A new language for algebraic dynamic programming

Georg Sauthoff, Robert Giegerich

Faculty of Technology, Bielefeld University

Algebraic Dynamic Programming

Algebraic Dynamic Programming (ADP) is a declarative style of dynamic programming, which has emerged in the area of biosequence analysis. Based on the concepts of signatures, algebras, and regular tree grammars, a dynamic programming algorithm can be expressed at a convenient level of abstraction, obviating the development and the tedious debugging of the matrix recurrences typical of dynamic programming. The perfect separation of search space composition and evaluation of solution candidates, as enforced in ADP, leads to unprecedenced versatility in the combination of different analyses via product algebras. The ADP method has been used in implementing a good number of bioinformatics tools in the area of RNA structure prediction.

Historically, the implementation of ADP was prototyped in the lazy functional language Haskell, but efficiency as well as proliferation concerns require a stand-alone implementation.

Bellman's GAP and its compiler

Bellman's GAP is a new domain specific language for writing programs in the ADP paradigm. Bellman's GAP contains C/Java-like syntax elements and a notation for tree grammars resembling function calls. Compiling the declarative source code essentially means the derivation of and code generation for efficient dynamic programming recurrences. The current compiler implements non-trivial semantic analyses for yield size analysis and table design. Further examples are the automatic generation of different backtracing schemes or the generation of OpenMP-parallelized code.

Overview of the presentation

The talk will give a short introduction to the declarative concepts of algebraic dynamic programming. We will examplify the use of the new language with simple textbook style examples. We then report on some optimization by the compiler which lead to implementations competitive with handwritten code.