CS 520

Theory of Programming Language

04/07 – 04/14, 2021

1. Overview.

① Continuation. — (a) PL
goto, exception, callcc, coroutine.

(b) Semantics.
continuation as a mathematical tool,
weakest precondition.

(c) Logic

classical logic $(\neg\neg p \to p)$ → true in classical logic.
↳ not necessarily true. in intuitionistic logic.

(d) math.

Dual space L, $[L \to \mathbb{R}]$ $= L^*$.

② Our plan : To study continuation semantics. using the PL with fail, input/output

Imp.

make it clear that each operation in our lang. does 2 things.

1) state. output.

2) change control.

2. Continuation - What is it? , How to use it in Semantics?

(1) element $\kappa \in [\Sigma \to \Omega]$. It represents the rest of the computation.

Ans.

$\underline{c}$ $\quad \kappa$ ..... $x := e$, κ , σ $\dashrightarrow$ $\kappa(\sigma[x : [\![e]\!]\sigma])$

fail , κ , σ $\dashrightarrow$ σ

$$[\![\mathit{fail}]\!]\sigma = \iota_{abort}(\sigma)$$

$$[\![x := e]\!]\sigma = \iota_{term}(\sigma)$$

$$[\![c_1 ; c_2]\!]\sigma = [\![c_2]\!]_* ([\![c_1]\!]\sigma)$$

② Continuation Semantics

continuation κ — input state σ

$$[\![-]\!]^{cont} : \langle comm\rangle \to [[\Sigma \to_c \Omega] \to_c [\Sigma \to_c \Omega]]$$

ultimate final answer — ultimate final answer.

$$[\![c]\!]^{cont} \kappa\, \sigma = ??$$

... Guidance $[\![-]\!] : \langle comm\rangle \to [\Sigma \to_c \Omega]$.

$$[\![skip]\!]^{cont} \kappa\, \sigma = \kappa(\sigma)$$

$$[\![x := e]\!]^{cont} \kappa\, \sigma = \kappa\left([\sigma \mid x : [\![e]\!]\sigma]\right)$$

$$[\![c]\!]^{cont}\,\kappa\,\sigma = \kappa_*([\![c]\!]\,\sigma)$$

$$[\![c_1; c_2]\!]^{cont}\,\kappa\,\sigma = [\![c_1]\!]^{cont}(\lambda\sigma'.\,[\![c_2]\!]^{cont}\,\kappa\,\sigma')\,\sigma = [\![c_1]\!]^{cont}([\![c_2]\!]^{cont}\,\kappa)\,\sigma.$$

$$\underline{\lambda\sigma'.\,[\![c_2]\!]^{cont}\,\kappa\,\sigma'} = ([\![c_2]\!]^{cont}\,\kappa)$$

$$[\![\text{if } b \text{ then } c_1 \text{ else } c_2]\!]^{cont}\,\kappa\,\sigma = \text{if } [\![b]\!]\sigma = tt \text{ then } [\![c_1]\!]^{cont}\,\kappa\,\sigma \text{ else } [\![c_2]\!]^{cont}\,\kappa\,\sigma.$$

$$[\![\text{while } b \text{ do } c]\!]^{cont}\,\underline{\underline{\kappa}}\,\sigma = Y_{\underset{cont.}{\underbrace{[\Sigma\to\Omega]}}}(F)\,\sigma = \underbrace{\left(\bigsqcup_{n=0}^{\infty} F^n(\bot)\right)}(\sigma).$$

$$\left(\begin{array}{l} F : ([\Sigma \to_c \Omega]) \to_c [\Sigma \to_c \Omega] \\ F(\underline{\kappa'})(\sigma') = \text{if } [\![b]\!]\sigma' = tt \text{ then } [\![c]\!]^{cont}\,\underline{\kappa'}\,\sigma' \text{ else } \underline{\underline{\kappa}}(\sigma') \end{array}\right)$$

Recall (ι_{term}, ι_{abort}, ι_{in}, ι_{out}, $\iota_\bot$)

$$[\![?x]\!]^{cont}\,\kappa\,\sigma = \iota_{in}(\lambda n \in \mathbb{Z}.\ \kappa([\sigma \mid x : n]))$$

$$[\![!e]\!]^{cont}\,\kappa\,\sigma = \iota_{out}([\![e]\!]\sigma,\ \kappa(\sigma))$$

$$[\![\text{newvar } x := e \text{ in } c]\!]^{cont}\,\kappa\,\sigma$$
$$= [\![c]\!]^{cont}(\lambda\sigma'.\ \kappa[\sigma' \mid x : \sigma(x)])[\sigma \mid x : [\![e]\!]\sigma]$$

$\underline{\underline{ex}}$. 1) $\underline{\text{fill in } \square\text{'s !}}$ 2) How to prove ..?

Assume that our lang. doesn't contain while.

Prove by structural induction.

- skip

$$[\![\text{skip}]\!]^{cont}\ k\ \sigma = k(\sigma).$$

?= ... ok.

$$k_*([\![\text{skip}]\!]\ \sigma) = k_*(\eta_{term}(\sigma)) = k(\sigma).$$

- $c_1 ; c_2$.

$$[\![c_1 ; c_2]\!]^{cont}\ k\ \sigma = [\![c_1]\!]^{cont}\ (\lambda \sigma'.\ [\![c_2]\!]^{cont}\ k\ \sigma')\ \sigma.$$

?=

$$k_*([\![c_1 ; c_2]\!]\ \sigma) = k_*([\![c_2]\!]_*([\![c_1]\!]\ \sigma))$$

$$= (k_* \circ [\![c_2]\!])_*([\![c_1]\!]\ \sigma)$$

I.H. $= [\![c_1]\!]^{cont}\ (k_* \circ [\![c_2]\!])\ \sigma.$

$([\![c_2]\!]^{cont}\ k)$

By I.H on c_2.

$$k_*([\![c_2]\!]\ \sigma) = [\![c_2]\!]^{cont}\ k\ \sigma.$$

$$\| $$

$$(k_* \circ [\![c_2]\!])(\sigma).$$

$$\therefore\ k_* \circ [\![c_2]\!] = [\![c_2]\!]^{cont}\ k.$$

$$= [\![c_1]\!]^{cont}\ ([\![c_2]\!]^{cont}\ k)\ \sigma.$$

=

- $?x$ $\qquad [\![?x]\!]^{cont}\, \kappa\ b = \bar{\mu}_{in}(\lambda n.\ \kappa([b \mid x:n]))$. $=$

$$\kappa_*([\![?x]\!]\, b) = \kappa_*(\bar{\mu}_{in}(\lambda n.\ \mu_{term}([b \mid x:n])))$$

$$= \bar{\mu}_{in}(\lambda n.\ \kappa_*(\mu_{term}(\ \text{''}\ )))$$

$$= \bar{\mu}_{in}(\lambda n.\ \kappa([b \mid x:n]))$$

- $!e$, if ....

3. How to handle fail in cont. semantics? in the presence of newvar.

Problem:

(1) $[\![\mathrm{fail}]\!]^{cont}\,\kappa\ \sigma = \iota_{abort}(\sigma)$. .... candidate.

$\downarrow$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \iota_{abort}([\sigma \mid x:1])$

Has an issue wrt. newvar. //

$[\![\mathrm{newvar}\ x:=1\ \mathrm{in}\ \mathrm{fail}]\!]\,\kappa\ \sigma. = \iota_{abort}(\sigma)$. ... what we expect.

$\|$

$[\![\mathrm{newvar}\ y:=1\ \mathrm{in}\ \mathrm{fail}]\!]\,\kappa\ \sigma.$ \\ $\iota_{abort}([\sigma \mid y:1])$.

(2) Solution: Two continuations as parameters to the semantics.

$$[\![\ ]\!]_2^{cont} : \langle comm \rangle \to [\ \underbrace{[\Sigma \to_c \Omega]}_{\text{usual conti.}} \to_c \underbrace{[\Sigma \to_c \Omega]}_{\text{cont. for fail}} \to_c \underbrace{[\Sigma}_{\text{input state}} \to_c \Omega]\ ]$$

$$[\![c]\!]_2^{cont}\ \underline{\kappa_t}\ \kappa_f\ \sigma = \cdots$$

$$[\![skip]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma = \kappa_t(\sigma)$$

$$[\![v := e]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma = \kappa_t([\sigma \mid x : [\![e]\!]\sigma])$$

$$[\![c_1 ; c_2]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma = [\![c_1]\!]_2^{cont}\ (\lambda\sigma'.\ [\![c_2]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma')\ \kappa_f\ \sigma.$$

$$[\![\text{if } b \text{ then } c_1\ c_2]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma = \text{if } [\![b]\!]\sigma = tt \text{ then } [\![c_1]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma.$$
$$\text{else } [\![c_2]\!]_2^{cont}\ \kappa_t\ \kappa_f\ \sigma.$$

$$[\![\text{fail}]\!]_2.\ \kappa_t\ \underset{\sim}{\kappa_f}\ \sigma = \kappa_f.(\sigma).$$

$$[\![\text{while } b \text{ do } c]\!]_2^{cont}\ \underline{\underline{\kappa_t}}\ \underline{\kappa_f}\ \sigma = Y_{[\Sigma \Rightarrow \Omega]}(F)\ (\sigma).$$
$$F(\kappa')(\sigma') = \text{if } [\![b]\!]\sigma' = tt \text{ then } [\![c]\!]_2^{cont}.\ \kappa'\ \underline{\kappa_f}\ \sigma'$$
$$\text{else } \kappa_t(\sigma').$$

$$[\![\text{newvar } x := e \text{ in } c]\!]_2^{cont}\ \underline{\kappa_t}\ \boxed{\underset{\sim}{\kappa_f}}\ \sigma = .\ [\![c_2]\!].\ (\lambda\sigma'.\ \kappa_t([\sigma' \mid x : \sigma(x)]))$$
$$(\lambda\sigma'.\ \kappa_f([\sigma' \mid x : \sigma(x)]))$$
$$([\sigma \mid x : [\![e]\!]\sigma]).$$

exercise: 1) State the formal relationship between $[\![C]\!]_2^{\text{Cont}}$.
and $[\![C]\!]$.

2) Show the relationship holds for the restricted lang. w/o loops.

3). Think about how to handle loops.