CICM'20 Systems Description

Here we will put the consoldated list of authors

And here the consolidated list of institutes

Abstract. This paper gives an overview of new tools and improvements of existing tools in the CICM domain.

ELPI

Enrico Tassi, INRIA Sophia-Antipolis
Claudio Sacerdoti Coen, University of Bologna

Tool:	ELPI (v1.1)
Impl. in:	OCaml
License:	LGPL v2.1
Download:	
`https://github.com/LPCIC/elpi`	

Description ELPI (Embedded Lambda-Prolog Interpreter) is an implementation of an Higher Order Constraint Logic Programming Language that adds constraints and constraint handling rules to the λProlog interpreter. It is particularly well suited for the implementation of interactive theorem provers, type checkers, elaborators and translations between type systems or logics. It is implemented in the OCaml language in order to be easily integrated as an extension language into existing tools implemented in OCaml, like the Coq, Matita or HOL-light theorem provers. ELPI is almost completely backward compatible with Teyjus — the reference λProlog interpreter — but it adds new built-ins and an API to easily implement additional built-ins in OCaml, like stores that persist during backtracking.

Compared to the standard implementations of λProlog that try to optimize reduction, ELPI implements reduction naively, but it manages to avoid reduction at all in many frequently occurrent patterns [1].

Applications The ELPI tool has been used so far to implement a type-checker for the Grundlagen, an elaborator for the Calculus of Constructions and a prototype for a constructive version of HOL [2]; it has been used in the ProofCert project; finally it has been turned into a plug-in for Coq that allows to embed ELPI code in Coq scripts to implement tactics and to automatically generate definitions and proofs from declarations of inductive types.

The advantages of using ELPI to implement elaborators and interactive provers is that management of bindings, capture avoiding susbtitutions and α-conversion can be delegated to the meta-language (like in λProlog) and, in addition, existential variables that represent unknown terms/proofs can also be identified with the existential variables of the meta-level. The constraint programming extensions of ELPI are then used to impose typing constraints on the future instances of the meta-variables.

Changes from previous version ELPI v1.1 changes the semantics of constraint propagation rules to conform to the "revised operational semantics" of CHR, fixing a previous bug. A few API changes break backward compatibility with OCaml code that embeds ELPI. Finally the `halt` builtin is now able to report an error message to the calling program or to print it on the screen in case of interactive use.

References

1. Cvetan Dunchev, Ferruccio Guidi, Claudio Sacerdoti Coen, and Enrico Tassi. ELPI: fast, Embeddable, λProlog Interpreter. In *Proceedings of LPAR*, 2015.
2. Cvetan Dunchev, Claudio Sacerdoti Coen, and Enrico Tassi. Implementing HOL in an Higher Order Logic Programming Language. In *Proceedings of the Eleventh Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, LFMTP '16, pages 4:1–4:10. ACM, 2016.