Alloy Modeling Language meets SMT-LIB

Forrest Cinelli, Kyle McCormick, Dan Dougherty

WPI

NSF Support 1408745 and 1714431

April 30 2018

Plan

- Talk about relationship between Alloy Language and SMT-LIB
- Want to raise questions to think about . . .
- Claim: SMT as a backend doesn't really make sense

  SMT has its own worldview

- Talk about Alloy vs SMT at the level of user-facing semantics

State of the art

- El Ghazi and Taghdiri. *Relational Reasoning via SMT Solving.* FM 2011
- El Ghazi, Taghdiri, Herda. *First-Order Transitive Closure Axiomatization via Iterative Invariant Injections.* NFM 2015.
- Meng, Reynolds, Tinelli, Barrett. *Relational Constraint Solving in SMT.* CADE 2017
- Kyle McCormick and Forrest Cinelli. *Translating Alloy to SMT-LIB.* Worcester Polytechnic Institute Major Qualifying Project 2018.

Deep vs Shallow Embedding

- Deep Embedding: a theory of relations as a (first-order) SMT theory
- The obvious Shallow Embedding:
  - Relations as Boolean-valued functions

  [show sample translations...]
- Immediate virtue of Meng et al: the translation itself is clear
- Also: initial experiments suggesting better performance
- Seems ideally suited to translating Alloy “as is”
  - But if we want (translation of) Alloy spec and native SMT to coexist, we have two different notions of relation. Potential for awkwardness.

Deep vs Shallow Embedding

- Deep Embedding: a theory of relations as a (first-order) SMT theory
- The obvious Shallow Embedding:
  - Relations as Boolean-valued functions

  [show sample translations...]
- Immediate virtue of Meng et al: the translation itself is clear
- Also: initial experiments suggesting better performance
- Seems ideally suited to translating Alloy “as is”
  - But if we want (translation of) Alloy spec and native SMT to coexist, we have two different notions of relation. Potential for awkwardness.

Compositionality

- Easy to encode relations as Boolean-valued functions
- But core operators of Alloy are *second-order!*
  - Cannot have a translation for *union, intersection, join,* etc
- Our MQP approach:
  1. Alloy to an Intermediate Language, a higher-order lambda-calculus
  2. Intermediate Language to SMT, by beta-reduction

Subsorting

- Sorts as Predicates

  Even here there is a choice:

  - Everything is of sort univ, all sigs are unary predicates, enforce hierarchy with assertions
  - Top-level sigs are sorts, all others are just unary predicates, enforce hierarchy with assertions

  Have to model *functions* out of a sub-sig as SMT-*relations* (domain will have to be the super-sig)

- Coercion Functions

  relnship between OSA and MSA well studied:

  - equivalence of categories

Dueling Typing Philosophies

Alloy: “A type error is associated with an expression that can be proved to be irrelevant, in the sense that it can be replaced by an empty set or relation without affecting the value of its enclosing constraint.” [Edwards, Jackson, Torlak. A Type System for Object Models (FSE 2012)]

SMT: “Well-sorted terms in SMT-LIB logic are terms that can be associated with a unique sort by means of a set of sorting rules similar to typing rules in programming languages.” [SMT-LIB Standard]

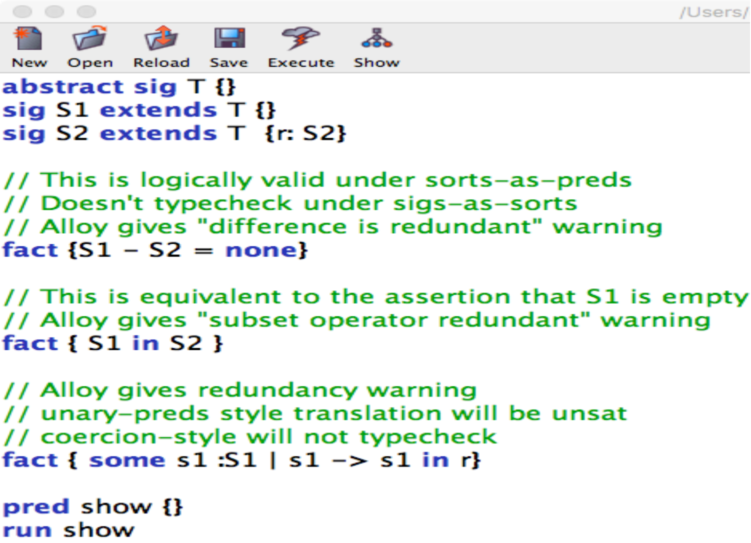
So: what is the type-preservation theorem for a translation?

Dueling Typing Philosophies

Alloy: “A type error is associated with an expression that can be proved to be irrelevant, in the sense that it can be replaced by an empty set or relation without affecting the value of its enclosing constraint.” [Edwards, Jackson, Torlak. A Type System for Object Models (FSE 2012)]

SMT: “Well-sorted terms in SMT-LIB logic are terms that can be associated with a unique sort by means of a set of sorting rules similar to typing rules in programming languages.” [SMT-LIB Standard]

So: what is the type-preservation theorem for a translation?



```
abstract sig T {}
sig S1 extends T {}
sig S2 extends T  {r: S2}

// This is logically valid under sorts-as-preds
// Doesn't typecheck under sigs-as-sorts
// Alloy gives "difference is redundant" warning
fact {S1 - S2 = none}

// This is equivalent to the assertion that S1 is empty
// Alloy gives "subset operator redundant" warning
fact { S1 in S2 }

// Alloy gives redundancy warning
// unary-preds style translation will be unsat
// coercion-style will not typecheck
fact { some s1 :S1 | s1 -> s1 in r}

pred show {}
run show
```

Summary

Some Design Considerations

- What are the goals of a translation
  - to get validity / unsat conclusions?
  - to expand to “modulo theories?”
- Which broad approach?
  - deep: a first-order “theory of Alloy” in SMT?
  - shallow: or a translation of Alloy into FOL a la SMT?
- How to reconcile the typing philosophies?
- Of course: lots of algorithmic challenges.

Transitive closure

- Jan van Eijck: transitive closure is first-order axiomatizable over finite models.
- My experience with translation was not awesome
- What's the status of this if we use Meng et al deep embedding?