A0. Introduction, overview.
A2. Hazard analysis.

B1. Introduction.
B2. The logical framework.
B3. Data types.
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 any questions).

All three coursework are preparation for the exam.
The structure of the exam:

- **3 Questions.**

  - **Question 1.** On software engineering aspects.
    - Mainly application of knowledge in examples.
    - Usually only 3 or 4 required, no need to learn long lists by heart.
    - Essentially "bullets" in the notes.
    - Some book work.

  - **Question 2.** On critical software in Agda.
    - Some general questions about Agda and one verification of Agda.

(C) Anton Setzer 2003 (except for pictures)
Structure of the Exam

- Question 3 formal treatment of Agda.
  - Simple questions about Agda.
  - One simple derivation using rules.
  - Some derivations in Agda.
Most software-engineering questions require

- application of your knowledge to something slightly different (essentially evaluation of techniques, methods)
- or simple calculations.

A few questions which test concrete knowledge.

Stream A
Introduction

Definition of "critical systems", "safety-mission-business-critical systems".

Areas of critical systems (only some).

Lessons to be learned from Asta train crash.

Tools for writing correct software: different levels of rigour.

What is the system aspect of safety-critical systems?

- Preliminary events, initiating/triggering event, intermediate events (ameliorating, propagating).
- Two aspects of safety-critical systems (software engineering vs. tools)

Root cause?

Some (at least 4; not complete list):

- Preliminary events, initiating/triggering event, intermediate events (ameliorating, propagating).
- What is the system aspect of safety-critical systems?

Two aspects of safety-critical systems (software engineering vs. tools).

What is the system aspect of safety-critical systems?

Tools for writing correct software: different levels of rigour.
A1: Safety Criteria

Requirements document.

- Definitions.
  - 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.).

- Calculations of the notions (reliability, availability, comparison of the two, unavailability).

- Mean time to repair, mean time to failure, mean time between failures.

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

"No need to learn this list:"

- Safety, security, data integrity.
- Reliability, availability, maintainability, system integrity, system recovery.

Dimensions of Dependability.
Identification of Safety Requirements,

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

Risk reduction (Design out hazard; safety devices; interlocks; warning signals; management techniques)

Safety case (Deﬁnition).

Why standards?

Safety Case

Identification of Safety Requirements,
A2 Hazard Analysis

- Which techniques?

- FMEA, FMECA, HAZOP, ETA, FTA.

- FMEA + calculation/estimation of:
  - Product = measure for the risk.
  - Probability of occurrence, probability of consequences, criticality.

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

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

(C) Anton Setzer 2003 (except for pictures)
HAZOP (cont.)

- 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.
- Examples of guide words.
- Outline.

HAZOP.
FTA starts with accidents and determines faults resulting in that event.

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

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

FTA (Cont.)
Criteria for choice of programming languages for critical systems;

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

Common reasons for programming errors.

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

Overview.
Ada

Why was Ada developed?

- 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).

Why is Ada used in critical systems?

- Critical Systems, CS-411, Lentterm 2003, Revision Lecture
  (C) Anton Setzer 2003 (except for pictures)
Basic principle. (Subset of Ada; compiles with Ada compilers; annotations which are verified by the SPARK Ada tools.)

What is the basis for verification conditions?

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

What is achieved by a data flow analysis?

- Input/ output behavior of parameters, initialization of variables.

What is achieved by a information flow analysis?

- User specifies interdependencies of variables. These are checked.

What's checked by a data flow analysis?

- Input/output behavior of parameters, initialization of variables.

What's checked by a information flow analysis?

- User specifies interdependencies of variables. These are checked.

What's checked by a data flow analysis?

- Input/ output behavior of parameters, initialization of variables.

What is the basis for verification conditions?

- Only necessary to know: Pre- and post-conditions in procedures.
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
Similar to what was done in the lecture and in coursework.

Stream B (cont.)
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.
- Interfaces.
- Function types, inductive data types.
- Simple compound types (records, arrays).
- Scalar types (e.g. Booleans, Integers).
4 kinds of judgments in dependent type theory:

- Only $A : \text{Type}, a : A$ are visible in Agda.
- $A : \text{Type}, a : A, \lambda q = a, q : B = a : \text{Type}$.

Examples of dependent types in programming:

- Predicates.
- Matrix multiplication.
- Templates (e.g., in C++), i.e., parametric types.

Dependent grammars in linguistics.
Simple Derivations using rules.

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

- Notion of Set vs. Type (how to use Set, Type).
- Structural rules (for use in derivations).
- Presuppositions.
- Let expressions.
- Use them in Agda.
- Dependent function type, dependent product.
- Constructors and canonical elements.
- 4 kinds of rules (formation, introduction, elimination, equality).
- Basic rules and how to use them in Agda.
Basic data types (Booleans, finite sets, disjoint union, \(\mathbb{Z}\)-set, natural numbers, vectors of length \(n\), lists).
• Derivations in the context of \( \mathbb{N} \) (Definition of \( \geq \), \( > \); simple proofs).

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

• Verification of a circuit (Coursework 3).

B3 Data Types (Cont.)