James Tolbert, II
6.871 Term Paper
5/12/05

Groove-X
Dancing the Way You Want

1. Introduction

We sat in the room one night mauling over what type of expert system we wanted to create. We knew we had knowledge about interacting with girls and we liked to party. It was only a natural conclusion that we would find knowledge about guy and girl interactions at a party; then use this knowledge to advise ourselves and other guys about whether they have a chance with a girl and how to dance with the girl the way they wanted to. Groove-X was born.

2. What task does it do?

The task of our system is to advise a guy about his interactions with girls at a party. This task had two primary goals and one secondary goal. The first primary goal is to advise a guy about his chance of dancing with a girl. The next primary goal is that if he satisfies the first one by dancing with the girl, then the system will advise the guy about how to dance with a girl they way he wants to. The secondary goal, which can be asked after the first primary goal is solved, is to advise a guy about the possible interactions he can have with the girl which include dancing with the girl the way that he wants to.

Since the task has multiple goals, there are two parts to the following example of our system. The first part is an example of input and output of the first primary goal while the second part is an example of input and output of the second primary goal. For the first part, the input can be described as follows:

> Jim is wearing an Ecko shirt and jeans with Timberland shoes on his feet and a flat peaked hat on his head. His shirt matches the color of his shoes. Also, Jim's clothes are clean without stains or holes on them. His Timberland shoes are crisp because they aren't dirty, there aren't any visible wrinkles on them, and they aren't scuffed. Jim took a shower before the party and has cologne on.
>
> Jim makes eye contact with the girl for a short while during the party. At the middle of the party he approaches the girl. He makes sure she doesn't notice him coming as he approaches her. The party has dim lighting. It is also crowded with people dancing. The girl doesn't appear to have any friends around.

The corresponding output would be:

> Groove-X indicates that Jim should dance with the girl.

For the second part, the input can be described as follows:

> Jim wants to grind with the girl on the wall with his back against the wall. Jim decides to dance with the girl and they have started to dance. They are dancing close to the wall, and the party is crowded. A song that is playing has a consistent beat and the majority of the crowd is dancing couple wise. Jim has his arms around her waist and they are dancing with the back of her body rubbing against the front of his body

The corresponding output would be:

> Groove-X indicates that Jim can grind the girl with his back against wall. He needs to continue to dance with the girl with her back against his front, and pull her by the waist towards the wall until his back is against the wall. Then he can grind on the wall.

3. Knowledge Acquisition

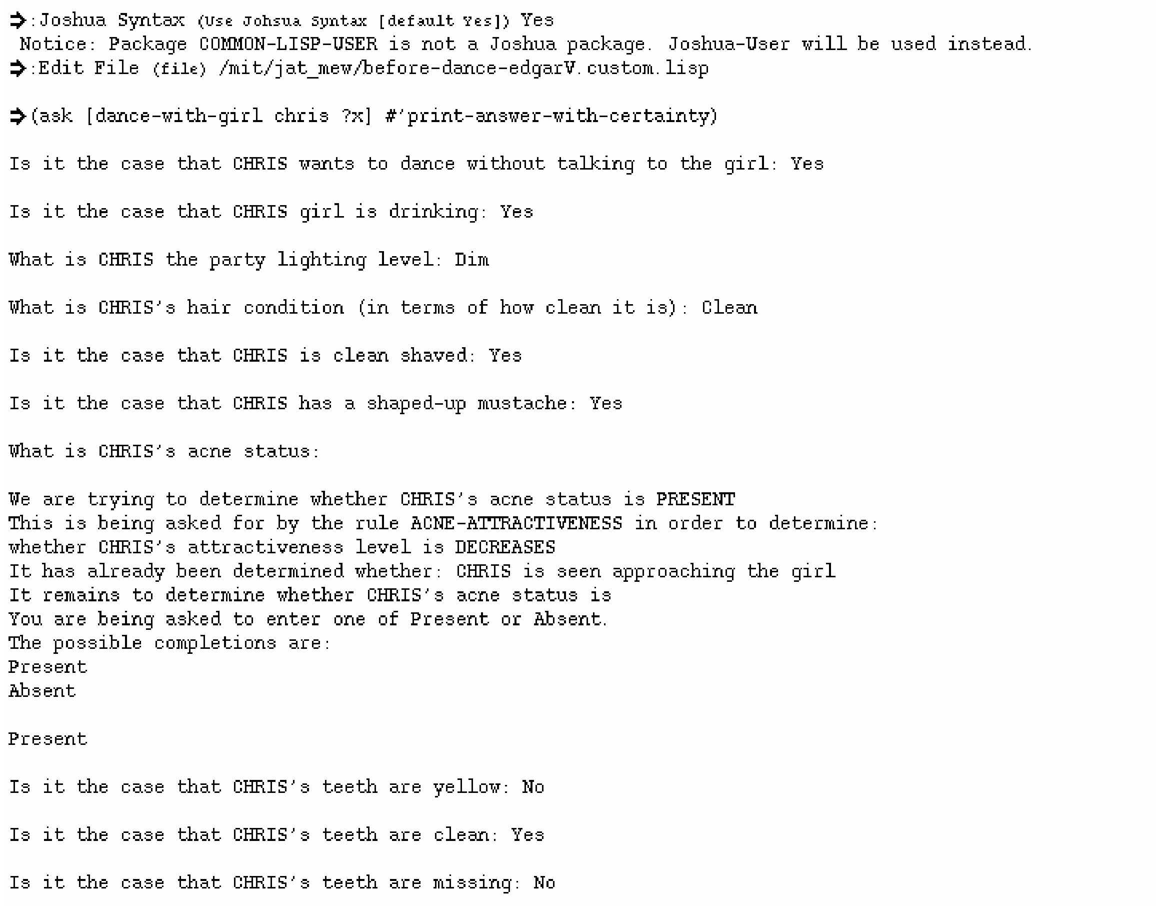
Not only is the knowledge representation of our system interesting, but also the process of our knowledge acquisition. We decided the best way to understand a guy chance to dance with a girl was to interview people who went to parties. Moreover, we didn't focus on one gender when interviewing about a guy chances. We figured that if compare both gender's viewpoint, that we would find a lot knowledge supported by both sides. Moreover, our knowledge base was exhaustive because we had information that only the guy or the girl would think about which actually affected the guy's success rate of dancing with the girl.

Edgar focused on interviewing the guys and I focused on interviewing the girls. Interviewing women was interesting for several reasons. First, I had to find a female expert who I could trust to be candid about knowledge that affected a guy's chance with a girl. I remember several times, I would hear a girl tell me they had no clue about what affected whether a guy had a chance to dance with them. Second, I needed to interview women who I knew were good at dancing as well as women who were bad at dancing. Another interesting part of interviewing was to see how comfortable people were in giving me answers, especially when I was collecting knowledge about how a guy can get a girl to dance the way he wants.

Not only did we interview people, but we went to different parties and made observations. We were able to use observations to confirm what people said in our interviews and also to create additional questions to ask people that we were interviewing.

4. Walk through one real, annotated problem.

The following figures are snapshots of a user interaction with our system.

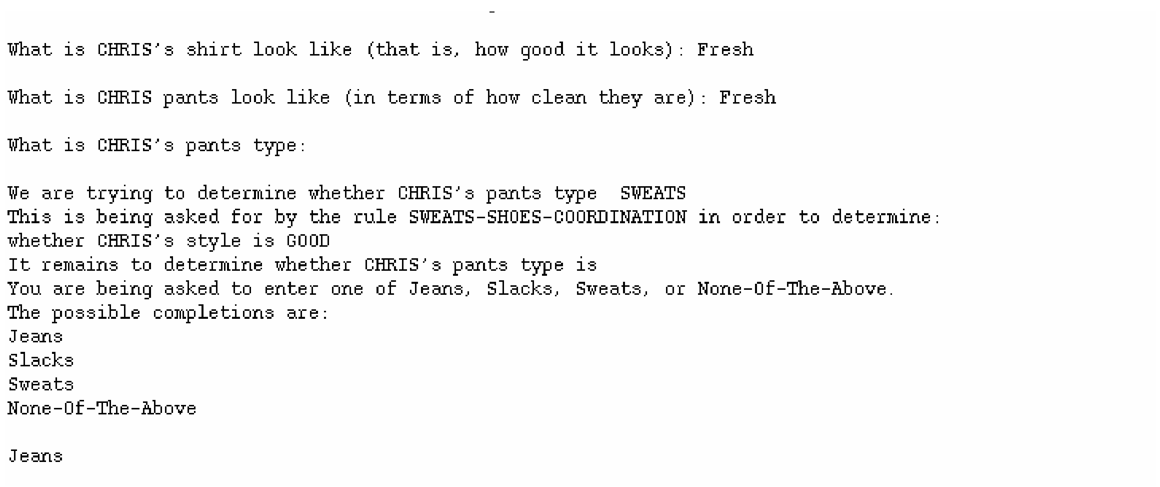


```
➔:Joshua Syntax (Use Johsua Syntax [default Yes]) Yes
 Notice: Package COMMON-LISP-USER is not a Joshua package. Joshua-User will be used instead.
➔:Edit File (file) /mit/jat_mew/before-dance-edgarV.custom.lisp

➔(ask [dance-with-girl chris ?x] #'print-answer-with-certainty)

Is it the case that CHRIS wants to dance without talking to the girl: Yes

Is it the case that CHRIS girl is drinking: Yes

What is CHRIS the party lighting level: Dim

What is CHRIS's hair condition (in terms of how clean it is): Clean

Is it the case that CHRIS is clean shaved: Yes

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

What is CHRIS's acne status:

We are trying to determine whether CHRIS's acne status is PRESENT
This is being asked for by the rule ACNE-ATTRACTIVENESS in order to determine:
whether CHRIS's attractiveness level is DECREASES
It has already been determined whether: CHRIS is seen approaching the girl
It remains to determine whether CHRIS's acne status is
You are being asked to enter one of Present or Absent.
The possible completions are:
Present
Absent

Present

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

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

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

Figure 1

The user starts off by asking the system the question should he dance with the girl by typing (ask [dance-with-girl user ?x] #'print-answer-with-certainty) as shown in the Figure 1. The system looks to find knowledge about this question, and the first piece of knowledge is the guy needs to tell the system whether he wants to dance with the girl without talking to her first. Since the user inputted yes, our system will not ask questions that involve the user having a conversation with the girl.

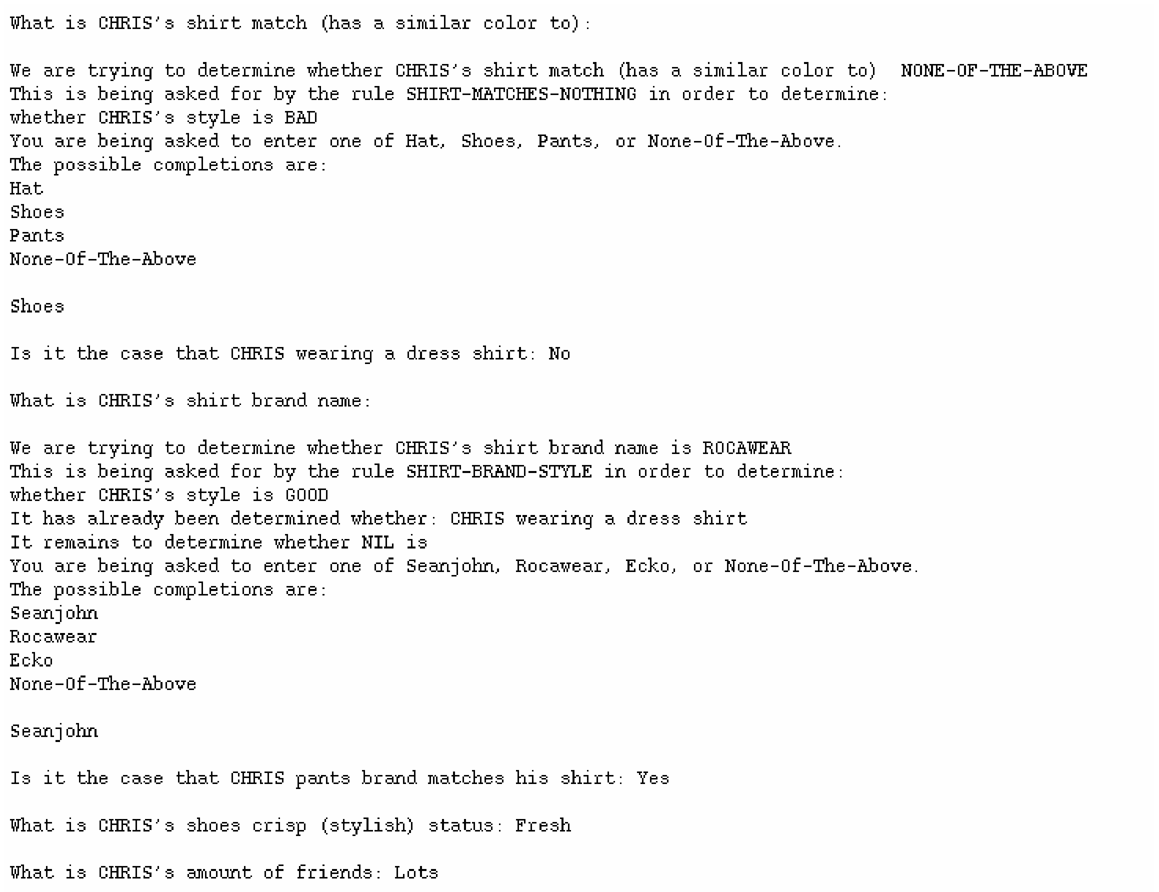
The questions following the user dance preference about whether to converse with the girl are all concerned with the girl attitude towards the guy. The first question asked about the girl attitude is about drinking because drinking affects her attitude from the start. The question about party lighting is asked next because it is used to narrow down the questions asked about guy's appearance because certain things the girl can't notice when the party has dimmed lighting or the lighting is off. After asking these questions, the system wants to know if the user is making a good or bad impression on the girl. Therefore, the system needs to know what can make an impression on the girl and that is based on the attractiveness of the guy. Several different things play a role in how attractive a guy is to the girl: guy's appearance, hygiene, style, and dancing skills. First, the system asked the guy about appearance such as his hair being clean, not having visual acne, and teeth being clean so he has a nice smile.



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

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

What is CHRIS's pants type:

We are trying to determine whether CHRIS's pants type  SWEATS
This is being asked for by the rule SWEATS-SHOES-COORDINATION in order to determine:
whether CHRIS's style is GOOD
It remains to determine whether CHRIS's pants type is
You are being asked to enter one of Jeans, Slacks, Sweats, or None-Of-The-Above.
The possible completions are:
Jeans
Slacks
Sweats
None-Of-The-Above

Jeans
```

Figure 2

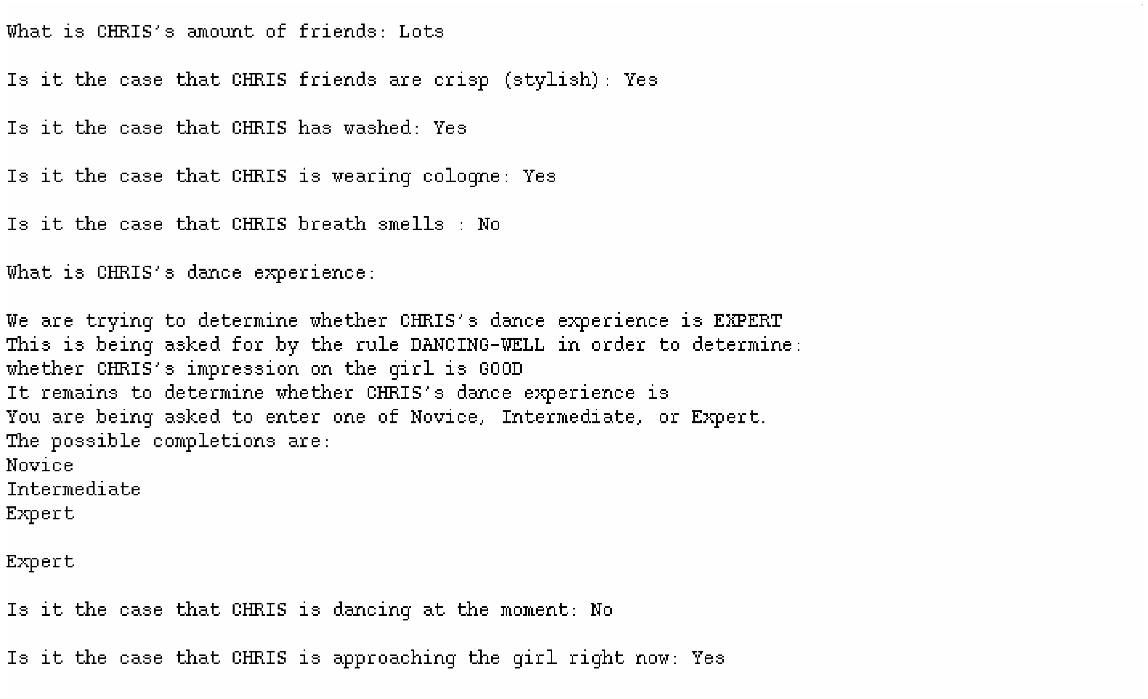
To finish these questions about appearance, the system ask the guy what is the status of the clothes that he is wearing. If he doesn't know, the system gives the guy the option of inputting 'I don't know' as answer to the questions. The system will then give the guy more detailed questions about clothing status such as his shirt having stains or holes.



```
What is CHRIS's shirt match (has a similar color to):

We are trying to determine whether CHRIS's shirt match (has a similar color to)  NONE-OF-THE-ABOVE
This is being asked for by the rule SHIRT-MATCHES-NOTHING in order to determine:
whether CHRIS's style is BAD
You are being asked to enter one of Hat, Shoes, Pants, or None-Of-The-Above.
The possible completions are:
Hat
Shoes
Pants
None-Of-The-Above

Shoes

Is it the case that CHRIS wearing a dress shirt: No

What is CHRIS's shirt brand name:

We are trying to determine whether CHRIS's shirt brand name is ROCAWEAR
This is being asked for by the rule SHIRT-BRAND-STYLE in order to determine:
whether CHRIS's style is GOOD
It has already been determined whether: CHRIS wearing a dress shirt
It remains to determine whether NIL is
You are being asked to enter one of Seanjohn, Rocawear, Ecko, or None-Of-The-Above.
The possible completions are:
Seanjohn
Rocawear
Ecko
None-Of-The-Above

Seanjohn

Is it the case that CHRIS pants brand matches his shirt: Yes

What is CHRIS's shoes crisp (stylish) status: Fresh

What is CHRIS's amount of friends: Lots
```

Figure 3

After asking questions about his appearance, the system asks questions to find out if he has any style. Therefore it asks questions concerning whether the guy is wearing name brand clothing as well as his clothes matching.

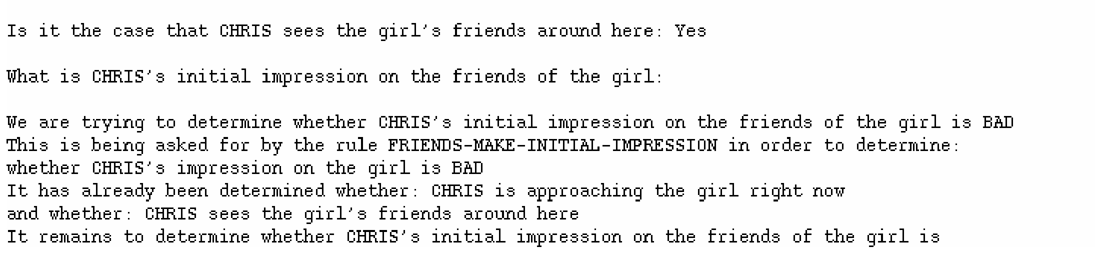


```
What is CHRIS's amount of friends: Lots

Is it the case that CHRIS friends are crisp (stylish): Yes

Is it the case that CHRIS has washed: Yes

Is it the case that CHRIS is wearing cologne: Yes

Is it the case that CHRIS breath smells : No

What is CHRIS's dance experience:

We are trying to determine whether CHRIS's dance experience is EXPERT
This is being asked for by the rule DANCING-WELL in order to determine:
whether CHRIS's impression on the girl is GOOD
It remains to determine whether CHRIS's dance experience is
You are being asked to enter one of Novice, Intermediate, or Expert.
The possible completions are:
Novice
Intermediate
Expert

Expert

Is it the case that CHRIS is dancing at the moment: No

Is it the case that CHRIS is approaching the girl right now: Yes
```

Figure 4

To finish off questions about a guy's attractiveness, the system asks questions about his hygiene and then his dance experience. All of the previous answers the guy gives are use to find how attractive the guy is to the girl, then his attractiveness is used to find what impression he made on the girl, and then the impression is used to find out if the girl attitude is being interested or not-interested in the guy. Therefore, the more attractive the guy is, the more she is impressed by him, the more she is interested in him and the more the system increases his chances of dancing with the girl. The same idea can be said about her being not interested and the system decreasing his chances of dancing with the girl.

The system then asked about whether the guy is dancing at the moment. This question is kind of awkward because the next question asked is whether he is approaching her. If he is approaching her, then he can't be dancing at the moment unless he dancing towards her. The system should flip the order of these two questions so it knows whether he is dancing with someone or is actually dancing towards her.

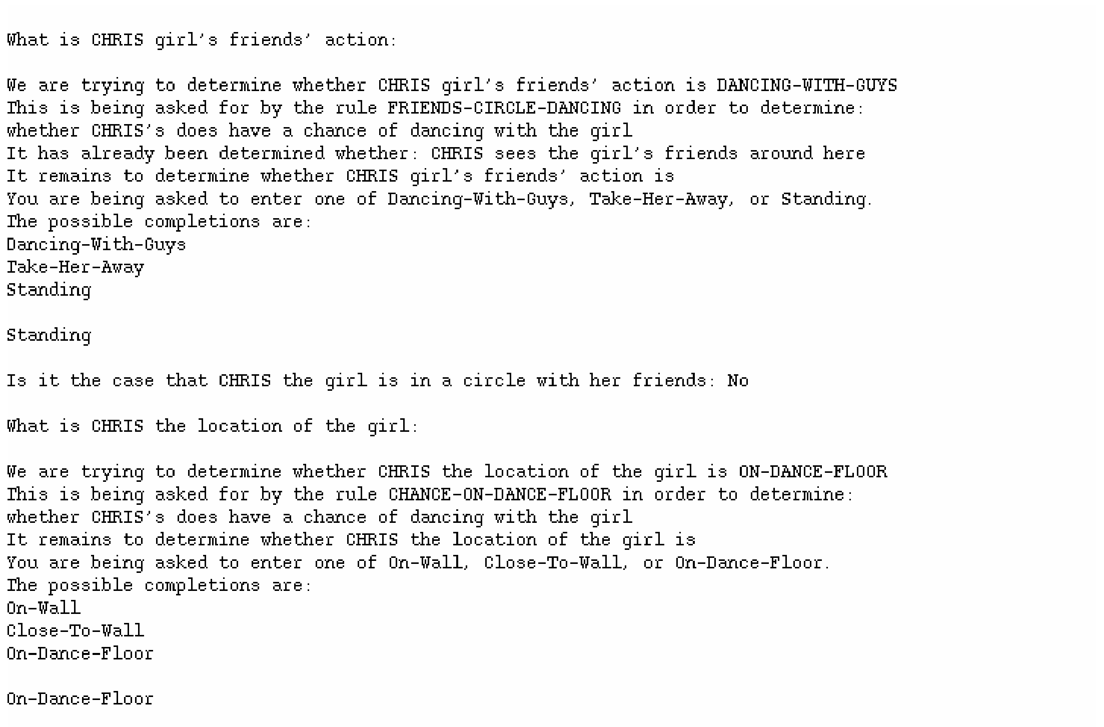


```
Is it the case that CHRIS sees the girl's friends around here: Yes

What is CHRIS's initial impression on the friends of the girl:

We are trying to determine whether CHRIS's initial impression on the friends of the girl is BAD
This is being asked for by the rule FRIENDS-MAKE-INITIAL-IMPRESSION in order to determine:
whether CHRIS's impression on the girl is BAD
It has already been determined whether: CHRIS is approaching the girl right now
and whether: CHRIS sees the girl's friends around here
It remains to determine whether CHRIS's initial impression on the friends of the girl is
```

Figure 5

Once the system knows that the guy is approaching her, it's going to want to ask questions about how she responds to him in conversation. However, since the guy stated that he doesn't want to be in a conversation with the girl before he dances with her, the system skips these questions and asked questions concerning the girl's comfortableness with him. Examples of questions about her comfortableness are whether he is looking the girl up and down or if they made eye contact. These questions about comfortableness increase or decrease his chance of dancing with the girl separately from questions concerning the girl's attitude.

The system asks questions about the party environment after it has asked all the questions it can about the girl's attitude and the girl's comfortableness with the guy.



```
What is CHRIS girl's friends' action:

We are trying to determine whether CHRIS girl's friends' action is DANCING-WITH-GUYS
This is being asked for by the rule FRIENDS-CIRCLE-DANCING in order to determine:
whether CHRIS's does have a chance of dancing with the girl
It has already been determined whether: CHRIS sees the girl's friends around here
It remains to determine whether CHRIS girl's friends' action is
You are being asked to enter one of Dancing-With-Guys, Take-Her-Away, or Standing.
The possible completions are:
Dancing-With-Guys
Take-Her-Away
Standing

Standing

Is it the case that CHRIS the girl is in a circle with her friends: No

What is CHRIS the location of the girl:

We are trying to determine whether CHRIS the location of the girl is ON-DANCE-FLOOR
This is being asked for by the rule CHANCE-ON-DANCE-FLOOR in order to determine:
whether CHRIS's does have a chance of dancing with the girl
It remains to determine whether CHRIS the location of the girl is
You are being asked to enter one of On-Wall, Close-To-Wall, or On-Dance-Floor.
The possible completions are:
On-Wall
Close-To-Wall
On-Dance-Floor

On-Dance-Floor
```

Figure 6

The range of questions the systems ask about party is very broad. It includes the party lighting, party phase, the song speed, and what people are doing to name a few things that the system wants to know about. Figure 6 shows the user answer to the question about what the girl's friends are doing at the party. When he answers that her friends are dancing with guys, the system asks if they are in a circle because his chance are extremely high if the girl is the only one in her group of friends that isn't dancing with a guy.

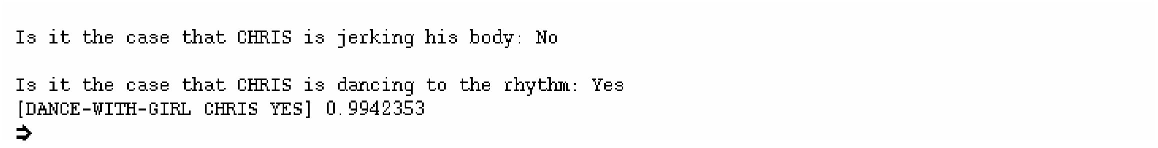

```
Is it the case that CHRIS is jerking his body: No

Is it the case that CHRIS is dancing to the rhythm: Yes
[DANCE-WITH-GIRL CHRIS YES] 0.9942353
➔
```

Figure 7

Once the system has asked all of the questions it can about the party environment, it increases or decreases the guy chance of dancing with the girl. Since the scenario of the girl being the only one among her friends not dancing with a guy, the chance of the guy dancing with the girl is extremely high, regardless of her attitude, and the system advises the guy to dance with the girl.

Once the system had told the guy whether he should dance with the girl, the guy can ask what to do to get her to dance with him the way he wants by typing (ask [dance-way-you-want user ?x] #'print-answer-with-certainty). The system asks him two main set of questions. First, it asked questions concerning how he is dancing with the girl at the moment. Where are his hands and arms? Is he dancing with her back against his front or dancing with his front body facing her front body at the moment? The second set of questions is concerned with how he wants to dance with the girl. Does he want to grind is one possible questions asked by the system. Our system gives him an output of how certain it is that he can dance the way he wants with the girl.

5. What range of problems can the system handle?

Our system can handle problems that focus on how a guy's style, appearance, hygiene, and action in relation to a girl's attitude effects a guy chance to dance with girl. It can also handle the problems where party environment influences the guy's chance to dance with the girl. Below, I will give examples of what problems our system actually solved and then the ones it can solve.

The system solved the problem of knowing what style of pants should be worn with what shoes. It solved the problem of matching pieces of clothing. Our system solved the problem of what criteria is needed for a guy to have nice smile compared to a busted smile. It solved the problem of recognizing when a girl is ignoring a guy.

It can solve the problem dealing with a guy who just threw on dirty clothes when going to the party. It can solve the problem that the style of dressing considered cool at party is based on the style of dress in NYC. It can solve the problem of what type of an attitude a guy should have when starting a conversation. It can solve the problem of when a guy should leave a girl alone when she isn't responding to him.

The previous examples were problems the system can handle and actually solved. I will now discuss several problems the system can't handle. A problem that our system can't solve is the chance of dancing with a girl after you have dance with her friend. The system does know whether a guy is dancing at the moment but it doesn't specify whether he is dancing alone or not. Moreover, the system does know when a girl's friends are dancing with guys but it doesn't know if the user is one of the guys or not. This problem is interesting because brings up several questions How many girls can you dance with that know each other at a party? What happen if you want to dance with a girl who is with her friends and her friend is the only girl not dancing with guys? Is it beneficial to dance with her friend and wait to the next song to play so that you when the girl is alone you can leave her friend to dance with her?

Another interesting problem also encompasses knowing the gender of the girl's friends. Our system assumes the girl is always with friends who are of the same sex. Therefore, our system wouldn't be able to handle the situation when the girl comes to party with male friends or her boyfriend. It would be interesting to see if the chance of dancing with the girl is affected in some way depending on the sex of the friends that are with her.

6. What does it know?

We used rules as our knowledge representation. We felt that rules were a concise way of expressing knowledge because our knowledge could be represented efficiently as predicates containing an attribute, object, and value. This predicate representation is based on problem set 2 of the class. The following figures show a complete representation of our knowledge base. I chose to show the representation over several tree structures. Moreover, the nodes of the tree represent what different sections of rules in our system are about. The parent node is the goal and child node is the information use to reach that goal.

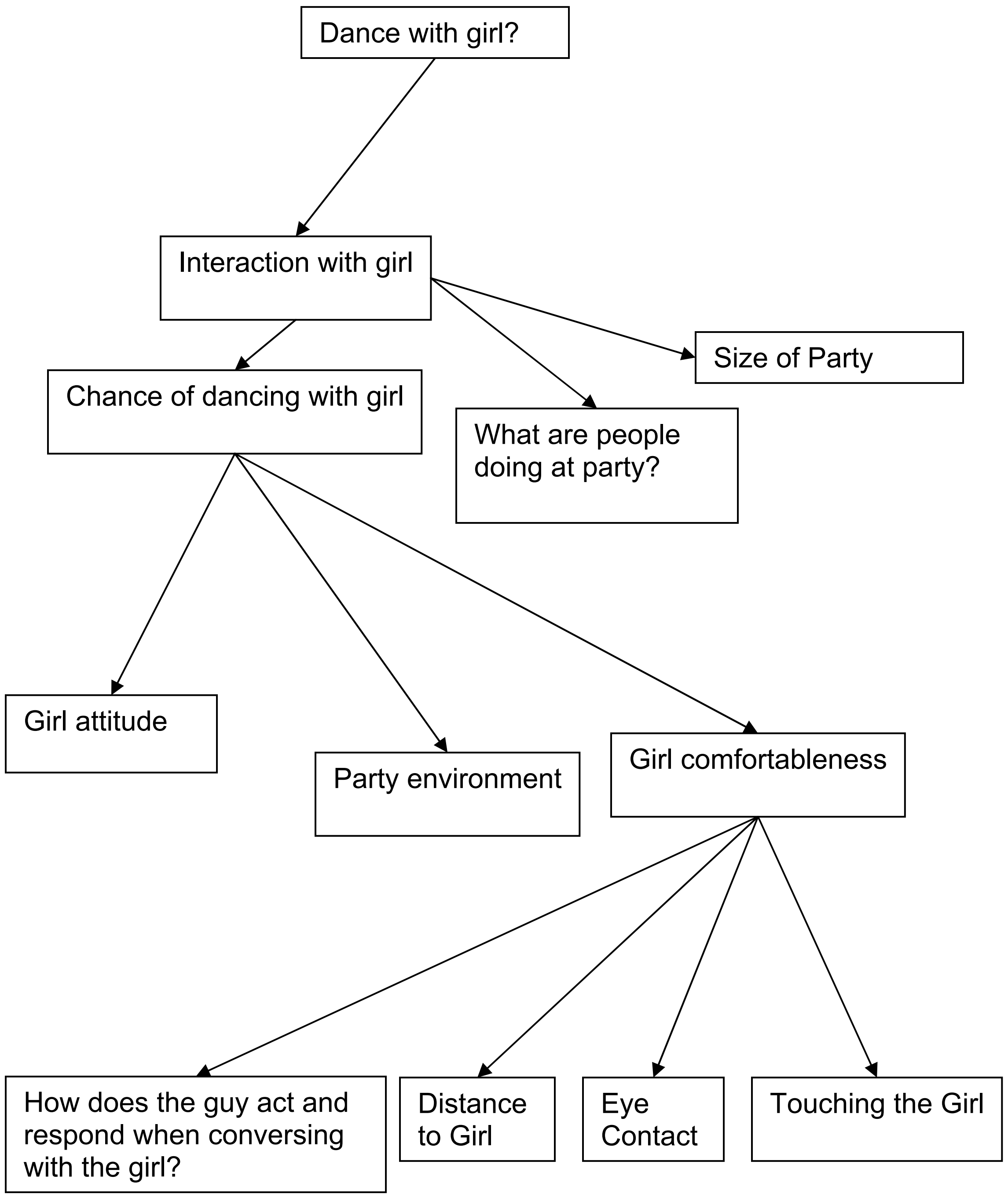


Figure 1: Dance with Girl as well as Girl's comfortableness

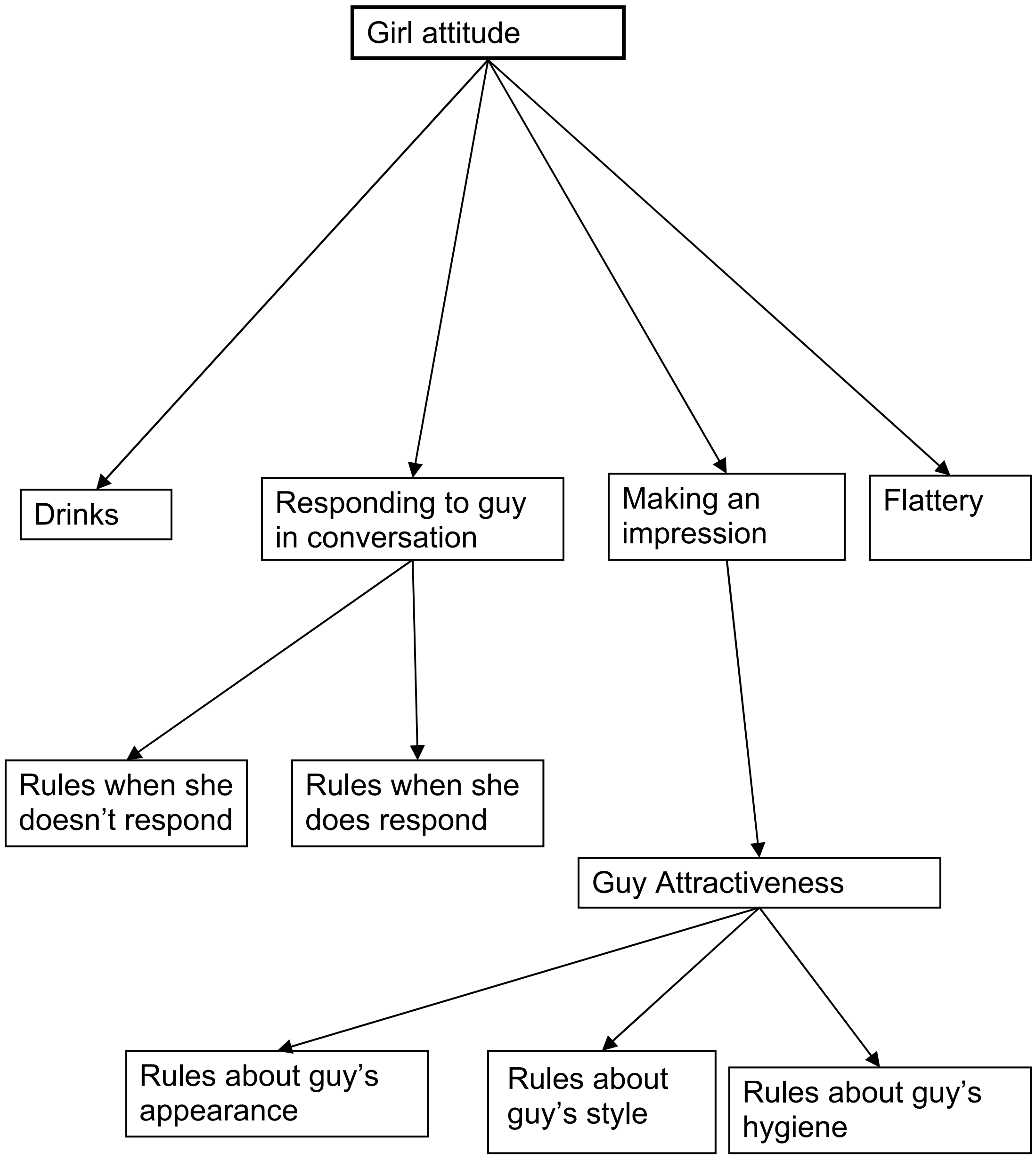

Girl attitude
Drinks
Responding to guy in conversation
Making an impression
Flattery
Rules when she doesn't respond
Rules when she does respond
Guy Attractiveness
Rules about guy's appearance
Rules about guy's style
Rules about guy's hygiene

Figure 2: Girl's Attitude

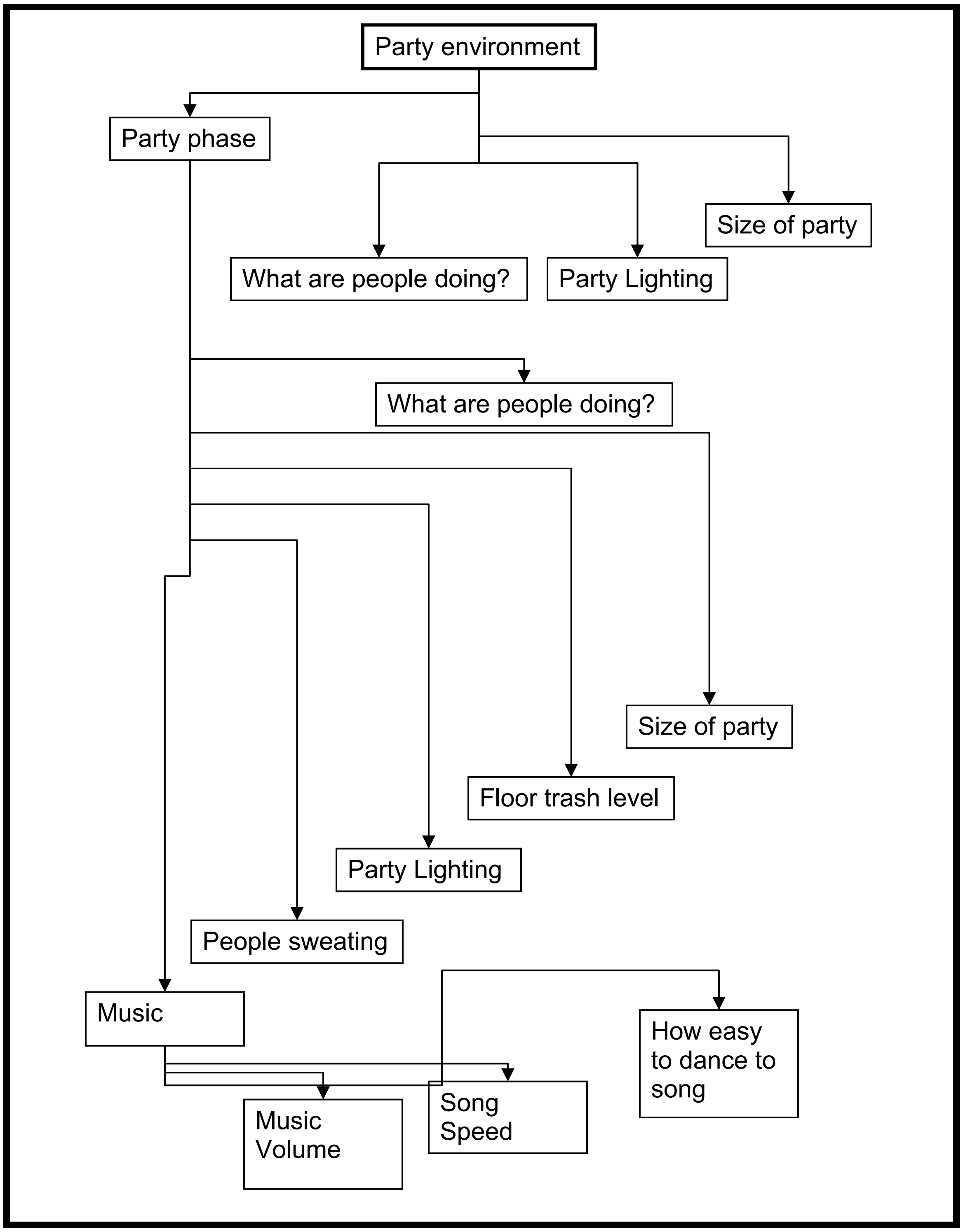


Figure 3: Party Environment

7. How does it work?

Problem Solving Paradigm

When picking the problem solving paradigm for our rule-based system, we had three options: forward chaining, backward chaining, or both. Using the KISS (KEEP IT SIMPLE STUPID) approach when deciding our problem solving paradigm, we eliminated the both option because we felt it would be hard to debug our knowledge representation. We chose backward chaining over forward chaining for several reasons. We were familiar with this paradigm from problem set 2. Moreover, our knowledge base can be easily walked through by backward chaining. We could stop asking unnecessary questions when backward chaining since the first predicate failing in an 'and' conjunction antecedent would mean the rest of predicates weren't asked. Moreover, the point that Edgar brought up when we talk about using backward chaining is that when we interviewed people the knowledge we got from them was in a rule format. If you do something, then your chance of dancing with a woman would decrease or increase.

Applying the Problem Solving Paradigm

The user starts off by asking the question should he dance with the girl by typing (ask [dance-with-girl user ?x] #'print-answer-with-certainty). Since the system doesn't know any information about the party, girl, or guy at the start, it is going to backward chain to see if it can answer whether it knows how the guy is interacting with the girl. Since that isn't known either, the system backward chains again to see if it knows any knowledge about the guy's chance to dance with the girl.

The system finds every rule that affects the chance of the guy dancing with the girl. These rules are split into three sections: girl attitude, party environment, and the girl comfortableness. I will discuss these sections in order. The first section of rules that affect a guys chance that our system backward chains through are rules that affect the girl's attitude. If she has positive attitude such as being interested in the guy, then the chance of dancing with the girl increases with some certainty factor. If she has a negative attitude towards the guy such as not being interested, then the chance of dancing with the girl decreases with some certainty factor.

There are multiple rules that affect the girl's attitude. The system backward chains through each group of rules that affect her attitude in this order: rules concerning the girl's drinking habits, rules about how responsive the girl is when they are in a conversation, rules concerning what impression the guy makes on the girl, and rules concerning whether the guy tries to flatter the girl. Rules concerning a girl's responsiveness to the guy in a conversation are split into two sections: ways that she does respond to the guy, and ways she doesn't respond to, ignore, the guy.

Next, the system applies backward chaining to the rules that make an impression on the girl. This group of rules leads to rules that either increase or decrease the guy's attractiveness using certainty factors. To find out whether to increase or decrease his attractiveness, the system fire rules about the guy's appearance, hygiene, style, and dance skills. These rules complete the path that makes an impression on the girl's attitude as follows:

Guy's Hygiene, Style, Conversational Skills, Appearance → Guy's Attractiveness
Guy's Attractiveness → Makes Impression → Girl's Attitude

The system finishes the girl attitude section by checking rules that deal with whether the guy flatters the girl and then it moves to the second section of rules to determine a guy's chance.

These rules are focused on answering how comfortable the girl is with the guy. Therefore, the system ask questions including the distance to the girl, how long the eye contact is made between girl and guy, as well as the guy looking the girl up and down. Each of these questions increases or decreases the girl's comfortableness with the guy. Every time the girl's comfortableness increases, the guy's chance of dancing with the girl increases with a certainty factor. Also, as the girl's comfortableness decreases, the guy's chance of dancing with the girl decreases with a certainty factor.

The third section of rules is about the party environment and they try to answer five key questions. What is the party phase? What are people doing at the party? What is the size of the party? What is the party lighting? What is happening with the music? Once our system backward chains through these three sections, it uses the chance of dancing with girl to find out with what certainty a guy should dance with the girl, leave her alone, or talk to her. Then the system compares the certainties of each interaction to a threshold and based on these comparisons decides whether to tell the guy that he should or shouldn't dance with the girl.

Two aids in our backward chaining strategy is the implementation of Meta rules and giving the user the choice on certain questions to answer 'I don't know'. Our system uses Meta rules to cut down the rules fired and therefore the questions that our system needs to ask. As described in the walk through, if the user answer yes to the question that he wants go and start dancing with the girl without talking to her first, our system doesn't ask questions about him having a conversation with her since we assume that he is not going to. Our system uses 'I don't know' answer to know whether it needs to ask the user more detailed questions to solve a certain rule.

Besides backward chaining, Meta rules, and the option 'I don't know', part of our problem solving strategy was to exhaustively connect our knowledge across our different rules. Connecting rules was important because we wanted to control what questions should be asked, the order of questions to be asked, and making sure our system didn't ask the same questions repetitively. If the knowledge wasn't connected then the Meta rules as well as utilizing the option 'I don't know' wouldn't have been effective.

Our system also tries to solve how a guy can dance with a girl they way he wants to. The strategy to solve this problem was to backward chain through rules stating how a guy can get from the way he was dancing currently with a girl to the way he wants to dance. The system used pass knowledge from when the guy was asked if he should dance with the girl along with asking questions about how is dancing at the moment and the way he wanted to dance. Our system tells the guy how certain it is that he can dance with the girl the way he wants. (Note: The part of our system that deals with this task is partially implemented)

8. What Happened?

What Went Well?

As stated, one of the reason we were well-served by our knowledge representation is that the knowledge that we collected was already in rules format. When interviewing people they express there points in if-then format. We were well-served by our tools because we didn't have to code a rule-based system from scratch. Therefore, we could focus our attention on transferring our knowledge into rules and connecting the rules. Another thing that went well was the user testing of our system. My friend who tested the system loved that we created such an expert system. Moreover, people who we interviewed were also extremely interested in the fact that we were creating this expert system and they were willing to divulge knowledge to us about our task.

What Went Badly?

There were many things that went badly even though we had pretty good working system. First, the time to connect the rules was painstakingly long. Since we had a huge database of rules, it always seemed that we had missed a connection between certain rules; therefore, certain questions were asked when we didn't want them to be. Moreover, we continue to find bugs until we turned our project in. I believe the reason for this problem was because we had a large amount of knowledge and rules. Another problem we ran into was misunderstanding how to implement Meta rules into our system. We would think we had implemented the Meta rule and somehow a question that shouldn't be asked was asked.

Besides problems that resulted from having a large amount of knowledge, we felt that Joshua documentation was sub par. The documentation told us how to use Joshua but it didn't explain everything we wanted. For instance, certainty-factors which I assume are pretty substantial function in Joshua wasn't in the reference manual or the user manual. Not only were we ill-served by Joshua but also Lisp. When we had a compiler problem in Lisp, it was a pain to debug because it wasn't clear where the problem went wrong.

The hardest knowledge to capture in rules was knowledge about how a guy can dance with a girl the way he wanted. What made it hard was that once a guy is dancing with a girl, it is highly possible that he can do everything he wants by following different paths. The problem was that these path were circular. Therefore, it was hard to control what rules were fired and what questions were asked. On top of this circular reasoning, we wanted to give the guy steps to follow which was hard to capture in rules. I think we if we chose the second task of our system to be implemented in frames, then solving the problem of getting the girl to dance with you the way you want would have been easier.

Another problem with rules is that it is hard to capture the idea of something increasing or decreasing. Thankfully, we were saved by the certainty factor but our approach with certainty factor wasn't easy to implement. Applying the increasing/decreasing idea in certainty factors didn't always give us the right answer because rules were fired in our system when least expected therefore changing the certainty. Since it was tough spot these rules in a large system, we instead used comparisons as stated above to get the system to work like we wanted it to.

One problem that was easy to correct but hard to recognize was that we shouldn't put predicates in the antecedent and consequent of a rule. This implementation led to undesired results such as infinite looping. This infinite looping became a huge problem when Joshua tracing was enabled.

The Dreaded 'or' conjunction in Joshua drove me crazy. It never worked like you wanted it to. We tried to put the 'or' in the antecedent and we assumed it would work like the 'or' in other programming languages. If the first rule fires than the other rules aren't asked. However, this assumption was wrong. To get around this problem we just split a rule into multiple rules based on the number of predicates in the ‘or’ conjunction. However, this approach was very time consuming when had lot of rules the used the ‘or’ conjunction

9. What else did you learn from this?

I learned many things from creating an expert system. First, you have to be careful not to do too much. Doing too much is trying to implement all the rules in your system at once. The problem with this approach is that it’s hard to find bugs and even harder to connect the rules to make system actually act like an expert. Therefore, implement the system in stages.

The second thing I learned was that rules are powerful if you know how to use them. Once we understood how Joshua interpreted rules, we were able to overcome many of things that went badly with our system such as handling Meta rules. Third, knowledge acquisition isn’t the hard part in knowledge engineering, its constructing the knowledge representation. If I had known this, I would have started writing the rules earlier. Moreover, I would have tried to break the knowledge that we had into subsections and tested each subsection. Than put the subsections together as a whole.

Another thing I learned is that when creating an expert system, it is never complete. There is always room for more knowledge to be inputted. Moreover, it is hard to know when you have satisfied your expectations for your expert system. You always find something that you want to implement into your system.

When creating rules, it is very easy to fall into a trap of having a system full of breadth instead of depth. Furthermore, breadth extends the number of questions that a user is asked. For instance, in our system we first had everything affecting directly a guy’s chance to dance with a girl. Than we realize that certain set of rules don’t directly affect a girl’s chance but affect another rule which does affect a girl’s chance. A perfect example is a girl attitude and making an impression. We had making impression directly affecting a chance with a girl until we realized that making an impression affects a girl attitude and her attitude is really the thing that makes her want to dance with a guy.

Finally, just because you have a lot of rules don’t mean that you have an expert system. If they are all implemented with breadth instead of depth, a system is nothing but a lookup table. Moreover, those rules might not be connected so your system would not be able to guess smart solutions to your problems. One solution that I used was to create an outline of how one piece of knowledge should be inferred from another piece of knowledge. Once the outline was created, it was easy to put knowledge into rules.

Appendix: Rules

```
 (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])

 (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 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])

(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])

 (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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

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

(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])

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

(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])

(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])

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

(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])


(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])


(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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] :justification :none)) 0.9)
          (< (certainty-factor (tell [interact-with-girl ?guy leave-
her-alone] :justification :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] :justification :none)) 0.5)
          (<= (certainty-factor (tell [interact-with-girl ?guy dance-
with-her] :justification :none)) 0.9)]
  then [dance-with-girl ?guy no])


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

(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])

(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])

(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])

(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])

(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])

(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])

(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])

(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])

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

```
(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])

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

(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])

(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])
```