Abstract Interpretation of High-Level Transformations*

Ahmad Salim Al-Sibahi
IT University of Copenhagen

Problem and Motivation

I am investigating how to provide an effective automated verification method for programs written in *high-level transformation languages*. High-level transformation languages (Bravenboer et al. 2008; Jouault and Kurtev 2005; Klint, Storm, and Vinju 2009; Kolovos, Paige, and Polack 2008; Mitchell and Runciman 2007; Sloane 2011) provide rich first-class operations for manipulating structures, including native collections and deep type-directed matching, allowing easy traversal of all elements of target type reachable from a particular object.

Figure 1 presents an example transformation, the *rename-field* refactoring, implemented in a small formal transformation language called TRON (Al-Sibahi, Dimovski, and Wąsowski 2016a). It replaces the old field declaration with the new one using native set operations (Line 6-7) and continues finding all field access-expressions using deep type-directed matching (Line 8) updating the ones pointing at the old field to point at the new one.

The example shows how these rich operations allow expressing very concisely what otherwise requires setting up complex visitors. My hypothesis is that I can exploit the richness of operations to optimize my verification technique, because I can better target the right parts of code and provide specialized operations for the target collections.

```
1  input:  target_class: Class, old_field: Field, new_field: Field
2  precondition: old_field ∈ target_class.fields
3                  ∧ new_field ∉ target_class.fields
4
5  // the refactoring program
6  target_class.fields :=
7        (target_class.fields \ old_field) ∪ new_field
8  foreach faexpr ∈ target_class match* FieldAccessExpr do
9    if faexpr.field = old_field ∧
10       faexpr.target.type = target_class then
11         faexpr.field := new_field
12   else skip
```

Figure 1: Simplified *rename-field* refactoring

Background and Related Work

Existing automated verification approaches focus on encoding the transformation along with structural constraints in an automated reasoning tool (Büttner, Egea, and Cabot 2012; Jackson et al. 2010; Wang, Büttner, and Lamo 2014) to check the correctness. Unfortunately, these approaches do not scale to complex transformations; they also make it hard to gain domain insight about the problem and improve on, because they rely on specific capabilities of target reasoning tools. Symbolic execution techniques (Oakes et al. 2015) have shown good improvements in efficiency and extensibility, but have only so far been utilized on simple mapping-like transformations.

Translation validation (Pnueli, Siegel, and Singerman 1998; Samet 1975) is an alternative approach that proves target properties for each run of a transformation. However, since only each particular run is validated, undiscovered bugs can show up later e.g., in a production environment where it would be more costly and time-consuming to fix them.

Approach and Uniqueness

P1 Systematically derive an abstract semantics framework from my concrete TRON (Al-Sibahi, Dimovski, and Wąsowski 2016b) semantics using collecting semantics as intermediary

P2 Instantiate my framework with a set of suitably crafted abstract domains that enables me to verify structural and simple semantic properties

P3 Implement my framework as a tool which I evaluate on an extended set of transformations

*Part of this work has been done while visiting the Celtique team at INRIA Rennes. The work is supported by *The Danish Council for Independent Research* (grant no. 0602-02327B) under the Sapere Aude scheme, project VARIETE.

from my previous work (Al-Sibahi, Dimovski, and Wąsowski 2016a)

The design and implementation of abstract domains is particularly challenging, since transformations typically modify large varied structures. This means that one must be fairly careful in providing a compact abstraction with high sharing between possible states in order to avoid a combinatorial explosion. Further complications arise from the fact that we must transform these abstractions according to the rich operations (such as type-directed matching) available in our high-level transformations while maintaining their consistency, without falling back to an unfeasible strategy of brute-force exploration of possible states.

To tackle this challenge, I will first use a type-cardinality abstraction to gain insight:

$$\left(\wp\left(\text{Store} \times \text{Heap}\right), \subseteq, \cup, \cap\right) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \left(\text{Var} \to \text{Type} \times \text{Card}, \widehat{\sqsubseteq}, \widehat{\sqcup}, \widehat{\sqcap}\right)$$

This includes defining how to execute target statements using the abstract state, for example a rule for field updates could look like:

$$\mathcal{S}[\![e_1.f := e_2]\!]\widehat{\sigma} = \begin{cases} \widehat{\sigma} & \text{if} \begin{array}{l} \mathcal{E}[\![e_1]\!]\widehat{\sigma} = \langle c, 1\rangle \\ \wedge\, \mathcal{E}[\![e_2]\!]\widehat{\sigma} = \langle c', \epsilon'\rangle \\ \wedge\, c' <: \text{type}(c, f) \\ \wedge\, \epsilon' \sqsubseteq \text{card}(c, f) \end{array} \\ \bot & \text{otherwise} \end{cases}$$

This abstraction will eliminate the errors that relate to structural typing errors, e.g., assigning an instance of type Expr to a field type Method or assigning a set of multiple instances to a field which expects a single instance.

I will then gradually define more refined abstract domains that build on top of existing shape abstraction techniques with inductive predicates (Albarghouthi et al. 2015; Chang and Rival 2008; Rival, Toubhans, and Chang 2014; Toubhans, Chang, and Rival 2013) in combination with abstraction inspired by the structural constraint in my previous work (Al-Sibahi, Dimovski, and Wąsowski 2016a). This would allow eliminating more complex structural errors and simple inductive semantic ones, e.g., using an instance of type Variable whose name does not exist in the object-level context of the manipulated structure.

Results and Contributions

I have so far derived an abstract semantics for `TRON` (P1) and instantiated with a sophisticated type-cardinality abstract domain (P2). In addition to avoiding subject-level type and cardinality errors, it immediately seems that it would also allow me to better guide my symbolic executor from previous work (Al-Sibahi, Dimovski, and Wąsowski 2016a) since it allows me to automatically detect some unreachable branches. My current goals are to move towards the more sophisticated abstract domains (P2), so I can create one of the first automated tools (P3) that can verify interesting properties of transformations. My contributions also include providing concrete insights on designing and building abstract domains for transformations, and hopefully this allows further research that scales up to larger industrial-strength transformations.

References

Albarghouthi, Aws, Josh Berdine, Byron Cook, and Zachary Kincaid (2015). "Spatial Interpolants". In: *ESOP 2015, London, UK, April 11-18, 2015. Proceedings*, pp. 634–660.

Al-Sibahi, Ahmad Salim, Aleksandar S. Dimovski, and Andrzej Wąsowski (2016a). "Symbolic Execution of High-Level Transformations". In: *SLE 2016, Amsterdam, Netherlands, October 31-November 1, 2016*, pp. 207–220.

– (2016b). *SymexTRON: Symbolic Execution of High-level Transformations*. Tech. rep. TR-2016-196. IT University of Copenhagen, Denmark.

Bravenboer, Martin, Karl Trygve Kalleberg, Rob Vermaas, and Eelco Visser (2008). "Stratego/XT 0.17. A language and toolset for program transformation". In: *Sci. Comput. Program.* 72.1-2, pp. 52–70.

Büttner, Fabian, Marina Egea, and Jordi Cabot (2012). "On Verifying ATL Transformations Using 'off-the-shelf' SMT Solvers". In: *MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*, pp. 432–448.

Chang, Bor-Yuh Evan and Xavier Rival (2008). "Relational inductive shape analysis". In: *POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pp. 247–260.

Jackson, Ethan K., Wolfram Schulte, Daniel Balasubramanian, and Gabor Karsai (2010). "Reusing Model Transformations While Preserving Properties". In: *FASE 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, pp. 44–58.

Jouault, Frédéric and Ivan Kurtev (2005). "Transforming Models with ATL". In: *Satellite Events at the MoDELS 2005 Conference, Revised Selected Papers*, pp. 128–138.

Klint, Paul, Tijs van der Storm, and Jurgen J. Vinju (2009). "RASCAL: A Domain Specific Language for Source Code Analysis and Manipulation". In: *SCAM 2009, Edmonton, Alberta, Canada*, pp. 168–177.

Kolovos, Dimitrios S., Richard F. Paige, and Fiona Polack (2008). "The Epsilon Transformation Language". In: *ICMT 2008, Zürich, Switzerland, July 1-2, 2008, Proceedings*, pp. 46–60.

Mitchell, Neil and Colin Runciman (2007). "Uniform boilerplate and list processing". In: *Proceedings of the ACM SIGPLAN Workshop on Haskell, Haskell 2007, Freiburg, Germany, September 30, 2007*, pp. 49–60.

Oakes, Bentley James, Javier Troya, Levi Lucio, and Manuel Wimmer (2015). "Fully verifying transformation contracts for declarative ATL". In: *18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS 2015, Ottawa, ON, Canada, September 30 - October 2, 2015*, pp. 256–265.

Pnueli, Amir, Michael Siegel, and Eli Singerman (1998). "Translation Validation". In: *TACAS '98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, pp. 151–166.

Rival, Xavier, Antoine Toubhans, and Bor-Yuh Evan Chang (2014). "Construction of Abstract Domains for Heterogeneous Properties (Position Paper)". In: *ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*, pp. 489–492.

Samet, Hanan (1975). "Automatically proving the correctness of translations involving optimized code - research sponsored by Advanced Research Projects Agency, ARPA order no. 2494". PhD thesis. University Stanford.

Sloane, Anthony M. (2011). "Lightweight Language Processing in Kiama". English. In: *Generative and Transformational Techniques in Software Engineering III*. Ed. by João M. Fernandes, Ralf Lämmel, Joost Visser, and João Saraiva. Vol. 6491. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 408–425.

Toubhans, Antoine, Bor-Yuh Evan Chang, and Xavier Rival (2013). "Reduced Product Combination of Abstract Domains for Shapes". In: *VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, pp. 375–395.

Wang, Xiaoliang, Fabian Büttner, and Yngve Lamo (2014). "Verification of Graph-based Model Transformations Using Alloy". In: *ECEASST* 67.