INTRODUCTION TO DATALOG WITH DES

This practical lab is designed to walk you through the user interface of the ***Datalog Education Systeem (DES)*** that we will use in the advanced INFO2820 class. Please follow the steps given and ask your tutor if there is anything you don't understand.

1.1 Downloading DES

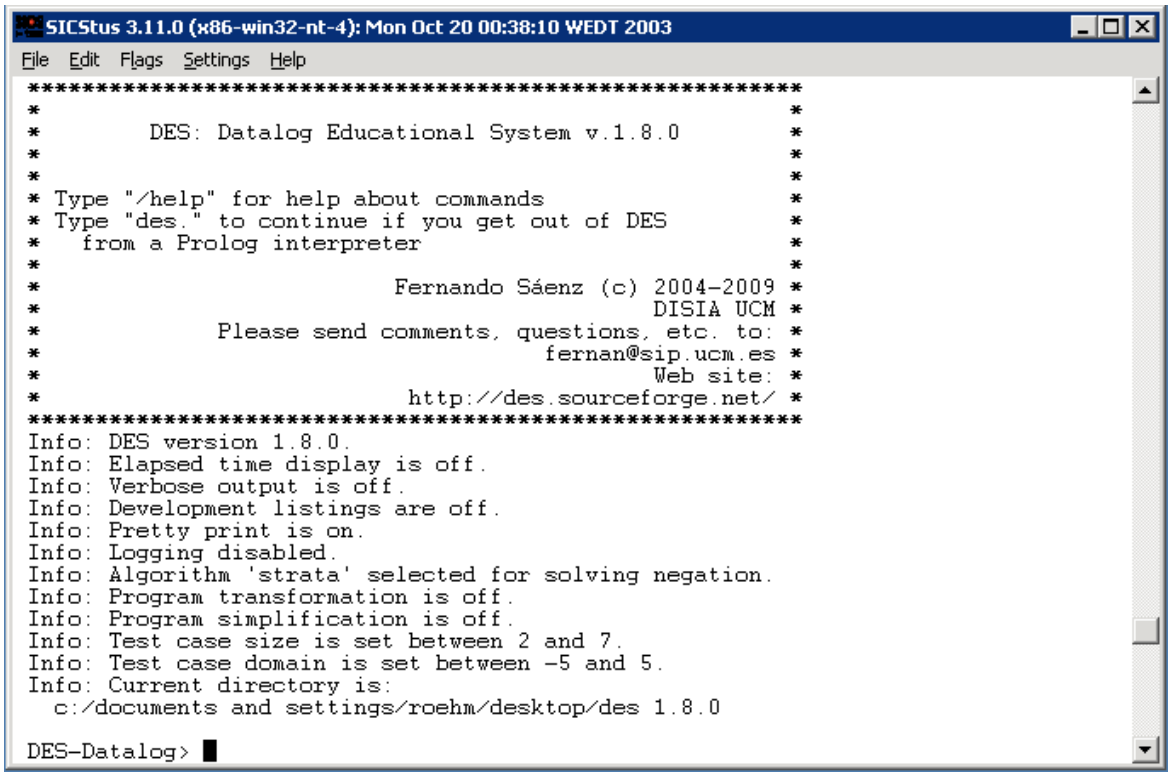
There is currently no DES installed on the Lab machines, so that you have to first download a copy from the website. Go to http://des.sourceforge.net/

and download the 'Executable Distribution' for Windows. with GUI
You should receive a Zip archive ***DES2.1ACIDE0.7Windows.zip***

Extract this archive in your user directory.

1.2 Starting DES

Navigate to the directory where you extracted the DES 2.1.0 archive, then start ***deswin.exe***



Congratulations! You now have your own copy of the Datalog Education System up and running. The interface is completely text-based. You can now execute individual Datalog commands or load and execute a datalog file from the file system that you had written earlier.

1.3 Executing Datalog Commands

Let's start with some basic commands:

/listing	lists the currently define Datalog program
/status	prints out a summary of the current configuration
/consult *filename*	loads a Datalog program into DES
/help	lists all available DES commands

There is also a PDF manual of the system in the *doc/* subdirectory.

1.4 A First Small Example

We want to create a first deductive database. For this, we will need the following commands:

/assert *rule*	to add a new Datalog *rule* into the database
/retract *rule*	removes an existing Datalog *rule* from the current database (if needed)

The ***/assert*** command allows us to introduce new facts and rules in our deductive database:

```
DES-Datalog> /assert frequents(uwe, rose).
DES-Datalog> /assert frequents(jon, abhotel).
DES-Datalog> /listing
```

Now let's run the first Datalog query: Who is frequently visting the rose hotel?

```
DES-Datalog> frequents(X, rose).
{
   frequents(uwe, rose)
}
Info: 1 tuple computed
```

Please note that we do not need to specify an explicit '?-' to start a query (in contrast to other datalog interpreters and most textbook notations).
Also note that **constants** (representing *facts* in our datalog database) and predicates always use lower case letters in their name, while **variables** have a capital letter at the beginning. The *X* in the previous query is a variable that DES has to find matching values for. Instead of giving any kind of name to these variables, you also can use **anonymous variables** using the underscore '_':

```
DES-Datalog> frequents(_, rose).
{
   frequents(uwe, rose)
}
Info: 1 tuple computed
```

1.5 Datalog Rules

Now let's check something slightly more complicated:

```
DES-Datalog> /assert sells(rose, vb, 3.50).

DES-Datalog> /assert sells(abhotel, tooheys, 4.0).

DES-Datalog> /assert sells(abhotel, loewenbraeu, 5.30).

DES-Datalog> /assert checkBeers(Bar, Beer) :- sells(Bar, Beer, Price), Price < 4.0.

DES-Datalog> cheapBeers(X,Y).
{
   cheapBeers(rose, vb)
}
Info: 1 tuple computed
```

Note again: you have to use capital 'X' and capital 'Y' to distinguish free variables from constants in Datalog! DES is very picky about its case-sensitivity...

1.6 Datalog Programs

The DES 'database' is not automatically persistent as our normal relational databases are.
We rather have to create the facts and datalog rules everytime that we start DES again.
To make this easier, we can write a complete datalog 'program' in a file editor of your choice and save it to the file system (standard file ending: .dl). DES can the *consult* this file to initialize its extensional and intensional database:

```
DES-Datalog> /consult 'drinkerdb.dl'

DES-Datalog> /listing
```