1. Reading: K. Rosen *Discrete Mathematics and Its Applications*, 3.5
2. The main message of this lecture:

> **A syntactic correctness of a program is easy to verify (compilers do it). A semantic correctness stating that the program will be producing the right output is impossible to verify automatically. Semantic correctness usually consists of two parts: a partial correctness stating that the correct answer is obtained if the program terminates, and a proof that the program always terminates.**

Proving **semantical correctness** automatically is impossible.

Theorem 16.1. *There is no algorithm to decide, given any program P and input x, whether P will eventually halt.*

Proof. Imagine an operating system Φ capable of compiling and executing any program P on any input y. Since every input is a string of characters in a special input alphabet, we may regards Φ as a function of two string arguments $x =$ the code of a program P, and $y =$ an input of P such that for any given x $\Phi(x, y) \cong P(y)$. Here $\cong$ means that both parts are simultaneously defined or not defined, and if they are defined their values coincide. Suppose the halting problem is decidable, then there is a computable function $f(x)$ from strings to $\{0, 1\}$:

$$f(x) = \begin{cases} 0, & \text{if } \Phi(x,x) \text{ halts} \\ 1, & \text{if } \Phi(x,x) \text{ does not halt} \end{cases}$$

Consider another function $g(x)$ such that

$$g(x) = \begin{cases} 0, & \text{if } f(x) = 1, \\ \textit{loops forever}, & \text{if } f(x) = 0 \end{cases}$$

Here is a description of a program that computes g: "Given x run $f(x)$. If $f(x) = 1$, print 0 and halt. Otherwise, loop forever." Let i be a program code for g. Then

$$\Phi(i,i) \textit{ halts} \quad \Leftrightarrow \quad g(i) = 0 \quad \Leftrightarrow \quad f(i) = 1 \quad \Leftrightarrow \quad \Phi(i,i) \textit{ does not halt},$$

which contradicts the assumption that the halting problem is decidable.

Definition 16.2. A program segment S is said to be **partially correct** with respect to the **initial assertion** p and the **final assertion** q if whenever p holds for the initial values of S and S terminates, then q holds for the output values of S. Notation: $p\{S\}q$ is also known as the **Hoare implication**[1]

Example 16.3. Show that the program segment S $\begin{array}{l} y := 2 \\ z = x + y \end{array}$ is correct with respect to the initial assertion $x = 3$ and the final assertion $z = 5$.

Termination is obvious since there are not loops there and the execution stops after performing the assignments. The partial correctness means that the Hoare implication $p\{S\}q$ holds in this

[1]Tony Hoare, an Oxford professor, was recently knighted by the British Queen.

case. Note, that S in fact depends on x, therefore $S = S(x)$, and a more proper notation for the Hoare implication would be $p\{S(p)\}q$. A simple analysis of the assignments made by S immediately convinces us that if p (i.e. $x = 3$) holds before S has been executed, then q (i.e. $z = 5$) holds afterwards. Therefore, $p\{S\}q$ is true.

A general strategy of proving correctness of a program P is **decomposing** P into smaller manageable segments $S_1, S_2, S_3 \ldots$ corresponding to elementary programmistic steps: assignments, loops, conditional branching, etc. and then proving every segment S_i correct for corresponding matching initial and final assertions. Each elementary segment is treated by its own **inference rule** for the Hoare implication.

Definition 16.4. The **composition rule** covers the task of agreeing correctness proofs of two consecutive segments S_1 and S_2. Notation: $S_1 : S_2$ stands for the composite segment consisting of S_1 followed by S_2. Suppose also that we have already established individual correctness of S_1 and S_2 with respect to *matching conditions*, i.e. $p\{S_1\}q$ and $q\{S_2\}r$ both hold. Then

$$\frac{\begin{array}{c} p\{S_1\}q \\ q\{S_2\}r \end{array}}{p\{S_1 : S_2\}r}$$

Definition 16.5. A program segment "**if** *condition* **then** S" is treated by the inference rule

$$\frac{\begin{array}{c} (p \wedge \mathit{condition})\{S\}q \\ (p \wedge \neg \mathit{condition}) \rightarrow q \end{array}}{p\{\textbf{if } \mathit{condition} \textbf{ then } \mathrm{S}\}q}$$

Example 16.6. Show that "**if** $x > y$ **then** $y := x$" is correct with respect to the initial assertion $\mathbf{T}$ (i.e. *true*) and the final assertion $y \geq x$. Again, termination is obvious. According to the standard semantics of "**if** ... **then**" operator, both premises of the rule 16.5 take place, i.e. $(\mathbf{T} \wedge x > y)\{y := x\}y \geq x$ and $(\mathbf{T} \wedge \neg x > y) \rightarrow y \geq x$. Therefore we can conclude $\mathbf{T}\{\textbf{if } x > y \textbf{ then } y := x\}y \geq x$.

Definition 16.7. A program segment "**if** *condition* **then** S_1 **else** S_2" is treated by the inference rule

$$\frac{\begin{array}{c} (p \wedge \mathit{condition})\{S_1\}q \\ (p \wedge \neg \mathit{condition})\{S_2\}q \end{array}}{p\{\textbf{if } \mathit{condition} \textbf{ then } S_1 \textbf{ else} S_2\}q}$$

Example 16.8. Show that "**if** $x < 0$ **then** $abs := -x$ **else** $abs := x$" is correct with respect to the initial assertion $\mathbf{T}$ and the final assertion $abs = |x|$. Termination is obvious. For the partial correctness check the premises of the rule 16.7. $(\mathbf{T} \wedge x < 0)\{abs := -x\}abs = |x|$ and $(\mathbf{T} \wedge x \geq 0)\{abs := x\}abs = |x|$ both hold, therefore, by 16.7, the corresponding Hoare implication $\mathbf{T}\{\textbf{if } x < 0 \textbf{ then } abs := -x \textbf{ else } abs := x\}(abs = |x|)$ is true.

Definition 16.9. Partial correctness of a loop segment "**while** *condition* S" is proven by induction on the counter i. The induction proposition $p(i)$ is called a **loop invariant**. The rule of inference is

$$\frac{(p \wedge \mathit{condition})\{S\}p}{p\{\textbf{while } \mathit{condition}\ S\}(\neg \mathit{condition} \wedge p)}$$

Example 16.10. The *factorial* example from Section 3.5.

Homework assignments. (due Friday 03/02).

16A:Rosen3.5-2; 16B:Rosen3.5-4; 16C:Rosen3.5-12.