Master internship offer: Eliminating clocks from X10 Polyhedral Programs

1 Internship supervisor and host laboratory

Supervisor : Éric Violard
E-mail : Eric.Violard@inria.fr
Phone : (+33) 03 68 85 02 43
`http://icube-icps.unistra.fr/index.php/Eric_Violard`

ICube Laboratory - Team ICPS (Parallel and Scientific Computing)
300 bd Sébastien Brant - BP 10413 - F-67412 Illkirch Cedex
`http://icube-icps.unistra.fr`

2 Context

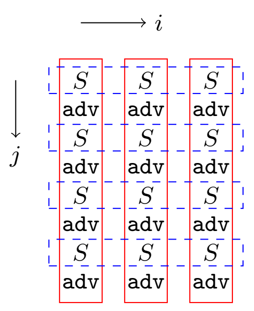
This work is related to parallel programming languages and program transformations and is set in a collaboration between some people (including Paul Feautrier, Alain Ketterlin and Eric Violard) which are members of two INRIA teams (CAMUS and COMPSYS). These two teams are interested in developing parallelizing and optimizing techniques, as well as proof and certification methods, for the efficient use of parallel architectures.

This project is particularly concerned with optimizing X10 programs. The X10 language [3] is a promising recent parallel language, developed by IBM Research and designed specifically to address the challenges of productively programming a wide variety of target platforms including complex hardware systems such as clusters of multi-core CPUs and accelerators. The sequential core of X10 is an object-oriented language in the Java family. This core is augmented by a few parallel constructs that create activities and synchronize them. Like many other parallel languages, these constructs are redundant and may be used interchangeably in some circumstances, allowing a programmer to choose amongst several program shapes or viewpoints. In particular, synchronization can be achieved by using clocks which can be seen as a generalization of the classical barriers. Synchronization on a clock is specified by the `advance()` method call. Activities that execute advances stall until all extent activities have done the same, and then are released at the same (logical) date.

We recently introduced a program transformation which is applicable to a large class of parallel programs (which are called "polyhedral programs"). This systematic transformation is able to switch between two extreme program shapes: – a parallel composition of sequential activities that synchronizes using clocks ; – a sequence of parallel unclocked activities. Obviously, these two extreme cases can be combined to produce many intermediate solutions. Such an ability opens up a large space of new potential optimizations, extending the scope of automatic parallelization.

Example 1. To understand the idea of this transformation, consider Figure 1: the center graph depicts the execution of an imaginary X10 program, where activities are represented by vertical red boxes that contain a sequence of normal instructions (denoted by `S`) and clock synchronization operations. These activities ”align” on their calls to `advance()`. The code on the left side of the figure is one possible source of this program. The idea of the transformation is to extract ”slices” (or phases) across activities, represented by horizontal boxes on the graph. A possible corresponding program appears on the right of the figure: the usage of clocks has been replaced by the barrier ending finish blocks. Both programs have exactly the same behavior, except for clocks and the number (and duration) of activities.

```
clocked finish {
  for(i in 0..2)
    clocked async
      for(j in 0..3) {
        S;
        advance();
      }
```



```
for(j in 0..3)
  finish
    for(i in 0..2)
      async S;
```

Fig. 1. Parallelism and synchronization in X10, with and without clocks

◇

This transformation is based on the polytope model [2] which is particularly used to model program control. Polyhedral operations are used to manipulate the program.

3 Objective

The objective of this project is to implement this source-to-source transformation for the X10 language and in the simplest case where all dates are affine index expressions. (A further goal is to include this transformation in a compiler within either static or dynamic optimizations). The student will have to :

– Study the polyhedral X10 programs class and the algorithm for transforming them

– Use some existing libraries for modeling programs and applying polyhedral operations (CLooG [1], ISL [4])
– Manipulate an AST (Abstract Syntax Tree) corresponding to an X10 program: extract and then gather the required syntactic elements
– Code the algorithm : compute the date symbolic expressions, build a new AST and produce a new X10 program written in concrete syntax.

References

1. C. Bastoul. Code generation in the polyhedral model is easier than you think. In *PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques*, pages 7–16, Juan-les-Pins, september 2004.
2. Paul Feautrier and Christian Lengauer. The polyhedral model. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.
3. Vijay Saraswat, Bard Bloom, Igor Peshansky, Olivier Tardieu, and David Grove. X10 language specification version 2.2, March 2012. `http://x10.sourceforge.net/documentation/languagespec/x10-latest.pdf`.
4. S. Verdoolaege. isl: An integer set library for the polyhedral model. *Mathematical Software–ICMS 2010*, pages 299–302, 2010.