Introduction to Declarative Modelling

Michael R. Hansen

mrh@imm.dtu.dk

Informatics and Mathematical Modelling
Technical University of Denmark
Welcome

The teachers

- Michael R. Hansen
- Anne Haxthausen
- Jørgen Villadsen

welcome you to the new course:

02153 DECLARATIVE MODELLING
Welcome

The teachers

- Michael R. Hansen
- Anne Haxthausen
- Jørgen Villadsen

welcome you to the new course:

02153 DECLARATIVE MODELLING

What is a declarative model?
Today

- Introduction to declarative modelling (308.11 — here)
- Introduction to the programming language SML (308.11)
- Make your first programs in the databar (303.43 — G-databar)
- Introduction to lists (SML) (303.43 — G-databar)
- Computations with polynomials in SML (303.43 — G-databar)
Imperative models of computations are expressed in terms of states and sequences of state-changing operations.

Example:

```plaintext
i := 0;
s := 0;
while i < length(A)
    do s := s+A[i];
        i := i+1
    od
```
Imperative models

• Imperative models of computations are expressed in terms of states and sequences of state-changing operations

Example:

\[
\begin{align*}
i &:= 0; \\
s &:= 0; \\
\text{while } i < \text{length}(A) &
\begin{align*}
do \quad s &:= s + A[i]; \\
i &:= i + 1
\end{align*}
\od
\end{align*}
\]

An imperative model describes how a solution is obtained
Object-oriented models

- An object is characterized by a state and an interface specifying a collection of state-changing operations.
- Object-oriented models of computations are expressed in terms of a collection of objects which exchange messages by using interface operations.
Object-oriented models

- An **object** is characterized by a **state** and an **interface** specifying a collection of **state-changing operations**.

- **Object-oriented models of computations** are expressed in terms of a collection of objects which exchange messages by using interface operations.

Object-oriented models add structure to imperative models

An object-oriented model describes *how* a solution is obtained
Declarative models

• In declarative models focus is on *what* the solution is.
Declarative models

- In **declarative models** focus is on *what* the solution is.

Some examples

- \( s = \sum_{i=0}^{\text{length}(A) - 1} A_i \)
- Queries on relational databases (*select ⋯ from ⋯ where \( \phi \)*)
- \( \Box (\text{Press} \Rightarrow \Diamond_{\leq 5} \text{DoorOpen}) \)
- \( \text{man}(\text{Socrates}) \land \forall x. (\text{man}(x) \Rightarrow \text{mortal}(x)) \)
- \[
\begin{aligned}
\text{Register} &= \text{ArticleCode} \to (\text{Name} \times \text{Price}) \\
\text{Purchase} &= \text{ArticleCode}^* \\
\text{Bill} &= \ldots \\
\text{makeBill} &\colon \text{Purchase} \ast \text{Register} \to \text{Bill}
\end{aligned}
\]
Declarative models

• In **declarative models** focus is on *what* the solution is.

Some examples

• \( s = \sum_{i=0}^{\text{length}(A)-1} A_i \)

• Queries on relational databases (**select** \( \cdots \) **from** \( \cdots \) **where** \( \phi \))

• \( \Box (\text{Press} \Rightarrow \Diamond_{\leq 5} \text{DoorOpen}) \) **temporal logic with time**

• \( \text{man}(\text{Socrates}) \land \forall x. (\text{man}(x) \Rightarrow \text{mortal}(x)) \) **first-order logic**

\[
\begin{align*}
\text{Register} &= \text{ArticleCode} \rightarrow (\text{Name} \times \text{Price}) \\
\text{Purchase} &= \text{ArticleCode}^* \\
\text{Bill} &= \ldots \\
\text{makeBill} : \text{Purchase}^* \text{Register} \rightarrow \text{Bill}
\end{align*}
\]

No (explicit) notion of a state and state-changing operations
Declarative modelling

Focus on what allows you to

• describe ideas, concepts, designs, constructions, etc. succinctly at a high level of abstraction

Formal specification languages based on mathematics and logic support declarative modelling.

B, Z, VDM, RAISE, TLA (to mention a few)

Such specifications are (in general) not executable.
Declarative programming

— logic programming or functional programming.

- In logic programming languages, programs are (typically) expressed in a fragment of first-order logic. The formulas has a standard declarative meaning, as well as a procedural interpretation based on logical inferences.

- In functional programming languages, a program is expressed as a mathematical function $f : A \rightarrow B$, and evaluations of function applications guides the computations.
Declarative programming

— logic programming or functional programming.

- In logic programming languages, programs are (typically) expressed in a fragment of first-order logic. The formulas has a standard declarative meaning, as well as a procedural interpretation based on logical inferences.

- In functional programming languages, a program is expressed as a mathematical function $f : A \rightarrow B$, and evaluations of function applications guides the computations.

Some advantages

- executable models
- fast prototyping
- more advanced applications are within reach
- good supplement of modelling and problem solving techniques
Overview of the course

Major parts:

1. Modelling and programming using
   - the functional programming language SML, and
   - the logical programming language Prolog.

2. Program correctness.

Homepage for the course: www.imm.dtu.dk/courses/02153