CPSC 312
Functional and Logic Programming

November 6, 2014

Administrative stuff

- Project 1 due 6:00pm tomorrow, November 7

From the previous lecture

- Arithmetic in Prolog

Database manipulation

A Prolog program can be viewed as a database: a specification of a set of relations given as facts and rules.

We might encounter situations where we'd like to update the database, either by adding facts or rules, or by deleting facts or rules.

The built-in predicates `assert` and `retract` allow us to do exactly that.

Make no mistake however -- these predicates take us way outside of Pure Prolog.

Database manipulation

```
?- assert(<some clause>).
```

Adds the clause to the corresponding procedure. Where? In SWI-Prolog, the clause is added to the end of the procedure.

```
?- asserta(<some clause>).
```

Adds the clause to the beginning of the corresponding procedure.

```
?- assertz(<some clause>).
```

Adds the clause to the end of the corresponding procedure.

Database manipulation

```
?- retract(<some clause>).
```

Finds the first clause in the database that matches (i.e., unifies with) `<some clause>` and removes it from the database. (How that's implemented may vary from one Prolog implementation to another.)

Database manipulation

`assert` and `retract` work only on dynamic predicates. So far, everything you've created is a static predicate. Built-in predicates are also static. If you want a predicate to be modifiable, you need to declare it as such:

```
:- dynamic <pred_name>/<arity>, … ,
           <pred_name>/<arity>.
```

Here's an example:

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r66,r67).
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).


?- retract(connects_to(r66,r67)).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).



?- retract(connects_to(r66,r67)).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).



?- retract(connects_to(r66,r67)).

true
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).



?- assert(connects_to(r66,r67)).
```

Database manipulation

```
:- dynamic connects_to/2.

connects_to(r11,r12).
connects_to(r12,r13).
    :
    :
connects_to(r56,r66).
% below is the one
connects_to(r67,r57).
    :
    :
connects_to(r77,r87).
connects_to(r87,r88).
connects_to(r66,r67).

path(X,Y) :- connects_to(X,Y).
path(X,Y) :- connects_to(X,Z),path(Z,Y).


?- assert(connects_to(r66,r67)).

true
```

Database manipulation

Things to keep in mind:

- `assert` and `retract` are computationally expensive (`assert` compiles its clause; `retract` has to decompile the program to unify its clause before retracting)

- "Excessive and careless use of these facilities cannot be recommended as good programming style….relations that hold at some point will not be true at some other time. At different times the same questions receive different answers. The resulting behaviour of the program may become difficult to understand, difficult to explain and to trust." [from *Prolog Programming for Artificial Intelligence* by Ivan Bratko]

Database manipulation

Things to keep in mind:

- "The predicates `assert` and `retract` introduce to Prolog the possibility of programming with side effects. Code depending on side effects for its successful execution is hard to read, hard to debug, and hard to reason about formally. Hence these predicates are somewhat controversial, and using them is sometimes a result of intellectual laziness or incompetence. They should be used as little as possible when programming." [from your textbook, *The Art of Prolog*]

Database manipulation

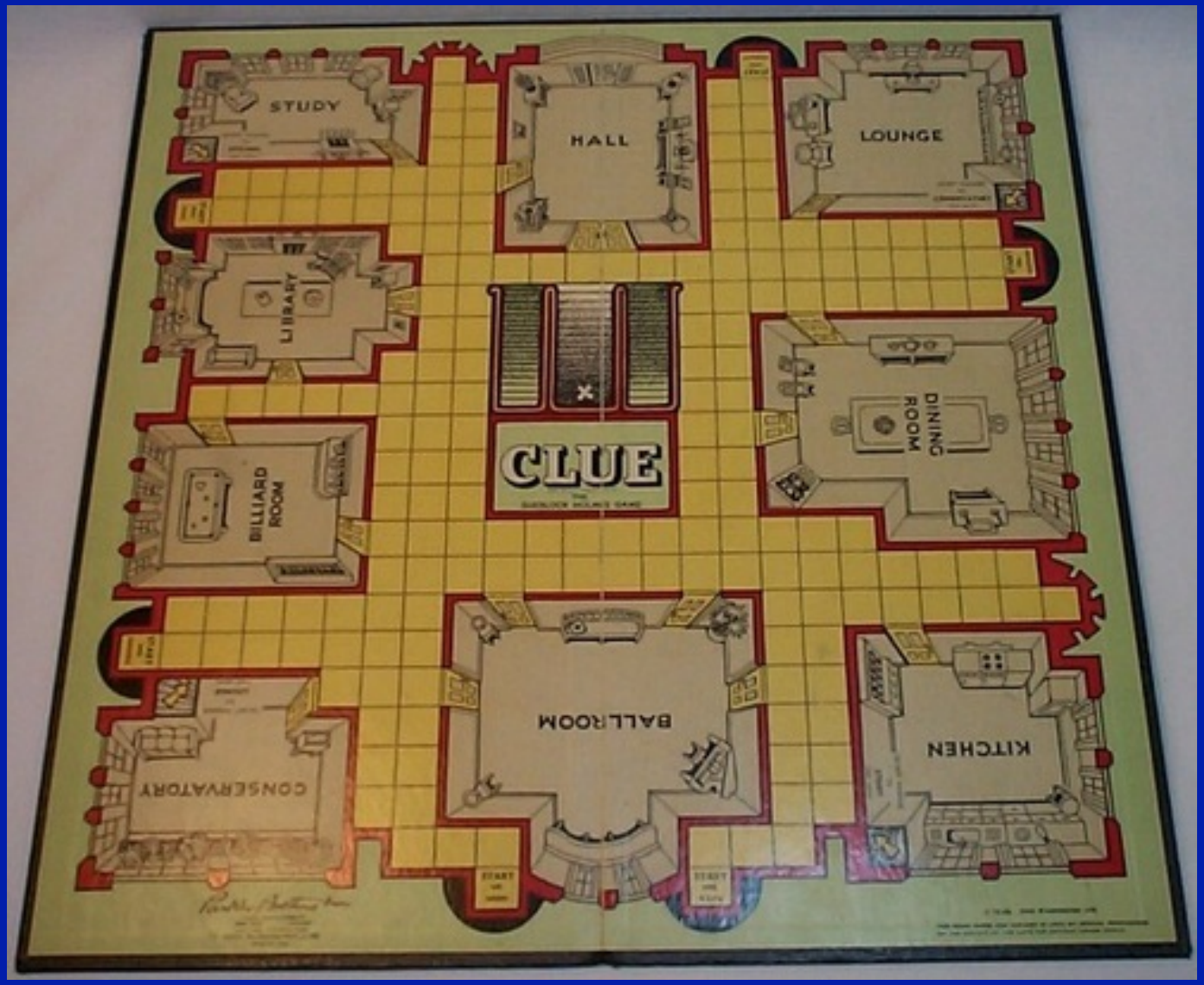
Things to keep in mind:

- Sometimes you gotta do what you gotta do. [Me]
- Read Chapter 12.2 and
  the SWI-Prolog manual (4.13) for more details.
- In fact, it's what makes your final project tractable.
  What exactly is that project?...

Your term project

The classic detective game! In **Clue**, players move from room to room in a mansion to solve the mystery of: who done it, with what, and where? Players are dealt character, weapon, and location cards after one card from each card type is secretly placed in the confidential file in the middle of the board. Players must move to a room and then make a suggestion against a character saying they did it in that room with a specific weapon. The player to the left must show one of any cards suggested to the player who made the suggestion if it's in that player's hand. Through deductive reasoning each player must figure out which character, weapon, and location are in the secret file. To do this, each player must uncover what cards are in other players hands by making more and more suggestions. Once a player knows what cards the other players are holding they will know what cards are in the secret file and can make their final accusation. A great game for those who enjoy reasoning and thinking things out. (adapted from www.boardgamegeek.com)

Your term project



Your term project

The premise:

I need help to play the board game Clue (known as Cluedo in Europe). Your job is to build me a Clue Player Assistant using the Prolog programming language.

The more work that your program does for me while I'm playing -- in other words, the more it makes me look like an expert Clue player -- the better your program is.

Your term project

Minimum deliverables:

The program should allow me to:

- initialize the game setup easily
  - which rooms and weapons are used in this version of the game?
  - how many players?
  - the order of play (whose turn next?)
  - which cards am I holding?
  - and so on...

Your term project

Minimum deliverables:

The program should allow me to:

- keep track of my own play
  - I tell the program my suggestion.
    - ("I suggested Mrs. White, the rope, and the kitchen.")
  - I tell the program what I learned
    - ("Player X showed me the rope" or "Nobody showed me anything")

Your term project

Minimum deliverables:

The program should allow me to:

- see the contents of the database on demand
- know when to make an accusation

  "Hey, Kurt, all that's left is Mr. Green, the knife, and the library. Go for it dude!"

Your term project

Minimum deliverables:

The program should also have:

- easy-to-understand documentation describing what the program can and can't do, how to use it, how it works
- easy-to-use interface

Your term project

Minimum deliverables:

The program just described would be an electronic note pad, and would be worth a C or maybe a low B, depending on factors such as quality of programming, extensibility, documentation, interface, and maybe other factors. This is easy.

Your term project

The next level:

In addition to the minimum deliverables, the program should:

- take advantage of what can be inferred from the suggestions of other players

  "Player X suggested Miss Scarlet, the candlestick, and the conservatory. Player Y showed her a card."

Your term project

The next level:

In addition to the minimum deliverables, the program should:

- tell me which suggestion to make next
  "You should suggest Col. Mustard, the wrench, and the billiard room."

Your term project

The next level:

This program would be much more than a note pad, and would earn a B to an A, depending on the factors noted previously.

Your term project

The A+ level:

The program should go beyond what's been described so far. Some suggestions:

- build models of what the other players might know and use the models to assess how close they might be to winning
- advise me to make a suggestion that might throw other players off
- other things you might think of

Your term project

As with the first project, this is a team project only. No solo efforts. Teams consist of two members.

If you don't want to work with the same partner that you worked with on the first project, that's fine. You must send me email no later than Monday, November 10, including the name of the partner whom you have abandoned. You must also clearly copy (cc) your abandoned partner with that email. Wait until after 6pm Friday.

Your term project

Your team's project is due by noon, Thursday, November 27. Use handin to submit your solution. For handin purposes, the assignment name is 'project2'.

Your term project

Bring a laptop with your program to class later that same day (November 27, our last class) for the in-class Clue program showdown. Your participation in this event factors into your mark for this term project.

So how does that Clue game work?

If you've never played Clue, you probably don't understand much of what has been said.

It might help to have a set of rules in hand so you can familiarize yourself with the game...

And even if you have played Clue, it may have been a long time since the last time you played. A refresher on how it's played couldn't hurt...

As you play...

Don't just get bogged down in rules and trying to win. Think about the following:

- What kinds of things could a Prolog program do for you?
- What information will you need to represent in your program? How should you represent that information?
- How will you communicate that information to your program? How will the program communicate information to you?

Old school Clue vs. New school Clue

Old school	New school
knife	knife
candlestick	candlestick
revolver	pistol
rope	rope
lead pipe	bat
wrench	ax
kitchen	kitchen
ballroom	patio
conservatory	spa
billiard room	theatre
library	living room
study	observatory
hall	hall
lounge	guest house
dining room	dining room

I've changed new school games to look like old school, but the board layouts still have differences.