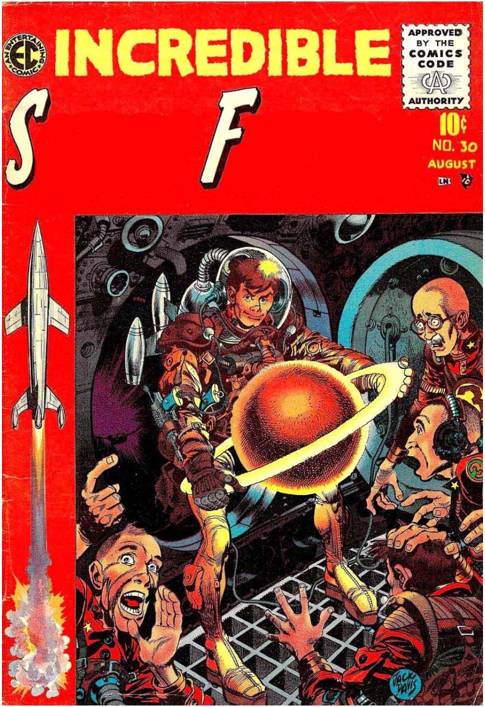
Control Flow Analysis for SF Combinator Calculus

Martin Lester

Department of Computer Science,
University of Oxford

Verification and Program Transformation,
2015–04–11



Motivation

- Programs that transform other programs are a form of metaprogramming.
- In most languages, program code is not a first-class citizen.
  - This makes formal reasoning about the behaviour of program-transforming programs difficult.
- Recent work on analysis of metaprogramming focuses mainly on extensional operations, such as composition of well-formed code templates.
  - Many program transformations are intensional: they decompose code.
- SF Combinator Calculus succinctly expresses intensional operations.
  - Let's try to analyse that too.
- 0CFA is a well-understood and widely-used program analysis.
  - Let's try to formulate it for SF Combinator Calculus.

Outline

SK Combinator Calculus — Review

SF Combinator Calculus is similar to SK Combinator Calculus, so let us first review that.

SK Combinator Calculus is a Turing-powerful model of computation [HS08].

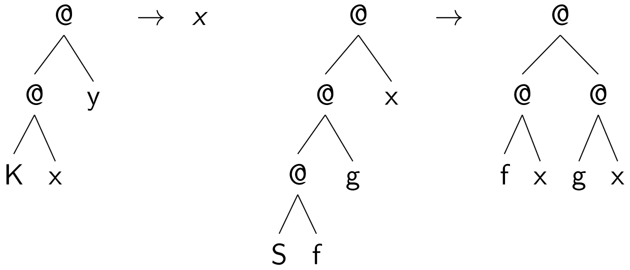
Consider terms:

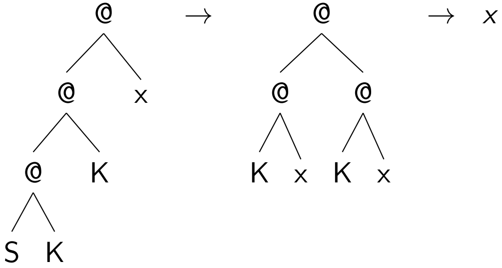
- built from two combinators S and K...
  - ...each with an associated rewrite rule:
  - $S\ f\ g\ x \to f\ x\ (g\ x)$
  - $K\ x\ y \to x$
- using application;
- viewed as trees.

Then:

- a term is a function or a program;
- a sequence of rewrites is execution of a program.

SK Combinator Calculus — Terms as Trees





SK Combinator Calculus — and λ-Calculus

How does SK-calculus relate to λ-calculus?

- S and K can be translated into λ-calculus by *lambda*:
  - $lambda(S) \equiv \lambda f.\lambda g.\lambda x.f\ x\ (g\ x)$
  - $lambda(K) \equiv \lambda x.\lambda y.x$
- Any closed term e of λ-calculus can be written as a purely applicative term t built from S and K using a translation *unlambda*.
  - *unlambda* is left-inverse to *lambda*:

$$t \quad \xrightarrow{lambda} \quad e \quad \xrightarrow{unlambda} \quad t$$

  - But in general:

$$e \quad \xrightarrow{unlambda} \quad t \quad \overset{lambda}{\not\longrightarrow} \quad e$$

  - The relationship is preserved by reduction:

$$\begin{array}{ccc} t & \xrightarrow{reduce} & t' \\ lambda\downarrow & & \uparrow unlambda \\ e & \xrightarrow{reduce^+} & e' \end{array}$$

SK Combinator Calculus — Advantages and Disadvantages

Advantages:

- no need to reason about bound variables;
- all transformations are local;
- practical as an "assembly language" for functional programs.

Disadvantages:

- hard for humans to read.
- can be larger than equivalent λ-terms (unless extended combinator set is used).

SF Combinator Calculus — Reductions

S behaves the same as in SK-calculus.

F allows factorisation of its first argument.

$$\begin{array}{rcll} F\ f\ x\ y & \to & x & \text{if } f = S \text{ or } f = F \\ F\ (u\ v)\ x\ y & \to & y\ u\ v & \text{if } u\ v \text{ is a factorable form} \end{array}$$

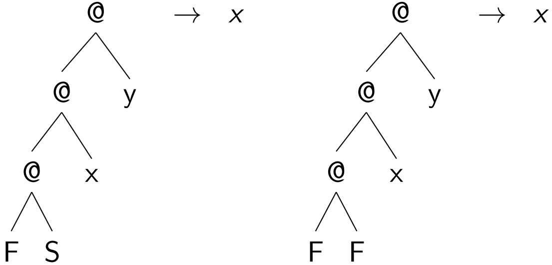
A factorable form is a term of form S, $S\ u$, $S\ u\ v$, F, $F\ u$ or $F\ u\ v$.

Properties:

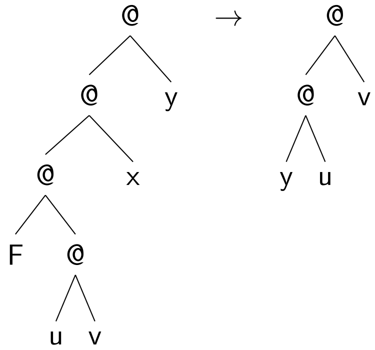
- By encoding K as $F\ F$, we get all the power of SK-calculus.
- We can check for equality of terms in normal form.
- We can distinguish between two terms that compute the same function in a different way.
- The restriction to factorable forms ensures confluence, hence a consistent equational theory [GJ11].
- Adding types in the style of System F, we can type a self-interpreter for SF-calculus [JP11].

Claim: SF-calculus is a good formalism for writing program-transforming programs.

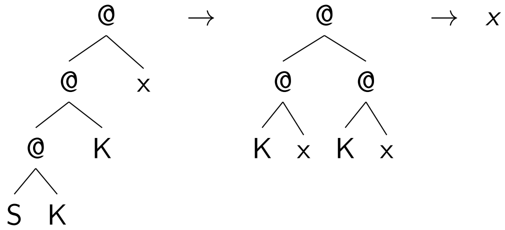
SF Combinator Calculus — Terms as Trees

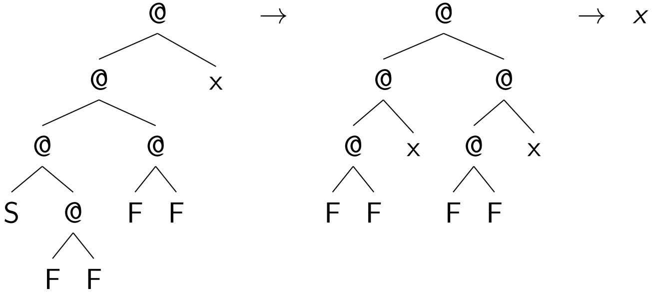


SF Combinator Calculus — Terms as Trees



SF Combinator Calculus — Terms as Trees





0CFA for λ-calculus — Overview

0CFA is a Control Flow Analysis [Mid12]:

- For each variable x in a program, it tell us what functions can be bound to x.
- This is important in functional programs, where functions determine control flow.
- Numerous applications, including compiler optimisations and as the foundation for other analysis, such as abstract interpretation frameworks.

Properties of 0CFA:

- Practical time complexity: $\mathcal{O}(n^3)$ or better.
- Necessarily some imprecision; this arises when the same function is called from two different places.

More advanced analyses:

- k-CFA adds k levels of calling context.
  - When $k = 0$, this is 0CFA.
  - For $k \geq 1$, precision improves, but EXPTIME-complete.
- CFA2 uses a pushdown abstraction.
  - Often faster and more precise than k-CFA.

0CFA for λ-calculus — Analysis Rules

Give each subexpression a distinct label l.
Generate and solve set constraints over Γ.

Labels	$Label$	$\ni$	l		
Variables	Var	$\ni$	x		
Labelled Expressions			e	$::=$	$x^l \mid e_1 @^l e_2 \mid \lambda^l x.e$
Abstract Values	Abs	$\ni$	v	$::=$	$FUN(x, l)$
Abstract Environment			Γ	$:$	$Label \uplus Var \to \mathcal{P}(Abs)$

$$
\begin{array}{lcl}
\Gamma \models x^l & \iff & \Gamma(x) \subseteq \Gamma(l) \\
\Gamma \models \lambda^{l_1} x.e^{l_2} & \iff & \Gamma \models e^{l_2} \wedge FUN(x, l_2) \in \Gamma(l_1) \\
\Gamma \models e_1^{l_1} @^l e_2^{l_2} & \iff & \Gamma \models e_1^{l_1} \wedge \Gamma \models e_2^{l_2} \wedge \\
& & (\forall FUN(x, l_3) \in \Gamma(l_1). \Gamma(l_2) \subseteq \Gamma(x) \wedge \Gamma(l_3) \subseteq \Gamma(l))
\end{array}
$$

- $FUN(x, l_2)$ — any function that binds x with body labelled l_2
- $FUN(x, l_2) \in \Gamma(l_1)$ — such a function could occur at l_1
- $\Gamma(l_3) \subseteq \Gamma(l)$ — anything that occurs at l_3 could occur at l

0CFA for SK-calculus

To work out how to do 0CFA for SF-calculus, let us first look at SK-calculus.

- 0CFA tracks which functions can be bound to which variables.
- How do we do that with no variables?
- Easy: Just use the *lambda* translation.
- But that won't help us with SF-calculus.
- So reformulate the constraints to get rid of variable binding.
- Give each combinator and application a distinct label n or l.
- Key idea: Abstract values indicate a node's local left children in weak normal form.
  - $S_0^n \in \Gamma(l)$ — S^n may occur at the node labelled l
  - $S_1^n \in \Gamma(l)$ — S^n may occur at l's left child
  - $S_2^n \in \Gamma(l)$ — S^n may occur at l's left child's left child
  - S_3^n — not needed, as it can never occur in normal form
  - $\varphi(n)$ — if true, S^n might be reduced

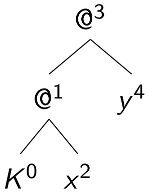
0CFA for SK-calculus — Analysis Rules

$$
\begin{array}{llcl}
\text{Base Labels} & & n & \\
\text{Sublabel Names} & & s & ::= \; S.0 \mid S.1 \mid S.2 \mid S.3 \mid S.L \mid S.R \mid K.0 \\
\text{Labels} & \mathit{Label} \ni & l & ::= \; n \mid n.s \\
\text{Labelled Terms} & & t & ::= \; S^n \mid K^n \mid t_1 @^l t_2 \mid \langle x \rangle^l \\
\text{Abstract Values} & \mathit{Abs} \ni & v & ::= \; S_0^n \mid S_1^n \mid S_2^n \mid K_0^n \mid K_1^n \\
\text{Abstract Environment} & & \Gamma & : \; \mathit{Label} \to \mathcal{P}(\mathit{Abs}) \\
\text{Abstract Activation} & & \varphi & : \; \mathit{Label} \to \mathit{Bool}
\end{array}
$$

$$
\begin{array}{lcl}
\Gamma, \varphi \models S^n & \iff & S_0^n \in \Gamma(n) \wedge (\varphi(n) \Rightarrow \Gamma, \varphi \models t_{S^n}) \\
\Gamma, \varphi \models K^n & \iff & K_0^n \in \Gamma(n) \\
\Gamma, \varphi \models t_1^{l_1} @^{l_3} t_2^{l_2} & \iff & \quad \Gamma, \varphi \models t_1 \wedge \Gamma, \varphi \models t_2 \\
& & \wedge \forall S_0^n \in \Gamma(l_1).\Gamma(l_2) \subseteq \Gamma(n.S.0) \wedge S_1^n \in \Gamma(l_3) \\
& & \wedge \forall S_1^n \in \Gamma(l_1).\Gamma(l_2) \subseteq \Gamma(n.S.1) \wedge S_2^n \in \Gamma(l_3) \\
& & \wedge \forall S_2^n \in \Gamma(l_1).\Gamma(l_2) \subseteq \Gamma(n.S.2) \wedge \Gamma(n.S.3) \subseteq \Gamma(l_3) \wedge \varphi(n) \\
& & \wedge \forall K_0^n \in \Gamma(l_1).\Gamma(l_2) \subseteq \Gamma(n.K.0) \wedge K_1^n \in \Gamma(l_3) \\
& & \wedge \forall K_1^n \in \Gamma(l_1).\Gamma(n.K.0) \subseteq \Gamma(l_3) \\
\Gamma, \varphi \models \langle x \rangle^l & \iff & \mathit{true}
\end{array}
$$

$$
t_{S^n} \stackrel{\mathit{def}}{=} (\langle f \rangle^{n.S.0} @^{n.S.L} \langle x \rangle^{n.S.2}) @^{n.S.3} (\langle g \rangle^{n.S.1} @^{n.S.R} \langle x \rangle^{n.S.2})
$$

0CFA for SK-calculus — Analysis Rules



At K^0: $K_0^0 \in \Gamma(0)$

$\Downarrow$

At $@^1$: $K_1^0 \in \Gamma(1)$ $\qquad \Gamma(2) \subseteq \Gamma(0.K.0)$

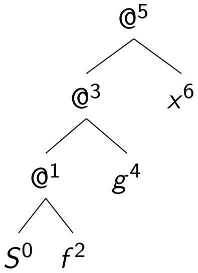
$\Downarrow$

At $@^3$: $\Gamma(0.K.0) \subseteq \Gamma(3)$

$\Downarrow$

So: $\Gamma(2) \subseteq \Gamma(0.K.0) \subseteq \Gamma(3)$

0CFA for SK-calculus — Analysis Rules



$$\text{At } S^0: \quad S^0_0 \in \Gamma(0)$$

$$\Downarrow$$

$$\text{At } @^1: \quad S^0_1 \in \Gamma(1) \quad \Gamma(2) \subseteq \Gamma(0.S.0)$$

$$\Downarrow$$

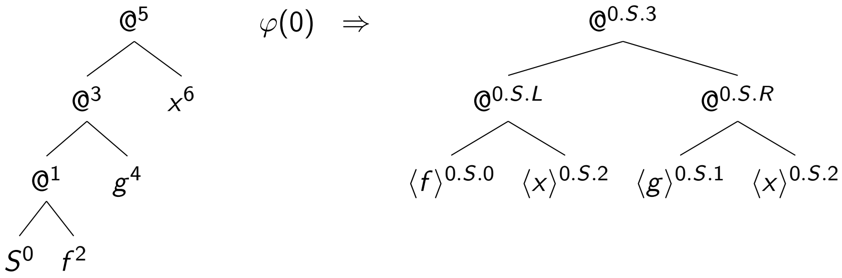
$$\text{At } @^3: \quad S^0_2 \in \Gamma(3) \quad \Gamma(4) \subseteq \Gamma(0.S.1)$$

$$\Downarrow$$

$$\text{At } @^5: \quad \varphi(0) \quad \Gamma(6) \subseteq \Gamma(0.S.2) \quad \Gamma(0.S.3) \subseteq \Gamma(5)$$

$$\Downarrow$$

0CFA for SK-calculus — Analysis Rules



Suppose $f = K$:

$$K_0^2 \in \Gamma(2)$$
$$\Downarrow$$
$$K_0^2 \in \Gamma(0.S.0)$$
$$\Downarrow$$
$$\Gamma(0.S.2) \subseteq \Gamma(0.S.3)$$

$$So: \quad \Gamma(6) \subseteq \Gamma(0.S.2) \subseteq \Gamma(0.S.3) \subseteq \Gamma(5)$$

0CFA for SF-calculus — Challenges

S behaves the same in SK-calculus and SF-calculus, so use the same rules.

How can we handle F? There are two challenges:

- How do we determine which reduction is used?
  - Abstract values already record how many arguments a combinator has.
  - Atoms have 0 arguments — abstractly S_0^n and F_0^n.
  - Compounds have ≥ 1 argument — $S_1^n, S_2^n, F_1^n, F_2^n$.
- How do we factorise abstractly?
  - Add new abstract values that record the source of an application.
  - $@^{l_1,l_2} \in \Gamma(l_3)$ — the subtree at l_3 might have been built by applying the subtree at l_1 to the subtree at l_2.

0CFA for SF-calculus — Analysis Rules

Base Labels			n		
Sublabel Names			s	$::=$	$S.0 \mid S.1 \mid S.2 \mid S.3 \mid S.L \mid S.R \mid$ $F.0 \mid F.1 \mid F.2 \mid F.3 \mid F.L \mid F.R \mid F.M$
Labels	$Label$	$\ni$	l	$::=$	$n \mid n.s$
Labelled Terms			t	$::=$	$S^n \mid F^n \mid t_1 @^l t_2 \mid \langle x \rangle^l$
Abstract Values	Abs	$\ni$	v	$::=$	$S_0^n \mid S_1^n \mid S_2^n \mid F_0^n \mid F_1^n \mid F_2^n \mid @^{(l_1,l_2)}$
Abstract Environment			Γ	$:$	$Label \to \mathcal{P}(Abs)$
Abstract Activation			φ	$:$	$Label \to Bool$

$$\Gamma, \varphi \models S^n \iff S_0^n \in \Gamma(n) \wedge (\varphi(n) \Rightarrow \Gamma, \varphi \models t_{S^n})$$

$$\begin{aligned}
\Gamma, \varphi \models F^n \iff & \ F_0^n \in \Gamma(n) \\
& \wedge \varphi(n) \Rightarrow (\exists n_0 . S_0^{n_0} \in \Gamma(n.F.0) \vee F_0^{n_0} \in \Gamma(n.F.0)) \Rightarrow \Gamma(n.F.1) \subseteq \Gamma(n.F.3) \\
& \wedge \varphi(n) \Rightarrow (\exists n_0 . S_1^{n_0} \in \Gamma(n.F.0) \vee S_2^{n_0} \in \Gamma(n.F.0) \vee \\
& \qquad F_1^{n_0} \in \Gamma(n.F.0) \vee F_2^{n_0} \in \Gamma(n.F.0)) \Rightarrow \Gamma, \varphi \models t_{F^n} \wedge \\
& \qquad \forall @^{l_1,l_2} \in \Gamma(n.F.0) . \Gamma(l_1) \subseteq \Gamma(n.F.L) \wedge \Gamma(l_2) \subseteq \Gamma(n.F.R)
\end{aligned}$$

$$\begin{aligned}
\Gamma, \varphi \models t_1^{l_1} @^{l_3} t_2^{l_2} \iff & \ \Gamma, \varphi \models t_1 \wedge \Gamma, \varphi \models t_2 \\
& \wedge \exists @^{l_4,l_5} \in \Gamma(l_3) . \Gamma(l_1) \subseteq \Gamma(l_4) \wedge \Gamma(l_2) \subseteq \Gamma(l_5) \\
& \wedge \forall S_0^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.S.0) \wedge S_1^n \in \Gamma(l_3) \\
& \wedge \forall S_1^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.S.1) \wedge S_2^n \in \Gamma(l_3) \\
& \wedge \forall S_2^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.S.2) \wedge \Gamma(n.S.3) \subseteq \Gamma(l_3) \wedge \varphi(n) \\
& \wedge \forall F_0^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.0) \wedge F_1^n \in \Gamma(l_3) \\
& \wedge \forall F_1^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.1) \wedge F_2^n \in \Gamma(l_3) \\
& \wedge \forall F_2^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.2) \wedge \Gamma(n.F.3) \subseteq \Gamma(l_3) \wedge \varphi(n)
\end{aligned}$$

$$\Gamma, \varphi \models \langle x \rangle^l \iff true$$

$$t_{S^n} \stackrel{def}{=} (\langle f \rangle^{n.S.0} @^{n.S.L} \langle x \rangle^{n.S.2}) @^{n.S.3} (\langle g \rangle^{n.S.1} @^{n.S.R} \langle x \rangle^{n.S.2})$$

$$t_{F^n} \stackrel{def}{=} (\langle y \rangle^{n.F.2} @^{n.F.M} \langle u \rangle^{n.F.L}) @^{n.F.3} \langle v \rangle^{n.F.R}$$

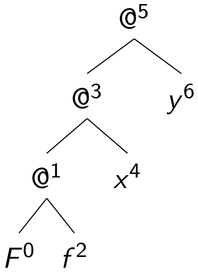
0CFA for SF-calculus — Analysis Rules

$$
\begin{array}{llcll}
\text{Base Labels} & & n & & \\
\text{Sublabel Names} & & s & ::= & \ldots \mid F.0 \mid F.1 \mid F.2 \mid F.3 \mid F.L \mid F.R \mid F.M \\
\text{Labels} & \mathit{Label} \ni & l & ::= & n \mid n.s \\
\text{Labelled Terms} & & t & ::= & S^n \mid F^n \mid t_1 @^l t_2 \mid \langle x \rangle^l \\
\text{Abstract Values} & \mathit{Abs} \ni & v & ::= & \ldots \mid F_0^n \mid F_1^n \mid F_2^n \mid @^{(l_1,l_2)} \\
\text{Abstract Environment} & & \Gamma & : & \mathit{Label} \to \mathcal{P}(\mathit{Abs}) \\
\text{Abstract Activation} & & \varphi & : & \mathit{Label} \to \mathit{Bool}
\end{array}
$$

$$
\begin{array}{lcl}
\Gamma, \varphi \models F^n & \iff & F_0^n \in \Gamma(n) \\
& & \wedge\, \varphi(n) \Rightarrow (\exists n_0 . S_0^{n_0} \in \Gamma(n.F.0) \vee F_0^{n_0} \in \Gamma(n.F.0)) \Rightarrow \Gamma(n.F.1) \subseteq \Gamma(n.F.3) \\
& & \wedge\, \varphi(n) \Rightarrow (\exists n_0 . S_1^{n_0} \in \Gamma(n.F.0) \vee S_2^{n_0} \in \Gamma(n.F.0) \vee \\
& & \qquad F_1^{n_0} \in \Gamma(n.F.0) \vee F_2^{n_0} \in \Gamma(n.F.0)) \Rightarrow \Gamma, \varphi \models t_{F^n} \wedge \\
& & \qquad \forall @^{l_1,l_2} \in \Gamma(n.F.0) . \Gamma(l_1) \subseteq \Gamma(n.F.L) \wedge \Gamma(l_2) \subseteq \Gamma(n.F.R) \\
\Gamma, \varphi \models t_1^{l_1} @^{l_3} t_2^{l_2} & \iff & \Gamma, \varphi \models t_1 \wedge \Gamma, \varphi \models t_2 \\
& & \ldots \\
& & \wedge\, \exists @^{l_4,l_5} \in \Gamma(l_3) . \Gamma(l_1) \subseteq \Gamma(l_4) \wedge \Gamma(l_2) \subseteq \Gamma(l_5) \\
& & \wedge\, \forall F_0^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.0) \wedge F_1^n \in \Gamma(l_3) \\
& & \wedge\, \forall F_1^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.1) \wedge F_2^n \in \Gamma(l_3) \\
& & \wedge\, \forall F_2^n \in \Gamma(l_1) . \Gamma(l_2) \subseteq \Gamma(n.F.2) \wedge \Gamma(n.F.3) \subseteq \Gamma(l_3) \wedge \varphi(n) \\
\Gamma, \varphi \models \langle x \rangle^l & \iff & \mathit{true}
\end{array}
$$

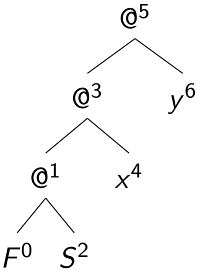
$$
t_{F^n} \overset{def}{=} (\langle y \rangle^{n.F.2} @^{n.F.M} \langle u \rangle^{n.F.L}) @^{n.F.3} \langle v \rangle^{n.F.R}
$$

0CFA for SF-calculus — Analysis Rules



$$
\begin{array}{llll}
\text{At } F^0\text{:} & F^0_0 \in \Gamma(0) & & \\
& \Downarrow & & \\
\text{At } @^1\text{:} & F^0_1 \in \Gamma(1) & \Gamma(2) \subseteq \Gamma(0.F.0) & \\
& \Downarrow & & \\
\text{At } @^3\text{:} & F^0_2 \in \Gamma(3) & \Gamma(4) \subseteq \Gamma(0.F.1) & \\
& \Downarrow & & \\
\text{At } @^5\text{:} & \varphi(0) & \Gamma(6) \subseteq \Gamma(0.F.2) & \Gamma(0.F.3) \subseteq \Gamma(5) \\
& \Downarrow & &
\end{array}
$$

0CFA for SF-calculus — Analysis Rules

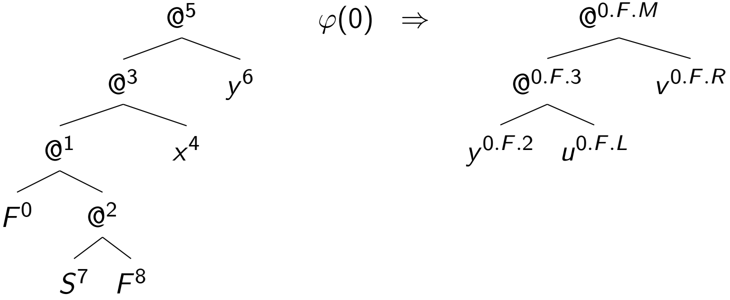


Suppose $f = S$:

$$S_0^2 \in \Gamma(2)$$
$$\Downarrow$$
$$S_0^2 \in \Gamma(0.F.0)$$
$$\Downarrow$$
$$\Gamma(0.F.1) \subseteq \Gamma(0.F.3)$$

$$So: \quad \Gamma(4) \subseteq \Gamma(0.F.1) \subseteq \Gamma(0.F.3) \subseteq \Gamma(5)$$

0CFA for SF-calculus — Analysis Rules



Suppose $f = u\ v = S\ F$:

$$
\begin{array}{ccccc}
\varphi(0) & S_1^7 \in \Gamma(2) & \text{At } @^2\text{:} & @^{7,8} \in \Gamma(2) \\
\Downarrow & \Downarrow & & \Downarrow \\
 & S_1^7 \in \Gamma(0.F.0) & & @^{7,8} \in \Gamma(0.F.0) \\
 & \Downarrow & & \Downarrow \\
 & \Gamma(7) \subseteq \Gamma(0.F.L) & & \Gamma(8) \subseteq \Gamma(0.F.R)
\end{array}
$$

0CFA for SF-calculus — Properties

Relationship with 0CFA for SK-calculus:

- Encoding an SK-calculus program in SF-calculus with K as $F\ F$, the two analyses give the same results.
- 0CFA for SF-calculus retains polynomial time-complexity.

Sources of imprecision:

- As with 0CFA for λ-calculus, the analysis conflates multiple uses of the same argument.
- If our analysis of the 1st argument of F is imprecise, we may activate constraints for both rules, when really only one case applies.
- If an F can use both rules, we apply both sets of constraints to all arguments, not just those that activate the rules.
- If two distinct applications are factorised by the same F, we lose co-ordination between them.
- When we factorise abstractly, we have no way of discarding unfactorable forms.

Conclusion

Contributions:

- A formulation of 0CFA for SK-calculus.
- A formulation of 0CFA for SF-calculus.
  - The first static analysis for SF-calculus.
- Correctness proofs and preliminary evaluation for the above.

Future work:

- A translation from a higher-level language into SF-calculus.
- Address sources of imprecision in analysis.
- Try extending other CFA-style analyses (k-CFA, CFA2, ...) to SF-calculus.

Conclusion

Contributions:

- A formulation of 0CFA for SK-calculus.
- A formulation of 0CFA for SF-calculus.
  - The first static analysis for SF-calculus.
- Correctness proofs and preliminary evaluation for the above.

Future work:

- A translation from a higher-level language into SF-calculus.
- Address sources of imprecision in analysis.
- Try extending other CFA-style analyses (k-CFA, CFA2, ...) to SF-calculus.

Thanks for listening. Questions are welcome.

References

Thomas Given-Wilson and Barry Jay.
A combinatory account of internal structure.
J. Symb. Log., 76(3):807–826, 2011.

J. Roger Hindley and Jonathan P. Seldin.
Lambda-Calculus and Combinators: An Introduction.
Cambridge University Press, New York, NY, USA, 2 edition, 2008.

C. Barry Jay and Jens Palsberg.
Typed self-interpretation by pattern matching.
In Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy, editors, *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming, ICFP 2011, Tokyo, Japan, September 19-21, 2011*, pages 247–258. ACM, 2011.

Jan Midtgaard.
Control-flow analysis of functional programs.
ACM Comput. Surv., 44(3):10, 2012.