Equational Abstraction Via ACU-Generalization

1 Preliminaries

1.1 Equational Abstraction

Given a rewrite theory $\mathcal{R} = (\Sigma, E, \phi, R)$, which satisfy executability conditions and protects Bool, we generate another theory $\mathcal{R}/\mathcal{G} = (\Sigma, E \cup G, \phi, R)$, where $\mathcal{R}/\mathcal{G}$ also protects bools and satisfies executability conditions.

Let $\mathcal{T}_{\mathcal{R}/\mathcal{G}}$ be the initial Model of $\mathcal{R}/\mathcal{G}$. Due to initiality, there exists a unique reachability homomorphism $\_{}_{\mathcal{T}_{\mathcal{R}/\mathcal{G}}} : \mathcal{T}_{\mathcal{R}} \to \mathcal{T}_{\mathcal{R}/\mathcal{G}}$, which leads to for an invariant I of our interest, the theorem

$$\mathcal{T}_{\mathcal{R}/\mathcal{G}}, [t]_{E \cup G} \vDash \Box I \implies \mathcal{T}_R, [t]_E \vDash \Box I$$

$\mathcal{T}_{\mathcal{R}/\mathcal{G}}$ is called abstraction by equations $\mathcal{G}$ of $\mathcal{T}_{\mathcal{R}}$

1.2 ACU - Generalization

Given t, t', we find Least General Generalizers, or terms of the form $\{t_l\}$ such that there exists substitutions θ, θ' s.t. $\theta t_l = t$ and $\theta' t_l = t'$.

ACUOS2 [1] is one such tool that can calculate Least General Generalizers modulo associativity and commutativity. We plan use ACOUS2 in this work, although our work is independent of the generalization tool itself.

2 Problem Statement

Given $\mathcal{R}$, where $\mathcal{T}_{\mathcal{R}}, [t]_E \vDash \Box I$, where $\mathcal{T}_{\mathcal{R}/\mathcal{G}}$ cannot be Model Checked since the state space reachable from $[t]_E$ is infinite. We would like to use can we use ACU-Generalization to find a set of equations G, such that $\mathcal{T}_{\mathcal{R}/G}, [t]_{E \cup G} \vDash \Box I$ can be Model Checked as the reachable state space becomes finite.

3 Motivating Example

```
mod R+W is
  op <_,_> : Nat Nat -> Config [ctor] .
    --- readers/writers
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  vars R W : Nat .

  rl < 0, 0 > => < 0, s(0) > .
  rl < R, s(W) > => < R, W > .
  rl < R, 0 > => < s(R), 0 > .
  rl < s(R), W > => < R, W > .
```

```
endm
```

3.1 Model Checking

The reachable state space from initial state $< 0, 0 >$ is not finite. Hence, for an invariant of interest I, any procedure for determining $\mathcal{T}_R, [< 0, 0 >]_E \vDash I$ will be non-terminating. Usually, $\mathcal{T}_R, [< 0, 0 >]_E \vDash I$ is determined via Bounded Model Checking or (BMC).

4 Abstraction via Equations

4.1 Problematic Rules

Assume a rewrite theory $\mathcal{R} = (\Sigma, E \cup G, \phi, R)$ that satisfies executability conditions. We define a threshold number n, with the goal of determining which subset $R' \subseteq R$ of rules potentially make the model infinite. We call such rules "problematic" rules. Given an initial term φ, we determine the problematic rules as -

- We main two sets *good* and *problematic* of rules. Initially, *good* is empty, while *problematic* $= R$.
- We maintain a set *exploredTerms*, which initially holds φ, the initial term.
- We maintain a queue of terms called *bfs-queue*, which initially holds φ.
- (Rule-Application-Iteration) - Let ϕ be the term at head of *bfs-queue*. For each rule $r \in$ *problematic*, we attempt to rewrite ϕ using r. We get two cases -
  - We can successfully rewrite ϕ using R. Let ϕ' be the result of applying the rewrite rules. If $\phi' \in$ *exploredTerms*, we move r from the *problematic* set, to the *good*. If $\phi' \notin$ *exploredTerms*, we add ϕ' to the end of the *bfs-queue*, and to *exploredTerms*.
  - If our attempt to rewrite is unsuccessful, we move on to the next rule in *problematic*.

  After attempting to apply each rule in *problematic*, term ϕ is removed from the queue.
- After n Rule-Application-Iterations, we consider set *problematic* to be the set of rules that potentially cause the model to be infinite. In 3, we get $rl < 0, 0 >=>< 0, s(0) >$ as the only rule in the *problematic* set.

4.2 Generalization using *problematic* set

Let $r \in$ *problematic* be a problematic rule. Let *initialTrace* $= \varphi_1 \rightarrow_r \varphi_2, \ldots, \varphi_n$ be a sequence of rewrites arising out of repeated application of r, where n is a user defined threshold. Before generalization, we consider a simple heuristic -

- We maintain a set *relevant*, initially empty.
- (Finding Relevant States) Consider φ_i, a term in the sequence arising out of repeated application of r, the problematic rule. For each rule $r' \in R - \{r\}$, we attempt to rewrite φ_i using r'. If the rewrite is successful, then let φ_i' denote the result of rewriting φ_i using r'. If $\varphi_i' \notin$ *relevant*, then we add φ_i to *relevant*.
- We repeat the process of finding relevant states at most n times.

The intuition for the heuristic above is as follows -

- Our abstraction equations must operate over terms that do not contain information relevant in model checking. We try to ensure this by checking that the application of rules other than the problematic rule leads to a term containing new information.

We then perform our generalization over terms $irrelevantTrace \equiv initialTrace - relevant$. We take three terms, say ϕ_1, ϕ_2 and ϕ_3 s.t. $\phi_1, \phi_2 \in irrelevantTrace$ and $\phi_1 \rightarrow_r \phi_2 \rightarrow_r \phi_3$ (where r is the problematic rule). We use ACOUS2, to calculate least general generalizations $\Psi_a = lgg(\phi_1, \phi_2)$ and $\Psi_b = lgg(\phi_2, \phi_3)$. For each pair of generalizers $(\psi_a, \psi_b) \in \Psi_a \times \Psi_b$, we suggest an abstraction equation $\psi_a = \psi_b$. In our running example, using this simple heuristic will yield the equation $< s(X), 0 >=< s(s(X), 0 >$, which makes our module finite.

5 Implementation

Our equational abstraction tool, called "Abate", is available at https://github.com/msaxena2/abate. Abate uses core maude and the maude's Meta Level to implement the algorithm about finding problematic rules described above. The immediate next step is to extend the tool to work fully with ACOUS2, and generate equation for our simple reader+writers example.

6 Future Work

Beside completing the implementation the immediate future work is -

- Our Generalization using problematic sets algorithm can be easily extend to consider not just one next state, but a sequence of states such that under a given threshold, an old state is observed, in which case we can consider the original state to consider "irrelevant" information.
- Trying the tool on larger examples.
- Establishing that the with suggested equations, executability conditions still hold, and more importantly, the module with suggested equations protects Booleans. At the time of writing, there exists unpublished work on checking executability conditions which our tool plans to integrate with.

References

[1] María Alpuente, Demis Ballis, Angel Cuenca-Ortega, Santiago Escobar, and José Meseguer. Acuos2: A high-performance system for modular acu generalization with subtyping and inheritance.