Proof of program correctness

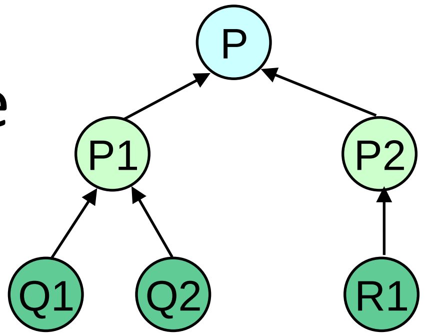
Istvan Majzik
majzik@mit.bme.hu

Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Proof of correctness for structured programs

Proving correctness for structured programs

- "Composition" of properties:
  - If a program P consists of syntactic units P_1 and P_2 then the properties of P can be derived on the basis of the properties of the syntactic units P_1 and P_2
  - The principle of structural induction



- Structured programs: PLW language

P::= x:=e | skip | P_1; P_2 | if B then P_1 else P_2 fi | while B do P od

- Example (positive integer division):

P_{div}: r:=x; q:=0; while r≥y do r:=r-y; q:=q+1 od

Operational semantics of PLW

- Configuration: $C_i=(P_i, \sigma_i)$ where
  - P_i is the syntactic continuation (E denotes empty cont.)
  - σ_i is the observable state (variables)
- Transition relation: $C \to C'$
  - $(x:=e, \sigma) \to (E, \sigma[e/x])$
  - $(\text{skip}, \sigma) \to (E, \sigma)$
  - $(P_1; P_2, \sigma) \to (P_1'; P_2, \sigma')$ if $(P_1, \sigma) \to (P_1', \sigma')$
  - $(\text{if } B \text{ then } P_1 \text{ else } P_2 \text{ fi}, \sigma) \to (P_1, \sigma)$ if $\sigma[B]=\text{true}$
    $\to (P_2, \sigma)$ if $\sigma[B]=\text{false}$
  - $(\text{while } B \text{ do } P \text{ od}, \sigma) \to (P; \text{while } B \text{ do } P \text{ od}, \sigma)$ if $\sigma[B]=\text{true}$
    $\to (E, \sigma)$ if $\sigma[B]=\text{false}$

Here $E;P \equiv P$ is applied at the end

D deduction system for proving partial correctness (1)

- Axioms:
  - ASS: $\{p[e/x]\}\ x:=e\ \{p\}$
  - SKIP: $\{p\}\ \text{skip}\ \{p\}$

p holds as postcondition, if p[e/x] holds as precondition

- Rules for the syntactic constructs:

Rule format:

$$\frac{\text{Condition}}{\text{Consequence}}$$

  - SEQ:

$$\frac{\{p\}\ P_1\ \{r\}\ \text{ and }\ \{r\}\ P_2\ \{q\}}{\{p\}\ P_1;\ P_2\ \{q\}}$$

  - COND:

$$\frac{\{p\wedge B\}\ P_1\ \{q\}\ \text{ and }\ \{p\wedge\neg B\}\ P_2\ \{q\}}{\{p\}\ \text{if } B \text{ then } P_1 \text{ else } P_2 \text{ fi}\ \{q\}}$$

  - REP:

$$\frac{\{p\wedge B\}\ P\ \{p\}}{\{p\}\ \text{while } B \text{ do } P \text{ od}\ \{p\wedge\neg B\}}$$

p is a loop invariant

D deduction system for proving partial correctness (2)

- General rules:
  - CONS: $$\frac{p \Rightarrow p_1 \ \text{ and } \{p_1\}\ P\ \{q_1\} \ \text{ and } q_1 \Rightarrow q}{\{p\}\ P\ \{q\}}$$
    Strengthening precondition and weakening postcondition
  - AND: $$\frac{\{p\}\ P\ \{q_1\} \quad \text{and} \quad \{p\}\ P\ \{q_2\}}{\{p\}\ P\ \{q_1 \wedge q_2\}}$$
    Separated proof of conjunctive postcondition
  - OR: $$\frac{\{p_1\}\ P\ \{q\} \quad \text{and} \quad \{p_2\}\ P\ \{q\}}{\{p_1 \vee p_2\}\ P\ \{q\}}$$
    Separating cases of disjunctive precondition
- Domain axioms and rules:
  To be included in the deduction system

Example: Proving partial correctness

{x≥0∧y≥0} r:=x; q:=0; while r≥y do r:=r-y; q:=q+1 od {x=q·y+r ∧ 0≤r<y}

1. $\{x = 0 \cdot y + x \wedge x \geq 0\}\ q := 0\ \{x = q \cdot y + x \wedge x \geq 0\}$ (ASS)
2. $\{x = q \cdot y + x \wedge x \geq 0\}\ r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (ASS)
3. $\{x = 0 \cdot y + x \wedge x \geq 0\}\ q := 0; r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (1)(2)(SEQ)
4. $x \geq 0 \wedge y \geq 0 \Rightarrow x = 0 \cdot y + x \wedge x \geq 0$ (ARITHMETIC)
5. $\{x \geq 0 \wedge y \geq 0\}\ q := 0; r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (3)(4)(CONS)
6. $\{x = (q+1) \cdot y + r - y \wedge r - y \geq 0\}\ r := r - y\ \{x = (q+1) \cdot y + r \wedge r \geq 0\}$ (ASS)
7. $\{x = (q+1) \cdot y + r \wedge r \geq 0\}\ q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (ASS)
8. $\{x = (q+1) \cdot y + r - y \wedge r - y \geq 0\}\ r := r - y; q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (6)(7)(SEQ)
9. $x := q \cdot y + r \wedge r \geq 0 \wedge r \geq y \Rightarrow x = (q+1) \cdot y + r - y \wedge r - y \geq 0$ (ARITHMETIC)
10. $\{x = q \cdot y + r \wedge r \geq 0 \wedge r \geq y\}\ r := r - y; q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (8)(9)(CONS)
11. $\{x = q \cdot y + r \wedge r \geq 0\}$ while $r \geq y$ do $r := r - y; q := q + 1$ od $\{x = q \cdot y + r \wedge 0 \leq r < y\}$ (10)(REP)
12. $\{x \geq 0 \wedge y \geq 0\}\ q := 0; r := x;$ while $r \geq y$ do $r := r - y; q := q + 1$ od $\{x = q \cdot y + r \wedge 0 \leq r < y\}$ (5)(11)(SEQ)

D* deduction system for proving correctness

- Goal: Proving termination of loops
  - while B do P od constructs
- Basic idea: Parametric assertions
  - Parameters from well-founded set
  - E.g., selecting n natural number: arithmetic extension of the specification language is needed
  - pi(x,n) parameterized loop invariant
- Modified REP rules for proving correctness:

Loop invariant, decreases n

  - REP*:

$$\frac{pi(\underline{x},n) \Rightarrow B \ \text{ and } \ \langle pi(\underline{x},n)\rangle P \langle pi(\underline{x},n-1)\rangle \ \text{ and } \ pi(\underline{x},0) \Rightarrow \neg B}{\langle \exists n{:}pi(\underline{x},n)\rangle \ \text{while B do P od} \ \langle pi(\underline{x},0)\rangle}$$

  - All other rules are the same, writing <...> instead of {...}

Properties of the deduction system

- Notation for the proof of a statement C: $Tr_I \vdash_D C$ where
  - I domain, Tr_I the axioms and deduction rules of the domain
  - D the deduction system
- Properties:
  - The correctness of D defined above can be proven
    - $Tr_I \vdash_D \{p\}P\{q\}$ results in $\models_I \{p\}P\{q\}$
  - The completeness of D cannot be proven:
    - If the axioms and rules of the domain are complex enough (e.g., contain the arithmetic of natural numbers): Gödel's first incompleteness theorem holds, i.e., there are statements that are not provable
- Practical implementation:
  - The semantics of the programming language (syntactic constructs) have to be mapped to axioms and rules
  - The theorem prover shall include the axioms and rules of the domain
  - Strategy (or search) is needed for selecting proper domain rules
  - The specification language shall be expressive enough

Summary

- For low-level flow languages:
  - Partial correctness for loop-free programs
    - Backward computational induction
  - Partial correctness for programs with loops
    - Inductive assertions
  - Correctness for programs with loops: Proving termination
    - Inductive assertions with a decreasing parameter from a well-founded set
- Structured languages (while programs):
  - Partial correctness:
    - Deduction system with structural induction
  - Correctness:
    - Deduction system with parameterized inductive assertions
    - Arithmetic extension to have a well-founded set

Proving program correctness in practice

Classic examples:

- Spec# Programming System: C# extension
  - Preconditions, postconditions (for methods) can be specified
  - Object level invariants (e.g., ranges for variables) can be given
  - Boogie2: To prove postconditions in an automated way
- JML: Java Modelling Language
  - Preconditions, postconditions, invariants can be specified
  - ESC/Java2: Proof of postconditions for a JML subset
- SPARK: Ada language subset
  - Proof by using an interactive theorem prover
- B method: Specific modelling language and approach
  - B4Free, Rodin: The derivation of verification conditions (to be proven) and theorem proving are automated (with some manual support)
  - Focus: Consistent refinement of a specification