The content of TDA383/DIT390 and the languages used

K. V. S. Prasad
Department of Computer Science
Chalmers University

August 23, 2016

This course teaches the principles of concurrent programming through a textbook, lectures and exercises. In this run of the course, autumn 2016, the exam will test you only on these principles. The labs, completely separate from the exam, will teach you to apply these principles in real-world languages. Your grade depends only on your performance on the exam and the labs.

1 The textbook

It is not always fashionable or even possible to tie a course strongly to a textbook, but in our case we have a book that agrees quite largely with our course aims: *Principles of Concurrent and Distributed Programming*, 2nd Edition, by Mordechai (Moti) Ben-Ari, Addison-Wesley, 2006. This book is therefore the major teaching tool for the course, and will be referred to in most lectures. We strongly recommend you ensure access to a copy.

2 The programming languages used

2.1 For the labs

Java Labs 1 and 2 are to be done in Java. You will get a lecture on how to write concurrent programs in Java before the first lab. The textbook offers very good support.

Erlang Labs 3 and 4 are to be done in Erlang, a functional language widely used in the industry to write programs with thousands of concurrent processes. You will get two lectures on Erlang during the third week of the course. But if you have never used a functional language before, do skim right away the first two chapters of the quick Erlang tutorial (also available in PDF format from the same page). Download Erlang OTP, install it on your computer, and work through the first 10 pages or so of the tutorial. You might also try the first few pages of learn you some Erlang.

2.2 For the lectures and the exam

Pseudo-code The textbook uses a pseudo-code notation (summarised here) to write the programs to be discussed. The notation concentrates on concurrency, stripping down all other issues. (The book then often shows how to implement the pseudo-code programs in C, Java and Ada). We use the same pseudo-code notation in the lectures and the exam; indeed, it is the only programming language you need to know for the exam.

Pseudo-code notation is flexible, so we can easily add new constructs, but it is not executable, and it is essential to run programs to understand them. We can of course design in pseudo-code, implement in C or Java, and then test the implemented program. But the gap between the languages means we may not be testing what we designed.

Promela Fortunately, there is a runnable language close to pseudo-code, Promela. (Or you can think of the pseudo-code notation as a blackboard simplification of Promela). Like the pseudo-code, Promela too can model most concurrency constructs, including those of Java and Erlang. This tutorial introduces the needed Promela constructs as it goes along.

We use Promela as a teaching aid to quickly try out programs and test their properties. It runs on Spin, an interpreter for Promela. Spin can also check properties of programs. You will see shortly that this last is essential to understand concurrent programs.

Neither the labs nor the exam require knowledge of Promela. Even the lectures will mostly use the pseudo-code; only briefly will we use Promela to demonstrate the behaviour of the program being discussed. So **Promela is not really needed in the lectures either**. It is only needed if you choose to attend exercise sessions based on it.

2.3 For the optional exercises: Java, Erlang and Promela

The exercises are not to be submitted; they carry no weight for your grade, but are offered simply as learning aids. We give you example programs in Java, Erlang and Promela that you can read, run and modify to see what happens. You are usually left to do the exercises in your own time, but we will sometimes organise a supervised session.

2.4 The content of the exam

We hope the lectures and exercises will help you understand concurrency, but to prepare for the exam, you need strictly speaking only Chapters 1 through 9 of the textbook. Sections of the textbook specifically dealing with languages we don't use (Ada, C and BACI) can be skipped with caution (be aware that important ideas might be illustrated in those sections; for example, protected objects in Ada). Promela and Spin are not examinable material; they will be used only as teaching aids.

Remember that the pseudo-code notation of the textbook and the lectures is the only programming language you need to know for the exam.

2.5 Teaching style (PLEASE MAKE YOUR OWN NOTES)

The lectures will be keyed to the textbook whenever possible. We will use slides from the book as well as slides of our own. These will not in general be available before the lecture, as the content of each lecture is decided dynamically depending on classroom discussions. Many of these discussions will use the blackboard.

It is therefore **important that you bring a notebook and make your own notes**, and that you understand that the slides are in no sense "lecture notes", or any kind of substitute for the textbook or even for the lectures. Slides are visual aids for the lecture, that is all. After the lecture, they may help you remember what was discussed.