

# Minimal thunkification

Torben Amtoft

Kansas State University

# Outline of talk

1. Present a type inference system for strictness analysis, using annotations on the function arrows (a la Wright);
2. Use the strictness information to avoid some superfluous “thunkifications” when translating from CBN to CBV (folklore);
3. Give a combined correctness proof of the analysis/translation (in the trend of Wand).

# Our Typed Language

$e := x \mid c \mid e_1 e_2 \mid \lambda x. e \mid \text{if } e_1 \ e_2 \ e_3 \mid \text{rec } f \ e$

$$\Gamma \vdash c : Ct(c)$$

$$\Gamma \vdash x : \Gamma(x)$$

$$\frac{\Gamma \cup \{x := t_1\} \vdash e : t}{\Gamma \vdash \lambda x. e : t_1 \rightarrow t}$$

$$\frac{\Gamma \vdash e_1 : t_2 \rightarrow t_1 \text{ and } \Gamma \vdash e_2 : t_2}{\Gamma \vdash e_1 e_2 : t_1}$$

$$\frac{\Gamma \vdash e_1 : \text{Bool} \text{ and } \Gamma \vdash e_2 : t, \text{ and } \Gamma \vdash e_3 : t}{\Gamma \vdash (\text{if } e_1 \ e_2 \ e_3) : t}$$

$$\frac{\Gamma \cup \{f := t\} \vdash e : t}{\Gamma \vdash (\text{rec } f \ e) : t}$$

# Strictness types [Wright]

$$t := \text{Base} \mid t_1 \rightarrow_0 t_2 \mid t_1 \rightarrow_1 t_2$$

Want to impose an ordering  $\leq$ , such that if  $t_1 \leq t_2$  then  $t_1$  is more precise than  $t_2$ :

$$\begin{array}{l} \text{Int} \rightarrow_0 \text{Int} \\ \leq \\ \text{Int} \rightarrow_1 \text{Int} \end{array}$$

$$\begin{array}{l} (\text{Int} \rightarrow_1 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_0 \text{Int}) \\ \leq \\ (\text{Int} \rightarrow_0 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_0 \text{Int}) \end{array}$$

So the usual co/contravariant ordering:

$$\text{Base} \leq \text{Base}$$

$$t_1 \rightarrow_b t_2 \leq t'_1 \rightarrow_{b'} t'_2 \text{ iff } t'_1 \leq t_1, b \leq b', t_2 \leq t'_2.$$

# Judgements $\Gamma \vdash_{sa} e : t, W$

$x \in W \Rightarrow$  “ $x$  needed to evaluate  $e$  to HNF”.

$$\frac{\Gamma \vdash_{sa} e : t, W}{\Gamma \vdash_{sa} e : t', W'} \text{ if } t \leq t', W' \subseteq W$$

$$\Gamma \vdash_{sa} c : CT_{sa}(c), \emptyset$$

$$\Gamma \vdash_{sa} x : \Gamma(x), \{x\}$$

$$\frac{\Gamma \cup \{x := t_1\}, \vdash_{sa} e : t, W}{\Gamma \vdash_{sa} \lambda x. e : t_1 \rightarrow_b t, W \setminus \{x\}} \text{ where } b = 0 \text{ iff } x \in W$$

$$\frac{\Gamma \vdash_{sa} e_1 : t_2 \rightarrow_b t_1, W_1 \quad \Gamma \vdash_{sa} e_2 : t_2, W_2}{\Gamma \vdash_{sa} e_1 e_2 : t_1, W_1 \cup W'_2}$$

where  $W'_2 = W_2$  if  $b = 0$ , otherwise  $\emptyset$

# Judgements (II)

$$\frac{\Gamma \vdash_{sa} e_1 : \text{Bool}, W_1 \quad \Gamma \vdash_{sa} e_2 : t, W_2 \quad \Gamma \vdash_{sa} e_3 : t, W_3}{\Gamma \vdash_{sa} (\text{if } e_1 \ e_2 \ e_3) : t, W_1 \cup (W_2 \cap W_3)}$$

$$\frac{\Gamma \cup \{f := t\} \vdash_{sa} e : t, W}{\Gamma \vdash_{sa} (\text{rec } f \ e) : t, W} \quad \text{if } f \notin W$$

# Properties of Type System

Expressions having standard types also have strictness types  
(just use  $\rightarrow_1$  everywhere)

No least typing: with  $twice = \lambda f.\lambda x.f(f(x))$

$\emptyset \vdash_{sa} twice : (\text{Int} \rightarrow_0 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_0 \text{Int}), \emptyset$

$\emptyset \vdash_{sa} twice : (\text{Int} \rightarrow_1 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_1 \text{Int}), \emptyset$

But for each assignment to the arrows in contravariant position,  
there exists a least assignment to the covariant.

# Examples (no formal comparison)

Define  $f$ , strict in all arguments, by

$$\text{rec } f \lambda x.\lambda y.\lambda z.\text{if } (z = 0) (x + y) (f \ y \ x \ (z - 1))$$

In [Kuo,Mishra]'s framework this is coded as

$$0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \wedge 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \wedge 1 \rightarrow 1 \rightarrow 0 \rightarrow 0.$$

$f$  has strictness type  $\text{Int} \rightarrow_0 \text{Int} \rightarrow_0 \text{Int} \rightarrow_0 \text{Int}$ .

Weak point: recursion with free variables.

$$g = \lambda y.\text{rec } f \lambda x.\text{if } (x = 0) \ y \ (f \ (x - 1))$$

$ge_1e_2$  loops if  $e_1$  loops.

But we can only infer  $g : \text{Int} \rightarrow_1 \text{Int} \rightarrow_0 \text{Int}$ . since we cannot record in  $\Gamma$  that  $f$  needs  $y$ .

Proposal: annotate arrows not only with 0/1 but also with which free variables are needed

# Application: translation from CBN to CBV

Two tools:

- “Thunkification”:  $e \Rightarrow \lambda z.e$  (shorthand:  $\underline{e}$ )
- “Dethunkification”:  $x \Rightarrow (x \ d)$  (shorthand:  $\mathcal{D}(x)$ )

$\mathcal{D}(\underline{e})$  evaluates in one step to  $e$

Naive translation algorithm  $T$ :

- All applications:  $T(e_1 \ e_2) = T(e_1) \ \underline{T(e_2)}$ .
- All variables:  $T(x) = \mathcal{D}(x)$ .

Idea: only thunkify non-strict applications.

# Type considerations

Suppose  $e$  has strictness type  $t$ . Then

- $e$  as a CBN-term has ordinary type  $E(t)$
- $e$  should be translated into a CBV-term of ordinary type  $Z(t)$ .

Here  $E$  and  $Z$  are defined by

$$\begin{aligned}E(\text{Base}) &= \text{Base} \\E(t_1 \rightarrow_0 t_2) &= E(t_1) \rightarrow E(t_2) \\E(t_1 \rightarrow_1 t_2) &= E(t_1) \rightarrow E(t_2)\end{aligned}$$

$$\begin{aligned}Z(\text{Base}) &= \text{Base} \\Z(t_1 \rightarrow_0 t_2) &= Z(t_1) \rightarrow Z(t_2) \\Z(t_1 \rightarrow_1 t_2) &= [Z(t_1)] \rightarrow Z(t_2)\end{aligned}$$

Here  $[t] = \text{Unit} \rightarrow t$ .

# Translation (I)

Transformation of  $e$  defined inductively in the proof tree for  $\Gamma \vdash_{sa} e : t, W$ :

- Suppose  $\Gamma \vdash_{sa} e_1 e_2 : t_1, W$  because

$$\Gamma \vdash_{sa} e_1 : t_2 \rightarrow_b t_1, W_1 \text{ and } \Gamma \vdash_{sa} e_2 : t_2, W_2$$

Suppose  $e_1, e_2$  translate into  $e'_1, e'_2$ .

If  $b = 0$ ,  $e$  translates into  $e'_1 e'_2$ .

If  $b = 1$ ,  $e$  translates into  $e'_1 \underline{e'_2}$ .

- $\lambda x.e$  translates into  $\lambda x.e'$ .
- $x$  translates into  $x$ , if  $x$  has been captured by a strict  $\lambda$ .  
 $x$  translates into  $\mathcal{D}(x)$  otherwise.
- ...

# Translation (II)

Suppose

$$\Gamma \vdash_{sa} e : t', W'$$

because with  $t \leq t'$  and  $W' \subseteq W$

$$\Gamma \vdash_{sa} e : t, W$$

Suppose  $e$  translates into  $e'$  using the latter proof tree. Then using the former proof tree  $e$  translates into  $e'' = \dots???$

Example: let  $t = \text{Int} \rightarrow_0 \text{Int}$ ,  $t' = \text{Int} \rightarrow_1 \text{Int}$ .

$e'$  has type  $Z(t) = \text{Int} \rightarrow \text{Int}$ .  $e''$  should have type  $Z(t') = [\text{Int}] \rightarrow \text{Int}$ .

Choose  $e'' = \lambda z. (e \mathcal{D}(z))$ .

We can define  $C_t^{t'}$ . Then, if  $t = t_1 \rightarrow_0 t_2$  and  $t' = t'_1 \rightarrow_1 t'_2$  (with  $t'_1 \leq t_1$ ,  $t_2 \leq t'_2$ ) then

$$e'' = \lambda z. C_{t_2}^{t'_2} (e C_{t'_1}^{t_1} (\mathcal{D}(z)))$$

# Optimality issues

Translation not optimal:

$twice = \lambda f. \lambda x. f(fx)$  has strictness type

$$(\text{Int} \rightarrow_1 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_1 \text{Int})$$

and by the algorithm translates into

$$\lambda f. \lambda x. f f \underline{\underline{\mathcal{D}(x)}}$$

$\mathcal{D}(x)$  could be replaced by  $x$

Use context information?

$twice$  also has strictness type

$$(\text{Int} \rightarrow_0 \text{Int}) \rightarrow_0 (\text{Int} \rightarrow_0 \text{Int})$$

and then translates into itself.

# Proving Correctness

Translation folklore [Danvy etc.]; **correctness proof not.**

**Goal:** to show the correctness of the translation, at the same time in some sense showing that the analysis is correct (but avoiding model-theoretic details).

[Wand POPL'93]: The goal of flow analysis is to annotate a program with certain propositions about the behavior of that program. One can then apply optimizations to the program that are justified by those propositions. However, it has proven remarkably difficult to specify the semantics of those propositions in a way that justifies the resulting optimizations.

# Semantics (SOS), CBN

WHNF (wrt. CBN as well as CBV):  $c$  or  $\lambda x.e$ .

$$(\lambda x.e)q \Rightarrow_N e[q/x]$$

$$\frac{q_1 \Rightarrow_N q'_1}{q_1 q_2 \Rightarrow_N q'_1 q_2}$$

$$c_1 c_2 \Rightarrow_N \text{Applycon}(c_1, c_2)$$

$$\frac{q_2 \Rightarrow_N q'_2}{c q_2 \Rightarrow_N c q'_2}$$

$$\text{if True } q_2 \ q_3 \Rightarrow_N q_2$$

$$\text{if False } q_2 \ q_3 \Rightarrow_N q_3$$

$$\frac{q_1 \Rightarrow_N q'_1}{\text{if } q_1 \ q_2 \ q_3 \Rightarrow_N \text{if } q'_1 \ q_2 \ q_3}$$

$$\text{rec } f \ e \Rightarrow_N e[(\text{rec } f \ e)/f]$$

# Semantics (SOS), CBV

$(\lambda x.e)q \Rightarrow_V e[q/x]$ , if  $q$  in WHNF

$$\frac{q_1 \Rightarrow_V q'_1}{q_1 q_2 \Rightarrow_V q'_1 q_2}$$

$c_1 c_2 \Rightarrow_V \text{Applycon}(c_1, c_2)$

$$\frac{q_2 \Rightarrow_V q'_2}{q_1 q_2 \Rightarrow_V q_1 q'_2}, \text{ if } q_1 \text{ in WHNF}$$

if True  $q_2 q_3 \Rightarrow_V q_2$

if False  $q_2 q_3 \Rightarrow_V q_3$

$$\frac{q_1 \Rightarrow_V q'_1}{\text{if } q_1 \text{ } q_2 \text{ } q_3 \Rightarrow_V \text{if } q'_1 \text{ } q_2 \text{ } q_3}$$

$\text{rec } f \text{ } e \Rightarrow_V e[(\text{rec } f \text{ } e)/f]$

# Correctness predicates (I)

First consider *closed* expressions  $q$ .

Relation  $\sim_t$ ,  $t$  a strictness type.

$q \sim_t q'$ :  $q'$  is a correct translation of  $q$ ; here  $q$  has ordinary type  $E(t)$  and  $q'$  has ordinary type  $Z(t)$ .

# Correctness predicates (II)

Definition of  $q \sim_t q'$ :

•  $q \sim_{\text{Base}} q'$  iff

$$q \Rightarrow_N^* c \text{ iff } q' \Rightarrow_V^* c$$

•  $q \sim_{t_1 \rightarrow_0 t_2} q'$  iff

$$q_1 \sim_{t_1} q'_1 \Rightarrow qq_1 \sim_{t_2} q' q'_1$$

•  $q \sim_{t_1 \rightarrow_1 t_2} q'$  iff

$$q_1 \sim_{t_1} q'_1 \Rightarrow qq_1 \sim_{t_2} q' \underline{q'_1}$$

Logical relation, but adapted to translation – cf. Wand, POPL '93:

This work suggests that the proposition associated with a flow analysis can simply be that “the optimization works”.

# Extending Correctness to open terms

Suppose  $\Gamma \vdash_{sa} e : t, W$ , and suppose  $T$  are the variables in the domain of  $\Gamma$  which are captured by a non-strict  $\lambda$ . Then  $e'$  is a correct translation of  $e$  provided

1. Suppose  $q_i \sim_{\Gamma(i)} q'_i$ . Then

$$\begin{aligned} & e[\{q_1 \dots q_n\} / \{x_1 \dots x_n\}] \\ \sim_t & e'[\{Q'_1 \dots Q'_n\} / \{x_1 \dots x_n\}] \end{aligned}$$

with  $Q'_i = q'_i$  if  $x_i \notin T$ ;  $Q'_i = \underline{q'_i}$  if  $x_i \in T$ .

2. Suppose  $x_i \in W$ , and  $q_i \sim_{\Gamma(i)} \Omega$ . Then

$$e[\{q_1 \dots q_n\} / \{x_1 \dots x_n\}] \sim_t \Omega$$

# Proof sketch

1. A number of properties of  $\sim_t$  are proved (by induction on  $t$ ). For instance, we have that if  $q \Rightarrow_N q_1$  and  $q \sim_t q'$  then also  $q_1 \sim_t q'$ .
2. The translation is proved correct by induction in the proof tree for  $\Gamma \vdash_{sa} e : t, W$ .

Only problem: the SOS-rule

$$\text{rec } f \ e \Rightarrow_N e[(\text{rec } f \ e)/f]$$

where we want to (inductively) use properties of the latter  $\text{rec}$  to prove properties of the former  $\text{rec}$ . Solution: introduce “bounded recursion”, with rules like

$$\text{rec}_{n+1} \ f \ e \Rightarrow_N e[(\text{rec}_n \ f \ e)/f]$$

# Concluding remarks

- Analysis and translation proved correct simultaneously.
- Other applications of the above paradigm?
- There exists a type inference algorithm, using constraints.
- A system has been implemented.