EECS 492 SGP/Lisp Info

In this assignment, you will be using Sensory GraphPlan (SGP), a Lisp implementation of GraphPlan. This requires a working Lisp interpreter. The most widely used version of Lisp, and the one we use, is Allegro Common Lisp (Allegro CL). To install a version of LISP on your computer, check out the following

http://franz.com/downloads/clp/survey (Windows/Linux/FreeBSD/OSX)

(If you don't want to give your email address, you can use a throwaway account from someplace like ThrowAwayMail.com). Once lisp is setup, download and decompress SGP from

http://aiweb.cs.washington.edu/ai/sgp.html

This will create a directory structure and sgp files.

For Windows:

1. Edit the loader.lisp file, line 33 from

(defparameter *gp-files* '(|domains| |logic| |gp| |sgp|))

to

(defparameter *gp-files* '(|domains.lisp| |logic.lisp| |gp.lisp| |sgp.lisp|))

Remember to leave spaces between four file names and type the single quote by hand. It will not work if you copy directly from pdf.

2. Explicitly set directory paths in loader.lisp, line 30&31, like:

```
(defparameter *gp-dir* "C:\\Users\\username\\Desktop\\sgp\\")
(defparameter *pddl-dir* "C:\\Users\\username\\Desktop\\sgp\\domains\\")
```

3. Start Allegro Lisp

For Mac:

1. Edit the loader.lisp file, line 33 from

(defparameter *gp-files* '(|domains| |logic| |gp| |sgp|))

to

(defparameter *gp-files* '(|domains.lisp| |logic.lisp| |gp.lisp| |sgp.lisp|))

Remember to leave spaces between four file names and Type the single quote by hand. It will not work if you copy directly from pdf

2. From terminal, change current directory to your sgp folder.

3. You do not need to modify loader.lisp line 30&31, because your current folder is sgp folder.

4. Start Allegro Lisp by command:

> /Applications/AllegroCLexpress.app/Contents/Resources/alisp

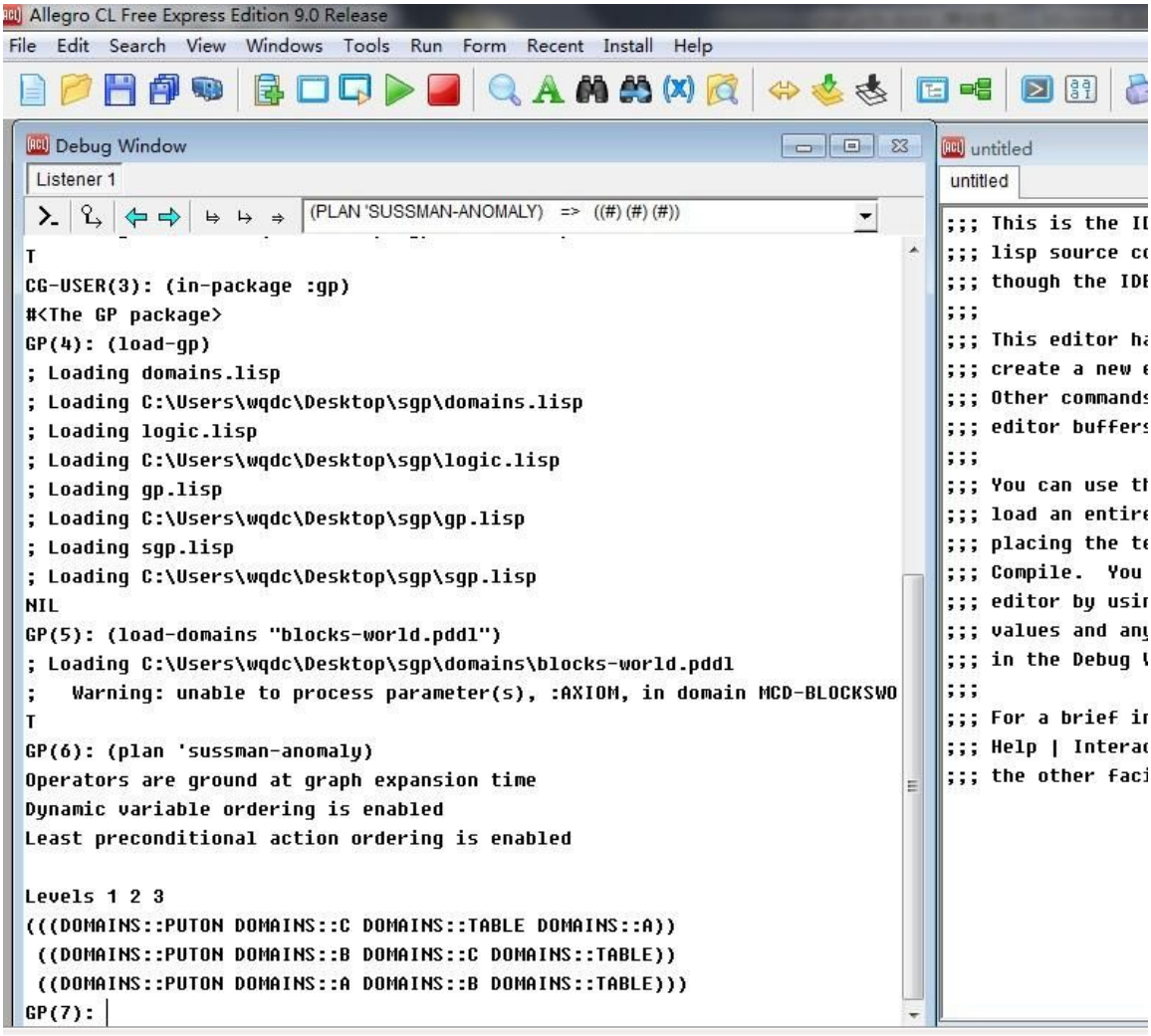
Running

At this point, you should be able to check out some of the examples, by issuing the following commands (one at a time) into your lisp interpreter:

```
      (load "C:\\Users\\username\\Desktop\\sgp\\loader.lisp")
;;or
      (load "loader.lisp") ;;for Mac/linux
      (in-package :gp)
      (load-gp)
      (load-domains "blocks-world.pddl")
      (plan 'sussman-anomaly)
```

Note: The apostrophe before the problem name (e.g. sussman-anomaly) is necessary.

An example output is attached as follows.



You will follow basically the same sequence of commands when defining your own domain, replacing the "blocks-world.pddl" and "sussman-anomaly" portions with your filename and the problem name respectively. If it is taking a long time to run, you should try replacing the (load-gp) with (compile-gp).

The output of the algorithm can be augmented by issuing the commands

```
(TRACE-GP top-level)
(TRACE-GP actions)
```

among others. These two are especially important for this assignment, because they will give you information about both the state and action levels.

NOTE: you may find the "dribble" function useful. Before executing plan, execute the form (dribble "output.txt"), and your input and output will be saved to the file output.txt. Running it again without an argument (i.e. (dribble)) will close output.txt. (You can of course choose to name your output file something else.) The file is saved to your Allegro

Common Lisp directory.

NOTE: Common Lisp should be assumed to not be case-sensitive.

By default, if the planning graph reaches 10 levels without finding the goal, plan will fail. This default behavior can be changed with the levels keyword. For example:
(plan 'sussman-anomaly :levels 2)
will cause the plan to fail.

Writing actions

The following is a list of tips for using sgp with as little pain as possible.

- The file containing the domain must be kept in the "domains" subdirectory in order for sgp to find it.
- Each action must have :action and :effect defined, and may also have :parameters and :precondition defined.
- The :action is just the action name.
- The :parameters is a list of the form (?param1 ?param2…). Begin the name of each parameter with a ?, and if this list is empty, you still need the ( ).
- The :precondition and :effect are logical statements written in PREFIX notation. The predicates and constants (see top of file) should be used where applicable. You may also use the (and cond1 cond2 …) , (not cond), and (= ?x ?y) constructions (see next).
- Logical statements may be written using prefix notation: for instance NOT(pred1(?x) AND NOT pred2(?y)) is written as (not (and (pred1 ?x) (not (pred2 ?y))))
- Equality of variables may be tested using (= ?x ?y).

You are free to look at the examples in the domains directory in order to figure out the syntax.

One final note, for some reason, CMUCL sometimes will not quit after planning, so you have to kill it.

A Brief Description of SGP's Output

Initially, the program outputs information about its settings. Without turning on the traces, SGP prints the number of the current level in the planning graph, and, when it has finished, it prints the plan it extracts. With top-level tracing turned on, the plan function also prints out its current progress within each level of the planning graph. In each level, SGP does four things: add the persistence actions, add the standard actions whose preconditions are satisfied, add the mutex arcs, and check the new predicate level to determine if the goals can be satisfied and are non-mutex.

With action tracing turned on, the output balloons. During the persistence action stage of each planning level, SGP prints out each proposition that requires a persistence action. During the normal action stage, for each action added, it prints out the operator and its parameters, as well as the propositions it achieves. Finally, (and still during the normal action stage), it prints out a list of all the new propositions that have been achieved.