Evaluation of Systems for Higher-order Logic (ESHOL)

Christoph Benzmüller[1], Florian Rabe[2], Carsten Schürmann[3], Geoff Sutcliffe[4]

[1]Saarland University, Germany [2]Jacobs International University, Germany
[3]IT University of Copenhagen, Denmark [4]University of Miami, USA

1 Introduction

The ESHOL sessions of the PAAR workshop focussed on the use of higher-order reasoning systems. A particular focus was on means to evaluate higher-order reasoning systems. The notion of higher-order included, but was not limited to, ramified type theory, simple type theory, intuitionistic and constructive type theory, and logical frameworks. The notion of reasoning systems included automated and semi-automated provers, model generators, as well as proof and model checkers. There were two parts to the ESHOL sessions: (i) higher-order system demonstrations, and (ii) a panel discussion. Additionally, one of the PAAR invited speakers, Rob Arthan, gave a talk in the ESHOL topic area.

2 Higher-order System Demonstrations

The following higher-order systems were demonstrated in the system demonstration sessions. Each presenter gave a 10 minute "talk" slot to present the system to the audience in the traditional laptop+projector mode (giving a brief overview of the system and a demonstration of it running and solving some of the problems in Appendix A). Following the 10 minute presentations there was an open forum during which presenters were all available to give individual and more detailed information and demonstrations.

– Coq, *Guillaume Melquiond*
– Delphin, *Carsten Schürmann*
– HOL, *Joe Hurd*
– Isabelle, *Stefan Berghofer*
– IsaPlanner, *Lucas Dixon*
– LEO-II, *Christoph Benzmüler and Frank Theiss*
– Mizar, *Josef Urban*
– Omega, *Frank Theiss and Christoph Benzmüller*
– ProofPower, *Rob Arthan*
– TPS, *Mark Kaminski*

3 Panel Discussion

The ESHOL panelists were Rob Arthan, Lucas Dixon, and Joe Hurd. The panel discussed ideas, suggestions, and potential problems related to:

- The buildup of an higher-order TPTP infrastructure.
- The development of automated reasoning systems for higher-order logic (or fragments of it).
- Promising application areas for automated higher-order reasoning systems.
- The planned organization of a higher-order CASC at CADE-22 in 2009.

References

1. C. Benzmüller, F. Rabe, and G. Sutcliffe. THF0 - The Core TPTP Language for Classical Higher-Order Logic. In P. Baumgartner, A. Armando, and D. Gilles, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, Lecture Notes in Artificial Intelligence, page Accepted, 2008.

A Sample Problems for System Demonstrations

The two first problems should be simple enough for every system, to provide a starting point for comparisons and discussion. The third example is Cantor's Theorem, which might be more difficult. The problems are presented in the TPTP "THF" language for simple type theory, which was recently developed by the organizers [1]. The language is based on Church's simple type theory, and is a syntactically conservative extension of the untyped first-order TPTP language.

A.1 Puzzle Example

```
%------------------------------------------------------------------------
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 )).

thf(kk_6_1,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ islander )
     => ( ( is_a @ X @ knight )
        | ( is_a @ X @ knave ) ) ) )).

thf(kk_6_2,axiom,(
    ! [X: $i] :
      ( ( is_a @ X @ knight )
     => ! [A: $o] :
          ( ( says @ X @ A ) => A ) ) )).

thf(kk_6_3,axiom,
    ! [X: $i] :
      ( ( is_a @ X @ knave )
     => ! [A: $o] :
          ( ( says @ X @ A ) => ~ ( A ) ) )).

thf(kk_6_4,axiom,
    ( ( is_a @ zoey @ islander )
    & ( is_a @ mel @ islander ) )).

thf(kk_6_5,axiom,
    ( says @ zoey @ ( is_a @ mel @ knave ) )).

thf(kk_6_6,axiom,
    ( says @ mel
    @ ~ ( ( is_a @ zoey @ knave )
        | ( is_a @ mel @ knave ) ) )).

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

A.2 Set Theory Example

```
%------------------------------------------------------------------------
%----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 ) )).

%----Some axioms for basic set theory.
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 ) ) ) )).
%------------------------------------------------------------------------
```

A.3 Cantor's Theorem

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