Supplementary data for SPIN2015

title: On Refinement of Buchi Automata for Explicit Model Checking

This document contains the data and commands we used to prepare our paper. You can also use the pdf version of this document.

Attached files description

We provide four kind of files:

- promela models stored in the directory `models`
- the list of verification tasks used in our benchmark, file `verification_tasks.csv`
- statistics collected from the translations and Spin runs stored in `logall_typeofrefinement.csv`, 4 files (`form_ref`, `aut_ref`, `no_ref`, `simp`, where simp stands for automata after simplification)
- merged and filtered data used to create the figures and tables used in the paper, 3 files (`aut_ref_vs_no_ref.csv`, `form_ref_vs_aut_ref_spot.csv`, `form_ref_vs_no_ref.csv`)

All the described benchmark results as well as the list f verification tasks can be found in the directory `csv`

List of verification tasks

The file `verification_tasks.csv` contains 4 columns:

- model,
- formula,
- refinement_informatioin,
- source.

The pair (`model`,`formula`) defines the verification task as described in the paper. The column '`refinement_information` shows the information used in all refinements. This column should be interpreted as a *space*-separated list of *comma*-separated lists of incompatible APs. To be explicit, `'p1,p2' 'p3'` means that atomic propositions `p1` and `p2` are mutually incompatible (there is no restriction imposed by a list of length 1). Note that it is often the case that the names of APs are enclosed in quotes (`"`). And finally in the column *source* you can find two possible values: `beem` and `generated` regarding how we obtain the formula of the verification task.

Statistics – logall

The files of `logall_refinfo.csv` store stats that we derived from Spin runs of the tasks, and the automata generated by corresponding LTL translator. For each task there are 6 rows with different translators. Note that in `simp` and `aut_ref` we used only unique (non-isomorphic) automata so some rows (out of the 6) can be missing. Description of some columns follow:

- `tool` – LTL translator
- `stored`, `visited`, and `matched` – number of stored, visited, and matched states given by Spin
- `trans` – number of visited product transition given by Spin
- `max_reached_depth` – given by Spin, used to detect incomplet searches (= 999999999)
- `error` – indicates whether Spin found an counterexamle, i.e. the undesired behaviour was (1) or was not (0) present in the model
- `por` – indicates whether Partial Order Reduction was used (it was used for all X-free formulas)
- `timeout`:
  - 0 – no timeout arises
  - 1 – timeout in Spin-verifier run
  - 2 – timeout in LTL translation
  - 3 – timeout in generation of the verifier by Spin
  - 4 – timeout in automaton refinement
  - 5 – timeout in post-simplifications
- `memoryout` – the verifier (./pan) ran out of memory (20GB limit)
- `translation_cmd` – command for the LTL translator
- `stripped_formula` – the formula refined
- `result_file` – some kind of id used to merge different results (aut_ref with form_ref)
- `autfilt\_{states,edges,transitions}` – stats about the automata used for model-checking produced by autfilt (from the Spot library)

Merged data

The data for different refinement setting were merged and filtered. The files `X_vs_Y.csv` are filtered from all timeouts, memoryouts and cases where max_depth reached the limit of 999999999. The columns are the same as at `logall` files, but with corresponding prefix.

Commands used

We used the following commands to manipulate formulae and automata. They run with timeout set to 20 minutes.

- `ltlfilt -f <formula> [--exclusive-ap=<refinfo>]+` performs the formula refinement (use one `--exclusive-ap` for each list of incompatible APs),
- `autfilt --exclusive-ap=<refinfo> -s -F <neverclaim_path>` performs the automaton refinement,
- `autfilt --high --small -s -F <neverclaim_path>` performs postoptimizations used for comparison of the impact of automaton refinement only.

For each combination of *model* and *neverclaim* we run Spin using the following sequence of commands (30 minutes TO, 20GB MemoryOut).

1. `spin -a -N <neverclaim_path> <model>` to generate the source code of the verifier.
2. `gcc -DMEMLIM=20480 -DNOSTUTTER -o pan pan.c` to compile tasks with stutter invariant properties, and `gcc -DMEMLIM=20480 -DNOSTUTTER -DNOREDUCE -o pan pan.c` for properties that are not stutter invariant. The option `-DNOSTUTTER` was used to avoid the *then-present* bug in Spin described in the paper, `-DNOREDUCE` disables *partial order reduction*.
3. `./pan -a -m100000000` to run the verification itself.

We also used `autfilt` and `ltlcross` to gather various statistics about the produced automata.

Example

Here we ilustrate how to use the commands on model `anderson.4.pm` and formula `GF"P0@CS" -> GF"P0@NCS"` with refinement information `'"P0@CS","P0@NCS"'`.

1. Generating neverclaims:
   - `ltl2tgba -f 'GF"P0@CS" -> GF"P0@NCS"' -s > no_ref`
   - `ltlfilt -f 'GF"P0@CS" -> GF"P0@NCS"' --exclusive-ap='"P0@CS","P0@NCS"' | ltl2tgba -s -F - > ltl_ref`
   - `autfilt --exclusive-ap='"P0@CS","P0@NCS"' -s -F no_ref > aut_ref`
2. Run Spin for ltl refinement, the results are stored in the file `res`:
   - `spin -a -N ltl_ref anderson.4.pm`
   - `gcc -DMEMLIM=20480 -DNOSTUTTER -o pan pan.c`
   - `./pan -a -m100000000 > res`