SCALE FOR PROJECT FT_TURING (/PROJECTS/FT_TURING)

Introduction

For the good of this evaluation, we ask you to:

- Stay mannerly, polite, respectful and constructive dunring this evaluation. The trust between you and the 42 community depends on it.

- Bring out to the graded student (or team) any mistake she or he might did.

- Accept that there might be differences of interpretation of the subject or the rules between you and the graded student (or team). Stay open minded and grade as honnestly as possible.

Guidelines

- You must grade only what is present and the graded student's (or team) repository.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/3708/ft_turing.en.pdf)

Preliminaries

This section is dedicated to setup the evaluation and to test the prerequisits. It doesn't rewards points, but if something is wrong at this step or at any point of the evaluation, the grade is 0, and an approriate flag might be checked if needed.

Respect of the rules

- The graded student (or team) work is present on her or his repository.

- The graded student (or team) is able to explain her or his work at any time of the evaluation.

- The generic rules of the subject are respected at any time of the evaluation.

- According to the subject, the program can be built with ocamlc or ocamlopt by using a Makefile. This makefile also installs anything necessary by using OPAM.

☑ Yes ☒ No

Mandatory part - The program

The first section of the mandatory part is to write a program in OCaml (or Haskell) able to simulate a turing machine according to a machine description.

Usage

Launch the program without parameters to display its usage. Is the usage what is expected according to the subject ?

☑ Yes ☒ No

Json description

Machine descriptions are fed to the program as Json descriptions. Is the program able to :

- Read this description ?
- Assert that this description is valid syntaxically and semanticaly according to the subject ?
- Robust enough to reject empty and ill formated descriptions, inexistant file description, etc ?

☑ Yes ☒ No

Execution

Test if the program is able to execute the machine given in the Json description on the input given as a parameter to the program. You must ensure that:

- The machine computes the expected result, including if that result is the machine being blocked. In that case a correct error handling is expected.

- The program displays at least the state of the tape for each transition. Thus it is easier to observe the behaviour of the machine. If the states of the tape are logged into a file to free display room on the standard output to displays a dynamic observation of the tape by using the char '\r', give the points.

☑ Yes ☒ No

Mandatory part - The 5 machine descriptions

The second section of the mandatory part is to write 5 machine descriptions that the program can simuate.

Unary addition

A machine able to compute an unary addition.

Test the machine with different valid and invalid inputs. Does the machine compute the good result or report an error consitently ?

☑ Yes ☒ No

Palindrome

A machine able to detect a palindrome.

Does the machine write a 'n' or a 'y' on the tape before halting, and is this result always consistent with the input ?

☑ Yes ☒ No

0^n1^n

A machine able to decide if the input is a word of the language 0^n1^n. For instance the words 000111 or 0000011111.

Does the machine write a 'n' or a 'y' on the tape before halting, and is this result always consistent with the input ?

☑ Yes ☒ No

0^2n

A machine able to decide if the input is a word of the language 0^2n. For instance the words 00 or 0000, but not the words 000 or 00000.

Does the machine write a 'n' or a 'y' on the tape before halting, and is this result always consistent with the input ?

☑ Yes ☒ No

Simulation of simulation

A machine able to simulate the first machine 'unary_addition'. The simulated machine's alphabet, states, transitions and input ARE the input of the simulating machine, encoded as the group has seen fit.

Is the simulated machine's result always consistent with its input ?

☑ Yes ☒ No

Bonus part

The bonus part is accessible if and only if the mandatory part is completed and perfect. I know you like free bonuses, but for this project, going beyond the mandatory part means dealing with time complexity.

Complexity

Is the program able to compute the time complexity of the algorithm of a machine ? Is this complexity consitant with the expected value ?

Yes

No

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!!!