Introduction to Unity

This part of the course introduces the specificationimplemmentation language Unity.

The Unity approach to program development suggests that we start with a very high-level description of the program. We then proceed by a sequence of verified program transformations to derive more efficient versions of the program. Finally, we map the program on a variety of architectures, such as asynchronous shared-memory, synchronous shared-memory, distributed architectures, synchronous circuits, and synchronous circuits (speed independent).

All the intermediate levels are described by the same language. The final mapping to architectures can be done by restriction to schemas specific to each architecture. It can also done by mapping to native languages such as C and RTL.

A special temporal programming logic is developed and used for validating the correctness of the program transformations.
Example: Scheduling a Meeting

Assume 3 persons: $F$, $G$, and $H$ who wish to coordinate a time for a meeting. Their time constraints are given by functions $g, f, h : \mathbb{N} \mapsto \mathbb{N}$.

For a given $t \in \mathbb{N}$, $f(t)$ is the earliest time not earlier than $t$, at which $F$ is ready to meet. Similarly for $G, H$, and their scheduling functions $g, h$.

Introduce the abbreviation

$$com(t) \equiv [t = f(t) = g(t) = h(t)]$$

characterizing a time at which all associates are ready to meet. We assume that there exists a $z : \mathbb{N}$ such that $com(z)$ holds.
Problem Specification

Variables $r, t, z$ appearing in the specification range over the naturals.

Given monotone nondecreasing natural functions $f, g, h$ where, for all $t$

\[
\begin{align*}
    f(t) &\geq t \land g(t) \geq t \land h(t) \geq t \\
    f(f(t)) &= f(t) \land g(g(t)) = g(t) \land h(h(t)) = h(t)
\end{align*}
\]

and given $z$ satisfying $\text{com}(z)$, design a program which, on termination, yields a value $r$, satisfying

\[
r = \min\{t \mid \text{com}(t)\}
\]
First Program

A top level version of a Unity program for this task can be given as follows:

Program $P_1$

assign \( r := \min \{ u \mid 0 \leq u \leq z \land com(u) \} \)

end $P_1$

This program is obviously correct. High-level as it is, the program is still executable. For example, mapped on a von Neumann architecture, we can implement it by a loop over \( u \), starting at 0 and iterating until we reach a \( u \) satisfying \( com(u) \).
A More Efficient Program

A better program can be presented as follows:

Program $P^2$

initially $r = 0$

assign $r := f(r)$ \[\square r := g(r)\] \[\square r := h(r)\]

end $P^2$

The meaning of such a program is that we start with an initial state, satisfying the initially requirement. Subsequently, at each step we non-deterministically choose one of the assignments and execute it. Each assignment should be chosen infinitely many times (weak fairness).

The program never officially terminates. However, it may reach a fixed-point state. This is a state $s$ which remains invariant under the execution of any assignment in the assign section.

The claim about program $P^2$ is that it eventually reaches a fixed point and, when it does, $r$ is the minimal value satisfying $\text{com}(r)$. 
Proof of Correctness

Here we wish to prove that $P_2$ refines $P_1$. This consists of two claims:

1. If $s$ is a fixed-point state of $P_2$, then $s \models com(r) \land \forall t : t < r :: \neg com(t)$.

2. $P_2$ eventually reaches a fixed point.

Since we do not have a proof system yet, our proof will be informal, yet reflect the main approaches to such proofs.
Proving Claim 1 by Invariants

Claim 1 requires showing that, if \( s_f \) is a fixed-point state, then it satisfies

\[
\text{com}(r) \land \forall t: t < r :: \neg \text{com}(t)
\]

We start by showing that the following assertion is an invariant of the program:

\[
\varphi: \forall t: t < r :: \neg \text{com}(t)
\]

This is established by showing that \( \varphi \) holds for the initial state and then, that every \( P2 \)-successor of a \( \varphi \)-state is also a \( \varphi \)-state.

The initial state satisfies \( r = 0 \) which implies \( \varphi \) due to the fact that there does not exist a natural \( t \) such that \( t < 0 \).

As to the induction step, assume that \( s \) is a \( \varphi \)-state and \( s' \) is its \( P2 \) successor. Denoting the value of \( r \) in \( s' \) by \( r' \), we know that either \( r' = f(r) \), or \( r' = g(r) \) or \( r' = h(r) \). We will consider the case \( r' = f(r) \). It is therefore necessary to prove

\[
\forall t: t < r :: \neg \text{com}(t) \rightarrow \forall t': t' < f(r) :: \neg \text{com}(t')
\]

To prove this, it is sufficient to show \( \forall t: r \leq t < f(r) :: \neg \text{com}(t) \), which is a direct consequence of the monotonicity of \( f \).

Due to the invariance of \( \varphi \), state \( s_f \) satisfies \( \forall t: t < r :: \neg \text{com}(t) \). Denoting \( r_f = s_f[r] \), the fact that \( s_f \) is a fixed point implies \( r_f = f(r_f) = g(r_f) = h(r_f) \) leading to \( \text{com}(r_f) \).