API Project 2018

Davide Lorenzi

10 marzo 2019

Indice

1 Introduction

Here there is a little guide about the Final Project of the Algorithms and Data Structures course of the 2017-2018 year that gave me a 29 + Laude mark.

In the first part there is a little description of the assignement with specifications, rules and strategies applied.

In the second part there is a description of the main structures of the c code.

The objective of the assignement was project a Not Deterministic Turing Machine that was able to process an input string and return 1 if was accepted, 0 if any computation accepted the string, U if some machine still running but was reached the maximum number of steps.

1.1 Tools

The project has been totally developed on Clion running on Ubuntu 16.04 LTS virtual machine Hosted on Virtual Box. This was necessary to allow to use dynamic buffers taking advantage of the "%ms" acquisition in `scanf`.

To allow control about memory leaks I used Valgrind and Callgrind tools. The code is **without memory leaks**, except final unallocated buffers. The compiler is gcc while the debugger used was gdb.

2 Requirements & Limitations

Here requirements and limitation that caused the grade of difficulty of this project.

2.1 Requirements

In this project there were 6 test bench (that You can find in the "Test" folder), each one divided in 4 parts: 1 public test, that was released at the beginning of the project, used to give a little feedback about the work done, and 3 tests of different difficulties. The "Cum Laude" test was a particular test to stress particular condition that allowed to reach "Laude" grade.

The verification of the code was done submitting the .c file on a PoliMi internet page that run the code on private tests and assigned "Passed" or "Not Passed" mark showing also the time needed for execution and memory peak.

These are time and memory limits for each test bench.

Task	Time Limit[s]	Memory Limit[MiB]
IncreasingStuff	17	4
FancyLoops	20	1
MindYourLeft	20	1
UnionStuck	21	1
DontGetLost	21	7
ToCOrNotToC	19	20
CumLaude	16	16

Input stream is loaded by the test server in `stdin`.

2.2 Limitations

- The only library admitted is `stdlib.h`.
- Is not possible to run multithread operations

3 Implementation

Here there is an explanation about main features of the project.

3.1 Tape Management

The input tape is saved in an array of char. I've decided to pass to the machine (new or cloned)just a fragment of the full tape to reduce the memory used by every single machine. There is a function dedicated to extend the tape in a machine adding a chunk or blank characters. A defined variable is used to changhe the size of chunks.

3.2 Transition Management

The list of transitions is saved in an irregular multidimentional array (matrix). There are 5 branches: Input State, Charater Read, Character to Write, Tape Head Movement, Output State.

I've decided to create an Hash Map of ordered (by Input State) pointers to the random matrix to allow direct access, reducing linear search only for transitions with the same Input State.

3.3 MT Management

MT are organized in `linked list`. Every list is a `struct` containing all parameters necessary for the execution. In my project there are 2 list. The first one is the list of fathers, created in the previous step. The second list is the list of sons that contains all new machines created. When all fathers have been visited the list of sons is linked to the list of fathers and the obtained linked list is the new list of fathers. If the father has multiple sons (not determinist chase), every son is created and if the next step doesn't exceed the max step or if next step isn't an acceptance status or if exist a next step, the son machine is added to list of sons. Otherwise the machine is deleted.

Is implemented Copy On Write to reduce the number of operations.

The choosen visit algorithm is BFS because DFS isn't enought fast.

3.4 Cleaning Management

At the end of each tape the cleaner free the Input Tape array and all still existing machines.

4 Special Thanks

I would like to thanks my mates Lorenzo Gadolini, Lorenzo Magni, Daniele Nicolo' and Gioele Mombelli for the great support and collaboration during those 2 months of coding, recoding, failures and successes. Thank you.