Name: Jinghan Wang

Student Number: z5286124

The University of New South Wales

COMP3161/9164 22T3
Concepts of Programming Languages

Assignment 0 Proofs

Term 3, 2022

Marks: 24 Hours. 15% of overall marks for the course.
A mark of x (out of 100) on this assignment will translate to .15x course marks.

Due date: Sunday, 9th of October 2022, 12 noon (Sydney Time)

Task

In this assignment you will formally model a language of boolean computations using a variety of semantic techniques, including its syntax and sematics, and its compilation to various target languages.

Prepare your answers in one PDF file, preferably using LaTeX, where all prose is typeset. Figures may be drawn, but make sure they are legible. Please ensure all mathematics is formatted correctly. Some guidance will be posted on the course website.

Submit your PDF using the CSE give system, by typing the command

give cs3161 Proofs Proofs.pdf

or by using the CSE give web interface.

Part A (25 marks)

Consider the language of boolean expressions $\mathcal{P}$ containing just literals (`True`, `False`), parentheses, conjunction ($\wedge$) and negation ($\neg$):

$$\mathcal{P} = \{\texttt{True}, \texttt{False}, \neg\texttt{True}, \neg\texttt{False}, \texttt{True} \wedge \texttt{False}, \neg(\texttt{True} \wedge \texttt{False}), \dots\}$$

1. Write down a set of inference rules that define the set $\mathcal{P}$. The rules may be ambiguous. *(5 marks)*

Solution:

$$\frac{c \in \{\texttt{True}, \texttt{False}\}}{c\ \texttt{Bool}} \qquad \frac{e\ \texttt{Bool}}{\neg e\ \texttt{Bool}} \qquad \frac{e\ \texttt{Bool}}{(e)\ \texttt{Bool}} \qquad \frac{e_1\ \texttt{Bool} \quad e_2\ \texttt{Bool}}{e_1 \wedge e_2\ \texttt{Bool}}$$

2. The operator $\neg$ has the highest precedence, and conjunction is right-associative. Define a set of simultaneous judgements to define the language without any ambiguity. *(5 marks)*

Solution:

$$\frac{c \in \{\texttt{True}, \texttt{False}\}}{c\ \texttt{Atom}} \qquad \frac{e\ \texttt{Atom}}{\neg e\ \texttt{Atom}} \qquad \frac{e\ \texttt{AndEx}}{(e)\ \texttt{Atom}}$$

$$\frac{e_1\ \texttt{Atom} \quad e_2\ \texttt{AndEx}}{e_1 \wedge e_2\ \texttt{AndEx}} \qquad \frac{e\ \texttt{Atom}}{e\ \texttt{AndEx}}$$

3. Here is an abstract syntax B for the same language:

$$\mathcal{B} ::= \mathsf{Not}\ \mathcal{B} \mid \mathsf{And}\ \mathcal{B}\ \mathcal{B} \mid \mathsf{True} \mid \mathsf{False}$$

Write an inductive definition for the *parsing* relation connecting your unambiguous judgements to this abstract syntax. *(5 marks)*

Solution:

$$\frac{}{\mathsf{True}\ \mathcal{B} \longleftrightarrow \mathsf{True}}\mathsf{True} \qquad \frac{}{\mathsf{False}\ \mathcal{B} \longleftrightarrow \mathsf{False}}\mathsf{False}$$

$$\frac{x\ \mathcal{B} \longleftrightarrow x' \quad y\ \mathcal{B} \longleftrightarrow y'}{\mathsf{And}\ a\ b\ \mathcal{B} \longleftrightarrow a' \wedge b'}\mathsf{And} \qquad \frac{x\ \mathcal{B} \longrightarrow x'}{\mathsf{Not}\ x\ \mathcal{B} \longrightarrow \neg x'}\mathsf{Not}$$

4. Here is a big-step semantics for the language $\mathcal{B}$

$$\frac{x \Downarrow \mathsf{True}}{\mathsf{Not}\ x \Downarrow \mathsf{False}} \qquad \frac{x \Downarrow \mathsf{False}}{\mathsf{Not}\ x \Downarrow \mathsf{True}} \qquad \frac{x \Downarrow \mathsf{False}}{\mathsf{And}\ x\ y \Downarrow \mathsf{False}} \qquad \frac{x \Downarrow \mathsf{True} \quad y \Downarrow v}{\mathsf{And}\ x\ y \Downarrow v}$$

$$\frac{}{\mathsf{True} \Downarrow \mathsf{True}} \qquad \frac{}{\mathsf{False} \Downarrow \mathsf{False}}$$

a) Show the evaluation of And (Not (And True False)) True with a derivation tree. *(5 marks)*

Solution:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{True} \Downarrow \mathsf{True}} \quad \dfrac{}{\mathsf{False} \Downarrow \mathsf{False}}}{\mathsf{And\ True\ False} \Downarrow \mathsf{False}}}{\mathsf{Not\ (And\ True\ False)} \Downarrow \mathsf{True}} \qquad \dfrac{}{\mathsf{True} \Downarrow \mathsf{True}}}{\mathsf{And\ (Not\ (And\ True\ False))True} \Downarrow \mathsf{True}}$$

b) Consider the following inference rule:

$$\frac{y \Downarrow \mathsf{False}}{\mathsf{And}\ x\ y \Downarrow \mathsf{False}}$$

If we assume that $x\ \mathcal{B}$ holds, is this rule derivable? Is it admissible? And if we *don't* assume that $x\ \mathcal{B}$ holds, how does this change your answers? Justify your answers. *(5 marks)*

Solution:

* If the $x\ \mathcal{B}$ is not holds, the inference rule can derivable and admissible.

  According to the topic, the And $x\ y\ \Downarrow$ False have two composition method.

  First is followed the forth semantics which the x is True situation, which is followed by the inference rule.

$$\frac{\dfrac{}{x \Downarrow \mathsf{True}} \quad \dfrac{}{y \Downarrow \mathsf{False}}}{\mathsf{And}\ x\ y\ \Downarrow \mathsf{False}}$$

  Second is followed the third semantics which the x is False situation. Whatever y is, the inference rule is following.

$$\frac{\dfrac{}{x \Downarrow \mathsf{False}} \quad \dfrac{}{y \Downarrow \mathsf{False}}}{\mathsf{And}\ x\ y\ \Downarrow \mathsf{False}}$$

  According to the two discussion above, the inference rule can derivable and is admissible.

* If the $x\ \mathcal{B}$ is not holds, the inference rule is also admissible but cannot derivable. For the And operation, if any element of the And operation is not True, the result is False. So this inference rule is admissible.

Part B (20 marks)

Here is a small-step semantics for a language $\mathcal{L}$ with True, False and if expressions:

$$\frac{c \mapsto c'}{(\text{If } c\ t\ e) \mapsto (\text{If } c'\ t\ e)}(1) \quad \frac{}{(\text{If True } t\ e) \mapsto t}(2) \quad \frac{}{(\text{If False } t\ e) \mapsto e}(3)$$

1. Show the full evaluation of the term (If True (If True False True) False). (5 marks)

Solution:

$$\begin{aligned} & (\text{If True } (\text{If True False True}) \text{ False}) \\ \mapsto\ & (\text{If True False False}) && \text{According to (2)} \\ \mapsto\ & \text{False} && \text{According to (2)} \end{aligned}$$

2. Define an equivalent *big-step* semantics for $\mathcal{L}$. *(5 marks)*

Solution:

$$\frac{}{\text{True} \Downarrow \text{True}}\,\mathsf{True} \qquad \frac{}{\text{False} \Downarrow \text{False}}\,\mathsf{False}$$

$$\frac{c \Downarrow \text{True} \quad t \Downarrow t'}{(\text{If } c\ t\ e) \Downarrow t'}\,\mathsf{If}_1 \qquad \frac{c \Downarrow \text{False} \quad e \Downarrow e'}{(\text{If } c\ t\ e) \Downarrow e'}\,\mathsf{If}_2$$

3. Prove that if $e \Downarrow v$ then $e \overset{\star}{\mapsto} v$, where $\Downarrow$ is the big-step semantics you defined in the previous question, and $\overset{\star}{\mapsto}$ is the reflexive and transitive closure of $\mapsto$. Use rule induction on $e \Downarrow v$. *(10 marks)*

Solution:

According to the topic, $\overset{\star}{\mapsto}$ be the reflexive and transitive closure of $\mapsto$. We can mark them refl and trans.

$$\frac{}{e \overset{\star}{\mapsto} e}\,\mathsf{refl} \qquad \frac{e \mapsto e' \quad e' \overset{\star}{\mapsto} e''}{e \overset{\star}{\mapsto} e''}\,\mathsf{trans}$$

We also want to prove $\mathsf{If\ True}\ t\ e \overset{\star}{\mapsto} t$ and $\mathsf{If\ False}\ t\ e \overset{\star}{\mapsto} e$ is true.

$$\frac{\dfrac{}{(\mathsf{If\ True}\ t\ e) \mapsto t}(2) \quad \dfrac{}{t \overset{\star}{\mapsto} t}\,\mathsf{refl}}{(\mathsf{If\ True}\ t\ e) \overset{\star}{\mapsto} t}\,\mathsf{trans}\ (2') \qquad \frac{\dfrac{}{(\mathsf{If\ False}\ t\ e) \mapsto e}(3) \quad \dfrac{}{e \overset{\star}{\mapsto} e}\,\mathsf{refl}}{(\mathsf{If\ False}\ t\ e) \overset{\star}{\mapsto} e}\,\mathsf{trans}\ (3')$$

Base Case ($e = v$, *from rule* True) We must show that $\mathsf{True} \overset{\star}{\mapsto} \mathsf{True}$, which is true by rule refl.

Base Case ($e = v,\ from\ rule$ False) We must show that $\mathsf{False} \overset{\star}{\mapsto} \mathsf{False}$, which is true by rule refl.

Inductive Case ($from\ rule$ $\mathsf{If_1}$) We know that $c \Downarrow \mathsf{True}$ and $t \Downarrow t'$, which gives us the inductive hypotheses:

- $IH_1 -\ c \overset{\star}{\mapsto} \mathsf{True}$
- $IH_2 -\ t \overset{\star}{\mapsto} \mathsf{t'}$

Showing our overall goal:

$$\begin{aligned} \text{If } c\ t\ e &\overset{\star}{\mapsto} \text{If } \mathsf{True}\ t\ e && IH_1 \\ &\overset{\star}{\mapsto} t && (2') \\ &\overset{\star}{\mapsto} \mathsf{t'} && IH_2 \end{aligned}$$

Inductive Case ($from\ rule$ $\mathsf{If_2}$) We know that $c \Downarrow \mathsf{False}$ and $e \Downarrow e'$, which gives us the inductive hypotheses:

- $IH_1 -\ c \overset{\star}{\mapsto} \mathsf{False}$
- $IH_2 -\ e \overset{\star}{\mapsto} \mathsf{e'}$

Showing our overall goal:

$$\begin{aligned} \text{If } c\ t\ e &\overset{\star}{\mapsto} \text{If } \mathsf{False}\ t\ e && IH_1 \\ &\overset{\star}{\mapsto} e && (3') \\ &\overset{\star}{\mapsto} \mathsf{e'} && IH_2 \end{aligned}$$

Part C (15 marks)

1. Define a recursive *compilation function* $c : \mathcal{B} \mapsto \mathcal{L}$ which converts expressions in $\mathcal{B}$ to expressions in $\mathcal{L}$. (5 marks)

Solution:

$$c\ (\mathsf{True}) = \mathsf{True}$$

$$c\ (\mathsf{False}) = \mathsf{False}$$

$$c\ (\mathsf{Not}\ e) = \mathsf{If}\ e\ \mathsf{False}\ \mathsf{True}$$

$$c\ (\mathsf{And}\ e_1\ e_2) = \mathsf{If}\ e_1\ (\mathsf{If}\ e_2\ \mathsf{True}\ \mathsf{False})\ \mathsf{False}$$

2. Prove that for all e, $e \Downarrow v$ implies $c(e) \Downarrow v$, by rule induction on the assumption that $e \Downarrow v$. *(10 marks)*

Solution:

Base Case ($e = \mathsf{True}$) According to the Part B, $\mathsf{True} \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. We must show that $c\ (\mathsf{True}) \Downarrow \mathsf{True}$, which is true that $c\ (\mathsf{True}) = \mathsf{True} \Downarrow \mathsf{True}$.

Base Case ($e = \mathsf{False}$) According to the Part B, $\mathsf{False} \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that $c\ (\mathsf{False}) \Downarrow \mathsf{False}$, which is true that $c\ (\mathsf{False}) = \mathsf{False} \Downarrow \mathsf{False}$.

Inductive Case ($e = \mathsf{Not}\ e'$)

Assuming $e' = \mathsf{True}$, According to the Part A, $e = (\mathsf{Not\ True}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that $c\ (e) \Downarrow \mathsf{False}$, which is true that

$$c\ (e) = c\ (\mathsf{Not}\ e') = (\mathsf{If\ False\ False\ True}) \Downarrow \mathsf{False}$$

.

Assuming $e' = \mathsf{False}$, According to the Part A, $e = (\mathsf{Not\ False}) \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. We must show that $c\ (e) \Downarrow \mathsf{True}$, which is true that

$$c\ (e) = c\ (\mathsf{Not}\ e') = (\mathsf{If\ False\ False\ True}) \Downarrow \mathsf{True}$$

.

Inductive Case ($e = \mathsf{And}\ e_1\ e_2$)

$$c\ (e) = c\ (\mathsf{And}\ e_1\ e_2) = \mathsf{If}\ \mathsf{e}_1\ (\mathsf{If}\ e_2\ \mathsf{True}\ \mathsf{False})\ \mathsf{False}$$

Assuming $e_1 \Downarrow \mathsf{False}$, According to the Part A, $e = (\mathsf{And\ False}\ e_2) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that $c\ (e) \Downarrow \mathsf{False}$, which is true that

$$\dfrac{\dfrac{}{\mathsf{e}_1 \Downarrow \mathsf{False}}\mathsf{False} \quad \dfrac{}{\mathsf{False} \Downarrow \mathsf{False}}\mathsf{False}}{(\mathsf{If}\ \mathsf{e}_1\ (\mathsf{If}\ e_2\ \mathsf{True}\ \mathsf{False})\ \mathsf{False}) \Downarrow \mathsf{False}}\mathsf{If}_2$$

Assuming $e_1 \Downarrow \mathsf{True}, e_2 \Downarrow \mathsf{True}$, According to the Part A, $e = (\mathsf{And\ True\ True}) \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. We must show that $c\ (e) \Downarrow \mathsf{True}$, which is true that

$$\dfrac{\dfrac{}{\mathsf{e_1} \Downarrow \mathsf{True}}\mathsf{True} \quad \dfrac{\dfrac{}{\mathsf{e_2} \Downarrow \mathsf{True}}\mathsf{True} \quad \dfrac{}{\mathsf{True} \Downarrow \mathsf{True}}\mathsf{True}}{(\mathsf{If\ e_2\ True\ False}) \Downarrow \mathsf{True}}\mathsf{If_1}}{(\mathsf{If\ e_1\ (If\ e_2\ True\ False)\ False}) \Downarrow \mathsf{True}}\mathsf{If_1}$$

Assuming $e_1 \Downarrow \mathsf{True}, e_2 \Downarrow \mathsf{False}$, According to the Part A, $e = (\mathsf{And\ True\ False}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that $c\ (e) \Downarrow \mathsf{False}$, which is true that

$$\dfrac{\dfrac{}{\mathsf{e_1} \Downarrow \mathsf{True}}\mathsf{True} \quad \dfrac{\dfrac{}{\mathsf{e_2} \Downarrow \mathsf{False}}\mathsf{False} \quad \dfrac{}{\mathsf{False} \Downarrow \mathsf{False}}\mathsf{False}}{(\mathsf{If\ e_2\ True\ False}) \Downarrow \mathsf{False}}\mathsf{If_2}}{(\mathsf{If\ e_!\ (If\ e_2\ True\ False)\ False}) \Downarrow \mathsf{True}}\mathsf{If_1}$$

Part D (40 marks)

1. Here is a term in λ-calculus:

$$(\lambda n.\ \lambda f.\ \lambda x.\ (n\ f\ (f\ x)))\ (\lambda f.\ \lambda x.\ f\ x)$$

a) Fully β-reduce the above λ-term. Show all intermediate beta reduction steps. *(5 marks)*

Solution:

$$\begin{aligned} & (\lambda n.\ \lambda f.\ \lambda x.\ (n\ f\ (f\ x)))\ (\lambda f.\ \lambda x.\ f\ x) \\ \mapsto_\beta\ & (\lambda f.\ \lambda x.\ ((\lambda f.\ \lambda x.\ f\ x)\ f\ (f\ x))) \\ \mapsto_\beta\ & (\lambda f.\ \lambda x.\ ((\lambda x.\ f\ x)\ (f\ x))) \\ \mapsto_\beta\ & (\lambda f.\ \lambda x.\ f\ (f\ x)) \end{aligned}$$

b) Identify an η-reducible expression in the above (unreduced) term. *(5 marks)*

Solution:

$$\begin{aligned} & (\lambda n.\ \lambda f.\ \lambda x.\ (n\ f\ (f\ x)))\ (\lambda f.\ \lambda x.\ f\ x) \\ \mapsto_\eta\ & (\lambda n.\ \lambda f.\ \lambda x.\ (n\ f\ (f\ x)))\ (\lambda f.\ f) \end{aligned}$$

2. Recall that in λ-calculus, booleans can be encoded as binary functions that return one of their arguments:

$$\mathbf{T} \equiv (\lambda x.\ \lambda y.\ x)$$

$$\mathbf{F} \equiv (\lambda x.\ \lambda y.\ y)$$

Either via $\mathcal{L}$ or directly, define a function $d : B \to \lambda$ which converts expressions in $\mathcal{B}$ to λ-calculus. *(5 marks)*

Solution:

$$d\ (\mathsf{True}) = ((\lambda x.\ \lambda y.\ x)\ \mathsf{True}\ \mathsf{False})$$

$$d\ (\mathsf{False}) = ((\lambda x.\ \lambda y.\ y)\ \mathsf{True}\ \mathsf{False})$$

$$d\ (\mathsf{Not}\ e) = ((\lambda x.\ x\ \mathsf{False}\ \mathsf{True})\ d\ (e))$$

$$d\ (\mathsf{And}\ e_1\ e_2) = ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ d\ (e_1)\ d\ (e_2))$$

3. Prove that for all e such that $e \Downarrow v$ it holds that $d(e) \equiv_{\alpha\beta\eta} v'$, where v' is the λ-calculus encoding of v. *(10 marks)*

Solution:

Base Case ($e = \mathsf{True}$), According to the Part B, $\mathsf{True} \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. We must show that v' is the λ-calculus encoding of True, which is true that

$$\begin{aligned} d\ (\mathsf{True}) &= ((\lambda x.\ \lambda y.\ x)\ \mathsf{True}\ \mathsf{False}) \\ &\mapsto_\beta (\lambda y.\ \mathsf{True})\ \mathsf{False} \\ &\mapsto_\beta \mathsf{True} \end{aligned}$$

Base Case ($e = \mathsf{False}$) According to the Part B, $\mathsf{False} \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that v' is the λ-calculus encoding of False, which is true that

$$\begin{aligned} d\ (\mathsf{False}) &= ((\lambda x.\ \lambda y.\ y)\ \mathsf{True}\ \mathsf{False}) \\ &\mapsto_\beta (\lambda y.\ y)\ \mathsf{False} \\ &\mapsto_\beta \mathsf{False} \end{aligned}$$

Inductive Case ($e = \mathsf{Not}\ e'$)

- Assuming $e' = \mathsf{True}$, According to the Part A, $e = (\mathsf{Not\ True}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. Also according to the prove above, $d\ (\mathsf{True}) = \mathsf{True}$. We must show that v' is the λ-calculus encoding of False.

$$\begin{aligned} d\ (e) = d\ (\mathsf{Not}\ e') &= ((\lambda x.\ x\ \mathsf{False}\ \mathsf{True})\ d\ (e')) \\ &= ((\lambda x.\ x\ \mathsf{False}\ \mathsf{True})\ \mathsf{True}) \\ &\mapsto_\beta (\mathsf{True\ False\ True}) \end{aligned}$$

  ($\mathsf{True\ False\ True}$) write in if-then-else structure is $\mathsf{If\ True\ False\ True}$ which can encode to False.

- Assuming $e' = \mathsf{False}$, According to the Part A, $e = (\mathsf{Not\ False}) \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. Also according to the prove above, $d\ (\mathsf{False}) = \mathsf{False}$. We must show that v' is the λ-calculus encoding of True.

$$\begin{aligned} d\ (e) = d\ (\mathsf{Not}\ e') &= ((\lambda x.\ x\ \mathsf{False}\ \mathsf{True})\ d\ (e')) \\ &= ((\lambda x.\ x\ \mathsf{False}\ \mathsf{True})\ \mathsf{False}) \\ &\mapsto_\beta (\mathsf{False\ False\ True}) \end{aligned}$$

  ($\mathsf{False\ False\ True}$) write in if-then-else structure is $\mathsf{If\ False\ False\ True}$ which can encode to True.

Inductive Case ($e = \mathsf{And}\ e_1\ e_2$)

According to the prove above

$$d\ (\mathsf{True}) = \mathsf{True} \quad d\ (\mathsf{False}) = \mathsf{False}$$

- Assuming $e_1 \Downarrow \mathsf{True}, e_2 \Downarrow \mathsf{True}$, According to the Part A, $e = (\mathsf{And\ True\ True}) \Downarrow \mathsf{True}$, therefore, $v = \mathsf{True}$. We must show that v' is the λ-calculus encoding of True.

$$\begin{aligned} d\ (e) = d\ (\mathsf{And}\ e_1\ e_2) &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ d\ (e_1)\ d\ (e_2)) \\ &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ \mathsf{True}\ \mathsf{True}) \\ &\mapsto_\beta ((\lambda y.\ \mathsf{True}\ y\ \mathsf{False})\ \mathsf{True}) \\ &\mapsto_\beta (\mathsf{True\ True\ False})) \end{aligned}$$

(True True False) write in if-then-else structure is If True True False which can encode to True.

- Assuming $e_1 \Downarrow \mathsf{True}, e_2 \Downarrow \mathsf{False}$, According to the Part A, $e = (\mathsf{And\ True\ False}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that v' is the λ-calculus encoding of False.

$$\begin{aligned} d\ (e) = d\ (\mathsf{And}\ e_1\ e_2) &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ d\ (e_1)\ d\ (e_2)) \\ &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ \mathsf{True}\ \mathsf{False}) \\ &\mapsto_\beta ((\lambda y.\ \mathsf{True}\ y\ \mathsf{False})\ \mathsf{False}) \\ &\mapsto_\beta (\mathsf{True\ False\ False})) \end{aligned}$$

(True False False) write in if-then-else structure is If True False False which can encode to False.

- Assuming $e_1 \Downarrow \mathsf{False}, e_2 \Downarrow \mathsf{True}$, According to the Part A, $e = (\mathsf{And\ False\ True}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that v' is the λ-calculus encoding of False.

$$\begin{aligned} d\ (e) = d\ (\mathsf{And}\ e_1\ e_2) &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ d\ (e_1)\ d\ (e_2)) \\ &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ \mathsf{False}\ \mathsf{True}) \\ &\mapsto_\beta ((\lambda y.\ \mathsf{False}\ y\ \mathsf{False})\ \mathsf{True}) \\ &\mapsto_\beta (\mathsf{False\ True\ False})) \end{aligned}$$

(False True False) write in if-then-else structure is If False True False which can encode to False.

- Assuming $e_1 \Downarrow \mathsf{False}, e_2 \Downarrow \mathsf{False}$, According to the Part A, $e = (\mathsf{And\ False\ False}) \Downarrow \mathsf{False}$, therefore, $v = \mathsf{False}$. We must show that v' is the λ-calculus encoding of False.

$$\begin{aligned} d\ (e) = d\ (\mathsf{And}\ e_1\ e_2) &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ d\ (e_1)\ d\ (e_2)) \\ &= ((\lambda x.\ \lambda y.\ x\ y\ \mathsf{False})\ \mathsf{False}\ \mathsf{False}) \\ &\mapsto_\beta ((\lambda y.\ \mathsf{False}\ y\ \mathsf{False})\ \mathsf{False}) \\ &\mapsto_\beta (\mathsf{False\ False\ False})) \end{aligned}$$

(False False False) write in if-then-else structure is If False False False which can encode to False.

4. Suppose we added *unary local function definitions* to our language $\mathcal{P}$. Here's an example in concrete syntax:

```
let
  g(x) = ¬x
in
  g(True)
end
```

We limit ourselves to non-recursive bindings (meaning functions can't call themselves), and first-order functions (meaning functions require boolean arguments).

a) Extend the *abstract* syntax for $\mathcal{B}$ from question A.3 so that it supports the features used in the above example. Use first-order abstract syntax with explicit strings. You don't have to extend the parsing relation. *(5 marks)*

Solution:

$\mathcal{B}$ ::= let $\mathcal{S}(\mathcal{S}) = \mathcal{B}$ in $\mathcal{B}$ end | Not $\mathcal{B}$ | And $\mathcal{B}$ $\mathcal{B}$ | True | False | $\mathcal{S}$

$\mathcal{S}$ :: *string*

b) Define a scope-checking judgement, similar to the **ok** judgement from the lectures. It should check (a) that all names of variables and functions are used only within their scopes; and (b) that names used in variable (or function) position are indeed the names of variables (or functions). Hence, the following expressions should both be rejected:

```
let                let                let
  f(x) = ¬x          f(x) = ¬x          f(x) = x(True)
in                 in                 in
  f(x)               f(f)               f(False)
end                end                end
```

The following are examples of things that should be accepted: nullary definitions, nested definitions, and shadowed definitions.

```
let                          let
  f(x) =                       f(x) = x(True)
    let                      in
      g(y) = ¬x ∧ y            let
    in                           f(x) = f(x)
      g(x) ∧ ¬g(x)           in
    end                        f(True)
in                           end
  f(False)                 end
end
```

Note that the latter example is not a recursive call. *(10 marks)*

Solution:

Γ means context. $\quad \Gamma \vdash$ True **ok** $\quad \Gamma \vdash$ False **ok**

$$\frac{s\ \mathit{boundv} \in \Gamma}{\Gamma \vdash s\ \mathbf{ok}} \qquad \frac{s\ \mathit{boundf} \in \Gamma \quad \Gamma \vdash e\ \mathbf{ok}}{\Gamma \vdash s(e)\ \mathbf{ok}}$$

$$\frac{\Gamma \vdash e\ \mathbf{ok}}{\Gamma \vdash \mathsf{Not}\ e\ \mathbf{ok}} \qquad \frac{\Gamma \vdash e_1\ \mathbf{ok} \quad \Gamma \vdash e_2\ \mathbf{ok}}{\Gamma \vdash \mathsf{And}\ e_1\ e_2\ \mathbf{ok}}$$

$$\frac{s_2\ \mathit{boundv},\ \Gamma \vdash e_1\ \mathbf{ok} \qquad s_1\ \mathit{boundf},\ \Gamma \vdash e_2\ \mathbf{ok}}{\Gamma \vdash \mathsf{let}\ s_1(s_2)\ =\ e_1\ \mathsf{in}\ e_2\ \mathsf{end}\ \mathbf{ok}}$$

Late Penalty

You may submit up to five days (120 hours) late. Each day of lateness corresponds to a 5% reduction of your total mark. For example, if your assignment is worth 88% and you submit it two days late, you get 78%. If you submit it more than five days late, you get 0%.

Course staff cannot grant assignment extensions—if you need an extensions, you have to apply for special consideration through the standard procedure. More information here: https://www.student.unsw.edu.au/special-consideration

Plagiarism

Many students do not appear to understand what is regarded as plagiarism. This is no defense. Before submitting any work you should read and understand the UNSW plagiarism policy https://student.unsw.edu.au/plagiarism.

All work submitted for assessment must be entirely your own work. We regard unacknowledged copying of material, in whole or part, as an extremely serious offence. In this course submission of any work derived from another person, or solely or jointly written by and or with someone else, without clear and explicit acknowledgement, will be severely punished and may result in automatic failure for the course and a mark of zero for the course. Note this includes including unreferenced work from books, the internet, etc.

Do not provide or show your assessable work to any other person. Allowing another student to copy from you will, at the very least, result in zero for that assessment. If you knowingly provide or show your assessment work to another person for any reason, and work derived from it is subsequently submitted, you will be penalized, even if the work was submitted without your knowledge or consent. This will apply even if your work is submitted by a third party unknown to you. You should keep your work private until submissions have closed.

If you are unsure about whether certain activities would constitute plagiarism ask us before engaging in them!