

Revision Lecture

A0. [Introduction, overview.](#)

A1. Safety criteria.

A2. Hazard analysis.

A3. Programming languages for writing safety-critical software.

B1. Introduction.

B1. The logical framework.

B3. Data types.

(C) Anton Setzer 2003 (except for pictures)

General

- According to my data, exam Monday 2/6, 14.00, Dining House.
 - Please check.
- **Essentially everything needed should be contained in** (For legal purposes I reserve the right to deviate from that)
- I will come to the exam in the first half hour (approx.; questions).
- All three **courseworks** are preparation for the exam.

Structure of the Exam

- **3 Questions.**
- **Question 1** on software engineering aspects.
 - Some book work.
 - * Essentially “bullets” in the notes.
 - * Usually only 3 or 4 required, no need to learn long lists.
 - Mainly application of knowledge in examples.
- **Question 2** some general questions about Agda and critical software in Agda.

Structure of the Exam

- **Question 3** formal treatment of Agda.
 - Simple questions about Agda.
 - One simple derivation using rules.
 - Some derivations in Agda.

(C) Anton Setzer 2003 (except for pictures)

Stream A

- A few questions which test concrete knowledge.
- Most software-engineering questions require
 - application of your knowledge to something slightly different (e.g. evaluation of techniques, methods)
 - or simple calculations.

A0: Introduction

- **Definition** of “critical system”, “safety-/mission-/business critical system”
- **Areas** of critical systems (only some).
- **Lessons** to be learned from Åsta train crash.
 - Some (at least 4; not complete list).
 - What is a **root cause**?
 - **Preliminary events, initiating/trigger event, intermediate events, and final event** (ameliorating, propagating).
 - **Two aspects of safety critical systems** (software engineering aspects for writing safety critical software).
 - * What is the **system aspect** of safety critical systems?
- **Tools for writing correct software**: different levels of rigor

A1: Safety Criteria

- **Requirements document.**
 - What is a requirements document?
 - Functional/nonfunctional requirements;
functional safety requirements/nonfunctional safety requirements
context of operation.
 - Hazard, accidents, incidents/near misses.
 - * Definitions.
 - Risk.
 - * Definition, calculation.

General System Requirements

- **Dimensions of Dependability.**
 - Reliability, availability, maintainability, system integrity, safety, security, data integrity.
 - No need to learn this list.
- **Definitions** of the notions (reliability, availability, failsafe)
- **Calculation** of reliability, availability, comparison of the two
- **Mean time to repair, mean time to failure, mean time to failure** Calculations.
- **Notions in the area of security** (exposure, vulnerability, control, survivability).

Identification of Safety Requirements, Safety Case

- **Classification** of hazards (w.r.t. what).
- **Risk reduction**
(Design out hazard; safety devices; interlocks; warning techniques)
- **Safety case** (Definition).
- Why **standards**?

A2 Hazard Analysis

- **Which techniques?**
 - FMEA, FMECA, HAZOP, ETA, FTA.
- **FMEA.**
 - Limitations.
 - Process of FMEA (outline).
- **FMECA.**
 - FMEA + calculation/estimation of probability of occurrence, probability of consequences, c
 - Product = measure for the risk.

Hazard Analysis (Cont.)

- **HAZOP.**

- Outline.
- Examples of guide words.
 - * General use and use for computer based systems.
 - * Some example of guide words specific for computer b
 - (Early, late, before, after).

- **ETA.**

- How does it work?
- Example (including calculation of probabilities).
- Problems (becomes too big)

Hazard Analysis (Cont.)

- **FTA.**

- How does it work?
- Examples (how to draw a circuit).
- No cut sets.
- Difference FTA, ETA
(ETA starts with faults and determines resulting accidents)
FTA starts with accidents and determines faults resulting in accidents

A3 Programming Languages for Writing Safety-Critical Software

- **Criteria for choice of programming languages for comparison of languages.**
 - Which factors make languages suitable/unsuitable for software?
 - Why does one not invent new programming languages? Why does one introduce subsets of existing ones?
- **Common reasons for programming errors.**
 - Overview.

(C) Anton Setzer 2003 (except for pictures)

Ada

- **Why was Ada developed?**
- **Why is Ada used in critical systems?**
 - Coming from department of defense.
 - High portion of safety critical systems is coming from c
 - Ada as well specially designed for such purposes (e.g. r

SPARK Ada

- **Basic principle.** (Subset of Ada; compiles with Ada compiler which are verified by the SPARK Ada tools).
- What's checked by a **data flow analysis?**
 - Input/output behaviour of parameters, initialization of input parameters are used and output parameters are c
- What's achieved by a **information flow analysis?**
 - User specifies interdependencies of variables. These are
- What is the basis for **verification conditions?**
 - Only necessary to know: Pre-and post-conditions in pro

(C) Anton Setzer 2003 (except for pictures)

Stream B

- Main part of the lecture.
- A few questions which test concrete knowledge.
- Most questions about type theory require
 - derivations using rules (1 subquestion)
 - derivations in Agda (main part of question 2/3),
 - modelling of critical systems.

Stream B (Cont.)

- Similar to what was done in the lecture and in coursework
- Understanding of basic Agda syntax (minor syntactic penalized, as long as it is clear that with the help of a r obtain a syntactically correct proof).
- Simplified examples, since no computers available.

B1 Introduction

- **4 Principal approaches** for writing verified software.
- **Concept of a type.**
 - Advantages of **typed**, of **untyped** languages.
 - Why are types good for writing correct software?
- **Examples of types** in other languages
 - Scalar types (eg. Booleans, Integers).
 - Simple compound types (records, arrays).
 - Function types, inductive data types.
 - Interfaces.

B1 Introduction (Cont.)

- **4 kinds of judgements** in dependent type theory:
 - $A : \text{Type}$, $A = B : \text{Type}$, $a : A$, $a = b : A$.
 - Only $A : \text{Type}$, $a : A$ are visible in Agda.
- **Examples of dependent types** in programming.
 - Templates (eg. in C++) ie. parametric types.
 - Matrix multiplication.
 - Predicates.
 - Dependent grammars in linguistics.

(C) Anton Setzer 2003 (except for pictures)

B1 Introduction (Cont.)

- **Simple Derivations using rules.**
- **Basic types** (function type and product) – more how to than to learn lots of details.

B2 The Logical Framework.

- **4 kinds of rules** (formation, introduction, elimination, eq)
- **Constructors** and **canonical elements**.
- **Dependent function type, dependent product.** (Basic use them in Agda).
- **let expressions.**
- **Presuppositions.**
- **Structural rules** (for use in derivations).
- Notion of **Set** vs. **Type** (how to use Set, Type).

B3 Data Types

- **Basic data types** (Booleans, finite sets, disjoint unions, numbers, vectors of length n , lists).
 - Universes and algebraic data types **not** treated in this lecture.
- Modelling of the **traffic light example**.
- **Atomic formulae**. atom.
- How to represent other **formulae** in type theory.
E.g. $\forall x : \mathbb{N}. A$ becomes $(x : \mathbb{N}) \rightarrow A$.
 - No details about constructive logic.

(C) Anton Setzer 2003 (except for pictures)

B3 Data Types (Cont.)

- Derivations in the context of **N**. (Definition of $==$, $<$; si
- **Termination checker** (what can it do? what are the limi
- Verification of a **circuit** (Coursework 3).