Progress Report on Leo-II, an Automatic Theorem Prover for Higher-Order Logic

Christoph E. Benzmüller

joint project with: L. Paulson, A. Fietzke, F. Theiss

University of Cambridge

(& Universität des Saarlandes)

ARG-Talk

Cambridge, England, July 3, 2007

LEO-II

An Effective Higher-Order Theorem Prover

UNIVERSITY OF CAMBRIDGE

UNIVERSITÄT DES SAARLANDES

Talk Outline

- Background
- LEO-II as Interactive Proof Assistant
- Automatic Proof Search
- Cooperation with other Reasoning Systems
- Term Sharing and Term Indexing
- First Experiments

LEO-II Background

Project Objectives

LEO-II
An Effective Higher-Order Theorem Prover

- Automatic theorem prover
  - resolution based HO reasoning
  - standalone system; implemented in Objective CAML
  - cooperation with specialist provers, e.g. FO ATPs
  - term sharing and term indexing
  - novel system architecture(s)
- Interactive proof assistant
- Cooperation with interactive proof assistants (not yet)
  - e.g. Isabelle/HOL
  - to support automatic proving of subproblems
  - for verification of own proof objects
- Problem representation language: TPTP THF Syntax

LEO-II Basics

LEO-II
An Effective Higher-Order Theorem Prover

- Logic
  - classical higher-order logic (Church's simple type theory)
  - base types other than ι and o can be specified
  - (strongly) limited support for polymorphism
- Syntax and Notation
  - typed variables: $X_\alpha, Y_\beta, Z_\gamma, X^1_\beta, X^2_\gamma \ldots$
  - typed constants: $c_\alpha, f_{\alpha\to\beta}, \ldots$
    including: T_o, F_o, $\neg_{o\to o}$, $\vee_{o\to o\to o}$, $\Pi_{(\alpha\to o)\to o}$, $=_{\alpha\to\alpha\to o}$
  - other logical connectives are defined as usual
  - abstraction and application terms defined as usual
- Target Semantics
  - Henkin models
  - $== (\lambda X.\lambda Y.Y = X)$

Literals, Uni-Constraints, Clauses

LEO-II An Effective Higher-Order Theorem Prover

- Clauses and Literals

$$\mathcal{C}_1 : [\mathbf{A}_o]^{=\mathsf{T}}, [\mathbf{B}_o]^{=\mathsf{F}}, [\mathbf{C}_\alpha =^\alpha \mathbf{D}_\alpha]^{=\mathsf{T}}, [\mathbf{F}_\alpha =^\alpha \mathbf{G}_\alpha]^{=\mathsf{F}}$$

- Literal atoms are always kept in $\beta\eta$-normal form
- Negative equation literals: *unification constraints*.

$$\mathcal{C}_2 : [\mathbf{A}_o]^{=\mathsf{T}}, [\mathbf{F}_\alpha =^\alpha \mathbf{G}_\alpha]^{=\mathsf{F}} \text{ corresponds to } (\mathbf{F}_\alpha =^\alpha \mathbf{G}_\alpha) \Rightarrow \mathbf{A}_o$$

  - explains the name 'unification constraint'
  - $\mathbf{F}_\alpha$ and $\mathbf{G}_\alpha$ have a free variable at head position: *flex-flex*.
  - only one has a free variable at head position: *flex-rigid*.

Literals, Uni-Constraints, Clauses

LEO-II
An Effective Higher-Order Theorem Prover

- HO unification / pre-unification undecidable and infinitary
- HO pre-unification semi-decidable
- Example of inifinite number of pre-unifiers (H variable, f and a constants)

$$[\mathsf{H}_{\iota\to\iota}(\mathsf{f}_{\iota\to\iota}\mathsf{a}_\iota) = \mathsf{f}_{\iota\to\iota}(\mathsf{H}_{\iota\to\iota}\mathsf{a}_\iota)]^{=\mathsf{F}}$$

$$\mathsf{H} \longleftarrow \lambda \mathsf{x}.\, \underbrace{\mathsf{f}(\mathsf{f} \ldots (\mathsf{f}}_{\mathsf{n}\geq 0}\, \mathsf{x}) \ldots)$$

- LEO-II operates with depth bounded pre-unification
- Definition of empty clause (modulo flex-flex pairs)

$$\mathcal{C} : [\mathsf{F}]^{=\mathsf{T}},\ \underbrace{[\mathbf{A}^1_\alpha =^\alpha \mathbf{B}^1_\alpha]^{=\mathsf{F}}, \ldots, [\mathbf{A}^\mathsf{n}_\alpha =^\alpha \mathbf{B}^\mathsf{n}_\alpha]^{=\mathsf{F}}}_{\text{only flex-flex unification constraints allowed}}$$

LEO-II's Input Language: TPTP THF

- developed together with Geoff Sutcliffe, Florian Rabe, Allen van Gelder, Chad Brown, and others
- supports exchange of HO problems between systems
- THF core (THF0) covers at least simple type theory
- THF0 will be released soon

`http://www.cs.miami.edu/~tptp/TPTP/Proposals/THF.html`

Example 1

LEO-II
An Effective Higher-Order Theorem Prover

- Definitions

$$\begin{aligned}
\text{reflexive} &\stackrel{\text{def}}{=} \lambda R_{\iota\to\iota\to o}.\forall X_\iota.(R\ X\ X) \\
\text{symmetric} &\stackrel{\text{def}}{=} \lambda R_{\iota\to\iota\to o}.\forall X_\iota.\forall Y_\iota.(R\ X\ Y) \Rightarrow (R\ Y\ X) \\
\text{transitive} &\stackrel{\text{def}}{=} \lambda R_{\iota\to\iota\to o}.\forall X_\iota.\forall Y_\iota.\forall Z_\iota.((R\ X\ Y) \wedge (R\ Y\ Z)) \Rightarrow (R\ X\ Z) \\
\text{equiv_rel} &\stackrel{\text{def}}{=} \lambda R_{\iota\to\iota\to o}.(\text{reflexive}\ R) \wedge (\text{symmetric}\ R) \wedge (\text{transitive}\ R)
\end{aligned}$$

- Theorem

$$\exists R_{\iota\to\iota\to o}.\neg(\text{equiv\_rel}\ R)$$

- Example solutions:

$$\begin{aligned}
\{(x,y) | x \neq y\} &\quad \text{represented by} \quad \lambda X_\iota.\lambda Y_\iota.\neg(X = Y) \\
\{(x,y) | \text{false}\} &\quad \text{represented by} \quad \lambda X_\iota.\lambda Y_\iota.F
\end{aligned}$$

THF Example 1

```
thf(reflexiv,definition,
    (reflexive :=
     (^[R:($i>($i>$o))]: (![X:$i]: ((R @ X) @ X))))).

thf(symmetric,definition,
    (symmetric :=
     (^[R:($i>($i>$o))]: (![X:$i,Y:$i]:
      ((R @ X) @ Y) => ((R @ Y) @ X))))).

thf(transitive,definition,
    (transitive :=
     (^[R:($i>($i>$o))]: (![X:$i,Y:$i,Z:$i]:
      ((((R @ X) @ Y) & ((R @ Y) @ Z)) => ((R @ X) @ Z)))))).

thf(equiv_rel,definition,
    (equiv_rel :=
     (^[R:($i>($i>$o))]:
      (reflexive @ R) & (symmetric @ R) & (transitive @ R)))).
thf(test,theorem,(?[R:($i>($i>$o))]: ~(equiv_rel @ R))).
```

LEO-II as Interactive Proof Assistant

LEO-II as Interactive Proof Assistant

- Interactive proof assistant for simple type theory
- Proof kernel: extensional-higher order resolution
- What is this good for?
  - teaching of higher-order reasoning, higher-order unification, and higher-order term data structures
  - debugging of calculus, strategies, heuristics, system architecture(s)
- However, main project goal is proof automation

```
LEO-II> help
* The list of interactive LEO-II commands is:
*  ***** interactive LEO-II calculus rules *****
*   bool <cl>                  - applies boolean extensionality to a clause
[...]
*   cnf-exhaustive <cl>        - exhaustive clause normalisation of a clause
[...]
*   res <cl1> <cl2>            - resolution between two clauses
[...]
*  ***** general commands *****
*   help                       - displays help screen;
*                                type help <command> for help about <command>
*   analyze-index              - displays information on the global index
[...]
*   clause-to-fotptp <cl>      - translates a clause to FOTPTP FOF syntax
[...]
*   flag-fo-translation        - determines the fo-translation to be used
*   flag-max-clause-count <max> - sets an upper limit for generating clauses
[...]
*   prove                      - starts automated proof search
*   prove-directory <dir>      - applies LEO-II to all files in a directory
*   prove-directory-with-fo-atp <dir> <prover> - applies LEO-II (with FO ATP)
*                                to all files in a directory
*   prove-with-fo-atp          - starts automated proof search (with FO ATP)
*   read-problem-string <str>  - reads a problem string in THF syntax
*   read-problem-file <file>   - reads a problem in THF syntax from a file
[...]
*   quit                       - type this if you have enough of LEO-II
LEO-II>
```

Automatic Proof Search

Problem Initialization

LEO-II
An Effective Higher-Order Theorem Prover

- Given:
  definitions $\mathbf{D}_1, \ldots, \mathbf{D}_\mathsf{n}$, axioms $\mathbf{A}_1, \ldots, \mathbf{A}_\mathsf{n}$, conjecture $\mathbf{C}$
- Initialization leads to

$$\mathcal{C}_1 : [\mathbf{A}_1]^{=\mathsf{T}} \quad \ldots \quad \mathcal{C}_\mathsf{n} : [\mathbf{A}_\mathsf{n}]^{=\mathsf{T}} \qquad \mathcal{C}_{\mathsf{n}+1} : [\mathbf{C}]^{=\mathsf{F}}$$

- For our example problem we obtain

$$\mathcal{C}_1 : [\exists \mathsf{R}_{\iota \to \iota \to \mathsf{o}} \centerdot \neg(\mathsf{equiv\_rel\ R})]^{=\mathsf{F}}$$

- What happens with the definitions $\mathbf{D}_1, \ldots, \mathbf{D}_\mathsf{n}$?
  - they are not explicitly represented as clauses
  - they are implicitly maintained as rewrite rules

Example 1 (Contd.)

```
LEO-II> read-problem-file ../problems/SIMPLE-MATHS-5.thf
[...]
LEO-II> show-state
SIGNATURE:
 <base types>  $i  $o
 <type variables> 'A
 <fixed logical symbols>
  false (false) : $o
  [...]
 <defined symbols>
  and (&) : ^ [X:$o,Y:$o] : (~ ((~ X) | (~ Y)))
  [...]
  equiv_rel (equiv_rel) :
    ^ [R:$i>($i>$o)] :
         ((reflexive @ R) & ((symmetric @ R) & (transitive @ R)))
  [...]
 <uninterpreted symbols (upper case: free variables; lower case: constants)>
INDEX:   [...]
ACTIVE: [
2:[ 0:<? [R:$i>($i>$o)] : (~ (equiv_rel @ R)) = $false>-w(1)-i()]
  -mln(1)-w(1)-i(neg_input 1)-fv([ ])
]
PASSIVE: []
[...]
FLAGS:  [...]
LEO-II>
```

Definition Unfolding

- currently LEO-II simultaneously unfolds all definitions before starting proof search
- thereby it benefits from the shared term data structures and the index
- delayed and stepwise definition unfolding, which is needed to successfully prove certain theorems, is future work

Example 1 (Contd.)

```
LEO-II> unfold-defs-exhaustive
[
2:[ 0:<? [R:$i>($i>$o)] : (~ (equiv_rel @ R)) = $false>-w(1)-i() ]
  -mln(1)-w(1)-i(neg_input 1)-fv([ ])
]--- unfold-defs --->
 [
3:[ 0:<~ (! [x0:$i>($i>$o)] : (~ (~ (~ ((~ (! [x1:$i] :
       ((x0 @ x1) @ x1))) | (~ (~ ((~ (! [x1:$i,x2:$i] :
       ((~ ((x0 @ x1) @ x2)) | ((x0 @ x2) @ x1)))) |
       (~ (! [x1:$i,x2:$i,x3:$i] : ((~ (~ ((~ ((x0 @ x1) @ x2)) |
       (~ ((x0 @ x2) @ x3))))) | ((x0 @ x1) @ x3))))))))))))
       = $false>-w(1)-i() ]
  -mln(1)-w(1)-i(unfold_def 2)-fv([ ])
]
LEO-II>
```

Clause Normalization

- CNF rules provided for logical primitives: $\mathsf{T}, \mathsf{F}, \neg, \vee, \Pi^\alpha$ and $=^\alpha$
  - speciality (extensionality in CNF normalization)

$$\frac{\mathcal{C}, [\mathbf{F}_{\beta\to\gamma} = \mathbf{G}_{\beta\to\gamma}]^{=\alpha}}{\mathcal{C}, [\forall \mathsf{X}_\beta\text{.}\mathbf{F}\ \mathsf{X} = \mathbf{G}\ \mathsf{X}]^{=\alpha}}\ =^{\mathsf{T},\mathsf{F}}_{\beta\to\gamma} \qquad \frac{\mathcal{C}, [\mathbf{F}_o = \mathbf{G}_o]^{=\alpha}}{\mathcal{C}, [\mathit{unfold}(\mathbf{F}_o \Leftrightarrow \mathbf{G}_o)]^{=\alpha}}\ =^{\mathsf{T},\mathsf{F}}_{o}$$

  - otherwise CNF normalization still quite naive
- Normalization of clause 3 leads to (the V^i are free variables)

$$\mathcal{C}_{15} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^1]^{=\mathsf{T}} \qquad \mathcal{C}_{25} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^2]^{=\mathsf{T}}, [\mathsf{V}^0\ \mathsf{V}^2\ \mathsf{V}^1]^{=\mathsf{F}}$$

$$\mathcal{C}_{31} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^2]^{=\mathsf{F}}, [\mathsf{V}^0\ \mathsf{V}^2\ \mathsf{V}^3]^{=\mathsf{F}}, [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^3]^{=\mathsf{T}}$$

Example 1 (Contd.)

```
LEO-II> cnf-exhaustive 3
3:[ 0:<~ (! [x0:$i>($i>$o)] : (~ (~ (~ ((~ (! [x1:$i] :
       ((x0 @ x1) @ x1))) | (~ (~ ((~ (! [x1:$i,x2:$i] :
       ((~ ((x0 @ x1) @ x2)) | ((x0 @ x2) @ x1)))) |
       (~ (! [x1:$i,x2:$i,x3:$i] : ((~ (~ ((~ ((x0 @ x1) @ x2)) |
       (~ ((x0 @ x2) @ x3))))) | ((x0 @ x1) @ x3))))))))))))
       = $false>-w(1)-i() ]-mln(1)-w(1)-i(unfold_def 2)-fv([ ])
--- cnf-exhaustive --->
 [
13:[ 0:<(V_x0_1 @ V_x1_2) @ V_x1_2 = $true>-w(1)-i() ]
   -mln(1)-w(1)-i(cnf 11)-fv([ V_x1_2 V_x0_1 ])

25:[ 0:<(V_x0_1 @ V_x1_3) @ V_x2_5 = $false>-w(1)-i()
     1:<(V_x0_1 @ V_x2_5) @ V_x1_3 = $true>-w(1)-i() ]
   -mln(2)-w(2)-i(cnf 23)-fv([ V_x2_5 V_x1_3 V_x0_1 ])

31:[ 0:<(V_x0_1 @ V_x1_4) @ V_x2_6 = $false>-w(1)-i()
     1:<(V_x0_1 @ V_x1_4) @ V_x3_7 = $true>-w(1)-i()
     2:<(V_x0_1 @ V_x2_6) @ V_x3_7 = $false>-w(1)-i() ]
    -mln(3)-w(3)-i(cnf 30)-fv([ V_x3_7 V_x2_6 V_x1_4 V_x0_1 ])
]
LEO-II>
```

Resolution & Co.

LEO-II
An Effective Higher-Order Theorem Prover

- Resolution

$$\frac{\mathcal{C}, [\mathbf{A}]^{=\alpha} \quad \mathcal{D}, [\mathbf{B}]^{=\beta} \quad \alpha \neq \beta \in \{\mathsf{T}, \mathsf{F}\}}{\mathcal{C}, \mathcal{D}, [\mathbf{A} = \mathbf{B}]^{=\mathsf{F}}} \; \mathsf{res}$$

- Factorization

$$\frac{\mathcal{C}, [\mathbf{A}]^{=\alpha}, [\mathbf{B}]^{=\alpha}}{\mathcal{C}, [\mathbf{A}]^{=\alpha}, [\mathbf{A} = \mathbf{B}]^{=\mathsf{F}}} \; \mathsf{fac}$$

  - currently restricted to identical $\mathbf{A}$, $\mathbf{B}$ and handled via simplification rule

- Simplification
  - trivial factorization, deletion of tautologies, deletion of trivially unsatisfiable literals, etc.

Extensional Pre-Unification

- Pre-unification

$$\frac{\mathcal{C}, [\mathbf{M}_{\alpha\to\beta}{=}\mathbf{N}_{\alpha\to\beta}]^{=\mathsf{F}} \quad \mathsf{s}_\alpha \text{ Sk. term}}{\mathcal{C}, [\mathbf{M}\ \mathsf{s}{=}\mathbf{N}\ \mathsf{s}]^{=\mathsf{F}}}\ \text{func}$$

$$\frac{\mathcal{C}, [(\mathsf{h}_\alpha\ \overline{\mathbf{U}^n}{=}\mathsf{h}_\alpha\ \overline{\mathbf{V}^n})]^{=\mathsf{F}}}{\mathcal{C}, [\mathbf{U}^1{=}\mathbf{V}^1]^{=\mathsf{F}}, \ldots, [\mathbf{U}^n = \mathbf{V}^n]^{=\mathsf{F}}}\ \text{dec} \qquad \frac{\mathcal{C}, [\mathbf{A}{=}\mathbf{A}]^{=\mathsf{F}}}{\mathcal{C}}\ \text{triv}$$

$$\frac{\mathcal{C}, [(\mathsf{F}_\gamma\ \overline{\mathbf{U}^n}{=}\mathsf{h}\ \overline{\mathbf{V}^m})]^{=\mathsf{F}} \quad \mathbf{G} \in \mathcal{AB}^{\mathsf{h}}_\gamma}{\mathcal{C}, [\mathsf{F}{=}\mathbf{G}]^{=\mathsf{F}}, [\mathsf{F}\ \overline{\mathbf{U}^n}{=}\mathsf{h}\ \overline{\mathbf{V}^m}]^{=\mathsf{F}}}\ \text{flex-rigid}(\mathsf{F} \leftarrow \mathbf{G})$$

$$\frac{\mathcal{C}, [\mathsf{X}{=}\mathbf{A}]^{=\mathsf{F}} \quad \mathsf{X} \notin \mathbf{Free}(\mathbf{A})}{\{\mathbf{A}/\mathsf{X}\}\mathcal{C}}\ \text{subst}$$

- Extensional Pre-unification

$$\frac{\mathcal{C}, [\mathbf{M}_o{=}\mathbf{N}_o]^{=\mathsf{F}}}{\mathcal{C}, [\mathit{unfold}(\mathbf{M}_o{\Leftrightarrow}\mathbf{N}_o)]^{=\mathsf{F}}}\ \text{bool}$$

  - clause normalization required after application of Bool

Primitive Substitution

LEO-II
An Effective Higher-Order Theorem Prover

- Primitive substitution (blind guessing of sets and relations)

$$\frac{\mathcal{C}, [\mathsf{P}\ \overline{\mathbf{U}^{\mathsf{n}}}]^{=\alpha} \quad \mathbf{G} \in \mathcal{AB}^{\mathsf{T},\mathsf{F},\neg,\vee,\Pi^{\alpha}}}{\{\mathbf{G}/\mathsf{P}\}(\mathcal{C}, [\mathsf{P}\ \overline{\mathbf{U}^{\mathsf{n}}}]^{=\alpha})}\ \text{prim-subst}(\mathsf{P} \leftarrow \mathbf{G})$$

- Example 1 (Contd.)

$$\mathcal{C}_{15} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^1]^{=\mathsf{T}} \qquad \mathcal{C}_{25} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^2]^{=\mathsf{T}}, [\mathsf{V}^0\ \mathsf{V}^2\ \mathsf{V}^1]^{=\mathsf{F}}$$

$$\mathcal{C}_{31} : [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^2]^{=\mathsf{F}}, [\mathsf{V}^0\ \mathsf{V}^2\ \mathsf{V}^3]^{=\mathsf{F}}, [\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^3]^{=\mathsf{T}}$$

$$\frac{[\mathsf{V}^0\ \mathsf{V}^1\ \mathsf{V}^1]^{=\mathsf{T}} \quad \mathbf{G} \in \{\ldots, (\lambda\mathsf{Y}, \mathsf{Z}.\mathsf{F}), \ldots\}}{[\mathsf{F}]^{=\mathsf{T}}}\ \text{prim-subst}(\mathsf{V}^0 \leftarrow \lambda\mathsf{Y}, \mathsf{Z}.\mathsf{F})$$

Paramodulation and Rewriting (ToDo)

- Literals as rewrite rules

$$\frac{[\mathbf{A}]^{=\alpha}, \mathcal{C} \quad \mathcal{D}[\mathbf{B}]_{\mathsf{pl}} \quad \sigma(\mathbf{A}) = \mathbf{B}}{\sigma(\mathcal{D}[\alpha]_{\mathsf{pl}}, \mathcal{C})} \text{ rewr-w-lit}$$

1. $[\mathbf{A}]^{=\alpha}$ is maximal in $[\mathbf{A}]^{=\alpha}, \mathcal{C}$ wrt. term ordering $>$
2. $\sigma(\mathcal{D}[\alpha]_{\mathsf{pl}}, \mathcal{C}) \not> \mathcal{D}[\mathbf{B}]_{\mathsf{pl}}$

- Paramodulation

$$\frac{[\mathbf{A} = \mathbf{C}]^{=\mathsf{T}}, \mathcal{C} \quad \mathcal{D}[\mathbf{B}]_{\mathsf{pl}} \quad \sigma(\mathbf{A}) = \mathbf{B}}{\sigma(\mathcal{D}[\mathbf{C}]_{\mathsf{pl}}, \mathcal{C})} \text{ para}$$

1. $[\mathbf{A} = \mathbf{C}]^{=\mathsf{T}}$ is max. in $[\mathbf{A} = \mathbf{C}]^{=\mathsf{T}}, \mathcal{C}$ wrt. term ordering $>$
2. $\mathbf{A} > \mathbf{C}$
3. $\sigma(\mathcal{D}[\alpha]_{\mathsf{pl}}, \mathcal{C}) \not> \mathcal{D}[\mathbf{B}]_{\mathsf{pl}}$

Example 1 (Contd.)

```
LEO-II> read-problem-file ../problems/SIMPLE-MATHS-5.thf
[...]
LEO-II> prove
3 4 5 6 [...] 317 318
Eureka --- Thanks to Corina!
Here are the empty clauses
 [
319:[ 0:<$false = $true>-w(1)-i() ]-mln(1)-w(1)-i(sim 33)-fv([ ])
]
0.54003: Total Reasoning Time (../problems/SIMPLE-MATHS-5.thf)
```

```
LEO-II (Proof Found!)> show-derivation 319
**** Beginning of derivation protocol ****
1: (? [R:$i>($i>$o)] : (~ (equiv_rel @ R)))=$true
  ---  theorem(file(../problems/SIMPLE-MATHS-5.thf,[test]))
2: (? [R:$i>($i>$o)] : (~ (equiv_rel @ R)))
   =$false
  ---  neg_input 1
3: (~ (! [x0:$i>($i>$o)] : (~ (~ (~ ((~ (! [x1:$i] : ((x0 @ x1) @ x1))) |
   (~ (~ ((~ (! [x1:$i,x2:$i] : ((~ ((x0 @ x1) @ x2)) | ((x0 @ x2) @ x1)))) |
   (~ (! [x1:$i,x2:$i,x3:$i] : ((~ (~ ((~ ((x0 @ x1) @ x2)) |
   (~ ((x0 @ x2) @ x3))))) | ((x0 @ x1) @ x3))))))))))))
   =$false
  ---  unfold_def 2
4: [...]
  ---  cnf 4
6: [...]
  ---  cnf 5
7: [...]
  ---  cnf 6
8: [...]
  ---  cnf 7
10: (~ (! [x1:$i] : ((V_x0_1 @ x1) @ x1)))=$false
   ---  cnf 8
11: (! [x1:$i] : ((V_x0_1 @ x1) @ x1))=$true
   ---  cnf 10
13: ((V_x0_1 @ V_x1_2) @ V_x1_2)=$true  ---  cnf 11
33: ($false)=$true
   ---  prim-subst (V_x0_1 --> lambda [V21]: lambda [V22]: false) 13
319: ($false)=$true  ---  sim 33
**** End of derivation protocol ****
**** no. of clauses: 13 ****
LEO-II (Proof Found!)>
```

```
LEO-II (Proof Found!)> show-derivation-tstp  319
%------------------------------------------------------------------------------
% File       : ../problems/SIMPLE-MATHS-5.thf
[...]
% Comments   : *todo*
%------------------------------------------------------------------------------
%**** Beginning of derivation protocol in tstp ****

 thf(1,theorem,((? [R:$i>($i>$o)] : (~ (equiv_rel @ R)))=$true),
     file(../problems/SIMPLE-MATHS-5.thf,[test])).

 thf(2,plain,((? [R:$i>($i>$o)] : (~ (equiv_rel @ R)))=$false),
     inference(neg_input,[status(thm)],[1])).

[...]

 thf(33,plain,(($false)=$true),
     inference(prim-subst (V_x0_1-->lambda [V21]: lambda [V22]: false),
               [status(thm)],[13])).

 thf(319,plain,(($false)=$true),
     inference(sim,[status(thm)],[33])).

%**** End of derivation protocol in tstp ****
%**** no. of clauses in derivation: 13 ****
LEO-II (Proof Found!)>
```

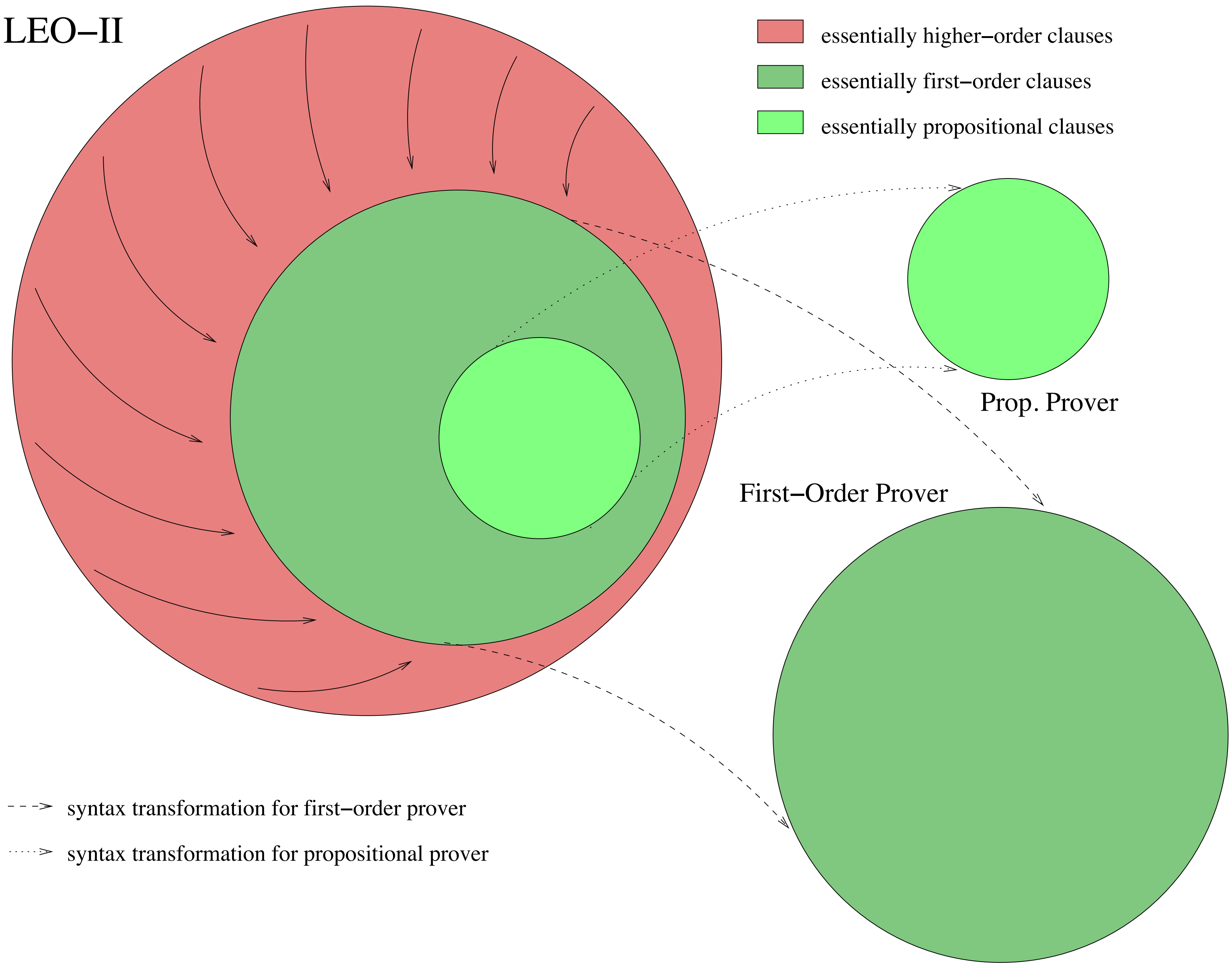
Proof Automation: ToDo List

LEO-II
An Effective Higher-Order Theorem Prover

- term orderings
- efficient paramodulation and rewriting
- adapt overall calculus after adding them
- adapt heuristics and strategies
- efficient realization of remaining rules
- efficient subsumtion
- clever and efficient CNF normalization
- . . .

Cooperation with FO-ATPs

Cooperation with Other Provers



Cooperation with Other Provers

LEO-II
An Effective Higher-Order Theorem Prover

- Provers supported (so far)
  - E, SPASS
- Translations supported so far
  - $@_\alpha$-FO-translation [Kerber94]:
    $(\mathsf{V}^0_{\iota\to\iota\to o}\ \mathsf{V}^1_\iota\ \mathsf{V}^1_\iota) \longrightarrow$
    $@_{(\iota\to o)\to\iota\to o}(@_{(\iota\to\iota\to o)\to\iota\to(\iota\to o)}(\mathsf{V}^0,\mathsf{V}^1),\mathsf{V}^1)$
  - fully typed FO-translation [Hurd02]:
    $(\mathsf{V}^0_{\iota\to\iota\to o}\ \mathsf{V}^1_\iota\ \mathsf{V}^1_\iota) \longrightarrow$
    $\mathsf{ti}(@(\mathsf{ti}(@(\mathsf{ti}(\mathsf{V}^0,\iota\to\iota\to o),\mathsf{ti}(\mathsf{V}^1,\iota)),\iota\to o),\mathsf{ti}(\mathsf{V}^1,\iota)),o)$

Communication with FO-ATPs: TPTP FOF

```
[...]
 fof(leo_II_clause_54,axiom,(((~ lit(ti(at(ti(neg,ft(o,o)),
  ti(at(ti(at(ti(V88,ft(i,ft(i,o))),ti(V_x2_6,i)),ft(i,o)),
  ti(V_x3_7,i)),o)),o))) | (lit(ti(at(ti(neg,ft(o,o)),
  ti(at(ti(at(ti(V88,ft(i,ft(i,o))),ti(V_x1_4,i)),ft(i,o)),
  ti(V_x3_7,i)),o)),o)) | (~ lit(ti(at(ti(neg,ft(o,o)),
  ti(at(ti(at(ti(V88,ft(i,ft(i,o))),ti(V_x1_4,i)),ft(i,o)),
  ti(V_x2_6,i)),o)),o))))))).
[...]
```

Cooperation with FO-ATPs: ToDo List

- add other provers, other systems
- use incremental provers
- provide more FO-translations
- parallel instead of sequential system architecture
- backtranslate proof objects
- . . .

Perfect Term Sharing and Term Indexing

Term Sharing and Term Indexing

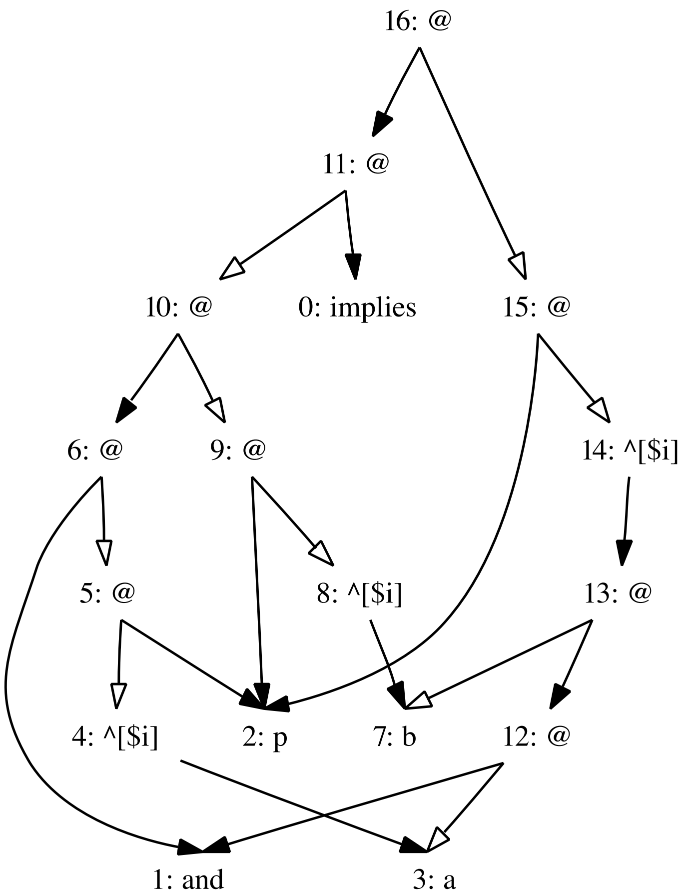
LEO-II
An Effective Higher-Order Theorem Prover

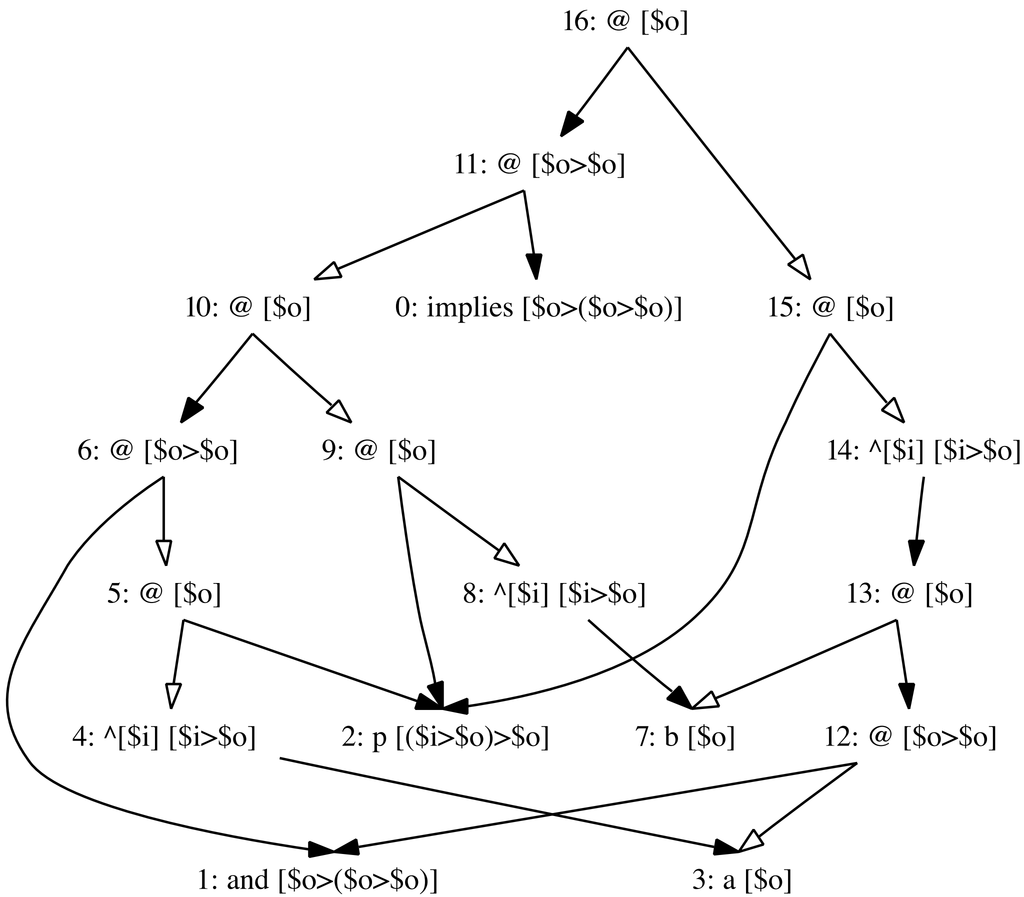
- Term sharing and term indexing widely employed in FO ATPs
- Not much used in HO systems so far
- LEO-II
  - Perfectly shared term data structure
  - DeBruijn-notation for bound variables
  - Indexing of various structural properties
  - Index realized via hashtables
  - Many operations (not all yet!) supported by term indexing
  - We provide tools to analyze the datastructure and the index

Perfect Term Sharing

$$p_{\iota\to o}(\lambda X_\iota.a_o) \wedge p_{\iota\to o}(\lambda X_\iota.b_o) \Rightarrow p_{\iota\to o}(\lambda X_\iota.a_o \wedge b_o)$$

$$\Rightarrow [(\wedge\ (p_{\iota\to o}(\lambda X_\iota.a_o)))\ (p_{\iota\to o}(\lambda X_\iota.b_o))]\ [p_{\iota\to o}(\lambda X_\iota.a_o \wedge b_o)]$$





```
[...]
--- cnf-exhaustive --->
 [
13:[ 0:<(V_x0_1 @ V_x1_2) @ V_x1_2 = $true>-w(1)-i() ]
   -mln(1)-w(1)-i(cnf 11)-fv([ V_x1_2 V_x0_1 ])
25:[ 0:<(V_x0_1 @ V_x1_3) @ V_x2_5 = $false>-w(1)-i()
     1:<(V_x0_1 @ V_x2_5) @ V_x1_3 = $true>-w(1)-i() ]
   -mln(2)-w(2)-i(cnf 23)-fv([ V_x2_5 V_x1_3 V_x0_1 ])
31:[ 0:<(V_x0_1 @ V_x1_4) @ V_x2_6 = $false>-w(1)-i()
     1:<(V_x0_1 @ V_x1_4) @ V_x3_7 = $true>-w(1)-i()
     2:<(V_x0_1 @ V_x2_6) @ V_x3_7 = $false>-w(1)-i() ]
   -mln(3)-w(3)-i(cnf 30)-fv([ V_x3_7 V_x2_6 V_x1_4 V_x0_1 ])
]
[contd.]
```

```
[contd.]
LEO-II> inspect-symbol V_x0_1
Inspecting:
  node 315: V_x0_1
Type:
  $i>($i>$o)
Structure:
  symbol V_x0_1
Parents:
 - as function term:
  node 323: V_x0_1 @ V_x2_5
  node 326: V_x0_1 @ V_x1_4
  node 317: V_x0_1 @ V_x1_2
  node 331: V_x0_1 @ V_x2_6
  node 320: V_x0_1 @ V_x1_3
 total: 5 parents
[contd.]
```

```
[contd.]
Occurs in terms indexed with role:
  node 318: (V_x0_1 @ V_x1_2) @ V_x1_2
   (in Clause:13/0 max pos)
  node 322: (V_x0_1 @ V_x1_3) @ V_x2_5
   (in Clause:25/0 max neg)
  node 324: (V_x0_1 @ V_x2_5) @ V_x1_3
   (in Clause:25/1 max pos)
  node 328: (V_x0_1 @ V_x1_4) @ V_x2_6
   (in Clause:31/0 max neg)
  node 330: (V_x0_1 @ V_x1_4) @ V_x3_7
   (in Clause:31/1 max pos)
  node 332: (V_x0_1 @ V_x2_6) @ V_x3_7
   (in Clause:31/2 max neg)
 total: 6 terms
LEO-II>
```

```
LEO-II> read-problem-file ../problems/SIMPLE-MATHS-5.thf
[...]
LEO-II> analyze-index
------------- The Termset -------------
  0: symbol exists    : ('A>$o)>$o              1 parent(s)
  1: symbol neg       : $o>$o                   1 parent(s)
  2: symbol equiv_rel : ($i>($i>$o))>$o         1 parent(s)
  3: bound($i>($i>$o),0) : $i>($i>$o)           1 parent(s)
  4: appl(2,3)        : $o                      1 parent(s)
  5: appl(1,4)        : $o                      1 parent(s)
  6: abstr($i>($i>$o),5) : ($i>($i>$o))>$o      1 parent(s)
  7: appl(0,6)        : $o                  ---        [$i>($i>$o)/'A]
------------- End Termset -------------
------------- The Termset Analysis -------------
Heavily shared nodes:
Statistics:
 From 0 to 0 bindings: 1 node(s)
 From 0 to 1 bindings: 7 node(s)
Details of dense areas:
 From 0 to 0 bindings: 1 node(s)
 From 1 to 1 bindings: 7 node(s)
Sharing rate: 8 nodes with 7 bindings
Average sharing rate:                                 0.875 bindings per node
Average term size:                                    2.75
Average number of supernodes:                         2.25
Average number of supernodes (symbols):               2.66666666667
Average number of supernodes (nonprimitive terms):  1.5
Rate of term occurrences PST size / term size:        0.440298507463
Rate of symbol occurrences PST size / term size:      0.510204081633
Rate of bound occurrences PST size / term size:       0.636363636364
------------- End Termset Analysis -------------
LEO-II> prove
```

```
LEO-II> prove
3 4 [...] 317 318
Eureka --- Thanks to Corina!
Here are the empty clauses
 [319:[ 0:<$false = $true>-w(1)-i() ]-mln(1)-w(1)-i(sim 33)-fv([ ])]
LEO-II (Proof Found!)> analyze-index
------------- The Termset -------------
  0: symbol exists    : ('A>$o)>$o                6 parent(s)
  1: symbol neg       : $o>$o                     1 parent(s)
[...]
687: appl(684,686)    : $o                         ---
------------- End Termset -------------
------------- The Termset Analysis -------------
Heavily shared nodes:
 6 bindings: exists (node 0)
 48 bindings: neg (node 1)
 [...]
Statistics:
 [...]
 From 22 to 32 bindings: 6 node(s)
 From 39 to 48 bindings: 3 node(s)
Sharing rate: 688 nodes with 1042 bindings
Average sharing rate:                               1.51453488372 bindings per nod
Average term size:                                  8.49273255814
Average number of supernodes:                       6.44040697674
Average number of supernodes (symbols):             16.5897435897
Average number of supernodes (nonprimitive terms): 3.68412162162
Rate of term occurrences PST size / term size:      0.26666121598
Rate of symbol occurrences PST size / term size:    0.405684754522
Rate of bound occurrences PST size / term size:     0.506734951593
------------- End Termset Analysis -------------
LEO-II (Proof Found!)>
```

Our Term Indexing Tools

LEO-II
An Effective Higher-Order Theorem Prover

- may be useful for other tasks
- need to be better exploited in LEO-II
- work out theoretical properties

First Experiments

LEO+FO-ATPs vs. LEO-II+FO-ATPs

- Previous experiments published in:
  - C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber:
    Combined Reasoning by Automated Cooperation.
    Journal of Applied Logic, 2007. To appear.
  - C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber:
    Can a Higher-Order and a First-Order Theorem Prover Cooperate?
    Proc. of LPAR, LNAI 3452, pp. 415-431, 2005. Springer.
- TPTP SET Category:
  - Problems on Sets, Relations and Functions
  - Formulated in first-order set theory
  - Reformulated for experiments in simple type theory
- Computer used in old experiments: 2.4 GHz Xenon, 1GB memory
- In new experiments: 1.6 GHz Intel Pentium, 1 GB memory

LEO+FO-ATPs vs. LEO-II+FO-ATPs

LEO-II
An Effective Higher-Order Theorem Prover

SET171+3	$\forall X_{o\alpha}, Y_{o\alpha}, Z_{o\alpha}. X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$
SET611+3	$\forall X_{o\alpha}, Y_{o\alpha}. (X \cap Y = \emptyset) \Leftrightarrow (X \setminus Y = X)$
SET624+3	$\forall X_{o\alpha}, Y_{o\alpha}, Z_{o\alpha}. \mathtt{Meets}(X, Y \cap Z) \Leftrightarrow \mathtt{Meets}(X, Y) \vee \mathtt{Meets}(X, Z)$
SET646+3	$\forall x_\alpha, y_\beta. \mathtt{Subrel}(\mathtt{Pair}(x, y), (\lambda u_\alpha. T) \times (\lambda v_\beta. T))$
SET670+3	$\forall Z_{o\alpha}, R_{o\beta\alpha}, X_{o\alpha}, Y_{o\beta}. \mathtt{IsRelOn}(R, X, Y) \Rightarrow \mathtt{IsRelOn}(\mathtt{RestrictRDom}(R, Z), Z, Y)$

$\_ \in \_$	:=	$\lambda x_\alpha, A_{o\alpha}.[Ax]$
$\emptyset$	:=	$[\lambda x_\alpha. F]$
$\_ \cap \_$	:=	$\lambda A_{o\alpha}, B_{o\alpha}.[\lambda x_\alpha. x \in A \wedge x \in B]$
$\_ \cup \_$	:=	$\lambda A_{o\alpha}, B_{o\alpha}.[\lambda x_\alpha. x \in A \vee x \in B]$
$\_ \setminus \_$	:=	$\lambda A_{o\alpha}, B_{o\alpha}.[\lambda x_\alpha. x \in A \vee x \notin B]$
$\mathtt{Meets}(\_,\_)$	:=	$\lambda A_{o\alpha}, B_{o\alpha}.[\exists x_\alpha. x \in A \wedge x \in B]$
$\mathtt{Pair}(\_,\_)$	:=	$\lambda x_\alpha, y_\beta.[\lambda u_\alpha, v_\beta. u = x \wedge v = y]$
$\_ \times \_$	:=	$\lambda A_{o\alpha}, B_{o\beta}.[\lambda u_\alpha, v_\beta. u \in A \wedge v \in B]$
$\mathtt{Subrel}(\_,\_)$	:=	$\lambda R_{o\beta\alpha}, Q_{o\beta\alpha}.[\forall x_\alpha, y_\beta. Rxy \Rightarrow Qxy]$
$\mathtt{IsRelOn}(\_,\_,\_)$	:=	$\lambda R_{o\beta\alpha}, A_{o\alpha}, B_{o\beta}.[\forall x_\alpha, y_\beta. Rxy \Rightarrow x \in A \wedge y \in B]$
$\mathtt{RestrictRDom}(\_,\_)$	:=	$\lambda R_{o\beta\alpha}, A_{o\alpha}.[\lambda x_\alpha, y_\beta. x \in A \wedge Rxy]$

LEO+FO-ATPs vs. LEO-II+FO-ATPs

TPTP-Problem	Difficulty	Saturate	Muscadet	E-Setheo	Vampire 7	LEO			LEO-Bliksem					LEO-Vampire				
						Strat.	Cl.	Time	Cl.	Time	FOcl	FOtm	GnCl	Cl.	Time	FOcl	FOtm	GnCl
SET014+4	.67	+	+	+	.01	ST	41	.16	34	6.76	19	.01	7	11	2.6	.01	.01	16
SET017+1	.56	–	–	+	.03	EXT	3906	57.52	25	8.54	16	.01	74	28	5.05	8	.01	22
SET066+1	1.00	?	–	–	–	–	–	–	26	6.80	20	.01	56	38	3.73	17	.01	53
SET067+1	.56	+	+	+	.04	ST	6	.02	13	.32	16	.01	12	9	.1	10	.01	17
SET076+1	.67	+	–	+	.00	–	–	–	10	.47	18	.01	35	12	.97	12	.01	27
SET086+1	.22	+	–	+	.04	ST	2	.01	2	.01	N/A	N/A	N/A	2	.01	N/A	N/A	N/A
SET096+1	.56	+	–	+	.03	–	–	–	27	7.99	14	.01	25	81	7.29	71	0.02	23
SET143+3	.67	+	+	+	68.71	EIR	37	.38	33	7.93	18	.01	19	8	.31	9	.01	9
SET171+3	.67	+	+	–	108.31	EIR	36	.56	25	4.75	19	.01	20	6	.38	10	.01	9
SET580+3	.44	+	+	+	14.71	EIR	25	.19	6	2.73	8	.01	13	8	.23	12	.01	4
SET601+3	.22	+	+	+	168.40	EIR	145	2.20	55	4.96	8	.01	13	20	1.18	31	.01	17
SET606+3	.78	+	–	+	62.02	EIR	21	.33	17	10.8	15	.01	5	5	.27	5	.01	3
SET607+3	.67	+	+	+	65.57	EIR	22	.31	17	7.79	15	.01	6	5	.26	8	.01	3
SET609+3	.89	+	+	–	161.78	EIR	37	.60	26	6.50	19	.01	17	6	.49	10	.01	9
SET611+3	.44	+	–	+	60.20	EIR	996	12.69	72	32.14	38	.01	101	39	4.00	40	0.03	23
SET612+3	.89	+	–	–	113.33	EIR	41	.54	18	3.95	6	.01	7	8	.46	11	.01	9
SET614+3	.67	+	+	–	157.88	EIR	38	.46	19	4.34	16	.01	17	8	.41	9	.01	9
SET615+3	.67	+	+	–	109.01	EIR	38	.57	17	3.59	6	.01	9	6	.47	8	.01	9
SET623+3	1.00	?	–	–	–	EXT	43	8.84	23	9.54	10	.01	14	9	2.27	10	.01	8
SET624+3	.67	+	–	+	.04	ST	4942	34.71	54	9.61	46	.01	212	47	3.29	44	.01	71
SET630+3	.44	+	–	+	60.39	EIR	11	.07	6	.08	8	.01	4	4	.05	6	.01	10
SET640+3	.22	+	–	+	70.41	EIR	2	.01	2	.01	N/A	N/A	N/A	2	.01	N/A	N/A	N/A
SET646+3	.56	+	–	+	59.63	EIR	2	.01	2	.01	N/A	N/A	N/A	2	.01	N/A	N/A	N/A
SET647+3	.56	+	–	+	64.21	EIR	26	.15	13	.30	13	.01	15	7	.12	7	.01	11
SET648+3	.56	+	–	+	64.22	EIR	26	.15	14	.30	13	.01	16	7	.12	9	.01	3
SET649+3	.33	–	–	+	63.77	EIR	45	.30	29	5.49	12	.01	16	10	.25	13	.01	8
SET651+3	.44	–	–	+	63.88	EIR	20	.10	11	.16	10	.01	11	7	.09	8	.01	2
SET657+3	.67	+	–	+	1.44	EIR	2	.01	2	.01	N/A	N/A	N/A	2	.01	N/A	N/A	N/A
SET669+3	.22	–	–	+	.34	EI	6	.19	7	.21	N/A	N/A	N/A	6	.2	N/A	N/A	N/A
SET670+3	1.00	?	–	–	–	EXT	15	.17	17	.36	16	.01	6	9	.14	11	.01	14
SET671+3	.78	–	–	+	218.02	EIR	78	.64	7	2.71	10	.01	14	13	.47	11	.01	9
SET672+3	1.00	?	–	–	–	EXT	27	.4	30	.70	21	.01	11	10	.23	12	.01	14
SET673+3	.78	–	–	+	47.86	EIR	78	.65	14	5.66	14	.01	16	13	.47	17	.01	6
SET680+3	.33	+	–	+	.07	ST	185	.88	29	4.61	18	.01	24	30	2.38	16	.01	27
SET683+3	.22	+	–	+	.06	ST	46	.20	35	8.90	18	.01	24	12	.27	15	.01	4
SET684+3	.78	–	–	+	.33	ST	275	2.45	46	5.95	26	.01	47	41	3.39	35	.01	38
SET686+3	.56	–	–	+	.11	ST	274	2.36	46	5.37	26	.01	46	42	3.55	37	.01	39
SET716+4	.89	+	+	–	–	ST	39	.45	18	3.81	18	.01	118	19	.4	24	0.02	73
SET724+4	.89	+	+	–	–	EXT	154	2.75	18	7.21	15	.01	23	10	1.91	14	.01	20
SET741+4	0.91	?	+	–	–	–	–	–	21	92.76	22	.01	104850	21	3.70	26	.01	570
SET747+4	.89	–	+	–	–	ST	34	.46	25	1.11	18	.01	10	11	1.18	8	.01	14
SET752+4	.89	?	+	–	–	–	–	–	50	6.60	48	.01	4363	50	516.0	48	.01	4145104
SET753+4	.89	?	+	–	–	–	–	–	15	3.07	12	.01	19	12	1.64	12	.01	47
SET764+4	.56	+	+	+	.02	EI	2	.01	2	.01	N/A	N/A	N/A	2	.01	N/A	N/A	N/A
SET770+4	.89	+	+	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Average Total LEO-Vampire ($\surd$) = 12.963

New Experiments with LEO-II

LEO-II
An Effective Higher-Order Theorem Prover

Filename	Fully Typed FO-Translation			$@_\alpha$-FO-Translation		
	Proof	LEO+E (s)	Total (s)	Proof	LEO+E (s)	Total (s)
SET014+4.thf	$\surd$	0.008 0.024	0.032	$\surd$	0.008 0.013	0.021
SET017+1.thf	$\surd$	0.040 0.035	0.075	$\surd$	0.040 0.029	0.069
SET066+1.thf	$\surd$	0.004 0.016	0.020	$\surd$	0.008 0.018	0.026
SET067+1.thf	$\surd$	0.008 0.036	0.044	$\surd$	0.008 0.032	0.040
SET076+1.thf	$\surd$	0.008 0.019	0.027	$\surd$	0.004 0.013	0.017
SET086+1.thf	$\surd$	0.004	0.004	$\surd$	0.004	0.004
SET096+1.thf	$\surd$	0.012 0.021	0.033	$\surd$	0.004 0.016	0.020
SET143+3.thf	$\surd$	0.028 0.037	0.065	$\surd$	0.008 0.015	0.023
SET171+3.thf	$\surd$	0.032 0.034	0.066	$\surd$	0.008 0.019	0.027
SET580+3.thf	$\surd$	0.240 0.083	0.323	$\surd$	0.052 0.038	0.090
SET601+3.thf	$\surd$	0.304 0.184	0.488	$\surd$	0.044 0.028	0.072
SET606+3.thf	$\surd$	0.024 0.034	0.058	$\surd$	0.012 0.015	0.027
SET607+3.thf	$\surd$	0.008 0.024	0.032	$\surd$	0.008 0.015	0.023
SET609+3.thf	$\surd$	0.044 0.047	0.091	$\surd$	0.024 0.036	0.060
SET611+3.thf	$\surd$	0.808 0.293	1.101	$\surd$	0.084 0.026	0.110
SET612+3.thf	$\surd$	0.040 0.041	0.081	$\surd$	0.012 0.016	0.028
SET614+3.thf	$\surd$	0.048 0.076	0.124	$\surd$	0.016 0.034	0.050
SET615+3.thf	$\surd$	0.044 0.056	0.100	$\surd$	0.012 0.019	0.031
SET623+3.thf	$\surd$	8.548 0.858	9.407	$\surd$	1.008 0.064	1.072
SET624+3.thf	$\surd$	0.048 0.092	0.140	$\surd$	0.020 0.021	0.041
SET630+3.thf	$\surd$	0.008 0.023	0.031	$\surd$	0.008 0.018	0.026
SET640+3.thf	$\surd$	0.012	0.012	$\surd$	0.008	0.008
SET646+3.thf	$\surd$	0.012	0.012	$\surd$	0.020	0.020
SET647+3.thf	$\surd$	0.016 0.020	0.036	$\surd$	0.012 0.018	0.030
...	...	...	...	...	...	...

New Experiments with LEO-II

	Fully Typed FO-Translation			$@_\alpha$-FO-Translation		
Filename	Proof	LEO+E (s)	Total (s)	Proof	LEO+E (s)	Total (s)
...	...	...	...	...	...	...
SET648+3.thf	√	0.012 0.020	0.032	√	0.016 0.015	0.031
SET649+3.thf	√	0.016 0.024	0.040	√	0.012 0.018	0.030
SET651+3.thf	√	0.016 0.024	0.040	√	0.012 0.018	0.030
SET657+3.thf	√	0.012	0.012	√	0.008	0.008
SET669+3.thf	√	0.020 0.023	0.043	√	0.020 0.019	0.039
SET670+3.thf	√	0.028 0.039	0.067	√	0.020 0.034	0.054
SET671+3.thf	√	0.020 0.031	0.051	√	0.016 0.019	0.035
SET672+3.thf	√	0.016 0.020	0.036	√	0.016 0.018	0.034
SET673+3.thf	√	0.020 0.031	0.051	√	0.020 0.019	0.039
SET680+3.thf	√	0.020 0.032	0.052	√	0.020 0.016	0.036
SET683+3.thf	√	0.012 0.023	0.035	√	0.032 0.034	0.066
SET684+3.thf	√	0.028 0.041	0.069	√	0.016 0.020	0.036
SET716+4.thf	√	0.012 0.020	0.032	√	0.008 0.019	0.027
SET724+4.thf	√	0.012 0.022	0.034	√	0.012 0.018	0.030
SET741+4.thf	√	0.016 0.037	0.053	√	0.012 0.017	0.029
SET747+4.thf	√	0.012 0.024	0.036	√	0.008 0.019	0.027
SET752+4.thf	√	0.028 0.267	0.295	√	0.020 0.056	0.076
SET753+4.thf	√	0.016 0.021	0.037	√	0.016 0.018	0.034
SET764+4.thf	√	0.008	0.008	√	0.008	0.008
SET770+4.thf	—			—		
	Average Total (√) = 0.312			Average Total (√) = 0.062		

SET770+4

LEO-II An Effective Higher-Order Theorem Prover

$$\forall \mathsf{R}_{\alpha\to\alpha\to o}, \mathsf{Q}_{\alpha\to\alpha\to o}.((\mathsf{equiv\_rel}\ \mathsf{R}) \wedge (\mathsf{equiv\_rel}\ \mathsf{Q})) \Rightarrow$$

$$((\mathsf{equiv\_classes}\ \mathsf{R}) = (\mathsf{equiv\_classes}\ \mathsf{Q}) \vee (\mathsf{disjoint}\ (\mathsf{equiv\_classes}\ \mathsf{R})\ (\mathsf{equiv\_classes}\ \mathsf{Q})))$$

Further Work

LEO-II
An Effective Higher-Order Theorem Prover

Filename	Fully Typed FO-Translation			$@_\alpha$-FO-Translation		
	Proof	LEO+E (s)	Total (s)	Proof	LEO+E (s)	Total (s)
n-bit-adder-base.thf	$\surd$	0.399 12.240	12.640	—		
n-bit-adder-step.thf	—			—		

Summary

- LEO-II: so far 12570 lines of OCAML code, easy to install
  - shared term datastructure, term indexing, inspection tools
  - TPTP THF/FOF parser
  - command line interface
  - calculus
  - proof objects, proof output
  - automated proof search
  - support tools for experiments
  - ...
- Long list of future work
- Now we are entering the fascinating phase
- Biggest problem: stay focused

Why no (full) Polymorphism?

LEO-II
An Effective Higher-Order Theorem Prover

- adds another dimension of complexity and non-determinism:

$\wedge_{o\to o\to o}$	T_o	F_o
T_o	T_o	F_o
F_o	F_o	F_o

$\lambda F_{o\to o}\lambda G_{o\to o}\lambda X_o.(G\,(F\,X))$	$\lambda X_o.X_o$	$\lambda X_o.T$
$\lambda X_o.X_o$	$\lambda X_o.X_o$	$\lambda X_o.T$
$\lambda X_o.T$	$\lambda X_o.T$	$\lambda X_o.T$

- general:

$Op_{\alpha\to\alpha\to\alpha}$	A_α	B_α
A_α	A_α	B_α
B_α	B_α	B_α

$$\begin{aligned}&\exists\alpha.\ \exists Op_{\alpha\to\alpha\to\alpha}.\exists A_\alpha.\exists B_\alpha.\\ &\quad A \neq B\\ &\quad \wedge (OpAA) = A \wedge (OpAB) = B\\ &\quad \wedge (OpBA) = B \wedge (OpBB) = B\end{aligned}$$

- negation and clause normalization (A, B, Op are free variables):

$$\begin{aligned}\mathcal{E}_1 : &[A_\alpha = B_\alpha]^{=T}, [(Op_{\alpha\to\alpha\to\alpha}A_\alpha A_\alpha) = A_\alpha]^{=F}, [(Op_{\alpha\to\alpha\to\alpha}A_\alpha B_\alpha) = B_\alpha]^{=F},\\ &[(Op_{\alpha\to\alpha\to\alpha}B_\alpha A_\alpha) = B_\alpha]^{=F}, [(Op_{\alpha\to\alpha\to\alpha}B_\alpha A_\alpha) = B_\alpha]^{=F}\end{aligned}$$

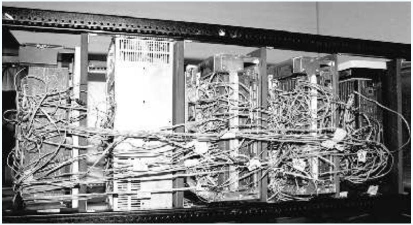
- blind guessing of instances for type variable α in combination with blind guessing of instances for term variable Op required

Overview

1. Higher-Order Logic (HOL)
   - The Good Thing: Expressivitity
   - The Bad Thing: Automation is a Challenge
2. The LEO-II Prover
   - Motivation and Architecture
   - Solving Lightweight Problems
   - Solving Less Lightweight Problems: Multimodal Logics
   - Ongoing and Future Work

Higher-Order Logic (HOL)

Some people say that HOL is like this:



I don't!

- Semantics (extensionality) [PhD-99,JSL-04]
- Proof theory [IJCAR-06]
- ATPs LEO and LEO-II [CADE-98,IJCAR-08]

Higher-Order Logic (HOL)
- on one slide -

Property	**FOL**	**HOL**	**Example**
Quantification over			
- individuals	✓	✓	$\forall x. P(F(x))$
- functions	–	✓	$\forall F. P(F(x))$
- predicates/sets/relations	–	✓	$\forall P. P(F(x))$
Unnamed			
- functions	–	✓	$(\lambda x. x)$
- predicates/sets/relations	–	✓	$(\lambda x. x \neq 2)$
Statements about			
- functions	–	✓	$\mathit{continuous}(\lambda x. x)$
- predicates/sets/relations	–	✓	$\mathit{reflexive}(=)$

Sets and Relations in HOL

$$
\begin{aligned}
A \cup B &:= \{x \mid x \in A \vee x \in B\} \\
A \cup B &:= (\lambda x.\, x \in A \vee x \in B) \\
\cup &:= \lambda A.\, \lambda B.\, (\lambda x.\, x \in A \vee x \in B)
\end{aligned}
$$

$$
\mathtt{symmetric} := \lambda F.\, (\forall x, y.\, F(x, y) = F(y, x))
$$

$$
\mathit{Theorem} : \quad \mathtt{symmetric}(\cup)
$$

Sets and Relations in HOL

$$A \cup B \quad := \quad \{x \mid x \in A \vee x \in B\}$$

$$A \cup B \quad := \quad (\lambda x.\, x \in A \vee x \in B)$$

$$\cup \quad := \quad \lambda A.\, \lambda B.\, (\lambda x.\, x \in A \vee x \in B)$$

$$\texttt{symmetric} \quad := \quad \lambda F.\, (\forall x, y.\, F(x, y) = F(y, x))$$

$$\textit{Theorem}: \quad \texttt{symmetric}(\cup)$$

Sets and Relations in HOL

$$A \cup B := \{x \mid x \in A \vee x \in B\}$$

$$A \cup B := (\lambda x.\, x \in A \vee x \in B)$$

$$\cup := \lambda A.\, \lambda B.\, (\lambda x.\, x \in A \vee x \in B)$$

$$\texttt{symmetric} := \lambda F.\, (\forall x, y.\, F(x, y) = F(y, x))$$

$$\mathit{Theorem}: \quad \texttt{symmetric}(\cup)$$

Sets and Relations in HOL

$$A \cup B := \{x | \ x \in A \vee x \in B\}$$

$$A \cup B := (\lambda x. x \in A \vee x \in B)$$

$$\cup := \lambda A. \lambda B. (\lambda x. x \in A \vee x \in B)$$

$$\texttt{symmetric} := \lambda F. (\forall x, y. F(x, y) = F(y, x))$$

$$\textit{Theorem}: \quad \texttt{symmetric}(\cup)$$

Sets and Relations in HOL

$$A \cup B := \{x \mid x \in A \vee x \in B\}$$

$$A \cup B := (\lambda x. x \in A \vee x \in B)$$

$$\cup := \lambda A. \lambda B. (\lambda x. x \in A \vee x \in B)$$

$$\texttt{symmetric} := \lambda F. (\forall x, y. F(x,y) = F(y,x))$$

$$\textit{Theorem}: \quad \texttt{symmetric}(\cup)$$

Sets and Relations in HOL

Sets and Relations in HOL

$$
\begin{array}{lcl}
\in & := & \lambda x. \lambda A. A(x) \\
\emptyset & := & \lambda x. \bot \\
\cap & := & \lambda A. \lambda B. (\lambda x. x \in A \wedge x \in B) \\
\cup & := & \lambda A. \lambda B. (\lambda x. x \in A \vee x \in B) \\
\setminus & := & \lambda A. \lambda B. (\lambda x. x \in A \vee x \notin B) \\
\ldots & & \\
\subseteq & := & \lambda A. \lambda B. (\forall x. x \in A \Rightarrow x \in B) \\
\mathcal{P} & := & \lambda A. (\lambda B. B \subseteq A) \\
\ldots & & \\
\texttt{reflexive} & := & \lambda R. (\forall x. R(x, x)) \\
\texttt{transitive} & := & \lambda R. (\forall x, y, z. (R(x, y) \wedge R(y, z)) \Rightarrow R(x, z)) \\
\ldots & &
\end{array}
$$

Types are Needed

Typed Sets and Relations in HOL

$$\begin{array}{lll}
\in & := & \lambda x_\alpha \blacksquare \lambda A_{\alpha\to o} \blacksquare A(x) \\
\emptyset & := & \lambda x_\alpha \blacksquare \bot \\
\cap & := & \lambda A_{\alpha\to o} \blacksquare \lambda B_{\alpha\to o} \blacksquare (\lambda x_\alpha \blacksquare x \in A \wedge x \in B) \\
\cup & := & \lambda A_{\alpha\to o} \blacksquare \lambda B_{\alpha\to o} \blacksquare (\lambda x_\alpha \blacksquare x \in A \vee x \in B) \\
\setminus & := & \lambda A_{\alpha\to o} \blacksquare \lambda B_{\alpha\to o} \blacksquare (\lambda x_\alpha \blacksquare x \in A \vee x \notin B) \\
\ldots & &
\end{array}$$

Polymorphism is a Challenge for Automation

- Another source of indeterminism / blind guessing

[TPHOLs-WP-07]

Automation of HOL: A Nightmare?

Undecidable and Infinitary Unification

$\exists F_{\iota\to\iota}. F(g(x)) = g(F(x))$

$\cancel{(1)}$ $F \leftarrow \lambda y_i. y$
$\cancel{(2)}$ $F \leftarrow \lambda y_i. g(y)$
$\cancel{(3)}$ $F \leftarrow \lambda y_i. g(g(y))$
(4) $\ldots$

Automation of HOL: A Nightmare?

Primitive Substitution

Example Theorem: $\exists S.\, \mathit{reflexive}(S)$

Negation and Expansion of Definitions:

$$\neg\exists S.\, (\forall x_\iota.\, S(x,x))$$

Clause Normalisation ($a(S)$ Skolem term):

$$\neg S(a(S), a(S))$$

Guess some suitable instances for S

$S \leftarrow \lambda y.\, \lambda z.\, \top$

$$\rightsquigarrow \neg\top$$

$S \leftarrow \lambda y.\, \lambda z.\, V(y,z) = W(y,z)$

$$\rightsquigarrow V(a(S), a(S)) \neq W(a(S), a(S))$$

$S \leftarrow \ldots$

Automation of HOL: A Nightmare?

Cut rule

$$\frac{A \Rightarrow \mathbf{C} \quad \mathbf{C} \Rightarrow B}{A \Rightarrow B}$$

considered as bad in ATP

Calculi that avoid axioms

- Axiom of excluded middle ✓
- Comprehension axioms ✓
- Functional and Boolean extensionality ✓[CADE-98,PhD-99]
- Leibniz and other definitions of equality ✓[CADE-99,PhD-99]
- Axiom of induction ?
- Axiom of choice –
- Axiom of description –

Automation of HOL: A Nightmare?

Cut rule

$$\frac{A \Rightarrow \mathbf{C} \quad \mathbf{C} \Rightarrow B}{A \Rightarrow B}$$

considered as bad in ATP

[IJCAR-06]: **Axioms that imply Cut** **Calculi that avoid axioms**

- Axiom of excluded middle ✓
- Comprehension axioms ✓
- Functional and Boolean extensionality ✓[CADE-98,PhD-99]
- Leibniz and other definitions of equality ✓[CADE-99,PhD-99]
- Axiom of induction ?
- Axiom of choice –
- Axiom of description –

Automation of HOL: A Nightmare?

Cut rule

$$\frac{A \Rightarrow \mathbf{C} \quad \mathbf{C} \Rightarrow B}{A \Rightarrow B}$$

considered as bad in ATP

Calculi that avoid axioms

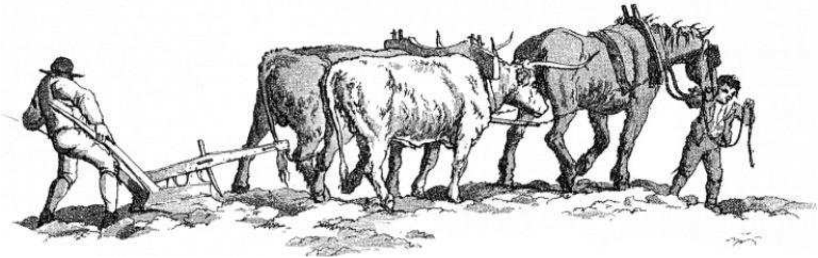
- Axiom of excluded middle ✓
- Comprehension axioms ✓
- Functional and Boolean extensionality ✓[CADE-98,PhD-99]
- Leibniz and other definitions of equality ✓[CADE-99,PhD-99]
- Axiom of induction ?
- Axiom of choice –
- Axiom of description –

LEO-II

An Effective Higher-Order Theorem Prover

UNIVERSITY OF CAMBRIDGE

UNIVERSITÄT DES SAARLANDES



LEO-II employs FO-ATPs: E, Spass, Vampire

Motivation for LEO-II

- TPS system of Peter Andrews et al.

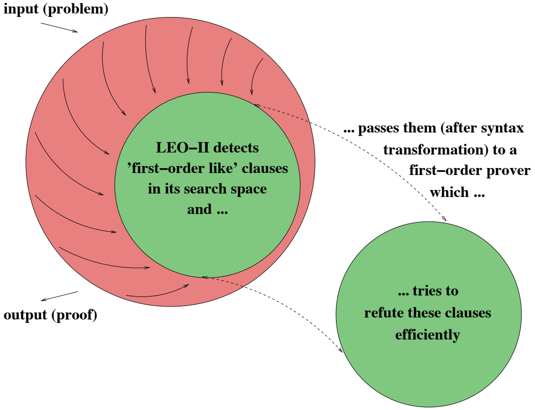
- LEO hardwired to ΩMEGA (predecessor of LEO-II)
- Agent-based architecture Ω-ANTS
  (with V. Sorge) [AIMSA-98,EPIA-99,Calculemus-00]

- Collaboration of LEO with FO-ATP via Ω-ANTS
  (with V. Sorge) [KI-01,LPAR-05,JAL-07]

- Progress in Higher-Order Termindexing
  (with F. Theiss and A. Fietzke) [IWIL-06]

Architecture of LEO-II

Expressivity of Higher-Order Logic



Efficiency of First-Order Logic

Automatic Knowledge Re-Representation

Solving Lightweight Problems

Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$

$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$

$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$

$\forall B, C. (B \cup C = C \cup B)$

$\forall B, C. (B \cap C = C \cap B)$

$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$

$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$

Proof Goal:

$$\forall B, C, D. \quad B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

```
% SPASS---3.0
% Problem : SET171+3
% SPASS beiseite: Ran out of time.

% E---0.999
% Problem : SET171+3
% Failure: Resource limit exceeded
(time)

% Vampire---9.0
% Problem : SET171+3
% Result : Theorem 68.6s
```

Performance: LEO-II + E

```
Eureka --- Thanks to Corina!
Total Reasoning Time: 0.03s
LEO-II (Proof Found!)
```

Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$

$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$

$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$

$\forall B, C. (B \cup C = C \cup B)$

$\forall B, C. (B \cap C = C \cap B)$

$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$

$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$

Proof Goal:

$$\forall B, C, D. \quad B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

```
% SPASS---3.0
% Problem : SET171+3
% SPASS beiseite: Ran out of time.

% E---0.999
% Problem : SET171+3
% Failure: Resource limit exceeded
(time)

% Vampire---9.0
% Problem : SET171+3
% Result : Theorem 68.6s
```

Performance: LEO-II + E

```
Eureka --- Thanks to Corina!
Total Reasoning Time: 0.03s
LEO-II (Proof Found!)
```

Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$

$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$

$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$

$\forall B, C. (B \cup C = C \cup B)$

$\forall B, C. (B \cap C = C \cap B)$

$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$

$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$

Proof Goal:

$\forall B, C, D.$

$B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$

Performance: FO-ATPs

```
% SPASS---3.0
% Problem : SET171+3
% SPASS beiseite: Ran out of time.

% E---0.999
% Problem : SET171+3
% Failure: Resource limit exceeded
(time)

% Vampire---9.0
% Problem : SET171+3
% Result : Theorem 68.6s
```

Performance: LEO-II + E

```
Eureka --- Thanks to Corina!
Total Reasoning Time: 0.03s
LEO-II (Proof Found!)
```

Example: TPTP Problem SET171+3

Axiomatization in FO Set Theory

Assumptions:

$\forall B, C, x. (x \in (B \cup C) \Leftrightarrow x \in B \vee x \in C)$

$\forall B, C, x. (x \in (B \cap C) \Leftrightarrow x \in B \wedge x \in C)$

$\forall B, C. (B \subseteq C \Leftrightarrow \forall x. x \in B \Rightarrow x \in C)$

$\forall B, C. (B \cup C = C \cup B)$

$\forall B, C. (B \cap C = C \cap B)$

$\forall B, C. (B = C \Leftrightarrow B \subseteq C \wedge C \subseteq B)$

$\forall B, C. (B = C \Leftrightarrow \forall x. x \in B \Leftrightarrow x \in C)$

Proof Goal:

$$\forall B, C, D. \quad B \cup (C \cap D) = (B \cup C) \cap (B \cup D)$$

Performance: FO-ATPs

```
% SPASS---3.0
% Problem  : SET171+3
% SPASS beiseite: Ran out of time.

% E---0.999
% Problem  : SET171+3
% Failure: Resource limit exceeded
(time)

% Vampire---9.0
% Problem  : SET171+3
% Result   : Theorem 68.6s
```

Performance: LEO-II + E

```
Eureka --- Thanks to Corina!
Total Reasoning Time: 0.03s
LEO-II (Proof Found!)
```

Example 1b:

$$\neg\forall B, C, D_{\blacksquare} (B \cup (C \cap D) = (B \cup C) \cap (B \cup D))$$

LEO-II: Normalisation, Skolemization ($B_{o\alpha}, C_{o\alpha}, D_{o\alpha}$ Skolem constants)

$$(B \cup (C \cap D)) \neq ((B \cup C) \cap (B \cup D))$$

LEO-II: Definition expansion ($\cap$ and $\cup$)

$$(\lambda x_{\alpha\blacksquare} Bx \vee (Cx \wedge Dx)) \neq (\lambda x_{\alpha\blacksquare} (Bx \vee Cx) \wedge (Bx \vee Dx))$$

LEO-II: Functional and Boolean Extensionality

$$\exists x_{\alpha\blacksquare} (Bx \vee (Cx \wedge Dx)) \neq ((Bx \vee Cx) \wedge (Bx \vee Dx))$$

$$\exists x_{\alpha\blacksquare} (Bx \vee (Cx \wedge Dx)) \not\Leftrightarrow ((Bx \vee Cx) \wedge (Bx \vee Dx))$$

LEO-II: Skolemization (x new Skolem constant)

$$(Bx \vee (Cx \wedge Dx)) \not\Leftrightarrow ((Bx \vee Cx) \wedge (Bx \vee Dx))$$

Example 1b (contd.)

$$(\mathsf{Bx} \lor (\mathsf{Cx} \land \mathsf{Dx})) \not\Leftrightarrow ((\mathsf{Bx} \lor \mathsf{Cx}) \land (\mathsf{Bx} \lor \mathsf{Dx}))$$

LEO-II: Normalization

$$\neg \mathsf{Bx} \qquad \mathsf{Bx} \lor \mathsf{Cx} \qquad \mathsf{Bx} \lor \mathsf{Dx} \qquad \neg \mathsf{Cx} \lor \neg \mathsf{Dx}$$

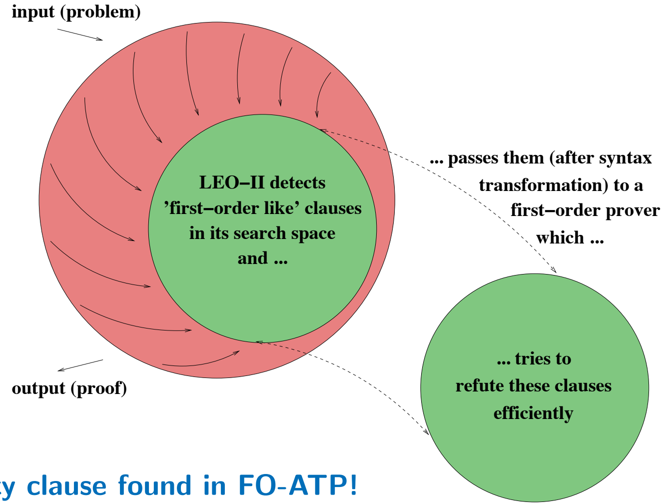
LEO-II: passes clauses to FO-ATP (modulo syntax transformation)

$$\neg @_{(\iota\to o)\to\iota\to o}(B, x) \qquad @_{(\iota\to o)\to\iota\to o}(B, x) \lor @_{(\iota\to o)\to\iota\to o}(C, x)$$

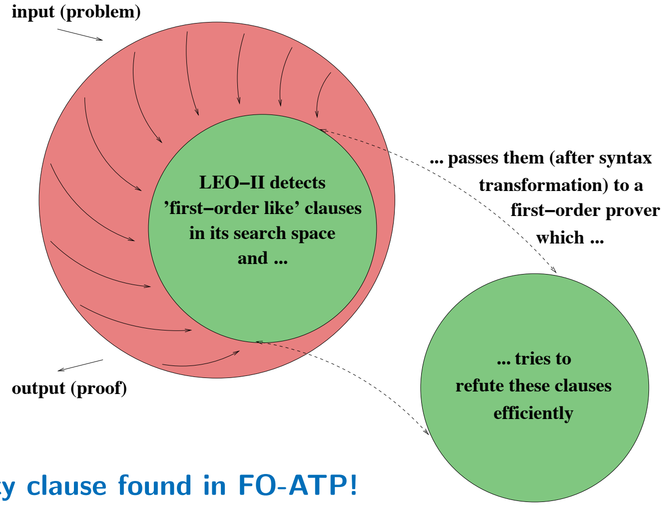
$$@_{(\iota\to o)\to\iota\to o}(B, x) \lor @_{(\iota\to o)\to\iota\to o}(D, x)$$

$$\neg @_{(\iota\to o)\to\iota\to o}(C, x) \lor \neg @_{(\iota\to o)\to\iota\to o}(D, x)$$

Example 1a-b

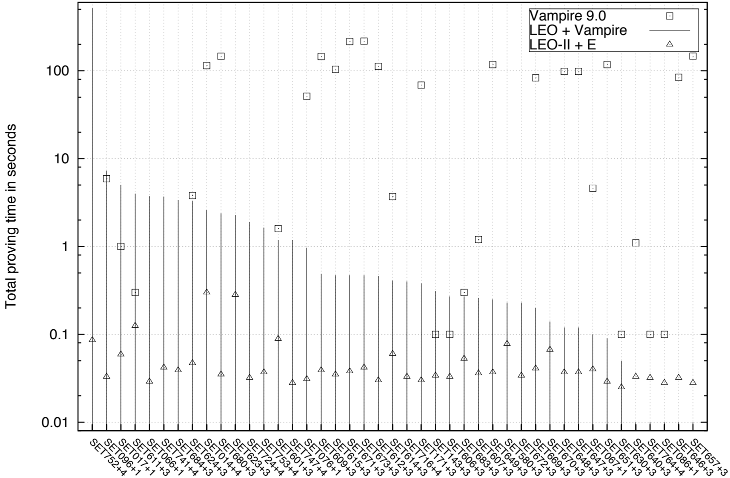


Empty clause found in FO-ATP!



Empty clause found in FO-ATP!

Results



Vampire 9.0
LEO + Vampire
LEO-II + E
Total proving time in seconds
100
10
1
0.1
0.01
SET752+4
SET096+1
SET017+1
SET611+3
SET066+1
SET741+4
SET684+3
SET624+3
SET014+4
SET680+3
SET623+3
SET724+4
SET753+4
SET601+3
SET747+4
SET076+1
SET609+3
SET615+3
SET671+3
SET673+3
SET612+3
SET614+3
SET716+4
SET171+3
SET143+3
SET606+3
SET683+3
SET607+3
SET649+3
SET580+3
SET672+3
SET669+3
SET670+3
SET648+3
SET647+3
SET067+1
SET651+3
SET630+3
SET640+3
SET764+4
SET086+1
SET646+3
SET657+3

Results

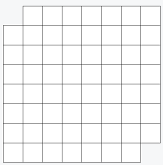
Problem	Vamp. 9.0	LEO+Vamp.	LEO-II+E
014+4	114.5	2.60	0.300
017+1	1.0	5.05	0.059
066+1	–	3.73	0.029
067+1	4.6	0.10	0.040
076+1	51.3	0.97	0.031
086+1	0.1	0.01	0.028
096+1	5.9	7.29	0.033
143+3	0.1	0.31	0.034
171+3	68.6	0.38	0.030
580+3	0.0	0.23	0.078
601+3	1.6	1.18	0.089
606+3	0.1	0.27	0.033
607+3	1.2	0.26	0.036
609+3	145.2	0.49	0.039
611+3	0.3	4.00	0.125
612+3	111.9	0.46	0.030
614+3	3.7	0.41	0.060
615+3	103.9	0.47	0.035
623+3	–	2.27	0.282
624+3	3.8	3.29	0.047
630+3	0.1	0.05	0.025
640+3	1.1	0.01	0.033
646+3	84.4	0.01	0.032
647+3	98.2	0.12	0.037

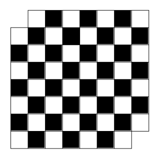
Problem	Vamp. 9.0	LEO+Vamp.	LEO-II+E
648+3	98.2	0.12	0.037
649+3	117.5	0.25	0.037
651+3	117.5	0.09	0.029
657+3	146.6	0.01	0.028
669+3	83.1	0.20	0.041
670+3	–	0.14	0.067
671+3	214.9	0.47	0.038
672+3	–	0.23	0.034
673+3	217.1	0.47	0.042
680+3	146.3	2.38	0.035
683+3	0.3	0.27	0.053
684+3	–	3.39	0.039
716+4	–	0.40	0.033
724+4	–	1.91	0.032
741+4	–	3.70	0.042
747+4	–	1.18	0.028
752+4	–	516.00	0.086
753+4	–	1.64	0.037
764+4	0.1	0.01	0.032

Vamp. 9.0: 2.80GHz, 1GB memory, 600s time limit
LEO+Vamp.: 2.40GHz, 4GB memory, 120s time limit
LEO-II+E: 1.60GHz, 1GB memory, 60s time limit

Representation (and the right System Architecture) Matters!

A general lesson in AI . . .





. . . and a specific lesson here

FOL	HOL
+	+
FO-ATP	LEO-II + FO-ATP

... there is much left to be done!

LEO-II

- Equational Reasoning
- Termindexing
- Handling of Definitions

Cooperat. with Specialist Reasoners

- Monadic Second-Order Logic, Prop. Logic, Arithmetic, ...
- Logic Translations
- Feedback for LEO-II
- Proof Transf./Verification
- Agent-based Architecture

Integration into Proof Assistants

- Relevance of Axioms
- Proof Transf./Verification

International Infrastructure

- TPTP Language(s) for HOL
- Repository of Proof Problems
- HOL Prover Contest

Applications

Logic System Interrelationships, Ontology Reasoning (SUMO, CYC), Formal Methods, CL, ...

... there is much left to be done!

LEO-II

- Equational Reasoning
- Termindexing
- Handling of Definitions

Cooperat. with Specialist Reasoners

- Monadic Second-Order Logic, Prop. Logic, Arithmetic, ...
- Logic Translations
- Feedback for LEO-II
- Proof Transf./Verification
- Agent-based Architecture

Integration into Proof Assistants

- Relevance of Axioms
- Proof Transf./Verification

International Infrastructure

- TPTP Language(s) for HOL
- Repository of Proof Problems
- HOL Prover Contest

Applications

Logic System Interrelationships, Ontology Reasoning (SUMO, CYC), Formal Methods, CL, ...

... there is much left to be done!

LEO-II

- Equational Reasoning
- Termindexing
- Handling of Definitions

Cooperat. with Specialist Reasoners

- Monadic Second-Order Logic, Prop. Logic, Arithmetic, ...
- Logic Translations
- Feedback for LEO-II
- Proof Transf./Verification
- Agent-based Architecture

Integration into Proof Assistants

- Relevance of Axioms
- Proof Transf./Verification

International Infrastructure

- TPTP Language(s) for HOL
- Repository of Proof Problems
- HOL Prover Contest

Applications

Logic System Interrelationships, Ontology Reasoning (SUMO, CYC), Formal Methods, CL, ...

... there is much left to be done!

LEO-II

- Equational Reasoning
- Termindexing
- Handling of Definitions

Cooperat. with Specialist Reasoners

- Monadic Second-Order Logic, Prop. Logic, Arithmetic, ...
- Logic Translations
- Feedback for LEO-II
- Proof Transf./Verification
- Agent-based Architecture

Integration into Proof Assistants

- Relevance of Axioms
- Proof Transf./Verification

International Infrastructure

- TPTP Language(s) for HOL
- Repository of Proof Problems
- HOL Prover Contest

Applications

Logic System Interrelationships, Ontology Reasoning (SUMO, CYC), Formal Methods, CL, ...

... there is much left to be done!

LEO-II

- Equational Reasoning
- Termindexing
- Handling of Definitions

Cooperat. with Specialist Reasoners

- Monadic Second-Order Logic, Prop. Logic, Arithmetic, ...
- Logic Translations
- Feedback for LEO-II
- Proof Transf./Verification
- Agent-based Architecture

Integration into Proof Assistants

- Relevance of Axioms
- Proof Transf./Verification

International Infrastructure

- TPTP Language(s) for HOL
- Repository of Proof Problems
- HOL Prover Contest

Applications

Logic System Interrelationships, Ontology Reasoning (SUMO, CYC), Formal Methods, CL, ...

More Information on LEO-II

- Website with online version of LEO-II:

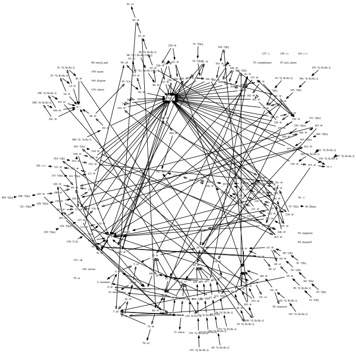
  `http://www.ags.uni-sb.de/~leo`

- System description [IJCAR-08]
- TPTP THF input syntax [IJCAR-THF-08]
- Reasoning in and about multimodal logic [Festschrift-Andrews-08]

Latest Application of LEO-II: Dancefloor Animation

Grooving to an animation of LEO-II's dynamically growing termgraph (while LEO-II is proving Cantor's theorem)

Termsharing



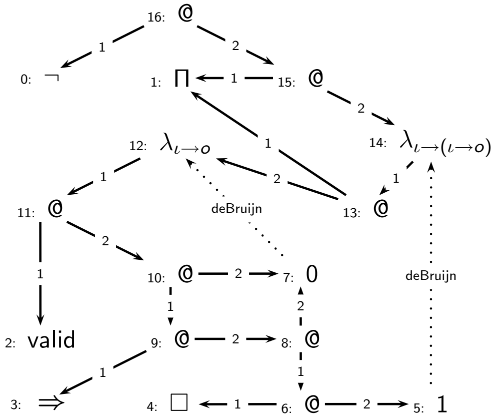
In LEO-II:

- Terms as unique instances
- Perfect Term Sharing
- Shallow data structures

Features:

- β-η-normalization
- DeBruijn indices
- local contexts for polymorphic type variables

Term Graph for: $\neg\forall R.\forall A.(\mathbf{valid}(\Box_R\, A \Rightarrow A))$



Term graph videos: `http://www.ags.uni-sb.de/~leo/art`

LEO-II version 1.5

Christoph Benzmüller[1] and Nik Sultana[2]

Freie Universität Berlin, Germany / Cambridge University, UK

Proof Exchange for Theorem Provers (PxTP)
Lake Placid, NY, USA, 2013

[1]Thanks to: DFG Heisenberg Fellowship BE-2501/9-1
[2]Thanks to: Grant from the German Academic Exchange Service (DAAD)

Talk Outline

Freie Universität Berlin

A: Introduction

Motivation for LEO prover(s)
Logic HOL / TPTP THF0
Reasoning principles of LEO provers
LEO-II

B: New Stuff in LEO-II

Support for different FOL translations
Integration of proofs from EP
Improved support for back-end provers
Detection/removal of Leibniz- and Andrews-equality
Support for choice in LEO-II
Further improvements
Experiments

C: Conclusion

A: Motivation for LEO prover(s)

OMEGA [BenzmüllerEtAl,CADE,1996][SiekmannEtAl,JApplLog,2006]:

- proof assistant with a focus on AI techniques
  - proof planning & agents
  - system integration: ATPs, computer algebra systems
  - knowledge management tools: MAYA
  - E-learning, tutorial NL dialog, user interfaces, ...
- foundation: classical higher-order logic (HOL) & ND calculus
- developed from early 90s until 'J. Siekmann's retirement'

LEO [BenzmüllerKohlhase,CADE,1998]

- **L**ogical **E**ngine of **O**MEGA
- traditional ATP for HOL; based on (RUE-)resolution
- originally implemented within the OMEGA framework
- early investigation of agent based cooperation with FO-ATPs in OMEGA

A: Motivation for LEO prover(s)

OMEGA [BenzmüllerEtAl,CADE,1996][SiekmannEtAl,JApplLog,2006]:

- proof assistant with a focus on AI techniques
  - proof planning & agents
  - system integration: ATPs, computer algebra systems
  - knowledge management tools: MAYA
  - E-learning, tutorial NL dialog, user interfaces, ...
- foundation: classical higher-order logic (HOL) & ND calculus
- developed from early 90s until 'J. Siekmann's retirement'

LEO [BenzmüllerKohlhase,CADE,1998]

- **L**ogical **E**ngine of **O**MEGA
- traditional ATP for HOL; based on (RUE-)resolution
- originally implemented within the OMEGA framework
- early investigation of agent based cooperation with FO-ATPs in OMEGA

A: Logic HOL / TPTP THF0

- Simple Types $\alpha ::= \iota \mid \mu \mid o \mid \alpha_1 \rightarrow \alpha_2$
- HOL Syntax

$$
\begin{aligned}
s, t \quad ::= \quad & c_\alpha \mid X_\alpha \\
& \mid (\lambda X_\alpha . s_\beta)_{\alpha\rightarrow\beta} \mid (s_{\alpha\rightarrow\beta}\, t_\alpha)_\beta \\
& \mid (\neg_{o\rightarrow o}\, s_o)_o \mid (s_o \vee_{o\rightarrow o\rightarrow o} t_o)_o \mid \quad \underbrace{(\forall X_\alpha . t_o)_o}_{(\Pi_{(\alpha\rightarrow o)\rightarrow o}\,(\lambda X_\alpha . t_o))_o}
\end{aligned}
$$

- HOL is (meanwhile) well understood
  - Origin [Church,J.Symb.Log.,1940]
  - Henkin-Semantics [Henkin,J.Symb.Log.,1950] [Andrews, J.Symb.Log.,1971,1972]
  - Extens./Intens. [BenzmüllerEtAl.,J.Symb.Log.,2004] [Muskens,J.Symb.Log.,2007]

- TPTP THF0: HOL with Henkin-Semantics and Choice

A: Reasoning principles of LEO provers

- extensional higher-order RUE-resolution
- see [Benzmüller,Synthese,2002] or [SultanaBenzmüller,IWIL,2012] for more information

Here, I sketch the idea using a very simple example: `SET171^3`

A: Reasoning principles of LEO provers (SET171^3)

$$\forall B_{\iota\to o}, C_{\iota\to o}, D_{\iota\to o}. (B \cup (C \cap D) = (B \cup C) \cap (B \cup D))$$

negation, def. expansion ($\cup := \lambda S. \lambda T. \lambda X. SX \vee TX$ / $\cap := \ldots$)

$$\neg\forall B, C, D. (\lambda X_\alpha. BX \vee (CX \wedge DX)) = (\lambda X_\alpha. (BX \vee CX) \wedge (BX \vee DX))$$

normalisation, Skolemization (b, c, d new Skolem constant)

$$(\lambda X_\alpha. bX \vee (cX \wedge dX)) \neq (\lambda X_\alpha. (bX \vee cX) \wedge (bX \vee dX))$$

functional and Boolean extensionality (extensional pre-unification)

$$\exists X_\alpha. (bX \vee (cX \wedge dX)) \not\Leftrightarrow ((bX \vee cX) \wedge (bX \vee dX))$$

Skolemization (x new Skolem constant)

$$(bx \vee (cx \wedge dx)) \not\Leftrightarrow ((bx \vee cx) \wedge (bx \vee dx))$$

A: Working principles of LEO-II (SET171^3)

$$(bx \vee (cx \wedge dx)) \not\Leftrightarrow ((bx \vee cx) \wedge (bx \vee dx))$$

normalization

$$\neg bx \qquad bx \vee cx \qquad bx \vee dx \qquad \neg cx \vee \neg dx$$

passes clauses to FO-ATP

$$\neg @_{(\iota\to o)\to\iota\to o}(b,x) \qquad @_{(\iota\to o)\to\iota\to o}(b,x) \vee @_{(\iota\to o)\to\iota\to o}(c,x)$$

$$@_{(\iota\to o)\to\iota\to o}(b,x) \vee @_{(\iota\to o)\to\iota\to o}(d,x)$$

$$\neg @_{(\iota\to o)\to\iota\to o}(c,x) \vee \neg @_{(\iota\to o)\to\iota\to o}(d,x)$$

syntax transformation used here: [Kerber,PhD,1992]

Remark: SET171+3 is still a challenge for problem for FO-ATPs — Vampire-2.6, SPASS-3.7, EP-1.7, and Z3-4.0 (in standard mode) do not return proofs within 600s!!!

A: Working principles of LEO-II (SET171^3)

Freie Universität Berlin

$$(bx \vee (cx \wedge dx)) \not\Leftrightarrow ((bx \vee cx) \wedge (bx \vee dx))$$

normalization

$$\neg bx \qquad bx \vee cx \qquad bx \vee dx \qquad \neg cx \vee \neg dx$$

passes clauses to FO-ATP

$$\neg @_{(\iota\to o)\to\iota\to o}(b,x) \qquad @_{(\iota\to o)\to\iota\to o}(b,x) \vee @_{(\iota\to o)\to\iota\to o}(c,x)$$

$$@_{(\iota\to o)\to\iota\to o}(b,x) \vee @_{(\iota\to o)\to\iota\to o}(d,x)$$

$$\neg @_{(\iota\to o)\to\iota\to o}(c,x) \vee \neg @_{(\iota\to o)\to\iota\to o}(d,x)$$

syntax transformation used here: [Kerber,PhD,1992]

Remark: SET171+3 is still a challenge for problem for FO-ATPs — Vampire-2.6, SPASS-3.7, EP-1.7, and Z3-4.0 (in standard mode) do not return proofs within 600s!!!

An Illustrating Example

$$(\mathsf{p}\ (\lambda \mathsf{X}_{\iota\to\iota}.((\mathsf{q}\ \mathsf{X}) \Rightarrow (\mathsf{R}\ \mathsf{X}))))$$

$$\neg(\mathsf{p}\ (\lambda \mathsf{Y}_{\iota\to\iota}.(\neg(\mathsf{q}\ \mathsf{Y}) \vee (\mathsf{r}\ \mathsf{Y}))))$$

An Illustrating Example

$$(\mathsf{p}\ (\lambda X_{\iota\to\iota}.((\mathsf{q}\ X) \Rightarrow (R\ X))))$$
$$\neg(\mathsf{p}\ (\lambda Y_{\iota\to\iota}.(\neg(\mathsf{q}\ Y) \vee (\mathsf{r}\ Y))))$$

- resolution:

$$(\mathsf{p}\ (\lambda X_{\iota\to\iota}.((\mathsf{q}\ X) \Rightarrow (R\ X)))) \neq (\mathsf{p}\ (\lambda Y_{\iota\to\iota}.(\neg(\mathsf{q}\ Y) \vee (\mathsf{r}\ Y))))$$

An Illustrating Example

$$(\mathsf{p}\ (\lambda X_{\iota\to\iota}.((\mathsf{q}\ X) \Rightarrow (R\ X))))$$

$$\neg(\mathsf{p}\ (\lambda Y_{\iota\to\iota}.(\neg(\mathsf{q}\ Y) \vee (\mathsf{r}\ Y))))$$

- resolution:

$$(\mathsf{p}\ (\lambda X_{\iota\to\iota}.((\mathsf{q}\ X) \Rightarrow (R\ X)))) \neq (\mathsf{p}\ (\lambda Y_{\iota\to\iota}.(\neg(\mathsf{q}\ Y) \vee (\mathsf{r}\ Y))))$$

- decomposition:

$$(\lambda X_{\iota\to\iota}.((\mathsf{q}\ X) \Rightarrow (R\ X))) \neq (\lambda Y_{\iota\to\iota}.(\neg(\mathsf{q}\ Y) \vee (\mathsf{r}\ Y)))$$

An Illustrating Example

$$(p\ (\lambda X_{\iota\to\iota}.((q\ X) \Rightarrow (R\ X))))$$

$$\neg(p\ (\lambda Y_{\iota\to\iota}.(\neg(q\ Y) \vee (r\ Y))))$$

- resolution:

$$(p\ (\lambda X_{\iota\to\iota}.((q\ X) \Rightarrow (R\ X)))) \neq (p\ (\lambda Y_{\iota\to\iota}.(\neg(q\ Y) \vee (r\ Y))))$$

- decomposition:

$$(\lambda X_{\iota\to\iota}.((q\ X) \Rightarrow (R\ X))) \neq (\lambda Y_{\iota\to\iota}.(\neg(q\ Y) \vee (r\ Y)))$$

- functional and Boolean extensionality:

$$\neg\forall Z_{\iota\to\iota}.(((q\ Z) \Rightarrow (R\ Z)) \Leftrightarrow (\neg(q\ Z) \vee (r\ Z)))$$

An Illustrating Example

$$(p\ (\lambda X_{\iota\to\iota}.((q\ X) \Rightarrow (R\ X))))$$

$$\neg(p\ (\lambda Y_{\iota\to\iota}.(\neg(q\ Y) \vee (r\ Y))))$$

- clause normalisation

$$\neg(q\ s_{\iota\to\iota}) \vee (R\ s_{\iota\to\iota})$$

$$(q\ s_{\iota\to\iota}) \qquad \neg(r\ s_{\iota\to\iota})$$

An Illustrating Example

$$(p\ (\lambda X_{\iota\to\iota}.((q\ X) \Rightarrow (R\ X))))$$
$$\neg(p\ (\lambda Y_{\iota\to\iota}.(\neg(q\ Y) \vee (r\ Y))))$$

- clause normalisation

$$\neg(q\ s_{\iota\to\iota}) \vee (R\ s_{\iota\to\iota})$$

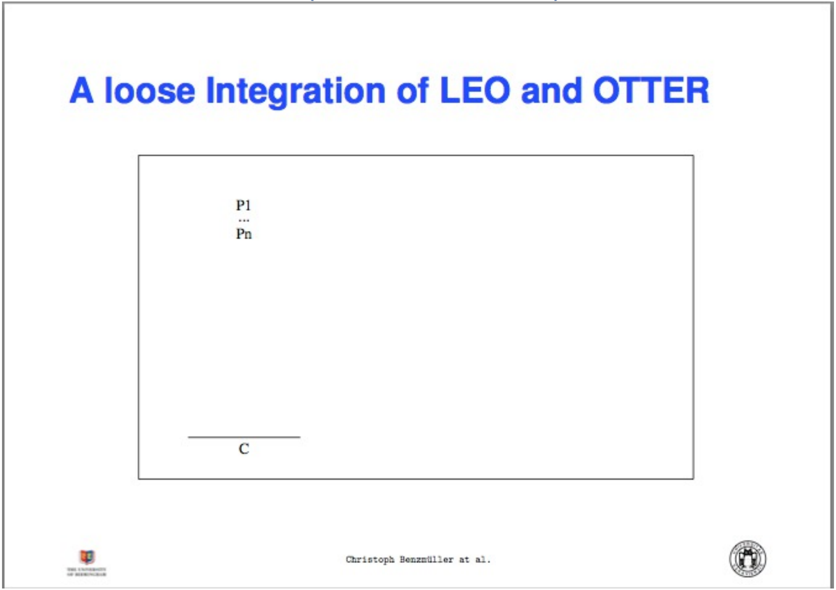
$$(q\ s_{\iota\to\iota}) \qquad \neg(r\ s_{\iota\to\iota})$$

- mapping to first-order

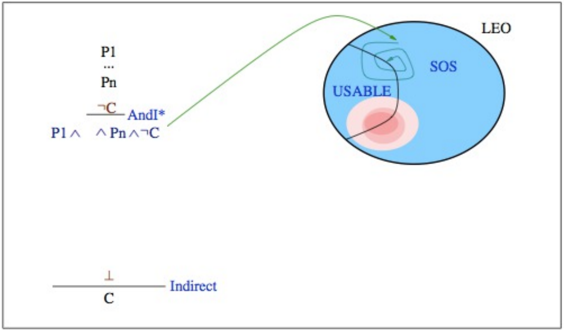
$$\neg @_{((\iota\to\iota)\to o)\_(\iota\to\iota)}(q, s) \vee @_{((\iota\to\iota)\to o)\_(\iota\to\iota)}(R, s)$$

$$@_{((\iota\to\iota)\to o)\_(\iota\to\iota)}(q, s) \qquad \neg @_{((\iota\to\iota)\to o)\_(\iota\to\iota)}(r, s)$$

A: Some slides from 2000 (Deduktionstreffen)



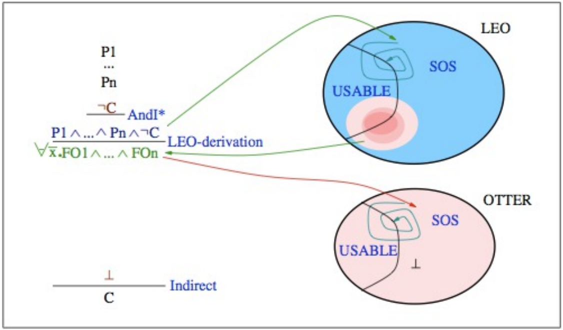
A loose Integration of LEO and OTTER



Christoph Benzmüller at al.

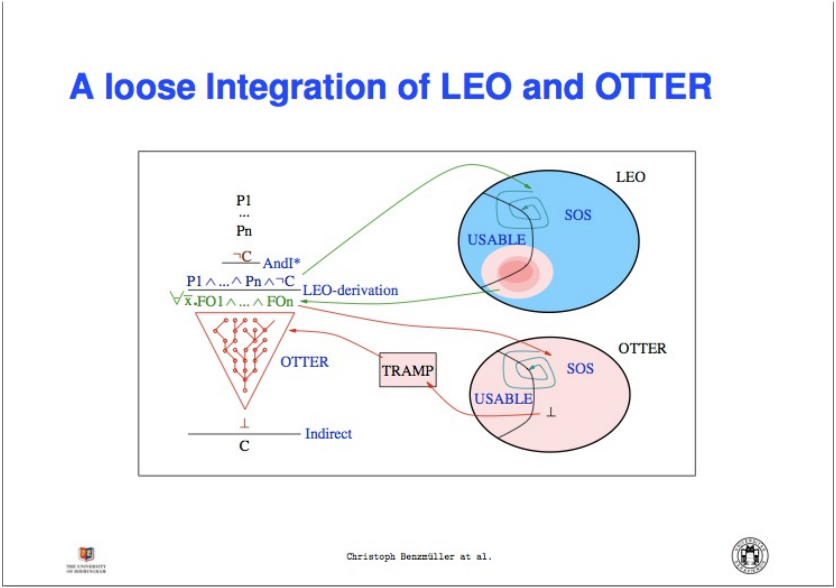
Freie Universität Berlin

A loose Integration of LEO and OTTER

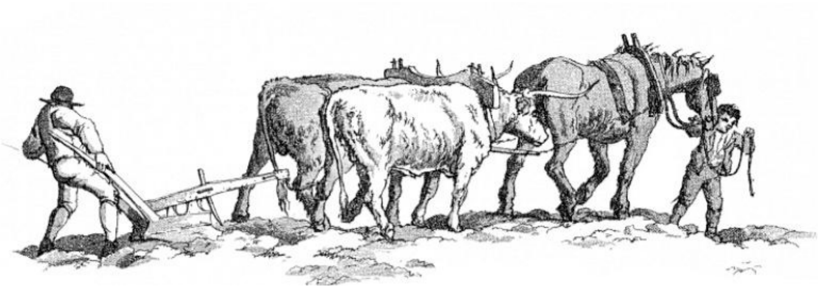


Christoph Benzmüller at al.

A: Some slides from 2000 (Deduktionstreffen)



A: Philosophy of LEO provers: tight collaboration



LEO resp. LEO-II **(Otter), EP, Spass, Vampire**

A: LEO-II

- website: http://leoprover.org
- developed since 2006/07
  (initial funding: project with Larry Paulson at Cambridge)
- independent implementation in OCaml
- direct collaboration with FO-ATPs: EP (Schulz) as first choice
- applications — THF0 provers as universal reasoners
  - HOL
  - quantified modal logics [ECAI,2012]
  - quantified conditional logics [IJCAI,2013]
  - ambitious logic puzzles [AnnMathArtifIntell,2012]
  - ontology reasoning (e.g. in SUMO) [JWebSemantics,2012]
  - access control logics [SEC,2009]
  - ... more is on the way
- integrated with HETS, SigmaKEE, Isabelle

A: LEO-II – architecture and organization

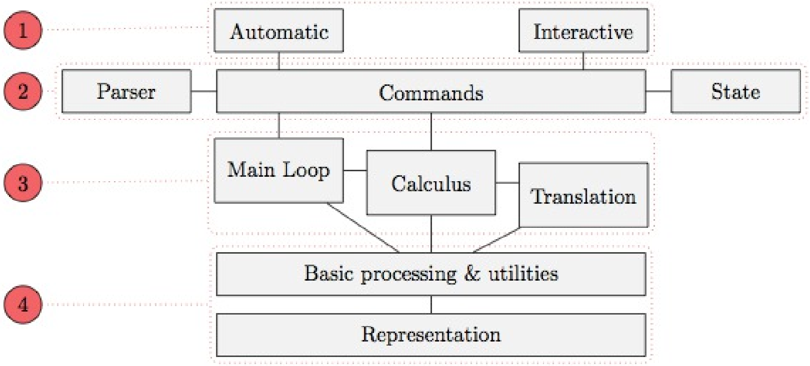


Figure 1: LEO-II's architecture

approx 30000 lines of Ocaml code

Talk Outline

Freie Universität Berlin

A: Introduction

- Motivation for LEO prover(s)
- Logic HOL / TPTP THF0
- Reasoning principles of LEO provers
- LEO-II

B: New Stuff in LEO-II

- Support for different FOL translations
- Integration of proofs from EP
- Improved support for back-end provers
- Detection/removal of Leibniz- and Andrews-equality
- Support for choice in LEO-II
- Further improvements
- Experiments

C: Conclusion

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]

$$\neg @_{(\iota \to o) \to \iota \to o}(b, x)$$

$$@_{(\iota \to o) \to \iota \to o}(b, x) \vee @_{(\iota \to o) \to \iota \to o}(c, x)$$

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]

$$\neg @_{(\iota\to o)\to\iota\to o}(b, x)$$

$$@_{(\iota\to o)\to\iota\to o}(b, x) \vee @_{(\iota\to o)\to\iota\to o}(c, x)$$

```
~(leoLit(leoTi(leoAt(leoTi(b,leoFt(i,o)),leoTi(x,i)),o)))

(leoLit(leoTi(leoAt(leoTi(c,leoFt(i,o)),leoTi(x,i)),o)) |
leoLit(leoTi(leoAt(leoTi(b,leoFt(i,o)),leoTi(x,i)),o)))
```

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]

shortcomings in the implementation; see e.g. example:

$$(=) = (=)$$

negation, input processing

```
~leoLit(leoTi(true,o))
```

but: LEO-II didn't provide axioms such as

```
leoLit(leoTi(true,o))
```

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]

Instead of

```
~leoLit(leoTi(true,o))
```

LEO-II now simply generates

```
~ $true
```

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]
- new (Nik Sultana): fof_full

When proxy terms are needed LEO-II adds axioms like

```
$true <=> leoLit(leoTi(true,o))
```

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]
- new (Nik Sultana): fof_full

In LEO-II's fully-typed translation lambda terms like $\lambda X_o. X$ were simply mapped to typed constants: `leoTi(abstrXX,leoFt(o,o))`

In the fof_full translation lambda-lifting is now employed.

B: Support for different FOL translations

FOL translations in LEO-II

- type-annotated @-operators [Kerber,PhD,1992]
- fully-typed [Hurd,CADE,2002]
- new (Nik Sultana): fof_full, fof_experiment

In the fof_experiment translation we are experimenting with lighter encodings of type information following [ClaessenEtal,CADE,2011].

Monotonicity analysis produces a SAT encoding; sent to MiniSat.

Interface for MiniSat has been adapted from Brown's Satallax.

B: Mapping of EP proofs

LEO-II supports different proof output modes

- no proof output (default, '`-po 0`' option)
- detailed proof by LEO-II / no EP proof ('`-po 1`' option)
- since v1.6.0 further options for LEO-II proof part available

```
% SZS status Theorem for SET171^3.p : (rf:0,axioms:0,...,translation:fully-typed)
%**** Beginning of derivation protocol ****
% SZS output start CNFRefutation
 ...
 thf(tp_intersection,type,(intersection: (($i>$o)>(($i>$o)>($i>$o))))).
 thf(tp_union,type,(union: (($i>$o)>(($i>$o)>($i>$o))))).
 ...
 thf(union,definition,(union = (^[X:($i>$o),Y:($i>$o),U:$i]: ((X@U) | (Y@U)))),
     file('SET171^3.p',union)).
 ...
 thf(1,conjecture,(![A:($i>$o),B:($i>$o),C:($i>$o)]:
     (((union@A)@((intersection@B)@C)) = ((intersection@((union@A)@B))@((union@A)@C)))),
     file('SET171^3.p',union_distributes_over_intersection)).
 ...
 thf(72,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[11,71,70,69,68,61,60,59,58,
     54,53,52,51,14])).
 thf(73,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[72])).
% SZS output end CNFRefutation
```

B: Mapping of EP proofs

LEO-II supports different proof output modes

- no proof output (default, '-po 0' option)
- detailed proof by LEO-II / no EP proof ('-po 1' option)
- since v1.6.0 further options for LEO-II proof part available

```
% SZS status Theorem for SET171^3.p : (rf:0,axioms:0,...,translation:fully-typed)
%**** Beginning of derivation protocol ****
% SZS output start CNFRefutation
 ...
 thf(tp_intersection,type,(intersection: (($i>$o)>(($i>$o)>($i>$o))))).
 thf(tp_union,type,(union: (($i>$o)>(($i>$o)>($i>$o))))).
 ...
 thf(union,definition,(union = (^[X:($i>$o),Y:($i>$o),U:$i]: ((X@U) | (Y@U)))),
     file('SET171^3.p',union)).
 ...
 thf(1,conjecture,(![A:($i>$o),B:($i>$o),C:($i>$o)]:
     (((union@A)@((intersection@B)@C)) = ((intersection@((union@A)@B))@((union@A)@C)))),
     file('SET171^3.p',union_distributes_over_intersection)).
 ...
 thf(72,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[11,71,70,69,68,61,60,59,58,
     54,53,52,51,14])).
 thf(73,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[72])).
% SZS output end CNFRefutation
```

B: Mapping of EP proofs

Since version 1.4.0; see also [SultanaBenzmüller,IWIL,2012]:

- mapping of EP proofs into LEO-II proofs ('-po 2' option)

```
% SZS status Theorem for SET171^3.p : (rf:0,axioms:0,...,translation:fully-typed)
%**** Beginning of derivation protocol ****
% SZS output start CNFRefutation
 ...
 thf(tp_intersection,type,(intersection: (($i>$o)>(($i>$o)>($i>$o))))).
 thf(tp_union,type,(union: (($i>$o)>(($i>$o)>($i>$o))))).
 ...
 thf(union,definition,(union = (^[X:($i>$o),Y:($i>$o),U:$i]: ((X@U) |
% (Y@U)))),
    file('SET171^3.p',union)).
 ...
 thf(1,conjecture,(![A:($i>$o),B:($i>$o),C:($i>$o)]:
    (((union@A)@((intersection@B)@C)) = ((intersection@((union@A)@B))@((union@A)@C)))),
    file('SET171^3.p',union_distributes_over_intersection)).
 ...
 fof(74, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[51])).
 fof(77, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[54])).
 fof(78, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[58])).
 fof(85, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[71])).
 ...
 cnf(128,plain,($false),inference(rw, [status(thm)],[114,115,theory(equality)])).
 cnf(129,plain,($false),inference(cn,[status(thm)],[128, theory(equality,[symmetry])])).
 thf(130,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[129])).
 thf(131,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[130])).
% SZS output end CNFRefutation
```

- very brittle for various reasons $\rightarrow$ PxTP Discussion?

B: Mapping of EP proofs

Since version 1.4.0; see also [SultanaBenzmüller,IWIL,2012]:

- mapping of EP proofs into LEO-II proofs ('-po 2' option)

```
% SZS status Theorem for SET171^3.p : (rf:0,axioms:0,...,translation:fully-typed)
%**** Beginning of derivation protocol ****
% SZS output start CNFRefutation
 ...
 thf(tp_intersection,type,(intersection: (($i>$o)>(($i>$o)>($i>$o))))).
 thf(tp_union,type,(union: (($i>$o)>(($i>$o)>($i>$o))))).
 ...
 thf(union,definition,(union = (^[X:($i>$o),Y:($i>$o),U:$i]: ((X@U) |
% (Y@U)))),
    file('SET171^3.p',union)).
 ...
 thf(1,conjecture,(![A:($i>$o),B:($i>$o),C:($i>$o)]:
    (((union@A)@((intersection@B)@C)) = ((intersection@((union@A)@B))@((union@A)@C)))),
    file('SET171^3.p',union_distributes_over_intersection)).
 ...
 fof(74, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[51])).
 fof(77, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[54])).
 fof(78, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[58])).
 fof(85, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[71])).
 ...
 cnf(128,plain,($false),inference(rw, [status(thm)],[114,115,theory(equality)])).
 cnf(129,plain,($false),inference(cn,[status(thm)],[128, theory(equality,[symmetry])])).
 thf(130,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[129])).
 thf(131,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[130])).
% SZS output end CNFRefutation
```

- very brittle for various reasons → PxTP Discussion?

B: Mapping of EP proofs

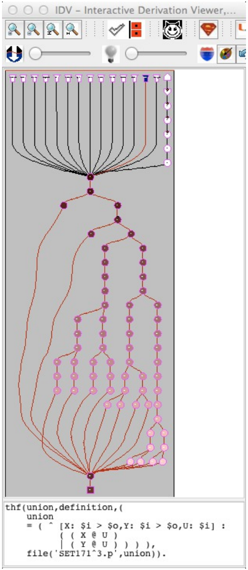
Since version 1.4.0; see also [SultanaBenzmüller,IWIL,2012]:

- mapping of EP proofs into LEO-II proofs ('-po 2' option)

```
% SZS status Theorem for SET171^3.p : (rf:0,axioms:0,...,translation:fully-typed)
%**** Beginning of derivation protocol ****
% SZS output start CNFRefutation
 ...
 thf(tp_intersection,type,(intersection: (($i>$o)>(($i>$o)>($i>$o))))).
 thf(tp_union,type,(union: (($i>$o)>(($i>$o)>($i>$o))))).
 ...
 thf(union,definition,(union = (^[X:($i>$o),Y:($i>$o),U:$i]: ((X@U) |
% (Y@U)))),
     file('SET171^3.p',union)).
 ...
 thf(1,conjecture,(![A:($i>$o),B:($i>$o),C:($i>$o)]:
     (((union@A)@((intersection@B)@C)) = ((intersection@((union@A)@B))@((union@A)@C)))),
     file('SET171^3.p',union_distributes_over_intersection)).
 ...
 fof(74, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[51])).
 fof(77, axiom, ((leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[54])).
 fof(78, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[58])).
 fof(85, axiom, ((~(leoLit(leoTi(leoAt( ... , inference(fof_translation, [status(thm)],[71])).
 ...
 cnf(128,plain,($false),inference(rw, [status(thm)],[114,115,theory(equality)])).
 cnf(129,plain,($false),inference(cn,[status(thm)],[128, theory(equality,[symmetry])])).
 thf(130,plain,((($false)=$true)),inference(fo_atp_e,[status(thm)],[129])).
 thf(131,plain,($false),inference(solved_all_splits,[solved_all_splits(join,[])],[130])).
% SZS output end CNFRefutation
```

- very brittle for various reasons → PxTP Discussion?

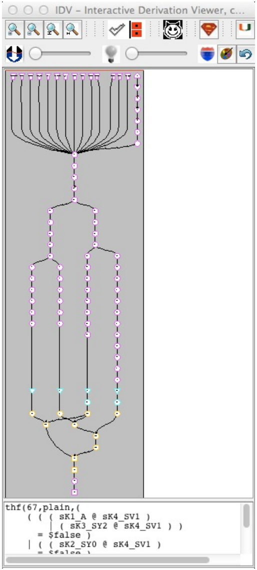
B: Mapping of EP proofs



'-po 1'

TSTP tools are applicable

IDV [TracPuzisSutcliffe,ENTCS,2007] visualization of (SET171^3.p)



'-po 2'

B: Improved support for back-end provers

Back-end provers in LEO-II

- first choice: EP
- new: better support for SPASS, Vampire and others
- new: support for remote provers on SystemOnTPTP
- ongoing: parallelization of EP, SPASS, Vampire
- ongoing: incremental Z3

Experiment — TPTP v5.4.0; LEO-II timout 60s; FO-ATP timeout 30s

- no. of problems exclusively proved
  LEO-II(E): 37 LEO-II(SPASS): 5 LEO-II(Vampire): 20
- no. of missed problems which one of the others could solve
  LEO-II(E): 31 LEO-II(SPASS): 95 LEO-II(Vampire): 98

B: Improved support for back-end provers

Back-end provers in LEO-II

- first choice: EP
- new: better support for SPASS, Vampire and others
- new: support for remote provers on SystemOnTPTP
- ongoing: parallelization of EP, SPASS, Vampire
- ongoing: incremental Z3

Experiment — TPTP v5.4.0; LEO-II timout 60s; FO-ATP timeout 30s

- no. of problems exclusively proved
  LEO-II(E): 37 LEO-II(SPASS): 5 LEO-II(Vampire): 20
- no. of missed problems which one of the others could solve
  LEO-II(E): 31 LEO-II(SPASS): 95 LEO-II(Vampire): 98

$$\lambda X_\alpha \lambda Y_\alpha \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y$$

$$\lambda X_\alpha \lambda Y_\alpha \forall Q_{\alpha\to\alpha\to o}.\ \forall Z_\alpha (Q\ Z\ Z) \Rightarrow Q\ X\ Y$$

They support cut-simulation due to their impredicative nature.

We added two new rules to the calculus

$$\frac{\mathbf{C}\ \vee\ [P\ \mathbf{A}]^{\mathrm{ff}}\ \vee\ [P\ \mathbf{B}]^{\mathrm{tt}}}{\mathbf{C}\{\lambda X.\ \mathbf{A} = X/P\} \vee [\mathbf{A} = \mathbf{B}]^{\mathrm{tt}}}\ \mathrm{LeibEQ} \qquad \frac{\mathbf{C}\ \vee\ [P\ \mathbf{A}\ \mathbf{A}]^{\mathrm{ff}}}{\mathbf{C}\{\lambda X \lambda Y.\ X = Y/P\}}\ \mathrm{AndrEQ}$$

These rules are obviously sound.

Some TPTP problems with rating 1.0 can now be solved:
SYO246^5.p, SYO244^5.p, NUM817^5.p, NUM816^5.p, NUM814^5.p.

Use of primitive substitution (blind guessing) can often be avoided.

B: Detection/removal of Leibniz- and Andrews-equality

$$\lambda X_\alpha \lambda Y_\alpha \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y$$

$$\lambda X_\alpha \lambda Y_\alpha \forall Q_{\alpha\to\alpha\to o}.\ \forall Z_\alpha (Q\ Z\ Z) \Rightarrow Q\ X\ Y$$

They support cut-simulation due to their impredicative nature.

We added two new rules to the calculus

$$\frac{\mathbf{C} \lor [P\ \mathbf{A}]^{\mathrm{ff}} \lor [P\ \mathbf{B}]^{\mathrm{tt}}}{\mathbf{C}\{\lambda X.\ \mathbf{A} = X/P\} \lor [\mathbf{A} = \mathbf{B}]^{\mathrm{tt}}}\ \text{LeibEQ} \qquad \frac{\mathbf{C} \lor [P\ \mathbf{A}\ \mathbf{A}]^{\mathrm{ff}}}{\mathbf{C}\{\lambda X \lambda Y.\ X = Y/P\}}\ \text{AndrEQ}$$

These rules are obviously sound.

Some TPTP problems with rating 1.0 can now be solved:
SYO246^5.p, SYO244^5.p, NUM817^5.p, NUM816^5.p, NUM814^5.p.

Use of primitive substitution (blind guessing) can often be avoided.

B: Detection/removal of Leibniz- and Andrews-equality

$$\lambda X_\alpha \lambda Y_\alpha \forall P_{\alpha\to o}.\ P\ X \Rightarrow P\ Y$$

$$\lambda X_\alpha \lambda Y_\alpha \forall Q_{\alpha\to\alpha\to o}.\ \forall Z_\alpha (Q\ Z\ Z) \Rightarrow Q\ X\ Y$$

They support cut-simulation due to their impredicative nature.

We added two new rules to the calculus

$$\frac{\mathbf{C}\ \vee\ [P\ \mathbf{A}]^{\mathrm{ff}}\ \vee\ [P\ \mathbf{B}]^{\mathrm{tt}}}{\mathbf{C}\{\lambda X.\ \mathbf{A} = X/P\} \vee [\mathbf{A} = \mathbf{B}]^{\mathrm{tt}}}\ \text{LeibEQ} \qquad \frac{\mathbf{C}\ \vee\ [P\ \mathbf{A}\ \mathbf{A}]^{\mathrm{ff}}}{\mathbf{C}\{\lambda X \lambda Y.\ X = Y/P\}}\ \text{AndrEQ}$$

These rules are obviously sound.

Some TPTP problems with rating 1.0 can now be solved:
SYO246^5.p, SYO244^5.p, NUM817^5.p, NUM816^5.p, NUM814^5.p.

Use of primitive substitution (blind guessing) can often be avoided.

B: Support for choice in LEO-II

$$\exists E_{(\alpha \to o) \to \alpha} \forall P_{(\alpha \to o)}.\ \exists X_{\alpha} (P\ X) \Rightarrow P\ (E\ P)$$

Partial support for choice before LEO-II 1.5 (naïve Skolemization).

B: Support for choice in LEO-II

$$\exists E_{(\alpha \to o) \to \alpha} \forall P_{(\alpha \to o)}.\, \exists X_\alpha (P\, X) \Rightarrow P\, (E\, P)$$

Partial support for choice before LEO-II 1.5 (naïve Skolemization).

Instances of AC axiom scheme could be added:

$$\exists E_{(\iota \to o) \to \iota} \forall P_{(\iota \to o)}.\, \exists X_\iota (P\, X) \Rightarrow P\, (E\, P)$$

However, such impredicative axioms support cut-simulation.

B: Support for choice in LEO-II

$$\exists E_{(\alpha\to o)\to\alpha} \forall P_{(\alpha\to o)}. \exists X_\alpha (P\ X) \Rightarrow P\ (E\ P)$$

We added two new rules (the set CFs maintains choice functions and is initialized with one choice function for each type).

$$\frac{[PX]^{\mathrm{ff}} \vee [P(f_{(\alpha\to o)\to\alpha} P)]^{\mathrm{tt}}}{\mathrm{CFs} \longleftarrow \mathrm{CFs} \cup \{f_{(\alpha\to o)\to\alpha}\}}\ \mathrm{detectChoiceFn}$$

$$\frac{C := \mathbf{C}' \vee [\mathbf{A}[E_{(\alpha\to o)\to\alpha}\mathbf{B}]]^p \qquad \begin{array}{c}\epsilon \in \mathrm{CFs},\ E = \epsilon \textit{ or } E \in \textit{freeVars}(C),\\ \textit{freeVars}(\mathbf{B}) \subseteq \textit{freeVars}(C), Y \textit{ fresh}\end{array}}{[\mathbf{B}\ Y]^{\mathrm{ff}} \vee [\mathbf{B}\ (\epsilon_{(\alpha\to o)\to\alpha}\mathbf{B})]^{\mathrm{tt}}}\ \mathrm{choice}$$

Rule choice is related to [Mints,JSL,1999].
Both rules are obviously sound.

B: Further improvements

- detection of satisfiable resp. countersatisfiable problems
  (supporting choice was essential for achieving this)
- improved support for flexible strategy scheduling
  (but: we still do not have good schedules!)
- reimplementation of depth-bounded extensional pre-unification
  (extensionality can now be disabled)
- parser, status reporting, avoiding redundant computations, factorisation, subsumption, clause selection, ...

B: Experiments

SZS Status	`fully-typed`	`fof_full`	`fof_experiment`
Thm	64.8	64.9	65.3
All	60.9	61	61.3

Table : Comparing FOL encodings in LEO-II version 1.5 (30s timeout). Table shows the percentage of matches between LEO-II's SZS output and the 'Status' field of problems.

Timeout (s)	v1.2		v1.4.3		v1.5	
	Thm	*All*	*Thm*	*All*	*Thm*	*All*
30	58.4	51.1	62.1	54.4	64.3	61.3
60	58.7	51.3	65	56.9	67.1	62.9

Table : Percentage match between different versions of LEO-II and the Status field of TPTP problems. LEO-II v1.2 was the winner of the CASC competition in 2010, and v1.4.3 was the last public release. Version 1.5 was run with the `fof_experiment` encoding.

B: Experiments

SZS Status	`fully-typed`	`fof_full`	`fof_experiment`
Thm	64.8	64.9	65.3
All	60.9	61	61.3

Table : Comparing FOL encodings in LEO-II version 1.5 (30s timeout). Table shows the percentage of matches between LEO-II's SZS output and the 'Status' field of problems.

Timeout (s)	**v1.2**		**v1.4.3**		**v1.5**	
	Thm	*All*	*Thm*	*All*	*Thm*	*All*
30	58.4	51.1	62.1	54.4	64.3	61.3
60	58.7	51.3	65	56.9	67.1	62.9

Table : Percentage match between different versions of LEO-II and the Status field of TPTP problems. LEO-II v1.2 was the winner of the CASC competition in 2010, and v1.4.3 was the last public release. Version 1.5 was run with the `fof_experiment` encoding.

LEO-II

- strongly collaborates with FO-ATPs
- proof exchange/verification is thus an important issue
- version 1.5 of LEO-II has several new, interesting features, some performance gain on TPTP (but not overwhelming yet)

Btw, did you know that LEO-II

- paralleled and strongly influenced the development of THF0 (EU project with Geoff Sutcliffe)
- has been the first prover to accept THF0, FOF and CNF
- is the **only** THF0 prover that has been running at CASC in proof producing mode!

Conclusion: Ongoing and future work

- parallelization of E, Vampire, SPASS
- exploitation of incremental provers (Z3)
- exploitations of term orderings (towards superposition for HOL)
- exploitation of term sharing information
- improvements for choice
- induction
- scheduling / parameter selection
- premise selection