MATHEMATICAL MODEL CHECKING FOR COMPUTER SCIENCE EDUCATION

Wolfgang Schreiner Research Institute for Symbolic Computation (RISC) Johannes Kepler University Linz, Austria





Formal Modeling&Reasoning in Education

1

			7
Definition 1.34 (Satisfaction). Satisfaction relative to a variable assignment s, in symbol as follows. (We write $\mathfrak{M}, s \not\models \varphi$ to mean "n	Is: $\mathfrak{M}, s \models \varphi$, is defined rec		
$ \begin{split} 1. \ \varphi &\equiv \bot: \ \mathfrak{M}, s \not\models \varphi. \\ 2. \ \varphi &\equiv \top: \ \mathfrak{M}, s \models \varphi. \end{split} $	$rac{\Gamma, Bdash\Delta}{\Gamma, A \wedge Bdash\Delta} (\wedge L_2)$	Γ⊢ Γ⊢∠	$(B, \Delta \ A \lor B, \Delta \ (\lor R_2))$
3. $\varphi \equiv R(t_1, \dots, t_n)$: $\mathfrak{M}, s \models \varphi$ iff $\langle \operatorname{Val}_{\mathfrak{R}}^{\mathfrak{R}}$ 4. $\varphi \equiv t_1 = t_2$: $\mathfrak{M}, s \models \varphi$ iff $\operatorname{Val}_{\mathfrak{R}}^{\mathfrak{M}}(t_1) =$	$\frac{\Gamma, A \vdash \Delta \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \lor B \vdash \Delta, \Pi} (\lor L)$	$\frac{\Gamma \vdash A, \Delta}{\Gamma, \Sigma \vdash}$	$ \begin{aligned} \mathcal{A} : \mathbf{Aexp} &\to (\Sigma \to \mathbf{N}) \\ \mathcal{B} : \mathbf{Bexp} \to (\Sigma \to \mathbf{T}) \\ \mathcal{C} : \mathbf{Com} \to (\Sigma \to \Sigma) \end{aligned} $
5. $\varphi \equiv \neg \psi$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$. 6. $\varphi \equiv (\psi \land \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \psi$. 7. $\varphi \equiv (\psi \lor \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \varphi$ of	$\frac{\Gamma \vdash A, \Delta \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \to B \vdash \Delta, \Pi} (\to L)$	$\frac{\Gamma, \lambda}{\Gamma \vdash \lambda}$	$\mathcal{C}[\mathbf{skip}] = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$ $\mathcal{C}[X := a] = \{(\sigma, \sigma[n/X]) \mid \sigma \in \Sigma \& n = \mathcal{A}[a]\sigma\}$
8. $\varphi \equiv (\psi \to \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$ 9. $\varphi \equiv (\psi \leftrightarrow \chi)$: $\mathfrak{M}, s \models \varphi$ iff either	$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta} (\neg L)$	$\frac{\Gamma}{\Gamma}$	$C[[c_0; c_1]] = C[[c_1]] \circ C[[c_0]]$
neither $\mathfrak{M}, s \models \psi$ nor $\mathfrak{M}, s \models \chi$. 10. $\varphi \equiv \forall x \psi$: $\mathfrak{M}, s \models \varphi$ iff for every x-v 11. $\varphi \equiv \exists x \psi$: $\mathfrak{M}, s \models \varphi$ iff there is an x-	$\frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall xA \vdash \Delta} (\forall L)$	$\frac{\Gamma \vdash}{\Gamma \vdash}$	$ \begin{split} &\mathcal{C}\llbracket \text{if } b \text{ then } c_0 \text{ else } c_1 \rrbracket = \\ &\{(\sigma, \sigma') \mid \mathcal{B}\llbracket b \rrbracket \sigma = \textbf{true } \& \ (\sigma, \sigma') \in \mathcal{C}\llbracket c_0 \rrbracket \} \cup \\ &\{(\sigma, \sigma') \mid \mathcal{B}\llbracket b \rrbracket \sigma = \textbf{false } \& \ (\sigma, \sigma') \in \mathcal{C}\llbracket c_1 \rrbracket \} \end{split} $
	$\frac{\Gamma, A[y/x] \vdash \Delta}{\Gamma, \exists xA \vdash \Delta} (\exists L)$	$\frac{\Gamma \vdash}{\Gamma \vdash}$	$\mathcal{C}[\![\mathbf{while} \ b \ \mathbf{do} \ c]\!] = f\!\!i\!x(\Gamma)$
L			$\begin{split} \Gamma(\varphi) =& \{(\sigma, \sigma') \mid \mathcal{B}\llbracket b \rrbracket \sigma = \mathbf{true} \ \& \ (\sigma, \sigma') \in \varphi \circ \mathcal{C}\llbracket c \rrbracket \} \ \cup \\ & \{(\sigma, \sigma) \mid \mathcal{B}\llbracket b \rrbracket \sigma = \mathbf{false} \}. \end{split}$

Typically still presented as "paper and pencil" topics.

Formal Modeling&Reasoning in Education

RISC Algorithm Li	anguage (RISCAL) _ = = ×	
e Edit Help		
:: /software/RISCAL/spec/gcd.txt	Analysis	
Computing the greatest common divisor by the Euclidean Algorithm	Exercise: Formal Specification	JKU LIT Project LogTechEdu
5 val N: N; 6 type nat = N[N]; 7	Submitter: Wolfgang Schreiner	
apred divides(minat,nirat) = 3pinat, m-p = m; 30 fun qud(minat,nirat): nat 11 requires m σ Φ ν n ν Φ; 12 reducer result(rat with 12 reducer result(rat with)	1 of 2 grade points have been earned so far. Unlock Exercise Reset Exercise Get Certificate	
13 divides(result,m) A divides(result,m) A -3rmat, divides(r,m) A divides(r,m) A r > result; 15 forces gcd(smat) = sed - gcd(s,d) = m; 17 thores gcd(smat),mat) = m r 0 m r 0 = gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = 1 m n n m e g qcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = 1 m n n g m e gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = 1 m n n g m e gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = 1 m n n g m e gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = 1 m n n g m e gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = gcd(m,n) = gcd(m,n); 18 thores gcd(smat),mat) = gcd(smat) = gcd(smat) = gcd(smat); 18 thores gcd(smat),mat) = gcd(smat); 18 thores gcd(smat),mat) = gcd(smat); 18 thores gcd(smat),mat) = gcd(smat); 18 thores gcd(smat),mat) = gcd(smat); 18 thores gcd(smat); 18 thores gcd(smat); 18 thores gcd(smat),mat) = gcd(smat); 18 thores gcd(s	TASK DESCRIPTION: In the following we consider arrays of maximum length N whose elements are natural numbers of maximum size M:	16.56 4 745.6
19 30 proc gggg(ninet,ninet): net 21: requires me4: ned; 22: ensures result = gd(n,n); 23: { 42: var annet m;	val N = 4 ; // choose small values val M = 3 ;	Problem and Rules
2 wrb bint = n;	Lype Elem = $N[M]$; type Arr - Array[N,Elem]; type Index = $Z[-1, Take the problem of finding the smallest index i a twhich an element o occurs among the first nelements of an array a. Your task is to develop a formal specification of this problem, i.e., to define a predicate P(a, n, e), the input condition of the problem, and a predicate P(a, n, e), the input condition of the problem, and a predicate P(a, n, e), the input condition.$	$((p \Rightarrow q) \vdash ((q \Rightarrow r) \Rightarrow (p \Rightarrow r)))$
 The bbag Future if a = 0 then b also as Fing officient, ninst): not Fing officient, not 	pred P(a:Arr,n:Index,a:Elem) = // formulate here the input condition 0 ≤ n A ∃1:Index. 0 ≤ i A i < n A a[i] = e	2
	pred 0(a:Arr,n:Index,e:Elem,i:Index) ↔ // formulate here the output condition 0 ≤ i ∧ i < n ∧ a[i] = e ∧ ∀i0:Index. 0 ≤ i0 ∧ i0 < n ∧ a[i0] = e ⊣ i ≤ i0	$\Delta, (\mathbf{w} \vdash (\mathbf{x} \Rightarrow \mathbf{y}), \mathbf{z}) \supseteq \Delta, (\mathbf{x}, \mathbf{w} \vdash \mathbf{y}) = \mathbf{z}$
	TASK: check whether your specification adequately specifies the following coc Check correcti) Execution completed for ALL inputs (268 m SUCCESS termination.	$\begin{array}{c} \Delta , ((\mathbf{x} \Rightarrow \mathbf{y}), \mathbf{z} \vdash \mathbf{w}) \stackrel{\bullet}{=} \Delta , (\mathbf{z} \vdash \mathbf{x},\\ \Delta , (\mathbf{w} \vdash \mathbf{x}, \mathbf{y}, \mathbf{z}) \stackrel{\bullet}{=} \Delta , (\mathbf{w} \vdash \mathbf{y}, \mathbf{z},\\ \Delta , (\mathbf{x}, \mathbf{y}, \mathbf{z} \vdash \mathbf{w}) \stackrel{\bullet}{=} \Delta , (\mathbf{y}, \mathbf{z}, \mathbf{x} \vdash \mathbf{w}) \stackrel{\bullet}{=} \Delta , \\ \end{array}$
		∧ = (ᢏ ע⊣ע ע) ∧ > 0 III

But today the educational process can be substantially supported by software.

Projects LOGTECHEDU and SemTech

LOGTECHEDU: Logic Technologies for Computer Science Education.

- □ JKU LIT (Linz Institute of Technology), 2018–2020.
- □ Institutes FMV (Biere, Cerna, Seidl) and RISC (Schreiner, Windsteiger).
- http://fmv.jku.at/logtechedu

SemTech: Semantic Technologies for Computer Science Education.

- □ Austrian OEAD WTZ and Slovak SRDA, 2018–2019.
- □ JKU Linz (Schreiner) and TU Kosice (Novitzká, Steingartner).
- https://www.risc.jku.at/projects/SemTech

Investigate the potential of formal modeling&reasoning software for education.

Educating with the Help of Formal Models

- I Today much of modeling&reasoning can be automated by computer software.
 - Substantial advances in computational logic (automated reasoning, model checking, satisfiability solving).
- By the application of such software education may be supported.
 - □ May demonstrate the practical usefulness of theory.
 - □ May increase the motivation of students to model and to reason.
- The ultimate goal is self-directed learning.
 - □ Teachers become "enablers" by providing basic knowledge and skills.
 - □ Students "educate themselves" by solving problems.
 - (Voluntary) quizzes, (mandatory) assignments, possibly (graded) exams.

Core idea: let students actively engage with lecturing material by solving concrete problems and by receiving immediate feedback from the software.

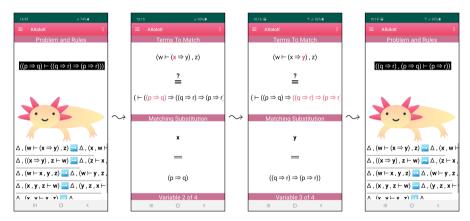
Research Strands

- Solver Guided Exercises (Limboole, Boolector)
- Teaching Solver Technology (Limboole, Boolector)
- Proof Assistants for Education (Theorema, AXolotl)
- Specification and Verification Systems for Education (RISCAL)
- Formal Semantics of Programming Languages (*Jane*)
- Logic across the Subjects in Primary, Secondary and Higher Education

Various aspects of the general idea.

Example: AXolotl

Author: David Cerna; Google Play Store (search for "AXolotl Logic Software")



Proving on a smartphone by a purely touch-based interface (no keyboard input). 6/23

The RISC Algorithm Language (RISCAL)

A language and software system for investigating finite mathematical models (i.e., a "mathematical model checker").

- Formulation of mathematical theories and theorems.
- Formulation and specification of (also non-deterministic) algorithms.
- Rooted in strongly typed first order logic and set theory.
- All types are finite (with sizes determined by model parameters).
- All formulas are automatically decidable.
- Correctness of all algorithms is decidable.
- Automatic generation of (again decidable) verification conditions.

Checking in some model of fixed size before proving in models of arbitrary size.

The RISCAL Software

RISC Algorithm Language (RISCAL) _					
File Edit Help					
ile: /usr2/schreine/repositories/RISCAL/trunk/spec/gcd.txt	Anatysis	Tasks			
1//		✓ gcdp(ℤ,ℤ) ♦ Execute operation			
<pre>2// Coputing the greatest common divisor by the fuclidean Algorithm </pre>	Transition Nondeterminant Default Values () Description: \$\$ Values in parks: Per Values () Valuatization: Trace: Trace () Parallelism: Multi-Threaded Threads: 4 Operation: \$\$ operator: \$\$ operator:	 Validate specification Execute specification Is precondition satisfiable? Is precondition not trivial? Is postcondition always satisfial 			
<pre>is "control of the a to be divide result as A divide (result, d) A divide result as A divide (result, d) divide result, d) divide result as A divide (result, d) divide result, d) divide result as A divide (result, d) divide result, d) divide result, d) divide result, d) divide result, d) divide result as A divide (result, d) divide result, d) divide resul</pre>	The approximate in the second				

https://www.risc.jku.at/research/formal/software/RISCAL

Theories and Theorems

```
val N: N:
type Literal = \mathbb{Z}[-N,N]:
type Clause = Set[Literal] with \neg=lEvalue. -lEvalue:
type Formula = Set[Clause]:
type Valuation = Set[Literal];
pred satisfies(v:Valuation, l:Literal) a lEv:
pred satisfies(v:Valuation, c:Clause) 

pred satisfies(v:Valuation, f:Formula) \Leftrightarrow \forall c \in f, satisfies(v,c):
pred satisfiable(f:Formula) \Leftrightarrow \exists v: Valuation. satisfies(v, f):
pred valid(f:Formula) \Leftrightarrow \forall v:Valuation. satisfies(v,f):
fun not(f: Formula):Formula = { c | c:Clause with \forall d \in f. \exists l \in d. -l \in c };
theorem notValid(f:Formula) \Leftrightarrow valid(f) \Leftrightarrow ¬satisfiable(not(f)):
```

First-order logic, integers, tuples/records, arrays/maps, sets, algebraic types.

Declarative Algorithms

```
fun literals(f:Formula):Set[Literal] = {l | l:Literal with \exists c \in f. l\in c};
fun substitute(f:Formula,l:Literal):Formula = \{c \setminus \{-1\} \mid c \in f \text{ with } \neg (l \in c)\};
multiple pred DPLL(f:Formula)
  ensures result 
satisfiable(f);
  decreases |literals(f)|;
0
  if f = \emptyset[Clause] then
    т
  else if ø[Literal] \in f then
    1
  else
    choose lEliterals(f) in
    DPLL(substitute(f.l)) v DPLL(substitute(f.-l));
```

Functions, predicates, implicitly defined constants and functions.

Imperative Algorithms

```
proc DPLL2(f:Formula): Bool
  ensures result . satisfiable(f):
 var satisfiable: Bool = \perp:
 var stack: Arrav[N+1.Formula] = Arrav[N+1.Formula](\emptyset[Clause]):
 var number: N[N+1] = 0:
  stack[number] = f: number = number+1:
 while -satisfiable A number>0 do
    invariant 0 \leq \text{number } \wedge \text{number} \leq N+1:
    invariant number > 0 \land stack[number-1] \neq  o[Clause] \land \neg o[Literal] \in stack[number-1] \Rightarrow number < N+1:
    invariant satisfiable(f) •• satisfiable v =i:N[N+1] with i<number. satisfiable(stack[i]):
    decreases if satisfiable then 0 else \bar{\mathbf{x}}_k: \mathbb{N}[\mathbf{N}] with k<number. size(stack[k]):
    number = number-1:
    var g:Formula = stack[number]:
    if q = \alpha[Clause] then
      satisfiable ⊨ T:
    else if ¬ø[Literal]€g then
      choose lEliterals(g):
      stack[number] = substitute(q.-l): number = number+1:
      stack[number] = substitute(q,l): number = number+1:
  return satisfiable:
```

Procedures, variables, loops.

Transition Systems

```
proc system(x0: Positions, y0: Positions): ()
  requires init(x0, y0);
{
  var x: Positions = x0: var v: Positions = v0:
  var rs: Array[N+1,Robot] = Array[N+1,Robot](0);
  var ds: Array[N+1, Direction] = Array[N+1, Direction](Direction!Stop);
  for var i:\mathbb{N}[\mathbb{N}] = 0; i < \mathbb{N}; i = i+1 do
    choose r: Robot. d: Direction with nextDir(x, y, r, d):
    rs[i] := r: ds[i] = d:
    x = moveX(x, r, d); y = moveY(y, r, d);
    assert noCollision(x, y) v print rs, ds in \bot;
}
```

Nondeterministic systems defined by initial state condition and next state relation.

RISCAL Checking

Using N=2. Type checking and translation completed.

Executing notValid(Set[\mathbb{Z}]) with selected 512 inputs. Execution completed for SELECTED inputs (111 ms, 512 checked, 0 inadmissible).

Executing DPLL(Set[Zet[\mathbb{Z}]]) with selected 512 inputs. Execution completed for SELECTED inputs (1219 ms, 512 checked, 0 inadmissible).

Executing DPLL2(Set[Set[\mathbb{Z}]]) with selected 512 inputs. 435 inputs (435 checked, 0 inadmissible, 0 ignored)... Execution completed for SELECTED inputs (2436 ms, 512 checked, 0 inadmissible).

Executing DPLL_OutputCorrect(Set[\mathbb{Z}]) with selected 512 inputs. Execution completed for SELECTED inputs (609 ms, 512 checked, 0 inadmissible).

Automatic checking of theorems, algorithms, generated verification conditions.

Application: Mathematical Modeling

```
val N:N: // variablex x0....xN
val M:N: // values 0....M
type Var = N[N]: // a variable
type Val = N[M]: // a value
type Ass = Map[Var.Val]: // an assignment of variables to values
type Pred = Set[Ass]: // a predicate as a set of assignments
val Ass = { a | a:Ass }:
pred independent(P:Pred, x:Var) •
  ∀a:Ass, v1:Val, v2:Val.
    (a with [x] = v1) \in P \Leftrightarrow (a with [x] = v2) \in P:
fun EXISTS(x:Var, P:Pred):Pred =
 { a | a:Ass with \exists v:Val. (a with [x] = v) \in P } ;
theorem Exists1(x:Var. P:Pred)
 \forall 0: Pred with independent(0,x), 0 = EXISTS(x,P)
    P \subseteq 0 \land \forall 00: Pred with independent (00, x), P \subseteq 00 \Rightarrow 0 \subseteq 00:
theorem Exists2(x:Var. P:Pred) .
  EXISTS(x,P) = \bigcap \{ 0 \mid 0 : \text{Pred with independent}(0,x) \land P \subseteq 0 \}:
```

Executing Exists1(\mathbb{Z} ,Set[Array[\mathbb{Z}]]) with all 768 inputs. Execution completed for ALL inputs (4311 ms, 768 checked, 0 inadmissible). Executing Exists2(\mathbb{Z} ,Set[Array[\mathbb{Z}]]) with all 768 inputs. Execution completed for ALL inputs (1674 ms, 768 checked, 0 inadmissible).

Validating conjectures (respectively the formalization of theorems).

Application: Specifying and Verifying Algorithms

<pre>proc gcdp(m:nat,n:nat): nat requires m≠0 v n≠0; ensures result = gcd(m,n);</pre>	
{	
var a:nat ⊨ m;	
var b:nat = n;	
while $a > 0 \land b > 0$ do	
invariant $a \neq 0$ v b $\neq 0$;	
invariant gcd(a,b) = gcd(ol	d blo. b)
decreases a+b;	u_u,otu_b),
{	
if a > b then	
a = a%b;	
else	
b = b%a;	
}	
return if $a = 0$ then b else a	;
}	,
,	

. . .

ordo(7.7) Validate specification Execute specification Is precondition satisfiable? Is precondition not trivial? Is postcondition always satisfiable? Is nostcondition always not trivial? Is result uniquely determined? Verify specification preconditions Verify correctness of result Is result correct? 23 Verify iteration and recursion C Does loop invariant initially hold? Does loop invariant initially hold? Is loop measure non-negative? Is loop invariant preserved? (A) Is loop invariant preserved? Is loop invariant preserved? Is loop measure decreased? Verify implementation preconditions

Does operation precondition hold?

Open operation precondition hold?

Does operation precondition hold?

Execute operation Validate specification Evenute specification Is precondition satisfiable? Is precondition not trivial? Is postcondition always satisfiable? Is postcondition always not trivial? Is postcondition sometimes not trivial? Is result uniquely determined? Verify specification preconditions Does operation precondition hold? Verify correctness of result Verify iteration and recursion Does loop invariant initially hold? Open loop invariant initially hold? Is loop measure non-negative? Is loop invariant preserved? Is loop invariant preserved? Is loop invariant preserved? Is loop measure decreased? (3) Is loop measure decreased? Verify implementation preconditions Does operation precondition hold? Does operation precondition hold?

Does operation precondition hold?

Does operation precondition hold?

Executing $gcdp(\mathbb{Z},\mathbb{Z})$ with all 121 inputs.

Execution completed for ALL inputs (172 ms, 120 checked, 1 inadmissible).

```
Executing \_gcdp\_5\_PreOp3(\mathbb{Z},\mathbb{Z}) with all 121 inputs.
87 inputs (86 checked, 1 inadmissible, 0 ignored)...
```

of inputs (of checked, i indumissible, o ignored)...

Execution completed for ALL inputs (2843 ms, 120 checked, 1 inadmissible).

Validating algorithms, their specification, annotations, verification conditions. 15/23

RISCAL Approach to Model Checking/Formula Decision

```
ComSem := Single + Multiple
Single := Command \rightarrow (Context \rightarrow Context)
Multiple := Command \rightarrow (Context \rightarrow Seq(Context))
Seq(T) := Unit \rightarrow (Null + Next(T, Seq(T)))
[[.]]: Command \rightarrow Single
[[if E then C]] := \lambda c. if [[E]](c) then [[C]](c) else c
```

```
interface ComSem {
   public interface Single extends ComSem, Function<Context,Context> { }
   public interface Multiple extends ComSem, Function<Context,Seq<Context> > { }
}
interface Seq<T> extends Supplier<Seq.Next<T> > { ... }
```

```
ComSem.Single ifThenElse(BoolExpSem.Single E, ComSem.Single C)
{ return (Context c) -> E.apply(c) ? C.apply(c) : c; }
```

Translation of every RISCAL phrase to its (potentially nondeterministic) semantics and the execution of this semantics. 16/23

RISCAL Formula Decision (Experimental Alternative)

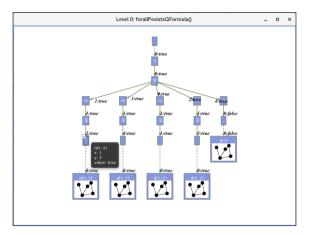
```
(set-logic QF_UFBV)
(declare-fun x() (_ BitVec 4))
(define-fun y() (_ BitVec 4)#b0001)
(assert (not (bvule x (bvadd x y))))
(check-sat)
(exit)
```

Translation of RISCAL theory to SMT-LIB.

- Author: Franz-Xaver Reichl (master thesis).
- □ QF_UFBV: quantifier-free formulas over bitvectors with uninterpreted functions.
- □ Well supported by various SMT solvers: Boolector, Z3, Yices, CVC4, ...
- Elimination of quantifiers by skolemization and expansion.
- □ Translation of integers, tuples/records, arrays/maps, sets, ... to bit vectors.
 - Non-trivial because, e.g., RISCAL uses "true" mathematical integers.

Much faster in many (not all) cases, systematic benchmarks under way.

RISCAL Visualization



Pruned evaluation trees to explain the truth value of a formula.

RISCAL Counterexample Generation

```
theorem _search_0_LoopOp6(a:array, x:elem) \Leftrightarrow

\forall i:int, r:int. ((((((0 \le i) \land (i \le N)) \land ...) \Rightarrow

(let i = i+1 in

(\forall j:int. (((0 \le j) \land (j < i)) \Rightarrow (a[j] \neq x))))));
```

```
ERROR in execution of _search_0_LoopOp6([0,0],0): evaluation of
_search_0_LoopOp6
at unknown position:
  theorem is not true
ERROR encountered in execution.
```

```
Executing __search_0_LoopOp6_refute().
This sequence of assignments leads to a counterexample
(note the underlined editor lines):
a=[0,0],x=0
i=0,r=-1
i=1
j=0
```

```
var i:int = 0;
var r:int = -1;
while i < N ∧ r = -1 do
invariant 0 ≤ i ∧ i ≤ N;
invariant y;:int. 0 ≤ j ∧ j < i = a[j] ≠ x;
invariant r = -1 v (r = i ∧ i < N ∧ a[r] = x);
decreases if r = -1 then N-i else 0;
{
if a[i] = x then r = i;
i = i+1;
}
return r;
```

Core information to explain the invalidity of a formula.

RISCAL Web Exercises

Exercise: Formal Specification	certificate
Submitter: Wolfgang Schreiner	webex.verify okay? submitter, points
1 of 2 grade points have been earned so far. Unicol Exercise Reset Exercise Get Certificate TASK DESCRIPTION:	Web Browser exercise.html input.uid webex.wsgi infid.dov/mut.log
In the following we consider arrays of maximum length N whose elements are ratural numbers of maximum see M. Val N = $\frac{4}{1-z}$; // choose small values	id televe. As of the second se
[type Elms = N[N]; type Arr = Array[N, Elms]; type Index = Z(-1,N); Take the problem of finding the smallest index i at which an element e occurs among the first in elements of an array a 'Nour task is to develop a formal specification of this problem, i.e., to define a predicate P(a,n,e), the input condition of the problem, and a predicate Q(a,n,e), the output condition.	Web Server
pred F(a;Arr,n:Index,:ellem) = // formita here the input condition 0 = n A 1:Index.0 = 1 A i < n A a[i] = 0	vebex
pred Q(a;Arr,n:Index,e:Elem,i:Index) = // formulate here the output condition 0 s i A i a n A A[] = e A Vi0:Index. 0 s 10 A 10 < n A a[10] = e - i s 10	
TASK: check whether your specification adequately specifies the following code Ores correct) Ores correct Execution completed for ALL inputs (268 ms, 1296 checked, 2880 inadmissible). SUCCESS termination.	

Framework for web-based exercises checked by a RISCAL server.

Educational Usage

 "Formal Methods in Software Development" (JKU, master programs "Computer Science" and "Computer Mathematics")

- RISCAL: formal problem specifications; specification and verification of imperative programs.
- "Formal Methods and Specification" (TU Prague, Stefan Ratschan, master program "Informatics")

□ RISCAL: formal specification and verification of imperative programs.

- "Formal Modeling" (JKU, bachelor program "Technical Mathematics")
 - RISCAL: formal modeling of computational problems, search and scheduling problems ("puzzles"), dynamic systems.
- "Logic" (JKU, bachelor prog. "Computer Science" and "Artificial Intelligence")
 - □ RISCAL, AXolotol, Theorema, Limboole, Boolector, Z3.
 - Bonus (RISCAL Web) and laboratory exercises (RISCAL desktop, AXolotol).

Various Bachelor and Master Theses

RISCAL Experience

Observations, results of questionnaires, test/exam results.

- Students with some technical/formal background (2nd year and higher):
 - □ High satisfaction with ease of use.
 - □ Much more liked than "proof-based" logic software.
 - Many students were indeed enabled to independently develop adequate formal specifications, models, program annotations.
- Absolute beginners (1st semester):
 - □ More used than other tools on FO and SMT (but less than SAT solvers).
 - □ Those who performed the exercises scored better in tests.
 - □ Students that scored poorly in tests did not use the software.
 - □ "Extrinsic motivation": mainly used to get additional grade points.

From a certain background/level on, substantial increase in motivation and interest (but not a statistically significant effect on grades).

Conclusions and Further Work

- Formal modeling&reasoning software can indeed be a factor to increase interest in "formal" topics and foster "self-directed" learning.
- However, students mainly profit if they already have certain abilities respectively some background.
- Care has to be taken to not "loose" the weaker beginners; these are easily overwhelmed by information overload or (trivial) syntactic/technical difficulties.
- We are currently running a beginner's course with an easier to use web-based interface and will evaluate the difference it makes.
- Future work will concentrate on development of software-based course materials and on technical extensions (integration with interactive provers, modeling and reasoning about concurrency).

https://www.risc.jku.at/research/formal/software/RISCAL