Normal forms

Terms in normal form should denote values or results of a computation. A essential condition for something to be deemed a value is that it no longer be reducible. Recall our judgments \( N \text{ nf} \) and \( R \text{ neutral} \) from class for normal forms and neutral expressions, respectively. They were inductively defined by the rules:

\[
\begin{align*}
\frac{N \text{ nf}}{\lambda x. N \text{ nf}} & \quad \text{(NF-LAM)} \\
\frac{R \text{ neutral}}{R \text{ nf}} & \quad \text{(NF-NE)} \\
\frac{x \text{ neutral}}{x} & \quad \text{(NE-VAR)} \\
\frac{R \text{ neutral}}{R N \text{ neutral}} & \quad \text{(NE-APP)}
\end{align*}
\]

For your reference, recall that the \( \beta \)-reduction relation \( \rightarrow_\beta \) is inductively defined by the rules:

\[
\begin{align*}
\frac{e \rightarrow_\beta e'}{\lambda x.e \rightarrow_\beta \lambda x.e'} & \quad \text{(B-LAM)} \\
\frac{\lambda x.e_1 e_2 \rightarrow_\beta [e_2/x]e_1}{(\lambda x.e_1)e_2 \rightarrow_\beta [e_2/x]e_1} & \quad \text{(B-RED)} \\
\frac{e_1 \rightarrow_\beta e'_1}{e_1 e_2 \rightarrow_\beta e'_1 e_2} & \quad \text{(B-L)} \\
\frac{e_2 \rightarrow_\beta e'_2}{e_1 e_2 \rightarrow_\beta e_1 e'_2} & \quad \text{(B-R)}
\end{align*}
\]

We may write \( e \rightarrow_\beta \) to mean there exists some \( e' \) such that \( e \rightarrow_\beta e' \). When no such \( e' \) exists, we write \( e \not\rightarrow_\beta \).
Task 1 (15 pt, WB). Show that for all $e$, either $e$ nf or $e \to_\beta$ (but not both).

Hint. Proceed by induction on the syntax of $e$. Split the case that $e$ is an application $e_1 e_2$ into the two following subcases: $e_1$ is $\lambda x.e_1'$, and $e_1$ is $x$ or an application.

Solution 1: We proceed by induction on the syntax of $e$.

Case $e$ is $x$. Then $x$ neutral by (NE-VAR), and we get $e$ nf by (NF-NE). There are no $\beta$-rules with a conclusion of the form $x \to_\beta e'$, so we conclude $x \not\to_\beta$ as desired.

Case $e$ is $\lambda x.e_x$. By the induction hypothesis on $e_x$, we get either $e_x$ nf or $e_x \to_\beta e'_x$ (but not both). In the first case, we get $\lambda x.e_x$ nf by (NF-LAM). Were it the case that $\lambda x.e_x \to_\beta e'$ for some $e'$, then by inversion via (B-LAM), we would have $e_x \to_\beta e'_x$, a contradiction. In the second case, we have $e \to_\beta \lambda x.e'_x$ by (B-LAM). Were it the case that $\lambda x.e_x$ nf, then by inversion via (NF-LAM), we would have $e_x$ nf, a contradiction.

Case $e$ is $e_1 e_2$, subcase $e_1$ is $\lambda x.e_1'$. Then $e \to_\beta [e_2/x]e_1'$ by (B-LAM). Suppose to the contrary $e$ nf, then by inversion via (NF-NE), $e$ neutral. By inversion via (NE-APP), this gives $\lambda x.e_1'$ neutral. By inversion on $\lambda x.e_1'$ neutral, we reach a contradiction (there are no rules with a conclusion of this form).

Case $e$ is $e_1 e_2$, subcases $e_1$ is $x$ or $e'_1 e''_1$. By the induction hypothesis, either $e_1$ nf or $e_1 \to_\beta$ (but not both), and either $e_2$ nf or $e_2 \to_\beta$ (but not both). We consider the four possible subsubcases:

1. $e_1$ nf and $e_2$ nf. By inversion on $e_1$ nf, we have that $e_1$ neutral. In this case, $e$ neutral by (NE-APP), so $e$ nf by (NF-NE). Their hypotheses contradict the induction hypothesis, so $e \not\to_\beta$.

2. $e_1$ nf and $e_2 \to_\beta e'_2$. Then $e \to_\beta e_1 e'_2$ by (B-R). Suppose to the contrary that $e$ nf, then by inversion via (NF-NE) we have $e$ neutral. By inversion again via (NE-APP), we have $e$ nf, contradicting the induction hypothesis.

3. $e_1 \to_\beta e'_1$, and $e_2$ nf or $e_2 \to_\beta e'_2$. Then $e \to_\beta e'_1 e'_2$ by (B-L). Suppose to the contrary that $e$ nf, then by inversion via (NF-NE) we have $e$ neutral. By inversion via (NF-APP), we get $e_1$ neutral. Applying (NF-NE), we get $e_1$ nf, contradicting the induction hypothesis. □
Bidirectional type checking

Our goal is to show that type checking is decidable for normal forms in the simply-typed \( \lambda \)-calculus. To do so, we will use a bidirectional type checking algorithm. We describe this algorithm using inference rules whose judgments are interpreted as having “inputs” and “outputs”.

The algorithm uses the following two new judgments:

- \( \Gamma \vdash N \leftarrow \tau \) says that normal form \( N \) checks against type \( \tau \) under \( \Gamma \),
- \( \Gamma \vdash R \Rightarrow \tau \) says that neutral expression \( R \) synthesizes type \( \tau \) under \( \Gamma \).

When using the synthesis judgments below, we read \( \Gamma \) and \( R \) as inputs and \( \tau \) as an output. In the checking judgments, \( \Gamma \), \( N \), and \( \tau \) should all be seen as inputs.

For legibility, we have colour-coded inputs and outputs below.

Throughout, we assume our contexts are well-formed, that is, that \( \Gamma \) is finite list \( x_1 : \tau_1, \ldots, x_n : \tau_n \) with \( n \geq 0 \) and \( x_i \neq x_j \) for \( 1 \leq i \neq j \leq n \). In this case, let \( \text{dom}(\Gamma) = \{ x_1, \ldots, x_n \} \). We assume throughout the judgments are well-formed, that is, that \( \text{fv}(e) \subseteq \text{dom}(\Gamma) \).

The algorithm is inductively defined by the following rules:

\[
\begin{align*}
\Gamma \vdash x \Rightarrow \tau & \quad \text{(VAR)} \\
\Gamma, x : \tau_1 \vdash N \leftarrow \tau_2 & \quad \text{(LAM)} \\
\Gamma \vdash R \Rightarrow \tau_1 \rightarrow \tau_2 & \quad \text{(APP)} \\
\Gamma \vdash RN \Rightarrow \tau_2 & \quad \text{(CS)}
\end{align*}
\]

To better understand these judgments, we recommend working through a few examples on your own. To help you get started, we have worked one out for you. Abbreviate \( f : \alpha \rightarrow \alpha \rightarrow \alpha \) by \( \Gamma \). To type check \( \cdot \vdash \lambda f.\lambda y.fy \leftarrow (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha \), the algorithm performs the following derivation, bottom-up. When it reaches the (APP) rule, it applies the (VAR) rule to \( f \) to synthesize the type \( \alpha \rightarrow \alpha \) (an output). From this, we know the type of \( y \) must be \( \alpha \) if we are to apply \( f \) to \( y \), so we check \( y \) against \( \alpha \) in \( \Gamma \). We succeed in completing the derivation and conclude that the type is correct.

\[
\begin{align*}
f : \alpha \rightarrow \alpha \in \Gamma & \quad \text{(VAR)} \\
\Gamma \vdash f \Rightarrow \alpha & \quad \text{(VAR)} \\
\Gamma \vdash y \Rightarrow \alpha & \quad \text{(VAR)} \\
\Gamma \vdash fy \Leftarrow \alpha & \quad \text{(CS)} \\
\Gamma \vdash fy \Rightarrow \alpha & \quad \text{(CS)} \\
\Gamma \vdash \lambda y.fy \leftarrow (\alpha \rightarrow \alpha) \rightarrow \alpha & \quad \text{(LAM)} \\
\Gamma \vdash \lambda f.\lambda y.fy \leftarrow (\alpha \rightarrow \alpha) \rightarrow \alpha & \quad \text{(LAM)}
\end{align*}
\]

\(^1\)Showing decidability for reducible terms requires adding type annotations in key places and may be covered in class at a later date.
As a programmer, it is often useful to get an “error message” when we have a type mismatch. For this reason, we introduce the following two judgments:

- \( \Gamma \vdash N \not\equiv \tau \) says that \( N \) does not check with type \( \tau \) under \( \Gamma \),
- \( \Gamma \vdash R \not\Rightarrow \) says that \( R \) does not synthesize a type under \( \Gamma \).

They are inductively defined by the following rules:

\[
\begin{align*}
&\frac{x \notin \text{dom}(\Gamma)}{\Gamma \vdash x \not\Rightarrow} \quad \text{(VAR)} \\
&\frac{\Gamma, x : \tau_1 \vdash N \not\equiv \tau_2}{\Gamma \vdash \lambda x.\, N \not\equiv \tau_1 \to \tau_2} \quad \text{(LAM)} \\
&\frac{\Gamma \vdash \lambda x.\, N \not\equiv c}{\Gamma \vdash \lambda x.\, N \not\equiv c} \quad \text{(LAMC)} \\
&\frac{\Gamma \vdash R \not\Rightarrow}{\Gamma \vdash R \not\Rightarrow} \quad \text{(APP)} \\
&\frac{\Gamma \vdash R \Rightarrow \tau \not\equiv \Gamma \vdash R \Rightarrow \tau}{\Gamma \vdash R \not\Rightarrow} \quad \text{(APP')} \\
&\frac{\Gamma \vdash R \Rightarrow \tau_1 \to \tau_2 \quad \Gamma \vdash N \not\equiv \tau_1}{\Gamma \vdash RN \not\Rightarrow} \quad \text{(APP'')} \\
&\frac{\Gamma \vdash R \Rightarrow \tau' \quad \tau \not\equiv \tau'}{\Gamma \vdash R \not\Rightarrow} \quad \text{(CS)} \\
&\frac{\Gamma \vdash R \not\Rightarrow}{\Gamma \vdash R \not\Rightarrow} \quad \text{(CS')} \\
\end{align*}
\]

Task 2 (5 points). If the following proposition is true, prove it, otherwise, find a counter-example:

If \( \Gamma \vdash N \leftarrow \tau \) and \( \Gamma \vdash N \leftarrow \tau' \), then \( \tau = \tau' \).

Solution 2: The proposition is false. The rules (LAM) and (VAR) give us that all for all types \( \tau \), \( \Gamma \vdash \lambda x.\, x \leftarrow \tau \to \tau \).

\[
\square
\]

Theorem 1 (Decidability of bidirectional type checking).

1. Given \( \Gamma, N, \) and \( \tau \), either \( \Gamma \vdash N \leftarrow \tau \) or \( \Gamma \vdash N \not\equiv \tau \) (but not both).

2. Given \( \Gamma \) and \( R, \) either there exists a unique \( \tau \) such that \( \Gamma \vdash R \Rightarrow \tau \) or \( \Gamma \vdash R \not\Rightarrow \) (but not both).

The proof of this theorem uses a mutual induction on the structures of \( N \) and \( R \). We need a mutual induction because normal forms can contain neutral terms, e.g., \( N = x \), and neutral terms can contain normal forms, e.g., \( R = R' N \). This induction requires care when we reach the case of \( N \) is \( R \) and consider the rule (CS), because we apply the induction hypothesis for synthesis to the same term whose type we are checking. This does not affect the inductive character of the proof or introduce any circularity in the argument, because whenever we apply the induction hypothesis for checkable terms in synthesis cases, we only do so to strictly smaller subterms.
Proof. By mutual induction on the structure of $N$ and $R$.

Case $N$ is $\lambda x. N$ and $\tau$ is some $c$. There are no rules with a conclusion of the form $\Gamma \vdash \lambda x. N \not\leftarrow c$. We conclude $\Gamma \vdash \lambda x. N \not\leftrightarrow c$ by (#LAMC).

Case $N$ is $\lambda x. N$ and $\tau$ is of the form $\tau_1 \rightarrow \tau_2$. By task[3]

Case $N$ is $R$. The only rules forming a judgment of the form $\Gamma \vdash R \leftarrow \tau$ or $\Gamma \vdash R \not\leftrightarrow \tau$ are (CS), (#CS), and (#CS'). Applying the induction hypothesis to $R$, either there exists a unique $\tau'$ such that $\Gamma \vdash R \Rightarrow \tau'$ or $\Gamma \vdash R \not\Rightarrow$ (but not both). Assume first there exists a $\tau'$, then (#CS') is not applicable. If $\tau = \tau'$, then (#CS) is not applicable and we are done by (CS). If $\tau \neq \tau'$, then (CS) is not applicable and we are done by (#CS). Now assume $\Gamma \vdash R \not\Rightarrow$. Then (CS) and (#CS) are not applicable and we are done by (#CS').

Case $R$ is $x$. By task[3]

Case $R$ is $RN$. The only rules with a conclusion of the form $\Gamma \vdash RN \Rightarrow \tau$ or $\Gamma \vdash RN \not\Rightarrow$ are (APP), (#APP), (#APP'), and (#APP''). Applying the induction hypothesis to $R$, either there exists a unique $\tau'$ such that $\Gamma \vdash R \Rightarrow \tau'$ or $\Gamma \vdash R \not\Rightarrow$ (but not both). If $\Gamma \vdash R \not\Rightarrow$, then the only applicable rule is (#APP), by which $\Gamma \vdash RN \not\Rightarrow$. Now assume there exists a $\tau'$. If $\tau' = c$, then the only applicable rule is (#APP'), by which $\Gamma \vdash RN \not\Rightarrow$. Now assume $\tau' = \tau_1 \rightarrow \tau_2$. By the induction hypothesis on $N$, either $\Gamma \vdash N \not\leftrightarrow \tau_1$ or $\Gamma \vdash N \not\leftrightarrow \tau_2$ (but not both). The only applicable rules in these two cases are (APP) and (#APP''), respectively, by which $\Gamma \vdash RN \Rightarrow \tau_2$ or $\Gamma \vdash RN \not\Rightarrow$ (but not both).

\[\square\]

Task 3 (10 points, WB). Complete the following cases in the proof of decidability of bidirectional type checking.

1. $N$ is $\lambda x. N$ and $\tau$ is of the form $\tau_1 \rightarrow \tau_2$, and

2. $R$ is $x$.

Solution 3: Case $N$ is $\lambda x. N$ and $\tau$ is of the form $\tau_1 \rightarrow \tau_2$. The only rules with conclusions of the form $\Gamma \vdash \lambda x. N \not\leftarrow \tau_1 \rightarrow \tau_2$ and $\Gamma \vdash \lambda x. N \not\leftrightarrow \tau_1 \rightarrow \tau_2$ are (LAM) and (#LAM). By the induction hypothesis on $N$, either $\Gamma, x : \tau_1 \vdash N \not\leftarrow \tau_2$ or $\Gamma, x : \tau_1 \vdash N \not\leftrightarrow \tau_2$ (but not both). In the first case, (#LAM) is not applicable. (In more detail: suppose there were a derivation of $\Gamma \vdash \lambda x. N \not\leftrightarrow \tau_1 \rightarrow \tau_2$, then it must have been by (#LAM). By inversion, we get $\Gamma, x : \tau_1 \vdash N \not\leftrightarrow \tau_2$, a contradiction of the “but not both” portion of the induction hypothesis.) We conclude $\Gamma \vdash \lambda x. N \not\leftarrow \tau_1 \rightarrow \tau_2$ by (LAM). In the second case, (LAM) is not applicable and we are done by (#LAM).

Case $R$ is $x$. The only rules with a conclusion of the form $\Gamma \vdash x \Rightarrow \tau$ or $\Gamma \vdash x \not\Rightarrow$ are the mutually exclusive rules (VAR) and (#VAR). Because $\Gamma$ is finite, we can decide if $x \in \text{dom}(\Gamma)$. If $x \in \text{dom}(\Gamma)$, then we have $\Gamma \vdash x \Rightarrow \tau$ by (VAR), and the uniqueness of the type is due to the well-formedness of $\Gamma$. If $x \notin \text{dom}(\Gamma)$, then we have $\Gamma \vdash x \not\Rightarrow$ by (#VAR). \[\square\]
The Church-Rosser Theorem

The Church-Rosser theorem says that β-reduction for the untyped λ-calculus is confluent. A relation → is confluent if whenever \( e \rightarrow^* e_1 \) and \( e \rightarrow^* e_2 \), there exists an \( e_3 \) such that \( e_1 \rightarrow^* e_3 \) and \( e_2 \rightarrow^* e_3 \):

\[
\begin{array}{c}
e_1 \\
\downarrow \quad \rightarrow^* \\
e_2 \\
\downarrow \quad \rightarrow^* \\
e_3 \\
\end{array}
\]

This is an instance of the diamond property. We say that a relation → satisfies the diamond property if whenever \( e \rightarrow e_1 \) and \( e \rightarrow e_2 \), then there exists an \( e_3 \) such that \( e_1 \rightarrow e_3 \) and \( e_2 \rightarrow e_3 \):

\[
\begin{array}{c}
e_1 \\
\downarrow \\
e_2 \\
\downarrow \\
e_3 \\
\end{array}
\]

We have used the Church-Rosser theorem several times in class, and we have received questions on how to prove the theorem. In this section, we have broken up a proof by Tait and Martin-Löf into manageable pieces. The high-level overview of the proof is as follows.

1. We define a new reduction relation \( \rightarrow_l \) that simultaneously reduces a function and its argument.
2. We show that \( \rightarrow_l \) is confluent.
3. We show that if a reduction is confluent, then so is its reflexive, transitive closure.
4. We show that \( \rightarrow_l^* = \rightarrow_\beta^* \).

We conclude that \( \rightarrow_\beta^* \) is confluent, i.e., the Church-Rosser theorem holds.

Let \( \rightarrow_l \) be the relation on \( \lambda \)-terms inductively defined by the following rules:

\[
\begin{align*}
e \rightarrow_l e & \quad \text{(L-REFL)} \\
\lambda x.e \rightarrow_l \lambda x.e' & \quad \text{(L-LAM)} \\
e_1 \rightarrow_l e_1' & \quad \text{e_2 \rightarrow_l e_2'} (\text{L-APP}) \\
\end{align*}
\]

\[
\begin{align*}
e_1 \rightarrow_l e_1' & \quad e_2 \rightarrow_l e_2' \quad (\text{L-RED}) \\
\end{align*}
\]
**Task 4** (5 points, WB). Prove that:

1. \( \lambda x.e_x \to_l e' \) implies \( e' \equiv \lambda x.e'_x \) with \( e_x \to_l e'_x \),

2. \( e_1 e_2 \to_l e' \) implies either
   - \( e' \equiv e'_1 e'_2 \) with \( e_i \to_l e'_i \) for \( i = 1, 2 \), or
   - \( e_1 \equiv \lambda x.e_x, e' \equiv [e'_2/x]e'_x, e_x \to_l e'_x, \) and \( e_2 \to_l e'_2 \).

**Solution 4:** By cases on the derivation of \( e \to_l e \).

Case (L-REFL). In the first part, \( \lambda x.e \to_l \lambda x.e \) and \( e \to_l e \) by (L-REFL). In the second part, if \( e_1 e_2 \to_l e' \) then \( e' = e_1 e_2 \), and by (L-REFL), \( e_1 \to_l e_1 \) and \( e_2 \to_l e_2 \).

Case (L-LAM). The first part is immediate. The second part is impossible.

Case (L-APP). The first part is impossible. The second part is immediate.

Case (L-RED). The first part is impossible. The second part is immediate. \( \square \)

**Theorem 2.** The relation \( \to_l \) is confluent.

**Proof (sketch).** By task 5, it is sufficient to show that \( \to_l \) satisfies the diamond property. We must show that for all \( e_0, e_1, \) and \( e_2 \) such that \( e_0 \to_l e_1 \) and \( e_0 \to_l e_2 \), there exists an \( e_3 \) such that \( e_1 \to_l e_3 \) and \( e_2 \to_l e_3 \). The proof is by induction on \( e \to_l e_1 \) to show that for all \( e \to_l e_2 \) there is an \( e_3 \). \( \square \)

**Transitive closure of confluent relations**

The reflexive, transitive closure \( \to^* \) of a relation \( \to \) is inductively defined by the rules:

\[
\frac{e \to^* e}{e \to^* e} \quad \frac{e_1 \to e_2 \quad e_2 \to^* e_3}{e_1 \to^* e_3} \quad \text{(T-STEP)}
\]

**Task 5** (5 points, WB). Show that \( \to^* \) is actually transitive, i.e., that if \( e_1 \to^* e_2 \) and \( e_2 \to^* e_3 \), then \( e_1 \to^* e_3 \). Conclude that \( (\to^*)^* = \to^* \).

**Solution 5:** We proceed by induction on \( e_1 \to^* e_2 \).

Case (T-REFL). Then \( e_1 \equiv e_2 \) and \( e_2 \to^* e_3 \) imply \( e_1 \to^* e_3 \).

Case (T-STEP). Then \( e_1 \to e'_1 \) and \( e'_1 \to^* e_2 \) for some \( e'_1 \). By the induction hypothesis on \( e'_1 \to^* e_2 \) and \( e_2 \to^* e_3 \), we get \( e'_1 \to^* e_3 \). By (T-STEP) on \( e_1 \to e'_1 \) and \( e'_1 \to^* e_3 \), we get \( e_1 \to^* e_3 \).

We show that \( (\to^*)^* = \to^* \), i.e., that \( (\to^*)^* \) is actually a closure operation. First observe that for any relation \( \to \), if \( e \to e' \), then \( e \to^* e' \) by (T-STEP) and (T-REFL) applied to \( e' \). So \( e \to^* e' \) implies that \( e \to^* e' \). To show the converse, we proceed by induction on the derivation of \( e \to^* e' \). The case (T-REFL) is immediate by (T-REFL). The case (T-STEP) is immediate by the induction hypothesis and the transitivity of \( \to^* \). \( \square \)
**Task 6** (10 points, WB). Show that if \( \rightarrow \) satisfies the diamond property, then so does its reflexive, transitive closure \( \rightarrow^* \).

**Hint.** Prove and use the “strip lemma”: if \( e \rightarrow e_1 \) and \( e \rightarrow^* e_2 \), then there exists an \( e_3 \) such that \( e_1 \rightarrow^* e_3 \) and \( e_2 \rightarrow e_3 \). Pictorially, this amounts to showing:

\[
\begin{array}{c}
e \\ \downarrow \text{lemma} \\ \downarrow * \\ e_2 \end{array} \quad \Rightarrow \quad \begin{array}{c}
e_1 \\ \downarrow \text{lemma} \\ \downarrow * \\ e_3.
\end{array}
\]

Then, “paste” these “strips” together to get the big confluence rectangle.

**Solution 6:** We show the lemma: if \( e \rightarrow e_1 \) and \( e \rightarrow^* e_2 \), then there exists an \( e_3 \) such that \( e_1 \rightarrow^* e_3 \) and \( e_2 \rightarrow e_3 \). Pictorially, we want to show:

\[
\begin{array}{c}
e \\ \downarrow \text{lemma} \\ \downarrow * \\ e_2 \end{array} \quad \Rightarrow \quad \begin{array}{c}
e_1 \\ \downarrow \text{lemma} \\ \downarrow * \\ e_3.
\end{array}
\]

We proceed by induction on \( e \rightarrow^* e_2 \).

**Case (T-REFL).** Then \( e \equiv e_2 \). Take \( e_3 = e_1 \). Then \( e_1 \rightarrow^* e_3 \) by (T-REFL) and \( e_2 \rightarrow e_3 \) by assumption.

**Case (T-STEP).** Then \( e \rightarrow e'_2 \rightarrow^* e_2 \) for some \( e'_2 \). Because \( \rightarrow \) satisfies the diamond property and \( e \rightarrow e_1 \) and \( e \rightarrow e'_2 \), there exists an \( e'_3 \) such that \( e_1 \rightarrow e'_3 \) and \( e'_2 \rightarrow e_3 \). By applying the induction hypothesis to \( e'_2 \rightarrow^* e_2 \) with \( e'_2 \rightarrow e'_3 \), we get an \( e_3 \) such that \( e_2 \rightarrow e_3 \) and \( e'_3 \rightarrow^* e_3 \). We get \( e_1 \rightarrow^* e_3 \) by (T-STEP) with \( e_1 \rightarrow e'_3 \) and \( e'_3 \rightarrow^* e_3 \). This case is captured by the diagram:

\[
\begin{array}{c}
e \\ \downarrow \text{Diamond} \quad \downarrow \text{IH} \\ e_1 \end{array} \quad \Rightarrow \quad \begin{array}{c}
e'_{e_2} \\ \downarrow \text{IH} \\ e_2 \\

\text{IH}_{e'_2 \rightarrow^* e_2} \\
\downarrow \\
\begin{array}{c}e_3 \\ \downarrow \text{IH} \\
\downarrow \\
e_3.
\end{array}
\end{array}
\]

This concludes the proof of the strip lemma.

We now show the main result. We proceed by induction on \( e \rightarrow^* e_2 \) to show that if \( e \rightarrow^* e_1 \) and \( e \rightarrow^* e_2 \), then there exists an \( e_3 \) such that \( e_1 \rightarrow^* e_3 \) and \( e_2 \rightarrow^* e_3 \).

**Case (T-REFL).** Then \( e \equiv e_2 \). Take \( e_3 = e_1 \). Then \( e_1 \rightarrow^* e_3 \) by (T-REFL) and \( e_2 \rightarrow^* e_3 \) by assumption.

**Case (T-STEP).** Then \( e \rightarrow e'_2 \rightarrow^* e_2 \) for some \( e'_2 \). By applying the lemma to \( e \rightarrow e'_2 \) and \( e \rightarrow^* e_1 \), there exists an \( e'_3 \) such that \( e_1 \rightarrow e'_3 \) and \( e'_2 \rightarrow^* e_3 \). By
Applying the induction hypothesis to $e_2 \rightarrow^* e_2$ with $e'_2 \rightarrow^* e'_3$, we get an $e_3$ such that $e_2 \rightarrow^* e_3$ and $e'_3 \rightarrow^* e_3$. We get $e_1 \rightarrow^* e_3$ by (T-Step) with $e_1 \rightarrow e'_3$ and $e'_3 \rightarrow^* e_3$. This case is captured by the diagram:

$$
\begin{array}{c}
\quad e_2 \rightarrow^* e_2' \\
\quad \Downarrow \text{lemma} \quad \Downarrow \text{IH} e_2' \rightarrow^* e_2' \\
\quad \Downarrow^* \quad \Downarrow^* \quad \Downarrow^*
\end{array}
\quad
\begin{array}{c}
e_1 \rightarrow e_3' \rightarrow^* e_3.
\end{array}
$$

\[\square\]

**Concluding Church-Rosser**

**Task 7** (10 points, WB). Show $\rightarrow^*_{\beta}$ is the reflexive, transitive closure of $\rightarrow_{\beta}$. To do so, (informally) prove that:

1. if $e \rightarrow_{\beta} e'$, then $e \rightarrow_{\iota} e'$,
2. if $e \rightarrow_{\iota} e'$, then $e \rightarrow^*_{\beta} e'$.

Then explain why it follows that $\rightarrow^*_{\iota} = \rightarrow^*_{\beta}$, i.e., $e \rightarrow^*_{\iota} e'$ if and only if $e \rightarrow^*_{\beta} e'$.

**Hint.** You will need to use task[5]

**Solution 7**: We did not require you to give all of the details for full credit. We did, however, require you to give at least an outline of a proof indicating which parts were by induction, etc. If you would like to see a proof with all of the details, please ask.

We show by induction on $\rightarrow_{\beta}$ that if $e \rightarrow_{\beta} e'$, then $e \rightarrow_{\iota} e'$.

**Case (B-LAM).** We can step by (L-LAM) and the induction hypothesis.

**Cases (B-L) and (B-R).** Use (L-REFL) to fix the appropriate hypothesis and apply (L-APP) and the induction hypothesis.

**Case (B-RED).** Use (L-REFL) to fix the hypotheses and apply (L-RED).

Next, we show by induction on $\rightarrow_{\iota}$ that if $e \rightarrow_{\iota} e'$, then $e \rightarrow^*_{\beta} e'$.

**Case (L-REFL).** We are done by the reflexivity of $\rightarrow^*_{\beta}$.

**Case (L-LAM), where $\lambda x.e_x \rightarrow_{\iota} \lambda x.e'_x$ with $e_x \rightarrow_{\iota} e'_x$.** Applying the induction hypothesis to $e_x \rightarrow_{\iota} e'_x$ gives $e_x \rightarrow^*_{\beta} e'_x$. Induction on $e_x \rightarrow_{\iota} e'_x$ gives that $\lambda x.e_x \rightarrow^*_{\beta} \lambda x.e'_x$.

**Case (L-APP), where $e \equiv e_1 e_2 \rightarrow_{\iota} e'_1 e'_2$ with $e_i \rightarrow_{\iota} e'_i$ for $i = 1, 2$.** Apply the induction hypothesis to the two premises to get $e_1 \rightarrow^*_{\beta} e'_1$ and $e_2 \rightarrow^*_{\beta} e'_2$. We get $e \rightarrow^*_{\beta} e'$ by induction on $e_1 \rightarrow_{\beta} e'_1$.

**Case (L-RED), where $e \equiv (\lambda x.e_1) e_2 \rightarrow_{\iota} [e'_2/x]e'_1$ with $e_i \rightarrow_{\iota} e'_i$ for $i = 1, 2$.** Apply the induction hypothesis to the two premises to get $e_1 \rightarrow^*_{\beta} e'_1$ and $e_2 \rightarrow^*_{\beta} e'_2$. Induction on $e_1 \rightarrow^*_{\beta} e'_1$ gives $(\lambda x.e_1) e_2 \rightarrow_{\beta} (\lambda x.e'_1) e'_2$. By (B-RED), we get $(\lambda x.e'_1) e'_2 \rightarrow_{\beta} [e'_2/x]e'_1$, so $(\lambda x.e'_1) e'_2 \rightarrow^*_{\beta} [e'_2/x] e'_1$. By transitivity (task[5]), $e \rightarrow_{\beta} e'$.  

9
We show that $\to_1^* = \to_2^*$. Let $\to_1 \subseteq \to_2$ mean $e \to_1 e'$ implies $e \to_2 e'$. We can show by induction that for all $\to_1$ and $\to_2$, if $\to_1 \subseteq \to_2$, then $\to_1^* \subseteq \to_2^*$. So $\to_\beta^* \subseteq \to_t^*$ because $\to_\beta \subseteq \to_t$. But $\to_l \subseteq \to_\beta^*$ implies $\to_t^* \subseteq (\to_\beta^*)^*$, and $(\to_\beta^*)^* = \to_\beta^*$ by task 5. We conclude that $\to_t^* = \to_\beta^*$.