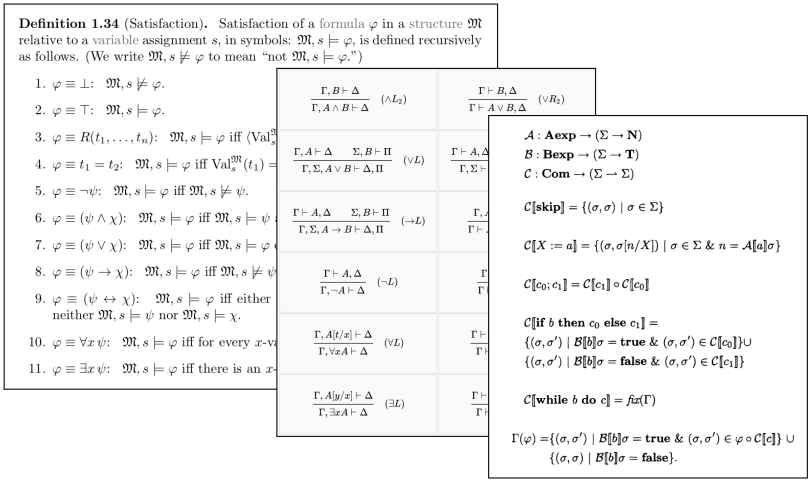
LOGIC & SEMANTIC TECHNOLOGIES FOR COMPUTER SCIENCE EDUCATION

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC), JKU Linz, Austria

Joint work with V. Novitzká, W. Steingartner (TU Kosice)
and A. Biere, D. Cerna, M. Seidl, W. Windsteiger (JKU Linz)

JOHANNES KEPLER
UNIVERSITY LINZ

Logic and Semantics in Education



Definition 1.34 (Satisfaction). Satisfaction of a formula φ in a structure $\mathfrak{M}$ relative to a variable assignment s, in symbols: $\mathfrak{M}, s \models \varphi$, is defined recursively as follows. (We write $\mathfrak{M}, s \not\models \varphi$ to mean "not $\mathfrak{M}, s \models \varphi$.")

1. $\varphi \equiv \bot$: $\mathfrak{M}, s \not\models \varphi$.
2. $\varphi \equiv \top$: $\mathfrak{M}, s \models \varphi$.
3. $\varphi \equiv R(t_1, \ldots, t_n)$: $\mathfrak{M}, s \models \varphi$ iff $\langle \mathrm{Val}_s^{\mathfrak{M}}$
4. $\varphi \equiv t_1 = t_2$: $\mathfrak{M}, s \models \varphi$ iff $\mathrm{Val}_s^{\mathfrak{M}}(t_1) =$
5. $\varphi \equiv \neg\psi$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \not\models \psi$.
6. $\varphi \equiv (\psi \wedge \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \psi$
7. $\varphi \equiv (\psi \vee \chi)$: $\mathfrak{M}, s \models \varphi$ iff $\mathfrak{M}, s \models \varphi$
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 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-va
11. $\varphi \equiv \exists x\, \psi$: $\mathfrak{M}, s \models \varphi$ iff there is an x-

$$\frac{\Gamma, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ (\wedge L_2) \qquad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ (\vee R_2)$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \vee B \vdash \Delta, \Pi}\ (\vee L) \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma, \Sigma \vdash}$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \to B \vdash \Delta, \Pi}\ (\to L) \qquad \frac{\Gamma, A}{\Gamma \vdash}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ (\neg L) \qquad \frac{\Gamma}{\Gamma}$$

$$\frac{\Gamma, A[t/x] \vdash \Delta}{\Gamma, \forall x A \vdash \Delta}\ (\forall L) \qquad \frac{\Gamma \vdash}{\Gamma}$$

$$\frac{\Gamma, A[y/x] \vdash \Delta}{\Gamma, \exists x A \vdash \Delta}\ (\exists L) \qquad \frac{\Gamma \vdash}{\Gamma}$$

$\mathcal{A} : \mathbf{Aexp} \to (\Sigma \to \mathbf{N})$
$\mathcal{B} : \mathbf{Bexp} \to (\Sigma \to \mathbf{T})$
$\mathcal{C} : \mathbf{Com} \to (\Sigma \to \Sigma)$

$\mathcal{C}[\![\mathbf{skip}]\!] = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$

$\mathcal{C}[\![X := a]\!] = \{(\sigma, \sigma[n/X]) \mid \sigma \in \Sigma \ \&\ n = \mathcal{A}[\![a]\!]\sigma\}$

$\mathcal{C}[\![c_0; c_1]\!] = \mathcal{C}[\![c_1]\!] \circ \mathcal{C}[\![c_0]\!]$

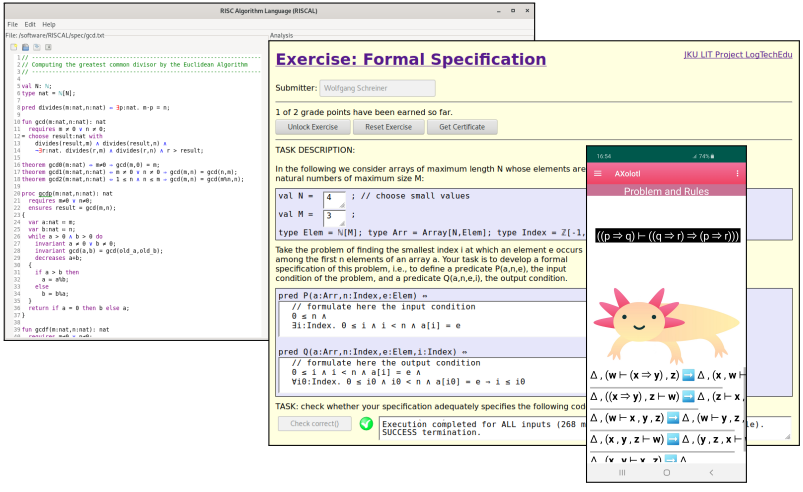
$\mathcal{C}[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!] =$
$\{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true}\ \&\ (\sigma, \sigma') \in \mathcal{C}[\![c_0]\!]\} \cup$
$\{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false}\ \&\ (\sigma, \sigma') \in \mathcal{C}[\![c_1]\!]\}$

$\mathcal{C}[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!] = \mathit{fix}(\Gamma)$

$\Gamma(\varphi) = \{(\sigma, \sigma') \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{true}\ \&\ (\sigma, \sigma') \in \varphi \circ \mathcal{C}[\![c]\!]\} \cup$
$\{(\sigma, \sigma) \mid \mathcal{B}[\![b]\!]\sigma = \mathbf{false}\}.$

Typically still presented as "paper and pencil" topics.

Logic and Semantics in Education



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 logic&semantics-based software for education.

Educating with the Help of Logic&Semantics

- Today much of logic&semantics 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 logic&semantics.
  - May increase the motivation of students to model and to reason.
- The ultimate goal is logic&semantics-based 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 ($\mathscr{J}ane$)
- Logic across the Subjects in Primary, Secondary and Higher Education

Various aspects of the general idea.

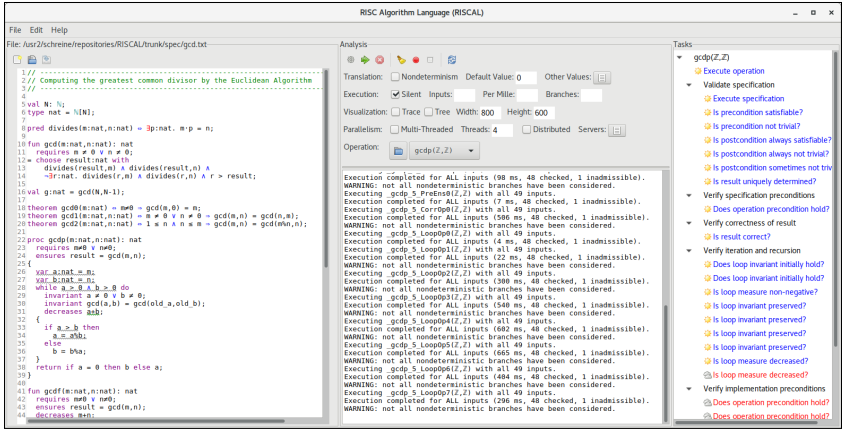
The RISC Algorithm Language (RISCAL)

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 (by evaluating their semantics).
- Correctness of all algorithms is decidable (by evaluating their semantics).
- Automatic generation of (again decidable) verification conditions.

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

The RISCAL Software



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

RISCAL Theories and Theorems

```
val N: ℕ;
type Literal = ℤ[-N,N];
type Clause  = Set[Literal];
type Formula = Set[Clause];
type Valuation  = Set[Literal];

pred satisfies(v:Valuation, l:Literal) ⇔ l∈v;
pred satisfies(v:Valuation, c:Clause) ⇔ ∃l∈c. satisfies(v, l);
pred satisfies(v:Valuation, f:Formula) ⇔ ∀c∈f. satisfies(v,c);
pred satisfiable(f:Formula) ⇔ ∃v:Valuation. satisfies(v,f);
pred valid(f:Formula) ⇔ ∀v:Valuation. satisfies(v,f);
fun not(f: Formula):Formula = { c | c:Clause with ∀d∈f. ∃l∈d. -l∈c };
...
theorem notValid(f:Formula)
  requires formula(f);
⇔ valid(f) ⇔ ¬satisfiable(not(f));
```

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

RISCAL Algorithms

```
multiple pred DPLL(f:Formula)
  requires formula(f);
  ensures result ⇔ satisfiable(f);
  decreases |literals(f)|;
⇔ if f = ∅[Clause] then
    ⊤
  else if ∅[Literal] ∈ f then
    ⊥
  else choose l∈literals(f) in
    DPLL(substitute(f,l)) ∨
    DPLL(substitute(f,-l));
```

```
proc DPLL2(f:Formula): Bool
  requires ...; ensures ...;
{
  var stack: Array[n+1,Formula] :=
    Array[n+1,Formula](∅[Clause]);
  ...
  while ¬satisfiable ∧ number>0 do
    ...
    invariant satisfiable(f) ⇔ ...;
    decreases if satisfiable then 0 else
      ∑k:ℕ[n] with k<number. size(stack[k]);
  {
    ...
    choose l∈literals(g);
    ...
  }
  return satisfiable;
}
```

Functions, procedures, recursion, nondeterminism.

RISCAL Systems

```
proc system(s0:State): Tuple[N[N],Array[N,State]]
  requires init(s0);
{
  var s:State = s0;
  var i:N[N] = 0;
  var t:Array[N,State] = Array[N,Action](s);
  while ¬goal(s) ∧ i < N do
  {
    choose s1:State with next(s,s1);
    s = s1; i = i+1; t[i] = s;
  }
  return hi,ti;
}
```

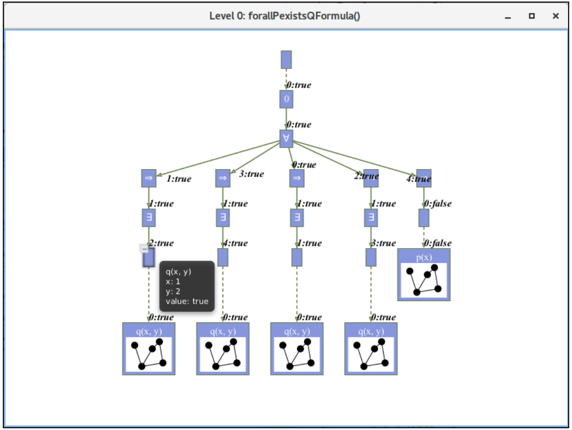
Modeling and analysis of nondeterministic transition systems described by their initial state condition and next state relation.

RISCAL Checking

```
Using N=2.
Type checking and translation completed.
...

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

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

Executing DPLL2(Set[Set[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[Set[ℤ]]) with selected 512 inputs.
Execution completed for SELECTED inputs (609 ms, 512 checked, 0 inadmissible).
```

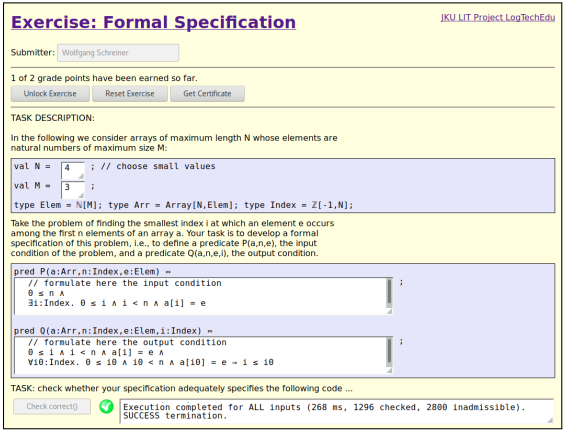
Automatic checking of theorems, algorithms, generated verification conditions.

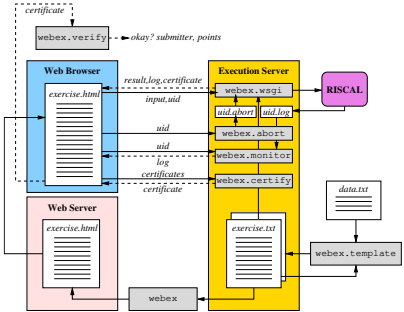
RISCAL Visualization



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

RISCAL Web Exercises

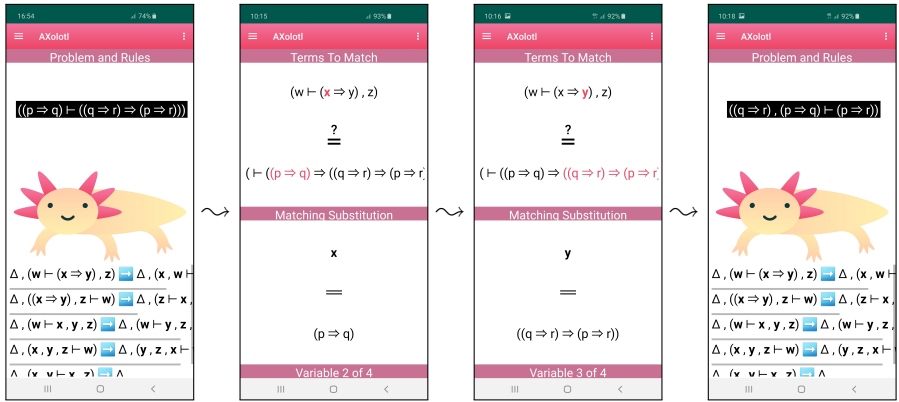




Framework for web-based exercises checked by a RISCAL server.

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).

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).
- 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 (1. sem. 2018, RISCAL web interface not yet available):
  - 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

- Logic&semantics 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 SMT solvers and interactive provers, modeling and reasoning about concurrency).

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