McMaster University Final Exam
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3
Day Class

COMP SCI 3EA3 — Software Specification and Correctness

April 25, 2017
Duration of Examination: **2.5 hours**

Name Student Number

This examination paper includes 18 pages (including this cover sheet); it consists of 7 questions (on the first 14 pages), plus a Theorem List (on pages 15–18). You are responsible for ensuring that your copy of the paper is complete. Bring any discrepancy to the attention of your invigilator.

Special Instructions:

- Make sure your name and student number are on all sheets.
- Do not separate the first 14 pages.
  You are allowed to separate the Theorem List pages 15–18.
- This is a **closed book** examination. No books, notes, texts, calculator or academic aids of any kind are permitted.
- Answer the questions in the space provided.
- **Read each question completely and carefully** before answering it.
- Answer all questions.
- You are **always allowed** to **introduce auxiliary definitions** and **prove auxiliary theorems**.
- **In doubt, document!**
- The marks add up to 100; the bonus question 7 is worth another three marks.

Contents

The following laws may be particularly useful in this examination,

“Abbreviation”	$a \le x < b \quad\equiv\quad a \le x \wedge x < b$
“Linear Order Negation”	$x < b \quad\equiv\quad \neg(b \le x)$
“Interval Split”	$a \le x < c \quad\equiv\quad a \le x < b \vee b \le x < c$ provided $a \le b \le c$
“if-fi Context”	**if** $x = y$ **then** $T\,x$ **else** $F\,y$ **fi** $=$ **if** $x = y$ **then** $T\,y$ **else** $F\,y$ **fi**
“if-fi Idempotency”	**if** b **then** s **else** s **fi** $=$ s

Name: Student Number:

1 Substitute It! — 15 marks —

Give the grammar for terms in the Backus-Naur Form presented in this class, then give the definition of substitution on terms, $\_[\_ := \_] : \mathsf{Term} \to \mathsf{Variable} \to \mathsf{Term} \to \mathsf{Term}$, by pattern matching on the first term. Afterwards prove $t[x := x] \;=\; t$.

Name: Student Number:

Name: Student Number:

2 The Other Half Of The Quadrivium — 10 marks —

1. Prove that any symmetric, associative, and idempotent operator '$\oplus$' is necessarily self-distributive:

$$x \oplus (y \oplus z) \;=\; (x \oplus y) \oplus (x \oplus z)$$

Be **explicit** about any properties of '$\oplus$' in your calculational proof.

2. Prove that provided x does not occur in P and R is non-empty, we have

"Superfluous Quantification for Idempotent $\oplus$": $(\oplus x \mid R \bullet P) \;=\; P$

As usual, a quantification operation is necessarily associative and symmetric; moreover in this context it is assumed to have an identity and to be idempotent.

Name: Student Number:

3 An Algorithm of the Old Sages — 15 marks —

Produce an algorithm —necessarily with proof— that satisfies the following informal specification:

> "Assign `present` to be *true* iff a given element E is in the ordered (monotone) array $f[0..N-1]$."

Name: Student Number:

Name: Student Number:

4 Assign Me To The Moon — 15 marks —

Using the heuristic of "programming is a goal-orented activity", step-by-step construct an algorithm to quantify over an array. Formally, solve:

$$\{\ 0 \leq N\ \}\ ?\ \{\ result\ =\ \left(\oplus\, i\ \mid\ 0 \leq i < N\ \bullet\ f\ i\right)\ \}$$

Name: Student Number:

Name: Student Number:

5 Quantify! Justify! Edify! — 30 marks —

The "maximum segment sum" is specified by computing, for integer array A,

$$\uparrow\ p, q \mid 0 \le p \le q \le \mathsf{length}\, A \bullet \big(\Sigma\, x \mid p \le x < q \bullet A\, x\big)$$

By the quantifier nesting law, we could formalise this in `ACSL` as

```
\max(0, len, \lambda integer p;
                \max(p, len, \lambda integer q;
                                \sum(p, q-1, \lambda integer x; A[x])))
```

An alternative is to name the important quantifications and give explicit definitions for them —we can always do this since a quantification over an interval is just a recursively defined notation! For $n : \mathbb{N}$,

$$\begin{aligned}
\mathsf{max2D}\, n &= \big(\uparrow\ p, q \mid 0 \le p \le q \le n \bullet S\, p\, q\big)\\
S\, p\, q &= \big(\Sigma\, x \mid p \le x < q \bullet A\, x\big)\\
\mathsf{max1D}\, n &= \big(\uparrow\ p \mid 0 \le p \le n \bullet S\, p\, n\big)
\end{aligned}$$

Now it suffices to compute $\mathsf{max2D}\, n$ where $n = \mathsf{length}\, A$. As such, let us find a recursive definition of $\mathsf{max2D}$.

Taking $n = 0$ and simplifying we find $\mathsf{max2D}\, 0 = 0$, then using split-off term we obtain that $\mathsf{max2D}\,(n+1) = \mathsf{max2D}\, n \ \uparrow\ \mathsf{max1D}\,(n+1)$. Consequently, it seems we need to find a recursive definition of $\mathsf{max1D}$. The case $n = 0$ again simplifies to 0, whereas the inductive case is more involved.

— 15 marks — For $0 \le n$, prove $\mathsf{max1D}\,(n+1) = \big(\mathsf{max1D}\, n + A\, n\big) \ \uparrow\ 0$

Name: Student Number:

(Additional space for previous proof)

Name: Student Number:

— 3 marks — Use the previous discussion to fill in the following axiomatisation —the last one is done for you.

```
#define max(a,b) ((a) > (b) ? (a) : (b))

/*@ axiomatic MyMaxOps {
  @ logic integer max1D(int* A, integer len)                                   ;
  @ logic integer max2D(int* A, integer len)                                   ;
  @
  @ axiom base1: \forall int* A                                                ;
  @       max1D(A, 0) ==                                                       ; // Here

  @ axiom splitOff1 : \forall int* A; \forall integer n                        ;
  @       max1D(A, n + 1) ==                                                   ; // Here

  @
  @ axiom base2 : \forall int* A                                               ;
  @       max2D(A, 0) ==                                                       ; // Here

  @ axiom splitOff2 : \forall int* A; \forall integer n                        ;
  @       max2D(A , n + 1) == max( max2D(A, n) , max1D(A, n + 1) )             ; // Done
  @ }
  @*/
```

— 12 marks — Provide appropriate specifications for the the following maximum segment sum program:

```
/*@ requires                                                                   ; // Here

  @ requires                                                                   ; // Here

  @ assigns                                                                    ; // Here

  @ ensures                                                                    ; // Here
  */
int kaldewaij(int* A, int len)
{
  int n , r , s; n = r = s = 0;
  /*@ loop invariant                                                           ; // Here

  @ loop invariant r ==                                                        ; // Here

  @ loop invariant s ==                                                        ; // Here

  @ loop assigns                                                               ; // Here

  @ loop variant                                                               ; // Here
  */
  while( n != len )
  {
    s = max(s + A[n] , 0) ;
    r = max(r, s)         ;
    n = n + 1             ;
  }
  return r;
}
```

Name: Student Number:

6 Knowing The Source Code — 15 marks —

One ought to be comfortable using a variety of notational languages —e.g., different programming languages!

As an analogue to the integral $\int_a^b f(x)\,dx$ of a traditional first-year education in *continuous* mathematics, let us introduce the "sum" operation as a *discrete* counterpart,

$$\sum_a^b f(x)\,\delta x \;=\; (+x : \mathbb{Z} \mid a \le x < b \bullet f\,x)$$

A host of familar laws from the continous setting also hold in the discrete setting. In-particular, prove "Additivity": For $a \le b \le c$,

$$\sum_a^b f(x)\,\delta x + \sum_b^c f(x)\,\delta x \;=\; \sum_a^c f(x)\,\delta x$$

Name: Student Number:

Recalling the definition,

$$\sum_a^b f(x)\,\delta x \;=\; (+x:\mathbb{Z} \mid a \le x < b \bullet f\,x)$$

Also, prove "Fubini's Theorem": For $a \le b$ and $c \le d$,

$$\sum_a^b \left(\sum_c^d f(x,y)\,\delta x \right) \delta\, y \;=\; \sum_c^d \left(\sum_a^b f(x,y)\,\delta y \right) \delta\, x$$

7 (Bonus) Why Are We Here? — 3 marks —

Define the term *correct-by-construction programming*.

Have A Great Summer!

End of Exam

McMaster University Final Exam
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3
"Program Construction"
Theorem List

COMP SCI 3EA3 — Software Specification and Correctness

April 18, 2017
Duration of Examination: **2.5 hours**

Environments

Our common environments will be the following distributive lattices

- Booleans: $(\mathbb{B}, \Rightarrow, \land, \lor, false, true)$ —Our ambient logic!
- Extended Number Line: $(\mathbb{R}, \le, \downarrow, \uparrow, -\infty, +\infty)$
- Naturals under division: $(\mathbb{N}, \mid, \mathsf{gcd}, \mathsf{lcm}, 1, 0)$
- Substructures of a given datatype with the substructure ordering.
  E.g., sets, lists, and graphs with subset, subsequence, and subgraph ordering.

Simultaneous Textual Substitution

Identity Substitution: $t[x := x] \ = \ t$

Superfluous Substitution: $t[x := E] \ = \ t$ provided $\neg occurs(`x', `t')$

Re-Substitution: $t[x := E][x := F] \ = \ t[x := E[x := F]]$

Iterated Substitution:

$t[x := E][y := F] \ = \ t[x := E[y := F]]$ provided $\neg occurs(`y', `t')$
$t[x := E][y := F] \ = \ (t[x := E[y := F]])[y := F]$ provided $\neg occurs(`y', `F')$
$t[x := E][y := F] \ = \ t[x, y := E[y := F], F]$ provided $\neg occurs(`y', `F')$

Axiom, Function Patching Definition: $f[x \mapsto E](y) \ = \ \textbf{if } x = y \textbf{ then } E \textbf{ else } f(y) \textbf{ fi}$

Propositonal Calculus

Metatheorem: Any two theorems are equivalent; '$true$' is a theorem.

Equivales is an equivalence relation that is associative —$((p \equiv q) \equiv r) \ \equiv \ (p \equiv (q \equiv r))$— and has identity $true$.

Discrepancy '$\not\equiv$' is symmetric, associtive, has identity '$false$', mututally associates with equivales —$((p \not\equiv q) \equiv r) \ \equiv \ (p \not\equiv (q \equiv r))$— and mutually interchanges with it as well —$p \not\equiv q \equiv r \ \equiv \ p \equiv q \not\equiv r$—.

Implication has the alternative definition $p \Rightarrow q \ \equiv \ \neg p \lor q$, has '$true$' as left identity and '$false$' as right zero, distributes over $\equiv$ in the second argument, and is self-distributive; and has the properties

Shunting:
$p \land q \Rightarrow r \quad \equiv \quad p \Rightarrow (q \Rightarrow r)$

Contrapositive:
$p \Rightarrow q \quad \equiv \quad \neg q \Rightarrow \neg p$

Modus Ponens:
$p \land (p \Rightarrow q) \quad \equiv \quad p \land q$
$p \land (q \Rightarrow p) \quad \equiv \quad p$
$p \land (p \Rightarrow q) \quad \Rightarrow \quad q$

Moreover it has the property "(3.62)": $p \Rightarrow (q \equiv r) \quad \equiv \quad p \land q \ \equiv \ p \land r$.

Conjunction and disjunction distributes over one another, $\lor$ distributes over $\equiv$, $\land$ distributes over $\equiv$-$\equiv$ in that $p \land (q \equiv r \equiv s) \quad \equiv \quad p \land q \ \equiv \ p \land r \ \equiv \ p \land s$, and they satisfy,

Excluded Middle:
$p \lor \neg p$

Contradiction:
$p \land \neg p \quad \equiv \quad false$

Absorption:
$p \land (\neg p \lor q) \quad \equiv \quad p \land q$
$p \lor (\neg p \land q) \quad \equiv \quad p \lor q$

De Morgan:
$\neg(p \land q) \quad \equiv \quad \neg p \lor \neg q$
$\neg(p \lor q) \quad \equiv \quad \neg p \land \neg q$

With quantifier properties,

Generalised De Morgan: $\neg(\forall\, x \mid R \bullet P) \equiv (\exists\, x \mid R \bullet \neg P)$

Trading:
$(\forall\, x \mid Q \land R \bullet P) \quad \equiv \quad (\forall\, x \mid Q \bullet R \Rightarrow P)$
$(\exists\, x \mid Q \land R \bullet P) \quad \equiv \quad (\exists\, x \mid Q \bullet R \land P)$

Metatheorem: P is a theorem iff $(\forall\, x \bullet P)$ is a theorem.

Metatheorem Witness: If $\neg occurs(`x', `Q')$, then:

$(\exists\, x \mid R \bullet P) \Rightarrow Q$ is a theorem iff $R \land P \ \Rightarrow \ Q$ is a theorem.

Finally, we have a few substitution laws:

(7.27) **Axiom, Context:** $e = f \ \land \ E[z := e] \quad \equiv \quad e = f \ \land \ E[z := f]$

(7.27a) **Context:** $e = f \ \Rightarrow \ E[z := e] \quad \equiv \quad e = f \ \Rightarrow \ E[z := f]$

(7.28) **Leibniz:** $e = f \quad \equiv \quad e = f \ \land \ (E[z := e] = E[z := f])$

Setoids

Axiom, Reflexivity of $\approx$: $a \approx a$

Axiom, Symmetry of $\approx$: $a \approx b \quad \equiv \quad b \approx a$

Axiom, Transitivity of $\approx$: $a \approx b \ \land \ b \approx c \quad \Rightarrow \quad a \approx c$

Axiom, Leibniz:
$a = b \quad \Rightarrow \quad a \approx b$
$a = b \quad \Rightarrow \quad f\ a = f\ b$

Context/Replacement: $a \approx b \ \land \ c \approx a \quad \equiv \quad a \approx b \ \land \ c \approx b$

Posets

Axiom, Reflexivity of $\sqsubseteq$: $a \sqsubseteq a$

Reflexivity of $\sqsubseteq$ wrt Equality: $a = b \quad \Rightarrow \quad a \sqsubseteq b$

Axiom, Transitivity of $\sqsubseteq$: $a \sqsubseteq b \ \land \ b \sqsubseteq c \quad \Rightarrow \quad a \sqsubseteq c$

Transitivity / Inclusion Absorbs Equality:
$a = b \ \land \ b \sqsubseteq c \quad \Rightarrow \quad a \sqsubseteq c$
$a \sqsubseteq b \ \land \ b = c \quad \Rightarrow \quad a \sqsubseteq c$

Axiom, Antisymmetry of $\sqsubseteq$: $a \sqsubseteq b \ \land \ b \sqsubseteq a \quad \Rightarrow \quad a = b$

Antisymmetry / Mutual Implication: $a \sqsubseteq b \ \land \ b \sqsubseteq a \quad \equiv \quad a = b$

Axiom, Dual Order: $b \sqsupseteq a \quad \equiv \quad a \sqsubseteq b$

Axiom, Top Element: $a \ \sqsubseteq \ \top$

Axiom, Bottom Element: $\bot \ \sqsubseteq \ a$

The following have the *provisio* "$R[z := a] \ \land \ R[z := b]$" —most often "$R \equiv true$". (§8.2)

Indirect Equality, from below: $a = b \quad \equiv \quad (\forall z \mid R \bullet z \sqsubseteq a \quad \equiv \quad z \sqsubseteq b)$

Indirect Equality, from above: $a = b \quad \equiv \quad (\forall z \mid R \bullet a \sqsubseteq z \quad \equiv \quad b \sqsubseteq z)$

Indirect Inclusion, from below: $a \sqsubseteq b \quad \equiv \quad (\forall z \mid R \bullet z \sqsubseteq a \quad \Rightarrow \quad z \sqsubseteq b)$

Indirect Inclusion, from above: $a \sqsubseteq b \quad \equiv \quad (\forall z \mid R \bullet a \sqsubseteq z \quad \Leftarrow \quad b \sqsubseteq z)$

In a lattice, these rules can be combined with 'induced definition of inclusion' and 'golden rule' to obtain other various forms of indirect (in)equality.

Heuristic : pick property R that 'a' and 'b' satisfy so that you can use the "Translation" rule below to bring more content about z into the body of the quantification to make it more amiable to calculation.

Lattices

Let $\square$ be one of $\sqcap$ or $\sqcup$.

Axiom, $\sqcup$-Characterisation: $a \sqsubseteq c \land a \sqsubseteq c \quad \equiv \quad a \sqcup b \sqsubseteq c$ (8.1)

Axiom, $\sqcap$-Characterisation: $c \sqsubseteq a \land c \sqsubseteq b \quad \equiv \quad c \sqsubseteq a \sqcap b$ (8.6)

Symmetry of $\square$: $a \square b \quad \equiv \quad b \square a$

Associativity of $\square$: $(a \square b) \square c \quad = \quad a \square (b \square c)$

Idempotency of $\square$: $a \square a \quad = \quad a$

Zero of $\square$:

$$a \sqcup \top \quad = \quad \top$$
$$a \sqcap \bot \quad = \quad \bot$$

Identity of $\square$:

$$a \sqcup \bot \quad = \quad a$$
$$a \sqcap \top \quad = \quad a$$

Self-Distributivity of $\square$: $a \square (b \square c) \quad = \quad (a \square b) \square (a \square c)$

Monotonicity of $\square$: $a \sqsubseteq b \land c \sqsubseteq d \quad \Rightarrow \quad a \square c \sqsubseteq b \square d$

Weakening/strengthening: (7.20)

$$a \sqsubseteq a \sqcup b$$
$$a \sqcap b \sqsubseteq a$$
$$a \sqcap b \sqsubseteq a \sqcup b$$
$$a \sqcup (b \sqcap c) \sqsubseteq a \sqcup b$$
$$a \sqcap b \sqsubseteq a \sqcap (b \sqcup c)$$

Absorption: (7.11)

$$a \sqcap (a \sqcup b) \quad = \quad a$$
$$a \sqcup (a \sqcap b) \quad = \quad a$$

Induced **Definition of Inclusion:**

$$a \sqsubseteq b \quad \equiv \quad a \sqcup b = b \qquad (7.16)$$
$$a \sqsubseteq b \quad \equiv \quad a \sqcap b = a \qquad (7.17)$$

Golden Rule: (7.9)

$$a \sqcap b = a \quad \equiv \quad b = a \sqcup b$$
$$a \sqcap b = a \sqcup b \quad \equiv \quad a = b$$
$$a \sqcup b \sqsubseteq a \sqcap b \quad \equiv \quad a = b$$

Duality Principle:
If a statement S is a theorem, then so is $S[(\sqsubseteq, \sqcap, \sqcup, \top, \bot) := (\sqsupseteq, \sqcup, \sqcap, \bot, \top)]$

Quantification

Assume $\oplus$ is associative and symmetric; if it has an identity we denote it by e, and if it has a zero we denote it by z. Note, "R non-empty $\equiv (\exists x \bullet R)$".

Laws apply only when each appearing quantification is defined.
(Quantifications over $\land$ or $\lor$ are always defined; whereas over '$+$' may not be.)

(11.-1) **Axiom, Abbreviations:** In the second case, 'x' must be a single identifer.

$$(\oplus x \bullet P) \quad = \quad (\oplus x \mid true \bullet P)$$
$$(\oplus x \mid R) \quad = \quad (\oplus x \mid R \bullet x)$$

(11.0) **Axiom, Substitution:** Provided $\neg occurs$('y', 'x, F'),

$$(\oplus y \mid R \bullet P)[x := F] \quad = \quad (\oplus y \mid R[x := F] \bullet P[x := F])$$

(11.44) **Axiom, Dummy renaming**/α-conversion: If $\neg occurs$('y', 'R, P'),

$$(\oplus x \mid R \bullet P) \quad = \quad (\oplus y \mid R[x := y] \bullet P[x := y])$$

(11.45) **Axiom, Nesting:** Provided $\neg occurs$('y', 'R'),

$$(\oplus x, y \mid R \land S \bullet P) \quad = \quad (\oplus x \mid R \bullet (\oplus y \mid S \bullet P))$$

(11.46) **Axiom, Dummy List Permutation:**

$$(\oplus x, y \mid R \bullet P) \quad = \quad (\oplus y, x \mid R \bullet P)$$

(11.47) **Axiom, Empty Range:** $(\oplus x \mid false \bullet P) \quad = \quad \mathsf{e}$
(11.47a) **Axiom, Unit Body:** $(\oplus x \mid R \bullet \mathsf{e}) \quad = \quad \mathsf{e}$
(11.47b) **Axiom, Zero Body:** $(\oplus x \mid R \bullet \mathsf{z}) \quad = \quad \mathsf{z}$ for non-empty R

(11.48) **Axiom, One-point Rule:** Provided $\neg occurs$('x', 'E'),

$$(\oplus x \mid x = E \bullet P) \quad = \quad P[x := E]$$

(11.49) **Axiom, Range Split:**

$$(\oplus x \mid R \lor S \bullet P) \oplus (\oplus x \mid R \land S \bullet P) \quad = \quad (\oplus x \mid R \bullet P) \oplus (\oplus x \mid S \bullet P)$$

(11.49v) **Range Split:** Provided $R \land S = false$,

$$(\oplus x \mid R \lor S \bullet P) \quad = \quad (\oplus x \mid R \bullet P) \oplus (\oplus x \mid S \bullet P)$$

(11.50) **Range Split for idempotent** $\oplus$:

$$(\oplus x \mid R \lor S \bullet P) \quad = \quad (\oplus x \mid R \bullet P) \oplus (\oplus x \mid S \bullet P)$$

(11.51) **Axiom, Range Split for idempotent** $\oplus$: Provided $\neg occurs$('x', 'R'),

$$(\oplus x \mid (\exists i \mid R \bullet S) \bullet P) \quad = \quad (\oplus i \mid R \bullet (\oplus x \mid S \bullet P))$$

(11.53) **Axiom, Trading:**

$$(\oplus x \mid R \land S \bullet P) \quad = \quad (\oplus x \mid S \bullet \textbf{if } R \textbf{ then } P \textbf{ else } \mathsf{e} \textbf{ fi})$$

(11.54) **Axiom,** (Quantification) **Distributivity**/Rearranging:

$$(\oplus x \mid R \bullet P \oplus Q) \quad = \quad (\oplus x \mid R \bullet P) \oplus (\oplus x \mid R \bullet Q)$$

(Thm) **Interchange of dummies:** if $\neg occurs$('y', 'R') and $\neg occurs$('x', 'S'),

$$(\oplus x \mid R \bullet (\oplus y \mid S \bullet P)) \quad = \quad (\oplus y \mid S \bullet (\oplus x \mid R \bullet P))$$

(11.55) **Axiom, Distributivity of $\otimes$ over $\oplus$:**

$$P \otimes (\oplus x \mid R \bullet Q) \quad = \quad (\oplus x \mid R \bullet P \otimes Q)$$

Provided $\neg occurs$('x', 'P'), R is non-empty, and $\otimes$ distributes over $\oplus$.

Our usual lattices are distributive and addition and positive multiplication distribute over $\uparrow$, $\downarrow$. Idempotents are self-distributive.
Non-emptiness proviso not required if $\mathsf{unit}\ \oplus = \mathsf{zero}\ \otimes$, such as $+/\times$ and $\sqcup/\sqcap$.

(11.57) **Translation**/Change of dummy:

$$(\oplus y : Y \mid R \bullet P) \quad = \quad (\oplus x : X \mid R[y := f\,x] \bullet P[y := f\,x])$$

Provided $f : X \to Y$ has an inverse and $\neg occurs$('y', 'x, R, P')

(11.57) **Translation**/Change of dummy **for idempotent** $\oplus$:

$$(\oplus y : Y \mid R \bullet P) \quad = \quad (\oplus x : X \mid R[y := f\,x] \bullet P[y := f\,x])$$

Provided $f : X \to Y$ is surjective and $\neg occurs$('y', 'x, R, P')

Heuristic : pick f so that $R[y := f\,x]$ is always $true$, thereby simplifying the quantification, and giving more content to the body 'P' thereby making it more friendly to calculation.

(Thm) **Theorem Split off term:** For $m, n : \mathbb{Z}$ and dummies $i : \mathbb{Z}$, provided $m < n + 1$:

$$(\oplus i \mid m \le i < n + 1 \bullet P) \quad = \quad (\oplus i \mid m \le i < n \bullet P) \oplus P[i := n]$$
$$(\oplus i \mid m \le i < n + 1 \bullet P) \quad = \quad P[i := m] \oplus (\oplus i \mid m < i < n + 1 \bullet P)$$

Lattice Quantification

Axiom, (Body) **Monotonicity of $\oplus$:** Body weakening/strengthening

$$(\forall x \mid R \bullet Q \sqsubseteq P) \quad \Rightarrow \quad (\oplus x \mid R \bullet Q) \sqsubseteq (\oplus x \mid R \bullet P)$$

Provided $\oplus$ preserves order: $a \sqsubseteq b \land c \sqsubseteq d \quad \Rightarrow \quad a \oplus c \sqsubseteq b \oplus d$

Axiom, Generalised $\square$-Characterisation:

$$Q \sqsubseteq (\sqcap x \mid R \bullet P) \quad \equiv \quad (\forall x \mid R \bullet Q \sqsubseteq P)$$
$$(\sqcup x \mid R \bullet P) \sqsubseteq Q \quad \equiv \quad (\forall x \mid R \bullet P \sqsubseteq Q)$$

$\sqcap$-Elimination/Instantiation: $R[x := E] \quad \Rightarrow \quad (\sqcap x \mid R \bullet P) \sqsubseteq P[x := E]$

$\sqcup$-Introduction/Witness: $R[x := E] \;\Rightarrow\; P[x := E] \sqsubseteq (\sqcup x \mid R \bullet P)$

Induced $\Box$-Definition:

$$\begin{array}{lll} m = (\sqcap x \mid R \bullet P) & \equiv & (\forall x \mid R \bullet m \sqsubseteq P) \wedge (\forall l \mid (\forall x \mid R \bullet l \sqsubseteq P) \bullet l \sqsubseteq m) \\ j = (\sqcup x \mid R \bullet P) & \equiv & (\forall x \mid R \bullet P \sqsubseteq j) \wedge (\forall u \mid (\forall x \mid R \bullet P \sqsubseteq u) \bullet j \sqsubseteq u) \end{array}$$

Meet is greatest lower bound: $(\sqcap x \mid R \bullet P) \;=\; (\sqcup l \mid (\forall x \mid R \bullet l \sqsubseteq P))$

Join is least upper bound: $(\sqcup x \mid R \bullet P) \;=\; (\sqcap u \mid (\forall x \mid R \bullet P \sqsubseteq u))$

Range-Antitonicity of $\sqcap$: $(\forall x \bullet Q \Rightarrow R) \;\Rightarrow\; (\sqcap x \mid R \bullet P) \sqsubseteq (\sqcap x \mid Q \bullet P)$

Range-Monotonicity of $\sqcup$: $(\forall x \bullet Q \Rightarrow R) \;\Rightarrow\; (\sqcup x \mid Q \bullet P) \sqsubseteq (\sqcup x \mid R \bullet P)$

Interchange of quantifications:

$$(\sqcup x \mid R \bullet (\sqcap y \mid Q \bullet P)) \;\sqsubseteq\; (\sqcap y \mid Q \bullet (\sqcup x \mid R \bullet P))$$

Provided $\neg occurs(`y', `R') \wedge \neg occurs(`x', `Q')$; and $\sqcup$ distributes over $\sqcap$.

Definition, Order Morphisms: For expressions R and P, with free variable i,

$$\begin{array}{llcl} P \text{ monotone} & \text{on } R & : & \forall x, y \mid R[i := x] \wedge R[i := y] \bullet x \le y \Rightarrow P[i := x] \sqsubseteq P[i := y] \\ P \text{ antitone} & \text{on } R & : & \forall x, y \mid R[i := x] \wedge R[i := y] \bullet x \le y \Rightarrow P[i := y] \sqsubseteq P[i := x] \end{array}$$

One-Point Rule For Monotone Body: If P monotone on R and $R[i := l] \wedge R[i := u]$,

$$\begin{array}{lcl} (\sqcap i \mid R \wedge l \le i \bullet P) & = & P[i := l] \\ (\sqcup i \mid R \wedge i \le u \bullet P) & = & P[i := u] \end{array}$$

One-Point Rule For Antitone Body: If P antitone on R and $R[i := l] \wedge R[i := u]$,

$$\begin{array}{lcl} (\sqcap i \mid R \wedge i \le u \bullet P) & = & P[i := u] \\ (\sqcup i \mid R \wedge l \le i \bullet P) & = & P[i := l] \end{array}$$

Induced $\Box$-Definition for Numbers: Provided R non-empty and finite,

$$\begin{array}{lcl} f\ m = (\downarrow x \mid R \bullet f\ x) & \equiv & R[x := m] \wedge (\forall x \mid R \bullet f\ m \le f\ x) \\ f\ m = (\uparrow x \mid R \bullet f\ x) & \equiv & R[x := m] \wedge (\forall x \mid R \bullet f\ x \le f\ m) \end{array}$$

Local Characterisation of Integer Extrema:
Provided R non-empty and finite, and $\neg R$ monotonic,

$$\begin{array}{lcl} l = (\uparrow i : \mathbb{Z} \mid R) & = & R[i := l] \wedge \neg R[i := l + 1] \\ s = (\downarrow i : \mathbb{Z} \mid R) & = & R[i := s] \wedge \neg R[i := s - 1] \end{array}$$

Hoare Triple Definitions

$\{Q\}\ S\ \{R\}$: execution of S begun in any state satisfying predicate Q would terminate in a state satisfying predicate R. The set of all such states Q is denoted $\mathsf{wp}\ S\ R$. Page 110

Axiom, Hoare Triple Definition: $\{Q\}\ S\ \{R\} \;\equiv\; Q \Rightarrow \mathsf{wp}\ S\ R$

Heuristic : When attempting to prove $\{Q\}\ S_1; S_2; \cdots; S_n\ \{R\}$ we "push" R left-wards using rule $\{\mathsf{wp}\ S\ R\}\ S\ \{R\}$ to obtain $\{Q\}\ S_1; \cdots; S_{n-1};\ \{\mathsf{wp}\ S_n\ R\}\ S_n\ \{R\}$. We use the *required goal* R to **guide** us in calculating/proving our program correct.

Axiom, Law of The Excluded Miracle: $\mathsf{wp}\ S\ \mathit{false} \;=\; \mathit{false}$

Axiom, Distributivity of Conjunction: $\mathsf{wp}\ S\ (P \wedge R) \;\equiv\; \mathsf{wp}\ S\ P \wedge \mathsf{wp}\ S\ R$

Monotoncity in the second argument: $(P \Rightarrow R) \;\Rightarrow\; (\mathsf{wp}\ S\ P \Rightarrow \mathsf{wp}\ S\ R)$

Precondition Strengthening: $G \Rightarrow G' \;\Rightarrow\; \{G'\}\ S\ \{R\} \Rightarrow \{G\}\ S\ \{R\}$

Postcondition Weakening: $R \Rightarrow R' \;\Rightarrow\; \{G\}\ S\ \{R\} \Rightarrow \{G\}\ S\ \{R'\}$

Semidistributivity of Disjunction: $\mathsf{wp}\ S\ P \vee \mathsf{wp}\ S\ R \;\Rightarrow\; \mathsf{wp}\ S\ (P \vee R)$

Axiom, Program Equality: $S \approx T \;\equiv\; (\forall R \bullet \mathsf{wp}\ S\ R \equiv \mathsf{wp}\ T\ R)$
Theorem: All program constructions preserve this equivalence.

Skip and Sequence

Axiom, Skip Rule: $\mathsf{wp}\ \text{“skip”}\ R \;=\; R$ §10.2

Axiom/Theorem, Sequence Rule: §10.1

$$\begin{array}{lcl} \mathsf{wp}\text{“}S; T\text{”}\ R & = & \mathsf{wp}\ S\ (\mathsf{wp}\ T\ R) \\ \{Q\}\ S; T\ \{R\} & \Leftarrow & \{Q\}\ S\ \{P\} \wedge \{P\}\ T\ \{R\} \end{array}$$

Sequencing is Associative: $(S; T); U \;\approx\; S; (T; U)$

Identity of sequence: $\mathsf{skip}; S \;\approx\; S; \mathsf{skip} \;\approx\; S$

Assignment

Axiom, Assignment Rule: $\mathsf{wp}\ \text{“}x := E\text{”}\ R \;=\; R[x := E] \wedge E$ well-defined §9.7

Superfluous Variable: $x := E;\ S \;\approx\; S$ provided $\neg occurs(`x', `S')$
"Execution of $x := E$ may change *only* x, and no other variable."

Simultaneous and Sequential Assignment Interchange:

$$x, y := E, F \;\approx\; x := E;\ y := F \;\approx\; y := F;\ x := E$$

provided $\neg occurs(`x', `F') \wedge \neg occurs(`y', `E')$

Identity Assignment:

$$\begin{array}{lcll} x := x & \approx & \mathsf{skip} & \\ x, y := E, y & \approx & x := E & \text{provided } \neg occurs(`x', `y') \end{array}$$

Conditional and Iterative Constructs

We use quantification notation for guarded commands, for example the alternative-command on the right. This notation is suggestive of certain *derived* properties: the order of the guarded commands does not matter —'$[\!]$ is symmetric'—, and identical guarded commands can be replaced with one instance —'$[\!]$ is idempotent'— without actually giving a theory of '$[\!]$' as an operator of the language. These properties follow from the definition of **if** .. **fi** in terms of quantifiers $\exists$ and $\forall$, which are themselves idempotent and symmetric.

```
  if i | 0 ≤ i < N • B_i → S_i fi
=
  if
   [] B_0 → S_0
   [] B_1 → S_1
   ⋮
   [] B_{N−1} → S_{N−1}
  fi
```

Axiom/Theorem, Conditional Rule: §10.4

$$\begin{array}{lcl} \mathsf{wp}\ \text{“}\mathbf{if}\ i \bullet B_i \to S_i\ \mathbf{fi}\text{”}\ R & \equiv & (\exists i \bullet B_i) \wedge (\forall i \bullet \{B_i\}\ S_i\ \{R\}) \\ & & \wedge \text{ each } B_i \text{ is defined} \\ \{Q\}\ \mathbf{if}\ i \bullet B_i \to S_i\ \mathbf{fi}\ \{R\} & \equiv & (Q \Rightarrow (\exists i \bullet B_i)) \wedge (\forall i \bullet \{Q \wedge B_i\}\ S_i\ \{R\}) \\ & & \wedge \text{ each } B_i \text{ is defined} \end{array}$$

Heuristic "Case Analysis": To solve "`Given G, establish R`", if we *find* B_i with $G \Rightarrow B_1 \vee \cdots \vee B_n$ then the solution is: **if** $i \bullet B_i \to$ "`Given G ∧ B_i, establish R`" **fi**.

Axiom, Iteration Definition: §13.1

```
   do i • B_i → S_i od
≈  if
     []i • B_i  →  S_i ; do i • B_i → S_i od
     []else     →  skip
   fi          where else = ¬(∃i • B_i)
```

Axiom, Iteration Rule: $\mathsf{wp}\ \text{“}\mathbf{do}\ i \bullet B_i \to S_i\ \mathbf{od}\text{”}\ R \;=\; (\exists i : \mathbb{N} \bullet f^{i+1}(\mathit{false}))$
where $f(X) = (\forall i \bullet \{B_i\}\ S_i\ \{X\}) \wedge (\mathit{else} \Rightarrow R)$ Sheet 6

End of Exam

$\boxed{\textbf{Program Construction}}$

Heuristic “Programming is a *goal-oriented* activity”:
1. Formalise ‘Givens’ and ‘Requireds’ of the problem.
2. Obtain an invariant P and initalise the variables to make it true.
3. Bridge from invariant to post-condition: solve for B in $P \land \neg B \Rightarrow R$
4. If $\neg B$ holds then we’re done, otherwise we construct a loop to obtain it.
5. Solve for a “bound function” bf in $P \land B \Rightarrow bf > 0$.
6. Make progress towards termination: find a program S that *decreases* the bound.
7. Refine program S so that it *maintains* the invariant!

```
$\{G\}$
intialisation
$\{$invariant $P$; bound $bf\}$
;do $B \to \{P \land B \land bf = \mathsf{C}\}$ $S$ $\{P \land bf < \mathsf{C}\}$ od
$\{R\}$
```

Heuristic “Variable Introduction”: Only introduce variables based on some reason or derivation and *define* them by an invariant —not on a hunch, or by magic! Perhaps at the end of a derivation, introduce a variable to replace a reoccurring expression thereby improving clarity. See the heuristic of “Conflict Resolution”.

Heuristic “Deleting A Conjunct”: When postcondition R is of the form $P \land B$, and P is “easily” truthified —by, say, $x := I$ under precondition G— while B is not, then one may try to use P as invariant and the other as negation of the guard of a repetition, leading to

$$\{G\}\ x := I; \textbf{do}\ \neg B \to ?\ \textbf{od}\ \{R\}$$

For example, to calculate $q, r = A \,\mathsf{div}\, B, A \,\mathsf{mod}\, B$, we obtain algorithm:

$$\{A \geq 0 \land B > 0\}\ q, r := 0, A;\ \textbf{do}\ r \geq B \to q, r := q + 1, r - B\ \textbf{od}\ \{A = q * B + r \land 0 \leq r \ \land\ r < B\}$$

Heuristic “Replacing Constants/Expressions by Variables”: It may be possible to “work up/down to” the postcondition R, by replacing a constant/expression C with a variable c and placing bounds on it, —good candidates to try are the named parameters occurring in both G and R— thereby obtaining (integer) template

$$\{G\}\ c := ?; \{\mathsf{Invariant}\ R[C := c] \land 0 \leq c \leq C,\ \mathsf{bound}\ C - c\}, \textbf{do}\ C \neq c \to ?\ \textbf{od}\ \{R\}$$

For example, to calculate the ‘$\oplus$-sum/reduction’ of a sequence we obtain algorithm:

$$\{N \geq 0\}\ n, s := 0, \mathsf{e};\ \textbf{do}\ N \neq n \to n, s := n + 1, s \oplus f\ n\ \textbf{od}\ \{s = (\oplus i \mid 0 \leq i < N - 1 \bullet f\ i)\}$$

A instance of this heuristic is sufficiently common to merit its own name:

Heuristic “Conflict Resolution”: When choosing an invariant, if parts of the required goal can be easily truthified by different initalisations to a variable, then resolve such conflicts by introducing new additional variables for each such part that are defined to satisfy those parts. For example, to establish ‘$R\ :\ A\ x\ \land\ B\ x$’ when it is easy to show that ‘$A\ a$’ and ‘$B\ (f\ a)$’ are true, resolve this conflict of what ‘x’ ought to be by introducing a new variable y *defined* by the invariant $P\ :\ f\ x \leq y\ \land\ A\ x\ \land\ B\ y$; thereby yielding

$$\{0 \leq a\}\ x, y := a, f\ a; \{\text{inv}\ P,\ \text{bound}\ y - f\ x\}\ \textbf{do}\ f\ x \neq y \to ?\ \textbf{od}\ \{R\}$$

Heuristic “Syntactic Similarity”: When it’s not at all clear where to begin, attempt to massage the expressions G and R so that they are syntactically similar —after all, if they coincide then skip solves the problem. For example, if G contains a quantification but R does not, then introduce a quantification using the one-point rule then continue by using “Conflict Resolution”.

Metatheorem Witness:
If $\neg occurs$(‘X’, ‘S, R’), then:

$$\begin{array}{lll} & \{\exists\, \mathsf{X} \bullet G\}\ S\ \{R\} & \text{is a theorem} \\ \textit{iff} & \{G\}\ S\ \{R\} & \text{is a theorem} \end{array}$$

Theorem Weakening:

$$\begin{array}{ll} & \{G\}\ S\ \{\exists\, \mathsf{X} \bullet R\} \\ \Leftarrow & \{G\}\ S\ \{R\} \end{array}$$

Linear Search:
Provided $b : \mathbb{Z} \to \mathbb{B}$

```
$\{\ \exists \mathsf{X} : \mathbb{Z} \bullet 0 \leq \mathsf{X} \land b\,\mathsf{X}\ \}$
$x := 0$; do $\neg b\, x \to x := x + 1$ od
$\{\ x = (\downarrow\ i : \mathbb{Z} \mid 0 \leq i \land b\, i)\ \}$
```

Bounded Linear Search:
Provided $b : 0..N - 1 \to \mathbb{B}$

```
$\{\ 0 \leq N\ \}$
$x, y := 0, N$
;do $x \neq y \to$
    if  $\neg b\, x$  $\to$  $x := x + 1$
    $\Box$   $b\, x$  $\to$  $y := x$
    fi
 od
$\{\ x = (\uparrow\ i : 0..N \mid (\forall j : 0..i - 1 \bullet \neg b\, j))\ \}$
```

(Dual) Linear Search:
Provided $b : \mathbb{Z} \to \mathbb{B}$ *and* $N : \mathbb{Z}$

```
$\{\ \exists \mathsf{X} : \mathbb{Z} \bullet \mathsf{X} \leq N \land b\,\mathsf{X}\ \}$
$x := N$; do $\neg b\, x \to x := x - 1$ od
$\{\ x = (\uparrow\ i : \mathbb{Z} \mid i \leq N \land b\, i)\ \}$
```

(§4.1) **Heuristic “Binary Search”:** Whenever a given informal specification requests assigning to an integer variable such that it and its neighbour —‘$x + 1$’ or ‘$x - 1$’— satisfy some proposition, then tackle the problem by finding a suitable relation $\mathcal{Z}$ and using Binary Search (below right). Notice that $a < m < b$ is a precondition to the occurrence of m in the loop body and, with this, one may postulate “ficitous/ghost elements” to remove the precondition ‘$a\,\mathcal{Z}\,b$’ for certain problems —in left below, the sequence b is never inspected at $-1, N$ and so their values are completely irrelevant to the (outcome of the) computation: they are thought variables for reasoning only. Other problems may have the alternative’s guards simplified further by the particular choice of $\mathcal{Z}$ occurring in the invariant —see the left code snippet below! When using the results of an algorithm annotated with fictitious elements, one must check that the result is valid —otherwise we may have “out of bounds errors”.

Predicate Instance of Binary Search
Provided predicate b defined on $0..N-1$,

```
$\{\ 0 \leq N \land b\,(-1) \land \neg\, b\, N$
                      $b$ is fictitious on -1 and $N\ \}$
$x, y := -1, N$
;do $x + 1 \neq y \to$
   $m\ :=\ (x + y) \div 2$
   ;  if    $b\, m$    $\to$    $x := m$
      $\Box$   $\neg\, b\, m$   $\to$    $y := m$
      fi
 od
$\{\ -1 \leq x < N \land b\, x \land \neg\, b\,(x + 1)$
                      $b$ is fictitious on -1 and $N\ \}$
```

(General) Binary Search
Provided $\mathcal{Z}$ is a co-transitive relation,
$\forall x, y, m : \mathbb{Z} \bullet x\,\mathcal{Z}\,m \lor m\,\mathcal{Z}\,y \Leftarrow x\,\mathcal{Z}\,y$,

```
$\{\ a < b \land a\,\mathcal{Z}\,b\ \}$
$x, y := a, b$
;{ Invariant $a \leq x < y \leq b \land x\,\mathcal{Z}\,y$, Bound $y - x$
do $x + 1 \neq y \to$
   $m := (x + y) \div 2$
   $\{x < m < y\}$
   ;  if   $m\,\mathcal{Z}\,y$   $\to$   $x := m$
      $\Box$   $x\,\mathcal{Z}\,m$   $\to$   $y := m$
      fi
 od
$\{\ a \leq x < b \land x\,\mathcal{Z}\,(x + 1)\ \}$
```

Observe that the co-transitives are precisely the complements of retracts of transitives.

Absurdly fast searching! If $\neg b$ is monotonic, then the left snippet ensures, in logarithmic time, that $x\ =\ (\uparrow\ i : -1..N - 1 \mid b\, i)$ and this is far superior to Linear Search!
Or is it . . . What if we’re designing an algorithm to compute $\log_7 N$?