes	950	
F2	flat with 3 single rooms	3	no	750	TC2
F3	expensive flat with climatization	4	yes	7000	
F4	Muntaner	2	yes	800	TC2
F5	passaige de gracia	2	yes	800	

F6	Rambla de la C.	4	yes	2500	
F7	Fancy flat at Catalunya	4	yes	1200	
F8	Fancy flat at Universitat	4	yes	1300	
F9	flat without windows	2	no	600	TC6

Rooms

	Title	Rent	District	Suitable for
R1	a double room in Raval	560	El Raval	
R2	luxury room with nice view	450	Eixample esquerra	
R3	Lousy room in center	310	Ciutat Vella	TC1
R4	room with no equipment at all	250	Ciutat Vella	
R5	room close to UPC	360	Les Corts	

Houses

	Title	Rent	District	Suitable for
H1	expensive house in Sarria	6000	Sarria	
H2	house at Torrasa	1350	Torrasa	TC2
H3	House for demolition	800	El Raval	

Test cases

For the sake of testing our program we created a few test cases with different requirements.
The test cases have been chosen in order to create enough diversity and to show us and the evaluator that our program gives adequate results. Please feel free to try and test out the program yourself.

See image 3.6 for a geographical representation of the location of the houses.

TC1 "Tom Barton" - Young

Young student is looking for a flat somewhere near to green line or with very good connection to Palau Real (UPC). He doesn't mind noise and prefer some Spanish seeking flatmates with friendly nature. Maximum price he can afford is 350€ mothly. He would like to have some pubs and places where to go out around, if not there must be a connection by night bus from centre of town.

Expected solution

We would choose a flat in centre, preferably Ciutat Vella, probably R3 would be better because R4 is too cheep to be good.

TC2 "Veenhof family" - Residential

Mr Veenhof is going to move to town because of promotion in company, he's going to be a director of a Catalan division and he will stay here for one year. Nick owns a car but he stays quite long in work and his wife Joliene doesn't have a car for picking up Jane from school. A younger daughter Annie is 5 years old and she will start attending school probably next year. They would like to live in a quite neighbourhood not too far from supermarkets. Joliene would love to be close to centre close to cultural events. They can afford to pay 2000€ per month because Nick has to still pay for mortgage on house in Belgium.

Expected solution

We do not have exact match there is just one offer H2 which is suitable for this case.

TC3 "The McQueen's" - Quiet place for 2 older people

The McQueen's are already retired and they are looking for a flat close to a park where they could go out with their dog Buddy. They prefer quiet locality but close to a market and if possible also close to hospital, but this is currently not so important. They are able to pay from their pension 900€ and they would like to have a double room.

Expected solution

We would have chosen flat close to park, which is F1 although is a bit more expensive (950€), it is the best offer we have.

TC4 "The Daltons family" Outskirts

The Dalton's is one of those lucky ones who do not suffer of financial crisis, so they are willing to pay 6000€, they have 4 children and they don't want to live in smoggy and noisy environment of town centre.

Expected solution

For this case we would choose H1, which seems to be quite nice.

TC5 "The Smiths" - centric

Smith's family do not have a car because they did not need it in centre of town. They have three children Jordi, Eli and Pol. They are 3,5 and 6 years old, Jordi and Pol can be in one room but for Eli they would like to have another room. They are willing to pay up to 2000€.

Expected solution

In this case we would choose one of centric flats either F7 or F8.

TC6 "Jack & Jim Daniels" - students

Jack and Jim just arrived to town for an Erasmus program and they would like to live together because they are very good friends since high school. They have grant from their school 200€ each. With money from their parents they can pay monthly 650€ together. But they would also like to spare some money for parties. They prefer to live in area close to other students rather than close to school. Although they are best friends Jack keeps working late till night and on the other hand Jim prefer to work in morning, so that they can't live one room. Two single rooms would be really nice.

Jack and Jim loves travelling and they don't care much about equipment of the apartment, they would just appreciate an internet connection.

Expected solution

We would choose for them some centric flat round 600€, so it's either F10 or F4.

Output of TC1 "Tom Barton" - Young

```
What is your firstname? Tom
What is your surname? Barton
What is your current occupation? (working student retired other) student
What type of real estate are you looking for? (office house flat) flat
You are looking for a place to live for? (alone partner children friends) alone
Do you have a car? (yes no y n) no
what kind of environment do you want to live in? (quiet centric young residential
outskirts) young
What is your maximum rental budget per month? [0, 90000] 350

Are you willing to pay more for the house of your dreams?? (yes no y n) n

What is the cheapest price you would pay? [0, 3000] 200

---------------------------------------------------------------------

Sr/a. Tom,Barton

This is the list of proposals

---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: Room
Offer: Lousy room in center
Price: 310.000000
Address: Ronda de la Universitat, Ciutat Vella  [1.000000;1.000000]
Score: 19.000000
Noise: 15.000000
Rooms: 1.000000
---------------------------------------------------------------------
Adequate offers
---------------------------------------------------------------------
```

```
Type: Room
Offer: room with no equipment at all
Price: 250.000000
Address: Placa Catalunya, Eixample Dreta  [0.000000;0.000000]
Score: 17.000000
Noise: 14.500000
Rooms: 1.000000
```

Evaluation

This answer seems to be correct, student does not want to pay anything extra, so the offer "room close to UPC" for 360€ is correctly excluded. We did not implement counting distance from metro lines but it might not be so difficult as long as we have limited number of stations.

Output of TC2 "Veenhof family" - Residential

```
What is your firstname? Nick
What is your surname? Veenhof
What is your current occupation? (working student retired other) working
What type of real estate are you looking for? (office house flat nevermind) nevermind
You are looking for a place to live for? (alone partner children friends) children
How many children are living with you? [0, 20] 2
Currently we estimated for you 3 rooms is this enough? (yes no y n) yes
Do you have a car? (yes no y n) yes
Are you bringing the children to school by car? (yes no y n) yes
what kind of environment do you want to live in? (quiet centric young residential
outskirts) residential
What is your maximum rental budget per month? [0, 90000] 2000
Are you willing to pay more for the house of your dreams?? (yes no y n) yes
What is the cheapest price you would pay? [0, 3000] 300
What is your age? [18, 100] 40
---------------------------------------------------------------------
Sr/a. Nick,Veenhof
This is the list of proposals

---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: House
Offer: house at Torrasa
Price: 1350.000000
Address: nowhere, Middle Of Nowhere  [-15.000000;15.000000]
Score: 26.000000
Noise: 0.000000
Rooms: 8.000000

Type: Flat
Offer: Rambla de la C.
Price: 2500.000000
Address: Rambla De Catalunya, Eixample Dreta  [-1.000000;4.000000]
Score: 25.000000
Noise: 2.500000
Rooms: 4.000000

---------------------------------------------------------------------
Adequate offers
---------------------------------------------------------------------
---------------------------------------------------------------------
Partially adequate offers
---------------------------------------------------------------------
Type: House
```

```
Offer: house for demolition
Price: 800.000000
Address: Carrer de Sant Rafael 12, El Raval  [-3.000000;-2.000000]
Score: 17.000000
Noise: 3.000000
Rooms: 3.000000

Type: Flat
Offer: flat with 3 single rooms
Price: 750.000000
Address: Carrer Congos 4, Hospital Clinic  [3.000000;4.000000]
Score: 15.000000
Noise: 4.500000
Rooms: 3.000000
```

Evaluation

This is a bit tricky test case, family is looking for a residential area, but we do not have any. System has not found any adequate offer. "house for demolition" would not be chosen by normal person, but there is no fact according to which we can decide that this is a lousy offer. From according to given criteria this is the best offer. It's pretty hard to find out which part of city is residential unless we have this fact from an "expert". Anyway we were pleasantly surprised by this results, flat in centre is also not exactly what is requested but math another criteria, so it seems as a very good result.

Output of TC3 "The Mcqueens" - Quiet place for 2 older people

```
What is your firstname? Arthur
What is your surname? Mcqueen
What is your current occupation? (working student retired other) retired
What type of real estate are you looking for? (office house flat nevermind) flat
You are looking for a place to live for? (alone partner children friends) partner
Currently we estimated for you 2 rooms is this enough? (yes no y n) yes
Do you have a car? (yes no y n) no
what kind of environment do you want to live in? (quiet centric young residential
outskirts) quiet
What is your maximum rental budget per month? [0, 90000] 900

Are you willing to pay more for the house of your dreams?? (yes no y n) y

What is the cheapest price you would pay? [0, 3000] 500

---------------------------------------------------------------------

Sr/a. Arthur,Mcqueen
This is the list of proposals
---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: Flat
Offer: flat near park
Price: 950.000000
Address: Carrer Parc Mar, Sant Marti: Diagonal Mar  [-4.000000;-4.000000]
Score: 1.000000
Noise: 2.500000
Rooms: 2.000000
```

Evaluation

This response was evaluated exactly as we wanted. Although the price is higher than was requested it still fits into our limit, which is 30% of budget. In many cases this settings of border might be tricky, maybe applying of fuzzy logic might give more precise results.

Output of TC4 "The Daltons family" Outskirt

```
What is your firstname? George
What is your surname? Dalton
What is your current occupation? (working student retired other) other
What type of real estate are you looking for? (office house flat nevermind) house
You are looking for a place to live for? (alone partner children friends) children
How many children are living with you? [0, 20] 4
Currently we estimated for you 4 rooms is this enough? (yes no y n) y
Do you have a car? (yes no y n) y
Are you bringing the children to school by car? (yes no y n) yes
what kind of environment do you want to live in? (quiet centric young residential
outskirts) outskirts
What is your maximum rental budget per month? [0, 90000] 6000
Are you willing to pay more for the house of your dreams?? (yes no y n) yes
What is the cheapest price you would pay? [0, 3000] 800
What is your age? [18, 100] 45
---------------------------------------------------------------------
Sr/a. George,Dalton
This is the list of proposals

---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: House
Offer: house in Sarria
Price: 6000.000000
Address: Passaige de Marcela Rubia, Sarria-Sant Gervasi  [-4.000000;4.000000]
Score: 7.000000
Noise: 2.500000
Rooms: 4.000000

---------------------------------------------------------------------
Adequate offers
---------------------------------------------------------------------
---------------------------------------------------------------------
Partially adequate offers
---------------------------------------------------------------------
Type: House
Offer: house at Torrasa
Price: 1350.000000
Address: nowhere, Middle Of Nowhere  [-15.000000;15.000000]
Score: 4.000000
Noise: 0.000000
Rooms: 8.000000

Type: House
Offer: house for demolition
Price: 800.000000
Address: Carrer de Sant Rafael 12, El Raval  [-3.000000;-2.000000]
Score: -3.000000
Noise: 3.000000
Rooms: 3.000000
```

Evaluation

In here could be a bit disputable the noise factor which is counted in our own units. If the noise does matter second offer could be better. With a car the house in “middle of nowhere” could be also interesting possibility.

Output of TC5 "The Smiths" - centric

```
What is your firstname? Will
What is your surname? Smith
What is your current occupation? (working student retired other) working
What type of real estate are you looking for? (office house flat nevermind) nevermind
You are looking for a place to live for? (alone partner children friends) children
How many children are living with you? [0, 20] 1
Currently we estimated for you 2 rooms is this enough? (yes no y n) y
Do you have a car? (yes no y n) yes
Are you bringing the children to school by car? (yes no y n) no
what kind of environment do you want to live in? (quiet centric young residential
outskirts) centric
What is your maximum rental budget per month? [0, 90000] 2000
Are you willing to pay more for the house of your dreams?? (yes no y n) n
What is the cheapest price you would pay? [0, 3000] 1000
What is your age? [18, 100] 35
---------------------------------------------------------------------
Sr/a. Will,Smith
This is the list of proposals

---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: House
Offer: house at Torrasa
Price: 1350.000000
Address: nowhere, Middle Of Nowhere  [-15.000000;15.000000]
Score: 30.000000
Noise: 0.000000
Rooms: 8.000000

---------------------------------------------------------------------
Adequate offers
---------------------------------------------------------------------
Type: Flat
Offer: Fancy flat at Universitat
Price: 1300.000000
Address: Placa Universitat, Eixample esquerra  [1.000000;1.000000]
Score: 22.000000
Noise: 15.000000
Rooms: 4.000000

---------------------------------------------------------------------
Partially adequate offers
---------------------------------------------------------------------
Type: Flat
Offer: Fancy flat at Catalunya
Price: 1200.000000
Address: Placa Catalunya, Eixample Dreta  [0.000000;0.000000]
Score: 20.000000
Noise: 14.500000
Rooms: 4.000000
```

Evaluation

Although we wanted centric location here is a flat which is obviously outside of centre on the first place. Here we come to non-trivial problem of settings weights to each criterion, this sorting priorities could be hard also for normal people. Our guess is in category “adequate” which is not that bad, 2nd position is probably caused by noisy environment which is almost characteristic for centre of town.

Output of TC6 - "Jack & Jim Daniels" - students

```
What is your firstname? Jack
What is your surname? Daniels
What is your current occupation? (working student retired other) student
What type of real estate are you looking for? (office house flat nevermind) flat
You are looking for a place to live for? (alone partner children friends) friends
Currently we estimated for you 2 rooms is this enough? (yes no y n) yes
Do you have a car? (yes no y n) n
what kind of environment do you want to live in?? (quiet centric young residential
outskirts) young
What is your maximum rental budget per month? [0, 90000] 650

Are you willing to pay more for the house of your dreams?? (yes no y n) n

What is the cheapest price you would pay? [0, 3000] 200

---------------------------------------------------------------------

Sr/a. Jack,Daniels
This is the list of proposals
---------------------------------------------------------------------
Very recommendable offers
---------------------------------------------------------------------
Type: Flat
Offer: flat without windows
Price: 600.000000
Address: Carrer de Sante Pere Mitja 10, Barri Gotic  [-2.000000;2.000000]
Score: 15.000000
Noise: 7.000000
Rooms: 2.000000
```

6. Testing

CLIPS language is not very programmer friendly, debugging is complicated and moreover there are not many sources on internet and forums with support. We created a few scripts to make our work easier.

Compilation

Firstly we have a PERL script to compile all program files into one "housing.clp" file, we call it "make".

```
#!/usr/bin/perl
#
# script for building CLIPS program from Protoge export
# saves output to "$OUT_FILE"
# @author deric
$CLIPS_FILE="clips.clp";
$OUT_FILE="housing.clp";

open OUT, ">$OUT_FILE" or die $!;
 my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime time;
 $year=$year+1900;
print OUT ";; Builded on  $year-$mon-$mday $hour:$min:$sec \n";
print OUT "\n;############### ontology ##################\n";
$pont="export/housing.pont";
open(FILE, $pont) || die("Could not open file $pont!");
while (<FILE>) {
  print OUT $_;
  $i++;
}
close FILE;
print OUT "\n\n
;############### instances ##################
(definstances inst \n";

$inst="export/instances.pins";
open(FILE, $inst) || die("Could not open file $inst!");
while (<FILE>) {
  print OUT $_;
  $i++;
}
close FILE;
print OUT ")";
print OUT ";;############### program ##################\n";open(FILE, $CLIPS_FILE) ||
die("Could not open file $CLIPS_FILE!");
@i=0;
while (<FILE>) {
  print OUT $_;
  $i++;
}
close FILE;
close OUT;
print "housing build completed. $i lines\n";
```

Then we have a CLIPS file "execute.clp" this file is passed as an argument to CLIPS command line interpreter in file "run".

```
(clear)
(load "housing.clp")
(reset)
(run)
```

"run" - in "f" file are CLIPS commands which are executed line by line, at the end must be an empty line.

```
#!/bin/sh
./make
clips -f execute.clp
```

By typing "./run", in our project directory, all files are compiled and CLIPS program is running.

Testing

For execution of our test cases we used an "ex" command which compile program, run it and pass all answers straight to CLIPS

```
#! /bin/sh
./make
clips -f $1
test case file "tests/house_centric":
(clear)
(load "housing.clp")
(reset)
(run)
Nick
Veenhof
student
flat
children
1
y
yes
no
centric
1300
n
300
```

For execution we used:

```
$ ./ex tests/house_centric
```

7. Conclusion

Developing and “solving” this project in a knowledge based system gives us certain advantages that are hard to solve in another programming language. By having a non-linear approach we can easier tackle bigger problems. Although the programming language was not our favourite in the beginning, nearing the end of the project we were busier discussion the conceptual issues then programming issues which is positive.

Quite complicated conditions, which would have been written in other languages with many “if” clauses, are pretty simple in CLIPS. When we separated rules into blocks or modules, code became more readeable. Sometimes it was complicated to find out which rule is going to be executed and which not. Development would be much easier with a friendly developing environment.

Also by creating the ontology we learned to generalize concepts and not to work in function of a specific program, even though it is dangerous to not look around the corner and adjust your ontology in function of the program.

Although beginning of this project was really tough, after we realized what rule based programming is all about, work became much easier. With this project we learned a lot about knowledge based problems and specifically how to solve this problem with CLIPS. The CLIPS language is a nice added value, mainly because it is familiar with other knowledge based languages and also because is completely different from most common languages.

We did not manage to implement all possible criteria which are hard to test together, because it is hard to eliminate side effects caused by different rules. Sill we think that we managed to implement functional application which can be further extended.

Future work

There is still a lot of work on extending questions and also setting weights to different facts and evaluating complicated cases. Also testing on "real" people would be nice and adding much more offers.

8. Bibliography

Culbert, R. S. *CLIPS Reference manual - Basic Programming guide.*
McGuinness, N. F. *Ontology Development 101.* Stanford University, Stanford, CA, 94305.
Wikipedia. *CLIPS.* Retrieved 21 30, 2009, from http://en.wikipedia.org/wiki/CLIPS