The Data-Parallel Categorical Abstract Machine *

Gaétan Hains and Christian Foisy

Département d’Informatique et Recherche Opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7, email: {hains,foisy}@iro.umontreal.ca.

Abstract. Data-parallel ML is proposed for compilation to a distributed version (DPCAM) of Cousineau, Curien and Mauny’s Categorical Abstract Machine. The DPCAM is a static network of CAMs which dynamically restrict the MIMD execution mode: nodes execute the same program and communicate only while executing the same function body. Programs violating this restriction or exhibiting unbounded spatial recursion abort or deadlock. The execution model thus defines a class of SPMD programs.

With respect to syntax and types, DPML is Mini-ML enriched with localization and remote evaluation functions. Its values are arrays of size matching the DPCAM network and containing ML values. Pointwise functions, data-parallel primitives and systolic algorithms are easily programmed. To improve the language’s portability there is no automatic virtualization mechanism. Hence DPML is an intermediate target for more elaborate data-parallel languages, bridging the gap between direct source-language processor allocation and fully automatic allocation.

1 Introduction

This paper describes a data-parallel version (DPCAM) of the Categorical Abstract Machine of Cousineau, Curien and Mauny [9] with its programming language called data-parallel ML or DPML. The syntax of DPML is that of Mini-ML augmented with functions for localization: local, first, last and for remote evaluation: get. The Damas-Milner system is used for polymorphic type inference so that types convey no parallel information. Closed well-typed terms are compiled to DPCAM code which involves CAM instructions and operations for new communication structures. For the sake of portability, the DPCAM is a MIMD program with complete connectivity, but since programs are not required to use this flexibility, they may be optimized for an underlying topology.

Closed well-typed terms whose evaluation satisfies certain constraints are executed correctly, others deadlock or abort at run time. This constraint defines a class of SPMD programs by excluding full MIMD execution (unlimited asynchrony), process creation or nested parallelism. The abstract machine supports pointwise functions but also spatially recursive functions and pipelining for systolic algorithms.

* Work supported by FCAR and NSERC.
All DPML values are similar to POMPC's physical collections [16, 22]: global data structures whose size matches the abstract machine network. Virtualization mechanisms are thus excluded from the language's semantics, again for the sake of portability. A DPML term denotes both an array and its elements. This dual view microscopic/macroscopic was explained in [6]. Of course a useful data-parallel programming system should support virtualization and routing (e.g. CMLisp) so DPML is intended as intermediate target for more elaborate data-parallel languages, bridging the gap between direct processor allocation in the source language and fully automatic allocation. The DPCAM is portable and so are DPML programs which do not depend on the size of the network. However, programs which take advantage of a particular implementation by communicating along preferred channels may slowdown on other implementations.

We believe that complex compilers and interpreters can target DPML, and benefit from the safety of its type system and abstraction of ML programming. An advantage of data-parallel programming with ML should be the possibility of interactive program development, through an interpreter running on the parallel back-end computer.

2 Language Description

2.1 Syntax

The syntax (Table 1) is a simple extension of Mini-ML [8]. Readers unfamiliar with ML are referred to [18] for an introduction and to [21] for more details.

Specific to DPML are the localization constants: local, first, last of type integer. Their values are arrays of integers used to localize control. Assuming an \((n_2 - n_1 + 1)\)-processor implementation of the DPCAM where processors are numbered \(n_1\) to \(n_2\), local evaluates to the integer \(i \in \{n_1, \ldots, n_2\}\) on processor \(i\). The value of first is \(n_1\) and the value of last is \(n_2\). The boolean constants first? and last? are syntactic sugar for local = first and local = last respectively.

Also specific to DPML is the remote evaluation operator get. The expression get \((e', e)\) returns the value of \(e\) evaluated in the last communication context saved by processor \(i\) where \(i\) is the result of evaluating \(e'\) on the current processor. This notion will be made precise in Sec. 3.4. Together with the slogan values are arrays it determines the language's operational semantics.

For completeness DPML should have an imperative input procedure to load the network, but this has no effect on the rest of the language and will not be discussed here.

2.2 Types

The Damas-Milner type system [11] is used with minor extensions so types encode no parallelism-related information. In fact every object has an overloaded meaning microscopic/macroscopic as explained earlier. The type inference rules