LEO-II

C. Benzmüller, L. Paulson, F. Theiss, A. Fietzke

Sydney, Australia, August 10, 2008

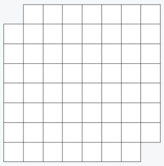
Overview

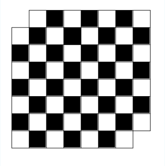
- Motivation
- How LEO-II solves the examples
- Term sharing and term indexing

Project Hypothesis

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

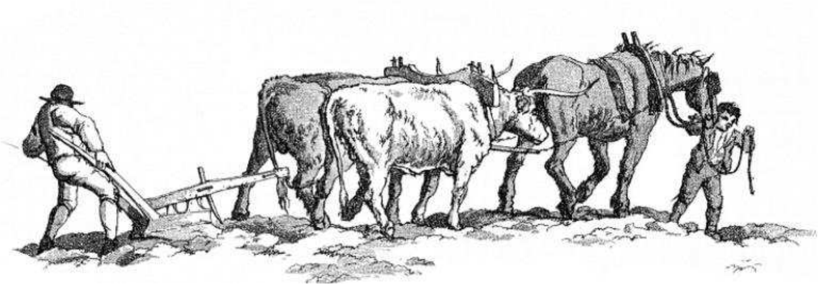
A Cooperative Prover

LEO-II

An Effective Higher-Order Theorem Prover

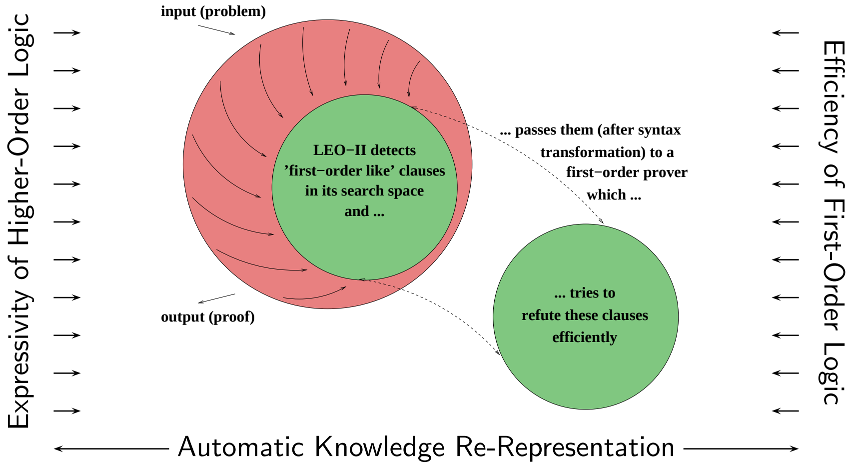
UNIVERSITY OF CAMBRIDGE

UNIVERSITÄT DES SAARLANDES



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

Architecture of LEO-II



How LEO-II solves the examples

How LEO-II solves the examples

▸ Example 1a – Sets:	0.136 sec
Example 1b – Sets (def. instead of =):	0.100 sec
▸ Example 2a – Knights and Knaves:	15.077 sec
Example 2b – Knights and Knaves (variant lucas):	0.064 sec
Example 2c – Knights and Knaves (variant chris):	40.447 sec
▸ Example 3 – Cantor:	2.856 sec

Example 1b:

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

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 $\cup$)

$$(\lambda x_\alpha. Bx \vee (Cx \wedge Dx)) \neq (\lambda x_\alpha. (Bx \vee Cx) \wedge (Bx \vee Dx))$$

LEO-II: Functional and Boolean Extensionality

$$\exists x_\alpha. (Bx \vee (Cx \wedge Dx)) \neq ((Bx \vee Cx) \wedge (Bx \vee Dx))$$

$$\exists x_\alpha. (Bx \vee (Cx \wedge Dx)) \not\Leftrightarrow ((Bx \vee Cx) \wedge (Bx \vee Dx))$$

LEO-II: Skolemization (x new Skolem constant)

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

Example 1b (contd.)

$$(Bx \lor (Cx \land Dx)) \not\Leftrightarrow ((Bx \lor Cx) \land (Bx \lor Dx))$$

LEO-II: Normalization

$$\neg Bx \qquad Bx \lor Cx \qquad Bx \lor Dx \qquad \neg Cx \lor \neg Dx$$

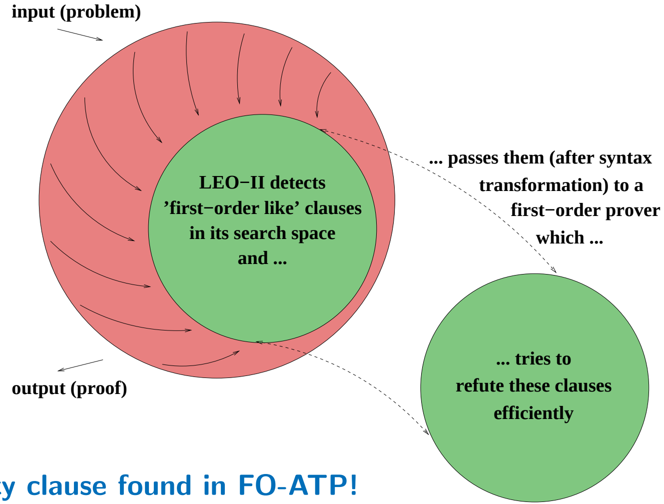
LEO-II: passes clauses to FO-ATP (modulo syntax transformation)

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

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

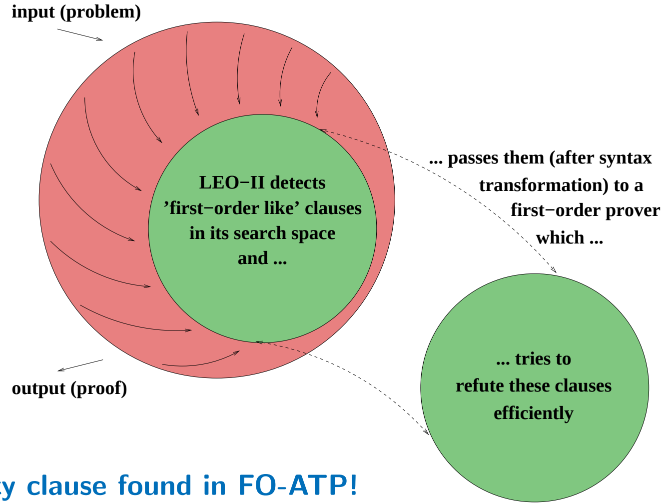
$$\neg @_{(\iota\to o)\to\iota\to o}(C, x) \lor \neg @_{(\iota\to o)\to\iota\to o}(D, x)$$

Example 1a-b



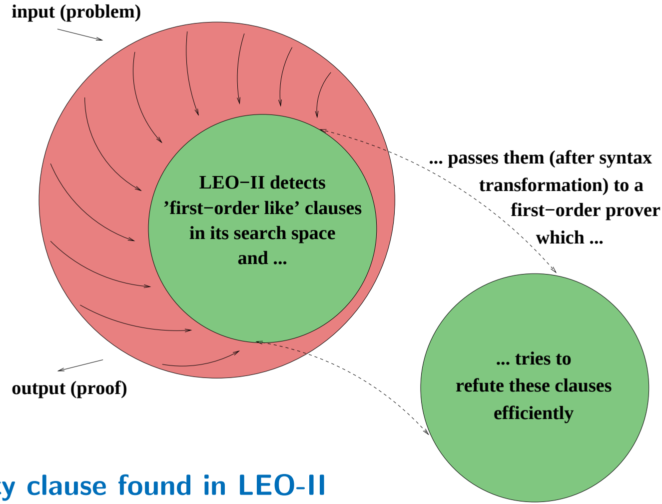
Empty clause found in FO-ATP!

Example 2a-c



Empty clause found in FO-ATP!

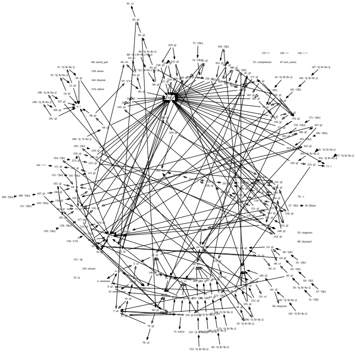
Example 3



Empty clause found in LEO-II

Term sharing and term indexing

Termsharing



In Leo II:

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

Adaption to HOL:

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

Conclusion

We need to foster higher-order ATP

- evidence that higher-order ATP strong in certain domains
- applications in S/H Verification and AI employ higher order
- interactive proof in proof assistants is costly

Try LEO-II

- Website: http://www.ags.uni-sb.de/~leo
- System description [IJCAR-08]
- TPTP THF input syntax [IJCAR-THF-08]
- Multimodal Logic [Festschrift-Andrews-08]

Example 1 – SET171

```
%------------------------------------------------------------------------
%----Signatures for basic set theory predicates and functions.
thf(const_in,type,(
    in: $i > ( $i > $o ) > $o  )).

thf(const_intersection,type,(
    intersection: ( $i > $o ) > ( $i > $o ) > ( $i > $o ) )).

thf(const_union,type,(
    union: ( $i > $o ) > ( $i > $o ) > ( $i > $o ) )).
```

Example 1 – SET171

```
%----Some axioms for basic set theory. These axioms define the set
%----operators as lambda-terms. The general idea is that sets are
%----represented by their characteristic functions.
thf(ax_in,axiom,(
    ( in
    = ( ^ [X: $i,S: ( $i > $o )] :
          ( S @ X ) ) ) )).

thf(ax_intersection,axiom,(
    ( intersection
    = ( ^ [S1: ( $i > $o ),S2: ( $i > $o ),U: $i] :
          ( ( in @ U @ S1 )
          & ( in @ U @ S2 ) ) ) ) )).

thf(ax_union,axiom,(
    ( union
    = ( ^ [S1: ( $i > $o ),S2: ( $i > $o ),U: $i] :
          ( ( in @ U @ S1 )
          | ( in @ U @ S2 ) ) ) ) )).

%----The distributivity of union over intersection.
thf(thm_distr,conjecture,(
    ! [A: ( $i > $o ),B: ( $i > $o ),C: ( $i > $o )] :
      ( ( union @ A @ ( intersection @ B @ C ) )
      = ( intersection @ ( union @ A @ B ) @ ( union @ A @ C ) ) ) )).
%------------------------------------------------------------------------
```

Example 2a – Knights and Knaves (chris original)

```
%------------------------------------------------------------------
%----Type declarations
thf(islander,type,(
    islander: $i )).

thf(knight,type,(
    knight: $i )).

thf(knave,type,(
    knave: $i )).

thf(says,type,(
    says: $i > $o > $o )).

thf(zoey,type,(
    zoey: $i )).

thf(mel,type,(
    mel: $i )).

thf(is_a,type,(
    is_a: $i > $i > $o )).
```

Example 2a – Knights and Knaves (chris original)

```
%----A very special island is inhabited only by knights and knaves.
thf(kk_6_1,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ islander )
     => ( ( is_a @ X @ knight )
        | ( is_a @ X @ knave ) ) ) )).

%----Knights always tell the truth.
thf(kk_6_2,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knight )
     => ( ! [A: $o] :
            ( says @ X @ A )
       => A ) ) )).

%-----Knaves always lie.
thf(kk_6_3,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knave )
     => ( ! [A: $o] : ( says @ X @ A )
       => ~ A ) ) )).
```

Example 2a – Knights and Knaves (chris original)

```
%----You meet two inhabitants: Zoey and Mel.
thf(kk_6_4,axiom,
    ( ( is_a @ zoey @ islander )
    & ( is_a @ mel @ islander ) )).

%----Zoey tells you that Mel is a knave.
thf(kk_6_5,axiom,
    ( says @ zoey @ ( is_a @ mel @ knave ) )).

%----Mel says, 'Neither Zoey nor I are knaves.'
thf(kk_6_6,axiom,
    ( says @ mel
    @ ~ ( ( is_a @ zoey @ knave )
        | ( is_a @ mel @ knave ) ) )).

%----Can you determine who is a knight and who is a knave?
thf(query,theorem,(
    ? [Y: $i,Z: $i] :
      ( ( Y = knight <~> Y = knave )
      & ( Z = knight <~> Z = knave )
      & ( is_a @ mel @ Y )
      & ( is_a @ zoey @ Z ) ) )).
%--------------------------------------------------------------------
```

Example 2b – Knights and Knaves (variant lucas)

```
%-------------------------------------------------------------------
%----A very special island is inhabited only by knights and knaves.
thf(kk_6_1,axiom,(
   ! [X: $i] :
    ( ( is_a @ X @ knight )
       <~> ( is_a @ X @ knave ) ) )).

%----Knights always tell the truth.
thf(kk_6_2,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knight )
     => ( ! [A: $o] :
            ( says @ X @ A )
       => A ) ) )).

%-----Knaves always lie.
thf(kk_6_3,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knave )
     => ( ! [A: $o] : ( says @ X @ A )
       => ~ A ) ) )).
```

Example 2b – Knights and Knaves (variant lucas)

```
%----You meet two inhabitants: Zoey and Mel.
% thf(kk_6_4,axiom,
%     ( ( is_a @ zoey @ islander )
%     & ( is_a @ mel @ islander ) )).

%----Zoey tells you that Mel is a knave.
thf(kk_6_5,axiom,
    ( says @ zoey @ ( is_a @ mel @ knave ) )).

%----Mel says, 'Neither Zoey nor I are knaves.'
thf(kk_6_6,axiom,
    ( says @ mel
    @ ~ ( ( is_a @ zoey @ knave )
        | ( is_a @ mel @ knave ) ) )).

%----Can you determine who is a knight and who is a knave?
thf(query,theorem,(
    ? [Y: $i,Z: $i] :
      ( ( is_a @ mel @ Y )
      & ( is_a @ zoey @ Z ) ) )).
%------------------------------------------------------------------
```

Example 2c – Knights and Knaves (variant chris)

```
%------------------------------------------------------------------
%----A very special island is inhabited only by knights and knaves.
thf(kk_6_1,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ islander )
     => ( ( is_a @ X @ knight )
        | ( is_a @ X @ knave ) ) ) )).

%----Knights always tell the truth.
thf(kk_6_2,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knight )
     => ( ! [A: $o] :
            ( says @ X @ A )
       => A ) ) )).

%-----Knaves always lie.
thf(kk_6_3,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knave )
     => ( ! [A: $o] : ( says @ X @ A )
       => ~ A ) ) )).
```

Example 2c – Knights and Knaves (variant chris)

```
%----You meet two inhabitants: Zoey and Mel.
thf(kk_6_4,axiom,
    ( ( is_a @ zoey @ islander )
    & ( is_a @ mel @ islander ) )).

%----Zoey tells you that Mel is a knave.
thf(kk_6_5,axiom,
    ( says @ zoey @ ( is_a @ mel @ knave ) )).

%----Mel says, 'Neither Zoey nor I are knaves.'
thf(kk_6_6,axiom,
    ( says @ mel
    @ ~ ( ( is_a @ zoey @ knave )
        | ( is_a @ mel @ knave ) ) )).

%----Can you determine who is a knight and who is a knave?
thf(query,theorem,(
    ? [Y: $i,Z: $i] :
      ( ( is_a @ Y @ knight )
      & ( is_a @ Z @ knave) ) )).
%------------------------------------------------------------------
```

Example 3 – Cantor's Theorem

```
thf(surjectiveCantorThm,conjecture,(
   ~ ( ? [G: $i > $i > $o] :
       ! [F: $i > $o] :
       ? [X: $i] :
         ( ( G @ X )
         = F ) ) )).
```