TSPP

Methodology for producing parallel probabilistic algorithms from specifications

José Rolim * Erik Urland * Mathieu Buffo * Didier Buchs†

Abstract. The general aim of the project is to give a framework to a designer of parallel algorithms for defining abstractly a problem, finding a first parallel solution which may not be the most efficient one, and refining the solution by exploiting massive parallelism and probabilistic features.

1. STATE OF THE WORK

1.1 Introduction

From the past few years the effort in parallel software engineering is concentrating on the development of correct and efficient software packages for different parallel architectures. However, up today there is no common methodology, which can guarantee the correctness and efficiency of the designed algorithms. The classical approach to this problem is usually based either on the experience of programmer, either on some ad-hoc method used only for this particular case, especially concerning the implementation on the target parallel architecture. The problem becomes much more crucial when a large software product is required. From the theoretical point of view, some formal methods, based on predicate logic are now under development [1] [2]. These methods can give excellent results for some special family of objects, usually graphs which are in some sense recursively decomposable, but they are not very useful for general purposes. As an alternative way for the design and the development of parallel software, based on exact theoretical background, we developed a technique using transformation on specifications. This technique - used in the past with some difficulties [3] [4] - is arising from the area of rewriting systems but in our case suited into the context of parallelism; by transformation we understand an equivalence relation between two specifications. The process of handling (deriving) specifications in direction from abstract to more concrete ones is called transformation process and the emphasis is given to a transformation steps, called transformation methods, which introduces parallelism and leads closer to the implementation level. We claim that this process can be done automatically and to support our conjecture the TTool - transformation tool package - was developed. The TTool performs transformations on specifications under the bisimulation condition of these specifications. This guarantees the correctness of the resulting specification on the implementation level. In order to obtain an efficient implementation of the initial specification, the application of certain transformation steps is evaluated partially to the target parallel architecture and partially to other possible transformation methods; the most suitable transformation step is then chosen and performed. The last step in the process of deriving parallel program is to apply a suitable compiler for generating executable code for the target machine. TSPP is an open technique in the sense of further improvements and developments. New transformation methods can be simply added into the transformation corpus. Moreover, this technique is independent of the chosen specification language and can be considered as a general framework for deriving parallel or distributed solutions from formally specified problems. Nevertheless, in order to illustrate the practical capabilities of the TSPP technique, we have included the specification language CO-OPN (Concurrent Object - Oriented Petri Nets [5]) into the TTool. This language takes advantage of managing algebraic specifications easily as well as good possibilities of expressing parallelism. The object - oriented features naturally supports distributed description of specifications.

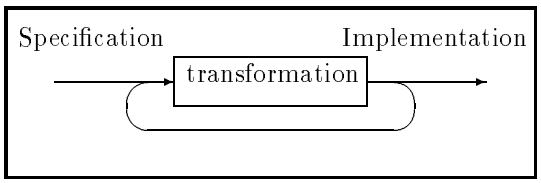
*Centre Universitaire d'Informatique, Université de Genève, 24 rue du général Dufour, 1211 Genève 4, Suisse, {rolim,urland.buffo}@cui.unige.ch

†LGL-DI, Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland, buchs@di.epfl.ch

1.2 Transformations on specifications

By "specification" we understand an exact formal description of a problem. Let S be a set of specifications and "$\approx$" be an equivalence relation defined on SxS.
Let $T = S$xS, and $T_C \subseteq T$, such that
$(x, y) \in T_C \Longleftrightarrow x \approx y$, where $x, y \in S$.
The set T is called "set of transformations", and an element $t \in T$ is called "transformation". The transformation $t \in T$ is said to be "correct" iff $t \in T_C$. Performing a "transformation step" passing from specification $x \in S$ to specification $y \in S$ is implied by the transformation $(x, y) \in T$.

Assume that we have an initial abstract description of a problem. Usually the formal description on this level does not contain any algorithmic construction. In order to obtain a parallel algorithm we have to perform a sequence of transformation steps belonging to the set T, refining the specification from an abstract level to a more concrete one, as it is shown on following figure:



Each transformation step consists of simple rewriting of a part of specification. In order to preserve the correctness of the resulting specification, the transformation steps are performed under the condition of some equivalence relation between specifications; in other words, transformations steps are chosen from the set T_C.

1.3 Selection of transformations

The most open question on the project is the set of transformations, which constitutes the library of the platform and which increases constantly. The goal is to build a collection as complete as possible of transformation methods and heuristic techniques for application. The process of selection of transformations has been started by defining two levels of transformations: the transformation rules and the transformation methods. The level of the rules is the first generic level of transformations. Simple and straightforward transformations can be defined at this level, using rewrite techniques. The level of the methods is based on abstract Petri nets, and is the second level of abstraction for describing transformations. Methods are more sophisticated than rules; axiomatic, algorithmic and methodological aspects are introduced in the specification using them. Unfortunately, up to now we has not yet been developed a theory about an automatic selection of the best rule or method to apply.

1.4 Specification language and transformation language

The language for expressing parallel computation chosen was based in CO-OPN; some refinement for our purposes, specially on the semantics, was necessary and we precise the notion of equivalence of objects using this specification language in the results described in [6] [7]. The main aspects of the work of the first year consisted on the study of transformations and the support of these transformations by a meta-language [8]. The strategy used for defining a transformation language was to mimic the CO-OPN language and then to have a language close to the language on which transformations are applied. Only a few syntactic constructions were added, such as the meta-variables representing part of the programs and the syntactic manipulation of terms. It must be noted that, the transformation rules and methods are based on pre and post conditions, the pre-conditions corresponding to the existence of properties of the axiomatic definition of the problems. The properties are consequence of the axioms and cannot be syntactically proved on a computationally decidable way; at this point human guidance is requested, either by directly asking the user to decide or through the use of a theorem prover. Unfortunately, the transformation process has not been connected yet to a theorem prover.

1.5 Parallelization and CO-OPN

One of the most important steps on the transformation process is the choice of transformations concerning the parallelization of the specification. Using CO-OPN, we split logically our rules and methods in two level: local and global transformations. The local transformations correspond to refining a specification and they occur at the level of algebraic operations and Petri net transitions. They can be split into three classes of transformations: the derivations, the parallelizations and the changements of structure. The derivation rules transform algebraic axioms into objects and aim a more concrete specification. The local parallelization rules transform objects into more suitable objects to the process of parallelization eventually increasing the performance. Still at the local

level we have the changements of structure transformations that concern the splitting and merging of objects leading to the adaptation of the execution platform. Finally, the so called global transformations work at the level of objects and groups of objects and concern a more algorithmic point of view; these high-level transformations use known techniques of design parallel algorithm. We can cite as examples of such techniques partition, branch-and-bound search, dynamic programming, divide-to-conquer, etc. [9]. More precisely, at the local level the transformations concern the data parallelism, the control parallelism and the pipelining, these aspects are expressed by corresponding transformations [8] and at a high level transformations concern to algorithmic techniques in a more global environment. The main idea, to get parallelism in the process of transformation, is to split the task into two, using initially the so called high level transformations, and then low level transformations. High level transformations work in a more abstract world, independently of details of implementation. They permit to obtain from a first abstract operation, which we would like to implement in a parallel way, a recursive operation. Low level transformations then implements such recursive functions, i.e. they allocate a tree structure of objects computing the operation, they parallelize the calculus of an object, doing operations such as dependence analysis. In fact, we can see high level transformation as algorithmic ones, and low-level as adaptation to the execution platform. As an example developed, suppose that from a very abstract specification of a sorting problem, we use first a high level definition of the divide to conquer technique to obtain a 2-recursive operation, in the case merge and sort, and then low level transformations such as tree allocation and simple recursive parallel objects creation allow us to obtain a parallel version of our sorting algorithm [9].

1.6 Probabilism and CO-OPN

Probabilistic computations can be naturally expressed using CO-OPN. Introducing probabilistic algorithms is performed by transforming deterministic specifications into non-deterministic ones and refining them. Therefore, real probabilistic transformations are part of high-level-transformations. However, low-level transformation can handle probabilism too, in the management of the parallel or distributed computation. In brief, probabilism is not a special case of our transformation system; it is just encompassed into the transformation corpus, like with parallelism.

1.7 The transformation tool package

The TTool - Transformation Tool package - was developed for performing transformations on specifications, following the TSPP technique. The TTool consists in the tool itself, which performs the transformation, and in the transformations library, which is used by the tool and which can be dynamically modified. This tool is a user-friendly xview-based application. The refinement is performed step by step, according to the transformation technique. At each step, the user can decide to stop the transformation process, to backtrack (if the developed specification is not satisfactory), to evaluate the adaptation of the current specification to the execution platform, or decide to perform a new transformation on the current specification. The transformation itself can be seen as a dialogue; the user select the correct transformation rule or a transformation method to apply, TTool displays then the possible transformation steps derived from the selected rule or method, the user choose one of these steps, and TTool apply it. The library of transformations is composed of several textual modules, written in a CO-OPN-like syntax. These modules are splited into two classes, according to the TSPP techniques. The first class is composed of the rules modules, which contains transformation rules and are similar to the algebraic modules in CO-OPN, because of the similarity between the rewriting rules and the algebraic operations.The second class is composed of the method modules, each of them containing a transformation method. They are similar to the object modules in CO-OPN, because of the fact that all are essentially Petri net descriptions.

1.8 The CO-OPN compiler

The parallel compiler is used to obtain real code from specifications. More precisely, it traduces CO-OPN specification into C code, which can then be executed on parallel machines. Although this compiler was designed for MIMD and SIMD machines, unfortunately up to now it is suited for MIMD machines only; more precisely the output code is targeted for transputer networks. Moreover, the compiler can also produce control code to analyze graphically the parallel performance of the programs. The monitor program is based on the Paragraph software [10].

2. VALORIZATION

The objective of this project is to manage the development of new parallel probabilistic algorithms that could use the up-to-date massively parallel computers to their best. In order to exploit this possibility, we provide a transformation tool to the design of parallel algorithms that allows the use of known techniques of algorithmic on the particular application suited. As cited before, we have implemented such a tool - TTool - and an associate parallel compiler. These tools are operational and can be accessed from a normal workstation.

Notwithstanding the publications, the international valorization of the project were mainly dealing with the collaboration with scientific partners and the participations to workshops.

2.1 Collaboration

We had strong collaborations with experts of several foreign centers, such as Lyon on the subject of deriving rules for transformations of optimization parallel algorithms and Paris on the subject of formal specification and study of equivalence notions. We should cite also our collaboration on the proposal of a ESPRIT project with several European partners. This project -SCOOP for short- treated of solving combinatorial problems in parallel and we were the work-package leader responsible for the probabilistic part of the project. The institutions and person involved on this project have been more detailed described on our request for prolongation of this project.

Moreover, we have had a one month visit of Dr. Ondrej Sykora, from the Slovak Academy of Sciences and participant of the project ESPRIT "Algorithms for Future Technologies" - ALTEC, who helped to expertise our project and produced some interesting co-joint results concerning the area-complexity on parallel models of complexity [11]. We also had the visit of Prof. Ludek Kucera, from Charles University, Prague and member of ALTEC, who helped us on the probabilistic aspects of the project.

As an indirect consequence of this project, we organized an international workshop sponsored by the IFIP, the EATCS and the FNRS on the subject of parallelism. Extensive research has been carried out concerning different ways for the automatic extraction of parallelism of several problems. Efficient solutions have been found to many problems but others still lack efficient and automatic methods for their solution and remain open to research in parallel algorithms. Among this latter category, we can often recognize irregularly structured problems that resist parallelization by both automatic and manual manipulation. In order to discuss such issues, the first international joint workshop and summer school on Parallel Algorithms for Irregularly Structured Problems was held at the University of Geneva from August 29th to September 2nd, 1994. The meeting was composed of a series of invited lectures on irregularly structured parallel algorithms, a session of research papers, and a session of hands-on laboratory experiments. This meeting consisted of presentations of the state of the art in research concerning irregularly structured parallel algorithms, and surveys of potential research areas. It provided a collaborative working and research environment, allotting time to research sessions between attendees.

2.2 Workshops

The interesting results of the project allowed us to take part to three workshops and one congress, where we received great attention from audience; the way we explored seems to be promising in the near future and many research groups in the world are starting to considerate similar techniques. Precisely, we attend to the following workshops and congress:

- IEEE/USP Workshop on High Performance Computing - WHPC 94,
  "Compiler and Tools for parallel processing"
  São Paulo, Brazil, March 28-30.
- World Transputer Congress - WTC 94
  Cernobbio, Italy, September 3-7 1994.
- SIPAR-Workshop
  Fribourg, Switzerland, October 14.
- IEEE International Workshop on Parallel Processing - IWPP 94
  Bangalore, India, December 26-31.

3. PUBLICATIONS

As a direct product of the project, three papers was accepted for publication:

- Buffo M., Buchs D. and Rolim J., "Developing parallel programs based on rewriting techniques", in the proceedings of IEEE/USP Workshop on High Performance Computing", J. Zuffo ed., São Paulo, Brazil, 1994.

Abstract:
Starting with a formal problem specification, we show how to progressively achieve a parallel distributed solution, using convenient transformations of specifications. In order to achieve a solution, we use rewriting techniques applied in the context of transformations.
This paper explains the methodology, points out its advantages and its requests. We first, introduce the specification language CO-OPN used to express both abstract specification and implementation. CO-OPN is a structured version of algebraic specification and Petri nets in which adequate notions of equivalence can be formalized to handle the concept of correct transformations. Then we introduce the general technique of applying transformations to get specifications progressively closer to the implementation level. These transformations are classified into two levels, the high level transformation performing abstract modification of algorithm and the low level transformations which are applicable on the specification closer to the implementation.
Finally, a concrete example will show how the simple "permutation sort" can be transformed quite automatically in a distributed implementation of a "quick sort" algorithm.

- Buchs D., Buffo M., Flumet J., Racloz P. and Urland E., "SANDS Tools: an environment for deriving parallel programs", in the proceedings of WTC 94 Workshop on Case tools for parallel systems development, 1994.
Abstract:
This paper describes the techniques and the tools developed to construct CO-OPN specifications (Concurrent Object Oriented Petri Nets) and to derive parallel programs. CO-OPN is a specification language allowing to describe concurrent aspects and data-structure of computer programs in an abstract way. The concurrent part of the formalism is described with Petri nets while the data-structure are described with algebraic abstract data types. In the CO-OPN formalism, this association is structured by the notion of object. The environment developed around CO-OPN is SANDS (Structured Algebraic Nets Development System). The production of quality software needs the use of a complete development method based on a life cycle including specification and implementation phases. This method must provide techniques in order to derive implementation from specification and for verifying if the implementation is adequate. The development of software must be supported by tools such as editors, validation and verification processes. A tool for obtaining correct parallel programs from formally specified problems is presented. The technique we propose is based on the concept of transformations oriented into the context of parallelism. In addition we introduce a methodology for deriving correct parallel algorithms.

* Buffo M., Urland E., Rolim J. and Buchs D., "Progressive generation of parallel solutions for formally specified problems", submitted and accepted for IWPP 94, to be published by McGraw Hill.
Abstract:
In this paper we show a progressive way for managing abstract algebraic specifications in order to obtain efficient parallel or distributed algorithms. Our approach is based on TSPP technique (Transformations de Spécifications de problèmes orientées vers le Parallélisme et le Probabilisme). We claim that the process of transformation of specifications can be done, in many cases, automatically in direction from a high level of abstraction to concrete one using a sequence of derivation steps. The resulting specification, obtained by such a transformation process, is then well suited for execution on target parallel machine.
Further we shortly describe the SANDS system (Structured Algebraic Nets Development System) and the TTool - transformation tool package which was developed for performing transformations on specifications.
To illustrate the TSPP technique we present here an example developed in SANDS system environment by application of TTool. Finally we discuss the obtained speedup of execution such a derived specification on parallel machine with various number of processors.

Moreover, we must cite the forthcoming technical report, written in french, which will contains all results obtained in the framework of this project:

* Urland E., Buffo M., Rolim J and Buchs D., "Le projet TSPP". This report will be at least be available as a "Cahier du CUI", at the University of Geneva.

REFERENCES

[1] R. B. Borie, R. G. Parker, and C. A. Towey. Automatic generation of linear time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. Technical report, Department of Computer Science, University of Alabama, 1993.

[2] D. Smith. Kids: A semantic program development system. *IEEE Transactions on Software Engineering*, 16(9), 1990.

[3] CIP Language Group. *The Munich Project CIP, volume 2: The Program Transformation System CIP-S*, volume 292 of *Lecture Note in Computer Science*. Springler-Verlag, 1987.

[4] B. Krieg-Brückner. PROgram development by SPEcification TRAnsformation. *Technique et Science Informatiques*, 9(2), 1990.

[5] Didier Buchs and Nicolas Guelfi. A concurrent object - oriented petri nets approach for system specification. In M. Silva, editor, *12th International Conference on Application and Theory of Petri Nets, Aahrus, Denmark*, June 1991.

[6] Mathieu Buffo. Co-opn semantics. Technical report, University of Geneva, 1993.

[7] Mathieu Buffo. Equivalence of co-opn objects. Technical report, University of Geneva, 1993.

[8] Mathieu Buffo. Transformations : Théorie et outils. Cahier du CUI 78, University of Geneva, 1993.

[9] Mathieu Buffo. Collection de transformations. Cahier du CUI 79, University of Geneva, 1993.

[10] M. T. Heath and J. A. Etheridge. Visualizing performance of parallel programs. Technical Report ORNL/Tm-11813, Oak Ridge National Laboratory, 1991.

[11] J. Rolim, O. Sykora, and I. Vrto. Bisection width of planar graphs. Cahier du CUI 70, University of Geneva, 1993.