# Specification Driven Software Design

## Seminar

**Robert L. Baber**
**Department of Computing and Software**
**McMaster University**
**Hamilton, ON, Canada**

**2002 May**

# Overview

1. Introduction
2. Design example
    2.1. relevant proof rules
    2.2. specification
    2.3. design
3. Observations and conclusions

# Introduction

Lemmata for proving correctness of a program (“proof rules”)
$\rightarrow$ design guidelines

Design goal: thesis of the relevant proof rule
subgoal: hypotheses of the rule

Iterate down to low level expressions in program

Underlying model of program execution:
program variable: a triple (name, set, value)
data environment: sequence of program variables
program statement: maps a d.e. to a d.e.

**Rules relevant for this example:** assignment, declare

assignment: $\{P_E^x\}$ x:=E {P}

also for declare: $\{P_E^x\}$ declare (x, S, E) {P}

(provided P does not refer to the set associated with x)

**Rules relevant for this example:** if

If

{V and B} S1 {P} and
{V and not B} S2 {P}

then

{V} if B then S1 else S2 endif {P}

When attempting {V} S? {P} leads to {V and B} S1 {P}, embed S1 in an if statement

and then design for {V and not B} S2? {P}.

**Rules relevant for this example:** initialized while loop

If

{V} init {I} and
{I and B} S {I} and
I and not B $\Rightarrow$ P [equivalently, I and not P $\Rightarrow$ B]

then

{V} init; while B do S endwhile {P}

Main design decision is I (generalization of V, P).

Rule decomposes design task into three **independent** subtasks: init?, B?, S?.

S must also make progress toward termination,
i.e. P true, B false.

## Design example

**Correctness proposition 1:** {V} twopartition {P}, where V and P are as follows:

V: $n \in \mathbf{Z}$ and $0 \leq n$

P: $n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $0 \leq gr \leq n$

$$\text{and}_{i=1}^{gr} \text{ prop}(X(Ptr(i))) \text{ and}_{i=gr+1}^{n} \text{ not prop}(X(Ptr(i)))$$

$$\text{and } (\&_{i=1}^{n} [Ptr(i)]) \text{ Perm } (\&_{i=1}^{n} [i])$$

**Correctness proposition 2:** For all data environments d in the domain of twopartition,

twopartition.d = [(gr, **Z**, ...)] & d

except for the values of gr and of the array elements Ptr(i) for i = 1, ... n.

**Correctness proposition 3:** The execution of the subprogram twopartition terminates.

Form of P indicates loop as main structure for program.

**The loop invariant:**

I: $n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $j \in \mathbf{Z}$ and $0 \leq gr \leq j \leq n$

$$\text{and}_{i=1}^{gr} \text{prop}(X(Ptr(i))) \text{ and}_{i=j+1}^{n} \text{not prop}(X(Ptr(i)))$$

$$\text{and } (\&_{i=1}^{gr} [Ptr(i)] \&_{i=j+1}^{n} [Ptr(i)]) \text{ Perm } (\&_{i=1}^{n+gr-j} [i])$$

**The initialization of the loop:** {V} init? {I}

V: $n \in \mathbf{Z}$ and $0 \leq n$

I: $n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $j \in \mathbf{Z}$ and $0 \leq gr \leq j \leq n$

$$\text{and}_{i=1}^{gr} \text{ prop}(X(Ptr(i))) \text{ and}_{i=j+1}^{n} \text{ not prop}(X(Ptr(i)))$$

$$\text{and } (\&_{i=1}^{gr} [Ptr(i)] \&_{i=j+1}^{n} [Ptr(i)]) \text{ Perm } (\&_{i=1}^{n+gr-j} [i])$$

$$V = [I_0^{gr}]_n^j$$

**init:** declare (j, **Z**, n); declare (gr, **Z**, 0)

**The loop condition:** I and not B? $\Rightarrow$ P,
I and not P $\Rightarrow$ B?

I: $n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $j \in \mathbf{Z}$ and $0 \le gr \le j \le n$

$$\text{and}_{i=1}^{gr} \text{prop}(X(Ptr(i))) \text{ and}_{i=j+1}^{n} \text{not prop}(X(Ptr(i)))$$

$$\text{and } (\&_{i=1}^{gr} [Ptr(i)] \; \&_{i=j+1}^{n} [Ptr(i)]) \text{ Perm } (\&_{i=1}^{n+gr-j} [i])$$

P: $n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $0 \le gr \le n$

$$\text{and}_{i=1}^{gr} \text{prop}(X(Ptr(i))) \text{ and}_{i=gr+1}^{n} \text{not prop}(X(Ptr(i)))$$

$$\text{and } (\&_{i=1}^{n} [Ptr(i)]) \text{ Perm } (\&_{i=1}^{n} [i])$$

# The loop condition

I and gr=j $\Rightarrow$ P

also, I and gr$\geq$j $\Rightarrow$ P

alternatively: I and not P $\Rightarrow$ gr$\neq$j

I and not P $\Rightarrow$ gr<j

B: gr$\neq$j

or

B: gr<j [stronger, termination possibly easier]

**The body of the loop:** {I and B} S? {I}

S must

- progress toward termination, i.e. making gr<j false
- keep I true

Two candidates, increase gr or decrease j:

{I and B} S1?; gr:=gr+1 {I and 1≤gr} {I}

{I and B} S2?; j:=j-1 {I and j≤n-1} {I}

or equivalently,

$$\{I \text{ and } B\}\ S1?\ \{[I \text{ and } 1 \leq gr]_{gr+1}^{gr}\}$$

$$\{I \text{ and } B\}\ S2?\ \{[I \text{ and } j \leq n-1]_{j-1}^{j}\}$$

Examine the three pre- and postconditions:

$I \text{ and } B$

$[I \text{ and } 1 \leq gr]_{gr+1}^{gr}$

$[I \text{ and } j \leq n-1]_{j-1}^{j}$

Precondition in both design tasks:

I and B

=

$n \in \mathbf{Z}$ and $gr \in \mathbf{Z}$ and $j \in \mathbf{Z}$ and $0 \le gr < j \le n$ [< vs. ≤]

$\text{and}_{i=1}^{gr}$ prop(X(Ptr(i))) $\text{and}_{i=j+1}^{n}$ not prop(X(Ptr(i)))

and $(\&_{i=1}^{gr}$ [Ptr(i)] $\&_{i=j+1}^{n}$ [Ptr(i)]) Perm $(\&_{i=1}^{n+gr-j}$ [i])

Postcondition of S1:

$$[I \text{ and } 1 \leq gr]_{gr+1}^{gr}$$

=

$$n \in \mathbf{Z} \text{ and } gr \in \mathbf{Z} \text{ and } j \in \mathbf{Z} \text{ and } 0 \leq gr < j \leq n$$

$$\text{and}_{i=1}^{gr} \text{ prop}(X(Ptr(i))) \text{ and}_{i=j+1}^{n} \text{ not prop}(X(Ptr(i)))$$

$$\boxed{\text{and prop}(X(Ptr(gr+1)))}$$

$$\text{and } (\&_{i=1}^{gr} [Ptr(i)] \boxed{\& [Ptr(gr+1)]} \&_{i=j+1}^{n} [Ptr(i)])$$

$$\text{Perm } (\&_{i=1}^{n+gr-j} [i] \boxed{\& [n+gr+1-j]})$$

I.e.

$$I \text{ and } B \text{ and } prop(X(n+gr+1-j))$$

=

$$[[I \text{ and } 1 \leq gr]_{gr+1}^{gr}]_{n+gr+1-j}^{Ptr(gr+1)}$$

So

$$\{I \text{ and } B \text{ and } prop(X(n+gr+1-j))\}$$

$$Ptr(gr+1):=n+gr+1-j;\ gr:=gr+1$$

$$\{I \text{ and } 1 \leq gr\}$$

Suggesting for S

{I and B}
if prop(X(n+gr+1-j))
then Ptr(gr+1):=n+gr+1-j; gr:=gr+1
else S3? endif
{I}

Where

{I and B and not prop(X(n+gr+1-j))} S3? {I}

Postcondition of S2:

$$[I \text{ and } j \leq n-1]^{j}_{j-1}$$

=

$$n \in \mathbf{Z} \text{ and } gr \in \mathbf{Z} \text{ and } j \in \mathbf{Z} \text{ and } 0 \leq gr < j \leq n$$

$$\text{and}_{i=1}^{gr} \text{ prop}(X(Ptr(i))) \text{ and}_{i=j+1}^{n} \text{ not prop}(X(Ptr(i)))$$

$$\boxed{\text{and not prop}(X(Ptr(j)))}$$

$$\text{and } (\&_{i=1}^{gr} [Ptr(i)] \boxed{\& [Ptr(j)]} \&_{i=j+1}^{n} [Ptr(i)])$$

$$\text{Perm } (\&_{i=1}^{n+gr-j} [i] \boxed{\& [n+gr+1-j]})$$

I.e.

$$I \text{ and } B \text{ and not } prop(X(n+gr+1-j))$$

=

$$[[I \text{ and } j \leq n-1]_{j-1}^{j}]_{n+gr+1-j}^{Ptr(j)}$$

So

$$\{I \text{ and } B \text{ and not } prop(X(n+gr+1-j))\}$$

$$Ptr(j) := n+gr+1-j;\ j := j-1$$

$$\{I \text{ and } j \leq n-1\}$$

suitable for the else part of the if statement

The complete subprogram twopartition then becomes:

```
declare (j, Z, n); declare (gr, Z, 0)
while gr<j do
    if prop(X(n+gr+1-j))
    then Ptr(gr+1):=n+gr+1-j; gr:=gr+1
    else Ptr(j):=n+gr+1-j; j:=j-1
    endif
endwhile
release j                    [cf. Correctness Proposition 2]
```

Most expressions lifted out of the correctness proof.

## Observations and conclusions

- Design from specification, not subjective description or understanding
- to understand, look at specification, not code
- Code needed ONLY to
  - verify executional characteristics, e.g. satisfies specification
  - control execution by computer (production, test)
- Much code can be mathematically derived, especially hard parts.