Fundamental Property of LR for $\leqslant_{id_w}$

If $\Gamma \vdash e : ty'$ with $loc(e) \subseteq w$,

then $\Gamma \vdash e \leqslant_{id_w} e : ty$.

More generally, if $\Gamma, x : ty \vdash e : ty'$ with $loc(e) \subseteq w$, then

$$\Gamma \vdash e_1 \leqslant_{id_w} e_2 \supset \Gamma \vdash e[e_1/x] \leqslant_{id_w} e[e_2/x] : ty'$$

Proved by showing that each syntactic construct of the language preserves $\Gamma \vdash e_1 \leqslant_r e_2 : ty$ (see [15] Propn 4.8).

For example...

If $\Gamma, f : ty_1 \to ty_2, x : ty_1 \vdash e \leq_r e' : ty_2$,

then

$$\Gamma \vdash (\text{fun } f = (x : ty_1) \to e) \leq_r (\text{fun } f = (x : ty_1) \to e') : ty_1 \to ty_2$$

This is proved via an important "compactness property" of $\langle s, \overline{Fs}, e \rangle \searrow$, namley ...

An unwinding theorem

Given $f : ty_1 \texttt{->} ty_2, x : ty_1 \vdash e_2 : ty_2$,
for each $0 \leq n \leq \omega$ define $f_n \in \mathbf{Prog}_{ty_1 \texttt{->} ty_2}$ by:

$$\begin{cases} f_0 & \triangleq \texttt{fun}\, f \texttt{ = (} x \texttt{ : } ty_1 \texttt{) -> } f\ x \\ f_{n+1} & \triangleq \texttt{fun(} x \texttt{ : } ty_1 \texttt{) -> } e_2[f_n/f] \\ f_\omega & \triangleq \texttt{fun}\, f \texttt{ = (} x \texttt{ : } ty_1 \texttt{) -> } e_2. \end{cases}$$

Then for all $f : ty_1 \texttt{->} ty_2 \vdash e : ty$ and all states s

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

(proof: see OS&PE, Theorem 5.3)

Unwinding Theorem implies

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

and more generally

$$f_\omega \leq_r g \equiv \forall n (f_n \leq_r g)$$

Unwinding Theorem implies

$$e[f_\omega/f] \leq_{ctx} g \equiv \forall n\,(e[f_n/f] \leq_{ctx} g)$$

and more generally

$$e[f_\omega/f] \leq_r g \equiv \forall n\,(e[f_n/f] \leq_r g)$$

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", Information & Computation 127(1996)66-90.

(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. $\cong_{ctx}$

Require: mathematical structure D plus operations on D for the prog. lang. constructs

permitting compositional definition of

$$\boxed{[\![e]\!] \in D \text{ denotation of program phrase } e}$$

that is at least computationally adequate:

$$\boxed{[\![e_1]\!] = [\![e_2]\!] \in D \supset e_1 \cong_{ctx} e_2}$$

($[\![\ ]\!] = [\![\ ]\!]$ <u>coinciding</u> with $\cong_{ctx}$ is called full abstraction)

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

"domain"

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 $D = \mathrm{rec}\, X.\, \Phi(X)$ which is "minimal" with property $D \cong \Phi(D)$

isomorphism

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

"domain"

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 $D = \mathrm{rec}\, X.\, \Phi(X)$ which is "minimal" with property $D \cong \Phi(D)$

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

```
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 int using a storage location for holding codes of functions int $\to$ int

denotations of expressions $\quad E \cong S \rightharpoonup (\mathbb{Z} \times S)$

denotations of states $\quad S \cong \mathbb{Z} \rightharpoonup E$

partial functions

Example

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

denotations of expressions $\quad E \cong S \to (\mathbb{Z} \times S)$

denotations of states $\quad S \cong \mathbb{Z} \to E$

partial functions

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

$$\Phi(-) \triangleq (\mathbb{Z} \to (-)) \to (\mathbb{Z} \times (\mathbb{Z} \to (-)))$$

(If $\to$ means all partial fns, then no such set E exists, by Cantor.)

Classic example: untyped λ-calculus

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

$$t ::= x \mid \lambda x.t \mid t\,t$$

as elements $[\![t]\!]\rho \in D$

environment mapping variables to elements of D

- $[\![x]\!]\rho \equiv \rho(x)$
- $[\![\lambda x.t]\!]\rho \equiv i^{-1}(d \in D \mapsto [\![t]\!](\rho[x \mapsto d]))$
- $[\![t\,t']\!]\rho \equiv i([\![t]\!]\rho)([\![t']\!]\rho)$

Classic example: untyped λ-calculus

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

$$t ::= x \mid \lambda x.t \mid t\,t$$

as elements $[\![t]\!] \rho \in D$

but there is no such <u>set</u>

$$\left(0 \not\cong 1 \cong 0 \to 0 \,;\ \text{and if } |D| \geq 1, \text{then} \right.$$

$$\left. |D \to D| \geq |D \to 1| = |\wp D| > |D| \right)$$

Cantor

History - selected highlights

Scott || Plotkin (1969)

cf properties of $\leq$ ctx

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 $D \cong D \to D$

History – selected highlights

Scott || Plotkin (1969)

"Limit-colimit" construction of $\mathrm{rec}\,X.\,\Phi(X)$

as inverse limit of posets

$$\begin{array}{ccccccc}
D_0 & \xleftarrow{\pi_0} & D_1 & \xleftarrow{\pi_1} & D_2 & \xleftarrow{\pi_2} \cdots & \mathrm{rec}\,X.\,\Phi(X) \\
\| & & \| & & \| & & \| \\
\{\perp\} & & \Phi(D_0) & & \Phi(D_1) & \cdots & \{ d \in \prod_n D_n \mid \forall n.\ \pi_n(d_{n+1}) = d_n \}
\end{array}$$

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 $\mathrm{rec}\, X . \Phi(X)$.

$\Longrightarrow$ generalization to solving domain equations with parameters and recursively defined domain constructions ("nested datatypes", GADTs, …)

History – selected highlights

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

$\Longrightarrow$ simplified proofs of adequacy w.r.t. operational semantics (AMP)

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