CEC: A System for the Completion of Conditional Equational Specifications†

Hubert Bertling, Harald Ganzinger, Renate Schäfers

Fachbereich Informatik, Universität Dortmund
D-4600 Dortmund 50, W. Germany, uucp, bitnet: hg@unido

The CEC-system has been developed to support various operational aspects of software specifications by conditional equations. It is part of the efforts in the PROSPECTRA-project to support the development of programs from specifications. CEC is implemented in Prolog and runs under C-Prolog 1.5 and Quintus-Prolog 1.6 and 2.0. The present system is a successor of a system that has been presented at STACS '87, Passau, and at the Workshop on Conditional Term Rewriting, Orsay 1987. The old version was based on a concept of completion relative to constraints as described in [1] whereas the new version is based on ideas investigated in [2].

The Specification Language

A specification consists of a many-sorted signature and conditional equations. Operators may be specified as constructors, with the understanding that different constructor ground terms must not be identified in the equational theory. Linear notation of terms as in C-Prolog is possible. Additional parsing and pretty-printing functions can be provided by the user.

The Termination Proof System

CEC supports two concepts for proving the termination of rewrite systems, recursive path orderings in the version of Kapur, Narendran and Sivakumar [3] and polynomial interpretation with the techniques described by Ben Cherifa and Lescanne [4].

The Completion Procedure

The main feature of CEC is a completion procedure for conditional equations. The theory behind the procedure is described in [2]. It can handle nonreductive conditional equations by a kind of narrowing technique in which nonreductive conditions are superposed by reductive rewrite rules.

In order to make completion terminate on nontrivial examples, CEC has implemented two powerful techniques for simplifying critical pair peaks and superposition instances of nonreductive conditional equations. One technique we call rewriting in contexts, meaning that the equations of a condition are oriented into rules as far as possible, and these (skolemized) rules are used when reducing a conditional equation to normal form. The second technique uses the nonreductive equations for simplification of other equations by a forward chaining derivation. If the condition of a nonreductive equation is instance of a subset of the condition of an equation to be considered for simplification, the corresponding instance of the conclusion is added as a further condition to the equation to be simplified. This forward chaining is allowed as long as the complexity of the resulting equation does not exceed a bound that is defined during the creation of the equation to be simplified. Accompanied by a subsumption test and by rewriting in these enriched contexts, the procedure is therefore able to detect a quite large class of loops in the narrowing process.

The completion can be run automatically or through manual guidance of the user. For that purpose, CEC makes the basic completion inference rules visible at the user level. The system automatically keeps

† This work is partially supported by the ESPRIT-project PROSPECTRA, ref#390.

track of the fairness constraints so that arbitrary changing between automatic and manual mode is possible. The procedure can also be applied to theories with associative-commutative operators.

The Rewrite Rule Compiler

Reductive conditional rewrite rules are upon creation immediately compiled into Prolog-code. The Prolog-code performs the matching and rewriting for the rule. In the non-AC case, matching is mapped to the matching in Prolog. Otherwise, the techniques are similar to what has been described by Kaplan. In addition, we exploit the specific properties of constructors and partially evaluate the matching procedure in the AC-case.

Support of Modular Specifications

CEC allows for (injective) renaming of signatures and for forming the union of two specifications. If possible, termination proofs for the old system(s) are carried over to the resulting specification. Also, overlaps between any two axioms of one module will not be recomputed upon enrichment or combination. CEC incorporates some specific optimization techniques in order to perform acceptably on large specifications acc

Environment-related Facilities

CEC offers to save and restore the state of specifications internally, as well as to and from external files. Moreover, user decisions may be taken back via a general undo mechanism.

The Data Base

In order to facilitate changes to the basic data structures, CEC contains a frame-oriented data base kit. The structure of the primitive objects can be specified so that computation of dependent attributes and objects are invoked automatically upon any creation of an object. Dependencies between objects can be specified, allowing e.g. to automatically dispose dependent objects upon deletion of an object.

Some Benchmarks

The following figures are for Quintus-Prolog 2.0 on a Sun 3/260.

Completion of integers with s, p, and $<$ (8 equations generating 25 superpositions): 18 sec.

Completion of ordered lists with $\leq$, $\neq$, *has*, *insert*, *delete*, *isordered* (about 25 equations including some inductive properties, generating 21 superpositions): 35 sec.

Completion of binary representations of natural numbers with $<$, $\neq$, $+$, $-$, $*$: In this example only the "true"-cases of $<$ and $\neq$ have been specified such that the negative case have to be inferred "by failure" through the narrowing process. The final system contains more than 50 axioms, including 12 nonoperational (nonreductive) equations; 85 superpositions are being computed): 200 sec. Computing 123456789*123456789 in the generated system: 0.1 sec.

Acknowledgements

H. Baumeister, U. Waldmann, and U. Wertz have helped in implementing major parts of the system.

References

[1] Ganzinger, H.: Ground term confluence in parametric conditional equational specifications. Proc. STACS 1987, LNCS 247, 1987.

[2] Ganzinger, H.: A completion procedure for conditional equations. Report 234, U. Dortmund, 1987, to appear in Proc. 1st Int'l Workshop on Conditional Term Rewriting, LNCS, 1988.

[3] Kapur, D., Narendran, P., and Sivakumar, G.: A path ordering for proving termination of term rewrite systems. LNCS 186, 1985, 173-187.

[4] Ben Cherifa, A. and Lescanne, P.: An actual implementation of a procedure that mechanically proves termination of rewriting systems based on inequalities between polynomial interpretations. Proc. CADE-8, LNCS 230, 1986.