Part II. Hoare Logic and Program Verification

Dilian Gurov

Part II. Hoare Logic and **Program Verification**

Props:	safety
Models:	source
Specs:	logic as
Method:	Hoare
Tool:	VeriFa

of data manipulation code ssertions logic, VCG ist.

Specification and Verification

- Program specification
 - states program correctness, formally
 - relates properties of states before och after the execution
- Program verification
 - proves program correctness, formally
 - relative to a specification

Why specify programs?

- Good for documentation: capture unambiguously what the program should do (and not how)
- Programs annotated with specs can be fed into static checkers
- However, specifications:
 - require expertise and time
 - can get large and difficult to handle

Why verify programs?

- Testing can only find errors, but cannot prove their absense
- However, verification is expensive:
 - requires formal specs
 - requires expertise and time
 - faces decidability and complexity issues
- Therefore:
 - use light-weight tools for critical parts

Code Verification

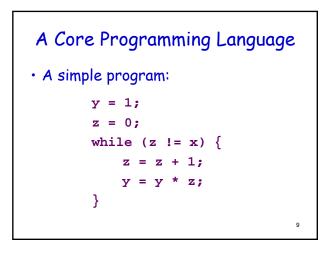
- Code verification
 - the code itself is the model!
 - i.e., no abstraction...
 - ...but must still be based on a formal description of execution: a formal semantics of the programming language

Semantics of Programming Languages

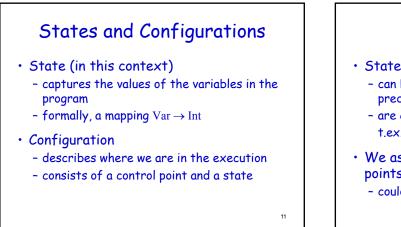
- Semantics
 - a formal definition of how programs execute
 - can be given in various ways
 - see course DD2457

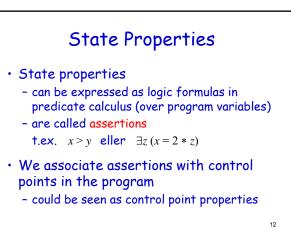
Semantics of Programming Languages

- Natural semantics
 - relates states before and after the execution
 - is defined inductively on the program structure
 - -here: informal understanding



Formal Syntax • Arithmetic expressions E ::= n | x | (E+E) | (E-E) | (E * E)• Boolean expressions B ::= true | false | (E < E) | (B | B) | (B | B)• Commands $C ::= x = E | C; C | if B \{C\} else \{C\} | while B \{C\}$





Program Specification

- Can be accomplished with two assertions: - precondition
 - postcondition
- Formal notation: Hoare tripples

 $\left<\phi\right>P\left<\psi\right>$

read (roughly): if execution of program P starts in a state where precondition ϕ holds, then the execution ends in a state where postcondition ψ holds

13

Specification Example

• The factorial program *Fac1*

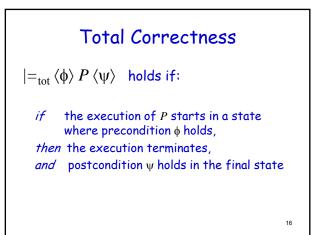
y = 1; z = 0; while (z != x) { z = z + 1; y = y * z; }

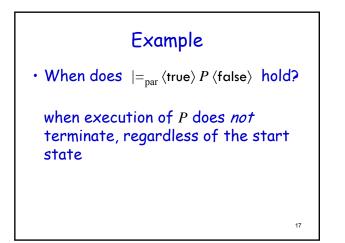
can be specified with the Hoare tripple

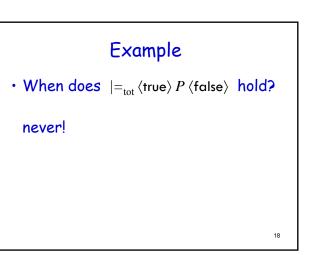
$$\langle x \ge 0 \rangle$$
 Fac1 $\langle y = x! \rangle$

14

Partial Correctness $|=_{par} \langle \phi \rangle P \langle \psi \rangle \text{ holds if:}$ *if* the execution of *P* starts in a state where precondition ϕ holds, *and* the execution terminates, *then* postcondition ψ holds in the final state







Logic Variables

```
• Can the factorial program Fac2

y = 1;

while (x != 0) {

y = y * x;

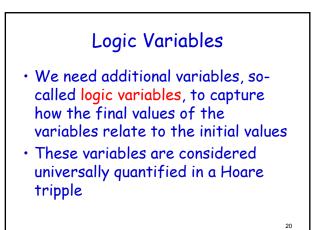
x = x - 1;

}

be specified with the Hoare tripple

\langle x \ge 0 \rangle Fac2 \langle y = x! \rangle
```

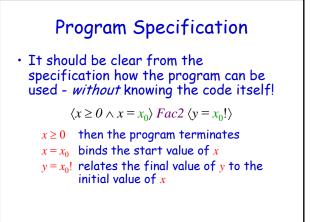
19

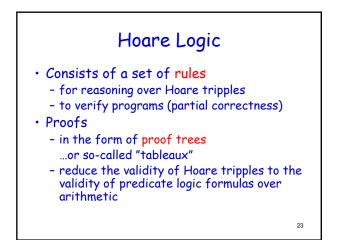


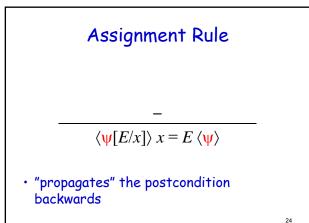
Logic Variables
• The factorial program Fac2

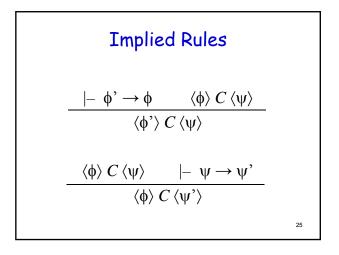
$$y = 1;$$

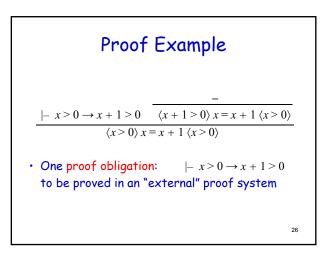
while $(x = 0) \{$
 $y = y * x;$
 $x = x - 1;$
}
can be specified with the Hoare tripple
 $\langle x \ge 0 \land x = x_0 \rangle$ Fac2 $\langle y = x_0! \rangle$

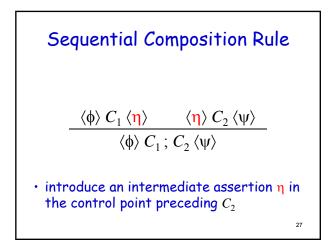


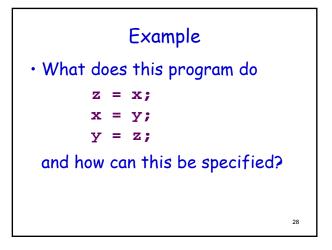


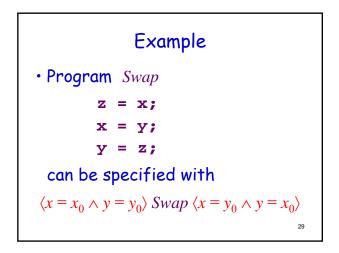


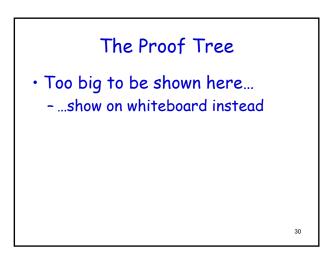










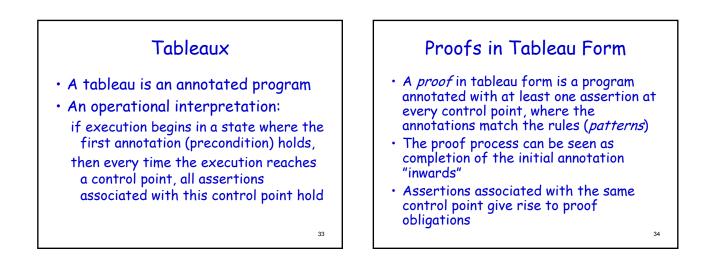


Proof Tableaux

- Alternative presentation:
 - as tableau
 - correctness proofs can be presented as commented (or annotated) programs, where the comments are assertions associated with control points

The Proof in Tableau Form

32



31

