Parallel Functional Programming Using D-Clean

The theses of the Ph.D. dissertation

Zsók Viktória

http://zsv.web.elte.hu/
2012
zsv@elte.hu

Supervisor: Prof. Dr. Horváth Zoltán
Eötvös Loránd University, Faculty of Informatics
H-1117 Budapest, Pázmány Péter sétány 1/C.

Ph.D. School of Informatics
Head of Ph.D. School: Prof. Dr. Benczúr András
Ph.D. Programme: Foundations and Methodologies of Informatics
Head of Ph.D. Programme: Prof. Dr. Demetrovics János

Supported by TÁMOP 4.2.1./B-09/1/KMR-2010-0003.
Introduction

Parallel and distributed programming are intensively studied specific areas in the domain of functional programming. Functional programming languages represent modern tools for designing and implementing software which meet the high requirements of the nowadays information society. Functional programs have inherent parallel features, which can be exploited by building up different language tools to obtain reliable parallel processing.

The state-of-the-art literature reports permanent search for expressing parallelism in the domain of functional paradigm. Different attempts for parallelization have been accomplished using various methodologies and approaches starting from programming low-level language primitive to higher level elements. However, parallelism remains a difficult concern especially in the case of lazy functional programming languages.

Parallel programming in the lazy functional programming language Clean had several development phases. At the very beginning, a transputer version of the language was developed. Later on annotations, dynamics, TCP/IP and Object IO elements were used, while nowadays web-based tasks are the main matters for parallelism.

The research question addressed in dissertation is how we can obtain parallel behaviour and communication for functional programs, namely for Clean functions. The continuous search for functional programming language elements that have higher and higher abstraction level for defining parallelism established the main settings for the inspections and research. The direction I followed resulted in increasing the expressing power of language elements for parallelism in functional programs. More specifically, it conducted me to the design of a Clean language extension for distributed programming and coordination of functional programs.

Aims and goals

The main objectives of the thesis are summarized and stated as:

1. to observe and further refine the existing parallel functional language elements,

2. to introduce parallel evaluation strategies suitable for Clean and to define the parallel communication channel objects for interconnecting Clean programs on clusters,

3. to design the D-Clean higher level coordination language as extension of Clean and to define the D-Box intermediate level language together with the mappings from the D-Clean language to the D-Box language,

4. to specify the semantics of the D-Clean language in the form of executable Clean functions,

5. to implement parallel algorithmic skeletons using the D-Clean primitives and to inspect their semantical soundness.
The structure of the dissertation

The dissertation is organized in seven chapters. The introduction presents the topics of the dissertation, while the related work chapter describes the background of the research and enumerates the related works from various perspectives. It points out the differences and similarities between the research results of the dissertation and some key references of the literature.

The chapter about parallel evaluation strategies describes the initial research phase regarding the strategies that have influences upon the evaluation degree of functions, while the chapter about the Clean-CORBA interface explores communication possibilities between Clean programs on clusters.

The D-Clean coordination language chapter describes in details the newly introduced D-Clean primitives and the structure of the D-Box language elements. For achieving parallel features, the lazy functional programming language Clean is extended by a relatively small number of specific, well defined D-Clean language constructs.

The chapter of executable semantics aims to provide the semantics of D-Clean in form of Clean function definitions. The skeleton library chapter presents skeleton based functional and distributed programming using the D-Clean and D-Box languages, since the main purpose of introducing the two languages was to define parallel computation schemes, i.e. skeletons.

At the end of the dissertation, conclusions are formulated and future research is considered. Although the D-Clean system was designed to coordinate functional programming tasks on clusters using high level coordination language parameterized by functional programming computation nodes, the coordination model may be applied at distributed computations implemented in other programming languages too.

Each chapter first introduces the topic by the motivation behind the presented work, followed by the description of the main concerns and important features. Case studies related to the topic ended by evaluations of their methods are also included. The chapters are focusing on the language elements and their expressing power concerning parallelism in functional programming. The novel results of the chapters were published as peer-reviewed journal papers. In the following these outcomes and their novelties are summarized in four main thesis (the first thesis has two parts).

The summary of the results

Thesis 1.1: Based on the parallel annotations of the pure lazy functional language Clean, we introduced parallel evaluation strategies in Clean. I defined compound parallel evaluation strategies applied on linear data structures.

Strategies define the dynamic behaviour of functional programs. Implemented as higher order functions, the most important feature of the strategies is to separate the evaluation behaviour of a function from its algorithmic part. The defined ‘par’ Clean parallel evaluation strategy influences the evaluation degree of expressions. Parallel evaluation strategies are also
very important in the case of compound types, because a high evaluation degree may be achieved in a shorter evaluation time for composed functions. As a case study for parallel compound strategies, the parallel elementwise processing method is presented.

The novelty of the evaluation strategy implementation is given by the Clean language specificity of using graph rewriting technique for the evaluation of the functions. Changing the evaluation degree by strategies, and creating simple and composed parallel evaluation strategies are the new parts of this thesis.


**Thesis 1.2:** I created type-safe functional communication channels for interconnecting Clean programs distributed on clusters using the Clean-CORBA interface and middleware services.

The second research phase dealt with the integration of communication channels into Clean functional programs. The interconnections are accomplished via direct calls of middleware services. The abstract communication layer is designed and implemented based on CORBA server objects. A Clean-CORBA interface is used for server-client communications. The communication between the distributed functional components is asynchronous. A skeleton for pipeline computation is included as a case study, which describes the main features of the channel-based communication approach. Using CORBA-based technique it became obvious that new higher order language elements are needed in order to achieve clearer and more transparent code while hiding implementation details and reducing the possibility of error appearances.

The approach of channel based communication between functional programs on clusters is novel for parallel functional programming.


Related publications: [30], [9], [2].

**Thesis 2:** I designed the set of distributed coordination language primitives D-Clean as an extension of the Clean lazy functional programming language.

D-Clean consists of high-level language elements for the coordination of the purely functional computational nodes in the designed distributed environment for a cluster. D-Clean constructs generate D-Box computation boxes connected via buffered tools for communications, that are channels.
D-Clean also controls the dataflow in a distributed process-network. The dataflows carried by the channels are typed according to the rules of the input-output protocols of the boxes generated for the evaluation of the computations. Using D-Clean the programmer indicates how a distributed computation can be organised into a generated computational graph.

The D-Box language is an intermediate level language based on Petri nets. The boxes are compiled into Clean programs, and they communicate via channels.

The D-Clean primitives offer the advantage of writing distributed and functional applications without knowing the details of the application environment and middleware services abstracting from the technical aspects. The distributed computation uses a multiparadigm-oriented environment with several different layers. On the highest level the D-Clean coordination language primitives are applied in order to define the distribution of the pure functional computation subtask. This distribution is made according to a predefined computational scheme, which is an algorithmic skeleton, parameterized by functions, types and data. The D-Clean programs are transformed into D-Box programs, which are lower level descriptions including the computational nodes implemented in the lazy functional language Clean. The D-Clean to D-Box mapping rules are also provided.

The introduced D-Clean coordination language is innovative in the abstraction level of its language primitives. The designed primitives are higher abstracted from the low-level constructs than the earlier introduced evaluation strategies and the communication channels are better conformed to the functional programming style. The language is a new high-level distribution approach of the coordination of Clean functional programs on a cluster using the intermediate language D-Box with well established channels for distributed communication and boxes for computation nodes.

The main publication is the journal paper [4]:


Related publications: [22], [24], [1].

**Thesis 3:** *I specified the operational semantics of the D-Clean language in the form of executable semantics using Clean functions and types in order to verify and prove the properties of the D-Clean language extension.*

The D-Clean and D-Box coordination languages introduced are applied for implementing high-level process description and communication coordination of functional programs distributed over a cluster. However, practical experiences with the two-language usage in distributed program development showed that it would be helpful to build a tool for prior automatic visualization of the computation network. Therefore, a new modelling approach is provided and a new view of the D-Clean distributed system is defined using semantical descriptions implemented in Clean. The emphasis is mainly on the executable description of the operational
semantics of the D-Clean language primitives. The descriptions of the D-Clean elements are abstracting from the real distributed environment and leaving out the details of the box and channel generation. The executable semantics tool is a software comprehension application for a better way of utilizing the distributed D-Clean coordination language.

D-Clean programs are also depicted graphically, formulating properties of the executable semantics of the visualized example. For well defined distributed computations the expected parallelism, the potential amount of parallelism (analogously to the real distributed system) are drawn by boxes and by channels generated in parallel. The graphical visualization of the distributed computation using the executable semantics provides useful support when programming in the real distributed system. A second model of D-Clean semantics is provided using C++ templates. The strong type system of C++ templates guarantees the correctness of the model. Using templates impressive efficiency is achieved by avoiding run-time overhead.

The new language implied new semantics, and the executable semantics approach is also relatively novel in the literature.

The main publications are the journal papers [32] and [35]:


Related publications: [36], [33], [34], [21].

**Thesis 4:** I defined and implemented parallel algorithms, that are skeletons based on the new language primitives. I verified the equivalences of the skeletons using the distributed environment and the semantical execution tool.

The main purpose of introducing the two coordination languages was to define parallel computation skeletons. The aim is to illustrate the appropriateness and applicability of the two languages for distributed evaluation of the functional programs on clusters. Therefore, distributed functional computational skeletons are provided. A set of known algorithmic skeletons are tested and implemented as D-Clean schemes.

In a large number of D-Clean examples high speed-up for parallelism was obtained. The actual amount of parallelism depends on many factors such as: the order of channel creation, the amount of work on one channel, the speed of the data retrieving and data storage in the channels, the complexity of the computation described in the distributed graph.

Some of the well-known classical skeleton examples were inspected both for semantical soundness in D-Clean and for distributed computation coordination, which represents the newness of the implemented skeletons.
The main publication is the revised selected lecture notes [25]:


Related publications: [35], [22], [24], [23], [5], [26], [27].

Conclusions

In the initial research, the most important task was to get acquainted with new parallel issues in functional programming. Afterwards, two important methods were applied to obtain parallel computation in Clean: changing the evaluation strategies and communicating via Clean-CORBA channels. As both methods are based on low-level language constructs, further investigation was needed to obtain language elements with high-level abstraction properties.

The design of the new D-Clean coordination language enabled the development of parallel skeleton algorithms in the built distributed and multilayered system. The introduced D-Clean language, extension of the Clean lazy pure functional language, uses the D-Box intermediate language for channel based parallel communication of Clean client programs embedded in boxes. The semantics of the D-Clean language was tested in executable way, and measurements were included in numerous papers published.

References


Books (eds.):


Chapters in books:


Publications in Hungarian:


