Short Talk: Celf – A Logical Framework for Deductive and Concurrent Systems

Anders Schack-Nielsen and Carsten Schürmann

IT University of Copenhagen
Denmark

Abstract. CLF (Concurrent LF) [?] is a logical framework for specifying and implementing deductive and concurrent systems from areas, such as programming language theory, security protocol analysis, process algebras, and logics. Celf is an implementation of the CLF type theory that extends the LF type theory by linear types to support representation of state and a monad to support representation of concurrency.

The Celf system is a tool for experimenting with deductive and concurrent systems prevalent in programming language theory, security protocol analysis, process algebras, and logics. It supports the specification of object language syntax and semantics through a combination of deductive methods and resource-aware concurrent multiset transition systems. Furthermore it supports the experimentation with those specifications through concurrent logic programming based on multiset rewriting with constraints.

Many case studies have been conducted in Celf including all of the motivating examples that were described in the original CLF technical report [?]. In particular, Celf has been successfully employed for experimenting with concurrent ML, its type system, and a destination passing style operational semantics that includes besides the pure core a clean encoding of Haskell-style suspensions with memoizations, futures, mutable references, and concurrency omitting negative acknowledgments. Other examples include various encodings of the π-calculus, security protocols, petri-nets, etc.

CLF is a conservative extension over LF, which implies that Celf's functionality is compatible with that of Twelf [?]. With a few syntactic modifications Twelf signatures can be read, type checked, and queries can be executed. Celf does not yet provide any of the meta-theoretic capabilities that sets Twelf apart from its predecessor Elf, such as mode checking, termination checking, coverage checking, and the like, which we leave to future work. In this presentation we concentrate on the two main features of Celf.

Specification. CLF was designed with the objective in mind to simplify the specification of object languages by internalizing common concepts used for specification and avail them to the user. Celf supports dependent types for the encoding of judgments as types, e.g. typing relations between terms and types, operational relations between terms and values, open and closed terms, derivability, and logical truth. It also supports the method of *higher-order abstract* *syntax*, which relieves the user of having to specify substitutions and substitution application. In CLF, every term is equivalent to a unique inductively defined β-normal η-long form modulo α-renaming and let-floating providing an induction principle to reason about the adequacy of the encoding. In addition, CLF provides linear types and concurrency encapsulating monadic types in support of the specification of resource aware and concurrent systems. Examples include operational semantics for languages with effects, transition systems, and protocol stacks.

Experimentation. Celf provides a logic programming interpreter that implements a proof search algorithm for derivations in CLF type theory in analogy to how Elf implements a logical programming interpreter based on uniform proof search. Celf's interpreter is inspired (with few modifications) by Lollimon [**?**], an extension of Lolli, the linear sibling of λ-Prolog. The interpreter implements backward-chaining search within the intuitionistic and linear fragment of CLF and switches to forward-chaining multiset rewriting search upon entering the monad. Celf programs may jump in and out of the concurrency monad and can therefore take advantage of both modes of operation. In addition, the operational semantics of Celf is conservative over the operational semantics of Elf, which means that any Twelf query can also be executed in Celf leading to the same result.

Celf is written in Standard ML and compiles with SML/NJ, MLton and MLKit. The source code and a collection of examples are freely available from http://www.twelf.org/∼celf.

References

[CPWW02a] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Carnegie Mellon University. Department of Computer Science, 2002.

[CPWW02b] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework II: Examples and applications. Technical Report CMU-CS-02-102, Carnegie Mellon University. Department of Computer Science, 2002.

[LPPW05] Pablo López, Frank Pfenning, Jeff Polakow, and Kevin Watkins. Monadic concurrent linear logic programming. In Pedro Barahona and Amy P. Felty, editors, *Proceedings of the 7th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pages 35–46, Lisbon, Portugal, 2005.

[PS99] Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.