AI Laboratory

Housing

Project Members:

Anshuman Mehta

Andrea Gritti

Matteo de Martino

1 Contents

2 Design And Implementation Methodology

2.1 Identification

2.1.1 Description

The public office for house renting of the Department of Housing wishes to facilitate the process of renting a house to its citizens as the housing market is at an all-time low.

We have to develop a knowledge based system using the CLIPS language that helps solve the above problem.

2.1.2 KB and Knowledge Sources

This office has information about house renting all over the country but our system will only recommend housing rentals for AnyWhereTown. This information as the source to construct our KB.

Each offer of housing rental includes the most relevant characteristics of the house eg. Monthly rent in euros, habitable space in metre ^{squared} , coordinates etc.

Also there is a list of all the service/areas of the city and their coordinates. This list of services will be used as the source for deducing information about the city and is used in the construction of the KB.

2.1.3 Goals and Expectations

This knowledge based system will interact with the citizens of AnyWhereTown and will try to profile them and then use the knowledge of the city to deduce characteristics of the city and the rental offer which could be more important to them.

The questionnaires would be oriented around the citizens' age group, daily routine and if applicable any of their specific preferences.
These facts about a citizens' profile will be run through an inference engine and will suggest to the citizen rental offers that would be more to his/her liking and will suit their lifestyle i.e the appropriate services/area would be near or far.

It is expected that this enjoyable interaction for the citizens of AnyWhereTown and the useful recommendations provided by the KBS will help revive the housing market by giving the citizens a one-stop information centre for rental-offers which automatically recommends rental offers based on their preferences.

2.2 Conceptualization

2.2.1 Domain

The domain for this project is the House Rental Market.
The ontology then describes all the information which is related to the House Rental Market including the knowledge sources.

The ontology will have to cover the types of houses available for rent in the market and the specific characteristics of the rental offer for eg. Only 1 room for rent and also specific characteristics of the house itself for eg. Has a good view and the address.
In addition, the services in the city and their addresses will be covered in the ontology.

Also the ontology will have to cover the recommendations of the rental offer that the KBS will provide as an output to a user whose profiling information will also be captured in the ontology.

2.2.2 Problems and Sub-Problems

The typical decisions that people make when they look for a house for eg. the rent, whether public transport is nearby or is there shop nearby etc. will be captured and identified by the system.
So the project will divide the Housing Rental problem into the following problems and sub-problem to achieve its goal.

- Profile The User
  - Determine Name
  - Age group
  - Mode of Transport
  - Weekend Activities
  - Shopping trends (holds discount cards for specific malls)
  - The budget.

- Fetch Rental Offers
  - Load the current available rental offers in the system.

- Filter Rental-Offers by User Budget
  - Determine user's flexibility
  - Disregard rental offers which are outside user limits.
  - Assign scores

- Advance Profiling of The User
  - Determine housing preferences like view, house or apartment etc.
  - Determine more user activities like sports, university etc.

- Filter Rental-Offers by Advanced Profiling
  - Filter by user housing preferences. Assign Scores

- Filter by user activities. Assign Scores.

- Filter Rental-Offers by Distance
  - According to user profile certain services/area should be close or far.
  - Check distances for services according to user profile.
  - Assign Scores.

- Asses Scores and Give Recommendations
  - Analyse scores and create recommendations.
  - Print recommendations and recommendation levels (calculated by scores).

2.2.3 Informal Solution

Given the methodology we have proposed above our KBS will actively receive information about a Citizen and can assert facts about this user's profile. These when passed through an inference engine will trigger the appropriate rules and the KBS will be able to filter out the rental offers which will tend towards the user's lifestyle, active preferences about renting and also given the user a good overview of which services/areas are near, far or mid-distance from those rental offers.

Also since a user will be profiled for instance according to his/her age group as young then for that user it may be more important to live in a place which is near to bars, clubs or his/her university.

2.3 Formalization

2.3.1 Ontology – choice

The Ontology has been built for the Housing Rental market domain using Protege. The abstracted concepts and relations then in this ontology developed are as follows:
(The detailed ontology description is in the next section)

- *Coordinates*
  - Represents the 2 dimensional coordinates of any place in AnyWhereTown.

- *Address*
  - Represents the address of any place in AnywhereTown and contains an instance of *Coordinates*.

- *Residential*
  - It represents a *house* or *apartment* which exists in AnyWhereTown.
  - As a house it can have these *variants* :
    - *villa, attic, detached, semi-detached*
  - As an apartment it can have these *variants* :
    - *duplex, flat, penthouse*
  - Will have an instance of *Address*

- Features of the residence like sunlight, rooms for rent etc.

- *Services*
  - Represents a service which exists in AnyWhereTown.
  - A service will have a *serviceType* which can be as following :
    - *greenArea, recreation, school, shops, health, publicTransport* etc.
  - A service will have a *serviceName* corresponding to a *serviceType*
    - service of *serviceType* '*health'* can be *hospital* or *clinic*
    - service of *serviceType* '*greenArea'* can be *park* or *garden*
    - service of *serviceType* '*recreation'* can be *bar* or *theatre* etc.
  - Will have an instance of *Address*

- *UserProfile*
  - Represents the current user being profiled.
  - Will store information like age, name, budge etc.

- *Recommendation*
  - Represents a recommendation of a rental offer.
  - Will have an instance of *Residential* which is the rental offer.
  - Will have a slot to store scores
  - A slot to display recommendation characteristics.
  - A slot to display recommdation levels as *rejected, partially_adequate, adequate and very_recommendable.*

With these set of abstracted concepts and relations the Knowledge Base System will be able to recommend rental offers in AnyWhereTown to a citizen based on his/her profile.

2.3.2 Ontology – description

We started making the ontology with a TOP-DOWN development process, firstly defining the most general concepts in the domain and secondly refining and detailing every single concept and subclass.

This brought us to have a well-defined ontology which can be seen in the annexed picture named prototype1.

Working on clips, we realize that using a simpler onotology semplify the code, making all the process much faster.

So, the final version of the Ontolgy can be seen in the 2.3.3 chapter.

In this final ontology there are two main classes: Residential and Services.

Residential is the class which contain all the information about the houses available for renting.

The classes are:

CITY_ADDRESS

Coordinates

RENTAL_OFFER

RESIDENTIAL

SERVICES

USER_PROFILE

Class: CITY_ADDRESS

This class is meant to describe the address of a residential or of a service, it contains only one slot, which is used to refer to the class coordinates.

Class: COORDINATES

This class is used to implement the coordinate system. It contains two slot which are X and Y coordinate inside the city of anywheretown.

Class: RENTAL_OFFER

This class is used by the KBS to save information in run time, and show the result as last step. its slots are:

- recomendationLevel: contains the level of recomendation on the 3 possible: {partially_adequate, adequate,very_recommendable}

-recommendationCharacteristics: contain the criteria on which the recomendation level has been decided, ie it lists the near service.

-rent: monthly cost of the residential offered.

-residentialOffer: is the instance of the house. It allows to bind the residential and the offer

Class: RESIDENTIAL

This is the core class, which describe the residential available into the KBS.

its slots are:

-address: contains the instance for the class city_address

-area: contains the dimension of the house in metre square

-fullFlatForRent: this boolean value indicates if the all flat is available for renting

-furnishing: contains the state of furnishing of a residency and can be of three values: semi, full, none

-locAttribute: contains the description of the block in which the house is located, it can be:posh, happening, classy, downtown, moderate

-number: Is the number inside the street

-parkingSpace: its boolean, it indicates if the house has a box for the car

-recomendationLevel: indicates the grade and the number of constrain satisfacted by the rental offer

-rent: monthly cost of the house

-roomsForRent: is the number of available rooms

-street: contains the name of the street in anywheretown

-sunlight: Additional attribute meant to describe about the sunlight recieved by this residency, it can be: all_day, morning,afternoon, evening

-swimmingPool: indicates if the residential has a swimming pool. it increase the quality of the house

-temperatureControl: indicates if the temperature control units present in this residency. it increase the quality of the house

-totalBedrooms: contains the total amount of bedrooms availabe in the house

-type: it describes the kind of residential, if it is an house or a flat

-variant: specifies the subkind of residential's type, and can be: duplex, attic, flat, detached, villa, semi_detached, penthouse

-view: Additional attribute that describes if the house has a good view. it increase the quality of the house

Class: SERVICES

this class contains the basic services available inside anywheretown.

its slots are:

-nameOfTheService:Identifier for the Service

-serviceLocation: instances of the adress of the service

-typeOfTheService:This decribes what is the service, it can be: GreenArea, recreation, school, shops, health, publicTransport

Class: USER_PROFILE

This class is used to profile the user, and it's filled in running time with the information gathered by asking the questionto the user.

Its slots are:

-age: age of the user

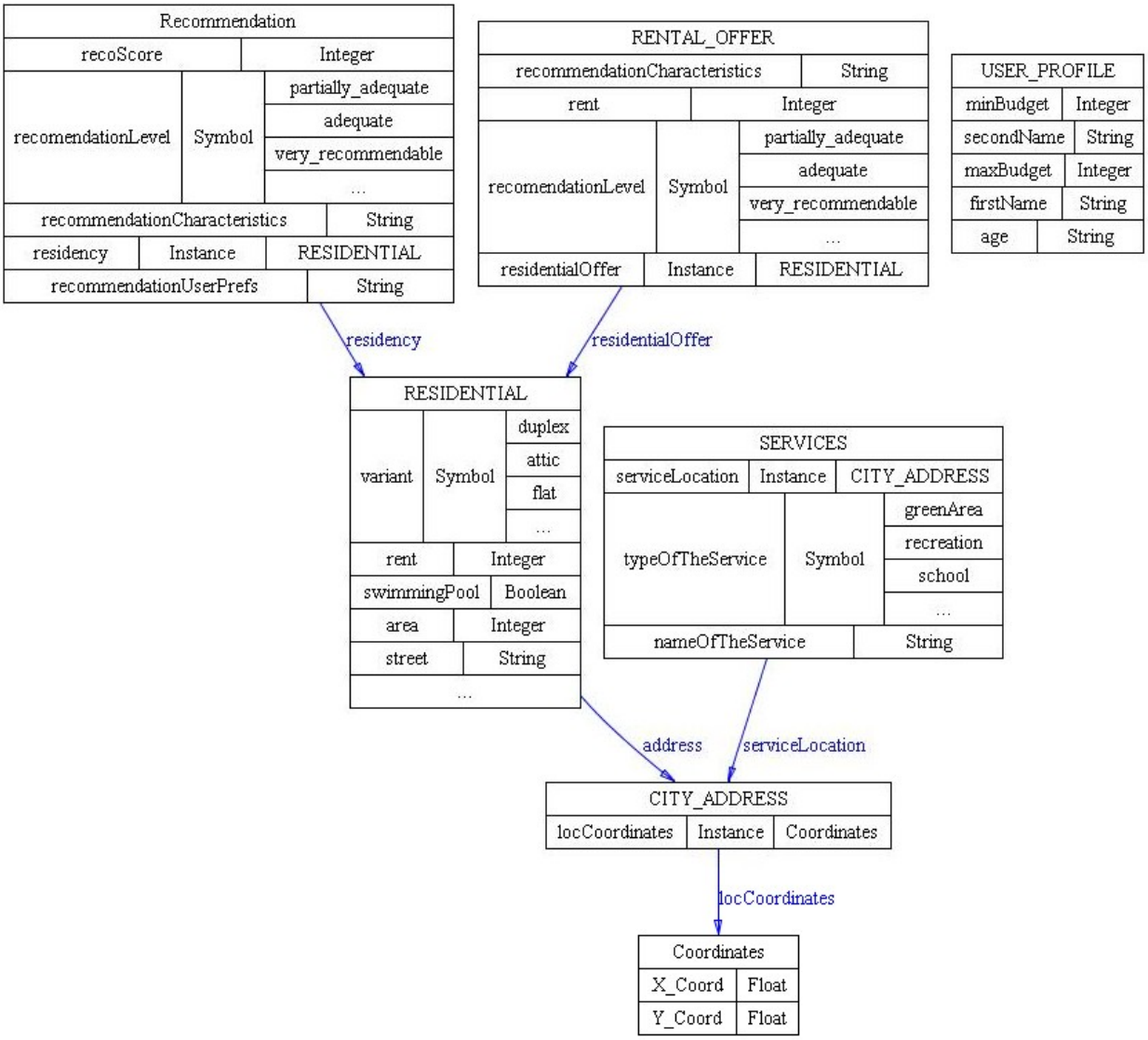
-firstName: first name of the user

-maxBudget: maximum amount of money the user would pay for renting

-minBudget: minimum amount of money the user wants to spend for renting

-secondName: surname od the user

2.3.3 Ontology – graph



2.3.4 Algorithms

The following are the algorithms used on the problems and sub-problems to reach the solution. We have created two levels of profile one simple and on advanced.
For the below kindly note that *userPreference* are facts related to his housing preferences eg. 2 rooms and *userProfile* are facts about that person itself eg. likes to swim etc.

- Profile The User (simple profiling)
  - Ask for first name
  - Ask for second name
  - Ask if user is aged above 35
    If *yes* then ask if person is aged below 55
    If *yes* then assert person is *middle-aged*
    Else assert the person is *elderly*.
    Else assert the person's age group is *young*.
  - Create an instance of a *UserProfile* with this information.

- Ask if person owns a car. Assert transport mode as *public* or *self*.
- Ask for regular weekend activities as one of (*dine*, *clubbing*, *sports*, *movie*). Assert the weekend activity.
- Ask if a person is a fan of the *AnyWhereTownClub*. Assert if person likes to watch matches in *stadium* or not.
- Ask if person shops in bulk.
  If *yes* ask if person like discounts.
  If *yes* assert person would prefer a *shoppingMarket* (as it gives discounts)
  Else assert person would prefer a *shoppingCenter* (just bulk shopping)
  Else assert person would shop in a *minimarket*.
- Ask for maximum budget in range [500-5000] including.
- Ask if this is flexible (not negotiable)
  If *yes* i.e it is flexible then increase the budget by a factor of 1.375

o Profile The User (advancedprofiling)

- Ask if person studies. (for young people)
  If *yes* which university (UPC, ESADE, IESE, UB)
  Assert that this person would want
- Ask if person is married
  If *yes* ask if person has children
  If *yes* ask what the person wants to rent house, apartment , undecided
  Assert user's preference.
  Else assert that a married person with no children is undecided.
  Else assert that an unmarried person does not want a house.
- Ask if user wants specific features in house
  If *yes*
  ask for temperature control
  ask for preference of furnishing as semi, full or none.
  ask if its important to have a good view
  ask if the user likes to swim
  assert these specific features.
  Else assert all specific featuers as false.

o Fetch Rental Offers

- *ForAll* instances of *Residential*
  Create a *Recommendation* with an instance of the i^{th} *Residential,*a *recoScore* = 0, *recommendationCharacteristics* as empty and *recommendationUserPrefs* as empty, *recommendationLevel* as empty.

o Filter Rental-Offers by User Budget

- Fetch *maxBudget* from *UserProfile*
- *ForAll* instances of *Residential*
  get i^{th} *Residential*'s *rent.*
  Check if *maxBudget* > *rent*
  If *yes* then add 10 to the i^{th} *Residential*'s *recoScore*.

Else set i^{th} *Residential*'s *recommendationLevel* to *rejected.*

- Filter Rental-Offers by Advanced Profiling
  - Fetch *userPreferences* facts asserted in system
  - *ForAll* instances of *Residential*
    Check *userPreference* fact against i^{th} *Residential*'s slots.
    Assign scores if *userPreference* is satisfied by this *Residential.*

- Filter Rental-Offers by Distance
  - Fetch *userProfile* facts asserted in system
  - *ForAll* instances of *Residential*
  - *ForAll* instances of *Services*
    get i^{th} *Residential*'s *address*
    get j^{th} *Service's address.*
    Calculate distance.
    Check *userProfile* fact against *serviceType*
    Assign scores if *serviceType* is required for a *userProfile.*
    If distance is < 500 then score +5
    If 500< distance < 1000 then score +2
    Else score +0.

- Asses Scores
  - *ForAll* instances of *Residential*
    Assess score of i^{th} *Residential*'s *recoScore*.
    If *recoScore* <15 then i^{th} *Residential*'s *recommendationLevel* is *partially_adequate.*
    Else if 15 < *recoScore* < *30* then i^{th} *Residential*'s *recommendationLevel* is *adequate*
    Else i^{th} *Residential*'s *recommendationLevel* is *very_recommendable.*.

- Give Recommendations
  - *ForAll* instances of *Residential*
    Print i^{th} *Residential*'s *recoScore, recommendationLevel, recommendationCharacteristics, recommendationUserPrefs*.

2.3.5 Justification

With the above mentioned Ontology and Algorithm the KBS will be able to give recommendations on questions that would be raised in person's mind while looking for a house.
For eg. A student who studies in UPC university would get a Recommendation for an apartment closer to UPC and closer to bars.
An elderly person would be given a recommendation of a house closer to a park and a hospital and lots of sunlight.

2.4 Implementation

2.4.1 Development Methodology

The tools used to develop this KBS are Protege and CLIPS. Protege is used to develop the Ontology and CLIPS to develop the KBS itself.
We used an iterative development methodology developing prototypes, creating an Ontology first and then trying out how the Classes and Instances defined via Protege can be accessed within CLIPS.
Also we developed a prototype Ontology using Abstracted information. For instance a *Residency* could have been an abstract class with two more abstract sub-classes *House* and *Apartment,* which would then in turn have the actual classes.
However there were certain problems in accessing the information of derived instances in CLIPS. We have left a sample of that leftover implementation in the delivered archive. Both the protégé and clips implementation have the name *prototype1*.

However we were able to solve this problem by using the slots themselves. Instead of abstracting information at a classes level, we abstracted the information using slots. So for instance a *Residency* will have a slot called *type* which will have the values *House* and *Apartment.* And it will also be required to have another slot called *variant* which can be a *villa* if the *type* is *House* or a *penthouse* if the *type* is *Apartment.*
We have used this implementation of abstraction as it helped us to modularize and iterate even faster.

2.4.2 Modularization

The CLIPS implementation has been divided into several modules and extensive use of deffunctions to encapsulate re-usable code.

- At the top level we use a profiler class to store the *userPreferences* and *userProfiling* information mentioned in the algorithms.
- *message-handlers* for the different types of object we need to print.
- *deffunction* ‘s to ask numeric and open questions and also to calculate distances.
- *(defrule startKBS* is the entry point into the expert system and brings the focus on the following modules
  - *(defmodule createUserProfile* "Generates a user Profile"

- *(defmodule kbsProfiler* "Module for Basic profiling of the user"
- *(defmodule advancedProfiler* "Module for advance Profiling of the user"
- *(defmodule factProfiler* "Module for Profiling the facts and filtering the Recommendations"
- *(defmodule output* "Module for printing the Recommendations"
- *(defmodule endKBS* "Termination MODULE"

Each Module has its own set of specific rules.

3 Testing

The map of AnyWhereTown is attached for reference to the test cases.

3.1 Coverage

3.1.1 Test Case 1

Single girl, student at ESADE looking for an apartment with a strict budget of 700 euro. Is an excercise freak.

KBS recommended a house at berlin street which is near ESADE as *very_recommendable*. Please refer to testCase1.txt for full details.

3.1.2 Test Case 2

Young married couple with a strict budget of 1200 euros and preference of a supermarket for discounts. They have a car and also would like a temperature control system in their house.

KBS recommended a house at mountain stree which has a supermarket close by and a house which has temperature control as *adequate*. For details refer to testCase2.txt.

3.1.3 Test Case 3

Old couple with a budget 1200 euro but negotiable and they like walking in gardens.

KBS recommended 3 houses as *very_recommendable.* For details please refer to testCase3.txt.

3.1.4 Test Case 4

A football player looking for a posh house for rent. Would like to practise in the stadium and has his own Ferrari. He has no price constraints.

Since the player is full of cash and doesn't mind anything most of the houses turned up as *very_recommendable.* For details please refer to testCase4.txt.

3.1.5 Test Case 5

Middle aged, executive couple. Live an active clubbing life and dine regularly. Budget of 2500 euros but negotiable. Would prefer a penthouse.

KBS recommended a few houses as *very_recommendable.* For details please refer to testCase5.txt.

3.1.6 Test Case 6

A middle-aged zoologist who likes bicycling and nature. Would prefer a flat in the range of 1200 euros. Wouldnt mind to share a flat with like minded person.

KBS recommended a few houses as *adequate*. For details please refer to testCase6.txt

3.1.7 Test Case 7

Professor who teaches at UPC. Looking for a house with his family. Budget of 1000 euros but negotiable.

KBS recommended some houses as *very_recommendable* and some as *adequate.*

3.2 Output

3.2.1 Input / System Knowledge

The quality of output that was obtained suggests that the KBS requires little more training and perhaps fine-tuning of the algorithm which works as the inference engine. In our current scenario the budget requirements are given the most weightage, and hence any person with an inflated budget would automatically make a jump of +10 points and end up with *adequate* or perhaps *very_recommendable* ratings.
We understand that by adding more concepts and relationships into the domain and interplaying with the variables at large will help make the KBS more responsive to better personalized recommendations.