Project 1
Perform a SAT campaign

Gilles Audemard - Ibn Sina School

Problem solving in C – http://ibnsina.alfweb.net/

July 2015

<u>Files to download</u>: projects/friday/project.tgz

1 Introduction

In this project, you have to perform a small SAT campaign: some supplied instances (random and application) are launched on different, also supplied, SAT solvers. The CPU time is limited to 100 seconds. The winner is the SAT solver which solves the most instances (In case of equality, the total time needed to solve all instances is used as a second criteria). You can see results of the last SAT competition at this address `http://www.satcompetition.org/2014/results.shtml`.

To perform that, you need to launch each solver on each instance and to record results in different files that will be analyzed one after one in order to get all results.

2 Solvers

Here is the list of the different 3 solvers used during the contest. All of them have been compiled and are expected to work on your linux machine. They are located in `solvers` directory.

1. `minisat` – `http://minisat.se/`
2. `glucose`, based on `minisat` – `http://www.labri.fr/perso/lsimon/glucose/`
3. `lingeling` – `http://fmv.jku.at/lingeling/`

3 Benchmarks

I provide a list of 20 benchmarks located in `instances` directory.

4 Getting the traces

Each solver that participated to competitions will produce some useless outputs (starting with letter `c`) AND a normalized trace.

- In case of unsatisfiability: `s UNSATISFIABLE`
- In case of satisfiability: `s SATISFIABLE`
```
v 1 -2 ...  0
```

If the solver does not print such trace, one considers that it is not able to answer the problem within the limited time.

One wants to limit CPU time. To do that, one uses the command : `ulimit -t 100`, if a processus needs more than 100 seconds to finish it will be killed by the system.

Furthermore, one does not want to trust the time needed by the solver to produce the response. Then, one needs to get this value by oneself. We are using the command `/usr/bin/time` to do that.

Then, we can have the following script to launch a solver on an instance.

```
ulimit -t 100
/usr/bin/time -o results/time.$1.$2.txt solvers/$1 instances/$2 > results/result.$1.$2.txt
```

run.sh

To run this script on `glucose` SAT solver with instance `foo.cnf`, one needs to run the following command:

`./run.sh glucose foo.cnf`

This will produce two files in `results` directory:

- `time.glucose.foo.cnf.txt`
- `result.glucose.foo.cnf.txt`

The first one contains the time needed by the solver on `foo.txt`, the second the trace of the solver. Now, you need to extract datas from these two files in order to know the result on `glucose` on this bench. If `result..` does not contain the line `s ...` then, it fails on this instance.

5 Work to do

5.1 First work

1. Create a script shell `runSolver.sh` that is able to run one solver on all instances of the contest. It has to take a parameter in input (the solver) and will run the script introduced above).

   Commands to use: `foreach` (in combination with `ls`), `run.sh`

2. Create a script shell `runAll.sh` that is able to run all solvers on all instances of the contest (use the previous script).

   Commands to use: `foreach` (in combination with `ls`), `runSolver.sh`

5.2 Second work

One wants to create scripts that extract results from traces.

1. Create a script `extractInstance.sh` that extracts results of each solver for a given instance. It has to take a parameter in input: the name of the instance. This will extract datas and produce a trace like that:

   ```
   instance.cnf XXX variables and YYY clauses (WWW unaries and ZZZ binaries)

   solver1 SAT 10.3

   solver2 SAT 15

   solver3 UNKNOWN

   ...
   ```

   The first line provides informations on the instance, the name of the instance, the number of clauses and variables, the number of unary and binary clauses (you can use the exercise 1 of session 1 for this question!).

For all other lines, the first column is the solver name, the second one the satisfiability of the instance (SAT/UNSAT/UNKNOWN), the last the time needed by the solver to solve it (if a solution is found).

Commands to use: `foreach` (in combination with `ls`), `grep`, `cut`

2. Create a script `extractSolver.sh` that extracts all results for a given solver. It has to take a parameter in input: the name of the solver. This will extract datas and produce a trace like that:

   ```
   intance1 SAT 3
   intance2 UNSAT 1.4
   intance3 UNKNOWN
   ...
   ```

   The first column is the instance name, the second one the satisfiability of the instance (SAT/UNSAT/UNKNOWN), the last the time needed by the solver to solve it (if a solution is found).

   Commands to use: `foreach`, `grep`, `cut`

3. Create a script `summarySolver.sh` that extracts a summary for a given solver. It has to take a parameter in input: the name of the solver. It will extract datas and produce a line like that.

   ```
   solver1 13 143.4 8 100.8 5 42.6
   ```

   Different columns represent solver name, number of instances solved, total time to solve them, number of UNSAT instance solved, total time to solve them, number of SAT instances solved, total time to solve them.

   Commands to use: `awk` (looks like script done during quiz), `extractSolver.sh`

4. Create a script that extracts all summary of all solvers.

   Commands to use: `foreach` and `summarySolver.sh`

5.3 A special (IMPORTANT) question

Take a look to the following code.

```
#!/bin/csh

set day = ‘date| cut -d’ ’ -f1‘
echo $day
```

setvar.sh

- What is displayed (using `echo`) on terminal?
- This can help you, isn't it ??

5.4 Some recommendations

- Do not be afraid, each script needs only 5/6 lines of code.
- Put all your scripts inside `scripts` directory.
- Always use CSH script. Do not forget, the first line is `#!/bin/csh`
- Do not forget to make your scripts executable : `chmod +x runAll.sh` for example
- For `foreach` instructions, look how to combine then with `ls` command
- For `cut`, choose carefully the separator of each data
- Running jobs on a single computer takes time, start with a small time limit (like 10 seconds) and with a small number of instances) in order to help you to generate the different scripts.

6 Deadline

The deadline is fixed to **saturday night** (you have another project on sunday...).

Send me by email (`gilles.audemard@gmail.com` the script directory (and just the script directory) in a compressed format.