Groove-X:
The Party Dance Advisor

Presented by: Edgar J. Terrero
6.871 Knowledge-Based Applications Systems
Thursday, May 12, 2005
Prof. Randall Davis

ABSTRACT

This paper presents Groove-X, a knowledge-based expert system that helps its user decide whether to dance with a girl at a party, and if so how to manipulate his dance moves such that he gets to dance with her the way he wants to without making her uncomfortable. In particular, this paper discusses four aspects of the system: the task it performs, the problem-solving paradigm involved, the knowledge contained in the system, and the lessons we learned from designing and building the system. Moreover, this paper briefly points the reader in the direction for future work on the system.

Introduction

Knowledge-based expert systems have seen very successful consulting applications in several fields, including medicine, can selection to contain food, mineral deposit finding, and even law [1,2,3,4]. Now, consider the situation in which you are a young, single man, walking into a party featuring urban music. You came to have a good time by dancing some songs with an attractive woman. Now this attractive woman is consistently looking in your direction. Several questions arise: "Should I approach her? Am I handsome enough and dressed well enough to make a good impression on her? What will her friends say or do?" Humans are very indecisive, and these can turn from simple questions into headaches very quickly.

Groove-X is a knowledge-based expert system designed to help you make that decision. Combining several factors, including how you are dressed, the party settings, and the girl's attitude, Groove-X tells you how to interact with the girl: by dancing with her, by speaking to her before dancing, or by ignoring her and trying to find another girl.

The System Task

Definition of the Task

The general task of the system consists of two parts. First, given a set of factors regarding the user's appearance and behavior, the party setting, and the girl's mood and location at the time, the program will advise the user on whether he should dance with the girl, just talk to her before dancing, or just ignore her and try another girl. Second, should the system advice the user to dance with the girl, it will ask about his current dancing position and will advice him on how to manipulate dance moves such that he dances with her the way he wants to, without offending her.

A specific example will shed more light on the task of the system. Consider John, who happens to go to a fraternity party with some friends. John is dressed well; that is, he is wearing a fitted baseball cap that matches his shirt and sneakers in color. His shirt is also of a popular brand of urban clothing (Rocawear) which happens to match the brand of his jeans. His clothes are clean and ironed, and he has showered and worn cologne, which adds to his attractiveness. Moreover, he is going with a few friends who all happened to be well dressed, which adds to his confidence. Once they arrive at the party, it is the beginning of the party and the lighting is dim. Loud music is playing, and people are just standing while drinking alcohol. He spots the girl he wants to dance with, and they make eye contact; moreover, her friends are not around. She is close to the wall, and there are not that many people around her. For these settings, the system strongly (0.65777) suggests that he should approach the girl and start a conversation with her.

The example below illustrates an actual instance of a user interacting with the system (bold font signifies the author's comments):

Sample Problem

```
 (ask [dance-with-girl edgar ?x] #'print-answer-with-certainty)
                                         {This is the highest-level goal}
Is it the case that EDGAR wants to dance without talking to the girl: No
            {The system will not advice talking if the user does not want to}
Is it the case that the girl is drinking: Yes

What is the party’s lighting level: Dim
            {Some physical traits are not noticeable depending on lighting}
What is EDGAR's hair condition (in terms of how clean it is): I-Dont-Know
            {The user can answer “I don’t know,” in which case the program
            will figure out the answer by asking more detailed questions}
Is it the case that EDGAR has dandruff in his hair: No

Is it the case that EDGAR has a hair shapeup: Yes

Is it the case that EDGAR has braids: No
            {Physical appearance is important, since it affects how the girl
            thinks about the user}
Is it the case that EDGAR is clean shaved: Yes

Is it the case that EDGAR has a shaped-up mustache: Yes

What is EDGAR's acne status: Absent

Is it the case that EDGAR's teeth are yellow: No

Is it the case that EDGAR's teeth are clean: Yes

Is it the case that EDGAR's teeth are missing: No

Is it the case that EDGAR's teeth are straight: Yes

What is EDGAR's shirt look like (that is, how good it looks): Fresh

What is EDGAR pants look like (in terms of how clean they are): Fresh

What is EDGAR's pants type: Jeans

What is EDGAR's shoes brand: Nike

What is EDGAR's shirt match (has a similar color to): Shoes

Is it the case that EDGAR is wearing a dress shirt: No

What is EDGAR's shirt brand name: Rocawear

Is it the case that EDGAR’s pants brand matches his shirt brand: Yes

What is EDGAR's shoes crisp (stylish) status:
```

```
We are trying to determine whether EDGAR's shoes crisp (stylish) status is
FRESH
This is being asked for by the rule SHOE-CRISP-USER-INPUT in order to
determine:
whether EDGAR is wearing crisp (stylish) shoes
You are being asked to enter one of Fresh, Unkempt, or I-Dont-Know.
The possible completions are:
Fresh
Unkempt
I-Dont-Know
```

{This is just Joshua's "WHY" explanation mechanism}

```
Fresh

What is EDGAR's amount of friends: Few

Is it the case that EDGAR's friends are crisp (stylish): Yes

Is it the case that EDGAR has washed: Yes
```

{Hygiene is equally as important as physical appearance}

```
Is it the case that EDGAR is wearing cologne: Yes

Is it the case that EDGAR's breath smells : No

What is EDGAR's dance experience: Expert
```

{A good dancer will make a good impression on the girl}

```
Is it the case that EDGAR is dancing at the moment: Yes

Is it the case that EDGAR is approaching the girl right now: No

Is it the case that EDGAR started a conversation with the girl: No

Is it the case that EDGAR is looking at the girl up and down: No
```

{Looking a girl up and down can be offensive to her}

```
Is it the case that EDGAR makes eye contact with the girl: Yes

What is EDGAR's eye contact length: Short

Is it the case that EDGAR shows interest in the girl's conversation: Yes
```

{This is dumb: above, we just told the program that we have not started a conversation with the girl, but still asks this}

```
Is it the case that EDGAR is drinking: Yes

Is it the case that EDGAR is cheesing (smiling goofily): No

What is the party phase (time): I-Dont-Know
```

{Do not know? Fine, we can help you figure it out}

```
What is EDGAR's party population: Crowded

Is it the case that EDGAR party has people sweating: Yes

Is it the case that EDGAR people are putting on their jackets: No

What is the people doing at the party: Dancing
What is the level of trash in the floor: None

What is the the volume of the music at the party: Loud

What is the party song speed: Slow

Is it the case that EDGAR sees the girl's friends around here: No

What is the location of the girl: On-Wall

[DANCE-WITH-GIRL EDGAR YES] 0.99286675
```
{The system strongly suggests the user to dance with the girl}

```
(ask [dance-way-you-want edgar ?x] #'print-answer-with-certainty)
```
{Since you can dance with her, we can now proceed to tell you how to dance with her. This is done by asking a series of questions regarding the user's current dance state and his desired dance state}

```
What is EDGAR's hand position on the girl:  Hands-On-Her-Hips

What hand contact does EDGAR want: Hands-On-Her-Hips

What is EDGAR arm position on the girl:

We are trying to determine whether EDGAR arm position on the girl is ARMS-
AROUND
-WAIST
This is being asked for by the rule USER-WANTS-ARM in order to determine:
whether EDGAR can dance the way he wants
It remains to determine whether EDGAR arm position on the girl is
You are being asked to enter one of Arm-Around-Waist, Arm-Around-Chest, or
Not-O
n-Her.
The possible completions are:
Arm-Around-Waist
Arm-Around-Chest
Not-On-Her

Not-On-Her

What arm contact does EDGAR want: Arm-Around-Chest
```
{This part of the system did not quite work the way we wanted it to, but the idea is there. The original intent of this part was for the user to provide his current dancing position and his desired one; we would then print out a series of steps indicating how to get to his desired state}.

Appropriateness of the Task

This task is well suited for a knowledge-based system. In particular, it passes the "phone interview" requirement; that is, the user can ask for advice from an expert over the phone, where the expert would then ask the user a similar set of questions to those that our system asks. Although much of the expertise in our system may seem as "common sense," it is in fact highly specialized. That is because the system deals with urban parties in an urban setting. This setting lends itself to a particular set of beliefs, including fashion, taste of women, party settings, among others. Moreover, this system consists of a decently large continuously changing body of knowledge. Finally, we were able to express knowledge in this field in a simple way (more on this below). Thus, all of these factors facilitated the building of our expert knowledge system.

The Range of Problems of the System

Problems that the System Can Handle

The system can handle a wide variety of problems. These problems have a set of characteristics that fall into three categories: problems regarding the user, problems regarding the party settings, and problems regarding the girl in question. In particular, it covers any type of party in a college setting where urban music is being played; for example, it covers fraternities, dormitory parties, and clubs. It handles situations where the party has different styles of lighting, different amounts of attendants, and different music volumes.

With respect to the user (the guy), it covers both urban fashion and regular fashion; that is, the system knows about how to best dress when the user is wearing sneakers and jeans, but also knows equally as well how to best dress when the user is wearing a button-down shirt with slacks and dress shoes. It covers cases where the user goes to the party alone or accompanied, and whether he is drinking or not drinking alcohol.

Moreover, the system can cover a wide array of problems with respect to the girl. For example, it can deal with situations where the girl is with female friends or alone; it covers situations where the girl is drinking or not drinking alcohol; and it covers several possible attitudes that the girl may be experiencing at the moment, such as anger, happiness, among others. The system, therefore, uses a combination of these properties to solve a wide array of problems. The user can present the system with any scenario containing any of the above features, and the system will provide a good prediction of what the user should do based on its knowledge.

Problems the System Cannot Handle

Simply put, the system cannot handle any problems that involve situations it does not know about. For example, it cannot advise a person going to a rock party or to a party that is not in a college setting, attended by college students. Furthermore, it does not know how what to do if the girl brings male friends; the system cannot assume this situation is similar to if the girl were with her female friends, because when a girl is accompanied by male friends her attitude towards unknown guys change.

A few examples will illustrate this idea in a clearer manner. Assume, for example, that John has all of the positive characteristics listed in the first example of this paper: he is well-dressed, neat, is relatively handsome, and is confident in speaking to girls. Moreover, the party settings are right and the girl is by herself. In this situation, the party will recommend John to approach the girl and either speak to her or dance with her, depending on whether a good song is playing and they are in the middle of the party. However, if she has a boyfriend that John does not know about, then John will most probably not get a dance with the girl.

Moreover, if another user, Albert, does not have many positive physical characteristics and the girl has many friends standing in a circle, the system will advise him to not dance with the girl. However, if Albert knows the girl (i.e. he has spoken to her before), then he *does* have a good chance of dancing with her. (Reasons for us not tackling the "friend" problem, among others, are listed in the following sections).

Lastly, our system is not a "talk advisor"—it does not tell you what to say to a girl. If you have a good chance of dancing with her and the program recommends you to have a conversation with her before dancing, the program is not responsible for what you say to her.

Knowledge of the System

What Does the System Know?

The system contains information regarding what aspects of a guy's appearance and actions, party settings, and girl mood will increase or decrease the guy's chances of dancing with the girl at a party. Thus, the knowledge divides itself into three categories, as described in the section above: aspects relating to the guy, aspects relating to the party, and aspects relating to the girl. The details on the knowledge contained in these groups can be seen in the section "The Range of Problems of the System" above. Moreover, the system knows about dancing; that is, given a current dance configuration (such as front-to-front), the system knows how to arrive at any other dance configuration in a manner that will be least offensive to the girl.

This knowledge takes the form of about 160 production rules. These rules contain an antecedent and a consequent, which take the form of a boolean statement, where each statement consists of an attribute-object-value triple. If the consequent is satisfied (i.e. if it is true), then the consequent will be true. Moreover, because most of the knowledge contained in the system is probabilistic, each rule contains a certainty factor. I will now illustrate this concept of rules with two examples of actual rules that can be found in the system:

```
1. (defrule beer-effect-guy (:backward :certainty 0.5 :importance 234)
     if [is-user-drinking ?guy yes]
     then [attitude ?guy confident])

2. (defrule dancing-well (:backward :certainty 0.4  :importance 252)
     if [dance-experience ?guy expert]
     then [make-impression ?guy good])
```

In plain English, these rules state:

1. *If* the guy is drinking beer, *then* with certainty 0.5, his attitude is *probably* confident. That is, when a guy drinks beer, he loosens up and becomes more confident.
2. *If* the guy has a lot of dance experience, and he is dancing by himself, *then* with certainty 0.5, he *probably* makes a good impression (on the girl)

With these rule samples, we can see how the knowledge is encoded. For the first rule, for example, the antecedent is "if the user is drinking", the consequent is "then the attitude is confident" and the certainty factor is 0.4. From here, we can also see from the antecedent's components: the object is the guy (the user interacting with the system, captured by the variable `?guy`), the attribute is `is-user-drinking`, and the value is `yes`. The same applies for the other boolean statements. The `backward` and `importance` keywords are part of the system's reasoning mechanism, and will be explained below.

Thus, rules relate attributes to other attributes in a causal manner. The knowledge base can be viewed as a tree structure with the overall goal—to dance with the girl—at the top (the arrows mean "is inferred by"):

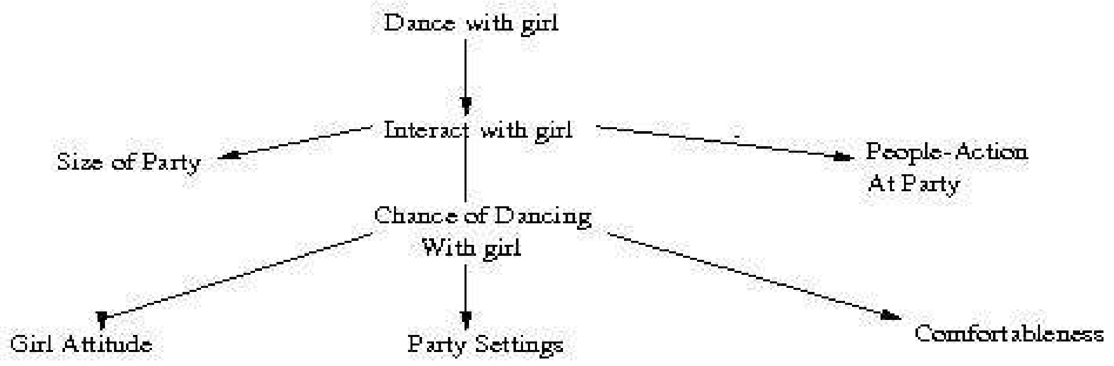

Figure 1. The top of the knowledge tree, with the main goal as the root node

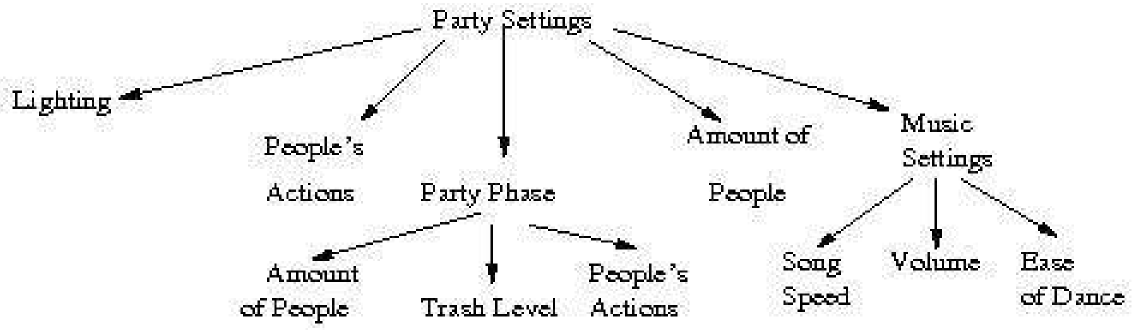

Figure 2. The knowledge sub-tree for "Party Settings"

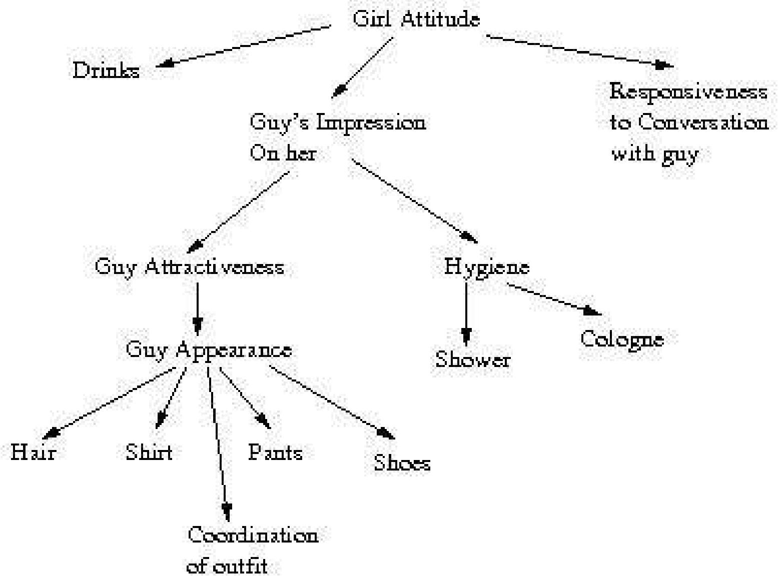

Figure 3. The knowledge subtree for "girl attitude."

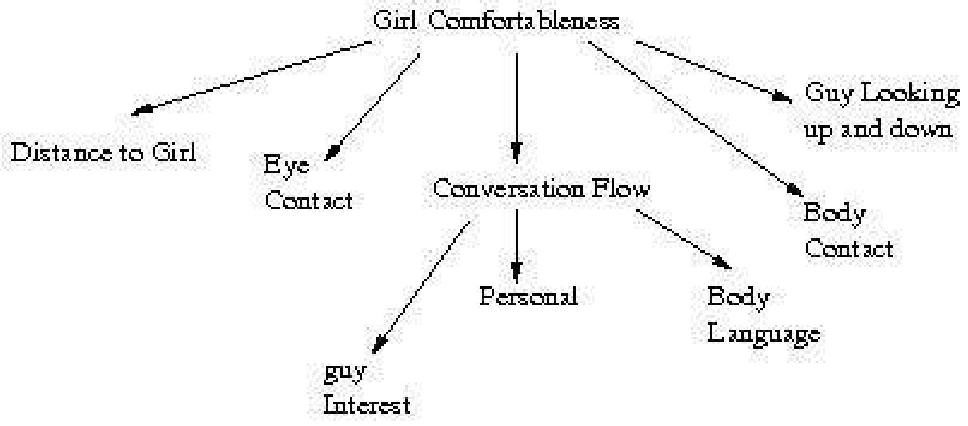

Figure 4. The knowledge sub-tree for "Girl's Comfortableness"

Justification for Production Rules as a Representation Technology

Production rules provided a convenient way to represent the system's knowledge in several ways. The most important reason to use rules was that during the interviews, the experts expressed their knowledge in a rule-like manner. For example, one of Marc's statements said, "If you show you have got moves on the dance floor, [then] the girl is most likely going to be impressed." It is easy to recognize this as the expert's expression of rule 2 above. Therefore, it was relatively simple to translate expert knowledge into rules. Since rules are supposed to be independent, single-step inferential associations they match very well the knowledge in the field [1].

Moreover, the probabilistic nature of the knowledge was conveniently encoded in rules in the form of certainties. This is a domain where nothing is definitely certain—that is, sometimes a girl may not like you for a random reason, no matter how handsome and well-dressed you are. Therefore, certainties facilitated the probabilistic reasoning that frames or cases could not provide.

How Does the System Know It?

An equally important aspect of the system's knowledge is how the knowledge was retrieved. This involves a detailed description of the knowledge acquisition process, which consisted primarily of what N.J. Cooke calls "unstructured interviews," "active participation" and "structured participation" [5]. Initially, most of the knowledge was retrieved via unstructured interviews, in which we conducted a series of interviews with experts in the field. These interviews did not have a predetermined format, but instead the questions were asked as the knowledge engineer and the expert saw fit. We divided the interviews such that I interviewed the male experts, and James Tolbert interviewed the female experts, thus minimizing the possible bias that may arise from interviewing just one gender group. The male experts consisted of three college students: Marc Wilson and Oliver De La Cruz, both who attend MIT; and Jean Terrero. I chose these male experts because I have known them for a long time, and having been with them to a wide variety of parties, I concluded that they all had excellent heuristics for determining when to dance with a girl or not.

The participation portion of the knowledge acquisition process came from several parties that I attended, including fraternity parties (Nu Delta at MIT) and clubs (The Roxy). I observed by focusing my attention on one particular individual for a long part of the night, and noting how this guy was dressed, and how and when he approached the girl he wanted to dance with. These individuals were not aware they were being noticed, thus making their actions more natural.

After collecting this knowledge, we analyzed how similar our knowledge was, since we collected it individually. To our surprise, much of the knowledge was the same: that is, the male experts overwhelmingly agreed with the female experts. These results contrasts with other fields, where experts usually disagree on what the heuristic to solve a particular problem should be.

How Does it Work?

The system uses backward chaining as its inference mechanism. Given a top-level goal (an action to achieve), the system produces a depth-first search tree, where the root node is the goal and its children are rules for which the consequent is some form of the goal. These nodes, in turn, have antecedents that are needed to prove the goal, and can themselves become sub-goals, thus generating recursive depth-first search trees. For example, a guy's interaction with the girl can be affected by the impression he makes on her. The impression he makes, in turn, can be affected by how he acts on the dance floor and how he's dressed. How he acts, in turn, can be affected by how much alcohol he has drunk, how many friends he is with at the party, and other factors; and so on.

Thus, these sub-goals take on generalized forms. For example, if the user of the system is trying to find out what his interaction with the girl should be, then the system will create a sub-goal which is [interact-with-girl user ?how] instead of [interact-with-girl user dance-with-her]. That is, it is trying to find out how the user should interact with the girl, instead of specifically trying to find out whether he should dance with her.

When a node in the tree has an antecedent that has nothing else that implies it, then the program "bottoms out." That is, if the node takes the form of a rule whose antecedent cannot be inferred because no other rule in the system has that clause as a consequent, that node will become a leaf in the tree. When the depth-first search reaches a leaf, it attains the boolean value (true or false) of the clause by asking the user at the prompt. To continue with our example, a guy's actions can be affected by how much he has drunk and how many friends he is with at the party. Both of these cannot be inferred from anything else, so our system will ask the user how many drinks he has had and how many friends he is at the party with.

Depth-first search goes top-to-bottom, left-to-right, giving the highest priority to the leftmost nodes. Given a sub-goal clause that generates multiple children in the tree, how does the sub-tree get formed? That is, how does the system arrange child nodes from left to right? When a rule has an antecedent that is present in multiple other rules as a consequent, the mechanism searches for the rules in order of *decreasing importance*, effectively considering rules with higher importance first. To illustrate with an example, consider the following three rules:

```
(defrule party-phase-dance-great  (:backward :certainty 0.5  :importance 77)
 if [and [is-party-phase ?guy middle]
         [people-action-party ?guy dancing]
         [size-of-party ?guy crowded]]
 then [chance-of-dancing-with-girl ?guy yes])

(defrule time-close-to-beginning  (:backward :certainty 0.6  :importance 139)
  if [and [user-time-input ?guy i-dont-know]
          [size-of-party ?guy not-crowded]]
  then [is-party-phase ?guy beginning])
```

```
(defrule people-are-sweaty (:backward :certainty 0.5  :importance 134)
  if [and [user-time-input ?guy i-dont-know]
          [people-sweaty-party ?guy yes]]
  then [is-party-phase ?guy end])
```

The first rule needs to know what the phase of the party is. If it is the middle of the party and the party is crowded with people dancing, then the guy has a good chance of dancing with the girl, because most probably she is in a mood to dance. That rule, therefore, sets up a sub-goal tree to find out what the phase of the party is. Then it finds rules 2 and 3, which have the consequent that proves the sub-goal clause. Because rule 2 has a higher importance than rule 3, rule 2 will fire first (that is, it is a branch more to the left of rule 3's branch in the search tree).

Once all of the leaves in the tree have clauses with assigned boolean values, the search stops and goes back up the tree, assigning boolean values to the higher-level sub-goals, until it assigns a value to the original goal. At that point, the program returns the value of the original goal to the user, and the user can either be satisfied with the solution or complain to the program by using Joshua's explanation mechanism to find out HOW the system arrived at the solution.

What I Learned from Building and Designing the System

The Good

The system performed comparably well with our expectations. After we finished the before-dance part of the program—the part of the program that tells the user how he should interact with the girl—we had some of our experts and non-expert friends try the program out. They were overwhelmingly pleased with the flow of the questions, found the program entertaining, and were satisfied with the results. For example, they liked how the program asked questions in a logical manner. If the questions were about the guy's appearance, it would ask them in an anatomically top-down manner, starting with the head, down the face, and so forth until the program finished asking questions about the guy's shoes.

Production rules provided us with an extremely powerful knowledge representation technology. As stated above, rules allowed us to convert expert knowledge easily into modularized chunks of knowledge with the right primitives—objects, attributes, and values. In fact, the way rules restricted us to express knowledge in terms of these primitives made us think about the knowledge in ways we had not thought about it. It empowered us with a way to express relationships between individuals and things in our knowledge base. Moreover, rules allowed us to explicitly express the probabilistic nature of the problem domain through the use of certainty factors.

The use of Joshua as an expert system tool made development of the program easier in some ways. For example, much of the code we used for the program was a subset of the Problem Set 2 code. This abstraction allowed us to really concentrate on what the system *knows*. Throughout the project, we spent about two-thirds of our total input time just engineering the knowledge. Moreover, Joshua's built-in HOW and WHY mechanisms allowed us to trace what kind of reasoning the system was using at different stages of the program. Joshua also facilitated the use of backward chaining as a reasoning mechanism.

The Bad

Given all of the features we initially wanted our program to have, our biggest obstacle was time. For example, we wanted to implement a feature in our program which would have the user just ask one global question (for example, "How should I dance with her?"), and if the program thought he did not have a good chance of dancing with the girl, then it would stop and would make no dance suggestions. However, if he had a chance, the program would go on asking the questions about his dancing preferences. We ended up dividing the program into two parts, one where the user asks the top-level goal question, "Do I have a chance to dance with her?" Then, should he have a good chance to dance with her (in which case the program would tell him to dance with her right away or go and talk to her), he needs to ask the program the other high-level question, "How should I dance with her?" Moreover, the system is redundant in the sense that if the user decides to approach another girl, the system will still ask him all of the questions it asked him before, such as how he is dressed. With more time, we would have addressed this issue.

Although production rules captured the knowledge conveniently, it did not solve all of our knowledge engineering problems. One big problem was trying to capture the probabilistic nature of the knowledge via the rules' certainty factors. For example, while interviewing Oliver, he said, "If you are going to the party with friends, make sure they look as good, if not even better, than you. That will *surely* make a good impression on the girl and her friends." The problem is, how do you translate the word *surely* into a numeric certainty value, such as 0.5? Although we had a general idea of what *range* of certainty values a word would fall into, within that range our certainty assignments were somewhat arbitrary.

Another knowledge engineering problem that we tackled was the large amounts of knowledge we had. After all of our interviews and observations were done, we had a large, disconnected body of knowledge. Converting all this knowledge into the right attribute-object-value triples, and tying all the knowledge together, was a rather complicated and extremely time-consuming task.

In a similar manner to rules, Joshua as an expert system tool made some tasks convenient for us while making others difficult. As humans, sometimes we made typing mistakes (by missing either a bracket or a parenthesis when writing a rule), and we often found ourselves doing a lot of debugging. Consider, for example, the following compiler error message:

```
Error: eof encountered on stream
       #<EXCL:FILE-SIMPLE-STREAM #p"/tmp/cdaaedda5.cl" for input pos
79654 @ #x52eb1a2> starting at position 73634.
  [condition type: END-OF-FILE]
```

This error message would be helpful if we knew where "pos 79654" was located with respect to our 150 or more rules.

A more serious problem with the Joshua engine was that it did not provide enough flexibility. For example, we wanted to have more control over the certainty factors and over the predications already present in the database. This led me to contemplate switching from Joshua to another Expert System engine, such as Java's JESS. However, JESS does not have nearly as many features as Joshua does, it is poorly documented, and rewriting our gigantic knowledge base to adjust to JESS was not worth the trouble.

Lessons Learned

This project exercise was a lesson about knowledge. That is, the most important part of the program is what it knows. If it does not know about a certain situation, no matter how fancy the reasoning mechanism is, it cannot solve a problem related to that situation. Our program works well and almost the way we originally envisioned it to work. We are satisfied with our program's performance, mainly because our focus was on the *knowledge* of the system, and the issues revolving that knowledge: knowledge representation and reasoning mechanism.

Another lesson we learned was that working in teams of two can be beneficial and not hurt productivity. Working with James made the design and implementation of this system more enjoyable, and I learned much more than I would have learned had I worked on the project alone. However, this lesson should not be abused, because I can foresee big scheduling and performance problems if the teams have more than two members.

Issues for Further Research

Given the current state of our system, it would benefit from a number of upgrades and enhancements, in which system engineers can:

- Link the two parts of our system; that is, provide the user with just one higher-level goal, such as "How should I dance with her?" where the system would stop if the user does not have a chance of dancing with the girl, and would continue providing advice on how to dance with her if he did have a chance.
- Learn from user experience. One way would be to get user feedback on his success after the program is done making its recommendation, and based on that feedback, the system would automatically update the certainties of all the rules in the knowledge base.
- Write the system as a Java (or any other programming language) program and install it in a cell phone, enabling people to actually use the program.

Conclusions

The Groove-X expert system has worked relatively well with our expectations. From working on this project over the course of the semester, we learned that the most important parts of a knowledge-based system revolve around issues of knowledge: what should the system know, how it should represent what it knows, and how it should reason about what it knows. For the field of dancing at a college party, rules with backward chaining provided us with much expressional power. This consulting program can be extended in several ways, and it can be integrated with other practical applications.

REFERENCES

1. Davis, R. Production Rules as a representation for a knowledge-based consultation Program, *Artificial Intelligence Journal*, 8:15-45, 1977.
2. Topolski A S, Reece D K. Packaging Advisor(tm): An expert system for rigid plastic food package design, in *Proc Innovative Applications of AI*, 1989, Schoor and Rappaport (eds.), pp. 348 357.
3. Campbell A. N. et al. Recognition of a hidden mineral deposit by an artificial Intelligence program, *Science*, 217:927-929, September 1982.
4. Sergot et. al, The British Nationality Act As A Logic Program, *Communications of the ACM*, May 1986, 370 386
5. Cooke, N.J,, Varieties of Knowledge Elicitation Techniques, *Int. J. Human-Computer Studies*, 41:801-849, 1994.

APPENDIX: Rules in the Groove-X knowledge base

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;META RULES META RULES META RULES START;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 (defrule dance-with-girl-no-talking  (:backward :certainty 0.1  :importance 415)
  if [dance-not-talk ?guy yes]
  then [interact-with-girl ?guy dance-with-her])

 (defrule guy-no-started-conversation (:backward :certainty 1.0 :importance 412)
  if [dance-not-talk ?guy yes]
  then [started-conversation ?guy no])

;; Rule for there is no conversation
(defrule guy-no-start-conversation (:backward :certainty 1.0 :importance 411)
  if [dance-not-talk ?guy yes]
  then [start-conversation ?guy no])

(defrule guy-start-conversation (:backward :certainty 1.0 :importance 410)
  if [and [initial-approach ?guy yes]
          [started-conversation ?guy yes]]
  then [start-conversation ?guy yes])

;;(defrule dance-without-talking (:backward :certainty 1.0  :importance 400)
;; if [dance-not-talk ?guy yes]
;; then [start-conversation ?guy no])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;META RULES META RULES META RULES END;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning the guy's hair state;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule hair-state-question (:backward :certainty 1.0 :importance 363)
  if [user-hair-state ?guy ?x]
  then [hair-state ?guy ?x])

(defrule hair-dandruff  (:backward :certainty 1.0  :importance 362)
  if [and [hair-state ?guy i-dont-know]
          [dandruff-in-hair ?guy yes]]
  then[hair-state ?guy unkempt])

(defrule hair-shapeup-kempt  (:backward :certainty 1.0  :importance 361)
  if [and [hair-state ?guy i-dont-know]
          [hair-shapeup ?guy no]]
  then[hair-state ?guy unkempt])

(defrule braid-fuzziness (:backward :certainty 1.0  :importance 360)
  if [and [hair-state ?guy i-dont-know]
          [has-braids ?guy yes]
          [braids-are ?guy fuzzy]]
  then[hair-state ?guy unkempt])

(defrule no-hair-dandruff (:backward :certainty 0.4 :importance 359)
 if [and [hair-state ?guy i-dont-know]
          [dandruff-in-hair ?guy no]]
  then[hair-state ?guy clean])

(defrule hair-shapeup-kempt2  (:backward :certainty 0.6  :importance 358)
  if [and [hair-state ?guy i-dont-know]
          [hair-shapeup ?guy yes]]
  then[hair-state ?guy clean])

(defrule braid-neatness (:backward :certainty 1.0  :importance 357)
  if [and [hair-state ?guy i-dont-know]
          [has-braids ?guy yes]
          [braids-are ?guy fuzzy]]
  then[hair-state ?guy unkempt])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning crispness of a guy's shoes;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule shoe-crisp-user-input (:backward :certainty 1.0 :importance 356)
  if [shoes-crisp-input ?guy fresh]
  then [shoes-crisp ?guy yes])

(defrule shoe-crisp-user-input2 (:backward :certainty 1.0 :importance 355)
  if [shoes-crisp-input ?guy unkempt]
  then [shoes-crisp ?guy no])

(defrule shoe-spots-crisp (:backward :certainty 1.0  :importance 354)
  if [and [shoes-crisp-input ?guy i-dont-know]
          [shoes-spots ?guy yes]]
  then[shoes-crisp ?guy no])

(defrule shoe-wrinkles-not-crisp (:backward :certainty 1.0  :importance 353)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-wrinkled ?guy yes]]
  then[shoes-crisp ?guy no])

(defrule shoe-dirty-not-crisp (:backward :certainty 1.0  :importance 352)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-dirty ?guy yes]]
  then[shoes-crisp ?guy no])

(defrule shoe-roll-not-crisp  (:backward :certainty 1.0  :importance 351)
  if [and [shoes-crisp-input ?who i-dont-know]
          [front-of-shoes-rolled ?guy yes]]
  then[shoes-crisp ?guy no])

(defrule shoe-faded-not-crisp (:backward :certainty 1.0  :importance 350)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-faded ?guy yes]]
  then [shoes-crisp ?guy no])

(defrule shoe-wrinkles-crisp (:backward :certainty 0.2  :importance 349)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-wrinkled ?guy yes]]
  then[shoes-crisp ?guy yes])

(defrule shoe-dirty-crisp (:backward :certainty 0.2  :importance 348)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-dirty ?guy yes]]
  then[shoes-crisp ?guy yes])
```

```
(defrule shoe-roll-crisp  (:backward :certainty 0.3  :importance 347)
  if [and [shoes-crisp-input ?who i-dont-know]
          [front-of-shoes-rolled ?guy yes]]
  then[shoes-crisp ?guy yes])

(defrule shoe-faded-crisp (:backward :certainty 0.3  :importance 346)
  if [and [shoes-crisp-input ?who i-dont-know]
          [shoes-faded ?guy yes]]
  then [shoes-crisp ?guy yes])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;Rules concerning your smile ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule bad-smile  (:backward :certainty 1.0  :importance 344)
  if [are-yellow ?guy yes]
  then [smile ?guy busted])

(defrule bad-smile2  (:backward :certainty 1.0  :importance 343)
  if [clean-teeth ?guy no]
  then [smile ?guy busted])

(defrule bad-smile3  (:backward :certainty 1.0  :importance 342)
  if [missing-teeth ?guy yes]
  then [smile ?guy busted])

(defrule nice-smile-rule  (:backward :certainty 0.7  :importance 341)
  if [and [are-yellow ?guy no]
          [missing-teeth ?guy no]
          [clean-teeth ?guy yes]
          [teeth-straight ?guy yes]]
  then [smile ?guy nice])


;;; Note: recommend guy not to talk if he has busted teeth

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning shirt state ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule shirt-state-rule (:backward :certainty 1.0 :importance 339)
  if [user-shirt-state ?guy fresh]
  then [shirt-state ?guy fresh])

(defrule shirt-state-rule2 (:backward :certainty 1.0 :importance 338)
  if [user-shirt-state ?guy unkempt]
  then [shirt-state ?guy unkempt])

(defrule wrinkled-shirt-state  (:backward :certainty 1.0  :importance 332)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-wrinkled ?guy yes]]
  then[shirt-state ?guy unkempt])

(defrule shirt-spots-state (:backward :certainty 1.0  :importance 331)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-spots ?guy yes]]
  then [shirt-state ?guy unkempt])

(defrule hole-shirt-state  (:backward :certainty 1.0  :importance 330)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-holes ?guy yes]]
  then [shirt-state ?guy unkempt])

(defrule wrinkled-shirt-state2  (:backward :certainty 0.5  :importance 329)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-wrinkled ?guy no]]
  then[shirt-state ?guy fresh])

(defrule shirt-spots-state2 (:backward :certainty 0.4  :importance 328)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-spots ?guy no]]
  then [shirt-state ?guy fresh])

(defrule hole-shirt-state2  (:backward :certainty 0.3  :importance 327)
  if [and [user-shirt-state ?guy i-dont-know]
          [shirt-holes ?guy no]]
  then [shirt-state ?guy fresh])



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning guy pant's  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule user-pant-input (:backward :certainty 1.0 :importance 322)
  if [user-pants-input ?guy fresh]
  then [pant-state ?guy fresh])

(defrule user-pant-input2 (:backward :certainty 1.0 :importance 322)
  if [user-pants-input ?guy unkempt]
  then [pant-state ?guy unkempt])

;; These next questions will only be asked if the user says "I dont know

(defrule pants-spots-unkempt  (:backward :certainty 1.0  :importance 321)
  if [and [user-pants-input ?guy i-dont-know]
          [pant-spots ?guy yes]]
  then[pant-state ?guy unkempt])

(defrule pants-holes-unkempt  (:backward :certainty 1.0  :importance 320)
  if [and [user-pants-input ?who i-dont-know]
          [pant-holes ?guy yes]]
  then[pant-state ?guy unkempt])

(defrule pants-wrinkled-unkempt  (:backward :certainty 1.0  :importance 319)
  if [and [user-pants-input ?who i-dont-know]
          [pant-wrinkles ?guy yes]]
  then[pant-state ?guy unkempt])

;; Rules -- there's no easy way to say "if the pants are not unkempt, then
;; they are kempt"

(defrule pants-spots-clean  (:backward :certainty 1.0  :importance 318)
  if [and [user-pants-input ?guy i-dont-know]
          [pant-spots ?guy no]]
  then[pant-state ?guy fresh])

(defrule pants-holes-clean  (:backward :certainty 1.0  :importance 317)
  if [and [user-pants-input ?who i-dont-know]
```

```
        [pant-holes ?guy no]]
  then[pant-state ?guy fresh])

(defrule pants-wrinkled-clean  (:backward :certainty 1.0  :importance 319)
  if [and [user-pants-input ?who i-dont-know]
        [pant-wrinkles ?guy no]]
  then[pant-state ?guy fresh])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning guy's body smell;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule washed-body-smell  (:backward :certainty 1.0  :importance 315)
  if [has-washed ?guy no]
  then[body-smell ?guy stinks])

(defrule washed-body-smell2  (:backward :certainty 1.0  :importance 314)
  if [has-washed ?guy yes]
  then[body-smell ?guy clean])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning guy style good/bad;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule shirt-state-adds-style (:backward :certainty 1.0  :importance 310)
  if [shirt-state ?guy clean]
  then [style ?guy good])

(defrule pant-state-adds-style (:backward :certainty 1.0  :importance 309)
  if [pant-state ?guy clean]
  then [style ?guy good])

(defrule pant-state-cancels-style (:backward :certainty 1.0  :importance 308)
  if [pant-state ?guy unkempt]
  then [style ?guy bad])

(defrule shirt-state-cancels-style (:backward :certainty 1.0  :importance 307)
  if [shirt-state ?guy unkempt]
  then [style ?guy bad])

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; Style subrule: Rules concerning type of shoes ;;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defrule adidas-brand  (:backward :certainty 1.0  :importance 306)
  if [shoe-name ?guy adidas]
  then[shoe-type ?guy sneakers])

(defrule nike-brand  (:backward :certainty 1.0  :importance 305)
  if [shoe-name ?guy nike]
  then[shoe-type ?guy sneakers])

(defrule gunit-brand  (:backward :certainty 1.0  :importance 304)
  if [shoe-name ?guy g-unit]
  then[shoe-type ?guy sneakers])

(defrule reebok-brand  (:backward :certainty 1.0  :importance 303)
  if [shoe-name ?guy reebok]
  then[shoe-type ?guy sneakers])

(defrule timberland-brand  (:backward :certainty 1.0  :importance 302)
  if [shoe-name ?guy timberland]
  then[shoe-type ?guy timbs])

(defrule sweats-shoes-coordination  (:backward :certainty 0.9  :importance 301)
  if [and [pant-type ?guy sweats]
          [shoe-type ?guy sneakers]]
  then [style ?guy good])

(defrule jean-shoes-coordination2  (:backward :certainty 0.9  :importance 300)
  if [and [pant-type ?guy jeans]
          [shoe-type ?guy sneakers]]
  then [style ?guy good])

(defrule jean-shoes-coordination3  (:backward :certainty 0.9  :importance 299)
  if [and [pant-type ?guy jeans]
          [shoe-type ?guy timbs]]
  then [style ?guy good])

 (defrule shirt-matches-nothing (:backward :certainty 0.8 :importance 298)
  if [matches-shirt ?guy none-of-the-above]
  then [style ?guy bad])

(defrule hat-match-shirt  (:backward :certainty 0.6  :importance 297)
  if [matches-shirt ?guy hat]
  then [style ?guy good])

(defrule shirt-matches-shoe (:backward :certainty 0.6 :importance 296)
  if [matches-shirt ?guy shoes]
  then [style ?guy good])

(defrule shirt-brand-style  (:backward :certainty 0.7  :importance 295)
  if [and [is-shirt-dress-shirt ?guy no]
          [or [shirt-brand-name ?guy Rocawear]
             [shirt-brand-name ?guy Rocawear]]]
  then [style ?guy good])

(defrule shirt-matching-shoes  (:backward :certainty 0.7  :importance 294)
  if [matches-shirt ?guy shoes]
  then[style ?guy good])

(defrule dress-shirt-rule  (:backward :certainty 0.7  :importance 293)
  if [and [is-shirt-dress-shirt ?guy yes]
         [or [shirt-brand-name ?guy Lacoste]
              [shirt-brand-name ?guy HM]]]
  then [style ?guy good])

(defrule shirt-pant-brand  (:backward :certainty 0.8 :importance 292)
  if [does-pant-brand-match-shirt ?guy yes]
  then [style ?guy good])

(defrule shoes-crisp-style  (:backward :certainty 0.7  :importance 291)
  if [shoes-crisp ?guy no]
  then [style ?guy bad])

;; This will take care of the friends question

(defrule friends-crisp-style (:backward :certainty 0.3 :importance 290)
  if [and [or [number-of-friends ?guy few]
              [number-of-friends ?guy lots]]
          [friends-crisp ?guy yes]]
  then[style ?guy good])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning your attractiveness to girl;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule not-visible-features (:backward :certainty 1.0  :importance 402)
 if [party-lighting ?guy dim]
 then [is-not-visible ?guy yes])

(defrule not-visible-features2 (:backward :certainty 1.0  :importance 401)
 if [party-lighting ?guy off]
 then [is-not-visible ?guy yes])

(defrule visible-features (:backward :certainty 1.0 :importance 400)
  if [party-lighting ?guy on]
  then [is-not-visible ?guy no])

(defrule unkempt-hair (:backward :certainty 0.7  :importance 280)
  if [and [is-not-visible ?guy yes]
          [hair-state ?guy unkempt]]
  then[attractiveness ?guy decreases])

(defrule clean-hair (:backward :certainty 0.5 :importance 279)
  if [and [is-not-visible ?guy no]
          [hair-state ?guy clean]]
  then [attractiveness ?guy increases])

(defrule clean-shaved-guy  (:backward :certainty 0.5  :importance 278)
  if [clean-shaved ?guy yes]
  then[attractiveness ?guy increases])

(defrule mustache-shapeup  (:backward :certainty 0.3  :importance 277)
  if [is-mustache-shaped-up ?guy yes]
  then [attractiveness ?guy increases])

(defrule acne-attractiveness (:backward :certainty 0.7  :importance 276)
  if [and [is-not-visible ?guy yes]
          [acne ?guy present]]
  then[attractiveness ?guy decreases])

(defrule nice-smile-attractiveness (:backward :certainty 0.7  :importance 275)
  if [smile ?guy nice]
  then[attractiveness ?guy increases])

(defrule good-style-attractive (:backward :certainty 0.8  :importance 274)
  if [style ?guy good]
  then[attractiveness ?guy increases])

(defrule bad-style-not-attractive  (:backward :certainty 0.8  :importance 273)
  if [style ?guy bad]
  then [attractiveness ?guy decreases])

(defrule body-smell-attractiveness  (:backward :certainty 0.8  :importance 272)
  if [body-smell ?guy stinks]
  then[attractiveness ?guy decreases])

(defrule cologne-smell-attractiveness  (:backward :certainty 0.7  :importance 271)
  if [and [body-smell ?guy clean]
        [wearing-cologne ?guy yes]]
  then[attractiveness ?guy increases])

(defrule breath-smelling-attractiveness (:backward :certainty 0.8  :importance 270)
  if [breath-smell ?guy yes]
  then[attractiveness ?guy decreases])

;; Moved rule below from 279 to 269 importance

(defrule darkness-increase-attractiveness (:backward :certainty 0.2  :importance 269)
 if [is-not-visible ?guy yes]
 then [attractiveness ?guy increases])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning the impression the guy makes;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule attractive-impression  (:backward :certainty 0.8  :importance 254)
  if [attractiveness ?guy increases]
  then[make-impression ?guy good])

(defrule not-attractive-impression (:backward :certainty 0.6  :importance 253)
  if [attractiveness ?guy decreases]
  then[make-impression ?guy bad])

(defrule dancing-well  (:backward :certainty 0.6  :importance 252)
  if [and [dance-experience ?guy expert]
          [dancing-at-moment ?guy yes]]
  then [make-impression ?guy good])

(defrule friends-make-initial-impression  (:backward :certainty 0.8 :importance 251)
  if [and [initial-approach ?guy yes]
          [are-her-friends-around ?guy yes]
          [friends-initial-impression ?guy bad]]
  then [make-impression ?guy bad])

(defrule approach-talk-to-group  (:backward :certainty 0.4  :importance 250)
  if [and [start-conversation ?guy yes]
          [are-her-friends-around ?guy yes]
          [talk-to-friends ?guy yes]]
  then [make-impression ?guy good])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning conversation             ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule personal-questions  (:backward :certainty 1.0  :importance 240)
  if [are-questions-personal ?guy yes]
  then[is-conversation-personal ?guy yes])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning guy's attitude   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule conversation-starter  (:backward :certainty 1.0  :importance 235)
  if [start-conversation ?guy ?yes]
  then[attitude ?guy confident])

(defrule beer-effect-guy (:backward :certainty 0.4 :importance 234)
  if [is-user-drinking ?guy yes]
  then [attitude ?guy confident])

(defrule guy-friends-number  (:backward :certainty 1.0  :importance 233)
  if [number-of-friends ?guy lots]
  then[attitude ?guy confident])

(defrule cheesy-attitude-eager  (:backward :certainty 1.0  :importance 232)
  if [is-cheesing ?guy yes]
  then[attitude ?guy eager])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning guy's attitude displayed ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule eager-attitude-negative  (:backward :certainty 0.6  :importance 231)
  if [attitude ?guy eager]
  then[attitude-displayed ?guy negative])

(defrule confident-attitude-positive  (:backward :certainty 0.8  :importance 230)
  if [attitude ?guy confident]
  then[attitude-displayed ?guy positive])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning girl comfortableness with guy;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule looking-at-her-upanddown  (:backward :certainty 0.5 :importance 217)
 if [is-looking-her-up-down ?guy yes]
 then [girl-comfortableness ?guy decreases])

;; Note: if make-eye-contact predicate fails
;; then eye-contact-lengh question shouldn't be asked

(defrule eye-contact-too-long-rule  (:backward :certainty 0.7  :importance 216)
  if [and [make-eye-contact ?guy yes]
          [eye-contact-length ?guy long]]
  then [girl-comfortableness ?guy decreases])

(defrule close-to-girl  (:backward :certainty 0.5  :importance 215)
  if [and [initial-approach ?guy yes]
          [half-arms-length ?guy yes]]
  then [girl-comfortableness ?guy increases])

(defrule touch-girl-initially-rule (:backward :certainty 0.4  :importance 214)
  if [and [initial-approach ?guy yes]
          [touch-girl-initially ?guy yes]]
  then [girl-comfortableness ?guy decreases])

(defrule personal-conversation-girl  (:backward :certainty 0.7  :importance 213)
  if [and [start-conversation ?guy yes]
          [is-conversation-personal ?guy yes]]
  then [girl-comfortableness ?guy decreases])

(defrule ask-name-uncomfortable  (:backward :certainty 0.3  :importance 212)
  if [and [start-conversation ?guy yes]
          [ask-for-name ?guy yes]]
  then [girl-comfortableness ?guy decreases])

(defrule being-obscene  (:backward :certainty 0.8  :importance 211)
 if [and [start-conversation ?guy yes]
```

```
         [is-obscene ?guy yes]]
 then [girl-comfortableness ?guy decreases])

(defrule show-interest-she-says2  (:backward :certainty 0.7  :importance 210)
  if [and [start-conversation ?guy yes]
          [show-interest-in-girl ?guy yes]]
  then [girl-comfortableness ?guy increases])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning flattery ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; Flattery subsetion: Rules concerning guy's comment to girl ;;
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule compliment-her-outfit-rule  (:backward :certainty 0.7  :importance 208)
  if [and [user-comment-input ?guy yes]
          [compliment-girl-outfit ?guy yes]]
  then [comment-to-girl ?guy nice])

(defrule compliment-her-eyes-rule  (:backward :certainty 0.7  :importance 207)
  if [and [user-comment-input ?guy yes]
          [compliment-girl-eyes ?guy yes]]
  then [comment-to-girl ?guy nice])

(defrule make-eye-contact-rule  (:backward :certainty 0.7 :importance 206)
  if [and [make-eye-contact ?guy yes]
          [eye-contact-length ?guy short]]
  then [girl-flattery ?guy increases])

(defrule talking-for-awhile  (:backward :certainty 0.5  :importance 205)
  if [and [start-conversation ?guy yes]
          [touch-girl-awhile ?guy yes]]
  then [girl-flattery ?guy increases])

(defrule paying-attention-to-her-rule  (:backward :certainty 0.8  :importance 204)
  if [and [start-conversation ?guy yes]
          [paying-attention-to-girl ?guy yes]]
        then [girl-flattery ?guy increases])

(defrule say-something-mean  (:backward :certainty 0.4  :importance 203)
  if [and [start-conversation ?guy yes]
          [user-comment-input ?guy yes]
          [comment-to-girl ?guy mean]]
  then [girl-flattery ?guy decreases])

(defrule say-something-lame  (:backward :certainty 0.2  :importance 202)
  if [and [start-conversation ?guy yes]
          [user-comment-input ?guy yes]
          [comment-to-girl ?guy lame]]
  then [girl-flattery ?guy decreases])

(defrule say-something-nice  (:backward :certainty 0.3  :importance 201)
  if [and [start-conversation ?guy yes]
          [user-comment-input ?guy yes]
          [comment-to-girl ?guy nice]]
  then [girl-flattery ?guy increases])
```

```
(defrule show-interest-she-says  (:backward :certainty 0.5  :importance 200)
  if [show-interest-in-girl ?guy yes]
  then [girl-flattery ?guy increases])


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning when a girl action        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule good-conversation-is-responsive (:backward :certainty 1.0 :importance 199)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy good]]
  then [is-girl-responsive ?guy yes])

(defrule bad-conversation-is-non-responsive (:backward :certainty 1.0 :importance 198)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy bad]]
  then [is-girl-responsive ?guy no])

(defrule responsive-conversation-rule  (:backward :certainty 1.0  :importance 197)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
         [is-girl-making-eye-contact ?guy yes]]
  then [is-girl-responsive ?guy yes])

(defrule responsive-conversation-rule2  (:backward :certainty 1.0  :importance 196)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
          [is-girl-laughing ?guy yes]]
  then [is-girl-responsive ?guy yes])

(defrule responsive-conversation-rule3  (:backward :certainty 1.0  :importance 195)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
          [is-girl-smiling ?guy yes]]
  then [is-girl-responsive ?guy yes])

(defrule responsive-conversation-rule4  (:backward :certainty 1.0  :importance 194)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
          [is-girl-moving-closer-conversate ?guy yes]]
  then [is-girl-responsive ?guy yes])

(defrule non-responsive-conversation-rule  (:backward :certainty 1.0  :importance 192)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
          [is-girl-giving-excuses ?guy yes]]
  then [is-girl-responsive ?guy no])

(defrule non-responsive-conversation-rule2  (:backward :certainty 1.0  :importance 191)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
          [is-girl-looking-away ?guy yes]]
  then [is-girl-responsive ?guy no])

(defrule non-responsive-conversation-rule3  (:backward :certainty 1.0  :importance 190)
  if [and [start-conversation ?guy yes]
          [user-conversate-input ?guy i-dont-know]
         [is-girl-talking-on-phone ?guy yes]]
  then [is-girl-responsive ?guy no])
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning girl's attitude ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule girl-drinking-effect (:backward :certainty 0.5 :importance 181)
  if [is-girl-drinking ?guy yes]
  then [girl-attitude ?guy interested])

(defrule good-impression-on-girl  (:backward :certainty 0.6  :importance 180)
  if [make-impression ?guy good]
  then [girl-attitude ?guy interested])

;; updated the girl-just walks away rule
(defrule girl-just-walks-away (:backward :certainty 0.9  :importance 179)
  if [and [initial-approach ?guy yes]
          [girl-walks-away ?guy yes]]
  then [girl-attitude ?guy not-interested])

(defrule girl-does-not-respond (:backward :certainty 0.8  :importance 178)
  if [and [start-conversation ?guy yes]
          [is-girl-responsive ?guy no]]
  then [girl-attitude ?guy not-interested])

(defrule girl-responds-to-you  (:backward :certainty 0.8  :importance 177)
  if [and [start-conversation ?guy yes]
          [is-girl-responsive ?guy yes]]
  then [girl-attitude ?guy interested])

(defrule girl-comfortableness-attitude-1  (:backward :certainty 0.9  :importance 176)
  if [girl-comfortableness ?guy increases]
  then [girl-attitude ?guy interested])

(defrule girl-comfortableness-attitude-2  (:backward :certainty 0.9  :importance 175)
  if [girl-comfortableness ?guy decreases]
  then [girl-attitude ?guy not-interested])

(defrule girl-flattery-attitude  (:backward :certainty 0.7  :importance 174)
  if [girl-flattery ?guy increases]
  then [girl-attitude ?guy interested])

(defrule bad-impression-on-girl (:backward :certainty 0.7  :importance 172)
  if [make-impression ?guy bad]
  then [girl-attitude ?guy not-interested])

(defrule negative-attitude-on-girl  (:backward :certainty 0.6  :importance 171)
  if [attitude-displayed ?guy negative]
  then [girl-attitude ?guy not-interested])

(defrule positive-attitude-on-girl  (:backward :certainty 0.7  :importance 170)
  if [attitude-displayed ?guy positive]
  then [girl-attitude ?guy interested])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                  ;;
;;                 Party Rules  (are below)                         ;;
;;                                                                  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning party phase  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Note: We need to make a new predicate-models to handle questions about
;; party for boolean and option-mixin

(defrule time-phase-beginning-rule (:backward :certainty 1.0 :importance 142)
  if [user-time-input ?guy beginning]
  then [is-party-phase ?guy beginning])

(defrule time-phase-middle-rule (:backward :certainty 1.0 :importance 141)
  if [user-time-input ?guy middle]
  then [is-party-phase ?guy middle])

(defrule time-phase-end-rule (:backward :certainty 1.0 :importance 140)
  if [user-time-input ?guy end]
  then [is-party-phase ?guy end])

(defrule time-close-to-beginning  (:backward :certainty 1.0  :importance 139)
  if [and [user-time-input ?guy i-dont-know]
          [size-of-party ?guy not-crowded]]
  then [is-party-phase ?guy beginning])

(defrule people-are-not-sweaty  (:backward :certainty 0.8  :importance 138)
  if  [and [user-time-input ?guy i-dont-know]
           [people-sweaty-party ?guy no]]
  then [is-party-phase ?guy beginning])

(defrule most-people-talking  (:backward :certainty 0.6  :importance 137)
  if [and [user-time-input ?guy i-dont-know]
          [people-action-party ?guy talking]]
  then [is-party-phase ?guy beginning])

(defrule most-people-dancing  (:backward :certainty 1.0  :importance 136)
  if [and [user-time-input ?who i-dont-know]
          [people-action-party ?guy dancing]]
  then [is-party-phase ?guy middle])

(defrule people-are-sweaty (:backward :certainty 0.5  :importance 134)
  if [and [user-time-input ?guy i-dont-know]
          [people-sweaty-party ?guy yes]]
  then [is-party-phase ?guy end])

(defrule trash-on-floor-lots  (:backward :certainty 1.0  :importance 133)
  if [and [user-time-input ?guy i-dont-know]
          [floor-trash-level ?guy lot]]
  then [is-party-phase ?guy end])

(defrule lights-are-on  (:backward :certainty 1.0  :importance 132)
  if [and [user-time-input ?guy i-dont-know]
          [party-lighting ?guy on]]
  then [is-party-phase ?guy end])

(defrule people-are-leaving  (:backward :certainty 1.0  :importance 131)
  if [and [user-time-input ?guy i-dont-know]
          [people-action-party ?guy leaving]]
  then [is-party-phase ?guy end])

(defrule music-volume-is-low-or-off  (:backward :certainty 1.0  :importance 130)
  if [and [user-time-input ?guy i-dont-know]
          [or [music-volume-at-party ?guy low]
              [music-volume-at-party ?guy off]]]
  then [is-party-phase ?guy end])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; More Rules concerning party ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defrule people-putting-on-jackets (:backward :certainty 1.0  :importance 125)
  if [are-people-putting-on-jackets ?guy yes]
  then [people-action-party ?guy leaving])

;; Change this rule to people dancing mean song type is easy
(defrule not-many-people-dancing (:backward :certainty 1.0  :importance 123)
  if [people-action-party ?guy dancing]
  then [party-song-dance-type ?guy easy])

(defrule too-fast-for-beginner (:backward :certainty 1.0  :importance 122)
  if [and [dance-experience ?guy novice]
          [party-song-speed ?guy fast]]
  then [party-song-dance-type ?guy hard])

(defrule song-too-slow  (:backward :certainty 1.0  :importance 121 )
  if [party-song-speed ?guy slow]
  then [party-song-dance-type ?guy hard])

(defrule guys-dance-fighting-hard  (:backward :certainty 1.0  :importance 120)
  if [people-action-party ?guy guys-dance-fighting]
  then [party-song-dance-type ?guy hard])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning girl's dance comfortableness;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule party-light-is-bright  (:backward :certainty 0.6  :importance 113)
  if [party-lighting ?guy on]
  then [girl-dancing-comfortableness ?guy decreases])

(defrule party-light-pitch-black  (:backward :certainty 0.9  :importance 112)
  if [party-lighting ?guy off]
  then [girl-dancing-comfortableness ?guy increases])

(defrule light-is-dim  (:backward :certainty 0.7  :importance 111)
  if [party-lighting ?guy dim]
  then [girl-dancing-comfortableness ?guy increases])

(defrule song-not-easy-dance (:backward :certainty 0.3  :importance 110)
  if [party-song-dance-type ?guy hard]
  then [girl-dancing-comfortableness ?guy decreases])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning smooth approach ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule notjerking-your-body  (:backward :certainty 1.0  :importance 102)
  if [and [initial-approach ?guy yes]
          [is-body-jerking ?guy yes]]
  then [is-approach-smooth ?guy no])
```

```
(defrule jerking-your-body  (:backward :certainty 1.0  :importance 102)
  if [and [initial-approach ?guy yes]
          [is-body-jerking ?guy yes]]
  then [is-approach-smooth ?guy no])

(defrule dance-to-the-rhythm-no  (:backward :certainty 1.0  :importance 101)
  if [and [initial-approach ?guy yes]
          [dancing-to-rhythm ?guy no]]
  then [is-approach-smooth ?guy no])

(defrule dance-to-the-rhythm  (:backward :certainty 1.0  :importance 100)
  if [and [initial-approach ?guy yes]
          [dancing-to-rhythm ?guy yes]]
  then [is-approach-smooth ?guy yes])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;---------Start of Questions about chance with Dance   -------;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning your chance to dance with girl;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

     ;;;;; Chance affected by the girl's attitude;;;;;;;;;;;
     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule chance-girl-girl-interested (:backward :certainty 0.9  :importance 91)
  if [girl-attitude ?guy interested]
  then [chance-of-dancing-with-girl ?guy yes])

(defrule chance-girl-not-interested (:backward :certainty 0.7 :importance 90)
  if [girl-attitude ?guy not-interested]
  then [chance-of-dancing-with-girl ?guy no])

(defrule party-phase-dance-great  (:backward :certainty 0.7  :importance 77)
 if [and [is-party-phase ?guy middle]
         [people-action-party ?guy dancing]
         [size-of-party ?guy crowded]]
 then [chance-of-dancing-with-girl ?guy yes])

(defrule party-phase-dance-good  (:backward :certainty 0.5  :importance 76)
 if [is-party-phase ?guy middle]
 then [chance-of-dancing-with-girl ?guy yes])

(defrule party-phase-dance  (:backward :certainty 0.2  :importance 75)
 if [or [is-party-phase ?guy beginning]
       [is-party-phase ?guy end]]
 then [chance-of-dancing-with-girl ?guy no])

(defrule comfortable-dance (:backward :certainty 0.7  :importance 74)
  if [girl-dancing-comfortableness ?guy increases]
  then[chance-of-dancing-with-girl ?guy yes])


(defrule friends-circle-dancing  (:backward :certainty 0.9  :importance 73)
  if [and [are-her-friends-around ?guy yes]
          [girl-friends-action ?guy dancing-with-guys]
          [are-girls-in-circle ?guy yes]]
  then [chance-of-dancing-with-girl ?guy yes])
```

```
(defrule friends-circle-not-dancing  (:backward :certainty 0.3  :importance 72)
  if [and [are-her-friends-around ?guy yes]
          [girl-friends-action ?guy standing]
          [are-girls-in-circle ?guy yes]]
  then [chance-of-dancing-with-girl ?guy no])

(defrule chance-on-dance-floor  (:backward :certainty 0.5  :importance 71)
  if [and [girl-location ?guy on-dance-floor]
          [size-of-party ?guy crowded]]
  then [chance-of-dancing-with-girl ?guy yes])

(defrule approach-smoothdancer  (:backward :certainty 0.6  :importance 70)
  if [and [initial-approach ?guy yes]
          [is-approach-smooth ?guy yes]
          [girl-location ?guy on-wall]]
  then [chance-of-dancing-with-girl ?guy yes])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Rules concerning Dance With Girl ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule dance-chance-yes (:backward :certainty 1.0  :importance 62)
  if [chance-of-dancing-with-girl ?guy yes]
  then [interact-with-girl ?guy dance-with-her])

(defrule dance-chance-no (:backward :certainty 1.0  :importance 61)
  if [chance-of-dancing-with-girl ?guy no]
  then [interact-with-girl ?guy leave-her-alone])

(defrule dance-chance-talk (:backward :certainty 0.6  :importance 60)
  if [and [size-of-party ?guy crowded]
          [people-action-party ?guy standing]]
  then [interact-with-girl ?guy talk-to-her])

;; Added rules that connect dance-with-girl to interact-with-girl
;; using Shrobe technique of certainty factors -JAT

(defrule dance-from-interact-yes (:backward :certainty 1.0 :importance 52)
  if [and [interact-with-girl ?guy dance-with-her]
          [interact-with-girl ?guy leave-her-alone]
          (> (certainty-factor (tell [interact-with-girl ?guy dance-with-her] :justifi-
cation :none)) 0.9)
          (< (certainty-factor (tell [interact-with-girl ?guy leave-her-alone] :justifi-
cation :none)) 0.5)]
 then [dance-with-girl ?guy yes])

(defrule dance-from-interact-yes-two (:backward :certainty 1.0 :importance 51)
  if [and [interact-with-girl ?guy dance-with-her]
  (> (certainty-factor (tell [interact-with-girl ?guy dance-with-her] :justification :
none)) 0.95)]
  then [dance-with-girl ?guy yes])

(defrule dance-from-interact-no (:backward :certainty 1.0 :importance 50)
  if [and [interact-with-girl ?guy leave-her-alone]
          [interact-with-girl ?guy dance-with-her]
          (>= (certainty-factor (tell [interact-with-girl ?guy leave-her-alone] :justi-
fication :none)) 0.5)
          (<= (certainty-factor (tell [interact-with-girl ?guy dance-with-her] :justifi-
cation :none)) 0.9)]
```

```
  then [dance-with-girl ?guy no])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Already dancing section;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                   How do you want to Dance                   ;;
;;                       With Girl Rules                        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; SUCCESSES-USER
(defrule user-wants-hand (:backward :certainty 1.0 :importance 589)
  if [and [user-hand-contact ?guy HANDS-ON-HER-HIPS]
          [want-hand-contact ?guy HANDS-ON-HER-HIPS]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-hand2 (:backward :certainty 1.0 :importance 588)
  if [and [user-hand-contact ?guy HANDS-ON-HER-BUTT]
          [want-hand-contact ?guy HANDS-ON-HER-BUTT]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-hand3 (:backward :certainty 1.0 :importance 587)
  if [and [user-hand-contact ?guy NOT-ON-HER]
          [want-hand-contact ?guy NONE-OF-THE-ABOVE]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-arm (:backward :certainty 1.0 :importance 586)
  if [and [user-arm-contact ?guy ARMS-AROUND-WAIST]
          [want-arm-contact ?guy ARMS-AROUND-WAIST]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-arm2 (:backward :certainty 1.0 :importance 585)
  if [and [user-arm-contact ?guy ARMS-AROUND-CHEST]
          [want-arm-contact ?guy ARMS-AROUND-CHEST]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-arm3 (:backward :certainty 1.0 :importance 584)
  if [and [user-arm-contact ?guy NOT-ON-HER]
          [want-arm-contact ?guy NONE-OF-THE-ABOVE]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-grind (:backward :certainty 1.0 :importance 583)
  if [and [user-grinding ?guy yes]
          [want-to-grind ?guy yes]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-no-grind (:backward :certainty 1.0 :importance 582)
  if [and [user-grinding ?guy no]
          [want-to-grind ?guy no]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-feel (:backward :certainty 1.0 :importance 581)
  if [and [user-feel-her ?guy yes]
          [want-to-feel-her ?guy yes]]
  then [dance-way-you-want ?guy yes])

(defrule user-wants-no-feel (:backward :certainty 1.0 :importance 580)
  if [and [user-feel-her ?guy no]
          [want-to-feel-her ?guy no]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-location (:backward :certainty 1.0 :importance 579)
  if [and [user-floor-wall ?guy yes]
          [want-dance-location ?guy FLOOR]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-location2 (:backward :certainty 1.0 :importance 578)
  if [and [user-floor-wall ?guy no]
          [want-dance-location ?guy WALL]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-position (:backward :certainty 1.0 :importance 577)
  if [and [user-b2f-f2f ?guy yes]
          [want-dance-position ?guy HER-BACK-TO-YOUR-FRONT]]
        then [dance-way-you-want ?guy yes])

(defrule user-wants-position2 (:backward :certainty 1.0 :importance 576)
  if [and [user-b2f-f2f ?guy no]
          [want-dance-position ?guy HER-FRONT-TO-YOUR-FRONT]]
        then [dance-way-you-want ?guy yes])

;; MOVING YOUR HANDS
(defrule move-hands-to-hip (:backward :certainty 0.5 :importance 569)
  if [user-hand-contact ?guy NOT-ON-HER]
        then [put-hands-on-hip ?guy yes])

(defrule move-hands-to-hip2 (:backward :certainty 1.0 :importance 568)
  if [user-hand-contact ?guy HANDS-ON-HER-HIPS]
        then [put-hands-on-hip ?guy yes])

(defrule move-hands-to-hip3 (:backward :certainty 1.0 :importance 570)
  if [user-hand-contact ?guy HANDS-ON-HER-BUTT]
        then [put-hands-on-hip ?guy yes])

(defrule move-hands-to-butt (:backward :certainty 1.0 :importance 567)
  if [and [user-hand-contact ?guy NOT-ON-HER]
          [grind-her ?guy yes]
         [front2front ?guy yes]
         [put-hands-on-hip ?guy yes]]
        then [put-hands-on-butt ?guy yes])

(defrule move-hands-to-butt2 (:backward :certainty 1.0 :importance 566)
  if [and [user-hand-contact ?guy HANDS-ON-HER-HIPS]
          [grind-her ?guy yes]
         [front2front ?guy yes]]
        then [put-hands-on-butt ?guy yes])

(defrule move-hands-to-butt3 (:backward :certainty 1.0 :importance 565)
  if [and [user-hand-contact ?guy HANDS-ON-HER-HIPS]
          [user-grinding ?guy yes]
         [front2front ?guy yes]]
        then [put-hands-on-butt ?guy yes])

        ;; MOVING YOUR ARMS ON WAIST
(defrule move-arm-around-waist (:backward :certainty 1.0 :importance 559)
  if [user-arm-contact ?guy NOT-ON-HER]
  then [put-arms-on-waist ?guy yes])
```

```
(defrule move-arm-around-waist2 (:backward :certainty 1.0 :importance 558)
  if [user-arm-contact ?guy HANDS-ON-HER-HIPS]
  then [put-arms-on-waist ?guy yes])

(defrule move-arm-around-waist3 (:backward :certainty 1.0 :importance 557)
  if [and [user-arm-contact ?guy NOT-ON-HER]
          [put-hands-on-hip ?guy yes]]
  then [put-arms-on-waist ?guy yes])

(defrule move-arm-around-waist4 (:backward :certainty 1.0 :importance 556)
  if [user-arm-contact ?guy ARM-AROUND-WAIST]
  then [put-arms-on-waist ?guy yes])

;;MOVING YOUR ARMS ON CHEST
(defrule move-arm-around-chest (:backward :certainty 1.0 :importance 549)
  if [and [user-arm-contact ?guy NOT-ON-HER]
          [put-arms-on-waist ?guy yes]
         [back2front ?guy yes]]
         then [put-arms-on-chest ?guy yes])

(defrule move-arm-around-chest2 (:backward :certainty 1.0 :importance 548)
  if [and [user-arm-contact ?guy ARM-AROUND-WAIST]
         [back2front ?guy yes]]
         then [put-arms-on-chest ?guy yes])

(defrule move-arm-around-chest3 (:backward :certainty 1.0 :importance 547)
  if [and [user-arm-contact ?guy ARM-AROUND-WAIST]
          [feel-on-her ?guy yes]
         [back2front ?guy yes]]
         then [put-arms-on-chest ?guy yes])

(defrule move-arm-around-chest4 (:backward :certainty 1.0 :importance 546)
  if [and [user-arm-contact ?guy ARM-AROUND-WAIST]
          [grind-her ?guy yes]
         [back2front ?guy yes]]
         then [put-arms-on-chest ?guy yes])

(defrule move-arm-around-chest5 (:backward :certainty 1.0 :importance 545)
  if [and [user-arm-contact ?guy ARM-AROUND-WAIST]
          [user-grinding ?guy yes]
         [back2front ?guy yes]]
         then [put-arms-on-chest ?guy yes])

;;Take girl to wall
(defrule move-to-wall (:backward :certainty 1.0 :importance 535)
  if [and [user-floor-wall ?guy yes]
          [want-dance-location ?guy WALL]
          [or [girl-location ?guy ON-WALL]
              [girl-location ?guy CLOSE-TO-WALL]]
          [size-of-party ?guy crowded]
          [people-action-party ?guy DANCING]
          [not [party-lighting ?guy on]]]
         then [pull-girl-to-wall ?guy yes])

(defrule move-to-wall2 (:backward :certainty 1.0 :importance 534)
  if [and [or [girl-location ?guy ON-WALL]
              [girl-location ?guy CLOSE-TO-WALL]]
          [size-of-party ?guy crowded]
          [people-action-party ?guy DANCING]
         [not [party-lighting ?guy on]]]
         then [pull-girl-to-wall ?guy yes])

;;Take girl to floor
(defrule move-to-floor (:backward :certainty 1.0 :importance 530)
  if [and [user-floor-wall no]
          [want-dance-location ?guy FLOOR]
          [or [girl-location ?guy ON-WALL]
              [girl-location ?guy CLOSE-TO-WALL]]
          [size-of-party ?guy crowded]
          [people-action-party ?guy DANCING]]
         then [pull-girl-to-floor ?guy yes])

(defrule move-to-floor2 (:backward :certainty 1.0 :importance 529)
  if [and [or [girl-location ?guy ON-WALL]
              [girl-location ?guy CLOSE-TO-WALL]]
          [size-of-party ?guy crowded]
          [people-action-party ?guy DANCING]]
         then [pull-girl-to-floor ?guy yes])

;;feel on her
(defrule feel-the-girl-up (:backward :certainty 1.0 :importance 525)
  if [she-rub-hands-on-self ?guy yes]
  then [feel-on-her ?guy yes])

;;front2front
(defrule b2ftof2frule1 (:backward :certainty 1.0 :importance 519)
  if [and [user-b2f-f2f ?guy yes]
          [want-dance-position ?guy HER-FRONT-TO-YOUR-FRONT]
          [turn-her-around ?guy yes]]
  then [front2front ?guy yes])

;;front2front 2
(defrule b2ftof2frule2 (:backward :certainty 1.0 :importance 518)
  if [user-b2f-f2f ?guy no]
  then [front2front ?guy yes])

;;back2front
(defrule f2ftob2frule1 (:backward :certainty 1.0 :importance 515)
  if [and [user-b2f-f2f ?guy no]
          [want-dance-position ?guy HER-BACK-TO-YOUR-FRONT]
          [turn-her-around ?guy yes]]
  then [back2front ?guy yes])

;;back2front 2
(defrule f2ftob2frule2 (:backward :certainty 1.0 :importance 514)
  if [user-b2f-f2f ?guy yes]
  then [back2front ?guy yes])

;; grind her
(defrule grindherrule (:backward :certainty 1.0 :importance 510)
  if [and [user-grinding ?guy no]
          [want-to-grind ?guy yes]
          [put-arms-on-waist ?guy yes]]
  then [grind-her ?guy yes])

(defrule grindherrule2 (:backward :certainty 1.0 :importance 510)
  if [user-grinding ?guy yes]
  then [grind-her ?guy yes])
```

```
;;SUCCESSES
(defrule success-hands (:backward :certainty 1.0 :importance 509)
  if [and [want-hand-contact ?guy HANDS-ON-HER-HIP]
         [put-hands-on-hip ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-hands2 (:backward :certainty 1.0 :importance 508)
  if [and [want-hand-contact ?guy HANDS-ON-HER-BUTT]
         [put-hands-on-butt ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-arm (:backward :certainty 1.0 :importance 507)
  if [and [want-arm-contact ?guy ARM-AROUND-WAIST]
         [put-arms-on-waist ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-arm2 (:backward :certainty 1.0 :importance 506)
  if [and [want-arm-contact ?guy ARM-AROUND-CHEST]
         [put-arms-on-chest ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-grind (:backward :certainty 1.0 :importance 505)
  if [and [want-to-grind ?guy yes]
         [grind-her ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-feel-her (:backward :certainty 1.0 :importance 504)
  if [and [want-to-feel-her ?guy yes]
         [feel-on-her ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-back2front (:backward :certainty 1.0 :importance 503)
  if [and [want-dance-position ?guy HER-BACK-TO-YOUR-FRONT]
         [back2front ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-front2front (:backward :certainty 1.0 :importance 502)
  if [and [want-dance-position ?guy HER-FRONT-TO-YOUR-FRONT]
         [front2front ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-location (:backward :certainty 1.0 :importance 501)
  if [and [want-dance-location ?guy WALL]
         [pull-girl-to-wall ?guy yes]]
         then [dance-way-you-want ?guy yes])

(defrule success-location2 (:backward :certainty 1.0 :importance 500)
  if [and [want-dance-location ?guy FLOOR]
         [pull-girl-to-floor ?guy yes]]
         then [dance-way-you-want ?guy yes])

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                            ;;;
;;; -------------------END OF RULES-----------------------------;;;
;;;                                                            ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```