AE-VAL: Horn clause-based Skolemizer for $\forall\exists$-formulas

[extended abstract]

Grigory Fedyukovich
USI

Arie Gurfinkel
SEI/CMU

Natasha Sharygina
USI

Various tasks in verification and synthesis rely on efficient techniques to remove existential quantifiers from formulas in First Order Logic. In particular, *functional synthesis* aims at computing a function that meets a given input/output relation. A function with an input x and an output y, specified by a relation $f(x,y)$, can be constructed as a by-product of deciding validity of the formula $\forall x\exists y\,.\,f(x,y)$. Due to a well-known *AE-paradigm* (also referred to as *Skolem paradigm* [11]), the formula $\forall x\exists y\,.\,f(x,y)$ is equivalent to the formula $\exists sk\,\forall x\,.\,f(x,sk(x))$, which means existence of a Skolem function sk, such that $f(x,sk(x))$ holds for every x. Thus the key feature in modern quantifier elimination approaches is their ability to produce witnessing Skolem functions (e.g., [8]).

Among different shapes of $\forall\exists$-formulas, we restrict our attention to the form $S(\vec{x}) \implies \exists\vec{y}\,.\,T(\vec{x},\vec{y})$, that enjoys a particular interest in program verification, and in particular, in the tasks of *Realizability Checking of Contracts* [3]. This line of research aims at analyzing the contracts of embedded systems before an implementation of the correspondent system is arrived. A contract specifies the desired behavior of a transition system, and it contains an assumption and a guarantee. Given a system state, the assumption specifies which inputs are valid; and the guarantees specify which states the system may start in and which states the system may transition to. Such contracts are required to be realizable, i.e., there should exists a transition system, such that *for any input* allowed by the contract assumptions, *there exists some output* for which the contract guarantees are satisfied. In the nutshell, the task of realizability of the contract is reduced to deciding validity of the correspondent $\forall\exists$-formula, and the forthcoming task of synthesis of the candidate transition system is reduced to extracting a Skolem relation.

Another research direction that requires handling $\forall\exists$-formulas, is *Property-directed Equivalence Checking* [1]. It extends recently established *proof-based* verification [7] that proceeds by producing and manipulating *unbounded verification certificates* that are essentially safe inductive invariants in First Order Logic. Given two programs where the former is proven safe and simulates the latter, the goal is to derive the invariants of the latter by lifting the invariants of the former through the given simulation relation [10]. Since this operation introduces existential quantifiers, deriving of each inductive invariant is reduced to solving validity of a $\forall\exists$-formula.

For both tasks of Realizability Checking and Synthesis of Contracts and Property-directed Equivalence Checking, we propose to use AE-VAL, a decision procedure for $\forall\exists$-formulas strengthened with Skolem extracting capabilities. The pseudocode of AE-VAL is outlined in Alg. 1, but the readers are cordially invited to [2] for more details. The algorithm is based on a notion of Model-Based Projection (MBP), introduced in [6], that under-approximates existential quantification. An MBP is a function from models of the formula $T(\vec{x},\vec{y})$ to quantifier-free formulas $T_{\vec{y}}(\vec{x})$. AE-VAL proceeds by iterative enumerating models of the quantified formula until all of them are distributed in a finite number of partitions. In order to construct a Skolem relation, AE-VAL first obtains a *local* Skolem relation for each partition; second, it uses a Horn solver to over-approximate each partition; and finally, computes a global Skolem relation $Sk_{\vec{y}}(\vec{x},\vec{y})$ (as in Alg. 2), by composing the local Skolem relations with minimal and *factored* approximation of the corresponding partition.

Submitted to:

Algorithm 1: cf. [2] AE-VAL $\left(S(\vec{x}), \exists \vec{y}.\, T(\vec{x},\vec{y})\right)$

Input: formulas $S(\vec{x}), \exists \vec{y}.\, T(\vec{x},\vec{y})$
Output: return value $\in$ {VALID, INVALID} of $S(\vec{x}) \implies \exists \vec{y}.\, T(\vec{x},\vec{y})$, Skolem relation $Sk_{\vec{y}}(\vec{x},\vec{y})$
Data: Incremental SMTSOLVER, model m, MBPs: $T_i(\vec{x})$, local Skolem relations: $\phi_i(\vec{x},\vec{y})$

```
1  SmtAdd(S(x⃗));
2  forever do
3      if (isUNSAT(SmtSolve())) then
4          return VALID, AE-SKOL(S(x⃗), {(T_i(x⃗), φ_i(x⃗,y⃗))})
5      SmtPush();
6      SmtAdd(T(x⃗,y⃗));
7      if (isUNSAT(SmtSolve())) then
8          return INVALID, ∅
9      m ← SmtGetModel();
10     T_i(x⃗), φ_i(x⃗,y⃗) ← GetMBP(y⃗, m, T(x⃗,y⃗)));
11     SmtPop();
12     SmtAdd(¬T_i);
```

Algorithm 2: cf. [2] AE-SKOL $\left(S(\vec{x}), \{(T_i(\vec{x}), \phi_i(\vec{x},\vec{y}))\}\right)$

Input: formula $S(\vec{x})$, set of pairs $\{(T_i(\vec{x}), \phi_i(\vec{x},\vec{y}))\}$
Output: Skolem relation $Sk_{\vec{y}}(\vec{x},\vec{y})$, as in (1)
Data: HORNSOLVER, Guards $I_i(\vec{x})$

```
1  n ← |{(T_i(x⃗))}|;
2  for (i = 1; i ≤ n; i++) do
3      HornAdd(S(x⃗) ∧ T_i(x⃗) ∧
           ¬T_1(x⃗) ∧ ¬T_2(x⃗) ∧ ··· ∧ ¬T_{i−1}(x⃗) ⟹ I_n(x⃗));
4      HornAdd(S(x⃗) ∧ I_i(x⃗) ∧ ¬T_i(x⃗) ⟹ ⊥);
5  I_1(x⃗), ..., I_n(x⃗) ← HornSolve();
6  return Sk_{y⃗}(x⃗,y⃗);
```

$$Sk_{\vec{y}}(\vec{x},\vec{y}) \equiv \begin{cases} \phi_{\vec{y}_1}(\vec{x},\vec{y}) & \text{if } I_1(\vec{x}) \\ \phi_{\vec{y}_2}(\vec{x},\vec{y}) & \text{else if } I_2(\vec{x}) \\ \cdots & \text{else } \cdots \\ \phi_{\vec{y}_n}(\vec{x},\vec{y}) & \text{else } I_n(\vec{x}) \end{cases} \tag{1}$$

AE-VAL relies on the external procedure [6] to obtain MBPs that is based on Loos-Weispfenning quantifier elimination [9]. For solving Horn clauses, AE-VAL uses PDR engine implemented in Z3 [5]. In future, we aim at extending AE-VAL to support MBPs for Linear Integer Arithmetic [6] and the theory of arrays [4]. We believe, after such enhancing, AE-VAL will increase the scope of its applicability. The tool, its description, the source code and other interesting results are available at `http://www.inf.usi.ch/phd/fedyukovich/niagara`.

References

[1] G. Fedyukovich, A. Gurfinkel & N. Sharygina (2014): *Incremental Verification of Compiler Optimizations*. In: *NFM*, *LNCS* 8430, Springer, pp. 300–306.

[2] Grigory Fedyukovich, Arie Gurfinkel & Natasha Sharygina (2014): *Automated Discovery of Simulation Between Programs*. Technical Report USI:2014/05, Università della Svizzera italiana. `http://www.inf.usi.ch/research_publication.htm?id=83`.

[3] Andrew Gacek, Andreas Katis, Michael W. Whalen, John Backes & Darren D. Cofer (2015): *Towards Realizability Checking of Contracts Using Theories*. In: *NFM*, *LNCS* 9058, Springer, pp. 173–187.

[4] Arie Gurfinkel, Temesghen Kahsai, Anvesh Komuravelli & Jorge A. Navas (2015): *The SeaHorn Verification Framework*. In: *CAV*, LNCS, Springer. To appear.

[5] Krystof Hoder & Nikolaj Bjørner (2012): *Generalized Property Directed Reachability*. In: *SAT*, 7317, Springer, pp. 157–171.

[6] Anvesh Komuravelli, Arie Gurfinkel & Sagar Chaki (2014): *SMT-Based Model Checking for Recursive Programs*. In: *CAV*, *LNCS* 8559, pp. 17–34.

[7] Anvesh Komuravelli, Arie Gurfinkel, Sagar Chaki & Edmund M. Clarke (2013): *Automatic Abstraction in SMT-Based Unbounded Software Model Checking*. In: *CAV*, LNCS, Springer, pp. 846–862.

[8] Viktor Kuncak, Mikaël Mayer, Ruzica Piskac & Philippe Suter (2013): *Functional synthesis for linear arithmetic and sets*. *STTT* 15(5-6), pp. 455–474.

[9] Rüdiger Loos & Volker Weispfenning (1993): *Applying Linear Quantifier Elimination*. *Comput. J.* 36(5), pp. 450–462.

[10] Kedar S. Namjoshi (2003): *Lifting Temporal Proofs through Abstractions*. In: *VMCAI*, *LNCS* 2575, Springer, pp. 174–188.

[11] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of a Reactive Module*. In: *POPL*, ACM Press, pp. 179–190.