Homework 11 Key

Section 10.2

1. Which of the following ordered sets are complete partial orders:

   b) Divides ordering on {2,3,6,12,18}.

   Not a cpo because it has no bottom.

   c) Divides ordering on {2,4,6,8,10,12}.

   A cpo since 2 serves as bottom and every ascending chain has an lub because it is finite.

   d) Divides ordering on the set of positive integers.

   1 serves as bottom because 1 | p for all p P, but not a cpo because 2 4 8 16 32 64 … has no least upper bound in P.

   f) Divides ordering on the set P {1}.

   1 serves as bottom because 1 | p for all p P. No ascending chain has more than two different value (two levels) because a prime has only one divisor other than itself, so must repeat a value forever, and that value is the least upper bound.

   h) (subset) on the collection of all finite subsets of the natural numbers.

   serves as bottom, but

   {1} {1,2} {1,2,3} {1,2,3,4} {1,2,3,4,5} …

   is an ascending chain whose lub is not a finite subset of the natural numbers; so not a cpo.

   i) (subset) on the collection of all subsets of the natural numbers whose complement is finite.

   Not a cpo since is not cofinite. The least upper bound of an ascending chain may be computed by taking the union of the sets in the chain. If each set in the chain is cofinite, the union, which includes all the sets, must also be cofinite. Definition: S N is cofinite it N-S is a finite set.

5. Show that $_{A+B}$ is a partial order on A+B.

Reflexive

<a.1> <a,1> because a $_A$ a.

<b.2> <b,2> because b $_B$ b.

by definition.

Transitive

<a_1.1> <a_2,1> and <a_2.1> <a_3,1> imply <a_1.1> <a_3,1> because a_1 a_2 and a_2 a_3 imply a_1 a_3.

<b_1.2> <b_2,2> and <b_2.2> <b_3,2> imply <b_1.2> <b_3,2> because b_1 b_2 and b_2 b_3 imply b_1 b_3.

All other situations with x_1 x_2 and x_2 x_3 involve duplicate items or .

Antisymmetric

<a_1.1> <a_2,1> and <a_2.1> <a_1,1> imply <a_1.1> = <a_2,1> because a_1 a_2 and a_2 a_1 imply $a_1 = a_2$.

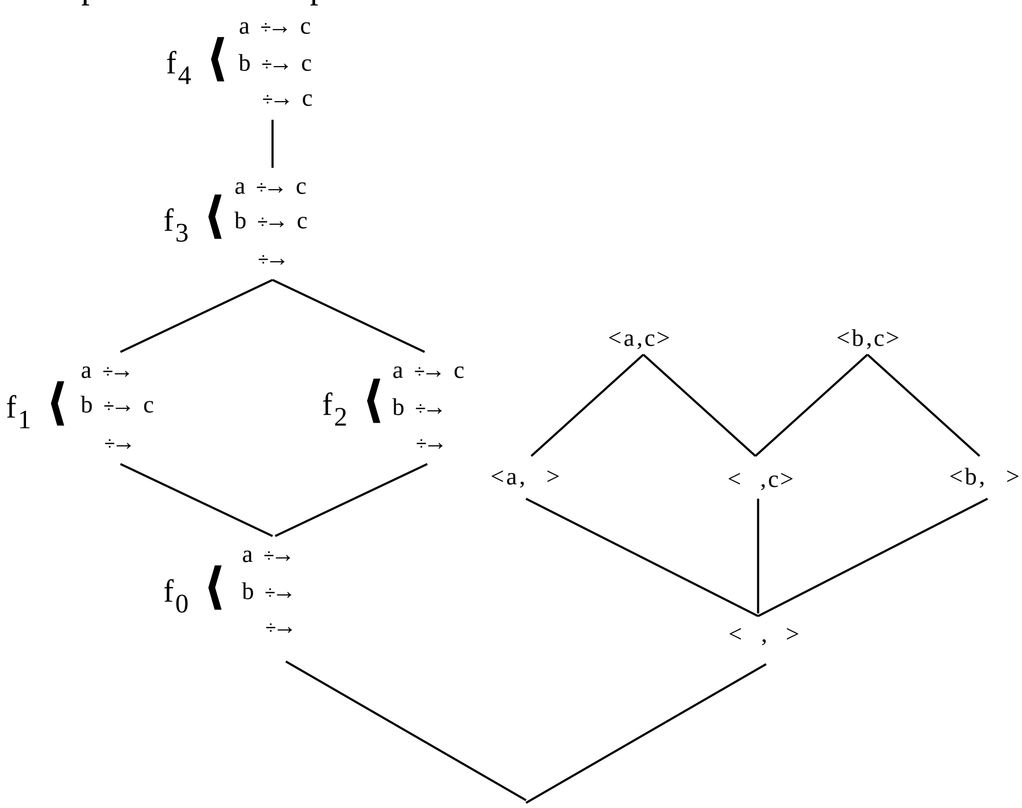
<b_1.2> <b_2,2> and <b_2.2> <b_1,2> imply <b_1.2> = <b_2,2> because b_1 b_2 and b_2 b_1 imply $b_1 = b_2$.

No other relations involving can go both ways unless both items are .

9. Suppose A = { ,a,b} and B = { ,c} are elementary domains.

   a) Sketch a Hasse diagram showing all the elements of (A→B)+(AxB) and their relationships. Represent functions as sets of ordered pairs.

      Since A→B and AxB are disjoint, omit the tags on the elements, but provide subscripts for the bottom elements.



b) Give one example of a function in Fun(A → B, AxB) that is continuous and one that is not monotonic.

Continuous: {<f_0,< , >>, <f_1,< ,c>>, <f_2,< ,a>>, <f_3,<a,c>>, <f_4,<a,c>>}

Not monotonic: {<f_0,<a,c>>, <f_1,< , >>, <f_2,< , >>, <f_3,< , >>, <f_4,< , >>}

12. Prove that *inS* and *outA* are continuous functions.

Let a_1 a_2 a_3 … be an ascending chain in domain A.
Observe that 1 1 1 1 … is an ascending chain in N.
Then <a_1,1> <a_2,1> <a_3,1> … is an ascending chain in S.
So *inS*(*lub*{a_i | i≥1}) = <*lub*{a_i | i≥1}, 1> = <*lub* {a_i | i≥1}, *lub* {1 | i≥1}>
= *lub*{<a_i,1> | i≥1} = *lub*{*inS*(a_i) | i≥1}.

Let s_1 s_2 s_3 … be an ascending chain in S.
Case 1: s_i = $_S$ for all i≥1.
Then *outA* (*lub*{s_i | i≥1}) =*outA* $_S$ = $_A$
and *lub* { *outA* s_i | i≥1} = *lub* { $_A$ | i≥1} = $_A$.
Case 2: For some k≥1, s_i = <a_i,1> for all i≥k where a_i A.
Then *outA* (*lub*{s_i | i≥1}) = *outA* <*lub* {a_i | i≥k},2> = *lub* {a_i | i≥k}
and *lub* { *outA* s_i | i≥1} = *lub* {a_i | i≥k} = *lub* {a_i | i≥k}.
Case 3: For some k≥1, s_i = <b_i,2> for all i≥k where b_i B.
Then *outA* (*lub*{s_i | i≥1}) = *outA* <*lub* {b_i | i≥k},2> = $_A$
and *lub* { *outA* s_i | i≥k} = *lub* { $_A$ | i≥k} = $_A$.

14. Tell whether these functions F : (N → N) → (N → N) are monotonic and / or continuous.

a) F g n = if total(g) then g(n) else where total(g) is true if and only if g(n) is defined (not) for all n N.

F is monotonic:
Let g_1,g_2 : N → N satisfy g_1 g_2.
Case 1: g_1 is total
Since g_1 g_2 and N is a flat domain, g_2 is also total, and so g_1 = g_2.
Therefore F g_1 = F g_2.
Case 2: g_1 is not total.
Subcase 2a: g_2 is total.

Then F g_1 = λn.   and F g_2 = g_2. So F g_1 F g_2.

Subcase 2b: g_2 is not total.

Then F g_1 = λn.   and F g_2 = λn. . So F g_1 F g_2.

F is not continuous:

Define g_i n = if n≤i then n else   .

So g_1 g_2 g_3 g_4 … is an ascending chain in N → N.

Since no g_i is total, for each i, F g_i = λn. , and so *lub*{F g_i} = λn. .

But *lub*{g_i} = λn.n and F(*lub*{g_i}) = λn.n.

b) F g n = if g = (λn .0) then 1 else 0

F is neither monotonic nor continous:

Define g_1 = λn.   and g_2 = λn.0. Then g_1 g_2.

But F g_1 = λn.0, F g_2 = λn.1, and these two functions are incomparable.

Since F is not monotonic, it is also not continous.

c) F g n = if n   dom(g) then 0 else   where dom(g) denotes the domain of g.

F is neither monotonic nor continous:

Define g_1 = λn.   and g_2 = λn.5. Then g_1 g_2.

But F g_1 = λn.0, F g_2 = λn. , and not(F g_1 F g_2).

Since F is not monotonic, it is also not continous.

16. Consider the function F : (N → N) → (N → N) defined by
for g   N → N, F g = λn . if g(n)=   then 0 else 1
Describe F g_1, F g_2, and F g_3 where the g_k : N → N are defined by

g_1(n) = n

g_2(n) = if n>0 then n / 0 else

g_3(n) = if *even*(n) then n+1 else

If g_1(n) = n then F g_1 = λn . 1

If g_2(n) = if n>0 then n / 0 else   then F g_2 = λn . 0

If g_3(n) = if *even*(n) then n+1 else   then F g_3 = λn . if *even*(n) then 1 else 0

Section 10.3

7. Find a simple (nonrecursive) definition of these functions in N → N using a fixed point construction.

b) h(n) = if n=0 then 0 else if n=1 then h(n+1)–1 else h(n–1)+1

Functional:

H h n = if n=0 then 0 else if n=1 then h(n+1)–1 else h(n–1)+1

$h_0(n)$ = (n) =

$h_1(n)$ = H h_0 n

= if n=0 then 0 else if n=1 then $h_0(n+1)$–1 else $h_0(n–1)$+1

= if n=0 then 0 else if n=1 then (n+1)–1 else (n–1)+1

= if n=0 then 0 else

$h_2(n)$ = H h_1 n

= if n=0 then 0 else if n=1 then $h_1(n+1)$–1 else $h_1(n–1)$+1

= if n=0 then 0 else if n=1

then (if n+1=0 then 0 else)–1 else (if n-1=0 then 0 else)+1

= if n=0 then 0

else if n=1

then (if n=-1 then 0 else)–1 else (if n=1 then 0 else)+1

= if n=0 then 0 else if n=1 then else

= if n=0 then 0 else

Conjecture: $h_i(n)$ = if n=0 then 0 else

Proof similar to calculation of h_2.

Least Fixed Point: *lub* { h_i | i≥0 } = λn . if n=0 then 0 else

Note: g(n)=n is another fixed point for the functional H.

c) f(n) = if n=0 then 0 else if n=1 then f(n-1)+1 else n^2

Functional: F f n = if n=0 then 0 else if n=1 then f(n-1)+1 else n^2

$f_0(n)$ = (n) =

$f_1(n)$ = F f_0 n

= if n=0 then 0 else if n=1 then $f_0(n-1)$+1 else n^2

= if n=0 then 0 else if n=1 then (n-1)+1 else n^2

= if n=0 then 0 else if n=1 then else n^2

$f_2(n)$ = F f_1 n

= if n=0 then 0 else if n=1 then $f_1(n-1)+1$ else n^2

= if n=0 then 0 else if n=1 then $f_1(0)+1$ else n^2

= if n=0 then 0 else if n=1 then 0+1 else n^2

= if n=0 then 0 else if n=1 then 1 else n^2

= n^2

$f_3(n)$ = F f_2 n

= if n=0 then 0 else if n=1 then $f_2(n-1)+1$ else n^2

= if n=0 then 0 else if n=1 then $f_2(0)+1$ else n^2

= if n=0 then 0 else if n=1 then 0^2+1 else n^2

= if n=0 then 0 else if n=1 then 1 else n^2

= n^2

Conjecture: For i≥2, $f_i(n) = n^2$

Proof similar to calculation of f_3.

Least Fixed Point: *lub* $\{ f_i \mid i \geq 0 \} = \lambda n \,.\, n^2$

e) h(n) = if n=0 then 1 else if n=1 then 2 else 4n–4+h(n–2)

Functional: H h n = if n=0 then 1 else if n=1 then 2 else 4n–4+h(n–2)

$h_0(n)$ = (n) =

$h_1(n)$ = H h_0 n

= if n=0 then 1 else if n=1 then 2 else 4n–4+$h_0(n–2)$

= if n=0 then 1 else if n=1 then 2 else 4n–4+

= if n=0 then 1 else if n=1 then 2 else

$h_2(n)$ = H h_1 n

= if n=0 then 1 else if n=1 then 2 else 4n–4+$h_1(n–2)$

= if n=0 then 1
else if n=1 then 2
else 4n–4+(if n–2=0 then 1 else if n–2=1 then 2 else)

= if n=0 then 1
else if n=1 then 2
else if n=2 then 4n–4+1 else if n=3 then 4n–4+2 else 4n–4+

= if n=0 then 1
else if n=1 then 2
else if n=2 then 4n–3 else if n=3 then 4n–2 else

$h_3(n)$ = H h_2 n

= if n=0 then 1 else if n=1 then 2 else 4n–4+h_2(n–2)

= if n=0 then 1 else if n=1 then 2
else 4n–4+(if n–2=0 then 1
else if n–2=1 then 2
else if n–2=2 then 4(n–2)–3
else if n–2=3 then 4(n–2)–2 else)

= if n=0 then 1 else if n=1 then 2
else (if n=2 then 4n–4+1
else if n=3 then 4n–4+2
else if n=4 then 4n–4+4(n–2)–3
else if n=5 then 4n–4+4(n–2)–2 else 4n–4+)

= if n=0 then 1 else if n=1 then 2
else if n=2 then 4n–3
else if n=3 then 4n–2
else if n=4 then 8n–15
else if n=5 then 8n–14 else

Conjecture: h_i(n) = if n<2i then n^2+1 else

Basis: h_0(n) = (n) =

Induction: Suppose h_i(n) = if n<2i then n^2+1 else .

Then

h_{i+1}(n) = H h_i n

= if n=0 then 1 else if n=1 then 2 else 4n–4+h_i(n–2)

= if n=0 then 1
else if n=1 then 2
else 4n–4+(if n-2<2i then $(n–2)^2$+1 else)

= if n=0 then 1
else if n=1 then 2
else (if n<2i+2 then 4n–4+$(n–2)^2$+1 else 4n–4+)

= if n=0 then 1
else if n=1 then 2
else if n<2i+2 then n^2+1 else

= if n<2(i+1) then n^2+1 else

Least Fixed Point: *lub* { h_i | i≥0 } = λn . n^2+1

f) f(n) = if n=0 then f(n+1)+1 else 1

Functional: F f n = if n=0 then f(n+1)+1 else 1

f_0(n) = (n) =

$f_1(n) = F\ f_0\ n$

$\quad = \text{if } n=0 \text{ then } f_0(n+1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } \quad (n+1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } \quad \text{ else } 1$

$f_2(n) = F\ f_1\ n$

$\quad = \text{if } n=0 \text{ then } f_1(n+1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then (if } n+1=0 \text{ then } \quad \text{ else } 1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } 1+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } 2 \text{ else } 1$

$f_3(n) = F\ f_2\ n$

$\quad = \text{if } n=0 \text{ then } f_2(n+1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then (if } n+1=0 \text{ then } 2 \text{ else } 1)+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } 1+1 \text{ else } 1$

$\quad = \text{if } n=0 \text{ then } 2 \text{ else } 1$

$\quad = f_2(n)$, the least fixed point for F.

8. Consider the following functional defined on functions over the natural numbers:

$G : (N \rightarrow N) \rightarrow (N \rightarrow N)$

$G = \lambda g\ .\ \lambda n\ .\ \textbf{if } n=0 \textbf{ then } 2 \textbf{ else } g(n)$

a) Give and justify a recursive definition that corresponds to this functional—that is, an operational definition of a function that will be a fixed point of G.

b) Define four different functions, g_0, g_1, g_2, and g_3, that are fixed points of the functional G, including the least fixed point, g_0. Carefully prove that g_0 and g_1 are fixed points of G.

g n = G g n = if n=0 then 2 else g(n) iff g n = G g n = if n=0 then 2 else g(n)

$g_0 = \lambda n\ .\ \text{if } n=0 \text{ then } 2 \text{ else }$

$g_1 = \lambda n\ .\ \text{if } n=0 \text{ then } 2 \text{ else } 1$

$g_2 = \lambda n\ .\ \text{if } n=0 \text{ then } 2 \text{ else } 2$

$g_3 = \lambda n\ .\ \text{if } n=0 \text{ then } 2 \text{ else } 3$

$G\ g_0\ n = \text{if } n=0 \text{ then } 2 \text{ else } g_0(n)$

$\quad = \text{if } n=0 \text{ then } 2 \text{ else (if } n=0 \text{ then } 2 \text{ else } \quad)$

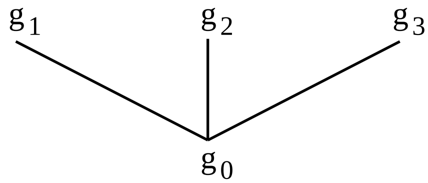
$\quad = \text{if } n=0 \text{ then } 2 \text{ else } \quad = g_0\ n$

$G\ g_1\ n = \text{if } n=0 \text{ then } 2 \text{ else } g_1(n)$

$\quad = \text{if } n=0 \text{ then } 2 \text{ else (if } n=0 \text{ then } 2 \text{ else } 1)$

$\quad = \text{if } n=0 \text{ then } 2 \text{ else } 1 = g_1\ n$

c) Draw a diagram showing the relationship “is less defined or equal” between these four functions.



d) Informally describe the operational behavior of the recursive definition in part a). Which of the four fixed-point functions has the closest behavior to the operational view?

The operational behavior suggests that g maps 0 to 2 and is nonterminating for all n>0. This is also the behavoir of g_0.

9. Let T = { ,true,false} be the elementary domain of Boolean values with the bottom element . The function *and* : T x T → T must agree with the following truth table:

and	true	false	
true	true	false	?
false	false	false	?
	?	?	?

Complete this truth-table in *two* ways to produce two different monotonic versions of the function *and* defined on T**.** Explain how these two *and* functions correspond to the possible interpretations of the predefined Boolean **and** function in a programming language such as Pascal.

Since *and*(true,) *and*(true,true) = true, and
and(true,) *and*(true,false) = false,
and(true,) must be .

Similarly, *and*(,) = *and*(,true) = .

On the other hand, *and*(false,) and *and*(,false) may be or false.

First Possibility:

and	true	false	
true	true	false	
false	false	false	

Strict and: Implies that both arguments are evaluated.

Second Possibility:

and	true	false	
true	true	false	

Conditional and: Implies that

false	false	false	false	if first (or second) argument
		false		evaluates to false, the other need not be evaluated.

Note: Combinations of these two possibilities are also allowed.

Section 10.4

4. Using the following definitions, calculate fib 4 by applying the Fixed-Point Identity.

   G = λg . λn . if n=0 then 0 else if n=1 then 1 else g(n-1) + g(n-2)

   fib = *fix* G = *fix* (λg . λn . if n<2 then n else g(n-1) + g(n-2)).

   (*fix* G) 4

   (G (*fix* G)) 4

   (λg . λn . if n<2 then n else g(n-1) + g(n-2)) (*fix* G) 4

   (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 4

   if 4<2 then 4 else (*fix* G)(4-1) + (*fix* G)(4-2)

   (*fix* G) 3) + (*fix* G) 2

   (G (*fix* G)) 3) + (G (*fix* G)) 2

   (λg . λn . if n<2 then n else g(n-1) + g(n-2) (*fix* G)) 3)
   + (λg . λn . if n<2 then n else g(n-1) + g(n-2) (*fix* G)) 2

   (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 3
   + (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 2

   if 3<2 then 3 else (*fix* G)(3-1) + (*fix* G)(3-2)
   + if n<2 then 2 else (*fix* G)(2-1) + (*fix* G)(2-2)

   if 3<2 then n else (*fix* G)(2) + (*fix* G)(1)
   + if 2<2 then n else (*fix* G)(1) + (*fix* G)(0)

   (*fix* G) 2 + (*fix* G) 1 + (*fix* G) 1 + (*fix* G) 0

   (G (*fix* G)) 2 + (G (*fix* G)) 1 + (G (*fix* G)) 1 + (G (*fix* G)) 0

   (λg . λn . if n<2 then n else g(n-1) + g(n-2)) (*fix* G) 2
   + (λg . λn . if n<2 then n else g(n-1) + g(n-2) (*fix* G)) 1
   + (λg . λn . if n<2 then n else g(n-1) + g(n-2) (*fix* G)) 1
   + (λg . λn . if n<2 then n else g(n-1) + g(n-2) (*fix* G)) 0

   (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 2
   + (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 1
   + (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 1
   + (λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 0

   if 2<2 then 2 else (*fix* G)(2-1) + (*fix* G)(2-2)
   + if 1<2 then 1 else (*fix* G)(1-1) + (*fix* G)(1-2)
   + (if 1<2 then 1 else (*fix* G)(1-1) + (*fix* G)(1-2)

+ if 0<2 then 0 else (*fix* G)(0-1) + (*fix* G)(0-2))

((*fix* G) 1) + ((*fix* G) 0) + 1 + 1 + 0

((G (*fix* G)) 1) + ((G (*fix* G)) 0) + 2

(λg . λn . if n<2 then n else g(n-1) + g(n-2)) (*fix* G) 1

+ (((λg . λn . if n<2 then n else g(n-1) + g(n-2)) (*fix* G)) 0) + 2

(λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2)) 1

+ (((λn . if n<2 then n else (*fix* G)(n-1) + (*fix* G)(n-2))) 0) + 2

if 1<2 then 1 else (*fix* G)(1-1) + (*fix* G)(1-2)

+ ((if 0<2 then 0 else (*fix* G)(0-1) + (*fix* G)(0-2))) + 2

1 + 0 + 2   3