A0. Introduction, overview.
A2. Hazard analysis.

B1. Introduction.
B2. The logical framework.
B3. Data types.

Rev-1
Revision Lecture

Critical Systems, CS-411, Lentterm 2003, Revision Lecture
(C) Anton Setzer 2003 (except for pictures)
According to my data, exam Monday 2/6, 14:00, Dining Room C, Fulton House.

- Please check.

Essentially everything needed should be contained in the notes.

For legal purposes I reserve the right to deviate from that rule.

I will come to the exam in the first half hour (approx.; in case you have questions).

According to my data, exam Monday 2/6, 14:00, Dining Room C, Fulton House.

All three coursework are preparation for the exam.

General

Critical Systems, CS-411, Lenterm 2003, Revision Lecture

Rev-2 (C) Anton Setzer 2003 (except for pictures)
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 by heart.

Mainly application of knowledge in examples.

Question 2

some general questions about Agda and one verification of critical software in Agda.

Question 3

Critical Systems, CS-411, Lenterm 2003, Revision Lecture

Rev-3
Structure of the Exam

- Some derivations in Agda.
- One simple derivation using rules.
- Simple questions about Agda.

• Question 3: Formal treatment of Agda.
A few questions which test concrete knowledge.

• Most software-engineering questions require

− or simple calculations.

− evaluation of techniques, methods

− application of your knowledge to something slightly different (essentially Stream A)
Definition of "critical system", "safety-/mission-/business-critical system".

Areas of critical systems (only some).

Lessons to be learned from Aasta train crash.

Tools for writing correct software: different levels of rigour.

- What is the system aspect of safety critical systems?
- What is the system aspect of safety critical software?
- Tools for writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
- Preliminary events, Initiating/trigger event, Intermediate events
- Ameliorating, Propagating.
- Two root causes?

Areas of critical systems (only some).

Lessons to be learned from Aasta train crash.

Tools for writing correct software: different levels of rigour.

- What is the system aspect of safety critical systems?
- What is the system aspect of safety critical software?
- Tools for writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
- Preliminary events, Initiating/trigger event, Intermediate events
- Ameliorating, Propagating.
- Two root causes?

Definition of "critical system", "safety-/mission-/business-critical system".

Areas of critical systems (only some).

Lessons to be learned from Aasta train crash.

Tools for writing correct software: different levels of rigour.

- What is the system aspect of safety critical systems?
- What is the system aspect of safety critical software?
- Tools for writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
- Preliminary events, Initiating/trigger event, Intermediate events
- Ameliorating, Propagating.
- Two root causes?

Definition of "critical system", "safety-/mission-/business-critical system".

Areas of critical systems (only some).

Lessons to be learned from Aasta train crash.

Tools for writing correct software: different levels of rigour.

- What is the system aspect of safety critical systems?
- What is the system aspect of safety critical software?
- Tools for writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
- Preliminary events, Initiating/trigger event, Intermediate events
- Ameliorating, Propagating.
- Two root causes?

Definition of "critical system", "safety-/mission-/business-critical system".

Areas of critical systems (only some).

Lessons to be learned from Aasta train crash.

Tools for writing correct software: different levels of rigour.

- What is the system aspect of safety critical systems?
- What is the system aspect of safety critical software?
- Tools for writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
- Preliminary events, Initiating/trigger event, Intermediate events
- Ameliorating, Propagating.
- Two root causes?
A1: Safety Criteria

Requirements document.

* Definitions, calculation.
  - Risk.
* Definitions.
  - Hazard, accidents, incidents/near misses.
  - Context of operation.

Functional safety requirements/nonfunctional safety requirements:

- Functional/nonfunctional requirements.

What is a requirements document?
General System Requirements

- Definitions of the notions (reliability, availability, fail-safe state etc.)
- Critical Systems, CS 411, Lent term 2003, Revision Lecture
- Mean time to repair, mean time to failure, mean time between failures.
- Calculation of reliability, availability, comparison of the two, unavailability.

Dimensions of Dependability
- Safety, security, data integrity.
- Reliability, availability, maintainability, system integrity, system recovery.
- No need to learn this list.

Notions in the area of security (exposure, vulnerability, attack, threat, control, survivability).

Calculations.
Identification of Safety Requirements

Classiﬁcation of hazards (w.r.t. what).

Risk reduction

Safety case (Deﬁnition).

Why standards?

Design out hazards; safety devices; interlocks; warning signs; management techniques

Safety Case

Identification of Safety Requirements

(C) Anton Setzer 2003 (except for pictures)
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, criticality.
- Product = measure for the risk.
HAZOP.

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

ETA.

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

HAZOP.

Hazard Analysis (Cont.)
FTA starts with accidents and determines faults resulting in that event.

- Difference FTA, ETA
- No cut sets.
- Examples (how to draw a circuit).
- How does it work?

FTA.

Hazard Analysis (Cont.)
Criteria for choice of programming languages for critical systems:

• Common reasons for programming errors:
  - Why does one not invent new programming languages, but instead introduce subsets of existing ones?
  - Why does one not invent new programming languages, but instead introduce subsets of existing ones?
  - Which factors make languages suitable/unuitable for writing critical software?

Common reasons for programming errors:

- Overview.
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 defense industry.

Ada as well specially designed for such purposes (e.g., real time systems).
Basic principle. (Subset of Ada; compiles with Ada compilers; annotations which are verified by the SPARK Ada tools).

What's checked by a data flow analysis?
- Input/output behaviour of parameters, initialization of variables;
- Input and output parameters are used and output parameters are changed.

What's achieved by a program flow analysis?
- User specified interdependencies of variables. These are checked.

What is the basis for verification conditions?
- Only necessary to know: Pre- and post-conditions in procedures.

What's checked by a data flow analysis?
- Input/output behaviour of parameters, initialization of variables;

What's checked by a program flow analysis?
- User specified interdependencies of variables. These are checked.

What's achieved by a program flow analysis?
- User specified interdependencies of variables. These are checked.

Basic principle. (Subset of Ada; compiles with Ada compilers; annotations which are verified by the SPARK Ada tools).
Main part of the lecture.

- Modelling of critical systems.
- Derivations in Agda (main part of question 2/3).
- Derivations using rules (1 subquestion).

Most questions about type theory require:

A few questions which test concrete knowledge.

- Stream B
Simplified examples, since no computers available.

- Obtain a syntactically correct proof.
- Simpler to what was done in the lecture and in coursework.

• Similar to what was done in the lecture and in coursework.

Stream B (cont.)
Introduction

Principal approaches for writing verified software.

Concept of a type.

Exemples of types in other languages.

Why are types good for writing correct software?

Advantages of typed, of untyped languages.

Interfaces.

Function types, inductive data types.

Simple compound types (records, arrays).

Scalar types (e.g. Booleans, Integers).

Critical Systems, CS-411, Lenterm 2003, Revision Lecture
4 kinds of judgments in dependent type theory:

- \textbf{Examples of dependent types in programming:}
  - Templates (e.g. in C++) i.e. parametric types.
  - Matrix multiplication.
  - Predicates.
  - Dependent grammars in linguistics.

- Only \( A : \text{Type}, a : A \) are visible in Agda.
- \( A : \text{Type}, a, q = a, A' : A \).
- \( A : \text{Type}, B = A, V' : V \).

- Dependent judgments in dependent type theory.

\textbf{BI Introduction (cont.)}
Simple Derivations using rules.

• Basic types (function type and product) – more how to use them rather than to learn lots of details.
The Logical Framework.

4 kinds of rules (formation, introduction, elimination, equality).

Constructors and canonical elements.

Dependent function type, dependent product.

Presuppositions.

Structural rules (for use in derivations).

Notion of Set vs. Type (how to use Set, Type).

let expressions.

use them in Agda.

Basic rules and how to use them in Agda.

Critical Systems, CS 411, Lent term 2003, Revision Lecture Rev-21
Basic DataTypes

Basic data types (Booleans, finite sets, disjoint union, \(\mathbb{Z}\)-set, natural numbers, vectors of length \(n\), lists).

Modeling of the traffic light example.

How to represent other formulæ in type theory.

Atomic formulæ.

No details about constructive logic.

E.g. \(\forall x : N. A \rightarrow A\).
• Derivations in the context of $\mathbb{N}$ (Definition of $\mathbb{N}$).

• Termination checker (what can it do? what are the limitations?).

• Verification of a circuit (Coursework 3).

• B3 Data Types (cont.).