Fundamental Property of LR for $\leq_{idw}$

If $\Gamma \vdash e : ty'$ with $\text{loc}(e) \subseteq \omega$, then $\Gamma \vdash e \leq_{idw} e : ty'$.

More generally, if $\Gamma, x : ty \vdash e : ty'$ with $\text{loc}(e) \subseteq \omega$, then

$$\Gamma \vdash e_1 \leq_{idw} e_2 \Rightarrow \Gamma \vdash e[e_1/x] \leq_{idw} e[e_2/x] : ty'$$

Proved by showing that each syntactic construct of the language preserves $\Gamma \vdash e_1 \leq_{r} e_2 : ty$ (see [15] Prop. 4.8).

For example...

If $\Gamma, f : ty_1 \rightarrow ty_2, x : ty_1 \vdash e \leq_{r} e' : ty_2$, then

$$\Gamma \vdash \text{fun } f = (x : ty_1) \rightarrow e \leq_{r} \text{fun } f = (x : ty_1) \rightarrow e' : ty_1 \rightarrow ty_2$$

This is proved via an important "compactness property" of $\langle s, \bar{f}s, e \rangle \downarrow$, namely ...
An unwinding theorem

Given $f : \text{ty}_1 \rightarrow \text{ty}_2$, $x : \text{ty}_1 \vdash e_2 : \text{ty}_2$, for each $0 \leq n \leq \omega$ define $f_n \in \text{Prog}_{\text{ty}_1 \rightarrow \text{ty}_2}$ by:

- $f_0 \triangleq \text{fun } f = (x : \text{ty}_1) \rightarrow fx$
- $f_{n+1} \triangleq \text{fun}(x : \text{ty}_1) \rightarrow e_2[f_n/f]$
- $f_\omega \triangleq \text{fun } f = (x : \text{ty}_1) \rightarrow e_2$.

Then for all $f : \text{ty}_1 \rightarrow \text{ty}_2 \vdash e : \text{ty}$ and all states $s$

$s, e[f_\omega/f] \downarrow \text{ iff } \exists n \geq 0. s, e[f_n/f] \downarrow.$

(proof: see OS&PE, Theorem 5.3)

Unwinding theorem implies

$f_\omega \leq_{\text{defx}} g \equiv \forall n ( f_n \leq_{\text{defx}} g )$

and more generally

$f_\omega \leq_r g \equiv \forall n ( f_n \leq_r g )$
Unwinding Theorem implies
\[ e[fw/f] \leq_{\text{ctx}} g \equiv \forall n (e[fn/f] \leq_{\text{ctx}} g) \]
and more generally
\[ e[fw/f] \leq_{r} g \equiv \forall n (e[fn/f] \leq_{r} g) \]

These “syntactic admissibility” properties provide a direct link with the use of chain-complete partial orders in denotational semantics.

Some observations

- Simple operational semantics does not imply simple properties! (in particular, properties of recursion can be subtle)

- Not all SOS’s are equally convenient for proofs

- The “ghost” of Domain Theory in operationally-based proof methods.
Second part of the course is based on section 3 of
AMP, "Relational Properties of Domains,"

(see also the Abramsky-Jung handbook chapter on Domain Theory)

Recursive Domain Equations

• why do we (semanticists) need to solve them? ...

• and why is it hard to do so?
Denotational semantics as a tool for reasoning about contextual equiv. $\simeq_{ctx}$

Require: mathematical structure $D$ plus operations on $D$ for the prog. lang. constructs permitting compositional definition of $[e] \in D$ denotation of program phrase $e$

that is at least computationally adequate:

$[e_1] = [e_2] \in D \Rightarrow e_1 \simeq_{ctx} e_2$

($[\_]=\_ \text{ coinciding with } \simeq_{ctx} \text{ is called full abstraction}$)

Denotational semantics as a tool for reasoning about contextual equiv. $\simeq_{ctx}$

Require: mathematical structure $D$ plus operations on $D$ for the prog. lang. constructs often(?) lead to use of recursively defined domains

given domain construction $D \rightarrow \Phi(D)$

seek domain $D = \text{rec } X. \Phi(X)$ which is "minimal" with property $D \simeq \Phi(D)$
Denotational semantics as a tool for reasoning about contextual equiv. \( \simeq_{\text{ctx}} \)

Require: mathematical structure \( D \) plus operations on \( D \) for the prog. lang. constructs (often?) lead to use of recursively defined domains

Given domain construction \( D \to \Phi(D) \)
Seek domain \( D = \text{rec } X. \Phi(X) \) which is "minimal" with property \( D \simeq \Phi(D) \)

Needed for computational adequacy results

Example

Domain \( E \) for denotations of expressions calculating an int using a storage location for holding codes of functions \( \text{int} \to \text{int} \)

E.g. of such an expression in Ocaml

\[
\begin{align*}
\text{let } y & = \text{ref } (\text{fun } x : \text{int} \to x) \text{ in} \\
y & = (\text{fun } x : \text{int} \to \text{if } x = 0 \text{ then } 1 \text{ else } x \times (\!y)(x - 1)) \\
(\!y) & 42
\end{align*}
\]

computes 42!
Example

Domain $E$ for denotations of expressions calculating an int using a storage location for holding codes of functions $\text{int} \to \text{int}$

\[
\begin{align*}
\text{denotations of expressions} & \quad E \cong S \to (\mathbb{Z} \times S) \\
\text{denotations of states} & \quad S \cong \mathbb{Z} \to E
\end{align*}
\]

So need $E \cong \Phi(E)$ where

\[
\Phi(-) \equiv (\mathbb{Z} \to (-)) \to (\mathbb{Z} \times (\mathbb{Z} \to (-))
\]

(If $\rightarrow$ means all partial $\Phi$s, then no such set $E$ exists, by Cantor.)
Classic example: untyped $\lambda$-calculus

Given iso $i : D \cong D \rightarrow D$ one can give denotations to $\lambda$-terms

\[ t ::= x \mid \lambda x . t \mid tt \]

as elements $\llbracket t \rrbracket_\rho \in D$

- $\llbracket x \rrbracket_\rho = \rho(x)$
- $\llbracket \lambda x . t \rrbracket_\rho = i^{-1}(d \in D \mapsto \llbracket t \rrbracket_\rho[x \mapsto d])$
- $\llbracket t t' \rrbracket_\rho = i(\llbracket t \rrbracket_\rho)(\llbracket t' \rrbracket_\rho)$

Classic example: untyped $\lambda$-calculus

Given iso $i : D \cong D \rightarrow D$ one can give denotations to $\lambda$-terms

\[ t ::= x \mid \lambda x . t \mid tt \]

as elements $\llbracket t \rrbracket_\rho \in D$

but there is no such set

\[ \emptyset \neq 1 \cong 0 \rightarrow 0 \; \text{; and if } |D| \geq 1, \text{ then } |D \rightarrow D| \geq |D\rightarrow 1| = |\mathcal{P}(D)| > |D| \quad \text{(Cantor)} \]
Scott || Plotkin (1969)

Denotational semantics in categories of domains = partially ordered sets with least element, lubs of chains, 
& continuous functions = monotone functions preserving lubs of chains

fewer functions allows possibility of things like \( D \sqsubseteq D \to D \)

---

Scott || Plotkin (1969)

"Limit-colimit" construction of \( \text{rec} X. \Phi (X) \) as inverse limit of posets

\[
D_0 \leftarrow \Phi (D_0) \leftarrow \Phi (D_1) \leftarrow \Phi (D_2) \leftarrow \cdots
\]

\[
\{ \bot \} \leftarrow \Phi (D_0) \leftarrow \Phi (D_1) \leftarrow \Phi (D_2) \leftarrow \cdots
\]

\[
\forall n. \quad \Phi (D_{n+1}) = \Phi (D_n)
\]

\[
\{ d \in \Pi_n D_n \mid d_n \}
\]
History - selected highlights

Ward; Lehmann; Smyth-Plotkin (1982)

Use of order-enriched category theory to provide a general framework for the limit-colimit construction of \( \text{rec} X. \Phi(X) \).

⇒ generalization to solving domain equations with parameters and recursively defined domain constructions ("nested datatypes", GADTs, ...)

History - selected highlights

Freyd (1992) Categorical axiomatization of \( \text{rec} X. \Phi(X) \) via notion of "algebraic compactness" & "free dialgebras".

⇒ simplified proofs of adequacy w.r.t. operational semantics (AMP)

induction/coinduction principles for recursive domains (Fiore, AMP)