Extended ML: Past, present and future

Donald Sannella* Andrzej Tarlecki†

Abstract

An overview of past, present and future work on the Extended ML formal program development framework is given, with emphasis on two topics of current active research: the semantics of the Extended ML specification language, and tools to support formal program development.

1 Introduction

The ultimate goal of work on program specification is to establish a practical framework for the systematic production of correct programs from requirements specifications via a sequence of verified-correct development steps. Such a framework should be fully formal and based on sound mathematical foundations in order to guarantee the correctness of the resulting program with respect to the original specification. The program development activity must be supported by computer-based tools which remove the burden of clerical work from the user and eliminate the possibility of human error.

Extended ML is a framework for the formal development of programs in the Standard ML functional programming language from high-level specifications of their required input/output behaviour. It strongly supports “development in the large”, producing modular programs consisting of an interconnected collection of generic and modular units. The Extended ML framework includes a methodology for formal program development which establishes a number of ways of proceeding from a given specification of a programming task towards a program. Each such step (modular decomposition, etc.) gives rise to one or more proof obligations which must be discharged in order to establish the correctness of that step.

The Extended ML language is a wide-spectrum language which encompasses both specifications and executable programs in a single unified framework. It is a simple extension of the Standard ML programming language in which axioms are permitted in module interfaces and in place of code in module bodies. This allows all stages in the development of a program to be expressed in the Extended ML language, from the initial high-level specification to the final program itself and including intermediate stages in which specification and program are intermingled.

Formally developing a program in Extended ML means writing a high-level specification of a generic Standard ML module and then refining this specification top-down by means of a sequence (actually, a tree) of development steps until an executable Standard ML program is obtained. The development has a tree-like structure since one of the ways to proceed from a specification is to decompose it into a number of smaller specifications which can then be independently refined further. In programming terms, this corresponds to implementing a program module by decomposing it into a number of independent sub-modules. The end-product is an interconnected collection of generic Standard ML modules, each with a complete and accurate specification of its interface with the rest of the system. The explicit interfaces enable correct reuse of the individual modules in other systems, and facilitate maintainability by making it possible to localize the effect on the system of subsequent changes in the requirements specification.

*LFCS, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland.
†Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.
This paper is intended as a report on the status of work on Extended ML with emphasis on two topics of current active research: the semantics of the Extended ML language, and tools to support program development. In an attempt to make the paper self-contained, a brief introduction to formal program development in Extended ML is included in Section 2. Past work on the theoretical underpinnings of Extended ML is summarized in Section 3. Current work on the semantics of Extended ML is discussed in Section 4, and plans for tools to support formal program development are outlined in Section 5.

2 An introduction to Extended ML

The aim of this section is to briefly outline the main ideas of Extended ML. Three topics are discussed: the Standard ML functional programming language, which is the target of formal program development and on which the Extended ML wide-spectrum language is based; the Extended ML wide-spectrum language; and the Extended ML formal program development methodology. This outline is necessarily brief, and readers with no prior knowledge of Extended ML will probably find it helpful to consult the references given below.

2.1 Standard ML

Standard ML consists of two sub-languages: the Standard ML "core language" and the Standard ML "module language". The core language provides constructs for programming "in the small" by defining a collection of types and values of those types. The module language provides constructs for programming "in the large" by defining and combining a number of self-contained program units. These sub-languages can be viewed as more or less independent since there are relatively few points of contact between the sub-languages.

A complete formal semantics of Standard ML is in [MTH 90]; see [MT 90] for valuable explanatory prose. The features of the language are introduced at a more tutorial level in [Wik 87] (core language only), [MacQ 86a] and [Tof 89] (module language only), [Har 89], and [Rea 89].

2.1.1 The Standard ML core language

The Standard ML core language is a strongly typed functional programming language. It has a flexible type system including polymorphic types, disjoint union, product and (higher-order) function types, and user-defined abstract and concrete types. Programs written in the core language look very similar to programs in Hope [BMS 80], Miranda [BW 88] or Haskell [HW 89]. The following example of an Standard ML program uses most of the main features of the Standard ML core language:

```sml
datatype ('a,'b) alist = default of 'a -> 'b
| cons of 'a * 'b * ('a,'b) alist

type dictionary = (string,string) alist

exception novalue

val empty = default(fn a => raise novalue)

fun lookup(a,default f) = f a
| lookup(a,cons(al,b,l)) = if a=al then b else lookup(a,l)

fun isin(a,l) = (lookup(a,l); true) handle novalue => false

exception conflict
```

2.2 Extended ML

The Extended ML wide-spectrum language builds upon the Standard ML core language by adding constructs for program development "in the large". The Extended ML formal program development methodology is a set of guidelines for writing formal program specifications and proofs. This section provides a brief introduction to the Extended ML language and methodology, with more detailed discussion deferred to later sections.

2.2.1 The Extended ML wide-spectrum language

The Extended ML wide-spectrum language is a powerful and expressive language for program development. It includes constructs for defining and combining self-contained program units, for programming "in the large". The following example of an Extended ML program uses most of the main features of the Extended ML wide-spectrum language:

```sml

fun add(a,b) = cons(a,b,1)

fun mul(a,b) = cons(a,b,1)

fun pow(a,b) = cons(a,b,1)

fun isprime(a) = (add(a,1); true) handle novalue => false

exception conflict
```