CS330 Project Assignments as of Wed. Sept.20, 2006

About EF Games

EF Games have been introduced in the lecture of Sept.20. There are various books introducing the game rules; unfortunately, these are all rather deep and difficult. If you know what graphs, isomorphisms, and induced isomorphisms are, it may be worth looking at p.16 of the slides at

http://www.cse.ucsc.edu/~kolaitis/talks/essllif.ps

The material on these slides is *NOT* part of the course and you do not need to study the slides for the exam. However, if you are curious what EF games are good for, the slides are a good source of information. In short, EF games characterize the expressive power of query languages such as relational algebra and relational calculus. They can be used to prove mathematically that certain questions about the data in a database cannot be formulated as queries in relational algebra.

The EF Human-vs-Computer Game Application

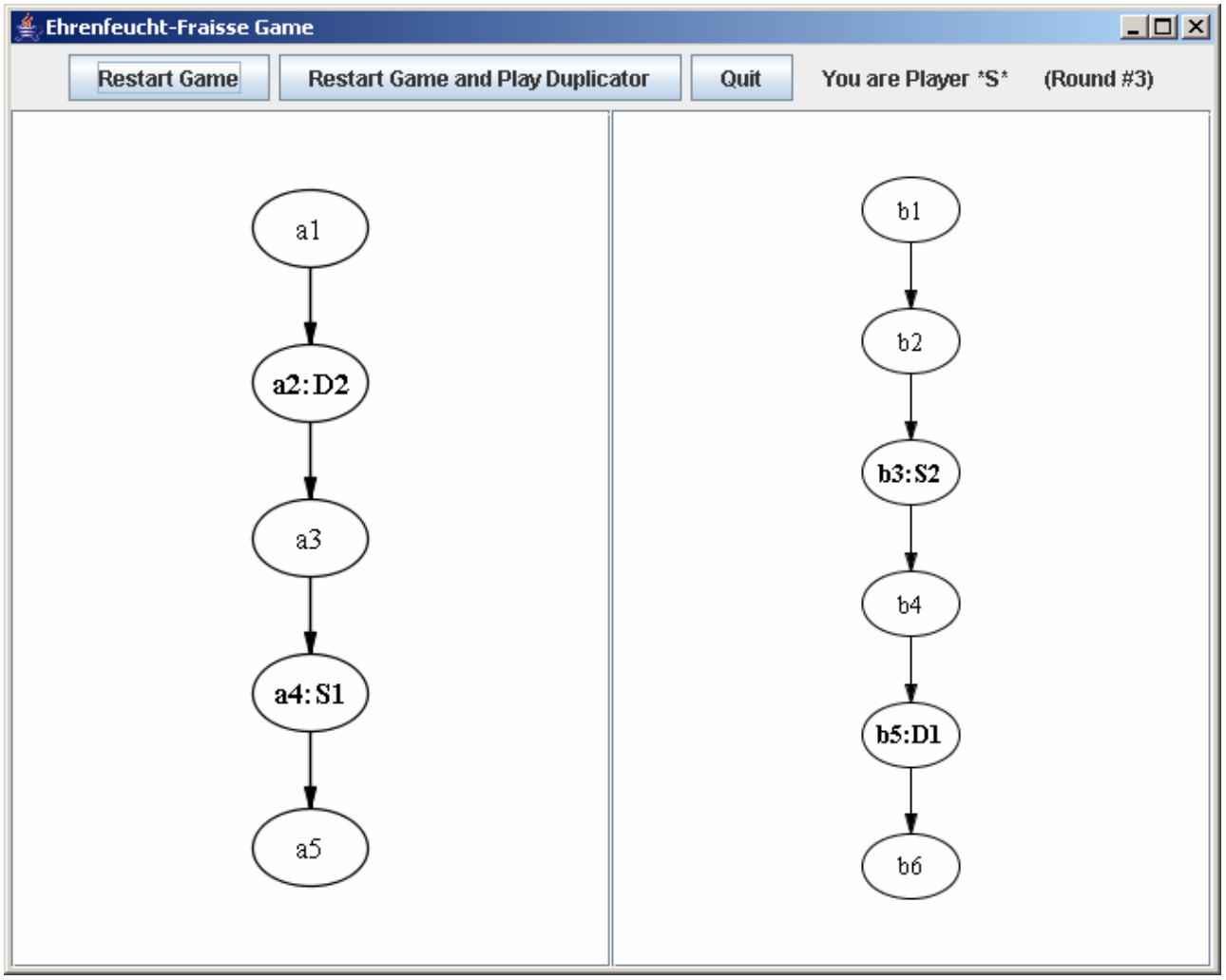
I recommend trying out the EF Game Application that is available on the course homepage:

http://www.cs.cornell.edu/courses/cs330/2006fa/efgame.zip

This archive contains a Java application (the .jar file, which you can start by double-clicking in case a Java VM is installed on your computer) and a number of graph files. (These were created using a program called GraphViz that you can download from the Web. If you like to.)

The application allows you to play against the computer and after very few games played I am quite sure you will understand the rules. When starting up, the application will ask you for two graphs to play on and for the maximal number of rounds to play. Since the computer's strategy computation was a very quick hack and is very inefficient, the strategy computation, which is performed just after you have made your choices, may take a few seconds and may even crash if the main memory granted to the Java VM is exceeded. This is also the reason why I have limited the

maximum number of rounds to four. For learning the game this should not matter, except that each game will be rather short. On startup, you will play Spoiler and the computer will play Duplicator, but there is a button for switching the roles.



Screenshot from the EF Game Application.

Goal of the Project: a Web-based Game Server for EF Games.

Note: the following is intentionally a mixture of requirements affecting the database structure and a description of the features of the Game Server to be implemented.

The finished EF Game Server allows for the playing of EF games by a community of players. Each player can pose challenges to play EF games and can accept challenges from other players.

Each *player* is identified by her netid and password. No name or further personal information is stored. A player's record is created on first login and further comprises a flag indicating whether the player is currently logged in and the date and time of the last login (null on the first login). The game server provides a logout option on each page. There is no automatic logout after a timeout. If a player leaves the game server (e.g. closes the browser) without logging out, the system simply updates the player's "last login date" on the next login. Until then, the game server will wrongly believe that the player is around.

On successful login, the game server displays the player's customized "main page", to be described later.

The game server stores *graphs* to be played on. These are uniquely identified by their name. Furthermore, a short description of the graph and a .jpg image file of the graph can be stored.

When posing a *challenge*, the player can choose the parameters of the game, i.e. the two graphs on which to play, the maximum number of rounds after which the game ends, and the role to play (Spoiler or Duplicator). The challenger may or may not chose an opponent from the players known to the system. If she does, the challenge is called direct; otherwise it is open and may be accepted by any player except the challenger. However, an open challenge can only be accepted by one player, not several, on a first-come-first-serve basis. Each challenge can be either accepted, refused, reversed or ignored. By accepting, the game starts and the challenge is not shown anymore. After refusing, the challenge is not shown on the refusing player's main page anymore. However, only a directed challenge can be refused. Reversing a challenge means to refuse a challenge but pose the same challenge to the challenger. Ignoring a challenge of course means to do nothing about it. A challenger can withdraw her challenge if it has not been accepted yet.

Each *game* follows the acceptance of exactly one challenge. Each challenge is followed by at most one game. The player who accepts another player's challenge is called her opponent. Accepting a challenge means to play according to exactly the parameters fixed in the challenge, i.e. two graphs, a maximum number of rounds, and taking the role (Spoiler or Duplicator) not chosen by the challenger. For each game we also store who has won (Spoiler, Duplicator, or no one yet), and each of the moves played.

For each *move of a game*, we store which move within the game it was (out of the sequence, S1, D1, S2, D2, S3, D3, S4, …, i.e., who – Spoiler or Duplicator -- played

and out of which round it was), whether the left ("A") or right graph ("B") was chosen, and which node within that graph was chosen. Each node within a graph is assumed to be identified by a positive integer (1, 2, 3, 4, …). We also store a timestamp indicating when the move was played.

While playing a game, the game screen is shown, which displays the graphs, the previous moves, the choices of moves the player has (in case it is her turn to play), and additional information such as the netids of the opponents and their roles and the maximum number of moves to be played. A player can give up a game or return to the main page to resume the game later (without giving up). A player can invite her previous opponent to play a finished game again (i.e. create a challenge from a finished game).

Of course each table needs a key. Decide whether a subset of the attributes required for each table above is appropriate or whether a special-purpose key-field (with generated keys) should be used. You do not have to avoid null values where they are helpful for representing data with yielding a bad schema design.

Assignment 1 (due Wed. Oct.4, 2006 11:59pm):

- Make sure you understand the rules of the game and the specification of the game server as provided so far. Use the EF game application provided on the course page to practice playing against the computer.
- Draw an entity-relationship (ER) diagram of the game server's database. This diagram is for your own use only and is not to be submitted.
- Translate this diagram into SQL create table-statements to build this database. Provide these statements in a text file. Note: test-build the tables using these statements/this file. The statements must work on the J2EE/Pointbase setup in the CSUGLab.
- Add SQL insert-statements to create a database with 10 assignments and 5 games. You may create graph records with null-values for the image fields.
- Add the following SQL queries:

  (1) For each player currently logged in, return the player's netid, the total number of games she has won, and the total number of games she has lost.

(2) Select all challenges relevant to the current player, i.e. those that have not been accepted yet and which either were posed by or are directed at the current player, or are open.

Observe:

- Only the file with the SQL statements – the create table, insert, and select statements -- is to be handed in. This must be a plain text file and these commands have to work in the CSUGLab (i.e., on PointBase).
- The assignement has to be handed in via CMS by Wed. Oct 4. 11:59pm.
- This is hard deadline and **no** exceptions can be made.

Future assignments.

What follows is a short description of the future assignments. More detailed specifications of these assignments will be posted later.

Assignment 2:

Develop a J2EE Web application using JSP and/or servlets that implements the display and summary page (the "main page") of the game server. This page must, after logging in, display the following information:

- The netid of the current player (i.e. the player to whom the page is shown).
- A timestamp showing when the page was created.
- An overview of the players currently logged in according to the database:
    - For each player show the netid, the number of games won and lost overall and, for all players other than the current player, the number of games won and lost against the current player.
- A tabular overview of the relevant (i.e. not yet accepted) challenges either by the current player, directed at her, or open.
- A tabular overview of the all current and past games (not just those involving the current player – in assignment 3 these alternatives can be switched between).

For an example of what such a page could look like see the picture ("main page of the game server" at the end of this document). Note that this picture already shows various links/buttons for functionality that will be part of later assignments.

IMPORTANT: Do not start implementing Assignment 2 before the deadline of Assignment 1. You will not work with the database schema that you have created but with a database schema that I will provide. I hope yours and mine will be the same or at least very similar, but there is no guarantee for that. If you start implementing Assignment 2 before you have seen the schema I provide, your will cause yourself additional work. (We will all use the same database schema because this will later allow us to run all Game Servers against the same database – with better opportunities for testing your code and playing with your colleagues.)

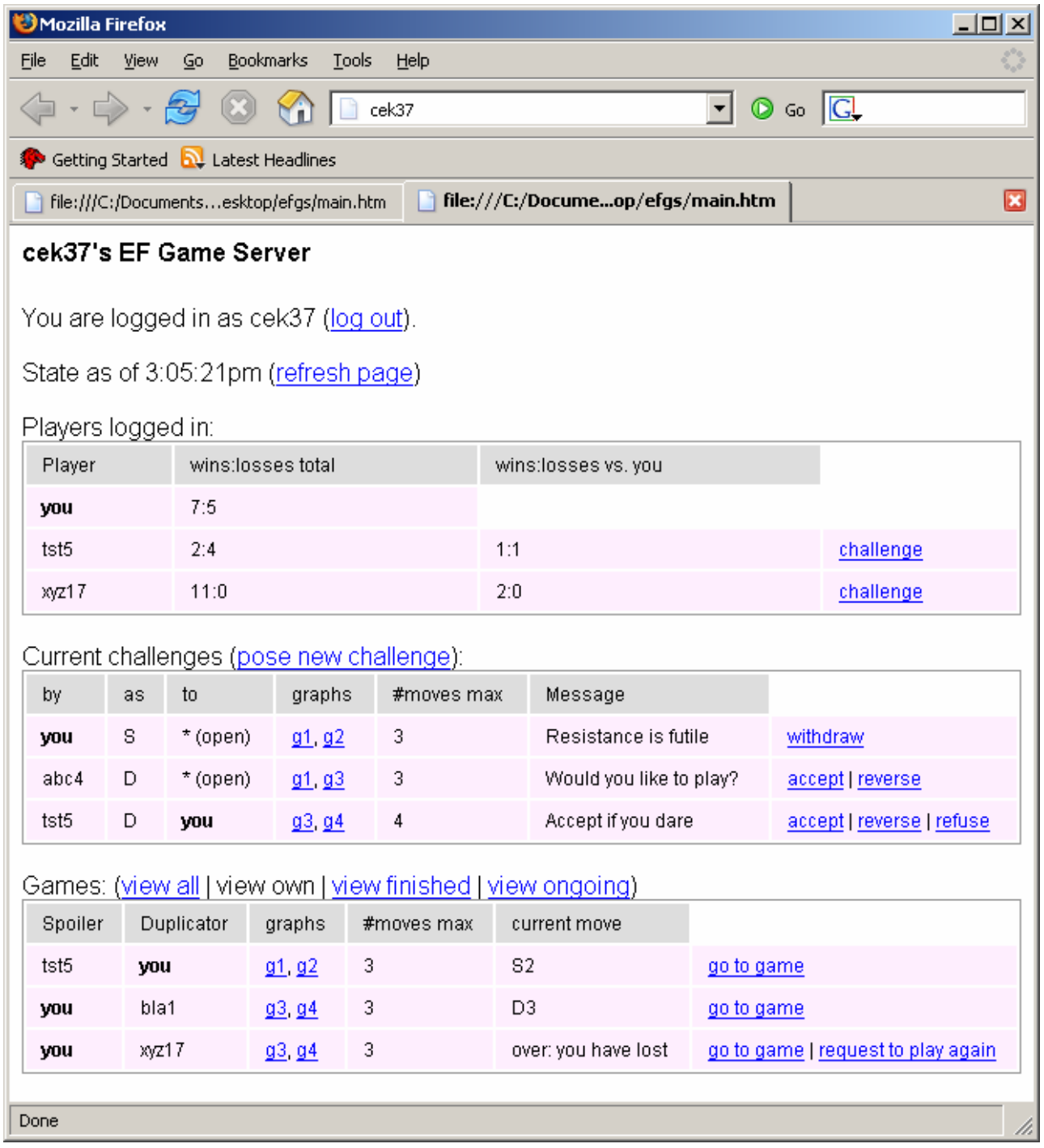
Assignment 3:

Add support for creating assignments and playing games (see the mockup screens at the end of this document).

Assignment 4:

Implement a computer player that human players can challenge via the game server using a Web service that suggests moves. (The Web service will be provided.)

Screen shot: Main page of the game server.



Screen shot: Page for EF game playing.