Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

# Contents

Assignment Project Exam Help

- Introduction
- Recursive functi
- Recursive functio
  - Binding for function name
  - Fixed point
  - Fixed point operato
  - self-application

https://eduassistpro.github.io/
Add WeChat edu_assist_pro

# Introduction

Assignment Project Exam Help

Recursive function $f$

$\downarrow$

Intermediate function

https://eduassistpro.github.io/

$\downarrow \quad \longleftarrow \quad$ F

Definition of $f$ in terms of Y

Add WeChat edu_assist_pro

$\downarrow \quad \longleftarrow \quad$ How to define Y in the lambda calculus

Final Lambda-calculus definition of $f$

# Recursion and the lambda calculus

Assignment Project Exam Help

- Consider :

$$
\begin{aligned}
f\ x &= 3, \text{ if } (x = 0) \\
&= 11, \text{ otherwise}
\end{aligned}
$$

https://eduassistpro.github.io/

- In the lambda calculus this is :

Add WeChat edu_assist_pro

$$\lambda\ x.(\text{if } (x = 0)\ 3\ 11)$$

# Recursion and the lambda calculus

Assignment Project Exam Help

- However, now consider :

$$
\begin{aligned}
\text{f x} &= 3, \text{ if } (x = 0) \\
&= 1 + f(x - 1), \text{ otherwise}
\end{aligned}
$$

https://eduassistpro.github.io/

- What's wrong with this ? :

Add WeChat edu_assist_pro

$$\lambda\ x.(\text{if } (x = 0)\ 3\ (1 + (f(x - 1))))$$

# Recursion and the lambda calculus

- Consider it again in the context of the whole program :

$$\begin{aligned} \text{f x} &= 3, \text{ if } (x = 0) \\ &= 1 + \text{f}(x - 1) \\ \text{main} &= \text{f } 7 \end{aligned}$$

- This would translate to the following, which still has a problem :

$$\lambda f.(f \;\; 7) \; (\lambda \, x.(\text{if } (x = 0) \; 3 \; (1 + (f(x \text{ - } 1)))) \;)$$

$$\rightarrow^{\beta} \quad \lambda \, x.(\text{if } (x = 0) \; 3 \; (1 + (f(x \text{ - } 1)))) \; 7$$

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

## Recursion and the lambda calculus

Assignment Project Exam Help

- SO how can we represent a recursive function in the lambda calculus ?

1. First define a new NO
   "f" but which takes "f

https://eduassistpro.github.io/

h f x = 3, if (x = 0)
= 1 + f(x - 1), otherwise

Add WeChat edu_assist_pro

1. NB here we are using a *curried* style of function definition.

# Recursion and the lambda calculus

Assignment Project Exam Help

1. Now the following lambda expression for "h" is fine, because "f" is bound :

   $\lambda$ f.($\lambda$ x.(if (x=

https://eduassistpro.github.io/

   BUT "h" is not "f", so we haven't solved the problem yet !

Add WeChat edu_assist_pro

2. However, notice that the partial application (h f) gives the same resul ≡ f (this is an identity, not a definition)

# Recursion and the lambda calculus

Assignment Project Exam Help

- A "fixed point" (or "fixpoint") of any function $g$ is a value $x$ from the input domain of $g$ such that $(g\ x) \equiv x$
- Example 1 :
  - *id* is called the "
  - Every value in t

https://eduassistpro.github.io/

- Example 2 :

$$three\ x = 3,\ if\ ($$
$$= (x + 1$$

Add WeChat edu_assist_pro

  - The input value 3 is the only fixed point of the function *three*
- Note that (from the previous slide) the function $f$ is a fixpoint of the function $h$ because $(h\ f) \equiv f$

# Recursion and the lambda calculus

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- There is a special operator (called the "fixpoint operator") that we can incorporate into the $\lambda$-calculus, which will return the fixed-point of any function.
- The fixpoint opera
- It takes a function as it $g$
  - Thus, by defin
  - If the identifie i.e. the definition with the least amount of arbitrary additional information) version of th
- So now (Y h) gives the least fixpoint of h, which we know is f (because ) $\equiv f$)

  f = Y h

2. We alredy know that in the $\lambda$ calculus we can easily pass functions as arguments, and return them as results.

3. We skate over some interesting problems : what if $g$ doesn't have a fixpoint ? can $g$ have more than one "least" fixpoint ? This is outside the scope of this module, but further explantions are found in Stoy's excellent book *Denotational Semantics : The Scott-Strachey Approach to Programming Language Theory* by J.E.Stoy, 1979.

# Recursion and the lambda calculus

- The reduction rule for operator $Y$ is trivial : $Y\ g \ \rightarrow\ g\ (Y\ g)$
- Now, for example (because $f$ is the least fixpoint of $h$), a Normal Order reduction[4] of $f$ 1 gives :

| | | |
|---|---|---|
| f 1 | $= (\text{Y h})\ 1$ | because f = Y h |
| | $= (\text{h}\quad (\text{Y h}))\ 1$ | because Y g $\rightarrow$ g (Y g) |
| | $= ((\lambda\ \text{f}.$ | |
| | $= (\lambda\ \text{x}.($ | $\beta$ reduction |
| | $= (\text{if}\ (1 = 0)\ 3\ (1$ | after another $\beta$ reduction |
| | $= (\text{if}\ \textit{false}\quad 3\ (1 + ((\text{Y h})(1 - 1))))$ | after $\delta$ reduction of = |
| | $= 1 + ((\text{Y h})\ (1 - 1))$ | after reduction of *if* loop ! |
| | $= 1 + ((\text{h}\ (\text{Y h}))\ (1 - 1))$ | because Y g $\rightarrow$ g (Y g) |
| | $= 1 + (((\lambda\ \text{f}.(\lambda\ \text{x}.(\text{if}\ (\text{x}{=}0)\ 3\ (1 + (\text{f}\ (\text{x-}1)))))) (\text{Y h}))\ (1\text{-}1$ | f |
| | $= 1 + ((\lambda\ \text{x}.(\text{if}\ (\text{x} = 0)\ 3\ (1 + ((\text{Y h})(\text{x - }1))))) \ (1 - 1))$ | after one $\beta$ reduction |
| | $= 1 + ((\text{if}\ ((1 - 1) = 0)\ 3\ (1 + ((\text{Y h})((1 - 1) - 1)))))$ | after another $\beta$ reduction |
| | $= 1 + ((\text{if}\ (0 = 0)\quad 3\ (1 + ((\text{Y h})((1 - 1) - 1)))))$ | after $\delta$ reduction of = |
| | $= 1 + 3$ | after $\delta$ reduction of *if* |

4. Other reduction orders may not terminate.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

# Recursion and the lambda calculus

Assignment Project Exam Help

- Now we have a lambd

https://eduassistpro.github.io/

Y ($\lambda$ f.($\lambda$ x.(if (x = 0) 3 (1 + (f (x - 1))))))

Add WeChat edu_assist_pro

- But haven't we really just shifted the problem ? — how do we define bda calculus ?

# Recursion and the lambda calculus

Assignment Project Exam Help

The real magic : self-appli

https://eduassistpro.github.io/

$$Y \equiv \lambda q.(\ (\lambda x.(q\ (x\ x)))\ (\lambda x.(q\ (x\ x)))\ )$$

Add WeChat edu_assist_pro

## Recursion and the lambda calculus

Assignment Project Exam Help

Example :

$Y\ h = \lambda q.(\ (\lambda x.($ https://eduassistpro.github.io/

$= (\lambda x.(h\ (x\ x))$ 2

$= h\ ((\lambda x.(h\ (x\ x)))\ (\lambda x.(h\ (x\ x))))$ *line*$_3$

$= h\ (Y\ h)$ because $= Y\ h$

Add WeChat edu_assist_pro

# Recursion and the lambda calculus

Assignment Project Exam Help

Deriving a $\lambda$-calculu

https://eduassistpro.github.io/

$$
\begin{aligned}
f &= (Y\ h)\\
&= (Y\ (\lambda\ f.(\lambda\ x.(\text{if}\ (x{=}0)\ 3\ (1{+}(f(x\text{-}1)))))))\\
&= ((\lambda q.((\lambda x.(q\ (x\ x)))\ (\lambda x.(q\ (x\ x)))))\ (\lambda\ f.(\lambda\ x.(\text{if}\ (x
\end{aligned}
$$

Add WeChat edu_assist_pro

# Summary

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Recursive functio
- Recursive functio
  - Binding for function name
  - Fixed point
  - Fixed point opera o
  - Self-application

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro