PLATO: A Tool to Assist Programming as Term Rewriting and Theorem Proving

Andre J. Sampaio, Armando M. Haeberer, Claudio T. Prates,
Cristina D. Ururahy, Marcelo F. Frias and Neudson C. Albuquerque

Pontifícia Universidade Católica do Rio de Janeiro, Department of Informatics,
Rua Marquês de São Vicente 225, CEP 22453-900, Rio de Janeiro, RJ, Brazil
e-mails: {andre,armando,terra,ururahy,mfrias,neudson}@inf.puc-rio.br

In this paper we present PLATO, a tool designed for theory presentation construction, theorem proving and formula derivation. It was motivated by equational algebraic theory manipulation needs, nevertheless its aim is software development by program calculation. PLATO has applications in algebra — boolean algebra, relation algebra, etc. — and mathematical logic — as a general framework for theorem proving under different logics —. In the field of software development, we show its application in program calculi construction and in program development under these calculi.

We will consider a programming calculus as being a theory presentation within a given logic. Thus, the task of constructing new calculi can be thought of as one of building new theory presentations. This is true for, in particular, programming calculi based on Relation Algebras, Functional Programming, Fork Algebras, Type Theory and the like.

We will consider a theory presentation as a pair $\langle \mathcal{T}, \mathcal{F} \rangle$, where $\mathcal{T}$ is a similarity type, i.e., declaration of operation symbols with their arity, and $\mathcal{F}$ is a *finite* set of formulas constituting an axiomatization for that theory.

Constructing Calculi as Theories¹. Since theories are presented as finite sets of axioms, to build from scratch a new theory presentation we must provide the axioms characterizing it. In order to do so the system has a pool of so-called “Platonic” formulas. These are abstract formulas whose semantics is undetermined until one instantiates their operation and relation symbols. Thus, platonic formulas are “incarnated” — the operation and relation symbols occurring in them are binded to the operation and relation symbols in the corresponding similarity type — to give raise to wffs of the theory under development. In order to construct theory presentations PLATO implements, among others, the following operations:

1. *Incarnation of Platonic Formulas:* Allows the addition of new wffs to presentations, for instance, new transformation rules to programming calculi.
2. *Sum of Presentations:* Joins two presentations. The theorems already proved about each presentation are inherited by the target presentation.
3. *Extension and Restriction of Presentations:* Allows addition and deletion of axioms from presentations. This facility is very useful, in mathematics, for con-

¹ The concept of theory manipulation for calculi construction was experimented in Carnaval, a prototype upon which Plato was developed.

structing bigger or smaller classes of models; while in programming, because axioms and theorems represent transformations or construction steps.

Theorem Proving and Software Development. Once a theory presentation is constructed, PLATO can be applied to the process of theorem proving and term manipulation. Keeping in mind our interest in developing software by means of programming calculi, theorem proving is necessary for establishing the correctness of guess steps, while term manipulation is required for performing correct steps. In order to be adequate for programming calculi, the process of theorem proving must have some particular characteristics. During the interactive proof of a theorem, an allowance is made for decisions which are not formally justified. It is useless to formally base a difficult proof step one is unsure will lead to a satisfactory proof of the theorem as a whole. Furthermore, a system that allows guessing, besides *logical proof steps*, also permits to perform *mathematical* proofs.

PLATO has a subsystem that deals with theorem proving and formula derivation. This subsystem allows the performance of guess steps, invoking automatic theorem provers or the interactive theorem prover SYFOM — SYmbolic FORmula Manipulator. SYFOM is a symbolic manipulation environment based on a rewriting system that allows derivations of formulas and theorem proving under well-defined theory presentations. Its development was motivated by the necessity of dealing, in a clean and organized way, with expressions, rules, axioms and theorems. SYFOM helps the user by suggesting only those axioms and previously proved theorems which are suitable for a determined derivation step.

SYFOM allows to “fold” sub-expressions and “unfold” names of previously folded expressions. It thus makes possible fold/unfold derivation steps in program construction, one of the most important strategies for program development by means of programming calculi. Furthermore, it allows one to choose subexpressions and to continue the derivation process starting from them.

Cases of Study. Currently, there are two cases of study implemented within PLATO. These are, a programming calculus based on Fork Algebras, and an interactive theorem prover for Relation Algebras. Once automatic theorem provers for Relation Algebras and Fork Algebras were implemented, some proof steps take place automatically.

Implementation Issues. PLATO is being developed in HyperCard, Mathematica and plain C. HyperCard, a popular hypertext system for the Apple Macintosh, is used as a man-machine interface generator. Mathematica is a powerful and comprehensive system for mathematical computation. It is used in SYFOM due to its powerful pattern matching mechanism and for its facilities for easy implementation of rewriting rules. Programs written for Mathematica are functionally compatible across a wide variety of hardware platforms. So, a PLATO user will deal with a friendly and customizable interface, which maintains transparency even when accessing SYFOM running on an unknown (and/or remote) hardware platform.