1 Why Y?

Can we "solve for" fact in this equation?

$$fact = \lambda n.\ \text{if } n=0 \text{ then } 1 \text{ else } n \times (fact\ (n-1))$$

Let $fact' = \lambda f.\ \lambda n.\ \text{if } n=0 \text{ then } 1 \text{ else } n \times (f\ (n-1))$

[2] We want a FIX such that

$$FIX\ fact' = fact'\ (FIX\ fact')$$

so that

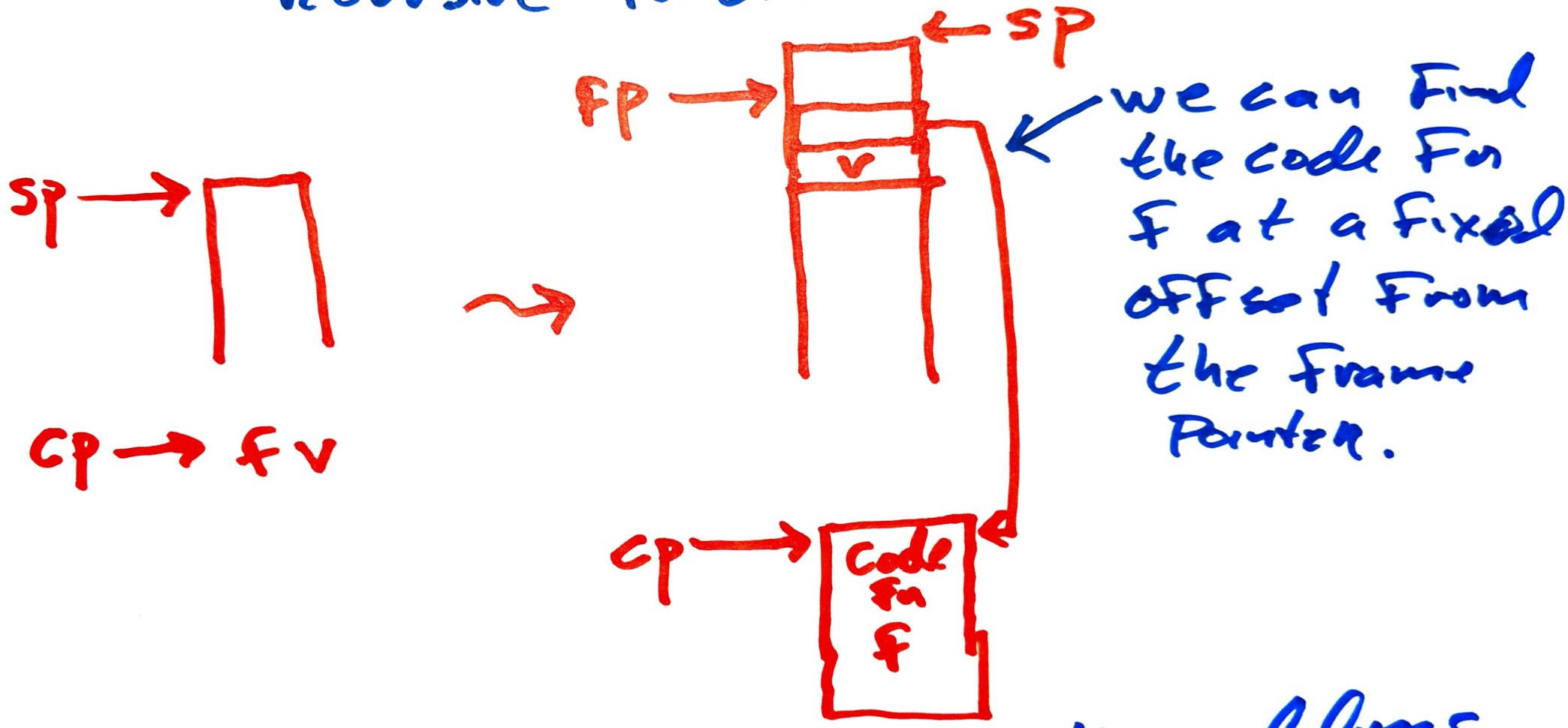
$$fact \triangleq FIX\ (fact')$$

why? $fact\ n = FIX(fact')\ n$

$= fact'\ (FIX\ fact')\ n$

$= IF\ n=0$
$\quad THEN\ 1$
$\quad ELSE\ n \times ((\underbrace{FIX\ fact}_{fact})\ (n-1))$

3 Think about compilation of a recursive function:



we can find the code for f at a fixed offset from the frame pointer.

This looks like passing the address of f to itself!

4 The "hat trick"

$$\hat{fact} = \lambda f.\lambda n.\ \text{if } n=0 \text{ then } 1 \text{ else } n \times (f\ f\ (n-1))$$

$\equiv$

Let $fact = \hat{fact}\ \hat{fact}$

So $fact\ 3 = \hat{fact}\ \hat{fact}\ 3$

$= \text{IF } 3=0 \text{ then } 1$
$\text{else } 3 \times ((\hat{fact}\ \hat{fact})\ 2)$

$= 3 \times (fact\ 2)$

$= \ldots$

5 Can we

$$\text{fact}' \overset{\text{automate?}}{\rightsquigarrow} \widehat{\text{fact}}$$

We want a G such that

$$G(\text{fact}') = \widehat{\text{fact}}$$

Let $G = \lambda f.\lambda g.\, f(gg)$

then $G\,\text{fact}' = \lambda g.\lambda n.\ \text{if } n = 0 \text{ then } 1 \text{ else } n \times (g\,g\,(n-1))$

$= \widehat{\text{fact}}$

6 All together now!

$$
\begin{aligned}
& fact \\
= \; & \widehat{fact}\ \widehat{fact} \\
= \; & (G\,fact')(G\,fact') \\
= \; & (\lambda f.(G\,f)(G\,f))\,fact' \\
= \; & (\lambda f.(\lambda g.f(gg))(\lambda g.f(gg)))\,fact' \\
= \; & Y(fact')
\end{aligned}
$$