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

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

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

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

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 \( \Gamma, f : ty_1 \rightarrow ty_2, x : ty_1 \vdash e \leq_e e' : ty_2 \), 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 \( \langle s, Fs, e \rangle \downarrow \), namely ...
An unwinding theorem

Given \( f : ty_1 \rightarrow ty_2, x : ty_1 \vdash e_2 : ty_2 \),
for each \( 0 \leq n \leq \omega \) define \( f_n \in \text{Prog}_{ty_1 \rightarrow ty_2} \) by:

\[
\begin{align*}
    f_0 & \triangleq \text{fun} \; f = (x : ty_1) \rightarrow f \; x \\
    f_{n+1} & \triangleq \text{fun} (x : ty_1) \rightarrow e_2[f_n/f] \\
    f_\omega & \triangleq \text{fun} \; f = (x : ty_1) \rightarrow e_2.
\end{align*}
\]

Then for all \( f : ty_1 \rightarrow ty_2 \vdash e : ty \) and all states \( s \)

\[
s, e[f_\omega/f] \downarrow \iff \exists n \geq 0. \; s, e[f_n/f] \downarrow.
\]

(proof: see OS & PE, Theorem 5.3)
Unwinding Theorem implies
\[ f_n \leq_{\text{ctx}} g \equiv \forall n \left( f_n \leq_{\text{ctx}} g \right) \]
and more generally
\[ f_n \leq_{r} g \equiv \forall n \left( f_n \leq_{r} g \right) \]
Unwinding Theorem implies
\[ e[fw/f] \leq_{dx} G \equiv \forall n (e[fw/f] \leq_{dx} G) \]
and more generally
\[ e[fw/f] \leq_r g \equiv \forall n (e[fw/f] \leq_r g) \]

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 "Relational Properties of Domains" in Information & Computation (27(1996)66-90).

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

- Why do we (semanticsists) 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:

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

($[I] = [I]$ coinciding with $\simeq_{ctx}$ is called full abstraction)
Denotational semantics as a tool for reasoning about contextual equiv. $\approx_{\text{ctx}}$

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

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

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

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

recursively defined domains

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

seek domain $D = \text{rec } X. \Phi(X)$ which is "minimal" with property $D \equiv \Phi(D)$

needed for computational adequacy results
Example

Domain $E$ for denotations of expressions calculating an `int` using a storage location for holding codes of functions `int → int`

E.g. of such an expression in OCaml:

```ocaml
let y = ref (fun (x: int) → x) in
  y := (fun (x: int) → if x=0 then 1 else x*(!y)(x-1));
(!y) 42 computes 42!
```
Example

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

\[
\{\text{denotations of expressions}\} \quad E \cong S \rightarrow (\mathbb{Z} \times S)
\]

\[
\{\text{denotations of states}\} \quad S \cong \mathbb{Z} \rightarrow E
\]
Example

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

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

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

\[
\Phi(-) \triangleq (\mathbb{Z} \rightarrow (\mathbb{Z} \rightarrow (\mathbb{Z} \rightarrow (\mathbb{Z} \rightarrow (-)))))
\]

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

Given iso \( i: D \cong D \to D \) one can give denotations to \( \lambda \)-terms

\[
\begin{align*}
t & ::= x \mid \lambda x.t \mid tt
\end{align*}
\]

as elements \( [t]\rho \in D \)

- \( [x]\rho = \rho(x) \)
- \( [\lambda x.t]\rho = i^{-1}(d \in D \mapsto [t](\rho[x \mapsto d])) \)
- \( [t\ t']\rho = i([t]\rho)([t']\rho) \)
Classic example: untyped λ-calculus

Given iso \( \iota : D \cong D \rightarrow D \) one can give denotations to λ-terms

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

as elements \( [t]_\iota \in D \)

but there is no such set \( 0 \cong 1 \cong \mathbb{O} \to \mathbb{O} \); and if \( |D| \geq 1 \), then

\[
|D \to D| \geq |D \to 1| = |\mathcal{P} D| > |D|
\]

(Cantor)
History - selected highlights

Scott II 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 $D \cong D \rightarrow D$

cf properties of $\leq_{ctx}$
History - selected highlights

Scott || Plotkin (1969)

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

as inverse limit of posets

\[
\begin{align*}
D_0 & \leftarrow D_1 \leftarrow D_2 \leftarrow \cdots & \text{rec} X. \Phi(X) \\
\{\perp\} & \Phi(D_0) \Phi(D_1) \cdots & \{d \in \prod_n D_n \mid \forall n. \prod_n (d_{n+1}) = d_n \}
\end{align*}
\]
Ward; Lehmann; Smyth-Plotkin (1982)

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

$\Rightarrow$ generalization to solving domain equations with parameters and recursively defined domain constructions ("nested data types", GADTs, ...)
Freyd (1992) Categorical axiomatization of rec\(X, \Phi(X)\) via notion of "algebraic compactness" & "free dialgebras".

\[\Rightarrow\] simplified proofs of adequacy w.r.t. operational semantics (AMP)

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