Designing Reliable, High-Performance Networks
with the Nuprl Logical Programming Environment

— Position Statement —

Christoph Kreitz
Department of Computer Science, Cornell University
Ithaca, NY 14853, U.S.A.

Formal methods tools have greatly influenced our ability to increase the reliability of software and hardware systems. Extended type checkers, model checkers and theorem provers have been used to detect subtle errors in prototype code and to clarify critical concepts in system design. Automated theorem proving now has the potential to support a formal development of reliable systems at the same pace as designs that are not formally assisted, provided it is engaged at the earliest stages of design and implementation.

An engagement of deductive methods at this stage depends on a formal language that is able to naturally express the ideas underlying the software systems, a knowledge base of formalized facts about systems concepts that a design team can use in its discussions, and a theorem prover capable of integrating a variety of different proof techniques while providing assurance for the correctness of the joint result.

The NUPRL Logical Programming Environment (Constable et al. 1986) is a framework for the development of formalized mathematical knowledge as well as for the synthesis, verification, and optimization of software. It provides an expressive formal logic (Martin-Löf 1984; Constable 1998) and a theorem proving environment (Constable et al. 1986; Allen et al. 2000; NuPRL ) that supports interactive and tactic-based proof development, extraction of programs from proofs, program evaluation, language extensions through a definition mechanism, and an extendable library of verified algorithmic knowledge.

Our goal is to demonstrate that the NUPRL LPE is capable of supporting the formal design and implementation of large-scale, high-performance network systems. We have already used the NUPRL LPE in the verification of protocols for the ENSEMBLE group communication toolkit (Kreitz, Hayden, & Hickey 1998; Hickey, Lynch, & van Renesse 1999), in verifiably correct optimizations of ENSEMBLE protocol stacks (Kreitz 1999; Liu et al. 1999), and in the formal design and implementation of new adaptive network protocols (Liu et al. 2001; Bickford et al. 2001a; 2001b). Currently we are working on providing formal support for the development of large distributed embedded systems.

Our experience shows that logical methods that have proven effective in program synthesis, verification, and optimization can be made to scale up to large software systems by employing several layers of formal abstraction and compositional reasoning techniques, by building large libraries of verified algorithmic knowledge, and by continuously expanding the logical foundations and automated proof capabilities of the reasoning environment.

References


Nuprl home page.
http://www.cs.cornell.edu/info/Projects/NuPrl.