Deriving Invariants in Propositional Logic by Algorithmic Learning, Decision Procedures, and Predicate Abstraction†

Yungbum Jung[1], Soonho Kong[1], Bow-Yaw Wang[2], and Kwangkeun Yi[1]

[1] Seoul National University [2] Academia Sinica

1. Problem : Find an invariant I for the Loop

$$\{\delta\}\ \texttt{while}\ \rho\ \texttt{do}\ S\ \texttt{end}\ \{\epsilon\}$$

Invariant must satisfy the following conditions:

(A) $\delta \Rightarrow I$ (B) $I \wedge \rho \Rightarrow Pre(I, S)$ (C) $I \wedge \neg\rho \Rightarrow \epsilon$

Example:

precondition

$\boxed{\delta = (\texttt{i} = 0)}$

loop guard ρ

```
while i < 10 do
  ret := random();
  if ret != 0 then i := i + 1
end
```

postcondition

$\boxed{\epsilon = (\texttt{i} = 10 \wedge \texttt{ret})}$

Invariant : $\boxed{i < 10 \vee (i = 10 \wedge \texttt{ret})}$

2. Idea : Using the CDNF Algorithm

Exact Learning Algorithm for Boolean formula

Asks two types of queries:

1) Membership Query $MEM(\mu)$ asks if the truth assignment μ satisfies λ.

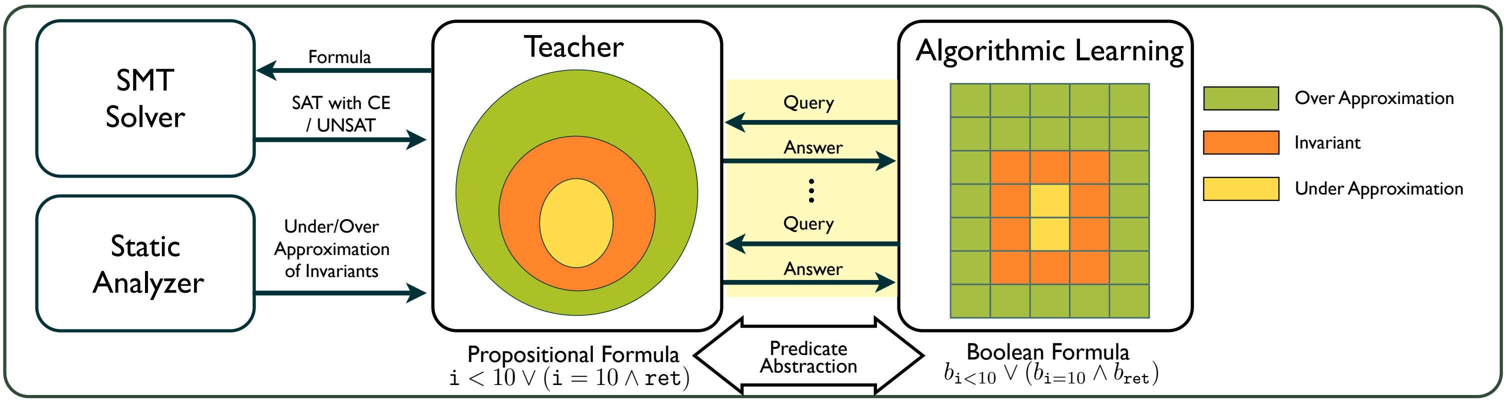
$$MEM(\mu) = Yes \quad \text{if } \mu \models \lambda$$
$$MEM(\mu) = No \quad \text{if } \mu \not\models \lambda$$

2) Equivalence Query $EQ(\beta)$ asks if the Boolean formula β is equivalent to λ, If not, the teacher returns a truth assignment as a counterexample μ.

$$EQ(\beta) = Yes \quad \text{if } \beta \equiv \lambda$$
$$EQ(\beta) = No \text{ with } \mu \quad \text{if } \beta \not\equiv \lambda \wedge (\mu \models \beta \oplus \lambda)$$

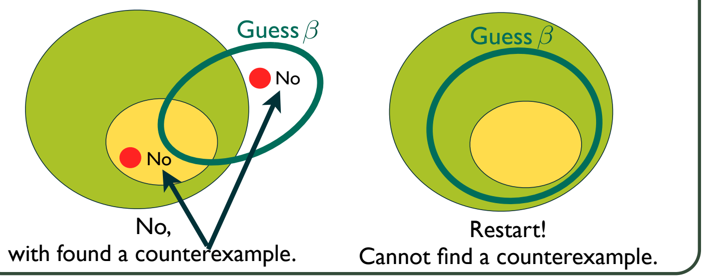
3. Solution: Implementing a Teacher to Answer Queries

Overview



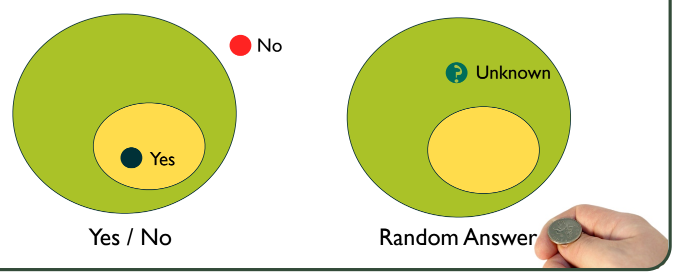
Resolving Equivalence Query: $EQ(\beta)$

1. "**Yes**", if the guess β meets the three conditions [A,B,C] to be an invariant.
2. Try to answer "**No**" with a counterexample. If not possible, then restart.



Resolving Membership Query: $MEM(\mu)$

1. Try to find an answer.
2. If not possible choose a random answer. It is still sound since we always check the three conditions in equivalence query resolution.



4. Experiment Results

For some Linux device drivers and SPEC2000 benchmarks.

Program	LOC	AP	MEM	EQ	Random	Restart	Time (sec)
ide-ide-tape	16	6	18.2	5.2	4.1	1.2	0.055
ide-wait-ireason	9	6	216.1	111.8	47.2	9.9	0.602
parser	37	20	6694.5	819.4	990.3	12.5	32.120
usb-message	18	10	20.1	6.8	1.0	1.0	0.128
vpr	8	7	14.5	8.9	11.8	2.9	0.055

The data are the average of 500 runs and collected on a 2.6GHz Intel E5300 Duo Core with 3GB memory running Linux 2.6.28.

5. Conclusion

* Novel approach to invariants generation.
* Multitude of invariants is the reason why this approach is working with random answers.
* We are currently working on its extension supporting quantified invariants.

† This work is to be presented at VMCAI'10

ROSAEC center
Research On Software Analysis for Error-free Computing
http://rosaec.snu.ac.kr

Programming Research Laboratory
http://ropas.snu.ac.kr

SEOUL NATIONAL UNIVERSITY