Domain Theory

Recursive Definitions

f(n) = if n=0 then 1 else f(n–1)

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

or

define f = λn . (if (zerop n) 1 (f (sub n 1)))

define g = λn . (if (zerop n) 1 (g (succ n)))

A function satisfies a recursive definition iff it is a solution to an equation:

f = λn . (if (zerop n) 1 (f (sub n 1)))

g = λn . (if (zerop n) 1 (g (succ n)))

Similar to solving a mathematical equation:

$$x = x^2 - 4x + 6.$$

Other Recursive Definitions

Concrete Syntax

<cmd> ::= **if** <boolean expr>
then <cmd seq> **end if**

<cmd seq> ::= <cmd>
| <cmd> **;** <cmd seq>

Lists of Numbers

List = {*nil*} ∪ (N x List)
where *nil* represents the empty list

Model for Pure Lambda Calculus

V = set of variables

D = V ∪ (D → D)

Problem with Cardinality

$|D \rightarrow D| \leq |D| < |\mathcal{P}(D)| \leq |D \rightarrow D|$

Modeling Nontermination

Domains

Sets with a lattice-like structure.

Each domain contains a bottom element ⊥ that is “less than” all other elements.

For domains of functions, bottom represents a computation that fails to complete normally.

Partial Order ⊆ on a Set S

A relation that is

- reflexive
- transitive
- antisymmetric

Definitions

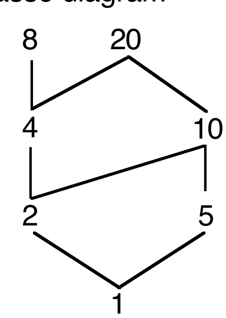
b∈S is a **lower bound** of a subset A of S if b⊆x for all x∈A.

u∈S is an **upper bound** of a subset A of S if x⊆u for all x∈A.

A **least upper bound** of A, *lub* A, is an upper bound of A that is less than or equal every upper bound of A.

Example: Divides relation on { 1,2,4,5,8,10,20 }

Hasse diagram



lub { 2, 5 } =

lub { 2, 4, 5, 10 } =

lub { 1, 2, 4 } =

lub { 8, 10 } =

lub { 20 } =

An **ascending chain** in a partially ordered set S is a sequence of elements $\{ x_1, x_2, x_3, x_4, \ldots \}$ with the property

$$x_1 \subseteq x_2 \subseteq x_3 \subseteq x_4 \subseteq \ldots.$$

A **complete partial order** (**cpo**) on a set S is a partial order $\subseteq$ with the two properties

a) There is an element $\bot \in S$ with $\bot \subseteq x$ for all $x \in S$.

b) Every ascending chain in S has a least upper bound in S.

On domains, $\subseteq$ is thought of as **approximates** or **is less defined than or equal to**.

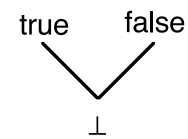
Any finite set with a partial order and a bottom element $\bot$ is a cpo. Why?

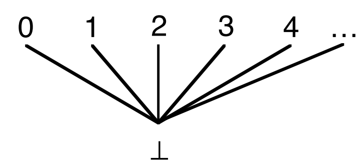
Elementary Domains

Natural numbers and Boolean values with a **discrete partial order**:

for $x,y \in S$, $x \subseteq y$ iff $x = y$ or $x = \bot$.

Elementary domains correspond to “answers”, the results produced by programs.





Proper and Improper values.

Also called **flat domains**.

Product Domains

If A with ordering $\subseteq_A$ and B with ordering $\subseteq_B$ are complete partial orders, the **product domain** of A and B is AxB with the ordering $\subseteq_{AxB}$ where

$AxB = \{\langle a,b\rangle \mid a \in A \text{ and } b \in B\}$, and

$\langle a,b\rangle \subseteq_{AxB} \langle c,d\rangle$ iff $a \subseteq_A c$ and $b \subseteq_B d$.

Thm: $\subseteq_{AxB}$ is a partial order on AxB.
Proof: Exercise

Thm: $\subseteq_{AxB}$ is a complete partial order on AxB.
Proof: $\bot_{AxB} = \langle \bot_A, \bot_B\rangle$ acts as bottom for AxB, since $\bot_A \subseteq_A a$ and $\bot_B \subseteq_B b$ for $a \in A$ and $b \in B$.

If $\langle a_1,b_1\rangle \subseteq \langle a_2,b_2\rangle \subseteq \langle a_3,b_3\rangle \subseteq \ldots$ is an ascending chain in AxB,

then $a_1 \subseteq_A a_2 \subseteq_A a_3 \subseteq_A \ldots$ is a chain in A with least upper bound, $lub\{a_i \mid i \geq 1\} \in A$, and

$b_1 \subseteq_B b_2 \subseteq_B b_3 \subseteq_B \ldots$ is a chain in B with least upper bound, $lub\{b_i \mid i \geq 1\} \in B$.

Therefore, $\langle lub\{a_i \mid i \geq 1\}, lub\{b_i \mid i \geq 1\}\rangle \in AxB$ is the least upper bound for original chain. ∎

Example

Level = $\{\bot_L$,undergraduate,graduate,nondegree$\}$

and

Gender = $\{\bot_G$, female, male$\}$

Gender x Level

$\langle f,u\rangle$ $\langle f,g\rangle$ $\langle f,n\rangle$

Imagine two processes to determine level and gender of a student.

Projection Functions

first : AxB→A
defined by *first* <a,b> = a for any <a,b>∈AxB

and

second : AxB→B
defined by *second* <a,b> = b for any <a,b>∈AxB

Generalize to arbitrary product domains:

$D_1 \times D_2 \times \ldots \times D_n$

Application: Calculator Semantics

evaluate ⟦=⟧ (a,op,d,m) = (a, *nop*, *op*(a,d), m)

is a more readable translation of

evaluate ⟦=⟧ st =
(*first*(st), *nop*,
second(st)(*first*(st),*third*(st)), *fourth*(st)).

Sum Domains

If A with ordering $\subseteq_A$ and B with ordering $\subseteq_B$ are complete partial orders, the **sum domain** of A and B is A+B with the ordering $\subseteq_{A+B}$ where

$A+B = \{\langle a,1\rangle \mid a \in A\} \cup \{\langle b,2\rangle \mid b \in B\} \cup \{\perp_{A+B}\}$,

$\langle a,1\rangle \subseteq_{A+B} \langle c,1\rangle$ if $a \subseteq_A c$,

$\langle b,2\rangle \subseteq_{A+B} \langle d,2\rangle$ if $b \subseteq_B d$,

$\perp_{A+B} \subseteq_{A+B} \langle a,1\rangle$ for each $a \in A$,

$\perp_{A+B} \subseteq_{A+B} \langle b,2\rangle$ for each $b \in B$, and

$\perp_{A+B} \subseteq_{A+B} \perp_{A+B}$.

Thm: $\subseteq_{A+B}$ is a complete partial order on A+B.

Proof: $\perp_{A+B} \subseteq x$ for any $x \in A+B$ by definition. An ascending chain $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ in A+B may repeat $\perp_{A+B}$ forever or eventually climb into either Ax{1} or Bx{2}. In first case, least upper bound will be $\perp_{A+B}$, and in the other two cases the least upper bound will exist in A or B. ∎

Functions on Sum Domains

Let S = A+B.

1. **Injection** (creation):

inS : A→S is defined for a∈A
as *inS* a = <a,1>∈S

inS : B→S is defined for b∈B
as *inS* b = <b,2>∈S

2. **Projection** (selection):

outA : S→A is defined for s∈S as
outA s = a∈A if s=<a,1>, and
outA s = $\perp_A$∈A if s=<b,2> or s=$\perp_S$.

outB : S→B is defined for s∈S as
outB s = b∈B if s=<b,2>, and
outB s = $\perp_B$∈B if s=<a,1> or s=$\perp_S$.

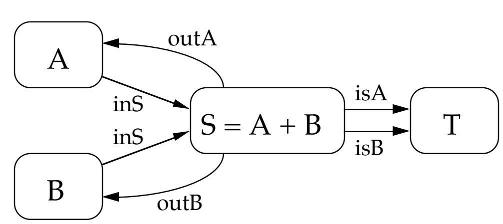
3. **Inspection** (testing):
Recall T = {true, false, $\perp_T$}.

isA : S→T is defined for s∈S as
isA s iff there exists a∈A with s=<a,1>.

isB : S→T is defined for s∈S as
isB s iff there exists b∈B with s=<b,2>.

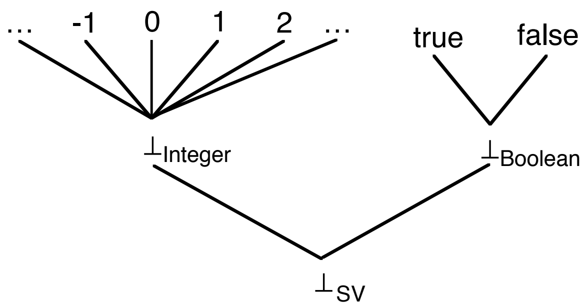
In both cases, $\perp_S$ is mapped to $\perp_T$.

Signature Diagram



Observe that inS is an overloaded function.

Storable Values for Wren



Functions for SV

inSV : Integer $\rightarrow$ SV
inSV : Boolean $\rightarrow$ SV
outInteger : SV $\rightarrow$ Integer
isInteger : SV $\rightarrow$ T
outBoolean : SV $\rightarrow$ Boolean
isBoolean : SV $\rightarrow$ T

SV = Integer + Boolean =
{<n,1>| n∈Integer}∪{<b,2>| b∈Boolean}∪{$\perp_{SV}$}

or

SV = *int*(Integer) + *bool*(Boolean) =
{*int*(n)| n∈Integer}∪{*bool*(b)| b∈Boolean}∪{$\perp_{SV}$}

Injection function

inSV : Integer $\rightarrow$ SV where *inSV* n = *int*(n).

The tags (constructors), *int* and *bool*, take the place of the overloaded injection function, *inSV*.

int : Integer $\rightarrow$ SV
bool : Boolean $\rightarrow$ SV

Projection function

outInteger : SV $\rightarrow$ Integer
where *outInteger int*(n) = n
outInteger bool(b) = ⊥
outInteger $\perp_{SV}$ = ⊥.

Inspection handled by pattern matching

execute ⟦**if** E **then** C⟧ sto =
if p then *execute* ⟦C⟧ sto else sto
where *bool*(p) = *evaluate* ⟦E⟧ sto,

stands for

execute ⟦**if** E **then** C⟧ sto =
if *isBoolean*(val)
then if *outBoolean*(val)
then *execute* ⟦C⟧ sto
else sto
else ⊥
where val = *evaluate* ⟦E⟧ sto

Generalizations

Finite sums: $D_1 + D_2 + D_3 + \ldots + D_n$
Infinite sums: $D_1 + D_2 + D_3 + \ldots = \{<d,i> \mid d \in D_i\}$

Domain of finite Sequences:

$D^* = \{ nil \} + D + D^2 + D^3 + D^4 + \ldots$
where *nil* represents the empty sequence.

Functions on D^*

Injection: $inD^* : D^k \rightarrow D^*$

Projection: $outD^k : D^* \rightarrow D^k$

Functions on Lists:

Let $L \in D^*$ and $e \in D$.
Then $L=<d,k>$ for $d \in D^k$ for some $k \geq 0$
where $D^0 = \{ nil \}$.

1. *head* : $D^* \rightarrow D$ where
head (L) = *first* ($outD^k$(L)) if k>0, and
head (<*nil*,0>) = ⊥.

2. *tail* : $D^* \rightarrow D^*$ where
tail (L) = inD^*(<*2nd* ($outD^k$(L)),
3rd ($outD^k$(L)), …,
kth ($outD^k$(L))>)
if k>0, and
tail (<*nil*,0>) = ⊥.

3. *null* : $D^* \to T$ where
null (<*nil*,0>) = true, and
null (L) = false if L = <d,k> with k>0.

Therefore, *null* (L) = isD^0(L)

4. *prefix* : $DxD^* \to D^*$ where
prefix (e,L) = *inD**(<e,*1st* ($outD^k$(L)), *2nd* ($outD^k$(L)),…, *kth* ($outD^k$(L))>)

5. *affix* : $D^*xD \to D^*$ where
affix (L,e) = *inD**(<*1st* ($outD^k$(L)), *2nd* ($outD^k$(L)), …, *kth* ($outD^k$(L)),e>)

Each of these five functions map bottom to bottom.

The binary functions *prefix* and *affix* produce $\perp$ if either argument is bottom.

Sets of Functions

A function from a set A to a set B is **total** if f(x)∈B is defined for every x∈A.

If A with ordering $\subseteq_A$ and B with ordering $\subseteq_B$ are complete partial orders, define **Fun(A,B)** to be the set of all total functions from A to B.

Define $\subseteq$ on Fun(A,B) as follows:

For f,g∈Fun(A,B),
$f \subseteq g$ if $f(x) \subseteq_B g(x)$ for all x∈A.

Lemma: $\subseteq$ is a partial order on Fun(A,B).

Proof: Follow the definition to show reflexive, transitive, and antisymmetric.

See text for complete proof.

Thm: $\subseteq$ is a complete partial order on Fun(A,B).

Proof: Define bottom for Fun(A,B) as the function $\perp(x) = \perp_B$ for all x∈A.
Since $\perp(x) = \perp_B \subseteq_B f(x)$ for all x∈A and f∈Fun(A,B),
$\perp \subseteq f$ for all f∈Fun(A,B).

Let $f_1 \subseteq f_2 \subseteq f_3 \subseteq \ldots$ be an ascending chain in Fun(A,B).

Then for any x∈A, $f_1(x) \subseteq_B f_2(x) \subseteq_B f_3(x) \subseteq_B \ldots$ is a chain in B, which has a least upper bound, y_x∈B. Note that y_x is *lub* $\{f_i(x) \mid i \geq 1\}$.

Define the function F(x) = y_x for each x∈A.

F serves as a least upper bound for the original chain. Set *lub* $\{f_i \mid i \geq 1\}$ = F. ∎

Fun(A,B) contains some strange functions.

Consider F ∈ Fun(N→N,N→N) defined by

F g = λn . if g(n)=⊥ then 0 else 1,
for g∈N→N

Restrictions on Fun(A,B)

Monotonic

A function f in Fun(A,B) is **monotonic** if $x \subseteq_A y$ implies $f(x) \subseteq_B f(y)$ for all x,y∈A.

If $\subseteq$ means "approximates", then when y has at least as much information as x, it follows that f(y) has at least as much information as f(x).

Continous

A function f∈Fun(A,B) is **continuous** if it preserves least upper bounds; that is, if $X = x_1 \subseteq_A x_2 \subseteq_A x_3 \subseteq_A \ldots$ is an ascending chain in A, then $f(lub_A X) = lub_B\{f(x) \mid x \in X\}$.

Also written $f(lub_A\{x_i\}) = lub_B\{f(x_i)\}$
or $f(lub_A\{x_i \mid i \geq 1\}) = lub_B\{f(x_i) \mid i \geq 1\}$.

No surprises when taking the least upper bounds (limits) of approximations

Lemma: If $f \in Fun(A,B)$ is continuous, then it is monotonic.

Proof: Suppose f is continuous and $x \subseteq_A y$. Then $x \subseteq_A y \subseteq_A y \subseteq_A y \subseteq_A \ldots$ is an ascending chain in A, and since f is continuous,

$$f(x) \subseteq_B lub_B\{f(x),f(y)\} = f(lub_A\{x,y\}) = f(y). \quad \blacksquare$$

Function Domains

Define **A→B** to be the set of functions in Fun(A,B) that are (monotonic and) continuous. This set is ordered by the relation $\subseteq$ from Fun(A,B).

$F \in Fun(N \to N, N \to N)$ defined by

$F\ g = \lambda n\ .\ \text{if } g(n)=\perp \text{ then } 0 \text{ else } 1,$ for $g \in N \to N$

is not monotonic.

Proof by Counterexample:

Let $g_1 = \lambda n\ .\ \perp$ and $g_2 = \lambda n\ .\ 0$. Then $g_1 \subseteq g_2$. But $F(g_1) = \lambda n\ .\ 0$, $F(g_2) = \lambda n\ .\ 1$, and functions $\lambda n\ .\ 0$ and $\lambda n\ .\ 1$ are not related at all by $\subseteq$.

Lemma: The relation $\subseteq$ restricted to A→B is a partial order.

Proof: The properties reflexive, transitive, and antisymmetric are inherited by a subset. $\blacksquare$

Lub Lemma: If $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ is an ascending chain in a cpo A, and $x_i \subseteq d \in A$ for each $i \geq 1$, then $lub\{x_i | i \geq 1\} \subseteq d$.

Proof: By the definition of least upper bound, if d is a bound for the chain, the least upper bound, $lub\{x_i | i \geq 1\}$, must be no larger than d. $\blacksquare$

Limit Lemma: If $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ and $y_1 \subseteq y_2 \subseteq y_3 \subseteq \ldots$ are ascending chains in cpo A, and $x_i \subseteq y_i$ for each $i \geq 1$, then $lub\{x_i | i \geq 1\} \subseteq lub\{y_i | i \geq 1\}$.

Proof: For each $i \geq 1$, $x_i \subseteq y_i \subseteq lub\{y_i | i \geq 1\}$. Therefore $lub\{x_i | i \geq 1\} \subseteq lub\{y_i | i \geq 1\}$ by the Lub lemma (take $d = lub\{y_i | i \geq 1\}$). $\blacksquare$

Thm: The relation $\subseteq$ on A→B is a complete partial order.

Proof: Since $\subseteq$ is a partial order on A→B, two properties need to be verified:

1. The bottom element in Fun(A,B) is also in A→B; that is, the function $\perp(x) = \perp_B$ is monotonic and continuous.
2. For any ascending chain in A→B, its least upper bound, which is an element of Fun(A,B), is also in A→B, namely it is monotonic and continuous.

Part 1: If $x \subseteq_A y$ for some $x,y \in A$, then $\perp(x) = \perp_B = \perp(y)$, which means $\perp(x) \subseteq_B \perp(y)$, and so $\perp$ is a monotonic function.

If $x_1 \subseteq_A x_2 \subseteq_A x_3 \subseteq_A \ldots$ is an ascending chain in A, then its image under the function $\perp$ will be the ascending chain $\perp_B \subseteq_B \perp_B \subseteq_B \perp_B \subseteq_B \ldots$, whose least upper bound is $\perp_B$. Therefore, $\perp(lub_A\{x_i | i \geq 1\}) = \perp_B = lub_B\{\perp(x_i) | i \geq 1\}$, and $\perp$ is a continuous function.

Part 2: Let $f_1 \subseteq f_2 \subseteq f_3 \subseteq \ldots$ be an ascending chain in A→B, and let $F = lub\{f_i | i \geq 1\}$ be its least upper bound (in Fun(A,B)). Remember the definition of F, $F(x) = lub\{f_i(x) | i \geq 1\}$ for each $x \in A$. We need to show that F is monotonic and continuous so that we know F is a member of A→B.

Monotonic:
If $x \subseteq_A y$, $f_i(x) \subseteq_B f_i(y) \subseteq_B lub\{f_i(y) | i \geq 1\}$ for any i since each f_i is monotonic.

Therefore, $F(y) = lub\{f_i(y) | i \geq 1\}$ is an upper bound for each $f_i(x)$, and so the least upper bound of all the $f_i(x)$ satisfies $F(x) = lub\{f_i(x) | i \geq 1\} \subseteq F(y)$ (Lub lemma), and F is monotonic.

Continuous: Let $x_1 \subseteq_A x_2 \subseteq_A x_3 \subseteq_A \ldots$ be an ascending chain in A. We need to show that $F(lub\{x_j | j \geq 1\}) = lub\{F(x_j) | j \geq 1\}$ where

$$F(x) = lub\{f_i(x) | i \geq 1\} \text{ for each } x \in A.$$

Note that "i" is used to index the ascending chain of functions from A→B while "j" is used to index the ascending chains of elements in A and B.

So F is continuous if

$$F(lub\{x_j|j\geq1\}) = lub\{F(x_j)|j\geq1\}.$$

Recall these definitions and properties:

1. Each f_i is continuous:
$$f_i(lub\{x_j|j\geq1\}) = lub\{f_i(x_j)|j\geq1\}$$
for each chain $\{x_j|j\geq1\}$ in A.

2. Definition of F:
$$F(x) = lub\{f_i(x)|i\geq1\} \text{ for each } x\in A.$$

So

$$\begin{aligned} F(lub_j\{x_j|j\geq1\}) &= lub\{f_i(lub\{x_j|j\geq1\})|i\geq1\} && \text{by 2} \\ &= lub\{lub\{f_i(x_j)|j\geq1\}|i\geq1\} && \text{by 1} \\ &= lub\{lub\{f_i(x_j)|i\geq1\}|j\geq1\} && \ddagger \\ &= lub\{F(x_j)|j\geq1\} && \text{by 2.} \end{aligned}$$

Look at Figure 10.9.

First Half

$$lub\{lub\{f_i(x_j)|j\geq1\}|i\geq1\} \subseteq lub\{lub\{f_i(x_j)|i\geq1\}|j\geq1\}$$

For all k and j, $f_k(x_j) \subseteq lub\{f_i(x_j)|i\geq1\}$ by the definition of F (the rows of Figure 10.9).

We have chains

$$f_k(x_1) \subseteq f_k(x_2) \subseteq f_k(x_3) \subseteq \ldots \text{ for each k}$$

and $lub\{f_i(x_1)|i\geq1\} \subseteq lub\{f_i(x_2)|i\geq1\}$
$\subseteq lub\{f_i(x_3)|i\geq1\} \subseteq \ldots$

So for each k,
$$lub\{f_k(x_j)|j\geq1\} \subseteq lub\{lub\{f_i(x_j)|i\geq1\}|j\geq1\}$$
by the Limit lemma.
This corresponds to the top row (remember each f_k is continuous).

Hence
$$lub\{lub\{f_k(x_j)|j\geq1\}|k\geq1\} \subseteq lub\{lub\{f_i(x_j)|i\geq1\}|j\geq1\}$$
by the Lub lemma. Now change k to i.

Second Half

$$lub\{lub\{f_i(x_j)|i\geq1\}|j\geq1\} \subseteq lub\{lub\{f_i(x_j)|j\geq1\}|i\geq1\}$$

For all i and k,
$f_i(x_k) \subseteq f_i(lub\{x_j|j\geq1\}) = lub\{f_i(x_j)|j\geq1\}$ by using the fact that each f_i is monotonic and continuous (the columns of Figure 10.9).

We have chains

$$f_1(x_k) \subseteq f_2(x_k) \subseteq f_3(x_k) \subseteq \ldots \text{ for each k}$$

and $lub\{f_1(x_j)|j\geq1\} \subseteq lub\{f_2(x_j)|j\geq1\}$
$\subseteq lub\{f_3(x_j)|j\geq1\} \subseteq \ldots$

So for each k,
$lub\{f_i(x_k)|i\geq1\} \subseteq lub\{lub\{f_i(x_j)|j\geq1\}|i\geq1\}$ by the Limit lemma.
This corresponds to the rightmost column.

Hence
$$lub\{lub\{f_i(x_k)|i\geq1\}|k\geq1\} \subseteq lub\{lub\{f_i(x_j)|j\geq1\}|i\geq1\}$$
by the Lub lemma. Now change k to j.

Therefore F is continuous. ∎

Example 10

Student = { ⊥, Autry, Bates }

Level =
{ ⊥, undergraduate, graduate, nondegree }

Fun(Student,Level) contains 64 (4^3) elements.

Only 19 of these functions are monotonic and continuous.

Which of these functions are monotonic?

f = { ⊥ ↦ ⊥, Autry ↦ nondegree, Bates ↦ ⊥ }

g = { ⊥ ↦ grad, Autry ↦ grad, Bates ↦ ⊥ }

h = { ⊥ ↦ grad, Autry ↦ grad, Bates ↦ grad }

Thm: If A and B are cpos, A is a finite set, and $f \in Fun(A,B)$ is monotonic, f is also continuous.

Proof: Let $x_1 \subseteq_A x_2 \subseteq_A x_3 \subseteq_A \ldots$ be an ascending chain in A.

Since A is finite, for some k, $x_k = x_{k+1} = x_{k+2} = \ldots$.

So the chain is a finite set, $\{x_1, x_2, x_3, \ldots, x_k\}$, whose least upper bound is x_k.

Since f is monotonic,
$f(x_1) \subseteq_B f(x_2) \subseteq_B f(x_3) \subseteq_B \ldots \subseteq_B f(x_k)$
$= f(x_{k+1}) = f(x_{k+2}) = \ldots$
is an ascending chain in B, which is also a finite set, namely $\{f(x_1), f(x_2), f(x_3), \ldots, f(x_k)\}$ with $f(x_k)$ as its least upper bound.

Therefore, $f(\mathit{lub}\{x_i \mid i \geq 1\}) = f(x_k) = \mathit{lub}\{f(x_i) \mid i \geq 1\}$, and f is continuous. ∎

Continuity of Functions on Domains

Thm: These functions on domains and their analogs are continuous:

1. *first* : $A \times B \to A$
2. *inS* : $A \to S$ where $S = A+B$
3. *outA* : $A+B \to A$
4. *isA* : $A+B \to \mathrm{T}$

Proof:

2. Let $a_1 \subseteq a_2 \subseteq a_3 \subseteq \ldots$ be an ascending chain in domain A.
   Observe that $1 \subseteq 1 \subseteq 1 \subseteq 1 \subseteq \ldots$ is an ascending chain in N.
   Then $\langle a_1,1\rangle \subseteq \langle a_2,1\rangle \subseteq \langle a_3,1\rangle \subseteq \ldots$ is an ascending chain in S.
   So $inS(\mathit{lub}\{a_i \mid i \geq 1\}) = \langle \mathit{lub}\{a_i \mid i \geq 1\}, 1\rangle$
   $= \langle \mathit{lub}\{a_i \mid i \geq 1\}, \mathit{lub}\{1 \mid i \geq 1\}\rangle$
   $= \mathit{lub}\{\langle a_i,1\rangle \mid i \geq 1\}$
   $= \mathit{lub}\{inS(a_i) \mid i \geq 1\}$.

3. An ascending chain $s_1 \subseteq s_2 \subseteq s_3 \subseteq \ldots$ in $S = A + B$ may repeat $\bot_{A+B}$ forever or eventually climb into either $A \times \{1\}$ or $B \times \{2\}$.
   In the first case, the least upper bound will be $\bot_{A+B}$,
   and in the other two cases the lub will be some $a \in A$ or some $b \in B$.

Case 1: $s_i = \bot_{A+B}$ for all $i \geq 1$.
Then $outA(\mathit{lub}_S\{s_i \mid i \geq 1\}) = outA(\bot_S) = \bot_A$,
and $\mathit{lub}_A\{outA(s_i) \mid i \geq 1\} = \mathit{lub}_A\{\bot_A \mid i \geq 1\} = \bot_A$.

Case 2:
For some $k \geq 1$, $s_i = \langle a_i,1\rangle$ for all $i \geq k$ where $a_i \in A$.
Then $outA(\mathit{lub}\{s_i \mid i \geq 1\})$
$= outA(\langle \mathit{lub}\{a_i \mid i \geq k\},1\rangle)$
$= \mathit{lub}\{a_i \mid i \geq k\}$
and $\mathit{lub}\{outA(s_i) \mid i \geq 1\} = \mathit{lub}\{a_i \mid i \geq k\}$.

Case 3:
For some $k \geq 1$, $s_i = \langle b_i,2\rangle$ for all $i \geq k$ where $b_i \in B$.
Then $outA(\mathit{lub}\{s_i \mid i \geq 1\}) = outA(\langle \mathit{lub}\{b_i \mid i \geq k\},2\rangle)$
$= \bot_A$
and $\mathit{lub}\{outA(s_i) \mid i \geq 1\} = \mathit{lub}\{\bot_A \mid i \geq 1\} = \bot_A$. ∎

Thm: The composition of continuous functions is continuous.

Proof: Suppose $f : A \to B$ and $g : B \to C$ are continuous functions.
Let $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ be an ascending chain in A.

Then $f(x_1) \subseteq f(x_2) \subseteq f(x_3) \subseteq \ldots$ is an ascending chain in B with $f(\mathit{lub}\{x_i \mid i \geq 1\}) = \mathit{lub}\{f(x_i) \mid i \geq 1\}$ by the continuity of f.

Since g is continuous,
$g(f(x_1)) \subseteq g(f(x_2)) \subseteq g(f(x_3)) \subseteq \ldots$ is an ascending chain in C with $g(\mathit{lub}\{f(x_i) \mid i \geq 1\}) = \mathit{lub}\{g(f(x_i)) \mid i \geq 1\}$.

Therefore $g(f(\mathit{lub}\{x_i \mid i \geq 1\})) = g(\mathit{lub}\{f(x_i) \mid i \geq 1\}) = \mathit{lub}\{g(f(x_i)) \mid i \geq 1\}$ and $g \circ f$ is continuous. ∎

Fixed Point Semantics

Goal: Provide meaning for recursive definitions.

First Step: Transform partial functions into total functions.

Example

f is a function with domain D = {0,1,2} and codomain C = {0,1,2} defined by:

$f(n) = 2/n$

or

$f = \{<1,2>,<2,1>\}$.

Note that f(0) is undefined; therefore f is a partial function.

Now extend f to make it a total function:

$f = \{<1,2>,<2,1>,<0,?>\}$.

Add an undefined element to the codomain, $C^+ = \{\perp,0,1,2\}$, and for symmetry, do likewise with the domain, $D^+ = \{\perp,0,1,2\}$.

Define the **natural extension** of f by having $\perp_D$ map to $\perp_C$ under f:

$f^+ = \{<\perp,\perp>,<0,\perp>,<1,2>,<2,1>\}$.

Define a relationship that orders functions and domains according to how "defined" they are, putting a lattice-like structure on the elementary domains:

For $x,y \in D^+$, $x \subseteq y$ if $x=\perp$ or $x=y$.

This relation is read "x approximates y" or "x is less defined or equal to y".

Thm: Let f^+ be a natural extension of a function between two sets D and C so that f^+ is a total function from D^+ to C^+**.**
Then f^+ is monotonic and continuous.

Proof: Let $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ be an ascending chain in the domain $D^+ = D \cup \{\perp\}$.
Two possibilities for the behavior of the chain:

Case 1: $x_i = \perp_D$ for all $i \geq 1$.
Then $lub\{x_i | i \geq 1\} = \perp_D$, and
$f^+(lub\{x_i | i \geq 1\}) = f^+(\perp_D)$
$= \perp_C = lub\{\perp_C\} = lub\{f^+(x_i) | i \geq 1\}$.

Case 2: $x_i = \perp_D$ for $1 \leq i \leq k$ and $x_{k+1} = x_{k+2} = x_{k+3} = \ldots$, since once the terms move above bottom, the sequence is constant in a flat domain.

Then $lub\{x_i | i \geq 1\} = x_{k+1}$, and
$f^+(lub\{x_i | i \geq 1\}) = f^+(x_{k+1})$
$= lub\{\perp_C, f^+(x_{k+1})\} = lub\{f^+(x_i) | i \geq 1\}$.

If f^+ is continuous, it is also monotonic. ∎

The **natural extension** of a function whose domain is a Cartesian product, namely
$f : D_1^+ x D_2^+ x \ldots x D_n^+ \rightarrow C^+$, has the property that
$f^+(x_1,x_2,\ldots,x_n) = \perp_C$ whenever at least one $x_i = \perp$.

Any function that satisfies this property is known as a **strict** function.

Thm: If $f^+: D_1^+ x D_2^+ x \ldots x D_n^+ \rightarrow C^+$ is a natural extension where D_i^+, $1 \leq i \leq n$, and C^+ are elementary domains, then f^+ is monotonic and continuous.

Proof: Consider the case where n=2. Show f^+ is continuous.
Let $<x_1,y_1> \subseteq <x_2,y_2> \subseteq <x_3,y_3> \subseteq \ldots$ be an ascending chain in $D_1^+ x D_2^+$. Since D_1^+ and D_2^+ are elementary domains, the chains $\{x_i | i \geq 1\}$ and $\{y_i | i \geq 1\}$ must follow one of the two cases in the previous proof, namely all $\perp$ or eventually a constant proper value in D_i^+.

Case 1: $lub\{x_i|i\geq1\} = \perp_{D_1^+}$ or $lub\{y_i|i\geq1\} = \perp_{D_2^+}$ (or both).

Then $f^+(lub\{<x_i,y_i>|i\geq1\})$
$= f^+(<lub\{x_i|i\geq1\},lub\{y_i|i\geq1\}>) = \perp_{C^+}$
because f^+ is a natural extension and one of its arguments is $\perp$, and

$lub\{f^+(<x_i,y_i>)|i\geq1\} = lub\{\perp_{C^+}|i\geq1\} = \perp_{C^+}$.

Case 2: $lub\{x_i|i\geq1\} = x\in D_1$
and $lub\{y_i|i\geq1\} = y\in D_2$

Since D_1^+ and D_2^+ are both elementary domains, there is an integer k
such that $x_i = x$ and $y_i = y$ for all $i\geq k$.

So $f^+(lub\{<x_i,y_i>|i\geq1\})$
$= f^+(<lub\{x_i|i\geq1\},lub\{y_i|i\geq1\}>)$
$= f^+(<x,y>)\in C^+$

and $lub\{f^+(<x_i,y_i>)|i\geq1\} = lub\{\perp_{C^+},f^+(<x,y>)\}$
$= f^+(<x,y>)$. ∎

Example

Consider the natural extension of the conditional expression operation:

(if a b c) = if a then b else c.

The natural extension unduly restricts the meaning of the conditional expression.

For example, we prefer that the following expression return 0 when x=1 and y=0:

if y>0 then x/y else 0.

If we interpret the undefined operation 1/0 as $\perp$,
when x=1 and y=0,
(if^+ y>0 x/y 0) = (if^+ false $\perp$ 0) = $\perp$
for a natural extension.

Second Step: Give meaning to recursive definitions.

Consider a recursively defined function f : N→N where N = {$\perp$,0,1,2,3,…} and

f(n) = (†)
if n=0 then 5 else if n=1 then f(n+2) else f(n-2)

Two questions:

1. What function, if any, does this equation in f denote?
2. Does it specify more than one function?

Define a **functional** F by

F : (N→N)→(N→N) where

(F(f))(n) = (‡)
if n=0 then 5 else if n=1 then f(n+2) else f(n-2)

Function application associates to the left; omit the parentheses with multiple applications, writing F f n for (F(f))(n).

A function, f : N→N, satisfies the original definition (†) if and only if it is a **fixed point** of the definition of F, (‡),
F f n = f(n) for all n∈N or just F f = f.

Once more:
Suppose f : D→C is a function defined recursively by f(x) = α(x,f) for each x∈D where α(x,f) is some expression in x and f.
Furthermore, let F : (D→C)→(D→C) be the functional defined by F f x = α(x,f).

Then
F f = f if and only if F f x = f x for all x∈D
if and only if α(x,f) = f x for all x∈D
which is the same as f(x) = α(x,f) for all x∈D.

Using lambda calculus notation:

F f = λn . if n=0 then 5 else if n=1 then f(n+2)
else f(n-2)
or
F = λf . λn . if n=0 then 5 else if n=1 then f(n+2)
else f(n-2)

Fixed Points in Mathematics

Function	Fixed Points
$g(n) = n^2 - 6n$	0 and 7
$g(n) = n$	all $n \in N$
$g(n) = n + 5$	none
$g(n) = 2$	2

Back to Functional F

The function $g = \lambda n . 5$ is a fixed point of F:

$$\begin{aligned} F\ g &= \lambda n . \text{ if } n=0 \text{ then } 5 \text{ else if } n=1 \text{ then } g(n+2) \text{ else } g(n-2) \\ &= \lambda n . \text{ if } n=0 \text{ then } 5 \text{ else if } n=1 \text{ then } 5 \text{ else } 5 \\ &= \lambda n . 5 = g. \end{aligned}$$

Problem

$g = \lambda n . 5$ does not agree with the operational behavior of the original recursive definition.

$f(1) = f(3) = f(1) = \ldots$ does not produce a value, whereas $g(1) = 5$.

Special Fixed Point

Of the possible fixed points of a functional, chose the one that is "least defined" according to $\subseteq$.

1. Any fixed point of F embodies the information that can be deduced from F.
2. The least fixed point includes no more information than what *must* be deduced.

Define the meaning of a recursive definition of a function to be the "least" fixed point, with respect to $\subseteq$, of the corresponding functional F.

Does a least fixed point always exist?

Notation: Define f^k for each $k \geq 0$ inductively:

$f^0(x) = x$ is the identity function and

$f^{n+1}(x) = f(f^n(x))$ for $n \geq 0$.

Thm: If D with $\subseteq$ is a complete partial order and $g : D \to D$ (g is any monotonic and continuous function on D), then g has a least fixed point with respect to $\subseteq$ on $D \to D$.

Proof: Since D is a cpo, $g^0(\bot) = \bot \subseteq g(\bot)$.

Since g is monotonic, $g(\bot) \subseteq g(g(\bot)) = g^2(\bot)$.

In general, since g is monotonic,
$g^i(\bot) \subseteq g^{i+1}(\bot)$ implies

$$g^{i+1}(\bot) = g(g^i(\bot)) \subseteq g(g^{i+1}(\bot)) = g^{i+2}(\bot).$$

So by induction,

$$\bot \subseteq g(\bot) \subseteq g^2(\bot) \subseteq g^3(\bot) \subseteq g^4(\bot) \subseteq \ldots$$

is an ascending chain in D, which must have a least upper bound $u = lub\{g^i(\bot) \mid i \geq 0\} \in D$.

$$\begin{aligned} \text{But } g(u) &= g(lub\{g^i(\bot) \mid i \geq 0\}) \\ &= lub\{g(g^i(\bot)) \mid i \geq 0\} \text{ since g is continuous} \\ &= lub\{g^{i+1}(\bot) \mid i \geq 0\} \\ &= lub\{g^i(\bot) \mid i > 0\} = u \end{aligned}$$

That is, u is a fixed point for g.

Note that $g^0(\bot) = \bot$ has no effect on the least upper bound of $\{g^i(\bot) \mid i \geq 0\}$.

Let $v \in D$ be another fixed point for g.

Then $\bot \subseteq v$ and $g(\bot) \subseteq g(v) = v$, the basis step for induction.

Suppose $g^i(\bot) \subseteq v$.

Then since g is monotonic,
$g^{i+1}(\bot) = g(g^i(\bot)) \subseteq g(v) = v$, the induction step.

Therefore, by mathematical induction, $g^i(\bot) \subseteq v$ for all $i \geq 0$.

So v is an upper bound for $\{g^i(\bot) \mid i \geq 0\}$.
Hence $u \subseteq v$, since u is the least upper bound for $\{g^i(\bot) \mid i \geq 0\}$. ∎

Corollary: Every continuous functional $F : (A \to B) \to (A \to B)$, where A and B are domains, has a least fixed point, $F_{fp} : A \to B$, which can be taken as the meaning of the (recursive) definition corresponding to F.

Example

Consider the functional G: (N→N)→(N→N) where

G g n = if n=0 then 1
 else if n=1 then g(3)-12
 else 4n+g(n-2) (‡)

that corresponds to the recursive definition

g(n) = if n=0 then 1
 else if n=1 then g(3)-12
 else 4n+g(n-2) (†)

Contemplate the ascending sequence

$\perp \subseteq G(\perp) \subseteq G^2(\perp) \subseteq G^3(\perp) \subseteq G^4(\perp) \subseteq \ldots$

and its least upper bound.

Use the abbreviation $g_k = (G^k \perp)$ for $k \geq 0$:

$g_0(n) = G^0 \perp n = \perp(n)$
$g_1(n) = G \perp n = G\ g_0\ n$
$g_2(n) = G\ (G \perp)\ n = G\ g_1\ n$
$g_3(n) = G^3 \perp n = G\ g_2\ n$

Now calculate a few terms in the ascending chain

$g_0 \subseteq g_1 \subseteq g_2 \subseteq g_3 \subseteq \ldots$.

$g_0(n) = G^0 \perp n = \perp(n) = \perp$ for $n \in N$,
the everywhere undefined function.

$g_1(n) = G \perp n = G\ g_0\ n$
= if n=0 then 1 else if n=1 then $g_0(3)$-12
 else 4n+g_0(n-2)
= if n=0 then 1 else if n=1 then $\perp$(3)-12
 else 4n+$\perp$(n-2)
= if n=0 then 1 else $\perp$

$g_2(n) = G^2 \perp n = G\ g_1\ n$
= if n=0 then 1 else if n=1 then $g_1(3)$-12
 else 4n+g_1(n-2)
= if n=0 then 1 else if n=1 then $\perp$-12
 else 4n+(if n-2=0 then 1 else $\perp$)
= if n=0 then 1 else if n=1 then $\perp$
 else (if n=2 then 4n+1 else $\perp$)
= if n=0 then 1 else if n=1 then $\perp$
 else if n=2 then 9 else $\perp$

Note Property

a + (if b then c else d) =
 = if b then a+c else a+d

$g_3(n) = G^3 \perp n = G\ g_2\ n$
= if n=0 then 1 else if n=1 then $g_2(3)$-12
 else 4n+g_2(n-2)
= if n=0 then 1 else if n=1 then $\perp$-12
 else 4n+(if n-2=0 then 1
 else if n-2=1 then $\perp$
 else if n-2=2 then 9 else $\perp$)
= if n=0 then 1 else if n=1 then $\perp$
 else (if n=2 then 4n+1
 else if n=3 then 4n+$\perp$
 else if n=4 then 4n+9
 else 4n+$\perp$)
= if n=0 then 1
 else if n=1 then $\perp$
 else if n=2 then 9
 else if n=3 then $\perp$
 else if n=4 then 25 else $\perp$

$g_4(n) = G^4 \perp n = G\ g_3\ n$
= if n=0 then 1 else if n=1 then $g_3(3)$-12
 else 4n+g_3(n-2)
= if n=0 then 1 else if n=1 then $\perp$-12
 else 4n+(if n-2=0 then 1
 else if n-2=1 then $\perp$
 else if n-2=2 then 9
 else if n-2=3 then $\perp$
 else if n-2=4 then 25
 else $\perp$)
= if n=0 then 1 else if n=1 then $\perp$
 else (if n=2 then 4n+1
 else if n=3 then 4n+$\perp$
 else if n=4 then 4n+9
 else if n=5 then 4n+$\perp$
 else if n=6 then 4n+25
 else 4n+$\perp$)
= if n=0 then 1
 else if n=1 then $\perp$
 else if n=2 then 9
 else if n=3 then $\perp$
 else if n=4 then 25
 else if n=5 then $\perp$
 else if n=6 then 49 else $\perp$

A pattern seems to be developing.

Lemma: For all $i \geq 0$,

$g_i(n) =$
if $n<2i$ then (if *even*(n) then $(n+1)^2$ else $\perp$)
else $\perp$.

Proof: The proof proceeds by induction on i.

a) By the previous computations, for i = 0,
$g_0(n) = \perp =$
if $n<2 \cdot 0$ then
(if *even*(n) then $(n+1)^2$ else $\perp$) else $\perp$

b) As the induction hypothesis, assume that
$g_i(n) =$
if $n<2i$ then
(if *even*(n) then $(n+1)^2$ else $\perp$) else $\perp$,
for any arbitrary $i \geq 0$.

Then $g_{i+1}(n) = G\ g_i\ n$
= if n=0 then 1
else if n=1 then $g_i(3)$-12 else 4n+$g_i(n-2)$

= if n=0 then 1
else if n=1 then $\perp$-12
else 4n+(if n-2<2i
then (if *even*(n–2) then $(n-1)^2$
else $\perp$)
else $\perp$)
= if n=0 then 1
else if n=1 then $\perp$
else (if n<2i+2
then (if *even*(n–2) then 4n+$(n-1)^2$
else 4n+$\perp$) else 4n+$\perp$)
= if n=0 then 1
else if n=1 then $\perp$
else if n<2(i+1)
then (if *even*(n) then $(n+1)^2$
else $\perp$) else $\perp$
= if n<2(i+1)
then (if *even*(n) then $(n+1)^2$ else $\perp$)
else $\perp$

Therefore our pattern for the g_i is correct. ∎

The least upper bound of the ascending chain

$g_0 \subseteq g_1 \subseteq g_2 \subseteq g_3 \subseteq \ldots$, where

$g_i(n)$ = if n<2i then (if *even*(n) then $(n+1)^2$ else $\perp$)
else $\perp$,

must be defined (not $\perp$) for any n where some g_i is defined, and must take the value $(n+1)^2$ there.

Hence the least upper bound is

$G_{fp}(n) = (lub\{g_i \mid i \geq 0\})\ n$

$= (lub\{G^i \perp \mid i \geq 0\})\ n$

= if *even*(n) then $(n+1)^2$ else $\perp$ for all $n \in N$,

and this function can be taken as the meaning of the original recursive definition.

Note that the function h n = $(n+1)^2$ is also a fixed point for G.

It is more defined than G_{fp}.

In fact, $G_{fp} \subseteq h$.

fix

The procedure for computing a least fixed point for a functional can be described as an operator on functions $F : D \to D$:

$fix : (D \to D) \to D$ where
$fix\ F = lub\{F^i(\perp) \mid i \geq 0\} \in D$.

The least fixed point of the functional

$F = \lambda f\ .\ \lambda n$. if n=0 then 5 else if n=1 then f(n+2)
else f(n-2)

can then be expressed as

$F_{fp} = fix\ F$, an element of $D = N \to N$.

For $F : (N \to N) \to (N \to N)$, *fix* has type

$fix : ((N \to N) \to (N \to N)) \to (N \to N)$.

The fixed point operator *fix* provides a fixed point for any continuous functional, namely, the least defined function with this fixed point property.

Fixed Point Identity: $F(fix\ F) = fix\ F$.

Continuous Functionals

Lemma: A constant function $f : D \rightarrow C$, where $f(x) = k$ for some fixed $k \in C$ and for all $x \in D$, is continuous given either of the two extensions:

a) The natural extension where $f(\perp_D) = \perp_C$.

b) The “unnatural” extension where $f(\perp_D) = k$.

Lemma: An identity function $f : D \rightarrow D$, where $f(x) = x$ for all x in a domain D, is continuous.

Proof: If $x_1 \subseteq x_2 \subseteq x_3 \subseteq \ldots$ is an ascending chain in D, it follows that

$f(\mathit{lub}\{x_i|i\geq 1\}) = \mathit{lub}\{x_i|i\geq 1\} = \mathit{lub}\{f(x_i)|i\geq 1\}$. ∎

Conditional Expression Function:

Natural extension of “if” is too restrictive.

Lazy if

if(a,b,c) = if a then b else c.
where if : TxDxD→D for some domain D
and T = { ⊥, true, false }

(if true then b else c) = b for any $b,c \in D$
(if false then b else c) = c for any $b,c \in D$
(if ⊥ then b else c) = $\perp_D$ for any $b,c \in D$

Lemma: The uncurried “if” function as defined above is continuous.

Proof: Consider three cases.

Lemma: The composition of continuous functions is continuous, namely if $f : C_1 x C_2 x \ldots x C_n \rightarrow C$ is continuous and $g_i : D_i \rightarrow C_i$ is continuous for each i, $1 \leq i \leq n$, then $f \circ \langle g_1,g_2,\ldots,g_n\rangle : D_1 x D_2 x \ldots x D_n \rightarrow C$, defined by

$f \circ (g_1,g_2,\ldots,g_n) \langle x_1,x_2,\ldots,x_n\rangle =$
$f \langle g_1(x_1),g_2(x_2),\ldots,g_n(x_n)\rangle$

is also continuous.

Proof: Exercise.

Composition Involving a Parameter

$F : (N \rightarrow N) \rightarrow (N \rightarrow N)$ where

F f n = n + if n=0 then 0 else f(f(n-1)).

Lemma: If $F_1, F_2, \ldots, F_n$ are continuous functionals, say $F_i : (D^n \rightarrow D) \rightarrow (D^n \rightarrow D)$ for each i, $1 \leq i \leq n$, the functional $F : (D^n \rightarrow D) \rightarrow (D^n \rightarrow D)$ defined by $F\ f\ d = f \langle F_1\ f\ d, F_2\ f\ d, \ldots, F_n\ f\ d\rangle$ for all $f \in D^n \rightarrow D$ and $d \in D^n$ is also continuous.

Proof: Consider the case where n=1.
So $F_1 : (D \rightarrow D) \rightarrow (D \rightarrow D)$,
$F : (D \rightarrow D) \rightarrow (D \rightarrow D)$, and $F\ f\ d = f \langle F_1\ f\ d\rangle$.
Let $f_1 \subseteq f_2 \subseteq f_3 \subseteq \ldots$ be a chain in $D \rightarrow D$.
The proof shows that $\mathit{lub}\{F\ f_i|i\geq 1\} = F(\mathit{lub}\{f_i|i\geq 1\})$ in two parts.

Part 1: $\mathit{lub}\{F(f_i)|i\geq 1\} \subseteq F(\mathit{lub}\{f_i|i\geq 1\})$.
For each $i\geq 1$, $f_i \subseteq \mathit{lub}\{f_i|i\geq 1\}$.
Since F_1 is monotonic, $F_1(f_i) \subseteq F_1(\mathit{lub}\{f_i|i\geq 1\})$, which means that
$F_1\ f_i\ d \subseteq F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d$ for each $d \in D$.
Since f_i is monotonic, $f_i(F_1\ f_i\ d) \subseteq f_i(F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d)$.
But $F\ f_i\ d = f_i \langle F_1\ f_i\ d\rangle$ and
$f_i \langle F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d\rangle \subseteq \mathit{lub}\{f_i|i\geq 1\} \langle F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d\rangle$.
Therefore, $F\ f_i\ d \subseteq \mathit{lub}\{f_i|i\geq 1\} \langle F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d\rangle$ for each $i\geq 1$ and $d \in D$.
So, $\mathit{lub}\{F(f_i)|i\geq 1\}\ d = \mathit{lub}\{F\ f_i\ d|i\geq 1\} \subseteq$
$\mathit{lub}\{f_i|i\geq 1\} \langle F_1\ \mathit{lub}\{f_i|i\geq 1\}\ d\rangle = F\ \mathit{lub}\{f_i|i\geq 1\}\ d$ for $d \in D$.

Part 2: $F(\mathit{lub}\{f_i|i\geq 1\}) \subseteq \mathit{lub}\{F(f_i)|i\geq 1\}$.
For any $d \in D$, $F\ \mathit{lub}\{f_i|i\geq 1\}\ d$
$= \mathit{lub}\{f_i|i\geq 1\} \langle F_1\ \mathit{lub}\{f_j|j\geq 1\}\ d\rangle$ by defn of F,
$= \mathit{lub}\{f_i|i\geq 1\}(\mathit{lub}\{F_1(f_j)|j\geq 1\}\ d)$ since F_1 is cont,
$= \mathit{lub}\{\mathit{lub}_i\{f_i|i\geq 1\} \langle\{F_1(f_j)|j\geq 1\}\ d\rangle|i\geq 1\}$
since $\mathit{lub}\{f_i|i\geq 1\}$ is continuous
$= \mathit{lub}\{\mathit{lub}\{f_i(\{F_1(f_j)|j\geq 1\}\ d)\}|i\geq 1\}$
by definition of $\mathit{lub}\{f_i|i\geq 1\}$. †

If $j \le i$, $f_j \subseteq f_i$,

$F_1\ f_j \subseteq F_1\ f_i$ since F_1 is monotonic,

$F_1\ f_j\ d \subseteq F_1\ f_i\ d$ for each $d \in D$, and

$f_i <F_1\ f_j\ d> \subseteq f_i <F_1\ f_i\ d>$ since f_i is monotonic.

If $i<j$, $f_i \subseteq f_j$, and

$f_i <F_1\ f_j\ d> \subseteq f_j <F_1\ f_j\ d>$ for each $d \in D$ by the meaning of $\subseteq$.

So, $f_i <F_1\ f_j\ d> \subseteq lub\{f_n(F_1\ f_n\ d) | n \ge 1\}$ for each $i,j \ge 1$.

But $lub_n\{f_n(F_1\ f_n\ d)\} = lub_n\{F\ f_n\ d | i \ge 1\}$

$= lub_n\{F(f_n) | i \ge 1\}\ d$ by the definition of F.

So $f_i <F_1\ f_j\ d> \subseteq lub\{F(f_n) | n \ge 1\}\ d$ for each $i,j \ge 1$, and

$lub\{f_i <F_1\ f_j\ d> | i \ge 1\} \subseteq lub\{F(f_n) | n \ge 1\} d$ for each $j \ge 1$.

Hence

$lub\{lub\{f_i <F_1\ f_j\ d> | i \ge 1\} | j \ge 1\} \subseteq lub\{F(f_n) | i \ge 1\}\ d$.

Combining with † gives

$F(lub\{f_i | i \ge 1\})\ d \subseteq lub_n\{F(f_n) | i \ge 1\}\ d$.

Theorem: Any functional H defined by the composition of naturally extended functions on elementary domains, constant functions, the identity function, the if-then-else conditional expression, and a function variable f, is continuous.

Proof: The proof follows by induction on the structure of the definition of the functional. The basis is handled by the continuity of natural extensions, constant functions, and the identity function. The induction step relies on the lemmas which state that the composition of continuous functions, possibly involving f, is continuous.

Look at Example 14:

$H : (N \to N) \to (N \to N)$ where

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

= if n=0 then n else n+h(h(n-1)).

Fixed Points for Nonrecursive Functions

Find the least fixed point for the function $h(n) = n^3 - 3n$ defined on the integers Z.

First Interpretation:

The natural extension h^+ of h is a continuous function on the elementary domain $Z \cup \{\perp\}$.

Then the least fixed point of h^+ may be constructed as the least upper bound of the ascending sequence:

$\perp \subseteq h^+(\perp) \subseteq h^+(h^+(\perp)) \subseteq h^+(h^+(h^+(\perp))) \subseteq \ldots$

But $h^+(\perp) = \perp$,

and so $(h^+)^k(\perp) = h^+((h^+)^{k-1}(\perp)) = h^+(\perp) = \perp$ for any $k \ge 1$.

Therefore, $lub\{(h^+)^k(\perp) | k \ge 0\} = lub\{\perp | k \ge 0\} = \perp$ is the least fixed point.

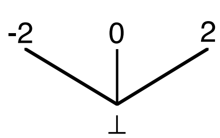
In fact, h^+ has four fixed points in $Z \cup \{\perp\}$:

$h^+(0) = 0$

$h^+(2) = 2$

$h^+(-2) = -2$

$h^+(\perp) = \perp$



Second Interpretation:

Think of $h(n) = n^3 - 3n$ as a rule defining a “recursive” function that just has no actual recursive call of h.

The corresponding functional

$H : (Z \to Z) \to (Z \to Z)$ is defined by the rule:

$H\ h\ n = n^3 - 3n$.

A function h satisfies definition $h(n) = n^3 - 3n$ if and only if it is a fixed point of H, that is H h = h.

The fixed point construction:

$H^0 \perp n = \perp(n) = \perp$

$H^1 \perp n = n^3 - 3n$

$H^2 \perp n = n^3 - 3n$

$H^3 \perp n = n^3 - 3n$

:

$H^k \perp n = n^3 - 3n$

:

Therefore, the least fixed point is $lub\{H^k(\perp) | k \ge 0\} = \lambda n\ .\ n^3 - 3n$, which follows the same definition rule as the original function h.

Revisiting Denotational Semantics

The recursive definition

execute ⟦**while** E **do** C⟧ sto =
 if *evaluate* ⟦E⟧ sto = *bool*(true)
  then *execute* ⟦**while** E **do** C⟧ (*execute* ⟦C⟧ sto)
  else sto

violates the principle of compositionality.

The function *execute* ⟦**while** E **do** C⟧ satisfies the recursive definition above if and only if it is a fixed point of the functional

W f s = if *evaluate* ⟦E⟧ s = *bool*(true)
    then f(*execute* ⟦C⟧ s) else s
  = if *evaluate* ⟦E⟧ s = *bool*(true)
    then ($f \circ$ *execute* ⟦C⟧) s else s.

We obtain a nonrecursive and compositional definition of the meaning of a **while** command by means of

execute ⟦**while** E **do** C⟧ = *fix* W.

We gain insight into both the **while** command and fixed point semantics by constructing a few terms in the ascending chain whose least upper bound is *fix* W,

$$W^0 \perp \subseteq W^1 \perp \subseteq W^2 \perp \subseteq W^3 \perp \subseteq \dots$$

where *fix* W = *lub*$\{W^i \perp \mid i \geq 0\}$.

The fixed point construction for W:

$W^0 \perp s = \perp$

$W^1 \perp s = W\,(W^0 \perp)\,s$
  = if *evaluate* ⟦E⟧ s = *bool*(true)
    then $\perp$(*execute* ⟦C⟧ s) else s
  = if *evaluate* ⟦E⟧ s = *bool*(true)
    then $\perp$ else s

Let exC stand for the function *execute* ⟦C⟧.

Then

$W^2 \perp s = W\,(W^1 \perp)\,s$
  = if *evaluate* ⟦E⟧ s = *bool*(true)
    then $W^1 \perp$ (exC s) else s
  = if *evaluate* ⟦E⟧ s = *bool*(true)
   then
    (if *evaluate* ⟦E⟧ (exC s) = *bool*(true)
     then $\perp$ else exC s)
   else s

$W^3 \perp s = W\,(W^2 \perp)\,s$
  = if *evaluate* ⟦E⟧ s = *bool*(true)
    then $W^2 \perp$ (exC s) else s
 = if *evaluate* ⟦E⟧ s = *bool*(true)
   then (if *evaluate* ⟦E⟧ (exC s) = *bool*(true)
    then
    (if *evaluate* ⟦E⟧ (exC (exC s)) = *bool*(true)
     then $\perp$ else exC (exC s))
    else (exC s))
   else s

 = if *evaluate* ⟦E⟧ s = *bool*(true)
   then (if *evaluate* ⟦E⟧ (exC s) = *bool*(true)
    then (if *evaluate* ⟦E⟧ (exC^2 s) = *bool*(true)
     then $\perp$ else (exC^2 s))
    else (exC s))
   else s

$W^4 \perp s$ =
 if *evaluate* ⟦E⟧ s = *bool*(true)
  then (if *evaluate* ⟦E⟧ (exC s) = *bool*(true)
   then (if *evaluate* ⟦E⟧ (exC^2 s) = *bool*(true)
    then (if *evaluate* ⟦E⟧ (exC^3 s) = *bool*(true)
     then $\perp$ else (exC^3 s))
    else (exC^2 s))
   else (exC s))
  else s

In general,
$W^{k+1} \perp s = W (W^k \perp) s$
= if *evaluate* ⟦E⟧ s = *bool*(true)
then (if *evaluate* ⟦E⟧ (exC s) = *bool*(true)
then (if *evaluate* ⟦E⟧ (exC^2 s) = *bool*(true)
then (if *evaluate* ⟦E⟧ (exC^3 s) = *bool*(true)
:
then ⊥ else (exC^k s))
else (exC^{k-1} s))
:
else (exC^2 s))
else (exC s))
else s

The function $W^{k+1} \perp$ allows the body C of the **while** to execute up to k times.
Thus this approximation to the meaning of the **while** command can handle any instance of a while with at most k iterations of the body.

Any application of a while command will have some finite number of iterations, say n. Therefore its meaning is subsumed in the approximation $W^{n+1} \perp$.

The least upper bound of this ascending sequence provides semantics for the while command:

execute ⟦**while** E **do** C⟧ = *fix* W = *lub*$\{W^i \perp | i \geq 0\}$.

View the definition of *execute* ⟦**while** E **do** C⟧ in terms of the fixed point identity,
W(*fix* W) = *fix* W, where

W f s = if *evaluate* ⟦E⟧ s = *bool*(true)
then f(*execute* ⟦C⟧ s) else s.
In this context,
execute ⟦**while** E **do** C⟧ = *fix* W

Now *define* loop = *fix* W. Then
execute ⟦**while** E **do** C⟧
= loop
where loop s = (W loop) s

= loop where loop s =
if *evaluate* ⟦E⟧ s = *bool*(true)
then loop(*execute* ⟦C⟧ s) else s.

This approach produces the compositional definition of *execute* ⟦**while** E **do** C⟧ used in the specification of Wren, Figure 9.11.

Fixed Point Induction

Induction on the construction of the least fixed point *lub*$\{F^i \perp | i \geq 0\}$.

Let Φ(f) be a predicate that describes a property for an arbitrary function f defined recursively.

To show Φ holds for the least fixed point F_{fp} of the functional F corresponding to a recursive definition of f, two conditions are needed:

Part 1: Show by induction that Φ holds for each element in the ascending chain
$\perp \subseteq F \perp \subseteq F^2 \perp \subseteq F^3 \perp \subseteq \ldots$ and

Part 2: Show that Φ remains true when the least upper bound is taken.

Part 2 is handled by defining a class of predicates with the necessary property.

A predicate is called **admissible** if it has the property that whenever the predicate hold for an ascending chain of functions, it also must hold for the least upper bound of that chain.

Theorem: Any finite conjunction of inequalities of the form $\alpha(F) \subseteq \beta(F)$, where α and β are continuous functionals, is an admissible predicate. This includes terms of the form $\alpha(F) = \beta(F)$.

Proof: See [Manna72]. ∎

Mathematical induction is used to verify the condition in Part 1:

Given a functional $F : (D \to D) \to (D \to D)$ for some domain D and admissible predicate Φ(f), show:

a) $\Phi(\perp)$ holds where $\perp : D \to D$, and

b) for any $i \geq 0$, if $\Phi(F^i(\perp))$, then $\Phi(F^{i+1}(\perp))$.

An alternate version of condition b) is:

b') for any $f : D \to D$, if Φ(f), then Φ(F(f)).

Either formulation is sufficient to infer that the predicate Φ holds for every function in the ascending chain $\{F^i \perp | i \geq 0\}$.

Example

H h n = if n=0 then 0 else (2n-1)+h(n-1) with least fixed point H_{fp}.

Prove that $H_{fp} \subseteq \lambda n \,.\, n^2$.

Let Φ(f) be the predicate $f \subseteq \lambda n \,.\, n^2$.

a) Since $\perp \subseteq \lambda n \,.\, n^2$, $\Phi(\perp)$ holds.

b') Suppose Φ(h), that is, $h \subseteq \lambda n \,.\, n^2$.

Then H h n= **if** n=0 **then** 0 **else** (2n-1)+h(n-1)
$\subseteq$ **if** n=0 **then** 0 **else** $(2n-1)+(n-1)^2$
= **if** n=0 **then** 0 **else** n^2
= n^2 for $n \geq 0$.

Therefore, Φ(H(h)) holds, and by fixed point induction $H_{fp} \subseteq \lambda n \,.\, n^2$. ∎

Paradoxical Combinator

An implementation of the fixed-point operator *fix* in the (untyped) lambda calculus:

define **Y** = λf . (λx . f (x x)) (λx . f (x x))

or in the lambda calculus evaluator

define **Y** = (L f ((L x (f (x x))) (L x (f (x x))))).

Reduction proves **Y** satisfies fixed-point identity.

Y E = (**λf** . (λx . f (x x)) (λx . f (x x))) **E**
⇒ (**λx** . E (x x)) **(λx . E (x x))**
⇒ E ((λx . E (x x)) (λx . E (x x)))
⇒ E (**λh** . (λx . h (x x)) (λx . h (x x)) **E**)
= E (**Y** E).

Calculation follows normal order reduction.

Applicative order strategy leads to a nonterminating reduction:

Y E = (λf . (**λx** . f (x x)) **(λx . f (x x))**) E
⇒ (λf . f ((**λx** . f (x x)) **(λx . f (x x))**)) E
⇒ (λf . f (f ((**λx** . f (x x)) **(λx . f (x x))**))) E
⇒ …

Fixed-Point Identity

F(*fix* F) = *fix* F

Add a reduction rule that carries out effect of fixed-point identity from right to left to replicate the functional F—namely, *fix* F ⇒ F(*fix* F).

Consider this definition of a function involving powers of 2 with its associated functional:

two n = if n=0 then 1 else 2•two(n-1)+1
and
Two = λh . λn . if n=0 then 1 else 2•h(n-1)+1.

The least fixed point of Two, (*fix* Two), serves as the definition of the two function.

The function (*fix* Two) is not recursive and can be “reduced” using the fixed-point identity

fix Two ⇒ Two (*fix* Two).

The replication of the function encoded in the *fix* operator enables a reduction to create as many copies of the original function as it needs.

(*fix* Two) 4

⇒ (Two (*fix* Two)) 4

⇒ (**λh** . λn . if n=0 then 1
else 2•h(n-1)+1) **(*fix* Two)** 4

⇒ (**λn** . if n=0 then 1
else 2•(*fix* Two)(n-1)+1) **4**

⇒ if 4=0 then 1 else 2•(*fix* Two)(4-1)+1

⇒ 2•(**(fix Two)** 3)+1

⇒ 2•((Two (*fix* Two)) 3)+1

⇒ 2•((**λh** . λn . if n=0 then 1
else 2•h(n-1)+1) **(*fix* Two)** 3)+1

$\Rightarrow$ 2•((**λn** . if n=0 then 1
else 2•(*fix* Two)(n-1)+1) **3**)+1

$\Rightarrow$ 2•((if 3=0 then 1 else 2•(*fix* Two)(3-1))+1)+1

$\Rightarrow$ 2•(2•(**(fix Two)** 2)+1)+1

$\Rightarrow$ 2•(2•((Two (*fix* Two)) 2)+1)+1

$\Rightarrow$ 2•(2•((**λh** . λn . if n=0 then 1
else 2•h(n-1)+1) **(*fix* Two)** 2)+1)+1

$\Rightarrow$ 2•(2•((**λn** . if n=0 then 1
else 2•(*fix* Two)(n-1)+1) **2**)+1)+1

$\Rightarrow$ 2•(2•(if 2=0 then 1
else 2•((*fix* Two) (2-1))+1)+1)+1

$\Rightarrow$ 2•(2•(2•(**(*fix* Two)** 1))+1)+1)+1

$\Rightarrow$ 2•(2•(2•((Two (*fix* Two)) 1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•((**λh** . λn . if n=0 then 1
else 2•h(n-1)+1) **(*fix* Two)** 1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•((**λn** . if n=0 then 1
else 2•(*fix* Two)(n-1)+1) **1**)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(if 1=0 then 1
else 2•((*fix* Two)(1-1))+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•(**(*fix* Two)** 0)+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•((Two (*fix* Two)) 0)+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•((**λh** . λn . if n=0 then 1
else 2•h(n-1)+1) **(*fix* Two)** 0)+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•((**λn** . if n=0 then 1
else 2•((*fix* Two) (n-1))+1) **0**)+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•(if 0=0 then 1
else 2•((*fix* Two) (0-1))+1)+1)+1)+1)+1

$\Rightarrow$ 2•(2•(2•(2•1+1)+1)+1)+1 = 31