CS 520

Theory of Programming Language

06/02 – 06/09, 2021

$\boxed{\text{Continuations in a Functional Language} \cdots \text{Chap12}}$

1. Motivation / Overview.

① Continuations. …. CPS transformation.
Defunctionalisations

> Two important concepts / program transformations used by compilers for functional lang.

(2) Continuations. What are they? Why do we care about them?

⋮

represent the rest of computation.

continuation for $3*4$.

(1) $(9 * (8 + (3*4)))$ .... $(9 * (8 + [\ ]))$

12

$\kappa \in [V \to_c A]$ .... $[V \to_c V_*]$

$A = V_* = (V + \{\text{error}, \text{typeerror}\})_\perp$.

$\kappa = (\lambda v \in V_{int}.\ \iota_{norm}((v+8)+9))$ ....

(2) – Programming style. .... Continuation-passing-style programming.

(i) $cps\text{-}+(x, y, k) = k(x+y)$, $cps\text{-}*(x, y, k) = k(x*y)$.

$cps\text{-}*(3, 4, \lambda n.\ cps\text{-}+(8, n, \lambda n.n))$.

– Control operator. ..... generalized goto, callcc, throw

backtracking, coroutine, generalized iterator.

– Denotational semantics. / mathematics.

V, $V^* = [V \to_L \mathbb{R}]$. .... linear conti.

– One compilation step ..... Program ---> Program in cont. passing style.

(1) no need to use stack.

(2) no longer depends on whether we use $\Rightarrow_{normal}$ or $\Rightarrow_{eager}$.

$E \to_c \check{V}_{cont} \to_c V_*$

2. Continuation-based Denotational Semantics.

① $[\![-]\!] \in [\langle exp \rangle \to E \to_c V_*]$, $\quad E = [\langle var \rangle \to V]$,

space for final answers.

$V_{cont} \stackrel{def}{=} [V \to_c V_*]$

$V_* = (V + \{error, typeerror\})_\perp$

$V \underset{\psi}{\overset{\phi}{\rightleftarrows}} V_{int} + V_{bool} + V_{fun} + V_{tuple} + V_{alt}$.

$V_{int} = \mathbb{Z}$, $V_{bool} = \mathbb{B}$, $V_{fun} = [V \to_c V_*]$, $V_{tuple} = \bigcup_{n \geq 0} V^n$, $V_{alt} = \mathbb{N} \times V$

$\checkmark\ V \to_c V_{cont} \to_c V_*$

(a) Add continuations as a new par. to $[\![-]\!]$, and add continuations to functions.
as a new para.

$\in V_{cont}$

$[\![e]\!]_c \, \eta \, \kappa \in V_*$.

$\Gamma \quad \iota_{int} : V_{int} \to V \quad \iota_{fun} : V_{fun} \to V \quad \ldots \quad \rfloor$.

$[\![3]\!]_c \eta \kappa = \kappa(\iota_{int}(3))$

$[\![e_0\, e_1]\!]_c \eta \kappa = [\![e_0]\!]_c \eta \left(\lambda f \in V_{fun}. \left([\![e_1]\!]_c \eta \left(\lambda v \in V. \; f\, v\, \kappa\right)\right)\right)_{fun}$

$\Gamma \quad g : V_\theta \to_c V_*$

$g_\theta : V \to_c V_* \qquad g_\theta(v) = \begin{cases} g(b) & \text{if } v = \iota_\theta(b) \\ \underbrace{\langle 1, \text{typeerror}\rangle}_{\text{tyerr}} & \text{otherwise} \end{cases} \quad \rfloor$

$[\![x]\!]_c \eta \kappa = \kappa(\eta(x))$ ✓

$[\![\lambda x. e]\!]_c \eta \kappa = \kappa(\iota_{fun}(\lambda v \in V. \lambda \kappa' \in V_{cont}. [\![e]\!]_c [\eta | x : v] \kappa'))$

$[\![e_1 + e_2]\!]_c \eta \kappa = [\![e_1]\!]_c \eta \, (\lambda n_1 \in V_{int}. ([\![e_2]\!]_c \eta \, (\lambda n_2 \in V_{int}. \kappa(\iota_{int}(n_1 + n_2)))_{int}))_{int}$.

$[\![\langle e_1, e_2 \rangle]\!]_c \eta \kappa = [\![e_1]\!]_c \eta \, (\lambda v_1 \in V. ([\![e_2]\!]_c \eta \, (\lambda v_2 \in V. \kappa(\iota_{tuple}\langle v_1, v_2\rangle))))$.

$[\![e.1]\!]_c \eta \kappa = [\![e]\!]_c \eta \, (\lambda v \in V_{tuple}.$ if $1 \in dom(v)$ then $\kappa(v_1)$ else $\langle 1, \text{typeerror}\rangle)_{tuple}$

③ Relationship bt conti. semantics $[\![-]\!]_c$ and direct semantics $[\![-]\!]$.

$$[\![e]\!]_c \eta \kappa = \kappa_* ([\![e]\!] \eta)$$

④ CPS-transform.

cps : <exp> × <funcfm> → <exp>.

$cps(3, z) = z(3)$

$cps(x, z) = z(x)$

$cps(e_0 e_1, z) = cps(e_0, \lambda f.\ cps(e_1, \lambda x.\ f\ x)\ z))$

$cps(\lambda x.e, z) = z(\lambda x.\lambda f.\ cps(e, f))$

☆ $[\![e]\!]_c \eta \kappa = [\![cps(e, v_{cont})]\!][\eta \mid v_{cont} : \iota_{fun}(\kappa)]$

✓ $cps(\langle e_1, e_2 \rangle, z) = cps(e_1, \lambda v_1.\ cps(e_2, \lambda v_2.\ z(\langle v_1, v_2 \rangle)))$

$cps(e.1, z) = cps(e, \lambda v.\ z(v.1))$

$$cps(\lambda x.\ 3+x, z) = z(\lambda x.\lambda f.\ cps(3+x, f))$$
$$= z(\lambda x.\lambda f.\ cps(3, \lambda x_1.\ cps(x, \lambda x_2.\ f(x_1+x_2))))$$
$$((\lambda x_1.\ cps(x, \lambda x_2.\ f(x_1+x_2)))\ 3)$$
$$((\lambda x_2.\ f(x_1+x_2))\ x).$$