General

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

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, please check.)

All three coursework assignments are preparation for the exam.

Structure of the Exam

Question 1
- Some definitions in Agda.
- Some derivations in Agda.
- Mainly application of knowledge in examples.
- Usually only 3 or 4 required; no need to learn long lists by heart.

Question 2
- Some general questions about Agda and one verification of Agda.

Question 3
- Some general questions about Agda and one verification of Agda.
- Some book work.
- Some derivations in Agda.

3 Questions.

Revision Lecture

A0. Introduction, overview.
A2. Hazard analysis.
B1. Logical framework.
B2. Introduction.
B3. Data types.
A2. Hazard analysis.
A0. Introduction, overview.
A0: Introduction

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

Areas of critical systems (only some):

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

A1: Safety Criteria

Requirements document.

- What is a requirements document?
- Functional & non-functional requirements;
- Context of operation;
- Functional & non-functional safety requirements;
- What is a requirements document?
- Calculation of safety criteria.

Risk

- Definition, calculation.
- Risk
- Definition.

Tools for writing correct software: different levels of rigor.

- What is the system aspect of safety critical systems?
- For writing safety critical software.
- Two aspects of safety critical systems (software engineering vs. tools)
  - Mathematical foundations
  - Preliminary events, initialising/trigger event, intermediate events
  - What is the root cause?
- Some (at least 4; not complete list).

Lessons to be learned from A3a train crash:

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

Stream A
Identification of Safety Requirements, Safety Case

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

Hazard Analysis (Cont.)

- FMEA.
  - FMECA.
  - ETA.
  - FTA.

Hazard Analysis (Cont.)

- Which techniques?
  - FMEA.
  - FMECA.
  - ETA.
  - FTA.

A2 Hazard Analysis

- How does it work?
  - Examples (how to draw a circuit).
  - Example (how to draw a circuit).

- Limitations.
  - Process of FMEA (outline).
  - Process of FMEA (outline).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).

- ETA starts with events and determines resulting accidents (events).
- ETA starts with events and determines resulting accidents (events).

- FTA starts with faults and determines resulting accidents (events).
- FTA starts with faults and determines resulting accidents (events).

- FMEA starts with faults and determines resulting accidents (events).
- FMEA starts with faults and determines resulting accidents (events).
A3 Programming Languages for Writing Safety-Critical Software

Criteria for choice of programming languages for critical systems; comparison of languages.

Which factors make languages suitable/unsuitable for writing critical software?

Why does one not invent new programming languages, but instead introduces subsets of existing ones?

Common reasons for programming errors.

Critical Systems, CS 411, Lent term 2003, Revision Lecture

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)

Critical Systems, CS 411, Lent term 2003, Revision Lecture

SPARK Ada

Basic principle (subset of Ada) complements Ada compiler's annotations

Only necessary to know: pre-and post-conditions in procedures.

What is the basis for verification conditions?

User specifies interdependences of variables. These are checked.

What is checked by a data flow analysis?

Input/output behaviour of parameters, initialization of variables.

Stream B

SPARK Ada

- Only necessary to know: pre- and post-conditions in procedures.

- What is the basis for verification conditions?

- User specifies interdependences of variables. These are checked.

- What is checked by a data flow analysis?

- Important parameters are used and outcomes of parameters are changed.

- Basic principle (subset of Ada) complements Ada compiler's annotations

- Which are verified by the SPARK Ada tools.

- Overview.

- Common reasons for programming errors.

- Introduction subjects of existing ones.

- Why does one not invent new programming languages, but instead introduces subsets of existing ones?

- Which reasons make languages suitable/unsuitable for writing critical systems?

Critical Systems, CS 411, Lent term 2003, Revision Lecture

A3 Programming Languages for Writing Safety-Critical Software
StreamB (Cont.)

Similarto what was done in the lecture and in coursework. •

Understanding of basic Agda syntax (minor syntactic errors will not be penalized, so long as it is clear that the help of a machine one would obtain as a non-exercising correct proof). •

4 kinds of judgements in dependent type theory:

- Basic types (function types and product) – more how to use them rather than to learn lots of details. •
- Simple derivations using rules. •

Examples of dependent types in programming:

- Only A: TyPe: a: A are visible in Agda. •
- A: q = q: TyPe: a: A, A: A. •
- A: q: TyPe: A = B: TyPe: a: A. •
- 4 kinds of judgements in dependent type theory:

- Interfaces. •
- Function types, inclusive data types. •
- Simple compound types (records, arrays). •
- Scalar types (e.g., Booleans, integers). •

Examples of types in other languages •

Advantages of typed, of untyped languages. •

Concept of a type. •

4 Principal approaches for writing verified software. •

Critical Systems, CS 411, LentTerm 2003, Revision Lecture

(B) Anton Setzer 2003 (except for pictures)
4 kinds of rules (formation, introduction, elimination, equality).

- Constructors and canonical elements.

- Dependently typed, dependent product (basic rules and how to use them in Agda).

- Structural rules (for use in derivations).

- Presuppositions.

- let expressions.

- Modelling of the traffic light example.

- - Universes and algebraic data types not treated in this lecture.

- - Numbers, vectors of length n, lists.

- - Basic data types (booleans, finite sets, disjoint union, z-set, natural.

- Atomic formulae.

- • How to represent atomic formulae in type theory.

- • Modelling of the traffic light example.

- • Universes and algebraic data types not treated in this lecture.

- • Numbers, vectors of length n, lists.

- • Basic data types (booleans, finite sets, disjoint union, z-set, natural.

- Critical systems.