The Integration of SMT Solvers into the RISCAL Model Checker

Second Master Thesis Report

Franz Reichl

January 31, 2020

Recapitulation

- Check validity of RISCAL theorems with SMT-Solvers
- Translate RISCAL declarations into SMT-LIB scripts
- Use the SMT-LIB logic QF_UFBV
- Translation requires:
    - Elimination of quantifiers
    - Encoding of RISCAL types

Recapitulation

Last time we already discussed

- Elimination of quantifiers
- Translation of integers

Outline

Translation of the Theories

Tuples and Record

- Difference between tuples and records: indexing
  - Tuples: Indexed by numbers
  - Records: Indexed by identifiers
- Treat tuples and records equally
- All RISCAL types can be represented by bit vectors

Encoding of Tuples

- Translate components of tuples
- Concatenate bit vector representations of components

Example

Let $\langle 3, \mathbb{T}, 10 \rangle$ denote a tuple t.

- Represent 3 by 11
- Represent *true* by 1
- Represent 10 by 1010

Represent t by 1010111

Operations on Tuples

- Tuple Builder: $\langle e_1, \cdots, e_n \rangle$
  - Translate $e_1, \cdots, e_n$ to $\hat{e}_1, \cdots, \hat{e}_n$
  - $concat(\hat{e}_n, concat(\cdots, concat(\hat{e}_2, \hat{e}_1) \cdots))$
- Tuple Access: $Access_i(t)$
  - Translate t to $\hat{t}$
  - Determine start/end (s, e) of sub bit vector representing the i^{th} component
  - $extract_{\langle e,s \rangle}(\hat{t})$
- Tuple Update: $Update_i(t, e)$
  - Translate t to $\hat{t}$ and e to $\hat{e}$
  - Determine start/end (s, e) of sub bit vector representing the i^{th} component
  - Extract sub-vectors before s ($\hat{t}_1$), after e ($\hat{t}_2$) from $\hat{t}$
  - $concat(\hat{t}_2, concat(\hat{e}, \hat{t}_1))$

Operations on Tuples

Example

Let e_1, e_2 be expressions of type $\{0, 1, 2, 3, 4\}$.
Translate $Access_2(\langle e_1, e_2 \rangle)$

- Translate e_1, e_2 to $\hat{e}_1, \hat{e}_2$
- Represent the tuple by: $concat(\hat{e}_2, \hat{e}_1)$
- $extract_{\langle 5,3 \rangle}(concat(\hat{e}_2, \hat{e}_1))$

Operations on Tuples

- Tuples provide equality and inequality
- Problem: Components can have different types
- Resize Components

Example

- Let t_1 be a tuple expression with two components in $\{0, 1, 2\}$
- Let t_2 be a tuple expression with two components in $\{0, 1\}$
- Let $\hat{t}_1, \hat{t}_2$ denote the translations of t_1, t_2
- $\hat{t}_1, \hat{t}_2$ have different vector lengths
- $\hat{t}_1 = \hat{t}_2$ not possible

Maps and Arrays

- Arrays: Maps with a domain of natural numbers
- Treat arrays as maps
- Proceed similarly as with tuples
- Require a linear ordering on the RISCAL types

Encoding of Maps

- Let M be a map type with domain D and image I
- Let $d_1, \cdots, d_n$ be the elements of D given with respect to the ordering
- Let m be of type M
- Translate $m(d_1), \cdots, m(d_n)$ to $m_1, \cdots, m_n$
- Concatenate $m_1, \cdots, m_n$

Encoding of Maps

Example

- Let $D = \{0, 1, 2\}$ and $I = \{0, 1, 2, 3, 4\}$
- Let m be a map from D to I with $m(x) = 2 \cdot x$
- Translate $m(0), m(1), m(2)$ to $000, 010, 100$
- Represent m by 100010000

Operations on Maps

- Map Access: $Access(m, x)$
- Translate m, x to $\hat{m}, \hat{x}$
- Introduce an enumeration function *enum* for the domain of m
- Introduce a new function f
  - Takes a bit vector of the length of $\hat{m}$
  - Takes a bit vector of the length of the enumeration
  - Gives a bit vector of the length of the representation of the image
- Assert that $f(m, 0 \cdots 0)$ retrieves the first component
- Assert that $f(m, 0 \cdots 01)$ retrieves the second component
- $\ldots$
- $f(\hat{m}, enum(\hat{x}))$

Sets

- Let U be a finite set, with some enumeration
- Let A be a subset of U
- Represent A by bit vectors of length $|U|$
- i^{th} bit is set iff i^{th} element of U is in A

Sets

Example

- Let $U = \{1, 2, 3, 4\}$
- Let $A = \{1, 4\}$
- Represent A by 1001

Operations on Sets

- Use bitwise-or for union
- Use bitwise-and for disjunction
- Use bitwise-negation for set-complement
- Count ones in a bit vector for cardinality
- ...

Basic Operations on Sets

Example

- $\{1, 2, 6\} \cup \{1, 5, 6\}$
- Represent $\{1, 2, 6\}$ by 100011
- Represent $\{1, 5, 6\}$ by 110001
- $bvor(100011, 110001)$

Basic Operations on Sets

- Problem: Sets with different types (universes)
- Find suitable common super-type

Example

- $\{1, 2\} \cup \{5, 6\}$
- Represent $\{1, 2\}$ by 11
- Represent $\{5, 6\}$ by 11
- $\mathit{bvor}(11, 11)$ does not represent $\{1, 2\} \cup \{5, 6\}$
- Represent $\{1, 2\}$ by 000011
- Represent $\{5, 6\}$ by 110000
- $\mathit{bvor}(000011, 110000)$

Advanced Operations on Sets

- Power Sets $\mathcal{P}(S)$
- Let U be the universe of S
- For $x \in U$: $\mathit{setsWith}(x)$ shall denote $\{s \mid s \subseteq U \wedge x \in s\}$
- $\mathcal{P}(S) = (\bigcup_{s \in U \setminus S} \mathit{setsWith}(s))^c$

Advanced Operations on Sets

Example

- Let $U = \{0, 1, 2\}$
- Enumeration: $\emptyset, \{0\}, \{1\}, \{0, 1\}, \{2\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}$
- *setsWith*: $10101010, 11001100, 11110000$
- $\mathcal{P}(\{1\}) : bvnot(bvor(10101010, 11110000))$
- This is 00000101

Improvements for the Translation

Improvements for the Translation

- Cut Declarations
- Auxiliary functions for quantifier expansion
- Limit use of Skolemisation

Auxiliary Functions for Quantifier Expansion

- Quantifier expansion with nested quantifiers can be costly
- Define functions that cover the individual quantifier levels

Example

- Let $Int[a, b] := \{x \in \mathbb{Z} \mid a \leq x \leq b\}$
- $\forall x : Int[0, 4].\ \forall y : Int[0, 4].\ x + y < 10$
- Introduce $f : Int[0, 4] \to Bool$
- Define $f(x) := \bigwedge_{y \in I[0,4]} x + y < 10$
- $\bigwedge_{x \in I[0,4]} f(x)$

Limiting Skolemisation

- Skolem-functions regularly require certain properties
- Assurance of properties involves universal quantifiers
- Expanding original existential quantifier can be more efficient than expanding universal quantifiers from properties.

Example

- $\forall x : Int[1, 10].\ \exists y : Int[1, 2].\ x - y \geq 0$
- Skolemisation: Use $f : I[1, 10] \rightarrow I[1, 2]$
- Bit vector representation of $\hat{f} : BitVec(4) \rightarrow BitVec(2)$
- Have to ensure that $01 \leq_{BV} \hat{f}(0001) \leq_{BV} 10,\ 01 \leq_{BV} \hat{f}(0010) \leq_{BV} 10, \cdots$

Results and Conclusions

Results

- 50 test cases covering all types
- User defined theorems and generated theorems
- Relatively large model parameters
- $\sim \frac{3}{4}$ valid

Results

	RISCAL	Boolector	Z3	Yices	CVC4
Fastest[1]	28%	14%	6%	54%	0%
Fastest valid[1]	18%	16%	8%	60%	0%
Fastest invalid[1]	58%	8%	0%	33%	0%
Faster than RISCAL		54%	52%	72%	42%
Faster than RISCAL valid		63%	61%	82%	47%
Faster than RISCAL invalid		25%	25%	42%	25%

[1] Row does not sum up to 100 due to equal timings and rounding

Results

- Results strongly depend on structure of RISCAL specifications
- RISCAL benefits from:
  - valid existentially quantified formulae
  - invalid universally quantified formulae
- SMT-Solver approach disbenefits from
  - Language constructs that need additional quantifier expansions (recursive functions, choose)

Future Work

- Support for recursive types
- Use SMT solvers incrementally
- Generation of counterexamples
- Usage of a SMT-LIB logic with quantifiers