CS 611
Advanced Programming Languages

Andrew Myers
Cornell University

Lecture 13
Domain Constructions
22 Sep 00

Administration

- Homework 2 due on Monday
- Scribes needed
- Winskel×2, Gunter available on reserve in Engineering library

Fixed points

- Denotational semantics for IMP rely on taking fixed point to define $\mathcal{C}[\![\mathbf{while}]\!]$
- Fixed points occur in most language definitions: needed to deal with loops
  – control flow loops: while
  – data loops: recursive functions, recursive data structures, recursive types
- Only know how to find least fixed pts for continuous functions *f*
- *Need easy way to ensure continuity*

Meta-language

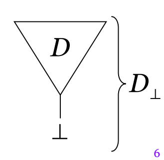
- Idea: define restricted language for expressing mathematical functions
- All functions expressible in this language are continuous
- Looks like a programming language (ML)
  – not executed: just mathematical notation
  – can talk about non-termination!
  – "evaluation" is lazy (vs. eager in ML)

"Types" for Meta-language

- Meta-language contains domain declarations indicating the set of values meta-variables can take on, e.g.
  $\lambda f\in\Sigma_\perp\to\Sigma_\perp.\ \lambda\sigma\in\Sigma_\perp.\ \mathit{if}\ \neg\mathcal{B}[\![b]\!]\ \sigma\ \mathit{then}\ \sigma\ \mathit{else}\ f\,(\mathcal{C}[\![c]\!])$
- Domains will function as types for meta-language
  – but with precisely defined meaning, ordering relation, etc.
  – $T_1 * T_2$ is not necessarily modeled by $T_1 \times T_2$!
- Meta-language consists of domains and associated operations

Lifting

- If D is a domain (for now: cpo), can "lift" by adding new bottom element to form pointed cpo $D_\perp$
- cpo defined by underlying set plus complete ordering relation $\sqsubseteq$
- Elements of $D_\perp$ are $\lfloor d_i \rfloor, \perp$ where $d_i \in D$
- Ordering relation:
  $\lfloor d_i \rfloor \sqsubseteq \lfloor d'_i \rfloor \iff d_i \sqsubseteq d_i$
  $\perp \sqsubseteq \lfloor d_i \rfloor$
- Complete?



Discrete cpos

- Various discrete cpos: booleans (**T**), natural numbers (ω), integers (**Z**), ...
- Corresponding functions over discrete cpos exist: $+ : \mathbf{Z}\to\mathbf{Z}$, $\wedge : \mathbf{T}\to\mathbf{T}$
- Often want to lift discrete cpos to take fixed points; helpful to extend fcns to pointed cpos
- If $f\in D\to E$, then $f_\perp\in D_\perp\to E_\perp$, $f^*\in D_\perp\to E$ are
  $f_\perp = \lambda d\in D_\perp.\mathit{if}\ d{=}\perp\ \mathit{then}\ \perp\ \mathit{else}\ f(d)$
  $f^* = \lambda d\in D_\perp.\mathit{if}\ d{=}\perp\ \mathit{then}\ \perp\ \mathit{else}\ f(d)$ (if E pointed)
- $2 +_\perp 2 = 4$, $3 +_\perp \perp = \perp$, $\perp \wedge_\perp \mathit{true} = \mathit{true}$
- If f continuous, are $f_\perp$, f^* ?

let

- Useful syntax: given $d\in D_\perp$

$$\mathit{let}\ x = d\ \mathit{in}\ e \quad\equiv\quad (\lambda x\in D.e)^* d$$

- Expresses evaluation of e that is *strict* in d
- *Example:* $\mathcal{C}[\![\textbf{while}]\!]$

$= \mathit{fix}\ \lambda f\in\Sigma_\perp\to\Sigma_\perp.\ \lambda\sigma'\in\Sigma_\perp.$
$\quad \mathit{let}\ \sigma{=}\sigma'\ \mathit{in}\ \mathit{if}\ \neg\mathcal{B}[\![b]\!]\ \sigma\ \mathit{then}\ \sigma\ \mathit{else}\ f\,(\mathcal{C}[\![c]\!]\,)$

Unit

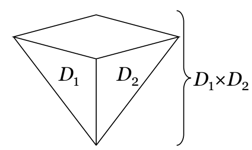
- Simplest cpo: empty set ($\varnothing$)
- Next simplest: *unit domain* (**U**)
  - single element: u
  - ordering relation: reflexive
  - complete: only directed set is {u}
- Used to represent computations that terminate but do not produce a value, argument for functions that need no argument
- Also building block for other domains

Hasse diagram

· u

Products

- If D_1, D_2 are domains, then $D_1\times D_2$ is a product domain
- Underlying set: pairs $\langle d_1, d_2\rangle$ where $d_i\in D_i$
- Ordering:
  $\langle d_1, d_2\rangle \sqsubseteq \langle d'_1, d'_2\rangle$ iff $d_1\sqsubseteq d'_1\ \&\ d_2\sqsubseteq d'_2$
- Extends to n-tuples
- Operations:
  - tupling: $\langle d_1,\ldots,d_m\rangle$
  - projection: $\pi_i\,\langle d_1,\ldots,d_m\rangle = d_i$



CPO?

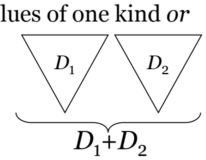
- Is product domain a cpo if D_1, D_2 are?
- Any chain $\langle d_0, d'_0\rangle \sqsubseteq \langle d_1, d'_1\rangle \sqsubseteq \langle d_2, d'_2\rangle \sqsubseteq \ldots$ must have LUB in $D_1\times D_2$
- Definition of $\sqsubseteq$: $d_0\sqsubseteq d_1\sqsubseteq d_2\sqsubseteq\ldots$ is chain in D_1, $d'_0\sqsubseteq d'_1\sqsubseteq d'_2\sqsubseteq\ldots$ is chain in D_2
- If $d_\infty\in D_1$, $d'_\infty\in D_2$ are respective LUBs, $\langle d_\infty, d'_\infty\rangle\in D_1\times D_2$ is LUB of chain of pairs
- Operations continuous?

$$\pi_i \bigsqcup_{n\in\omega} x_n = \bigsqcup \pi_i x_n = \bigsqcup d_{in}$$

$$\bigsqcup\langle x_{1n}, \ldots, x_{mn}\rangle = \langle \bigsqcup d_{1n}, \ldots, \bigsqcup d_{mn}\rangle$$

Sums

- Sometimes want to allow values of one kind *or* another: $D_1{+}D_2$
- Elements of domain are elements of D_1 or D_2 tagged with origin: $\{in_i(d_i) \mid d_i\in D_i\}$



- Form of in_i is irrelevant (could be $\lambda d.\langle i, d\rangle$)
- Preserves ordering of individual domains:
  $in_i(d_i) \sqsubseteq in_j(d_j)$ iff $i{=}j$, $d_i\sqsubseteq d_j$
- Injection function in_i is continuous
- Extends naturally to multi-domain sum
- CPO, but not pointed

Sums, cont'd

- Why tag? Distinguishes identical domains
  - $\mathbf{T} = \mathbf{U} + \mathbf{U}$, true = in_1(u), false = in_2(u)
- Sums unpacked with *case* construction:
  $\mathit{case}\ e\ \mathit{of}\ x_1.e_1 \mid x_2.e_2 \ \equiv\ \mathit{case}\ e\ \mathit{of}\ D_1(x_1).e_1 \mid D_2(x_2).e_2$
- Given $e = in_i(d_i)$, has value $f_i(d_i) \in E$ where $f_i \in D_i \rightarrow E = (\lambda x_i \in D_i\,.e_i)$
- Continuous function of e if all f_i continuous:

  $$\bigsqcup \mathit{case}\ e_n\ \mathit{of}\ \ldots = \mathit{case}\ \bigsqcup e_n\ \mathit{of}\ \ldots\ ?$$

  $$\bigsqcup f_i(d_{in}) = f_i(\bigsqcup d_{in})$$
- Also continuous function of each f_i

  $$\bigsqcup \mathit{case}\ e\ \mathit{of}\ f_{1n} \mid f_2 = \mathit{case}\ e\ \mathit{of}\ \bigsqcup f_{1n} \mid f_2 = \bigsqcup f_{1n}(d_1)$$

Continuous functions

- Given cpos D, E, define $D \rightarrow E$ as domain of *continuous* functions mapping D to E (subset of E^D)
- Pointwise ordering: $f \sqsubseteq g$ iff $f(d) \sqsubseteq g(d)$
- Complete?

  $$\bigsqcup_{n\in\omega} f_n = \lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d) \Big\}\ \textit{continuous?}$$

  $$(\lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d))\ (\bigsqcup_{m\in\omega} d_m) = \bigsqcup_{m\in\omega}\ (\lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d))\ \ (d_m)\ ?$$

Proof of Continuity

$$(\lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d))\ (\bigsqcup_{m\in\omega} d_m) =$$

$$\bigsqcup_{m\in\omega}\ (\lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d))\ \ (d_m)?$$

$$= \bigsqcup_{n\in\omega} f_n(\bigsqcup_{m\in\omega} d_m)$$

$$= \bigsqcup_{n\in\omega} \bigsqcup_{m\in\omega} f_n(d_m)$$

$$= \bigsqcup_{n\in\omega} f_n(d_n)$$

$$= \bigsqcup_{m\in\omega} \bigsqcup_{n\in\omega} f_n(d_m)$$

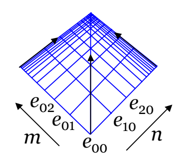
$$= \bigsqcup_{m\in\omega}\ (\lambda d \in D\,.\,\bigsqcup_{n\in\omega} f_n(d))\ \ (d_m)$$

Lemma

$$\bigsqcup_n \bigsqcup_m f_n(d_m) = \bigsqcup_n f_n(d_n) = \bigsqcup_m \bigsqcup_n f_n(d_m)$$

Let $e_{nm} = f_n(d_m)$

$n \le n'$, $m \le m' \Rightarrow e_{nm} \sqsubseteq e_{n'm'}$



$e_{nm} \sqsubseteq e_{n'n'}$ for $n' = \max(m,n)$, so $\bigsqcup_{n,m} e_{nm} \sqsubseteq \bigsqcup_n e_{nn}$

$e_{nn} \sqsubseteq \bigsqcup_m e_{nm}$, so $\bigsqcup_n e_{nn} \sqsubseteq \bigsqcup_n \bigsqcup_m e_{nm}, \bigsqcup_m \bigsqcup_n e_{nm}$

$\bigsqcup_m e_{nm} \sqsubseteq \bigsqcup_{n,m} e_{nm}$, so $\bigsqcup_n \bigsqcup_m e_{nm} \sqsubseteq \bigsqcup_{n,m} e_{nm}$

Operations on functions

- *apply* $\in (D \rightarrow E) \times D \rightarrow E = \lambda p.(\pi_1 p)(\pi_2 p)$
- *curry* $\in ((D \times E) \rightarrow F) \rightarrow (D \rightarrow E \rightarrow F)$
  $= \lambda f \in (D \times E) \rightarrow F\,.\,\lambda d \in D.\lambda e \in E.f\,\langle d,e \rangle$
- *compose* $= \cdot \circ \cdot \in (D \rightarrow E) \times (E \rightarrow F) \rightarrow (D \rightarrow F)$
  $= \lambda \langle f,g \rangle.\lambda d \in D.f\,(g(d))$
- *fix* $\in (D \rightarrow D) \rightarrow D$ (D pointed)
  $= \lambda g \in D \rightarrow D.\ \bigsqcup_n g^n(\bot)$
  $= \bigsqcup_n \lambda g \in D \rightarrow D.\ g^n(\bot)$ (LUB of cont.fcns!)

Meta-Language

- Have defined various constructs that we can use to define continuous functions
- Constructs are a syntax for a meta-language in which only continuous functions can be defined
- How do we know when expression $\lambda x.e$ is continuous?
- Idea: use structural induction on form of e so every syntactically valid e can be abstracted over any variable to produce continuous function
- Problem: structural induction $\Rightarrow$ need to consider open terms e

Continuity in variables

- Idea: consider a meta-language expression e to be implicitly function of its free variables
- e is continuous in variable x if $\lambda x.e$ is continuous for arbitrary values of other (non-x) free variables in e
- e is continuous in variables not free in e
- structural induction: for each syntactic form, show that term is continuous in variables assuming sub-terms are