A Humble Introduction to DIJKSTRA'S "A DISCIPLINE OF PROGRAMMING"

Do-Hyung Kim

School of Computer Science and Engineering

Sungshin Women's University

CONTENTS

BIBLIOGRAPHIC INFORMATION AND ORGANIZATION OF THE BOOK (1/5)

- Library Information
  - Published in 1976, as a volume of the “Series in Automatic Computation” by Prentice-Hall, Inc.
  - Total 223 pages
    - Foreword 1 page
    - Preface 5 pages
    - Main body 217 pages

BIBLIOGRAPHIC INFORMATION AND ORGANIZATION OF THE BOOK (2/5)

- Organization of the Book
  - Foreword by C. A. R. Hoare
  - Preface by the author
  - 28 chapters, semantically grouped into three parts: I (Ch. 0 to Ch. 11), II (Ch. 12 to Ch. 25), and III (Ch. 26 & Ch. 27)

BIBLIOGRAPHIC INFORMATION AND ORGANIZATION OF THE BOOK (3/5)

- Organization of the Book (Continued)
  - Part I (Framework): Executional Abstraction, The Role of Programming Languages, States and Their Characterization, The Characterization of Semantics, The Semantic Characterization of a Programming Language, Two Theorems, On the Design of Properly Terminating Constructs, Euclid's Algorithm Revisited, The Formal Treatment of Some Small Examples, On Nondeterminacy Being Bounded, An Essay on the Notion: "The Scope of Variables", Array Variables

BIBLIOGRAPHIC INFORMATION AND ORGANIZATION OF THE BOOK (4/5)

- Organization of the Book (Continued)
  - Part II (Examples): The Linear Search Theorem, The Problem of the Next Permutation, The Problem of the Dutch National Flag, Updating a Sequential File, Merging Problems Revisited, An Exercise Attributed to R. W. Hamming, The Pattern Matching Problem, Writing a Number as the Sum of Two Squares, The Problem of the Smallest Prime Factor of a Large Number, The Problem of the Most Isolated Villages, The Problem of the Shortest Subspanning Tree, Rem's Algorithm for the Recording of Equivalence Classes, The Problem of the Convex Hull in Three Dimensions, Finding the Maximal Strong Components in a Directed Graph

BIBLIOGRAPHIC INFORMATION AND ORGANIZATION OF THE BOOK (5/5)

- Organization of the Book (Continued)
  - Part III (Summary): On Manuals and Implementations, In Retrospect
- In This Talk,
  - Focus on Part I, i.e., the framework or the (programming) methodology of the author
- Translation of the Book
  - (Still!) Ongoing by the talker
  - Currently lie in the middle of Chapter 8

INTRODUCTION AND RATIONALE (1/6)

- Why Formal Semantics?
  - Essential for the design of consistent/unambiguous languages, validation of language translators, correctness proof of programs, and (automatic?) derivation of programs
- Dijkstra's Comment on Program Testing
  - "Program testing can be quite effective for showing the *presence* of bugs, but is hopelessly inadequate for showing their *absence*."

INTRODUCTION AND RATIONALE (2/6)

- Historical Background (My Personal Retrospect :-))
  - So-called "software crisis"
  - Responses of three groups
    - The conservatives (?): D. E. Knuth, ...
    - The progressives (?): E. W. Dijkstra, D. Gries, C. A. R. Hoare, ...
    - The radicals (?): J. Backus (!), R. Kowalski, ...

INTRODUCTION AND RATIONALE (3/6)

- Definition of Axiomatic Semantics
  - Axiomatic semantics define the semantics of a program, statement, or language construct by describing the effect its execution has on "assertions" (or "predicates") about the data manipulated by the program.
  - The term "axiomatic" is used because elements of mathematical logic are used to specify the semantics of programming languages, including logical axioms.

INTRODUCTION AND RATIONALE (4/6)

- Why Dijkstra's Book?
  - Dijkstra's framework (i.e., predicate transformers) cleanly and systematically summarizes research in this field since the seminal paper of Hoare's.
  - It's really great *fun*!

INTRODUCTION AND RATIONALE (5/6)

- Pre-condition and Post-condition
  - Assertions associated with language constructs are of two kinds: assertions about things that are true just before execution of the construct and assertions about things that are true just after the execution of the construct.
  - Assertions about the situation just before execution are called *pre-conditions*, and assertions about the situation just after execution are called *post-conditions*.

INTRODUCTION AND RATIONALE (6/6)

- Programming as a "Goal-directed Activity"
  - We need a way of associating to a language construct in concern a general relation between pre-condition and post-condition.
  - The way to do this is to use the property that programming is a *goal-directed activity*: We usually know what we want to be true after the execution of a language construct, and the question is whether the known conditions before the execution will guarantee that this becomes true.

PREDICATE TRANSFORMERS (1/5)

- Weakest Pre-condition
  - The condition that characterizes the set of *all* initial states such that activation will certainly result in a properly terminating happening leaving the system in a final state satisfying a given post-condition is called "the weakest pre-condition corresponding to that post-condition."
  - Denoted by wp(S, R) where S is a system (machine, mechanism, construct) and R is the desired post-condition

PREDICATE TRANSFORMERS (2/5)

- Semantics of a Mechanism
  - Given in the form of a rule describing how for any given post-condition *R* the corresponding weakest pre-condition wp(*S*, *R*) can be derived.
  - When we ask for the definition of the semantics of a mechanism *S* , what we really ask for is such a rule for that mechanism.

D.-H. Kim, PL Lab., Sungshin W. University

PREDICATE TRANSFORMERS (3/5)

- Definition of a Predicate Transformer
  - For a fixed mechanism *S* such a rule, which is fed with the predicate *R* denoting the post-condition and delivers a predicate wp(*S*, *R*) denoting the corresponding weakest pre-condition, is called "a predicate transformer."

PREDICATE TRANSFORMERS (4/5)

- Properties of Predicate Transformers
  - Property 1 (Law of the Excluded Miracle).
    - For any mechanism S we have

      $$\text{wp}(S, F) = F$$
  - Property 2 (Monotonicity).
    - For any mechanism S and any post-condition Q and R such that

      $$Q \Rightarrow R \text{ for all states}$$

      we also have

      $$\text{wp}(S, Q) \Rightarrow \text{wp}(S, R) \text{ for all states}$$

PREDICATE TRANSFORMERS (5/5)

- Properties of Predicate Transformers (Continued)
  - Property 3 (Distributivity of Conjunction).
    - For any mechanism S and any post-conditions Q and R, we have

      $$(\text{wp}(S, Q) \wedge \text{wp}(S, R)) = \text{wp}(S, Q \wedge R)$$
  - Property 4 (Distributivity of Disjunction).
    - For any mechanism S and any post-conditions Q and R, we have

      $$(\text{wp}(S, Q) \vee \text{wp}(S, R)) \Rightarrow \text{wp}(S, Q \vee R)$$

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (1/7)

- Sapir-Whorf Hypothesis
  - A (still controversial) linguistic theory
  - "The structure of language defines the boundaries of thought."
- L. Wittgenstein
  - "The limits of my language mean the limits of my world."

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (2/7)

- Semantic Characterization of a Programming Language
  - We consider the semantic characterization of a programming language given by the set of rules that associate the corresponding predicate transformer with each program written in that language.
  - We can regard the program as "a code" for a predicate transformer.

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (3/7)

- *skip* and *abort*
  - Two very simple predicate transformers
  - *skip*
    - Identity transformer
    - Semantics:

      wp(*skip*, R) = R for any post-condition R
  - *abort*
    - Constant transformer
    - Semantics:

      wp(*abort*, R) = F for any post-condition R

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (4/7)

- Assignment Statement
  - Substitution transformer
  - Syntax: "x := E"
  - Semantics:

    wp("x := E", R) = $R_{E \rightarrow x}$ for any post-condition R

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (5/7)

- Composition of Statements
  - <mechanism> ::= <primitive mechanism> |
    proper composition of <mechanism>'s>
  - "Functional composition" transformer
  - Syntax: "*S*1; *S*2"
  - Semantics:

    wp("*S*1; *S*2", *R*) = wp(*S*1, wp(*S*2, *R*))

D.-H. Kim, PL Lab., Sungshin W. University

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (6/7)

- **if-fi** Construct (Guarded if Statement)
  - Generalized selective statement
  - Syntax
    - **if** <guarded command set> **fi**
    - **if** $B_1 \rightarrow SL_1 \mid B_2 \rightarrow SL_2 \mid \ldots \mid B_n \rightarrow SL_n$ **fi**
  - Semantics:

    $$\text{wp}(\text{IF}, R) = (\exists\, j : 1 \le j \le n : B_j) \wedge$$
    $$(\forall\, j : 1 \le j \le n : B_j \Rightarrow \text{wp}(SL_j, R))$$

PROGRAMMING LANGUAGES AND DIJKSTRA'S MINI LANGUAGE (7/7)

- **do-od** Construct (Guarded do Statement)
  - Generalized repetitive statement
  - Syntax
    - **do** <guarded command set> **od**
    - **do** $B_1 \to SL_1 \mid B_2 \to SL_2 \mid \ldots \mid B_n \to SL_n$ **od**
  - Semantics:

$$\mathrm{wp}(\mathrm{DO}, R) = (\exists\, k : k \geq 0 : H_k(R)) \text{ where}$$

$$H_0(R) = R \wedge \neg (\exists\, j : 1 \leq j \leq n : B_j) \text{ and}$$

$$H_k(R) = \mathrm{wp}(\mathrm{IF}, H_{k-1}(R)) \vee H_0(R) \text{ for } k > 0$$

D.-H. Kim, PL Lab., Sungshin W. University

RELATED THEOREMS (1/5)

- Basic Theorem for the Alternative Construct
  - Let the alternative construct IF and a predicate pair Q and R be such that

$$Q \Rightarrow BB \text{ where } BB = (\exists\, j : 1 \leq j \leq n : B_j)$$

and

$$(\forall\, j : 1 \leq j \leq n : (Q \wedge B_j) \Rightarrow \mathrm{wp}(SL_j, R))$$

both hold for all states, then

$$Q \Rightarrow \mathrm{wp}(\mathrm{IF}, R)$$

holds for all states as well.

RELATED THEOREMS (2/5)

- Basic Theorem for the Repetitive Construct (or Fundamental Invariance Theorem for Loops)
  - Let a guarded command set with its derived alternative construct IF and a predicate P be such that

    $$(P \wedge BB) \Rightarrow \text{wp}(\text{IF}, P)$$

    holds for all states; then for the corresponding repetitive construct DO we can conclude that

    $$(P \wedge \text{wp}(\text{DO}, T)) \Rightarrow \text{wp}(\text{DO}, P \wedge \neg BB)$$

    for all states.

D.-H. Kim, PL Lab., Sungshin W. University

RELATED THEOREMS (3/5)

- Theorem for the Design of Properly Terminating Constructs
  - Let P be the relation that is kept invariant, i.e.,

    $$(P \wedge BB) \Rightarrow \text{wp}(\text{IF}, P) \text{ for all states,}$$

    let furthermore t be a finite integer function of the current state such that

    $$(P \wedge BB) \Rightarrow (t > 0) \text{ for all states,}$$

    and furthermore, for any value $t0$ and for all states

    $$(P \wedge BB \wedge t \leq t0 + 1) \Rightarrow \text{wp}(\text{IF}, t \leq t0).$$

    Then we can prove that

    $$P \Rightarrow \text{wp}(\text{DO}, T) \text{ for all states.}$$

RELATED THEOREMS (4/5)

- Basic Theorem for the Alternative Construct with

  $Q = (P \wedge BB \wedge t \leq t0 + 1)$

  $R = (t \leq t0)$

  - $(P \wedge BB \wedge t \leq t0 + 1) \Rightarrow$ wp(IF, $t \leq t0$) holds if

    $(\forall j : 1 \leq j \leq n : (P \wedge B_j \wedge t \leq t0 + 1) \Rightarrow$ wp$(SL_j, t \leq t0))$

    $\equiv (\forall j : 1 \leq j \leq n : (P \wedge B_j) \Rightarrow$ $(t \leq t0 + 1 \Rightarrow$ wp$(SL_j, t \leq t0)))$

RELATED THEOREMS (5/5)

- Summary
  - Let wdec$(SL_j, t) = (t \leq t0 + 1 \Rightarrow$ wp$(SL_j, t \leq t0))$.
  - The invariance of P and the effective decrease of t by at least 1 is guaranteed if we have for all j:

    $(P \wedge B_j) \Rightarrow$ (wp$(SL_j, P) \wedge$ wdec(SL_j, t))
  - Our B_j's must be strong enough so as to satisfy the above implication and as a result the now guaranteed post-condition $P \wedge \neg BB$ might be too weak to imply the desired post-condition R. In that case we have not solved our problem yet and we should consider other possibilities.

EXAMPLE 1 (1/2)

- Algorithm for Determining the Larger One
  - Problem: Establish for fixed x and y the relation

    $$R(m): (m = x \lor m = y) \land m \geq x \land m \geq y$$

  - Massaging operation(s): "$m := x$" or "$m := y$"
  - Derivation of guard(s):
    - wp("$m := x$", $R(m)$) = $R(x)$ =

      $((x = x \lor x = y) \land x \geq x \land x \geq y) = (x \geq y)$
    - wp ("$m := y$", $R(m)$) = $R(y)$ =

      $((y = x \lor y = y) \land y \geq x \land y \geq y) = (y \geq x)$

D.-H. Kim, PL Lab., Sungshin W. University

EXAMPLE 1 (2/2)

- Algorithm for Determining the Larger One (Continued)
  - Solution:

    if $x \geq y \rightarrow m := x$

    $\mid y \geq x \rightarrow m := y$

    fi
  - Our solution is not necessarily deterministic!

EXAMPLE 2 (1/4)

- Euclid's Algorithm (GCD Algorithm)
  - Problem: Establish for fixed X and Y the relation

    $R : ((x > 0 \wedge y > 0) \wedge$

    $(\mathrm{GCD}(X, Y) = \mathrm{GCD}(x, y)) \wedge (x = y))$
  - Weakened relation P (invariance):

    $((x > 0 \wedge y > 0) \wedge (\mathrm{GCD}(X, Y) = \mathrm{GCD}(x, y)))$
  - $\neg BB : x = y$
  - Initialization: "$x, y := X, Y$"

EXAMPLE 2 (2/4)

- Euclid's Algorithm (GCD Algorithm) (Continued)
  - Massaging operation(s):

    "$x, y := y, x$", "$x := x + y$", "$y := y - x$", …
  - Skeleton of the program:

    if $X > 0 \wedge Y > 0 \rightarrow$

    $x, y := X, Y$; {P has been established}

    do $x \neq y \rightarrow$ … {Massage x and y under the invariance of P}

    od; {$\neg BB$ has been established}

    fi {R has been established}

EXAMPLE 2 (3/4)

- Euclid's Algorithm (GCD Algorithm) (Continued)
  - t function: $t = x + y$ (or $t = | x - y |$?)
  - Derivation of guard(s):
    - wdec("$x := x - y$", $x + y$) = ($y > 0$)
    - wp("$x := x - y$", P) = (GCD(X, Y) = GCD($x - y$, y) $\land\ x - y > 0 \land y > 0$)

D.-H. Kim, PL Lab., Sungshin W. University

EXAMPLE 2 (4/4)

- Euclid's Algorithm (GCD Algorithm) (Continued)
  - The program:

if $X > 0 \land Y > 0 \rightarrow$

  $x, y := X, Y$; {P has been established}

  do $x > y \rightarrow x := x - y$

  | $y > x \rightarrow y := y - x$

  od; {$\neg BB$ has been established}

fi {R has been established}

EXAMPLE 3 (1/3)

- (Very Simple) Sorting Algorithm
  - Problem: For fixed $Q1$, $Q2$, $Q3$, and $Q4$ it is requested to establish R where

    $R = R1 \wedge R2$,

    $R1$: The sequence of values ($q1$, $q2$, $q3$, $q4$) is a permutation of the sequence of values ($Q1$, $Q2$, $Q3$, $Q4$), and

    $R2$: $q1 \leq q2 \leq q3 \leq q4$
  - Weakened relation P (invariance): $R1$

EXAMPLE 3(2/3)

- (Very Simple) Sorting Algorithm (Continued)
  - $\neg BB$: $R2$
  - Initialization: "$q1$, $q2$, $q3$, $q4$:= $Q1$, $Q2$, $Q3$, $Q4$"
  - Massaging operation(s): "$q1$, $q2$:= $q2$, $q1$", …
  - Skeleton of the program:

```
q1, q2, q3, q4 := Q1, Q2, Q3, Q4;
do   q1 > q2  →  …
   | q2 > q3  →  …
   | q3 > q4  →  …
od
```

EXAMPLE 3 (3/3)

- (Very Simple) Sorting Algorithm (Continued)
  - t function: $t = 4 * q1 + 3 * q2 + 2 * q3 + q4$
  - The program:

$q1, q2, q3, q4 := Q1, Q2, Q3, Q4;$

do $q1 > q2 \rightarrow q1, q2 := q2, q1$

$\mid q2 > q3 \rightarrow q2, q3 := q3, q2$

$\mid q3 > q4 \rightarrow q3, q4 := q4, q3$

od

EXAMPLE 4 (1/3)

- Approximate Square Root Algorithm
  - Problem: For fixed n ($n \geq 0$) the program should establish

$$R : a^2 \leq n \wedge (a + 1)^2 > n$$

  - Weakened relation P (invariance): $a^2 \leq n$
  - $\neg BB : (a + 1)^2 > n$
  - Initialization: "$a := 0$"
  - Massaging operation(s): "$a := a + 1$"

(Why? a is too small!)

EXAMPLE 4 (2/3)

- Approximate Square Root Algorithm (Continued)
  - Skeleton of the program:

    if $n \geq 0$ $\rightarrow$

    $a := 0$; {P has been established}

    do $(a + 1)^2 \leq n$ $\rightarrow$ $a := a + 1$

    od {$\neg BB$ has been established}

    fi {R has been established}
  - t function: $t = n - a^2$

EXAMPLE 4 (3/3)

- Approximate Square Root Algorithm (Continued)
  - Derivation of guard(s):

    wp(“$a := a + 1$”, P) = $((a + 1)^2 \leq n)$ (= BB!)

    wdec(“$a := a + 1$”, $n - a^2$)

    $= (n - (a + 1)^2 \leq n - a^2 - 1) = (a \geq 0)$
  - The program is not very efficient.

EXAMPLE 4' (1/6)

- Approximate Square Root Algorithm (Continued)
  - Weakened relation *P* (invariance):

    $a^2 \leq n \wedge b^2 > n \wedge 0 \leq a < b$
  - $\neg BB : (a + 1 = b)$
  - Initialization: "$a := 0; b := n + 1$"
  - Massaging operation(s):

    "Reduce $(b - a)$ until it reaches 1."

EXAMPLE 4' (2/6)

- Approximate Square Root Algorithm (Continued)
  - Skeleton of the program:

if $n \geq 0 \rightarrow$

$a, b := 0, n + 1$; {*P* has been established}

do $a + 1 \neq b \rightarrow$ decrease $(b - a)$ under invariance of *P*

od {$\neg BB$ has been established}

fi {*R* has been established}

EXAMPLE 4' (3/6)

- Approximate Square Root Algorithm (Continued)
  - t function: $t = b - a$
  - Derivation of guard(s):

$$wp(\text{“}a := a + d\text{”}, P) = (a + d)^2 \le n \wedge b^2 > n \wedge 0 \le a + d < b$$

$$wp(\text{“}b := b - d\text{”}, P) = a^2 \le n \wedge (b - d)^2 > n \wedge 0 \le a < b - d$$

$$wdec(\text{“}a := a + d\text{”}, b - a) = d \ge 1$$

$$wdec(\text{“}b := b - d\text{”}, b - a) = d \ge 1$$

EXAMPLE 4' (4/6)

- Approximate Square Root Algorithm (Continued)
  - Refined skeleton of the program:

```
if n ≥ 0 →
  a, b := 0, n + 1; {P has been established}
  do a + 1 ≠ b →
     d := (a "suitable" value between 0 and b – a);
     if (a + d)² ≤ n → a := a + d
      | (b – d)² > n → b := b – d
     fi
  od {¬ BB has been established}
fi {R has been established}
```

EXAMPLE 4' (5/6)

- Approximate Square Root Algorithm (Continued)
  - Determination of d:

$$\neg ((a + d)^2 \leq n) \Rightarrow ((b - d)^2 > n)$$

$$= ((a + d)^2 > n) \Rightarrow ((b - d)^2 > n)$$

$$= (a + d)^2 \leq (b - d)^2$$

$$= a + d \leq b - d$$

$$\therefore\ d = (b - a)\ \mathbf{div}\ 2$$

D.-H. Kim, PL Lab., Sungshin W. University

EXAMPLE 4' (6/6)

- Approximate Square Root Algorithm (Continued)
  - The program:

if $n \geq 0 \ \rightarrow$

  $a, b := 0, n + 1$; {P has been established}

  do $a + 1 \neq b \ \rightarrow$

    $d := (b - a)$ **div** 2;

    if $(a + d)^2 \leq n \ \rightarrow \ a := a + d$

      | $(b - d)^2 > n \ \rightarrow \ b := b - d$

    fi

  od {$\neg$ BB has been established}

fi {R has been established}

EXAMPLE 4" (1/6)

- Approximate Square Root Algorithm (Continued)
  - Weakened relation P (invariance):

    $a^2 \leq n \;\wedge\; (a + c)^2 > n \;\wedge\; (\exists\, i{:}\; i \geq 0{:}\; c = 2^i)$
  - $\neg\, BB : c = 1$
  - Initialization: "a := 0; c:= 2^k" ($k \geq 0$) (k = ?)
  - Skeleton of the program:

```
if  n ≥ 0  →
  a, c := 0, 1;
  do  c² ≤ n  →  c := 2 * c  od;
  do  c ≠ 1  →  …  od
fi
```


EXAMPLE 4" (2/6)

- Approximate Square Root Algorithm (Continued)
  - Massaging operation(s): "c := c / 2"
  - t function: $t = c$
  - Derivation of guard(s):

    wp("$c := c / 2$", P) = $a^2 \leq n \;\wedge\; (a + c / 2)^2 > n \;\wedge$

    $(\exists\, i{:}\; i \geq 0{:}\; c / 2 = 2^i)$

    wdec("$c := c / 2$; …", c) = $(c / 2 \leq c - 1) = (c > 1)$

EXAMPLE 4" (3/6)

- Approximate Square Root Algorithm (Continued)
  - The program:

if $n \geq 0 \rightarrow$
  $a, c := 0, 1;$
  do $c^2 \leq n \rightarrow c := 2 * c$ **od**; {P has been established}
  do $c \neq 1 \rightarrow c := c / 2;$ {P might have been destroyed}
    if $(a + c)^2 \leq n \rightarrow a := a + c$
    $\mid (a + c)^2 > n \rightarrow skip$
    fi {P has been recovered}
  od {$\neg BB$ has been established}
fi {R has been established}

D.-H. Kim, PL Lab., Sungshin W. University

EXAMPLE 4" (4/6)

- Approximate Square Root Algorithm (Continued)
  - Transformation of the program:

    $p = a * c$

    $q = c^2$

    $r = n - a^2$

  - Abstract variables (a, c) and concrete variables (p, q, and r)

EXAMPLE 4" (5/6)

- Approximate Square Root Algorithm (Continued)
  - The transformed program:

```
if n ≥ 0 →
   p, q, r := 0, 1, n;
   do q ≤ n → q, p := q * 4, p * 2 od;
   do q ≠ 1 → q := q / 4; p := p / 2;
              if r ≥ p * 2 + q →
                      p, r := p + q, r – p * 2 – q
               | r < p * 2 + q → skip
              fi
   od
fi
```


EXAMPLE 4" (6/6)

- Approximate Square Root Algorithm (Continued)
  - The final program after some optimizations:

```
if n ≥ 0 →
   p, q, r := 0, 1, n;
   do q ≤ n → q := q * 4 od;
   do q ≠ 1 → q := q / 4; h := p + q; p := p / 2;
              if r ≥ h → p, r := p + q, r – h
               | r < h → skip
              fi
   od
fi {p has the value desired for a}
```

EXAMPLE 5 (1/4)

- Remainder of Integer Division
  - Problem: For fixed a (≥ 0) and d (> 0), establish

    $$R : 0 \leq r < d \;\wedge\; d \mid (a - r).$$
  - Weakened relation P (invariance):

    $$0 \leq r \;\wedge\; d \mid (a - r)$$
  - $\neg\, BB : r < d$
  - Initialization: "$r := a$"
  - Massaging operation(s): "$r := r - d$"

EXAMPLE 5 (2/4)

- Remainder of Integer Division (Continued)
  - Skeleton of the program:

    if $a \geq 0$ **and** $d > 0 \;\rightarrow$
    $r := a$; {P has been established}
    do $r \geq d \rightarrow$ … {Massage r under the invariance of P}
    od {$\neg\, BB$ has been established}
    fi {R has been established}
  - t function: $t = r$

EXAMPLE 5 (3/4)

- Remainder of Integer Division (Continued)
  - Invariance and termination under the operation(s)

wp(“$r := r - d$”, P)

$= (0 \leq r - d) \wedge d \mid (a - r + d)$

$= (r \geq d) \wedge d \mid (a - r + d)$

(Implied by BB and P)

wdec(“$r := r - d$”, r)

$= r - d \leq r - 1 = d \geq 1 = d > 0$

(Implied by P)

EXAMPLE 5 (4/4)

- Remainder of Integer Division (Continued)
  - The program:

if $a \geq 0$ **and** $d > 0 \rightarrow$

$r := a$; {P has been established}

do $r \geq d \rightarrow r := r - d$

{P kept invariant and termination guaranteed}

od {$\neg BB$ has been further established}

fi {R has been established}

EXAMPLE 5' (1/4)

- Quotient and Remainder of Integer Division
  - Problem: For fixed a (≥ 0) and d (> 0), establish

    $$R : 0 \leq r < d \ \wedge\ d \mid (a - r) \ \wedge\ a = d * q + r.$$
  - Weakened relation P (invariance):

    $$0 \leq r \ \wedge\ d \mid (a - r) \ \wedge\ a = d * q + r$$
  - $\neg\, BB : r < d$
  - Initialization: "$r := a$; $q := 0$"
  - Massaging operation(s): "$r := r - d$; $q := q + 1$"

EXAMPLE 5' (2/4)

- Quotient and Remainder of Integer Division (Continued)
  - Skeleton of the program:

    if $a \geq 0$ **and** $d > 0 \ \rightarrow$

    $q, r := 0, a$; {P has been established}

    do $r \geq d \rightarrow$ … {Massage q and r under the invariance of P}

    od {$\neg\, BB$ has been established}

    fi {R has been established}
  - t function: $t = r$

EXAMPLE 5' (3/3)

- Quotient and Remainder of Integer Division (Continued)
  - Invariance and termination under the operation(s)

    wp(“$q, r := q + 1, r - d$”, P)

    $= (r \geq d) \ \wedge \ d \mid (a - r + d) \ \wedge \ a = d * q + r$

    (Implied by BB and P)

    wdec(“$q, r := q + 1, r - d$”, r)

    $= r - d \ \leq \ r - 1 = \ d \geq 1 = \ d \ > 0$

    (Implied by P)

EXAMPLE 5' (4/4)

- Quotient and Remainder of Integer Division (Continued)
  - The program:

    if $a \geq 0$ **and** $d > 0 \ \rightarrow$

    $q, r := 0, a$; {P has been established}

    do $r \geq d \rightarrow q, r := q + 1, r - d$

    od {$\neg BB$ has been established}

    fi {R has been established}

EXAMPLE 5" (1/6)

- Remainder of Integer Division (Continued)
  - Speedup of the first program
  - Weakened relation P (invariance):

    $0 \le r \ \wedge \ d \mid (a - r)$
  - $\neg BB : (r < d)$
  - Initialization: "$r := a$"
  - Massaging operation(s):

    "Reduce r by a suitable amount,

    which is not less than d (for speedup)"

EXAMPLE 5" (2/6)

- Remainder of Integer Division (Continued)
  - Skeleton of the program:

    if $a \ge 0$ **and** $d > 0 \ \rightarrow$

    $r := a$; {P has been established}

    do $r \ge d \rightarrow$ … {Massage r under the

    invariance of P}

    od {$\neg BB$ has been established}

    fi {R has been established}
  - t function: $t = r$

EXAMPLE 5" (3/6)

- Remainder of Integer Division (Continued)
  - Invariance and termination under the operation(s)

    wp("$r := r -$ (suitable amount)", P)

    = ($r \geq$ (suitable amount))

    $\wedge$ $d \mid (a - r +$ (suitable amount))

    (Implied by BB and P,

    if (suitable amount) is a multiple of d)

    wdec("$r := r -$ (suitable amount)", r)

    = (suitable amount) > 0

    (Implied by P,

    if (suitable amount) is a multiple of d)

EXAMPLE 5" (4/6)

- Remainder of Integer Division (Continued)
  - Invariant relation P' for establishing P:

    $0 \leq r \;\wedge\; d \mid (a - r) \;\wedge\; d \mid dd \;\wedge\; dd \geq d$
  - $\neg BB : T$
  - Initialization: "$dd := d$"
  - Massaging operation(s):

    "$r := r - dd \,;\, dd := dd + dd$"
  - t function: r

EXAMPLE 5" (5/6)

- Remainder of Integer Division (Continued)
  - Invariance and termination under the operation(s)

wp(“$r, dd := r - dd, dd + dd$”, P')

$= (r \geq dd) \ \wedge \ d \mid (a - r + dd)$

$\wedge \ d \mid 2 * dd \ \wedge \ 2 * dd \geq d$

(Implied by P', except for the first term)

wdec(“$r, dd := r - dd, dd + dd$”, r)

$= dd > 0$

(Implied by P')

EXAMPLE 5" (6/6)

- Remainder of Integer Division (Continued)
  - The program:

if $a \geq 0$ **and** $d > 0 \ \rightarrow$

$r := a$; {P has been established}

do $r \geq d \rightarrow$

$dd := d$; {P'}

do $r \geq dd \rightarrow r, dd := r - dd, dd + dd$ **od** {P'}

od {$\neg BB$ has been established}

fi {R has been established}

EXAMPLE 5''' (1/4)

- Remainder of Integer Division (Continued)
  - Further speedup of the first program
  - Weakened relation P (invariance):

    $0 \le r < dd \;\wedge\; dd \mid (a - r)$

    $\wedge\; (\exists\, i: i \ge 0: dd = d * 3^i)$
  - $\neg\, BB : (dd = d\,)$
  - Initialization: "$r, dd := a, d * 3^i$"

    (But, what must be the value of i? We need a **do-od** construct for this initialization.)

EXAMPLE 5''' (2/4)

- Remainder of Integer Division (Continued)
  - Skeleton of the program:

    if $a \ge 0$ **and** $d > 0 \;\rightarrow$

    $r, dd := a, d$;

    do $r \ge dd \rightarrow dd := dd * 3$ **od**; {P established}

    do $dd \ne d \rightarrow$ … {Massage dd under the invariance of P}

    od {$\neg\, BB$ has been established}

    fi {R has been established}
  - t function: $t = dd$
  - Massaging operation(s): $dd := dd / 3$

EXAMPLE 5''' (3/4)

- Remainder of Integer Division (Continued)
  - Invariance and termination under the operation(s)

    wp("$dd := dd / 3$", P)

    $= 0 \le r < dd / 3 \ \wedge \ (dd / 3) \mid (a - r)$

    $\wedge\ (\exists\ i: i \ge 0: dd / 3 = d * 3^i)$

    (The 2nd and 3rd terms are implied by P.)

    wdec("$dd := dd / 3; \ldots$ ", dd)

    $= dd > 1$

    (Implied by BB and the 3rd term of P)

EXAMPLE 5''' (4/4)

- Remainder of Integer Division (Continued)
  - The program:

```
if  a ≥ 0 and d > 0  →
  r, dd := a, d ;
  do r ≥ dd  →  dd := dd * 3 od; {P established}
  do dd ≠ d →
      dd := dd / 3;
      do r ≥ dd  → r := r − dd  od
      {Recovery of P may be necessary}
  od {¬ BB  has been established}
fi {R has been established}
```

CONCLUDING REMARKS (1/4)

- Summary
  - "Programs should be composed correctly, not just debugged into correctness."
  - Designing algorithms/programs is a goal-directed activity.

D.-H. Kim, PL Lab., Sungshin W. University

CONCLUDING REMARKS (2/4)

- Summary (Continued)
  - Clear separation between two of the programmer's major concerns, the mathematical correctness concerns and the engineering concerns about efficiency, by means of the predicate transformers
  - Explicit concerns about termination can be of great heuristic value for program design.

CONCLUDING REMARKS (3/4)

- Comments
  - Dijkstra's framework looks attractive, but how about its practicality?
  - Dijkstra's style is fascinating: his approach to programming as a high, intellectual challenge; his illuminating perception of problems at the foundations of program design; his eloquent presentation and deft demonstration of his own opinion.
  - The importance of culture: let's remind ourselves of Dijkstra's comment on the uselessness of program testing.

CONCLUDING REMARKS (3/3)

- References

  - C. A. R. Hoare, "An Axiomatic Basis for Computer Programming," *CACM* 12(10), 1969.
  - E. W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, 1976.
  - D.-H. Kim, "Annotated Translation of *A Discipline of Programming* ," *Transactions on Programming Languages*, KISS SIGPL, 1998-.
  - http://cs.sungshin.ac.kr/~dkim/tutorial.html