SCALE FOR PROJECT EXPERT SYSTEM (/PROJECTS/EXPERT-SYSTEM)

Introduction

In order to have a productive and tolerable grading session, we ask that you :

- Stay courteous, polite, respectful and constructive during this session. The bond of trust between members of the 42 community depends on it;
- Take care to show the graded person(s) the problems you notice, and explain them as best you can;
- Accept that there may be differences in interepretation on the featureset and/or what the subject requires. Stay open-minded, try to honestly determine who is right and who is not, and grade accordingly.

Guidelines

Remember that you must ONLY grade what's on the turn-in repository !

You have to "git clone" the repository, and grade what's on it, AND ONLY WHAT IS ON IT.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/2028/expert-system.en.pdf)

First and foremost

Preliminary checks

Check the following elements:

- There is something in the git repository
- The "auteur" file, if required by the subject, is present and valid
- The Makefile, if required, is present and has the required rules

If one of these elements is not in confirmity with what the subject requires, the session stops. You may still debate on the project, but

you are not to grade the student(s).

During the rest of this session, if the program has an inappropriate behaviour (Segfault, bus error, double-free, uncaught exception, etc ...), the session stops.

☑ Yes ☒ No

Functional tests

In this section, each question will have a theme, for example "AND conditions", and you must test whether the program actually manages to handle said theme. The students MUST provide some examples. Indeed, the subject says (verbatim) "Any capability of your program that is not demonstrated by your examples will NOT be counted as present". So, if the students do not have an example that proves a feature working, then you must consider that it is not there. You should also try input files of your own creation against the users' program, as long as they fit the question's theme. If you really have no examples of your own, use the examples in the question.

AND conditions and conclusions

The program can handle basic AND conditions, and AND conclusions.

Example input:
B => A
D + E => B
G + H => F
I + J => G
G => H
L + M => K
O + P => L + N
N => M

[INITIAL FACTS HERE]
?AFKP

With =DEIJOP, AFKP is true.
With =DEIJP, AFP is true, K is false.

☑ Yes ☒ No

OR conditions

The program can handle basic OR conditions.

Example input 1:
B + C => A

D | E => B
B => C

[INITIAL FACTS HERE]
?A

With =, A should be false.
With =D, A should be true.
With =E, A should be true.
With =DE, A should be true.

Yes No

Basic XOR conditions

The program can handle basic XOR conditions.

Example input 1:
B + C => A
D ^ E => B
B => C

[INITIAL FACTS HERE]
?A

With =, A should be false.
With =D, A should be true.
With =E, A should be true.
With =DE, A should be false.

Yes No

Negation

The program can handle negation.

Example input 1:
B + !C => A

[INITIAL FACTS HERE]
?A

With =, A should be false.

With =B, A should be true.
With =C, A should be false.
With =BC, A should be false.

☑ Yes ☒ No

Same conclusion in multiple rules

The program can handle multiple rules that have the same conclusion.

Example input 1:
B => A
C => A

[INITIAL FACTS HERE]
?A

With =, A should be false.
With =B, A should be true.
With =C, A should be true.
With =BC, A should be true.

☑ Yes ☒ No

Parentheses

The program can handle parenthesised expressions correctly

Example input 1:
A | B + C => E
(F | G) + H => E

[INITIAL FACTS HERE]
?E

With =, E should be false.

With =A, E should be true.
With =B, E should be false.
With =C, E should be false.
With =AC, E should be true.
With =BC, E should be true.

With =F, E should be false.
With =G, E should be false.

With =H, E should be false.
With =FH, E should be true.
With =GH, E should be true.

☑ Yes ☒ No

Data structures

Facts and rules storage

The students must have chosen a relevant data structure to handle their rules and facts.

- A global graph of fact nodes linked by rule nodes is worth 5
- A separate graph for each rule is worth 3
- Anything else that is still more efficient than simple lists of facts and rules is worth 2
- Lists of facts and rules are worth 1
- Anything worse than that is worth 0

Of course, the students must justify their choice. If they do not, their work is worth 0.

Rate it from 0 (failed) through 5 (excellent)

Bonuses

Bonuses

Any distinct and identifiable bonus that the students can show and prove working is worth 1.

Just in case, here are the bonuses that the subject suggests (Of course, it is still OK if they did something else, just make sure it is actually vaguely useful and not just a gimmick):

- Interactive fact validation in case of undetermined facts or just to fiddle with the input while running
- Visualization of the reasoning process, either with a GUI or a terminal interface, or even with text, whatever, as long as it is comprehensible
- "OR" and "XOR" in conclusions. This can lead to undetermined facts, and should also allow for negations in conclusions to be useful.
- Biconditional rules ("If-and-only-if").

Rate it from 0 (failed) through 5 (excellent)

Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

★ Outstanding project

Empty work

Incomplete work

No author file

Invalid compilation

Norme

Cheat

Crash

Incomplete group

Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!