Delta rules

The INFDEV@HR Team

Hogeschool Rotterdam
Rotterdam, Netherlands

Introduction

Introduction

Lecture topics

- Make it pretty: delta rules
- Booleans, boolean logic operators, if-then-else
- Naturals, arithmetic operators, comparison operators

Encoding boolean logic

Encoding boolean logic

Introduction

- We can decide that some specific lambda terms have special meanings
- For example, we could decide that a given lambda term means `TRUE`, another `FALSE`, etc.
- The important thing is that we choose terms that behave as we wish

Encoding boolean logic

As we wish?

- Suppose we define some lambda terms for `TRUE`, `FALSE`, and `AND`
- We expect these terms to reduce[a] following our expectations of boolean logic
- We can use truth tables to encode our expectations

[a] That is, computed according to $\rightarrow_\beta$

Encoding boolean logic

We want to formulate TRUE, FALSE, and AND so that

- $\texttt{TRUE} \wedge \texttt{TRUE} \rightarrow_\beta \texttt{TRUE}$
- $\texttt{TRUE} \wedge \texttt{FALSE} \rightarrow_\beta \texttt{FALSE}$
- $\texttt{FALSE} \wedge \texttt{TRUE} \rightarrow_\beta \texttt{FALSE}$
- $\texttt{FALSE} \wedge \texttt{FALSE} \rightarrow_\beta \texttt{FALSE}$

HOGESCHOOL ROTTERDAM

Defining terms with special meaning

Defining terms with special meaning

Choice terms

- Terms with special meaning essentially make a choice when given parameters
- The choice is expressed by either returning, or applying, the parameters

Defining terms with special meaning

Delta rules

- We wish to use special symbols to these terms with special meaning
- We define a series of delta rules, which are transformation from pretty symbols into lambda terms (and vice-versa)

Delta rules

This means that we will be able to write lambda programs such as 5+3, that will then be translated into the appropriate lambda terms

Booleans

Booleans

Idea

- Boolean operators such as `TRUE` and `FALSE` must be defined so as to identify themselves
- The choice is expressed by returning their identity from a choice of two options

Booleans

TRUE is defined as a selector of the representative for true, that is the first argument[a]

[a] by arbitrary convention

```
(λt f.t)
```

FALSE is defined as a selector of the representative for false, that is the second argument[a]

[a] by arbitrary convention, as long as different from the previous

```
(λt f.f)
```

Booleans


```
((TRUE bit1) bit0)
```

Booleans

```
((TRUE bit1) bit0)
```

```
((TRUE bit1) bit0)
```

Booleans


```
((TRUE bit1) bit0)
```

Booleans


```
(( TRUE  bit1) bit0)
```

```
(( (λt f.t)  bit1) bit0)
```

```
(((λt f.t) bit1) bit0)
```

Booleans


```
(((λt f.t) bit1) bit0)
```

```
( ((λt f.t) bit1) bit0)
```

Booleans

$$\boxed{(\ ((\lambda t\ f.t)\ \texttt{bit1})\ \ \texttt{bit0})}$$

Booleans


```
( ((λt f.t) bit1) bit0)
```

```
((λf. bit1 ) bit0)
```

Booleans


```
((λf.bit1) bit0)
```

Booleans

((λf.bit1) bit0)

((λf.bit1) bit0)

Booleans

$((\lambda f.\texttt{bit1})\ \texttt{bit0})$

Booleans


```
((λf.bit1) bit0)
```

```
bit1
```

Booleans

AND

- The conjunction[a] of two terms is a function that takes as input two booleans and returns a boolean
- Since we just defined booleans to be two-parameter functions, we know that the two input booleans can be applied to each other
- Given two booleans `p` and `q`, their conjunction is `q` if `p` was `true`, or `false` otherwise
-

```
(λp q.((p q) p))
```

[a] AND, or $\wedge$

Booleans

AND

Let us begin to with $\texttt{TRUE} \wedge \texttt{TRUE} \rightarrow_\beta \texttt{TRUE}$

Booleans

$$(\texttt{TRUE} \ \wedge \ \texttt{TRUE})$$

Booleans


```
(TRUE ∧ TRUE)
```

```
((∧ TRUE) TRUE)
```

Booleans


```
((∧ TRUE) TRUE)
```

Booleans


```
((∧ TRUE) TRUE)
```

```
(((λp q.((p q) p)) TRUE) TRUE)
```

Booleans

$$(((\lambda p\ q.((p\ q)\ p))\ \mathtt{TRUE})\ \mathtt{TRUE})$$

Booleans


```
(((λp q.((p q) p)) TRUE) TRUE)
```

```
(((λp q.((p q) p)) TRUE) TRUE)
```

Booleans


```
(((λp q.((p q) p)) TRUE) TRUE)
```

Booleans


```
(((λp q.((p q) p)) TRUE) TRUE)
```

```
(((λp q.((p q) p)) (λt f.t)) TRUE)
```

Booleans

$$\boxed{(((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.t))\ \mathtt{TRUE})}$$

Booleans


```
(((λp q.((p q) p)) (λt f.t)) TRUE)
```

```
( ((λp q.((p q) p)) (λt f.t)) TRUE)
```

Booleans


```
( ((λp q.((p q) p)) (λt f.t)) TRUE)
```

Booleans


```
( ((λp q.((p q) p)) (λt f.t)) TRUE)
```

```
((λq.(( (λt f.t) q) (λt f.t) )) TRUE)
```

Booleans

$$\texttt{((}\lambda\texttt{q.(((}\lambda\texttt{t f.t) q) (}\lambda\texttt{t f.t))) TRUE)}$$

Booleans

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((λq.(((λt f.t) q) (λt f.t))) TRUE)
```

```
((λq.(((λt f.t) q) (λt f.t))) TRUE)
```

$((\lambda \texttt{q}.(((\lambda \texttt{t f}.\texttt{t})\ \texttt{q})\ (\lambda \texttt{t f}.\texttt{t})))\ \boxed{\texttt{TRUE}})$

```
((λq.(((λt f.t) q) (λt f.t))) TRUE )
```

```
((λq.(((λt f.t) q) (λt f.t))) (λt f.t) )
```

Booleans

$$\boxed{((\lambda \mathtt{q}.(((\lambda \mathtt{t\ f.t})\ \mathtt{q})\ (\lambda \mathtt{t\ f.t})))\ (\lambda \mathtt{t\ f.t}))}$$

Booleans


```
((λq.(((λt f.t) q) (λt f.t))) (λt f.t))
```

```
((λq.(((λt f.t) q) (λt f.t))) (λt f.t))
```

Booleans

$((\lambda q.(((\lambda t\ f.t)\ q)\ (\lambda t\ f.t)))\ (\lambda t\ f.t))$

Booleans


```
((λq.(((λt f.t) q) (λt f.t))) (λt f.t))
```

```
(((λt f.t) (λt f.t)) (λt f.t))
```

Booleans

$$\texttt{(((}\lambda\texttt{t f.t) (}\lambda\texttt{t f.t)) (}\lambda\texttt{t f.t))}$$

Booleans


```
(((λt f.t) (λt f.t)) (λt f.t))
```

```
( ((λt f.t) (λt f.t)) (λt f.t))
```

Booleans


```
( ((λt f.t) (λt f.t)) (λt f.t))
```

Booleans


```
( ((λt f.t) (λt f.t)) (λt f.t))
```

```
((λf t f.t) (λt f.t))
```

Booleans


```
((λf t f.t) (λt f.t))
```

Booleans


```
((λf t f.t) (λt f.t))
```

```
((λf t f.t) (λt f.t))
```

Booleans


```
((λf t f.t) (λt f.t))
```

Booleans

```
((λf t f.t) (λt f.t))
```

```
(λt f.t)
```

Booleans


```
(λt  f.t)
```

Booleans

$(\lambda t\ \ f.t)$

$(\lambda t\ \ f.t)$

Booleans

$(\lambda \texttt{t f.t})$

Booleans

$(\lambda t\ f.t)$

TRUE

Booleans

It works, but it is probably only because of black magic.

Booleans

It works, but it is probably only because of black magic.

Or is it? Let's see if we can get lucky again...

Booleans

OR

- The disjunction[a] of two terms is a function that takes as input two booleans and returns a boolean
- Like with conjunction, remember that the two input booleans can be applied to one another
- Given two booleans p and q, their disjunction is `true` if p was `true`, or q otherwise
-
```
(λp q.((p p) q))
```

[a] `OR`, or $\vee$

OR

Let us begin to with $\texttt{TRUE} \vee \texttt{TRUE} \rightarrow_{\beta} \texttt{TRUE}$

Booleans

$$\boxed{(\mathtt{TRUE} \ \vee \ \mathtt{TRUE})}$$

Booleans

$(\mathtt{TRUE}\ \vee\ \mathtt{TRUE})$

$((\boxed{\vee}\ \mathtt{TRUE})\ \mathtt{TRUE})$

Booleans

$$((\boxed{\vee}\ \mathtt{TRUE})\ \mathtt{TRUE})$$

Booleans

$$((\boxed{\vee}\ \mathtt{TRUE})\ \mathtt{TRUE})$$

$$((\boxed{(\lambda p\ q.((p\ p)\ q))}\ \mathtt{TRUE})\ \mathtt{TRUE})$$

Booleans

$$\boxed{(((\lambda \texttt{p q.}((\texttt{p p}) \texttt{ q})) \texttt{ TRUE}) \texttt{ TRUE})}$$

Booleans


```
(((λp q.((p p) q)) TRUE) TRUE)
```

```
(((λp q.((p p) q)) TRUE) TRUE)
```

Booleans


```
(((λp q.((p p) q)) TRUE) TRUE)
```

Booleans

```
(((λp q.((p p) q)) TRUE) TRUE)
```

```
(((λp q.((p p) q)) (λt f.t)) TRUE)
```

Booleans


```
(((λp q.((p p) q)) (λt f.t)) TRUE)
```

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
(((λp q.((p p) q)) (λt f.t)) TRUE)
```

```
( ((λp q.((p p) q)) (λt f.t)) TRUE)
```

Booleans

(((λp q.((p p) q)) (λt f.t)) TRUE)

Booleans


```
( ((λp q.((p p) q)) (λt f.t)) TRUE)
```

```
((λq.(( (λt f.t) (λt f.t) ) q)) TRUE)
```

```
((λq.(((λt f.t) (λt f.t)) q)) TRUE)
```

Booleans

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

$$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \texttt{TRUE})$$

$$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \boxed{\texttt{TRUE}})$$

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \boxed{\texttt{TRUE}})$

Booleans


```
((λq.(((λt f.t) (λt f.t)) q)) TRUE)
```

```
((λq.(((λt f.t) (λt f.t)) q)) (λt f.t))
```

Booleans

$$\boxed{((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ (\lambda t\ f.t))}$$

Booleans

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

$((\lambda q.(((\lambda t\ \ f.t)\ \ (\lambda t\ \ f.t))\ \ q))\ \ (\lambda t\ \ f.t))$

$((\lambda q.(((\lambda t\ \ f.t)\ \ (\lambda t\ \ f.t))\ \ q))\ \ (\lambda t\ \ f.t))$

Booleans

$$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ (\lambda t\ f.t))$$

Booleans

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ (\lambda t\ f.t))$

$(((\lambda t\ f.t)\ (\lambda t\ f.t))\ (\lambda t\ f.t))$

Booleans

```
(((λt f.t) (λt f.t)) (λt f.t))
```

Booleans


```
(((λt f.t) (λt f.t)) (λt f.t))
```

```
( ((λt f.t) (λt f.t)) (λt f.t))
```

Booleans

$$(\boxed{((\lambda t\ f.t)\ (\lambda t\ f.t))}\ (\lambda t\ f.t))$$

Booleans


```
( ((λt f.t) (λt f.t)) (λt f.t))
```

```
((λf t f.t) (λt f.t))
```

Booleans

```
((λf t f.t) (λt f.t))
```

HOGESCHOOL ROTTERDAM

Booleans

$((\lambda f\ \ t\ \ f.t)\ \ (\lambda t\ \ f.t))$

$((\lambda f\ \ t\ \ f.t)\ \ (\lambda t\ \ f.t))$

Booleans


```
((λf t f.t) (λt f.t))
```

Booleans

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((λf t f.t) (λt f.t))
```

```
(λt f.t)
```

Booleans

$$\boxed{(\lambda t\ \ f.t)}$$

Booleans

$(\lambda t\ \ f.t)$

$(\lambda t\ \ f.t)$

Booleans

$(\lambda t\ f.t)$

Booleans

$(\lambda t\ f.t)$

TRUE

Booleans

if-then-else

- The conditional operator `if-then-else` chooses one of two parameters based on the value of the input condition
- Given a boolean `c` and two values `th` and `el`, the result is `th` if `c` was `true`, or `el` otherwise
- Since `c` is a boolean, it already performs this choice!
- $\boxed{(\lambda p\ th\ el.((p\ th)\ el))}$

Booleans

`if-then-else`

Let us try with `if TRUE` $\vee$ `FALSE then A else B` $\rightarrow_\beta$ `A`

Booleans


```
if TRUE then A else B
```

Booleans

```
if TRUE then A else B
```

```
(((if-then-else TRUE) A) B)
```

```
(((if-then-else TRUE) A) B)
```

Booleans


```
((( if-then-else  TRUE) A) B)
```

```
((( (λp th el.((p th) el))  TRUE) A) B)
```

Booleans

$$((((\lambda \texttt{p th el.((p th) el)) TRUE) A) B)}$$

Booleans

$((((\lambda p\ th\ el.((p\ th)\ el))\ TRUE)\ A)\ B)$

$((((\lambda p\ th\ el.((p\ th)\ el))\ \boxed{TRUE})\ A)\ B)$

Booleans


```
((((λp th el.((p th) el)) TRUE) A) B)
```

Booleans


```
((((λp th el.((p th) el)) TRUE) A) B)
```

```
((((λp th el.((p th) el)) (λt f.t)) A) B)
```

Booleans


```
((((λp th el.((p th) el)) (λt f.t)) A) B)
```

Booleans

$$((((\lambda p\ th\ el.((p\ th)\ el))\ (\lambda t\ f.t))\ A)\ B)$$

$$((\boxed{((\lambda p\ th\ el.((p\ th)\ el))\ (\lambda t\ f.t))}\ A)\ B)$$

Booleans

$$((\ ((\lambda p\ th\ el.((p\ th)\ el))\ (\lambda t\ f.t))\ \ A)\ \ B)$$

Booleans

`(( ((λp th el.((p th) el)) (λt f.t)) A) B)`

`(((λth el.(( (λt f.t) th) el)) A) B)`

Booleans


```
(((λth el.(((λt f.t) th) el)) A) B)
```

Booleans

$$(((\lambda \texttt{th el}.(((\lambda \texttt{t f}.\texttt{t})\ \texttt{th})\ \texttt{el}))\ \texttt{A})\ \texttt{B})$$

$$(((\lambda \texttt{th el}.(((\lambda \texttt{t f}.\texttt{t})\ \texttt{th})\ \texttt{el}))\ \texttt{A})\ \texttt{B})$$

Booleans

$$(((\lambda \texttt{th el}.(((\lambda \texttt{t f}.\texttt{t})\ \texttt{th})\ \texttt{el}))\ \texttt{A})\ \ \texttt{B})$$

Booleans


```
( ((λth el.(((λt f.t) th) el)) A)  B)
```

```
((λel.(((λt f.t)  A ) el)) B)
```

Booleans

$$\texttt{((}\lambda\texttt{el.(((}\lambda\texttt{t f.t) A) el)) B)}$$

Booleans

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((λel.(((λt f.t) A) el)) B)
```

```
((λel.(((λt f.t) A) el)) B)
```

Booleans

$((\lambda \mathtt{el}.(((\lambda \mathtt{t\ f.t})\ \mathtt{A})\ \mathtt{el}))\ \mathtt{B})$

Booleans


```
((λel.(((λt f.t) A) el)) B)
```

```
(((λt f.t) A) B)
```

```
(((λt f.t) A) B)
```

Booleans


```
(((λt f.t) A) B)
```

```
( ((λt f.t) A) B)
```

Booleans

$$(((\lambda t\ f.t)\ A)\ \ B)$$

Booleans


```
( ((λt f.t) A)  B)
```

```
((λf. A )  B)
```

Booleans


```
((λf.A) B)
```

Booleans

$((\lambda f.A)\ B)$

$((\lambda f.A)\ B)$

Booleans


```
((λf.A) B)
```

Booleans


```
((λf.A) B)
```

```
A
```

Natural numbers

Natural numbers

Idea

- Natural numbers such as 3 and 0 must be defined so as to identify themselves
- Their identity is determined by how many times they perform an action
- The only action we have available is applying a function to a term

Natural numbers

Idea

- We will use unary numbers
- A number is defined by how many times it applies a function to a given term
- Zero applications are also possible, in this case we default to the given term

Natural numbers

0, 1, etc.

A number is defined as an applicator of a term identifying as successor to another term identifying as zero[a]

[a] first and second arguments by arbitrary convention

Natural numbers

0 will thus look like

```
(λs z.z)
```

1 will look like

```
(λs z.(s z))
```

7 will look like

```
(λs z.(s (s (s (s (s (s (s z))))))))
```

etc.

Natural numbers

Addition

- Adding numbers is a function that takes as input two numbers (say `m` and `n`), and returns a number
- The first number applies its first parameter `m` times to its second parameter
- The second number applies its first parameter `n` times to its second parameter
- We can use the second number as the second parameter to the first, therefore obtaining something that applies `m+n` times
- ```
  (λm n→ (λs z.((m s) ((n s) z))))
  ```
  ```

Natural numbers

Addition

Let us try it out to $2 + 1 \rightarrow_\beta 3$

```
(2 + 1)
```

```
(2  +  1)
```

```
((+  2)  1)
```

Natural numbers


```
((+ 2) 1)
```

Natural numbers


```
(( + 2) 1)
```

```
(( (λm n→ (λs z.((m s) ((n s) z)))) 2) 1)
```

$$\boxed{\texttt{(((}\lambda\texttt{m n}\rightarrow\texttt{ (}\lambda\texttt{s z.((m s) ((n s) z)))) 2) 1)}}$$

Natural numbers

$(((\lambda m\ \ n \rightarrow\ (\lambda s\ \ z.((m\ \ s)\ \ ((n\ \ s)\ \ z))))\ \ 2)\ \ 1)$

$(((\lambda m\ \ n \rightarrow\ (\lambda s\ \ z.((m\ \ s)\ \ ((n\ \ s)\ \ z))))\ \ \boxed{2})\ \ 1)$

Natural numbers


```
(((λm n→ (λs z.((m s) ((n s) z)))) 2) 1)
```

Natural numbers


```
(((λm n→ (λs z.((m s) ((n s) z)))) 2) 1)
```

```
(((λm n→ (λs z.((m s) ((n s) z))))
    (λs z.(s (s z)))) 1)
```

$$(((\lambda m\ n \rightarrow\ (\lambda s\ z.((m\ s)\ ((n\ s)\ z))))\ (\lambda s\ z.(s\ (s\ z))))\ 1)$$

```
(((λm n→ (λs z.((m s) ((n s) z)))) (λs z.(s (s
    z)))) 1)
```

```
( ((λm n→ (λs z.((m s) ((n s) z)))) (λs z.(s (s z))))
    1)
```

Natural numbers


```
( ((λm n→ (λs z.((m s) ((n s) z)))) (λs z.(s (s z))))
    1)
```

```
( ((λm n→ (λs z.((m s) ((n s) z)))) (λs z.(s (s z))))
    1)
```

```
((λn s z.(( (λs z.(s (s z))) s) ((n s) z))) 1)
```

Natural numbers

```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) 1)
```

Natural numbers


```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) 1)
```

```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) 1)
```

Natural numbers


```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) 1)
```

```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) 1)
```

```
((λn s z.(((λs z.(s (s z))) s) ((n s) z)))
    (λs z.(s z)) )
```

Natural numbers

$$((\lambda n\ s\ z.(((\lambda s\ z.(s\ (s\ z)))\ s)\ ((n\ s)\ z)))\ (\lambda s\ z.(s\ z)))$$

Natural numbers


```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) (λs
    z.(s z)))
```

```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) (λs z.(s z)))
```

Natural numbers

$((\lambda \texttt{n s z.}(((\lambda \texttt{s z.(s (s z))) s) ((n s) z))) } (\lambda \texttt{s z.(s z)}))$

Natural numbers


```
((λn s z.(((λs z.(s (s z))) s) ((n s) z))) (λs z.(s z)))
```

```
(λs z.(((λs z.(s (s z))) s) (( (λs z.(s z)) s) z)
   ))
```

Natural numbers

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ s)\ (((\lambda s\ z.(s\ z))\ s)\ z)))$

Natural numbers

```
(λs z.(((λs z.(s (s z))) s) (((λs z.(s z)) s) z)
   ))
```

```
(λs z.( ((λs z.(s (s z))) s)  (((λs z.(s z)) s) z)))
```

Natural numbers

$$(\lambda s\ z.(\,((\lambda s\ z.(s\ (s\ z)))\ s)\ \ (((\lambda s\ z.(s\ z))\ s)\ z)))$$

Natural numbers

`(λs z.( ((λs z.(s (s z))) s) (((λs z.(s z)) s) z)))`

`(λs z.((λz.(s (s z))) (((λs z.(s z)) s) z)))`

$$\boxed{(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ (((\lambda s\ z.(s\ z))\ s)\ z)))}$$

```
(λs z.((λz.(s (s z))) (((λs z.(s z)) s) z)))
```

```
(λs z.((λz.(s (s z))) (((λs z.(s z)) s) z)))
```

$(\lambda s\ \ z.((\lambda z.(s\ \ (s\ \ z)))\ \ (((\lambda s\ \ z.(s\ \ z))\ \ s)\ \ z)))$

Natural numbers

$$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ (((\lambda s\ z.(s\ z))\ s)\ z)))$$

$$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ z))\ z)))$$

Natural numbers

$$\boxed{(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ z))\ z)))}$$

Natural numbers

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ z))\ z)))$

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ z))\ z)))$

Natural numbers


```
(λs z.((λz.(s (s z))) ((λz.(s z)) z)))
```

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
(λs z.((λz.(s (s z))) ((λz.(s z)) z)))
```

```
(λs z.((λz.(s (s z))) (s z)))
```

Natural numbers

$$(\lambda \texttt{s z.}((\lambda \texttt{z.}(\texttt{s (s z)})) \texttt{ (s z)}))$$

Natural numbers


```
(λs z.((λz.(s (s z))) (s z)))
```

```
(λs z.((λz.(s (s z))) (s z)))
```

Natural numbers

$(\lambda \mathtt{s\ \ z.}\ \mathtt{((}\lambda \mathtt{z.(s\ \ (s\ \ z)))\ \ (s\ \ z))}\ \mathtt{)}$

Natural numbers


```
(λs z.((λz.(s (s z))) (s z)))
```

```
(λs z.(s (s (s z))))
```

Natural numbers


```
(λs z.(s (s (s z))))
```

Natural numbers

```
(λs z.(s (s (s z))))
```

```
(λs z.(s (s (s z))))
```

$(\lambda s\ z.(s\ (s\ (s\ z))))$

Natural numbers

$(\lambda s\ z.(s\ (s\ (s\ z))))$

3

Multiplication

- Multiplying numbers is a function that takes as input two numbers (say `m` and `n`), and returns a number
- The first number applies its first parameter `m` times to its second parameter
- The second number applies its first parameter `n` times to its second parameter
- We can use the second number as the first parameter to the first, therefore obtaining something that applies `n+ m` times, starting from `z`
-
```
(λm n→ (λs z.((m (n s)) z)))
```

Multiplication

Let us try it out to $2 \times 2 \rightarrow_{\beta} 4$

Natural numbers


```
(2  ×  2)
```

Natural numbers

(2×2)

$((\times\ 2)\ 2)$

Natural numbers

$$((\times \quad 2) \quad 2)$$

Natural numbers

$$((\times\ \ 2)\ \ 2)$$

$$((\ (\lambda m\ n \rightarrow\ (\lambda s\ z.((m\ (n\ s))\ z)))\ \ 2)\ \ 2)$$

Natural numbers

$$\texttt{(((}\lambda\texttt{m n}\rightarrow\texttt{ (}\lambda\texttt{s z.((m (n s)) z))) 2) 2)}$$

$$(((\lambda \mathtt{m}\ \ \mathtt{n} \rightarrow\ (\lambda \mathtt{s}\ \ \mathtt{z}.((\mathtt{m}\ \ (\mathtt{n}\ \ \mathtt{s}))\ \ \mathtt{z})))\ \ 2)\ \ 2)$$

$$(((\lambda \mathtt{m}\ \ \mathtt{n} \rightarrow\ (\lambda \mathtt{s}\ \ \mathtt{z}.((\mathtt{m}\ \ (\mathtt{n}\ \ \mathtt{s}))\ \ \mathtt{z})))\ \ \boxed{2})\ \ 2)$$

Natural numbers

$$\texttt{(((}\lambda\texttt{m n}\rightarrow\texttt{ (}\lambda\texttt{s z.((m (n s)) z))) } \boxed{\texttt{2}}\texttt{) 2)}$$

Natural numbers


```
(((λm n→ (λs z.((m (n s)) z))) 2) 2)
```

```
(((λm n→ (λs z.((m (n s)) z))) (λs z.(s (s z)))) 
    2)
```

Natural numbers

$$\boxed{\texttt{(((λm n$\rightarrow$ (λs z.((m (n s)) z))) (λs z.(s (s z)))) 2)}}$$

```
(((λm n→ (λs z.((m (n s)) z))) (λs z.(s (s z))
   )) 2)
```

```
( (((λm n→ (λs z.((m (n s)) z))) (λs z.(s (s z)))) 2)
```

Natural numbers

$$(((\lambda m\ n \rightarrow (\lambda s\ z.((m\ (n\ s))\ z)))\ (\lambda s\ z.(s\ (s\ z))))\ 2)$$

Natural numbers


```
( ((λm n→ (λs z.((m (n s)) z))) (λs z.(s (s z)))) 2)
```

```
((λn s z.(( (λs z.(s (s z))) (n s)) z)) 2)
```

Natural numbers

$$\boxed{((\lambda n\ s\ z.(((\lambda s\ z.(s\ (s\ z)))\ (n\ s))\ z))\ 2)}$$

Natural numbers

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((λn s z.(((λs z.(s (s z))) (n s)) z)) 2)
```

```
((λn s z.(((λs z.(s (s z))) (n s)) z)) 2)
```

Natural numbers

$$\texttt{((}\lambda\texttt{n s z.((}(\lambda\texttt{s z.(s (s z))) (n s)) z))}\ \boxed{2}\texttt{)}$$

Natural numbers

$$((\lambda \mathtt{n\ s\ z}.(((\lambda \mathtt{s\ z}.(\mathtt{s\ (s\ z)}))\ (\mathtt{n\ s}))\ \mathtt{z}))\ \boxed{2})$$

$$((\lambda \mathtt{n\ s\ z}.(((\lambda \mathtt{s\ z}.(\mathtt{s\ (s\ z)}))\ (\mathtt{n\ s}))\ \mathtt{z}))\ \boxed{(\lambda \mathtt{s\ z}.(\mathtt{s\ (s\ z)}))})$$

```
((λn s z.(((λs z.(s (s z))) (n s)) z)) (λs z.(s
    (s z))))
```

Natural numbers


```
((λn s z.(((λs z.(s (s z))) (n s)) z)) (λs z.(s
    (s z))))
```

```
((λn s z.(((λs z.(s (s z))) (n s)) z)) (λs z.(s (s z))))
```

Natural numbers

$$((\lambda \mathtt{n\ s\ z}.(((\lambda \mathtt{s\ z}.(\mathtt{s\ (s\ z)}))\ (\mathtt{n\ s}))\ \mathtt{z}))\ (\lambda \mathtt{s\ z}.(\mathtt{s\ (s\ z)})))$$

Natural numbers


```
((λn s z.(((λs z.(s (s z))) (n s)) z)) (λs z.(s (s z))))
```

```
(λs z.(((λs z.(s (s z))) ((λs z.(s (s z))) s)) z)
)
```

Natural numbers

$(\lambda \texttt{s z.}(((\lambda \texttt{s z.}(\texttt{s (s z)})) ((\lambda \texttt{s z.}(\texttt{s (s z)})) \texttt{ s)) z}))$

Natural numbers

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ ((\lambda s\ z.(s\ (s\ z)))\ s))\ z))$

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ \boxed{((\lambda s\ z.(s\ (s\ z)))\ s)})\ z))$

Natural numbers


```
(λs z.(((λs z.(s (s z))) ((λs z.(s (s z))) s)) z))
```

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ ((\lambda s\ z.(s\ (s\ z)))\ s))\ z))$

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ (\lambda z.(s\ (s\ z))))\ z))$

Natural numbers

$$\boxed{(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ (\lambda z.(s\ (s\ z))))\ z))}$$

Natural numbers

Delta rules
The INFDEV@HR Team
Introduction
Encoding boolean logic
Defining terms with special meaning
Booleans
Natural numbers
Conclusion
Appendix

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ (\lambda z.(s\ (s\ z))))\ z))$

$(\lambda s\ z.(((\lambda s\ z.(s\ (s\ z)))\ (\lambda z.(s\ (s\ z))))\ z))$

$$(\lambda \texttt{s z.(} \boxed{\texttt{((}\lambda\texttt{s z.(s (s z))) (}\lambda\texttt{z.(s (s z))))}} \texttt{ z))}$$

Natural numbers

$(\lambda s\ z.(\ ((\lambda s\ z.(s\ (s\ z)))\ (\lambda z.(s\ (s\ z))))\ z))$

$(\lambda s\ z.((\lambda z.(\ (\lambda z.(s\ (s\ z)))\ (\ (\lambda z.(s\ (s\ z)))\ z)))\ z)$
$)$

```
(λs z.((λz.((λz.(s (s z))) ((λz.(s (s z))) z)))
    z))
```

Natural numbers


```
(λs z.((λz.((λz.(s (s z))) ((λz.(s (s z))) z)))
   z))
```

```
(λs z.((λz.((λz.(s (s z))) ((λz.(s (s z))) z))) z))
```

Natural numbers

$$(\lambda \texttt{s z.} \texttt{((}\lambda\texttt{z.((}\lambda\texttt{z.(s (s z))) ((}\lambda\texttt{z.(s (s z))) z))) z)})$$

Natural numbers

$(\lambda s\ z.((\lambda z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))\ z))$

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))$

Natural numbers


```
(λs z.((λz.(s (s z))) ((λz.(s (s z))) z)))
```

Natural numbers

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))$

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))$

Natural numbers

$$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))$$

Natural numbers

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ ((\lambda z.(s\ (s\ z)))\ z)))$

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ (s\ (s\ z))))$

Natural numbers

$$\boxed{(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ (s\ (s\ z))))}$$

Natural numbers

$(\lambda s\ z.((\lambda z.(s\ (s\ z)))\ (s\ (s\ z))))$

$(\lambda s\ z.\boxed{((\lambda z.(s\ (s\ z)))\ (s\ (s\ z)))})$

Natural numbers

$(\lambda \texttt{s z.} ((\lambda \texttt{z.(s (s z))) (s (s z)))})$

Natural numbers


```
(λs z.((λz.(s (s z))) (s (s z))))
```

```
(λs z.(s (s (s (s z)))))
```

Natural numbers

$$\boxed{(\lambda s\ z.(s\ (s\ (s\ (s\ z)))))}$$

Natural numbers


```
(λs z.(s (s (s (s z)))))
```

```
(λs z.(s (s (s (s z)))))
```

Natural numbers


```
(λs z.(s (s (s (s z)))))
```

Natural numbers


```
(λs z.(s (s (s (s z)))))
```

```
4
```

Zero checking

- We might wish to verify whether or not a number is zero
- We can simply pass the number parameters that fail the check (s) and pass it (z)
- $(\lambda n.((n\ \ (\lambda x.\mathtt{FALSE}))\ \ \mathtt{TRUE}))$

Zero checking

Let us try it out to $0 \; = \; 2 \rightarrow_\beta \texttt{FALSE}$

Natural numbers

$$\boxed{(2 \;=\; 0)}$$

$(2 \ = \ 0)$

$(\ 0? \ \ 2)$

Natural numbers


```
( 0?  2)
```

Natural numbers

(0? 2)

((λn.((n (λx.FALSE)) TRUE)) 2)

$$((\lambda n.((n\ (\lambda x.FALSE))\ TRUE))\ 2)$$

Natural numbers

$((\lambda n.((n\ (\lambda x.FALSE))\ TRUE))\ 2)$

$((\lambda n.((n\ (\lambda x.FALSE))\ TRUE))\ \boxed{2})$

Natural numbers

$$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ \boxed{2})$$

Natural numbers

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ 2)$

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ (\lambda s\ z.(s\ (s\ z))))$

Natural numbers

$$\boxed{\texttt{((}\lambda\texttt{n.((n (}\lambda\texttt{x.FALSE)) TRUE)) (}\lambda\texttt{s z.(s (s z))))}}$$

Natural numbers

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ (\lambda s\ z.(s\ (s\ z))))$

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ (\lambda s\ z.(s\ (s\ z))))$

Natural numbers

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ (\lambda s\ z.(s\ (s\ z))))$

$((\lambda n.((n\ (\lambda x.\mathrm{FALSE}))\ \mathrm{TRUE}))\ (\lambda s\ z.(s\ (s\ z))))$

$(((\lambda s\ z.(s\ (s\ z)))\ (\lambda x.\mathrm{FALSE}))\ \mathrm{TRUE})$

Natural numbers

$$\boxed{(((\lambda \mathtt{s}\ \mathtt{z}.(\mathtt{s}\ (\mathtt{s}\ \mathtt{z})))\ (\lambda \mathtt{x}.\mathtt{FALSE}))\ \mathtt{TRUE})}$$

Natural numbers


```
(((λs z.(s (s z))) (λx.FALSE)) TRUE)
```

```
( ((λs z.(s (s z))) (λx.FALSE)) TRUE)
```

Natural numbers

$$(((\lambda s\ z.(s\ (s\ z)))\ (\lambda x.\mathrm{FALSE}))\ \ \mathrm{TRUE})$$

Natural numbers

$(((\lambda s\ z.(s\ (s\ z)))\ (\lambda x.\mathrm{FALSE}))\ \mathrm{TRUE})$

$((\lambda z.((\lambda x.\mathrm{FALSE})\ ((\lambda x.\mathrm{FALSE})\ z)))\ \mathrm{TRUE})$

$$\boxed{((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ \mathtt{TRUE})}$$

Natural numbers

$((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ \mathtt{TRUE})$

$((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ \boxed{\mathtt{TRUE}})$

Natural numbers

$$((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ \boxed{\mathtt{TRUE}})$$

Natural numbers

$$((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ \mathtt{TRUE})$$

$$((\lambda z.((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ z)))\ (\lambda t\ f.t))$$

Natural numbers

```
((λz.((λx.FALSE) ((λx.FALSE) z))) (λt f.t))
```

Natural numbers

((λz.((λx.FALSE) ((λx.FALSE) z))) (λt f.t))

((λz.((λx.FALSE) ((λx.FALSE) z))) (λt f.t))

Natural numbers


```
((λz.((λx.FALSE) ((λx.FALSE) z))) (λt f.t))
```

Natural numbers


```
((λz.((λx.FALSE) ((λx.FALSE) z))) (λt f.t))
```

```
((λx.FALSE) ((λx.FALSE) (λt f.t)))
```

Natural numbers

$$((\lambda x.\mathrm{FALSE})\ ((\lambda x.\mathrm{FALSE})\ (\lambda t\ f.t)))$$

Natural numbers

$((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ (\lambda t\ f.t)))$

$((\lambda x.\mathtt{FALSE})\ ((\lambda x.\mathtt{FALSE})\ (\lambda t\ f.t)))$

Natural numbers

$$((\lambda x.\mathtt{FALSE})\ \ ((\lambda x.\mathtt{FALSE})\ \ (\lambda t\ f.t))\ )$$

Natural numbers


```
((λx.FALSE) ((λx.FALSE) (λt f.t)))
```

```
((λx.FALSE) FALSE)
```

$((\lambda x.\mathtt{FALSE})\ \mathtt{FALSE})$

Natural numbers

$((\lambda x.\texttt{FALSE})\ \texttt{FALSE})$

$((\lambda x.\texttt{FALSE})\ \texttt{FALSE})$

Natural numbers

$((\lambda x.\mathtt{FALSE})\ \mathtt{FALSE})$

Natural numbers

$((\lambda x.\mathtt{FALSE})\ \mathtt{FALSE})$

FALSE

```
FALSE
```

Natural numbers


```
FALSE
```

```
FALSE
```

Natural numbers

FALSE

Natural numbers

FALSE

$(\lambda t\ f.f)$

Natural numbers


```
(λt f.f)
```

Natural numbers

$(\lambda t\ \ f.f)$

$(\lambda t\ \ f.f)$

Natural numbers

$(\lambda t\ f.f)$

Natural numbers

$(\lambda t\ f.f)$

FALSE

Other arithmetic operators

- Division, subtraction, and all manners of comparison operators can be defined similarly
- The level of detail of the specification can be compared to that of a very high level CPU
- This means that we are, to an extent, programming in a sort of assembly
- This is the reason why the traces have been so verbose so far

Other arithmetic operators

- We could also define numbers in base two instead of base one
- This would save processing time, but would result in a slighter more complex specification
- We will just ignore these engineering details: we only focus on **what** can be done, not the best way to do it

Conclusion

Conclusion

Recap

- Lambda terms can be used to encode arbitrary basic data types
- The terms are always lambda expression which, when they get parameters passed in, identify themselves somehow
- Identification can be done by applying something (possibly even a given number of times), or returning one of the parameters

Conclusion

Recap

- There are many encodings of data types, but they all behave in the same way by producing the same outputs for the same inputs
- From now on we will start ignoring the reduction steps for simple terms such as 3+3
- We will instead focus on more complex data structures, such as tuples, discriminated unions, and even lists

Appendix

Appendix

```
((FALSE bit1) bit0)
```

Appendix


```
((FALSE bit1) bit0)
```

```
((FALSE bit1) bit0)
```

Appendix


```
((FALSE bit1) bit0)
```

```
((FALSE bit1) bit0)
```

```
(((λt f.f) bit1) bit0)
```

```
(((λt f.f) bit1) bit0)
```

Appendix


```
(((λt f.f) bit1) bit0)
```

```
( ((λt f.f) bit1) bit0)
```

Appendix

$(((\lambda t\ f.f)\ bit1)\ \ bit0)$

Appendix


```
( ((λt f.f) bit1)  bit0)
```

```
((λf.f) bit0)
```

```
((λf.f) bit0)
```

Appendix


```
((λf.f) bit0)
```

```
((λf.f) bit0)
```

```
((λf.f) bit0)
```

Appendix


```
((λf.f) bit0)
```

```
bit0
```

Remaining and derivations

Let us move to $\texttt{TRUE} \wedge \texttt{FALSE} \rightarrow_\beta \texttt{FALSE}$

$$\boxed{(\mathtt{TRUE} \wedge \mathtt{FALSE})}$$

$(\mathtt{TRUE} \wedge \mathtt{FALSE})$

$((\boxed{\wedge}\ \mathtt{TRUE})\ \mathtt{FALSE})$

```
((∧ TRUE) FALSE)
```

Appendix

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((∧ TRUE) FALSE)
```

```
(((λp q.((p q) p)) TRUE) FALSE)
```

Appendix

$$\boxed{(((\lambda p\ q.((p\ q)\ p))\ \mathtt{TRUE})\ \mathtt{FALSE})}$$

Appendix


```
(((λp q.((p q) p)) TRUE) FALSE)
```

```
(((λp q.((p q) p)) TRUE) FALSE)
```

Appendix


```
(((λp q.((p q) p)) TRUE) FALSE)
```

```
(((λp q.((p q) p)) TRUE ) FALSE)
```

```
(((λp q.((p q) p)) (λt f.t) ) FALSE)
```

Appendix


```
(((λp q.((p q) p)) (λt f.t)) FALSE)
```

Appendix


```
(((λp q.((p q) p)) (λt f.t)) FALSE)
```

```
( ((λp q.((p q) p)) (λt f.t)) FALSE)
```

$$(\boxed{((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.t))}\ \ \mathtt{FALSE})$$

```
( ((λp q.((p q) p)) (λt f.t)) FALSE)
```

```
((λq.(( (λt f.t) q) (λt f.t) )) FALSE)
```

$((\lambda q.(((\lambda t\ f.t)\ q)\ (\lambda t\ f.t)))\ \mathtt{FALSE})$

Appendix


```
((λq.(((λt f.t) q) (λt f.t))) FALSE)
```

```
((λq.(((λt f.t) q) (λt f.t))) FALSE)
```

Appendix

$$((\lambda q.(((\lambda t\ f.t)\ q)\ (\lambda t\ f.t)))\ \boxed{\mathtt{FALSE}})$$

Appendix


```
((λq.(((λt f.t) q) (λt f.t))) FALSE)
```

```
((λq.(((λt f.t) q) (λt f.t))) (λt f.f))
```

Appendix

$$\boxed{((\lambda q.(((\lambda t\ f.t)\ q)\ (\lambda t\ f.t)))\ (\lambda t\ f.f))}$$

Appendix


```
((λq.(((λt f.t) q) (λt f.t))) (λt f.f))
```

```
((λq.(((λt f.t) q) (λt f.t))) (λt f.f))
```

Appendix

$((\lambda \mathtt{q}.(((\lambda \mathtt{t}\ \ \mathtt{f}.\mathtt{t})\ \ \mathtt{q})\ \ (\lambda \mathtt{t}\ \ \mathtt{f}.\mathtt{t})))\ \ (\lambda \mathtt{t}\ \ \mathtt{f}.\mathtt{f}))$

```
((λq.(((λt f.t) q) (λt f.t))) (λt f.f))
```

```
(((λt f.t) (λt f.f)) (λt f.t))
```

Appendix


```
(((λt f.t) (λt f.f)) (λt f.t))
```

```
(((λt f.t) (λt f.f)) (λt f.t))
```

```
( ((λt f.t) (λt f.f)) (λt f.t))
```

Appendix

$$(((\lambda t\ \ f.t)\ \ (\lambda t\ \ f.f))\ \ (\lambda t\ \ f.t))$$

Appendix

`( ((λt f.t) (λt f.f)) (λt f.t))`

`((λf t f . f ) (λt f.t))`

```
((λf t f.f) (λt f.t))
```

```
((λf t f.f) (λt f.t))
```

```
((λf t f.f) (λt f.t))
```

$((\lambda f\ t\ f.f)\ (\lambda t\ f.t))$

Appendix


```
((λf t f.f) (λt f.t))
```

```
(λt f.f)
```

Appendix

$(\lambda t\ \ f.f)$

```
(λt f.f)
```

```
(λt f.f)
```

Appendix

$(\lambda \texttt{t f.f})$

Appendix

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

$(\lambda$t f.f)

FALSE

Remaining and derivations

Let us move to $\texttt{FALSE} \wedge \texttt{TRUE} \rightarrow_\beta \texttt{FALSE}$

Appendix

$$\boxed{(\texttt{FALSE} \wedge \texttt{TRUE})}$$

Appendix

$(\mathtt{FALSE} \wedge \mathtt{TRUE})$

$((\boxed{\wedge}\ \mathtt{FALSE})\ \mathtt{TRUE})$

Appendix


```
((∧ FALSE) TRUE)
```

Appendix

$$((\wedge \ \texttt{FALSE}) \ \texttt{TRUE})$$

$$((\ (\lambda p \ q.((p \ q) \ p)) \ \texttt{FALSE}) \ \texttt{TRUE})$$

Appendix


```
(((λp q.((p q) p)) FALSE) TRUE)
```

```
(((λp q.((p q) p)) FALSE) TRUE)
```

```
(((λp q.((p q) p)) FALSE) TRUE)
```

Appendix


```
(((λp q.((p q) p)) FALSE) TRUE)
```

```
(((λp q.((p q) p)) FALSE) TRUE)
```

```
(((λp q.((p q) p)) (λt f.f)) TRUE)
```

Appendix

$$\boxed{(((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.f))\ \mathtt{TRUE})}$$

```
(((λp q.((p q) p)) (λt f.f)) TRUE)
```

```
( ((λp q.((p q) p)) (λt f.f)) TRUE)
```

Appendix

$$(\ \boxed{((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.f))}\ \ \mathtt{TRUE})$$

Appendix


```
( ((λp q.((p q) p)) (λt f.f)) TRUE)
```

```
((λq.(( (λt f.f) q) (λt f.f) )) TRUE)
```

Appendix


```
((λq.(((λt f.f) q) (λt f.f))) TRUE)
```

```
((λq.(((λt f.f) q) (λt f.f))) TRUE)
```

```
((λq.(((λt f.f) q) (λt f.f))) TRUE)
```

Appendix


```
((λq.(((λt f.f) q) (λt f.f))) TRUE)
```

$((\lambda q.(((\lambda t\ \ f.f)\ \ q)\ \ (\lambda t\ \ f.f)))\ \ \boxed{\text{TRUE}})$

$((\lambda q.(((\lambda t\ \ f.f)\ \ q)\ \ (\lambda t\ \ f.f)))\ \ \boxed{(\lambda t\ \ f.t)})$

Appendix

$$\boxed{((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.t))}$$

Appendix

```
((λq.(((λt f.f) q) (λt f.f))) (λt f.t))
```

```
((λq.(((λt f.f) q) (λt f.f))) (λt f.t))
```

Appendix

$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.t))$

Appendix

$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.t))$

$(((\lambda t\ f.f)\ (\lambda t\ f.t))\ (\lambda t\ f.f))$

Appendix


```
(((λt f.f) (λt f.t)) (λt f.f))
```

```
(((λt f.f) (λt f.t)) (λt f.f))
```

```
( ((λt f.f) (λt f.t)) (λt f.f))
```

Appendix


```
( ((λt f.f) (λt f.t)) (λt f.f))
```

Appendix


```
( ((λt f.f) (λt f.t)) (λt f.f))
```

```
((λf.f) (λt f.f))
```

Appendix


```
((λf.f) (λt f.f))
```

```
((λf.f) (λt f.f))
```

```
((λf.f) (λt f.f))
```

```
((λf.f) (λt f.f))
```

Appendix

`((λf.f) (λt f.f))`

`(λt f.f)`

Appendix

$(\lambda t \ f.f)$

Appendix

$(\lambda t\ \ f.f)$

$(\lambda t\ \ f.f)$

Appendix

$(\lambda t\ f.f)$

Appendix

$(\lambda t\ f.f)$

FALSE

Remaining and derivations

Let us move to $\texttt{FALSE} \wedge \texttt{FALSE} \rightarrow_\beta \texttt{FALSE}$

Appendix


```
(FALSE ∧ FALSE)
```

Appendix


```
(FALSE ∧ FALSE)
```

```
((∧ FALSE) FALSE)
```

Appendix

```
((∧ FALSE) FALSE)
```

Appendix


```
((∧ FALSE) FALSE)
```

```
(( (λp q.((p q) p)) FALSE) FALSE)
```

Appendix


```
(((λp q.((p q) p)) FALSE) FALSE)
```

Appendix


```
(((λp q.((p q) p)) FALSE) FALSE)
```

```
(((λp q.((p q) p)) FALSE) FALSE)
```

```
(((λp q.((p q) p)) FALSE) FALSE)
```

Appendix

$(((\lambda p\ q.((p\ q)\ p))\ \texttt{FALSE})\ \texttt{FALSE})$

$(((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.f))\ \texttt{FALSE})$

Appendix


```
(((λp q.((p q) p)) (λt f.f)) FALSE)
```

Appendix


```
(((λp q.((p q) p)) (λt f.f)) FALSE)
```

```
( ((λp q.((p q) p)) (λt f.f)) FALSE)
```

Appendix

$(\boxed{((\lambda p\ q.((p\ q)\ p))\ (\lambda t\ f.f))}\ \texttt{FALSE})$

```
( ((λp q.((p q) p)) (λt f.f)) FALSE)
```

```
((λq.(( (λt f.f) q) (λt f.f) )) FALSE)
```

Appendix


```
((λq.(((λt f.f) q) (λt f.f))) FALSE)
```

Appendix


```
((λq.(((λt f.f) q) (λt f.f))) FALSE)
```

```
((λq.(((λt f.f) q) (λt f.f))) FALSE)
```

```
((λq.(((λt f.f) q) (λt f.f))) FALSE)
```

Appendix

$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ \boxed{\texttt{FALSE}})$

$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ \boxed{(\lambda t\ f.f)})$

Appendix

$$\boxed{((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.f))}$$

```
((λq.(((λt f.f) q) (λt f.f))) (λt f.f))
```

```
((λq.(((λt f.f) q) (λt f.f))) (λt f.f))
```

$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.f))$

Appendix

$$((\lambda q.(((\lambda t\ f.f)\ q)\ (\lambda t\ f.f)))\ (\lambda t\ f.f))$$

$$(((\lambda t\ f.f)\ (\lambda t\ f.f))\ (\lambda t\ f.f))$$

$$(((\lambda \texttt{t f.f})\ (\lambda \texttt{t f.f}))\ (\lambda \texttt{t f.f}))$$

Appendix


```
(((λt f.f) (λt f.f)) (λt f.f))
```

```
( ((λt f.f) (λt f.f)) (λt f.f))
```

$$(((\lambda t\ f.f)\ (\lambda t\ f.f))\ (\lambda t\ f.f))$$

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

$(\ ((\lambda t\ \ f.f)\ \ (\lambda t\ \ f.f))\ \ (\lambda t\ \ f.f))$

$((\lambda f.f)\ \ (\lambda t\ \ f.f))$

Appendix

```
((λf.f) (λt f.f))
```

Appendix

$((\lambda f.f)\ (\lambda t\ f.f))$

$((\lambda f.f)\ (\lambda t\ f.f))$

Appendix

$((\lambda \texttt{f.f})\ (\lambda \texttt{t f.f}))$

Appendix


```
((λf.f) (λt f.f))
```

```
(λt f.f)
```

Appendix


```
(λt f.f)
```

Appendix


```
(λt  f.f)
```

```
(λt  f.f)
```

Appendix


```
(λt f.f)
```

Appendix

$(\lambda t\ f.f)$

FALSE

Remaining or derivations

Let us begin to with $\texttt{TRUE} \vee \texttt{FALSE} \rightarrow_\beta \texttt{TRUE}$

Appendix

$$(\mathtt{TRUE} \lor \mathtt{FALSE})$$

Appendix


```
(TRUE ∨ FALSE)
```

```
((∨ TRUE) FALSE)
```

Appendix

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
((∨ TRUE) FALSE)
```

Appendix


```
((∨ TRUE) FALSE)
```

```
(( (λp q.((p p) q)) TRUE) FALSE)
```

Appendix


```
(((λp q.((p p) q)) TRUE) FALSE)
```

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
(((λp q.((p p) q)) TRUE) FALSE)
```

```
(((λp q.((p p) q)) TRUE) FALSE)
```

Appendix


```
(((λp q.((p p) q)) TRUE) FALSE)
```

Appendix


```
((((λp q.((p p) q)) TRUE ) FALSE)
```

```
((((λp q.((p p) q)) (λt f.t) ) FALSE)
```

$$\boxed{\texttt{(((}\lambda\texttt{p q.((p p) q)) (}\lambda\texttt{t f.t)) FALSE)}}$$

Appendix


```
(((λp q.((p p) q)) (λt f.t)) FALSE)
```

```
( ((λp q.((p p) q)) (λt f.t)) FALSE)
```

Appendix

$(\boxed{((\lambda p\ q.((p\ p)\ q))\ (\lambda t\ f.t))}\ \mathtt{FALSE})$

Appendix

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

```
( ((λp q.((p p) q)) (λt f.t)) FALSE)
```

```
((λq.(( (λt f.t) (λt f.t) ) q)) FALSE)
```

Appendix

$$\boxed{((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \mathtt{FALSE})}$$

Appendix

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \mathtt{FALSE})$

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \boxed{\mathtt{FALSE}})$

```
((λq.(((λt f.t) (λt f.t)) q)) FALSE)
```

Appendix

$$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \boxed{\mathtt{FALSE}})$$

$$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ \boxed{(\lambda t\ f.f)})$$

Appendix

$$((\lambda q.(((\lambda t\ \ f.t)\ \ (\lambda t\ \ f.t))\ \ q))\ \ (\lambda t\ \ f.f))$$

Appendix

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ (\lambda t\ f.f))$

$((\lambda q.(((\lambda t\ f.t)\ (\lambda t\ f.t))\ q))\ (\lambda t\ f.f))$

Appendix

$((\lambda \mathtt{q}.(((\lambda \mathtt{t}\ \mathtt{f}.\mathtt{t})\ (\lambda \mathtt{t}\ \mathtt{f}.\mathtt{t}))\ \mathtt{q}))\ (\lambda \mathtt{t}\ \mathtt{f}.\mathtt{f}))$

```
((λq.(((λt f.t) (λt f.t)) q)) (λt f.f))
```

```
(((λt f.t) (λt f.t)) (λt f.f))
```

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

Appendix

```
(((λt f.t) (λt f.t)) (λt f.f))
```

```
(((λt f.t) (λt f.t)) (λt f.f))
```

```
( ((λt f.t) (λt f.t)) (λt f.f))
```

```
( ((λt f.t) (λt f.t)) (λt f.f))
```

```
( ((λt f.t) (λt f.t)) (λt f.f))
```

```
((λf t f . t ) (λt f.f))
```

Appendix

$$\boxed{((\lambda f\ t\ f.t)\ (\lambda t\ f.f))}$$

Appendix


```
((λf t f.t) (λt f.f))
```

```
((λf t f.t) (λt f.f))
```

Appendix

$((\lambda$f t f.t) $(\lambda$t f.f))

Appendix

$((\lambda f\ t\ f.t)\ (\lambda t\ f.f))$

$(\lambda t\ f.t)$

Appendix

$(\lambda \texttt{t} \;\; \texttt{f.t})$

Appendix


```
(λt f.t)
```

```
(λt f.t)
```

Appendix


```
(λt f.t)
```

Appendix

$(\lambda t\ f.t)$

TRUE

Remaining or derivations

Let us begin to with $\texttt{False} \lor \texttt{TRUE} \rightarrow_\beta \texttt{TRUE}$

Appendix

$$\boxed{(\texttt{FALSE} \vee \texttt{TRUE})}$$

```
(FALSE ∨ TRUE)
```

```
((∨ FALSE) TRUE)
```

Appendix

```
((∨ FALSE) TRUE)
```

Appendix

$((\vee \;\; \mathtt{FALSE}) \;\; \mathtt{TRUE})$

$((\; (\lambda p \; q.((p \; p) \; q)) \;\; \mathtt{FALSE}) \;\; \mathtt{TRUE})$

Appendix


```
(((λp q.((p p) q)) FALSE) TRUE)
```

Appendix

$(((\lambda p\ q.((p\ p)\ q))\ \mathtt{FALSE})\ \mathtt{TRUE})$

$(((\lambda p\ q.((p\ p)\ q))\ \boxed{\mathtt{FALSE}})\ \mathtt{TRUE})$

Appendix


```
(((λp q.((p p) q)) FALSE) TRUE)
```

Appendix


```
(((λp q.((p p) q)) FALSE) TRUE)
```

```
(((λp q.((p p) q)) (λt f.f)) TRUE)
```

Appendix

$$\boxed{(((\lambda p\ q.((p\ p)\ q))\ (\lambda t\ f.f))\ \mathrm{TRUE})}$$

Appendix


```
(((λp q.((p p) q)) (λt f.f)) TRUE)
```

```
( ((λp q.((p p) q)) (λt f.f))  TRUE)
```

$(\boxed{((\lambda p\ q.((p\ p)\ q))\ (\lambda t\ f.f))}\ \mathtt{TRUE})$

HOGESCHOOL ROTTERDAM

Appendix

$\texttt{(} \colorbox{orange}{$\texttt{((}\lambda\texttt{p q.((p p) q)) (}\lambda\texttt{t f.f))}$} \texttt{ TRUE)}$

$\texttt{((}\lambda\texttt{q.((} \colorbox{yellow}{$(\lambda\texttt{t f.f)}$} \; \colorbox{yellow}{$(\lambda\texttt{t f.f)}$} \texttt{) q)) TRUE)}$

Appendix


```
((λq.(((λt f.f) (λt f.f)) q)) TRUE)
```

Appendix

$$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ \texttt{TRUE})$$

$$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ \boxed{\texttt{TRUE}})$$

Appendix


```
((λq.(((λt f.f) (λt f.f)) q)) TRUE )
```

Appendix


```
((λq.(((λt f.f) (λt f.f)) q)) TRUE )
```

```
((λq.(((λt f.f) (λt f.f)) q)) (λt f.t) )
```

Appendix

$$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ (\lambda t\ f.t))$$

Appendix


```
((λq.(((λt f.f) (λt f.f)) q)) (λt f.t))
```

```
((λq.(((λt f.f) (λt f.f)) q)) (λt f.t))
```

```
((λq.(((λt f.f) (λt f.f)) q)) (λt f.t))
```

Appendix

((λq.(((λt f.f) (λt f.f)) q)) (λt f.t))

(((λt f.f) (λt f.f)) (λt f.t))

Appendix


```
(((λt f.f) (λt f.f)) (λt f.t))
```

```
(((λt f.f) (λt f.f)) (λt f.t))
```

```
( ((λt f.f) (λt f.f)) (λt f.t))
```

Appendix

$$(((\lambda t\ f.f)\ (\lambda t\ f.f))\ (\lambda t\ f.t))$$

Appendix


```
( ((λt f.f) (λt f.f)) (λt f.t))
```

```
((λf.f) (λt f.t))
```

Appendix

$$((\lambda f.f)\ (\lambda t\ f.t))$$

Appendix


```
((λf.f) (λt f.t))
```

```
((λf.f) (λt f.t))
```

```
((λf.f) (λt f.t))
```

Appendix

$((\lambda f.f)\ (\lambda t\ f.t))$

$(\lambda t\ f.t)$

$(\lambda$t f.t)

Appendix

$(\lambda \texttt{t f.t})$

$(\lambda \texttt{t f.t})$

Appendix

$(\lambda t\ f.t)$

Appendix

$(\lambda t\ f.t)$

TRUE

Remaining or derivations

Let us begin to with $\texttt{FALSE} \vee \texttt{FALSE} \rightarrow_\beta \texttt{FALSE}$

Appendix

$$(\mathtt{FALSE} \lor \mathtt{FALSE})$$

Appendix


```
(FALSE ∨ FALSE)
```

```
((∨ FALSE) FALSE)
```

```
((∨ FALSE) FALSE)
```

Appendix


```
((∨ FALSE) FALSE)
```

```
(((λp q.((p p) q)) FALSE) FALSE)
```

Appendix


```
(((λp q.((p p) q)) FALSE) FALSE)
```

```
(((λp q.((p p) q)) FALSE) FALSE)
```

```
(((λp q.((p p) q)) FALSE) FALSE)
```

Appendix

$(((\lambda p\ q.((p\ p)\ q))\ \texttt{FALSE})\ \texttt{FALSE})$

Appendix

$$(((\lambda p\ q.((p\ p)\ q))\ \texttt{FALSE})\ \texttt{FALSE})$$

$$(((\lambda p\ q.((p\ p)\ q))\ (\lambda t\ f.f))\ \texttt{FALSE})$$

Appendix


```
(((λp q.((p p) q)) (λt f.f)) FALSE)
```

Appendix


```
(((λp q.((p p) q)) (λt f.f)) FALSE)
```

```
( ((λp q.((p p) q)) (λt f.f)) FALSE)
```

(((λp q.((p p) q)) (λt f.f)) FALSE)

Appendix


```
( ((λp q.((p p) q)) (λt f.f)) FALSE)
```

```
((λq.(( (λt f.f) (λt f.f) ) q)) FALSE)
```

```
((λq.(((λt f.f) (λt f.f)) q)) FALSE)
```

Appendix

$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ \mathtt{FALSE})$

$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ \boxed{\mathtt{FALSE}})$

Appendix


```
((λq.(((λt f.f) (λt f.f)) q)) FALSE)
```

$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ \text{FALSE}\ )$

$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ (\lambda t\ f.f)\ )$

$$((\lambda q.(((\lambda t\ \ f.f)\ \ (\lambda t\ \ f.f))\ \ q))\ \ (\lambda t\ \ f.f))$$

$$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ (\lambda t\ f.f))$$

$$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ (\lambda t\ f.f))$$

$((\lambda q.(((\lambda t\ f.f)\ (\lambda t\ f.f))\ q))\ (\lambda t\ f.f))$

Appendix

```
((λq.(((λt f.f) (λt f.f)) q)) (λt f.f))
```

```
(((λt f.f) (λt f.f)) (λt f.f))
```

Appendix


```
(((λt f.f) (λt f.f)) (λt f.f))
```

```
(((λt f.f) (λt f.f)) (λt f.f))
```

```
( ((λt f.f) (λt f.f)) (λt f.f))
```

Appendix

(((λt f.f) (λt f.f)) (λt f.f))

Appendix


```
( ((λt f.f) (λt f.f)) (λt f.f))
```

```
((λf.f) (λt f.f))
```

Appendix


```
((λf.f) (λt f.f))
```

Appendix


```
((λf.f) (λt f.f))
```

```
((λf.f) (λt f.f))
```

Appendix


```
((λf.f) (λt f.f))
```

Appendix

`((λf.f) (λt f.f))`

`(λt f.f)`

Appendix


```
(λt f.f)
```

```
(λt f.f)
```

```
(λt f.f)
```

Appendix

$(\lambda t\ f.f)$

Appendix

$(\lambda t\ f.f)$

FALSE

Remaining numeral derivations

Let us try out $0 \;=\; 0 \rightarrow_\beta \texttt{TRUE}$

Appendix

$$\boxed{(0 \quad = \quad 0)}$$

Appendix


```
(0 = 0)
```

```
( 0?  0)
```

Appendix


```
( 0?  0)
```

Appendix


```
( 0?  0)
```

```
( (λn.((n (λx.FALSE)) TRUE))  0)
```

Appendix


```
((λn.((n (λx.FALSE)) TRUE)) 0)
```

Appendix

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ 0)$

$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ \boxed{0})$

$$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ \boxed{0})$$

Appendix


```
((λn.((n (λx.FALSE)) TRUE)) 0)
```

```
((λn.((n (λx.FALSE)) TRUE)) (λs z.z))
```

Appendix


```
((λn.((n (λx.FALSE)) TRUE)) (λs z.z))
```

Appendix

$((\lambda n.((n\ (\lambda x.FALSE))\ TRUE))\ (\lambda s\ z.z))$

$((\lambda n.((n\ (\lambda x.FALSE))\ TRUE))\ (\lambda s\ z.z))$

Appendix

$$((\lambda n.((n\ (\lambda x.\mathtt{FALSE}))\ \mathtt{TRUE}))\ (\lambda s\ z.z))$$

Appendix

$((\lambda n.((n\ (\lambda x.\mathrm{FALSE}))\ \mathrm{TRUE}))\ (\lambda s\ z.z))$

$((\ (\lambda s\ z.z)\ \ (\lambda x.\mathrm{FALSE}))\ \ \mathrm{TRUE})$

Appendix

```
(((λs z.z) (λx.FALSE)) TRUE)
```

Appendix

$(((\lambda s\ z.z)\ (\lambda x.FALSE))\ TRUE)$

$(\ ((\lambda s\ z.z)\ (\lambda x.FALSE))\ \ TRUE)$

Appendix

(((λs z.z) (λx.FALSE)) TRUE)

Appendix

$$(\ ((\lambda s\ z.z)\ (\lambda x.FALSE))\ \ TRUE)$$

$$((\lambda z.z)\ \ TRUE)$$

Appendix

$$((\lambda z.z)\ \ \mathtt{TRUE})$$

Appendix

Delta rules

The INFDEV@HR Team

Introduction

Encoding boolean logic

Defining terms with special meaning

Booleans

Natural numbers

Conclusion

Appendix

((λz.z) TRUE)

((λz.z) TRUE)

$((\lambda z.z)\ \boxed{\mathtt{TRUE}})$

Appendix

((λz.z) TRUE)

((λz.z) (λt f.t))

$$\boxed{((\lambda z.z)\ \ (\lambda t\ \ f.t))}$$

Appendix


```
((λz.z) (λt f.t))
```

```
((λz.z) (λt f.t))
```

Appendix

$((\lambda z.z)\ (\lambda t\ f.t))$

Appendix

$((\lambda z.z)\ (\lambda t\ f.t))$

$(\lambda t\ f.t)$

Appendix


```
(λt  f.t)
```

Appendix

```
(λt  f.t)
```

```
(λt  f.t)
```

Appendix

$(\lambda t\ f.t)$

Appendix

`(λt f.t)`

`TRUE`

This is it!

The best of luck, and thanks for the attention!