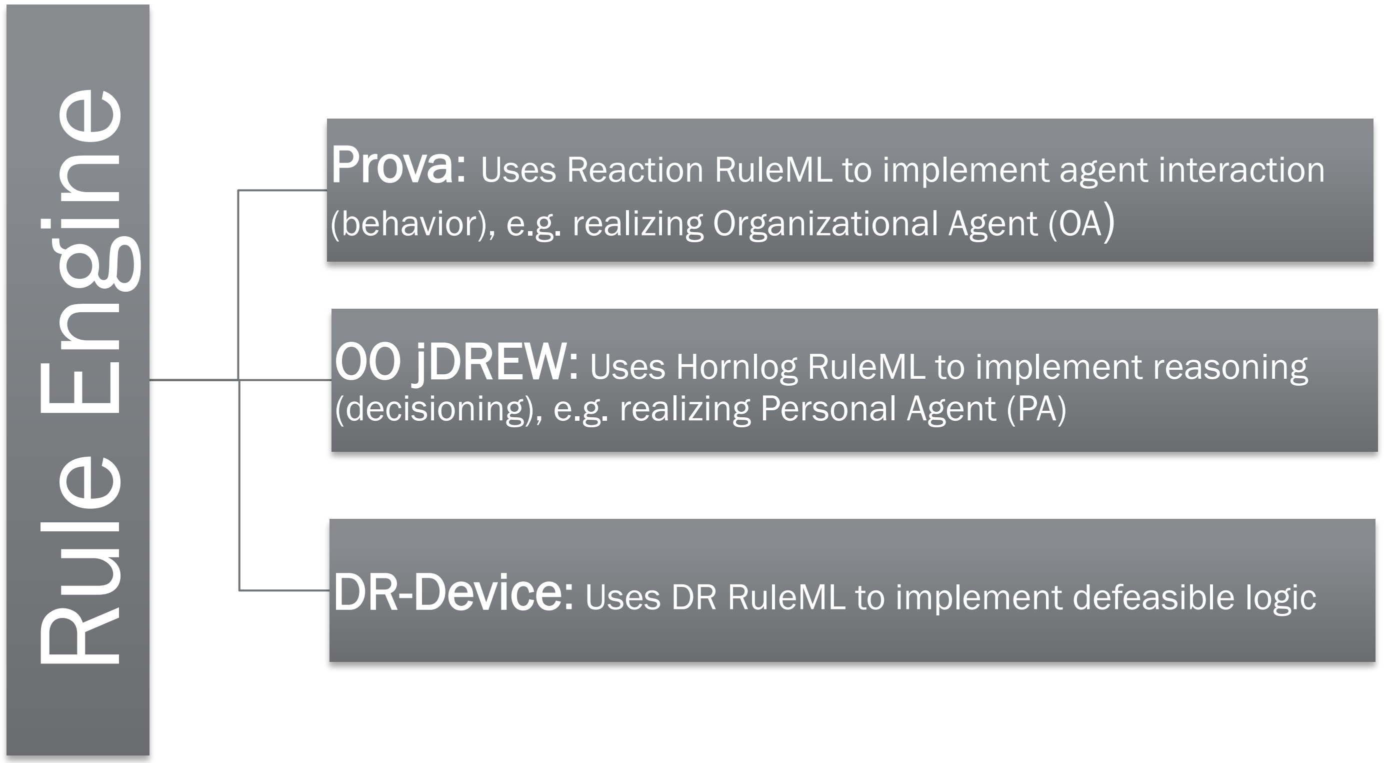
PEOPLE FRIEND ADVISOR

BASED ON INTERESTS AND DAILY ROUTINES

By: Mehdi Rohaninezhad
National University of Malaysia(UKM)

Feb 10, 2012

RULE RESPONDER RULE ENGINES



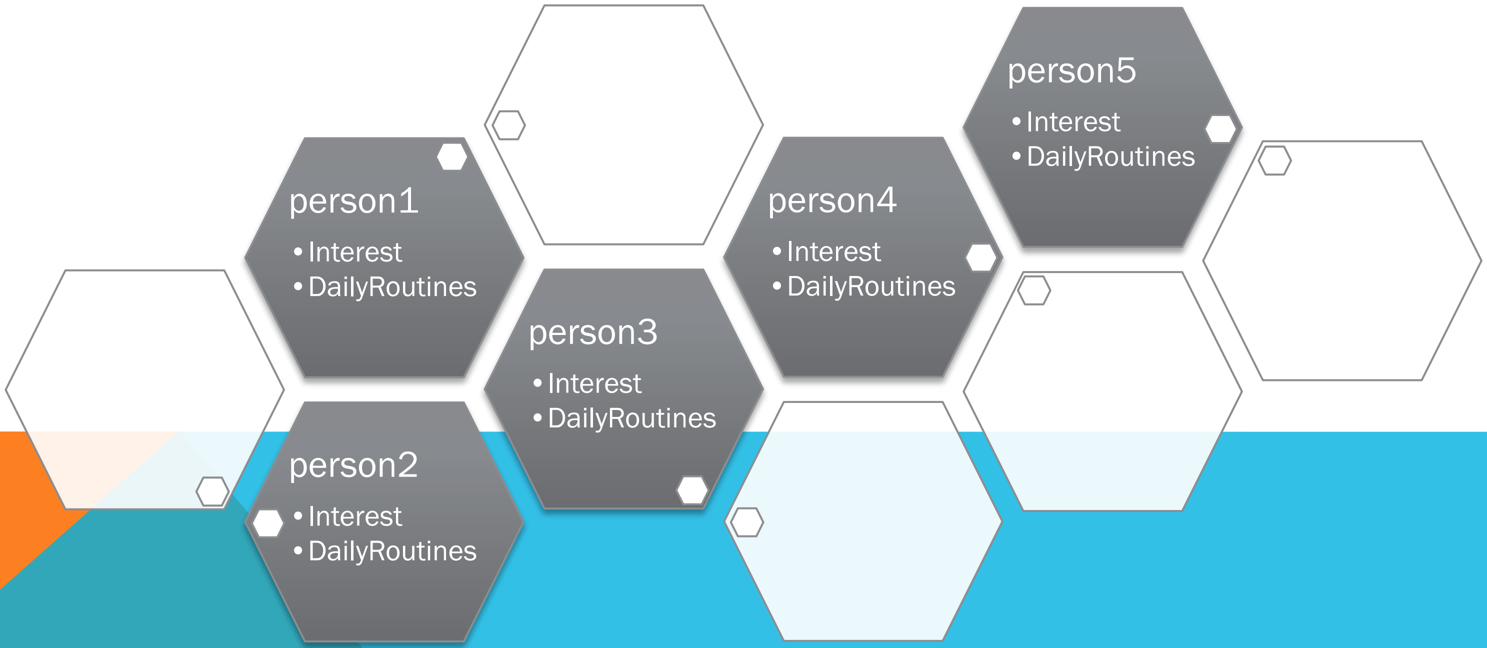
Rule Engine

- **Prova:** Uses Reaction RuleML to implement agent interaction (behavior), e.g. realizing Organizational Agent (OA)
- **OO jDREW:** Uses Hornlog RuleML to implement reasoning (decisioning), e.g. realizing Personal Agent (PA)
- **DR-Device:** Uses DR RuleML to implement defeasible logic

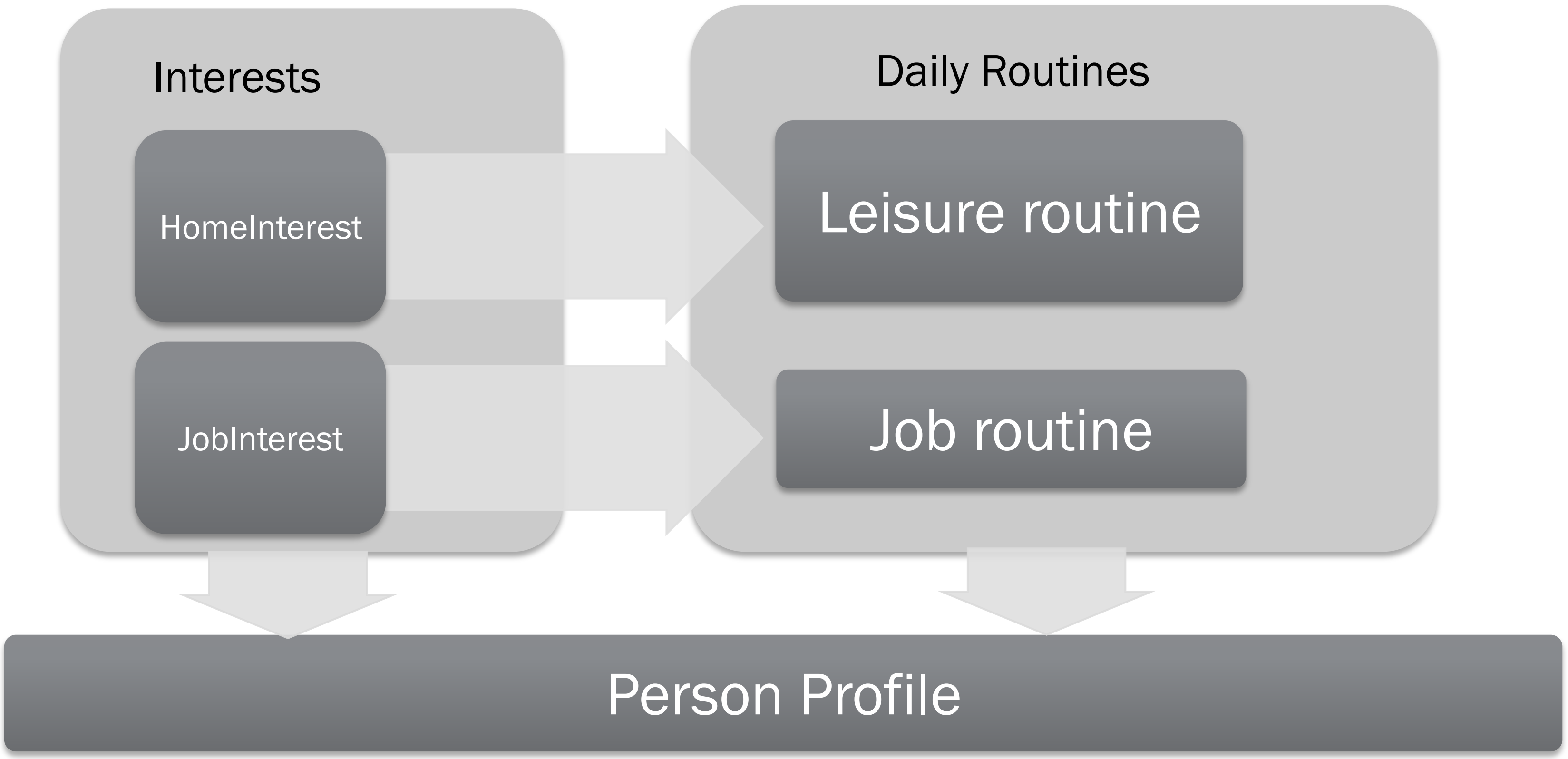
PERSON PROFILE

- Each person is a part of society. In this project persons have two main property classes, interest and daily routines.
- person’s profile:

```
        Person(John^
                name->John ;
                knows->[Mary] ;
                interest.no->4:Integer;
                address->place[country ->Canada ; city->Vancouver] ;
                phones->tel[cell->098 ; home->3435]).
```



FROM INTERESTS TO DAILY ROUTINES AND PERSON PROFILE



RELATED TO: INTERESTS, DAILY ROUTINES AND PERSONS

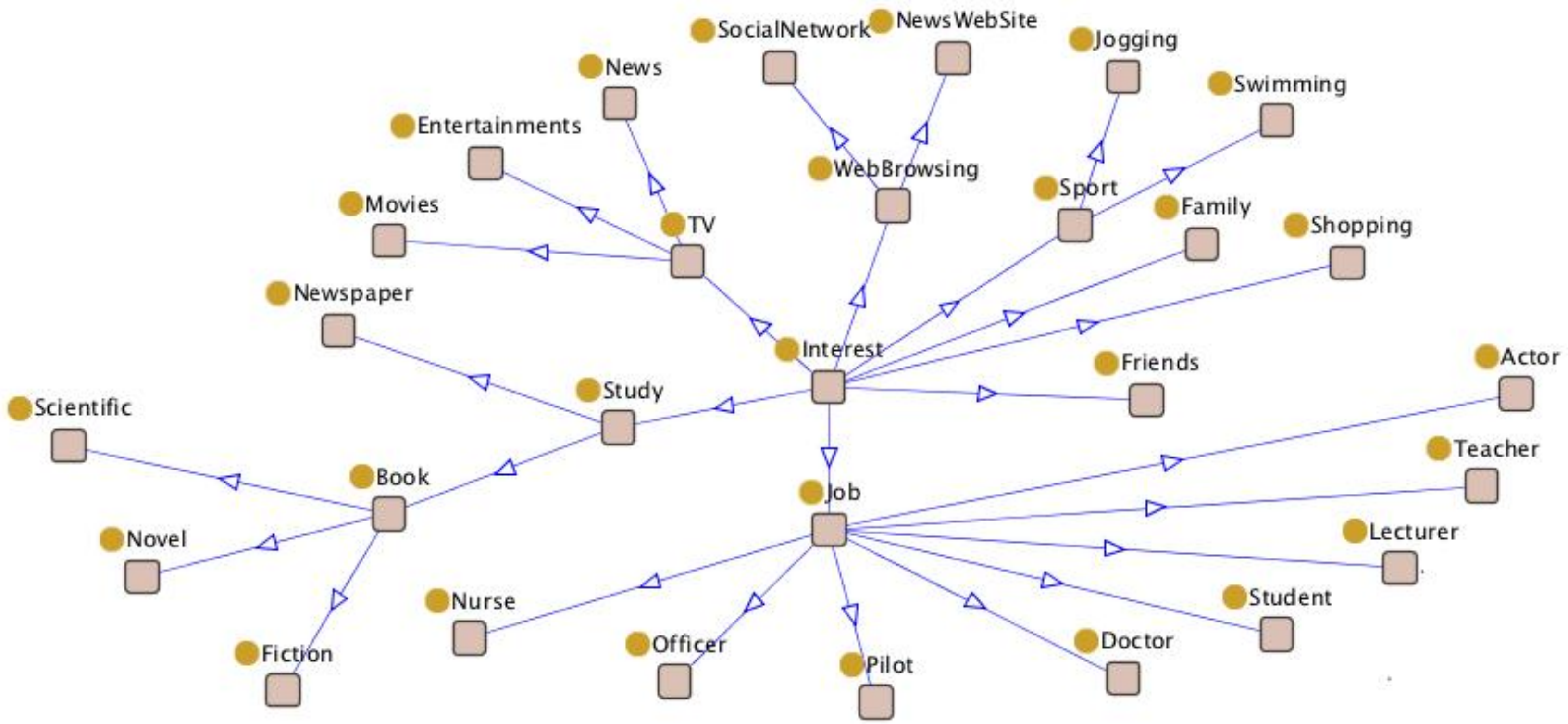
Interest, dailyRoutines and person are three main predicates in this project. They play the role of ground facts.

relatedTo is a slotted term in interest predicates, which implies a kind of priority relation between person's interests.

relatedTo is a slotted term in dailyRoutine predicates, which implies a kind of chronological order between person's daily routines.

knows is a Plex term in person predicates, which implies a list of persons who are known directly.

INTEREST TAXONOMY



INTEREST PROFILE

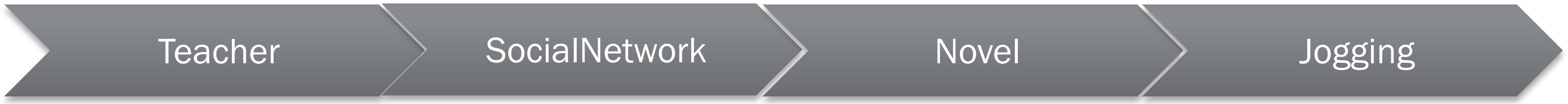
- Interests have RDF taxonomy. Person's interests are related to each other in a priority order.
- Here is an example of John's interest profile

interest(JackJob^^Teacher^ person->Jack ; class->Teacher; job->JackJobRoutine ; degree-> 1^^Integer ; relatedTo->JackFacebook^^ SocialNetwork).

interest(JackFacebook^^ SocialNetwork^ person->Jack ; class->SocialNetwork ; degree-> 2^^Integer ; relatedTo->JackNovel^^Novel).

interest(JackNovel^^Novel^ person->Jack ; degree-> 3^^Integer ;class->Novel; relatedTo->JackJogging^^Jogging).

interest(JackJogging^^Jogging^ person->Jack ; class->Jogging ; degree-> 4^^Integer).



INTEREST PROFILE RULES(1)

- Rules for generating person interest lists

1-

```
getAllInterestCategories(?p , ?allInterestCategories , ?n^^Integer) :-
   person(? ^ name->?p ; interest.no->?n^^Integer !?),
   interestCategorylist(?p, [] , ?allInterestCategories, ?n^^Integer).
```

2-

```
interestCategorylist(?p, ?visited , [] , 0^^Integer).
interestCategorylist(?p, ?visited , [ ?c| ?rest], ?n^^Integer) :-
   interest(?i^^Interest ^ person->?p ; class->?c ; degree->?dg !?),
   greaterThan(?n^^Integer , 0^^Integer),
   equal (?dg^^Integer ,?n^^Integer),
   subtract(?n1^^Integer , ?n^^Integer ,1^^Integer),
   notMember(?c , ?visited),
   interestCategorylist(?p , [ ?c| ?visited] , ?rest , ?n1^^Integer).
```

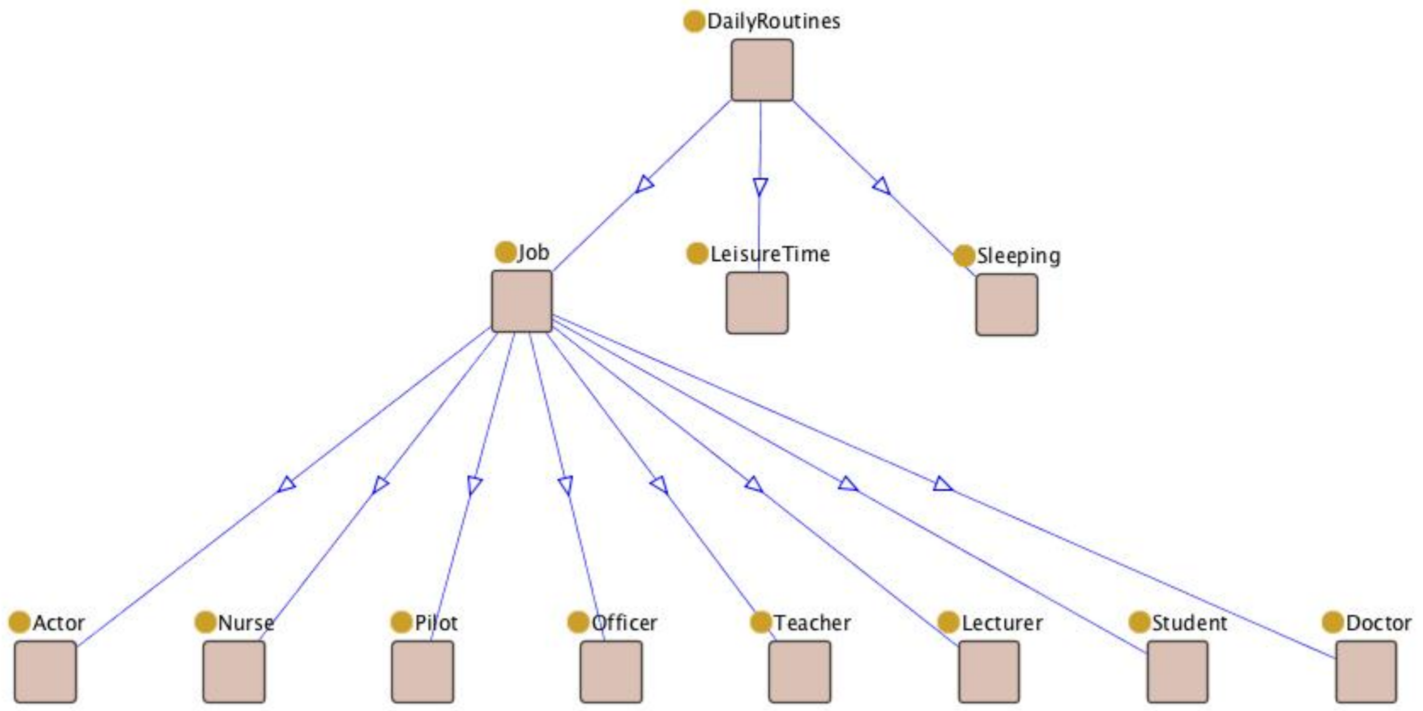
INTEREST PROFILE RULES(2)

- **Query:**

  getAllInterestCategories(Jack , ?allInterestCategories , ?n^^Integer).
- **OO jDREW TD Result :**

?allInterestCategories :	[Jogging, Novel, SocialNetwork, Teacher]
?n:	4:Integer

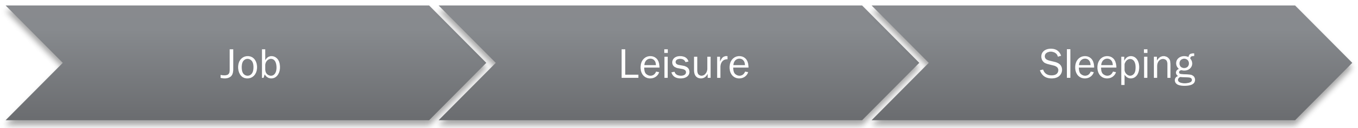
DAILY ROUTINE TAXONOMY



DAILY ROUTINE PROFILE

- Daily routines taxonomy is consisted of Job, Leisure activities and Sleeping. They are related with each other in chronological order.
- Here is the Jack's daily routines.

```
dailyRoutines(JackJobRoutine^^Teacher ^ person->Jack ; class->Teacher ;
duration->time[from->8.30^^Real ; to->16.30^^Real] ;
place->pl[UNB,Torento] ; relatedTo->JackLeisureTime^^LeisureTime).
%-------------------------------------------
dailyRoutines(JackLeisureTime^^LeisureTime ^ person->Jack ;
duration->time[from->18.30^^Real ; to->23.00^^Real];
place->pl[home , club] ; relatedTo->JackSleep^^Sleeping).
%-------------------------------------------
dailyRoutines(JackSleep^^Sleeping ^ person->Jack ;
duration->time[from->24.00^^Real ; to->7.00^^Real];
place->pl[home]).
```



ENJOY TIME(1)

- Enjoy time is a time that a person spend on his/her interests. For a person who is interested in his job , the job time also should be considered. This predicate is implemented via NAF.

```
getEnjoyTime(?p,?time) :-
    naf(getJobInterest(?p , ?i)),
    getLeisureTime(?p , ?time).
%-----------OR
getEnjoyTime(?p,?time) :-
    getJobInterest(?p , ?i),
    getLeisureTime(?p , ?time1),
    getJobTime(?p, ?time2),
    add(?time^^Real, ?time1^^Real, ?time2^^Real)
```

ENJOY TIME(2)

- Query:

  getEnjoyTime(Mary,?time)

- OO jDREW TD Result:

  ?time   13.7 : Real

WORKMATES(1)

- Another helpful predicate is used for finding workmate list.

```
getWorkmateList(?job, ?visited , [] , 0^^Integer).
getWorkmateList(?job, ?visited , [?p|?rest] , ?n^^Integer) :-
    greaterThan(?n^^Integer , 0^^Integer),
    getJob(?job , ?p),
    subtract(?n1^^Integer, ?n^^Integer ,1^^Integer),
    notMember(?p , ?visited),
    getWorkmateList(?job, [?p|?visited] , ?rest ,?n1^^Integer).
```

WORKMATES(2)

- Query:

  getWorkmateList(Teacher, [] , ?list , 2^^Integer).

- OO jDREW TD Result :

  ?list [Harry, Jack]

FRIEND ADVISOR BASED ON INTERESTS AND LEISURE TIME INTERVAL(1)

- People may be advised to make a friend based on common interests and leisure time interval. In this case their relation is not considered as an assumption. System offers a time interval for common leisure activities based on their leisure time intersection. System assign a grade for each friendship relation based on common interests(?grade).

```
friendAdviceBaseOnInterests(?p1 , ?p2, ?grade , [?lowerBound , ?upperBound]) :-
   haveIntersectionLeisureTime(?p1,?p2, ?lowerBound , ?upperBound),
   commonInterest(?p1 , ?p2 , ?grade).
```

FRIEND ADVISOR BASED ON INTERESTS AND LEISURE TIME INTERVAL(2)

- Query:

```
                    friendAdviceBaseOnInterests(John ,
                                                   ?p2, ?grade ,
                                                   [? lowerBound , ?upperBound]).
```

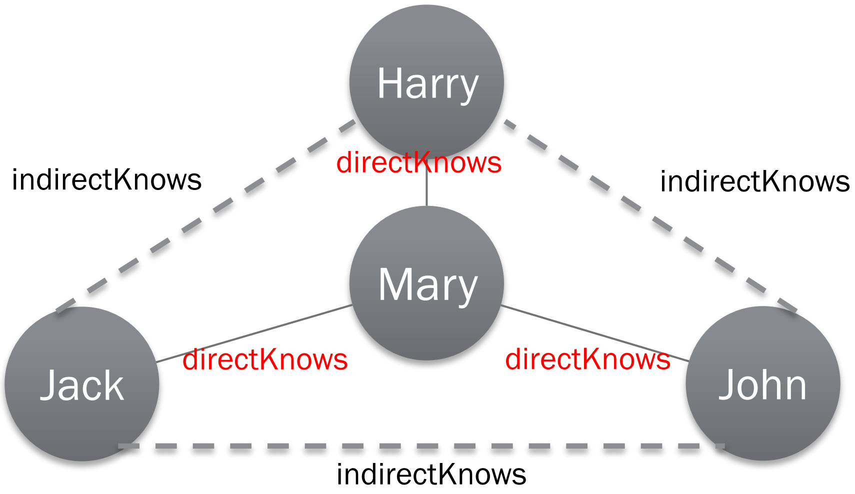
- OO jDREW TD Result :

?p: Mary ?grade: 2 : Integer ?lowerBound: 21.00 : Real ?upperBound: 22.00 : Real	?p: Jack ?grade: 3 : Integer ?lowerBound: 18:30 : Real ?upperBound: 22.00 : Real	?p: Harry ?grade: 1 : Integer ?lowerBound: 18.30 : Real ?upperBound: 22.00 : Real

KNOWS AND FRIENDSHIP

- FOAF-like knows relation is defined to be *symmetric* here. If knows(?p1,?p2) Then knows(?p2,?p1).

```
directKnows(?p1,?p2) :-
   person(?p1^ knows->?list !?),
   member(?p2,?list).
%-----------OR
directKnows(?p1,?p2) :-
   person(?p2^ knows->?list !?),
   member(?p1,?list).
```



- Person's knows relation satisfies transitivity here as in FOAF. The relation is DIRECT if they know each other or INDIRECT if there is a person who knows them directly.
- People may know each other but not necessarily have a friendship relation.

FRIEND ADVISOR BASED ON INTERESTS AND KNOWING(1)

- Two persons may know each other while they are not friends. So they may be advised to try to make friends based on their common activities. In this case person's relation is considered as an assumption (directly OR indirectly). System assign a grade for any friendship relation based on common interests(?grade).

```
friendAdvice(?p1 , ?p2, ?grade) :-
    indirectKnows(?p1,?p2),
    commonInterest(?p1 , ?p2 , ?grade).
%---------OR
friendAdvice(?p1 , ?p2, ?grade) :-
    directKnows(?p1,?p2),
    commonInterest(?p1 , ?p2 , ?grade).
```

FRIEND ADVISOR BASED ON INTERESTS AND KNOWING(2)

- Query:

  friendAdvice(John , ?p2, ?grade).

- OO jDREW TD Result :

```
?p2                 Mary   ?grade: 2
?p2                 Jack   ?grade: 3
?p2                 Harry  ?grade: 1
```

DEFEASIBLE REASONING(1)

- One of other crucial reasoning technique is defeasible reasoning which is a reasoning for typical consideration.
- POSL is extending towards d-POSL.

  See also: Fall 2011 Project Team 1 (http://www.cs.unb.ca/~boley/cs6795swt/fall2011projects.html).

  From POSL to d-POSL: Making the Positional-Slotted Language Defeasible: http://d-posl.wikispaces.com/
- Because of the fact that typically we do not have common interests with the most of persons whom we know, project could be extended towards defeasible reasoning.
- At the end of the day d-POSL extension of current POSL grammar is offered.

DEFEASIBLE REASONING(2)

- **Scenario:** Between *people* of *whom we know, only a small set we have common interests and activities. Most of them are our friends.*

1- strict rule:

r1: friend(?x , ?y) :- knows(?x ,?y).

2- defeasible rules:

r2: knows(?x , ?y) := ~commonInterest(?x , ?y).

r3: friend(?x , ?y) := commonInterest(?x , ?y).

3- superiority relation: r3>r2

CHANGING D-POSL GRAMMER(1)

- The grammar offered by Team1 only cover slotted terms and should be deployed for positional and positional-slotted predicates as well.
- Some parts of DPosNew.g grammar's rule is changed to accept positional-slotted atoms as well. Expr, slot rules are modified and pos is added:

```
expr:
   ( poses (SEMI slots)?
    |
   slots (SEMI poses)? (SEMI slots)?
   )+
   {System.out.println("Expression: " + $expr.text + "\n");}
   ;
```

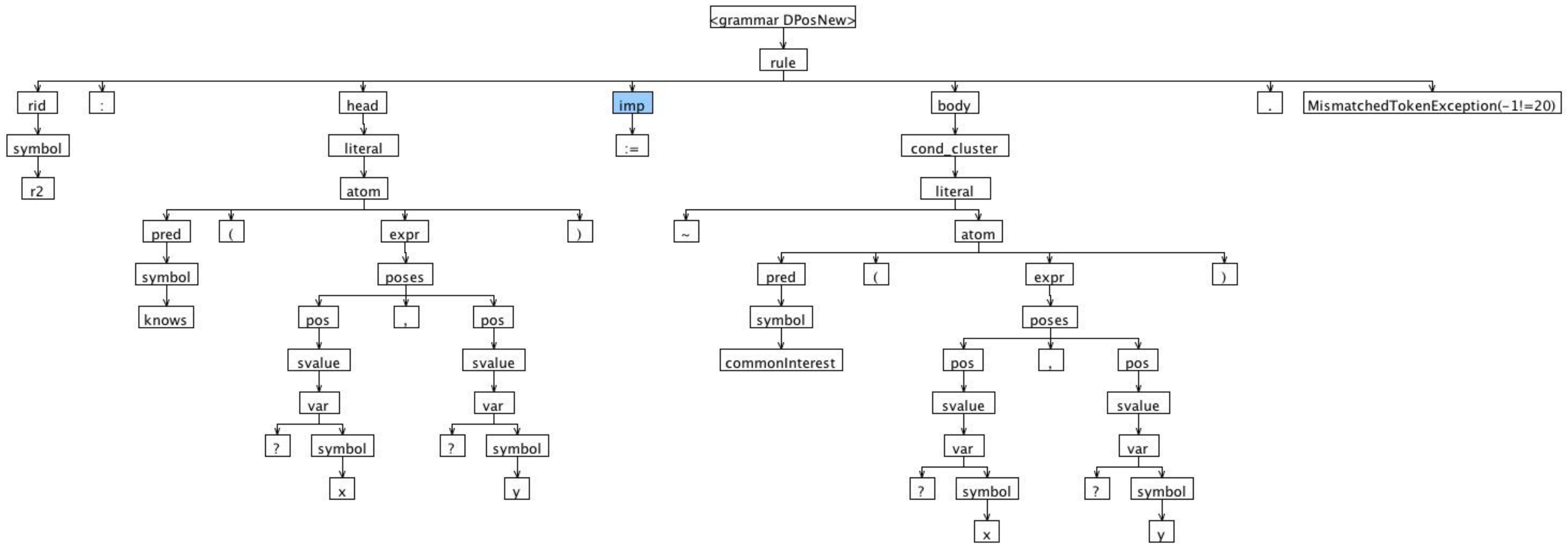
CHANGING D-POSL GRAMMER(2)

```
slots:
    slot (SEMI slot)*
    ;
slot:
   (sname ARROW svalue)
     | sname COLON sname ARROW svalue
     | sname HAT sname COLON sname ARROW svalue
   {System.out.println("Slot: " + $slot.text + "\n");}
    ;
```

CHANGING D-POSL GRAMMER(3)

```
poses:
       pos (COMMA pos)*
       ;
 pos:
      svalue
      | sname COLON svalue
      | sname HAT sname COLON svalue
     {System.out.println("Position: " + $pos.text + "\n");}
     ;
```

PARSE TREE OF SAMPLE INTERPRETATION



TEST RIG OF SAMPLE 1(TEAM1)

```
Rule ID: r1
SName apartment
Variable: ?x^^ACCOMMODATION
Expression: apartment->?x^^ACCOMMODATION
Atom: acceptable(apartment->?x^^ACCOMMODATION)
Literal: acceptable(apartment->?x^^ACCOMMODATION)
Literal: acceptable(apartment->?x^^ACCOMMODATION)
:
(
Predicate: carlo:apartment
^
:
SName apt845
SName carlo
SName age
)
Slot: apt845^carlo:age->12
Expression: apt845^carlo:age->12
.
Atom: carlo:apartment(apt845^carlo:age->12)
Literal: carlo:apartment(apt845^carlo:age->12)
Condition Cluster: carlo:apartment(apt845^carlo:age->12)
Body: carlo:apartment(apt845^carlo:age->12)
```

TEST RIG OF SAMPLE 2

```
Rule ID: r2
Expression: ?x,?y
Atom: knows(?x,?y)
Literal: knows(?x,?y)
Literal: knows(?x,?y)
(
line 1:47 mismatched input '<EOF>' expecting NEWLINE
?
,
?
)
Expression: ?x,?y
.
Atom: commonInterest(?x,?y)
Literal: ~commonInterest(?x,?y)
Condition Cluster: ~commonInterest(?x,?y)
Body: ~commonInterest(?x,?y)
```

CONCLUSION

- The idea of this project can be considered as a reasoning component for a PA in a virtual organization.
- Defeasible reasoning extension of the sample helps PA to make difference between knowing and friendship concepts based on interest. Knowing concept might be considered as just a relation, but common interests, daily routines, gender and job prepare the ground for friendship.
- Thanks to Harold Boley and RuleML colleagues for feedback on this project.

THANKS A LOT FOR YOUR ATTENTION