

# COMPUTATIONAL LOGIC

## Course Introduction and Organization



Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)

[Wolfgang.Schreiner@risc.jku.at](mailto:Wolfgang.Schreiner@risc.jku.at)



# Logic

Can be this. . .

$$\frac{\Gamma, A[t/x], \forall xA, \Delta \rightarrow \Lambda}{\Gamma, \forall xA, \Delta \rightarrow \Lambda} (\forall : left) \quad \frac{\Gamma \rightarrow \Delta, A[y/x], \Lambda}{\Gamma \rightarrow \Delta, \forall xA, \Lambda} (\forall : right)$$
$$\frac{\Gamma, A[y/x], \Delta \rightarrow \Lambda}{\Gamma, \exists xA, \Delta \rightarrow \Lambda} (\exists : left) \quad \frac{\Gamma \rightarrow \Delta, A[t/x], \exists xA, \Lambda}{\Gamma \rightarrow \Delta, \exists xA, \Lambda} (\exists : right)$$

Note that in both the  $(\forall : right)$ -rule and the  $(\exists : left)$ -rule, the variable  $y$  does *not* occur free in the lower sequent. In these rules, the variable  $y$  is called the *eigenvariable* of the inference. The condition that the eigenvariable does not occur free in the conclusion of the rule is called the *eigenvariable condition*. The formula  $\forall xA$  (or  $\exists xA$ ) is called the *principal formula* of the inference, and the formula  $A[t/x]$  (or  $A[y/x]$ ) the *side formula* of the inference.

The *axioms* of G are all sequents  $\Gamma \rightarrow \Delta$  such that  $\Gamma$  and  $\Delta$  contain a common formula.

## 5.4.2 Deduction Trees for the System G

First, we define when a sequent is falsifiable or valid.

**Definition 5.4.2** (i) A sequent  $A_1, \dots, A_m \rightarrow B_1, \dots, B_n$  is *falsifiable* iff for some structure  $\mathbf{M}$  and some assignment  $s$ :

$$\mathbf{M} \models (A_1 \wedge \dots \wedge A_m \wedge \neg B_1 \wedge \dots \wedge \neg B_n)[s].$$

Note that when  $m = 0$  the condition reduces to

$$\mathbf{M} \models (\neg B_1 \wedge \dots \wedge \neg B_n)[s]$$

# Logic

but also this:

The screenshot displays the RISCAL Algorithm Language (RISCAL) IDE. The main window is titled "RISCAL Algorithm Language (RISCAL)".

**File:** /home/schreine/papers/RISCALBook/models/sets.txt

**Code (Left Panel):**

```
110
111 theorem bigUnionDef(A:set2) =
112    $\forall x:\text{elem. } x \in \bigcup A \Rightarrow \exists a \in A. x \in a;$ 
113 theorem bigIntersectDef(A:set2) =
114    $\forall x:\text{elem. } x \in \bigcap A \Rightarrow \forall a \in A. x \in a;$ 
115
116 theorem bigUnionEmpty() =
117    $\bigcup \emptyset[\text{set}] = \text{empty};$ 
118 theorem bigIntersectEmpty() =
119    $\bigcap \emptyset[\text{set}] = \text{universe};$ 
120
121 type size = N[M+1];
122 type map = Map[size,elem];
123
124 pred bijection(m:map,n:size,a:set) =
125   ( $\forall i:\text{size. } i2:\text{size. } i1 < i2 \wedge i1 \neq m[i1] \neq m[i2]$ )  $\wedge$ 
126   ( $\forall x \in a. \exists i:\text{size. } i < n. m[i] = x$ );
127
128
129 theorem sizeEquiv(a:set) =
130    $\forall n:\text{size. } n = |a| \Rightarrow \exists m:\text{map. } \text{bijection}(m,n,a);$ 
131
132 theorem sizeEmptyset() =
133    $| \emptyset[\text{elem}] | = 0;$ 
134 theorem sizeUnionIntersect(a:set,b:set) =
135    $|a \cup b| + |a \cap b| = |a| + |b|;$ 
136 theorem sizePowerset(a:set) =
137    $| \text{Set}(a) | = 2^{|a|};$ 
138
139 // -----
140 // relations
141 // -----
142
143 type pair = Tuple[elem,elem];
144 type rel = Set[pair];
145
146 val identity:rel = { (x,x) | x:elem };
147 fun converse(r:rel):rel = { (p.2,p.1) | p \in r };
148 fun comp(r1:rel,r2:rel):rel =
149
```

**Analysis (Right Panel):**

Translation:  Nondeterminism Default Value: 0 Other Values: [ ]

Execution:  Silent Inputs: [ ] Per Mille: [ ] Branches: [ ] Depth: 50

Visualization:  Trace  Tree Width: 800 Height: 600

Parallelism:  Multi-Threaded Threads: 4  Distributed Servers: [ ]

Operation: [ ] sizeEquiv(Set[Z])

RISC Algorithm Language 3.8.6 (June 1, 2021)  
http://www.risc.jku.at/research/formal/software/RISCAL  
(C) 2016-, Research Institute for Symbolic Computation (RISC)  
This is free software distributed under the terms of the GNU GPL.  
Execute "RISCAL -h" to see the available command line options.

-----  
Reading file /home/schreine/papers/RISCALBook/models/sets.txt  
Using #m4.  
Computing the value of empty...  
Computing the value of universe...  
Computing the value of identity...  
Computing the value of univrel...  
Computing the value of N2...  
Using D=1.  
Type checking and translation completed.  
The SMT solver Yices started execution.  
Theorem sizeEquiv is valid.  
Theorem sizeEquiv\_47\_OutputCorrect0 is valid.  
Total time: 1108 ms, translation: 171 ms, decision: 916 ms.

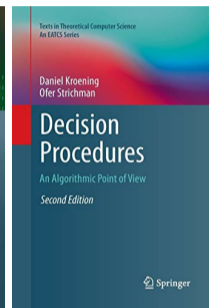
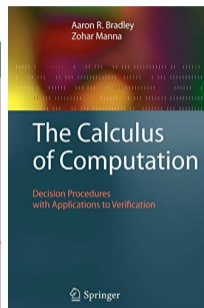
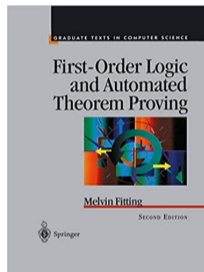
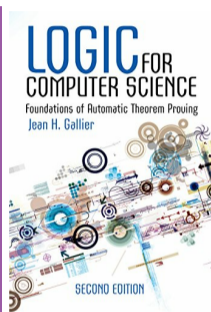
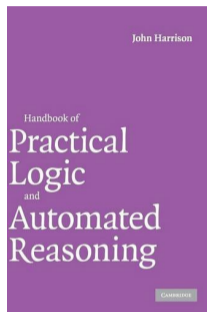
**Tasks (Right Panel):**

- sizeEquiv(Set[Z])
  - Execute operation
  - Validate specification
    - No precondition
    - No postcondition
  - Verify specification preconditions
  - Verify correctness of result
    - Is result correct?
  - Verify iteration and recursion
  - Verify implementation preconditions

# Contents

- **Goal:** an introduction to the **computational aspects** of formal logic.
  - Generally: abstract syntax, formal semantics, proving calculus.
  - Propositional logic: automatic decisions by “SAT solvers”.
  - First-order logic: model checking, automated proving, interactive proving.
  - Quantifier-free logic: automatic decisions over certain theories by “SMT solvers”.
  - Throughout the course: application examples.
- **Prerequisite:** already a **practical understanding** of formal logic.
  - Mathematics: course “Logic as a Working Language” (W. Windsteiger).
  - Computer science: course “Logic” (M. Seidl, W. Schreiner, W. Windsteiger).
  - ...
- **Prepares for:** more in-depth courses on selected topics.
  - Propositional logic: course “SAT Solving” (FMV/Seidl).
  - First-order logic: course “Automated Reasoning” (RISC/Jebelean, Kutsia).
  - ...

# Literature



We will variously present OCaml software from John Harrison's book.

# Software

- Code and resources for “Handbook of Practical Logic and Automated Reasoning”:  
<https://www.cl.cam.ac.uk/~jrh13/atp>
- Sequent Calculus Trainer:  
<https://www.uni-kassel.de/eecs/fmv/software/sequent-calculus-trainer>
- The MiniSat Page: <http://minisat.se/>
- Limboole: <http://fmv.jku.at/limboole>
- The RISC Algorithm Language (RISCAL):  
<https://www.risc.jku.at/research/formal/software/RISCAL>
- The RISC ProofNavigator: <https://www.risc.jku.at/research/formal/software/ProofNavigator/>
- Tree Proof Generator: <https://www.umsu.de/trees>
- SWI Prolog: <https://www.swi-prolog.org>
- Vampire: <https://vprover.github.io>
- Isabelle: <https://isabelle.in.tum.de>
- The Z3 Theorem Prover: <https://github.com/Z3Prover/z3>

# The Course Virtual Machine

No need for self-installation, a virtual machine provides all the software.

<https://www.risc.jku.at/people/schreine/courses/software/#virtual>

**Virtual Machine [Video Presentation]**

You can run a virtual GNU/Linux machine (Debian 10 "buster") with the course software pre-installed on your own (MS Windows or Mac OS X or Linux) computer. All you need is

- A computer with 4 GB main memory and 16 GB free disk space.
- The free "VirtualBox" virtualization software.

To download and install VirtualBox, visit

[VirtualBox](#)

Download the appropriate VirtualBox binary and start the installation as described (MS Windows: just click on the .exe file).

After the installation, download the virtual machine stored in file

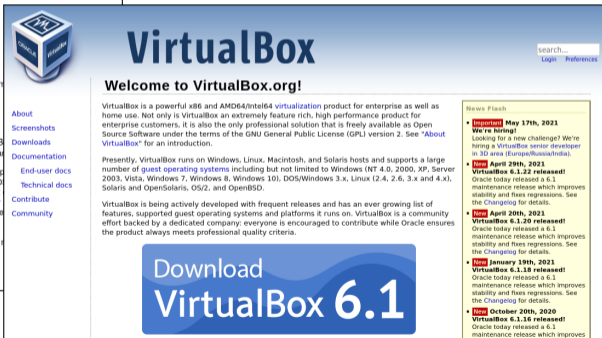
[Debian10RISC.ova](#)

(about 6 GB large). Then start VirtualBox (MS Windows: menu entry "Programs/Oracle VM VirtualBox/VirtualBox" or Linux: menu entry "Applications/Oracle VM VirtualBox/VirtualBox") (choose "File->Import appliance", select the downloaded file, potentially adapt the configuration). Then start the virtual machine.

This virtual machine runs a 64-bit operating system for which you need a 64-bit CPU with hardware virtualization support (VT-x for Intel, AMD-V for AMD). While all modern computers (desktops and notebooks) have this support, it is sometimes switched off in the BIOS. If you get an error when starting the virtual machine, please check whether your computer has hardware virtualization support ([here](#) for more information). If you do not manage to get the virtual machine running you may also try an older [Debian9RISC.ova](#) that does not depend on hardware virtualization support. This machine also provides the course software (and potentially outdated) versions.

When you start the virtual machine, a Debian GNU/Linux system with the Xfce desktop environment starts up. You can log in with

User: guest  
Password: guest



**VirtualBox**

Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

**Download VirtualBox 6.1**

**News Flash**

- **[Important] May 17th, 2021**  
**We're hiring!**  
Looking for a new challenge? We're hiring a VirtualBox senior developer in 3D area (Europe/Russia/India).
- **[New] April 29th, 2021**  
**VirtualBox 6.1.22 released!**  
Oracle today released a 6.1 maintenance release which improves stability and fixes regressions. See the [Changelog](#) for details.
- **[New] April 20th, 2021**  
**VirtualBox 6.1.20 released!**  
Oracle today released a 6.1 maintenance release which improves stability and fixes regressions. See the [Changelog](#) for details.
- **[New] January 19th, 2021**  
**VirtualBox 6.1.18 released!**  
Oracle today released a 6.1 maintenance release which improves stability and fixes regressions. See the [Changelog](#) for details.
- **[New] October 20th, 2020**  
**VirtualBox 6.1.16 released!**  
Oracle today released a 6.1 maintenance release which improves

Just install VirtualBox and import the virtual machine.

# John Harrison's OCaml Software

<https://www.cl.cam.ac.uk/~jrh13/atp>

The course VM provides shell scripts `ocamlprop` (propositional logic) and `ocamlfol` (first-order logic) for simple interactive use.

```
debian10!1> ocamlprop
      OCaml version 4.05.0
Camlp5 parsing version 7.01
# tautology << p /\ (p ==> q) ==> q >> ;;
- : bool = true
# (* press CTRL-D to stop OCaml interpreter *)
```

```
debian10!2> cat >> example.ml
tautology << p /\ (p ==> q) ==> q >> ;;
debian10!3> ocamlprop < example.ml
      OCaml version 4.05.0
Camlp5 parsing version 7.01
# - : bool = true
```

OCaml source code in `/software/Harrison/OCaml/atp_interactive.ml`



## Course Outline

1. Propositional Logic: Syntax and Semantics.
2. Propositional Logic: Proofs.
3. Propositional Logic: Modern SAT Solving.
4. Propositional Logic: Applications of SAT Solving.
5. First-Order Logic: Syntax and Semantics.
6. First-Order Logic: Proofs.
7. First-Order Logic: Software for Proving.
8. First-Order Logic: The Method of Analytic Tableaux.
9. First-Order Logic: The Resolution Method.
10. First-Order Logic: Reasoning about Equality.
11. SMT Solving: Decidable Theories.
12. SMT Solving: Combining Decision Procedures.

# Course Organization

12 units consisting of lectures and exercises.

- **Lectures:** Wolfgang Schreiner.
  - Theoretical and software presentations.
  - Graded by a written exam at the end of the semester.
- **Exercises:** Wolfgang Schreiner/Nikolaj Popov.
  - 10 exercise sheets to be elaborated within two weeks (paper&pencil/software).
  - Grading scheme will be explained in the first exercise unit.
- **Moodle Course:** <https://www.risc.jku.at/people/schreine/courses/ws2021/complogic>
  - Requires self-registration in Moodle and self-enrolment in course.
  - Questions per messages in the “Questions and Answers” forum.
  - Upload exercises as single PDF files (may include photos of handwritten sheets).
    - Possibly an archive with additional “formal” files for use by software.