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

ⓒ C. Benzmüller 2007

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
  - $== (\lambda X.\lambda Y.Y = X)$

© C. Benzmüller 2007

Literals, Uni-Constraints, Clauses

LEO-II An Effective Higher-Order Theorem Prover

- Clauses and Literals

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

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

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

  - 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*.

© C. Benzmüller 2007

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

- Definitions

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

- Theorem

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

- Example solutions:

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

© C. Benzmüller 2007

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

© C. Benzmüller 2007

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

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

- 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

ⓒ C. Benzmüller 2007

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

© C. Benzmüller 2007

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)

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

  - otherwise CNF normalization still quite naive

- Normalization of clause 3 leads to (the V^i are free variables)

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

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

© C. Benzmüller 2007

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.

- 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

LEO-II
An Effective Higher-Order Theorem Prover

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

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

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

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

- Extensional Pre-unification $$\frac{\mathcal{C}, [\mathbf{M}_o = \mathbf{N}_o]^{=\mathsf{F}}}{\mathcal{C}, [\textit{unfold}(\mathbf{M}_o \Leftrightarrow \mathbf{N}_o)]^{=\mathsf{F}}}\ \text{bool}$$
  - 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.)

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

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

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

© C. Benzmüller 2007

Paramodulation and Rewriting (ToDo)

LEO-II
An Effective Higher-Order Theorem Prover

- 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. $\sigma(\mathcal{D}[\alpha]_{\mathsf{pl}}, \mathcal{C}) \not> \mathcal{D}[\mathbf{B}]_{\mathsf{pl}}$

© C. Benzmüller 2007

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!)>
```

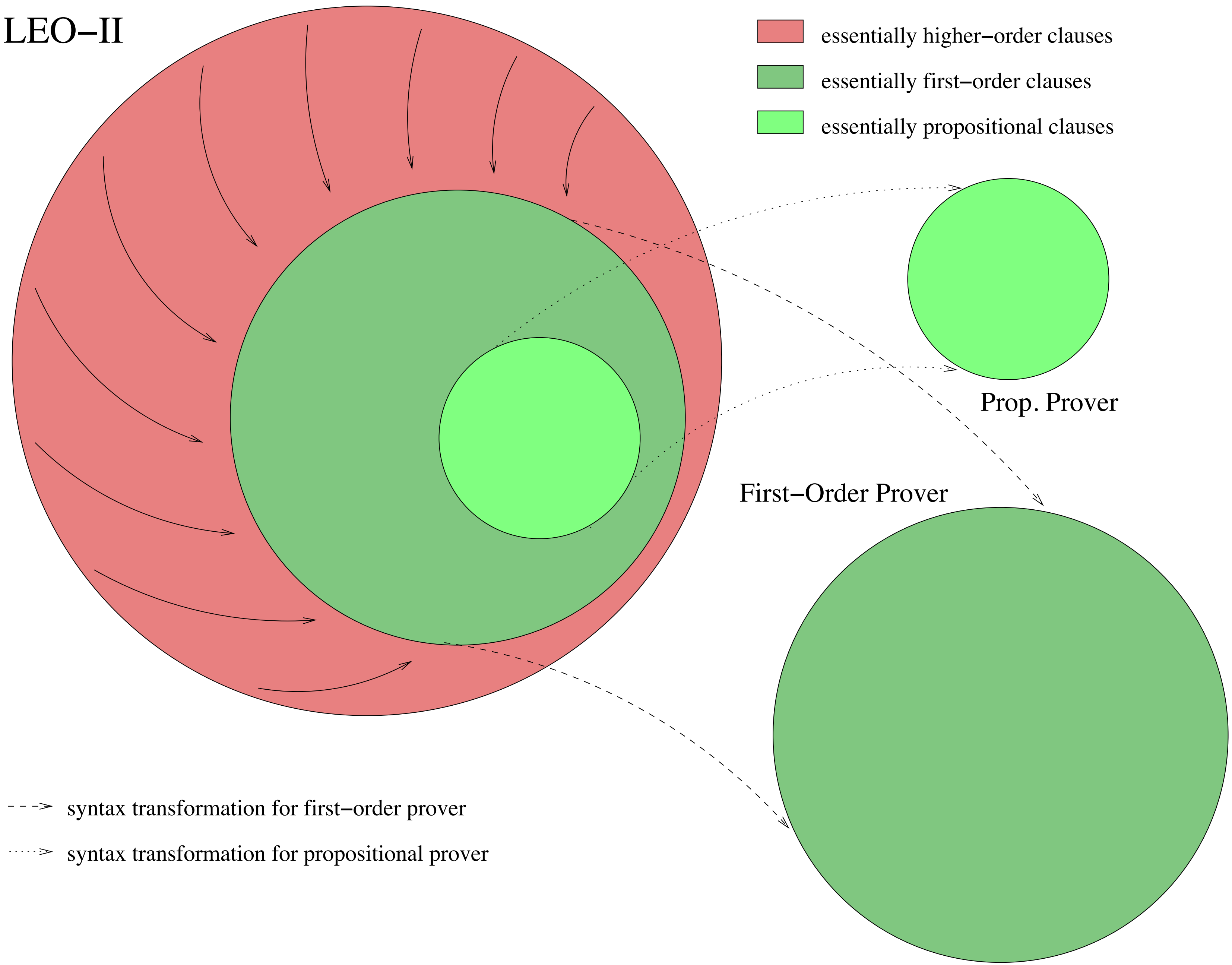
Proof Automation: ToDo List

- 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
- ...

ⓒC. Benzmüller 2007

Cooperation with FO-ATPs

Cooperation with Other Provers



©C. Benzmüller 2007

Cooperation with Other Provers

- 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

LEO-II
An Effective Higher-Order Theorem Prover

- 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

ⓒC. Benzmüller 2007

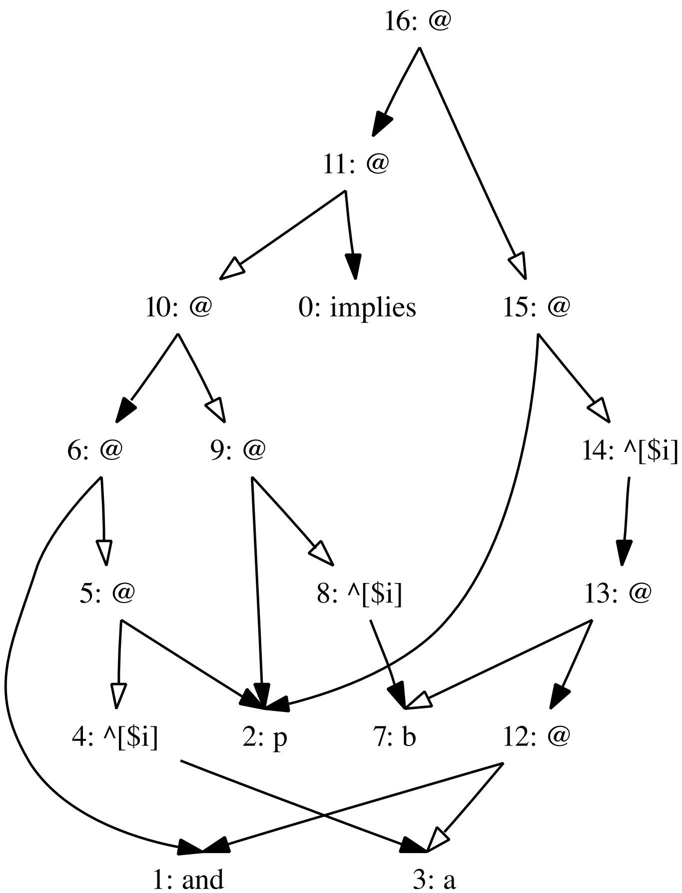
Term Sharing and Term Indexing

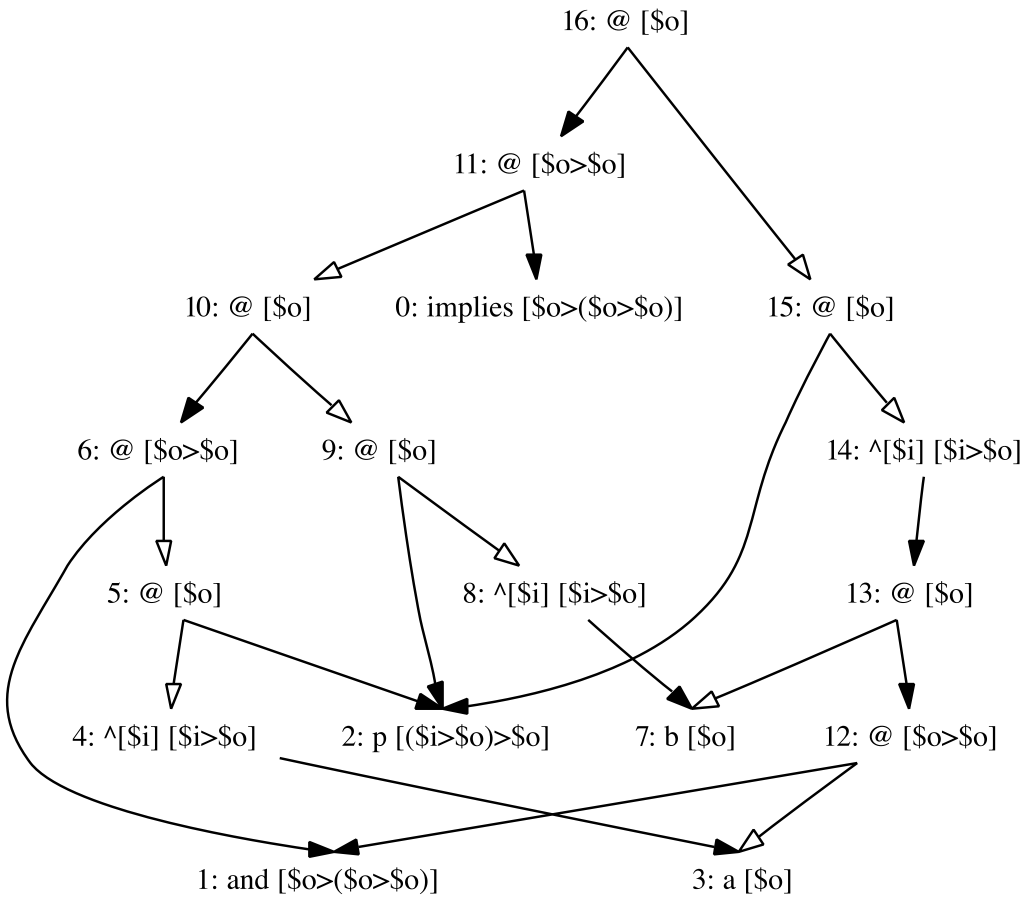
- Term sharing and term indexing widely employed in FO ATPs
- Not much used in HO systems so far
- LEO-II
  - Perfectly shared term data structure
  - DeBruijn-notation for bound variables
  - Indexing of various structural properties
  - Index realized via hashtables
  - Many operations (not all yet!)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

Perfect Term Sharing

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

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





ⓒ C. Benzmüller 2007

```
[...]
--- 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 node
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

- 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

LEO-II
An Effective Higher-Order Theorem Prover

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

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

© C. Benzmüller 2007

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

TPTP-Problem	Difficulty	Saturate	Muscadet	E-Setheo	Vampire 7	LEO Strat.	LEO Cl.	LEO Time	LEO-Bliksem Cl.	LEO-Bliksem Time	LEO-Bliksem FOcl	LEO-Bliksem FOtm	LEO-Bliksem GnCl	LEO-Vampire Cl.	LEO-Vampire Time	LEO-Vampire FOcl	LEO-Vampire FOtm	LEO-Vampire 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 (√) = 12.963

©C. Benzmüller 2007

New Experiments with LEO-II

LEO-II
An Effective Higher-Order Theorem Prover

	Fully Typed FO-Translation			$@_\alpha$-FO-Translation		
Filename	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
...	...	...	...	...	...	...

© C. Benzmüller 2007

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

© C. Benzmüller 2007

SET770+4

LEO-II
An Effective Higher-Order Theorem Prover

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

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

© C. Benzmüller 2007

Further Work

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
An Effective Higher-Order Theorem Prover

- 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

© C. Benzmüller 2007

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

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

- 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

© C. Benzmüller 2007