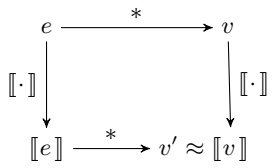
Both the CBV and CBN λ-calculus are *deterministic* reduction strategies in the sense that there is at most one reduction that is enabled in any term. When an expression e in a language is evaluated in a deterministic system, one of three things can happen:

1. There exists an infinite sequence of expressions $e_1, e_2, \ldots$ such that $e \xrightarrow{1} e_1 \xrightarrow{1} e_2 \xrightarrow{1} \ldots$. In this case, we write $e \Uparrow$ and say that e *diverges*.
2. The expression e produces a value v in zero or more steps. In this case we say that e *converges* to the value v and write $e \Downarrow v$.
3. The computation converges to a non-value. When this happens, we say the computation is *stuck*.[1]

A semantic translation is *adequate* if these three behaviors in the source system are accurately reflected in the target system, and vice versa. This relationship is illustrated in the following diagram:



If e converges to a value v in the source language, then $[\![e]\!]$ must converge to some value v' that is equivalent (e.g. β-equivalent) to $[\![v]\!]$ in the target language, and vice-versa. This is formally stated as two properties, *soundness* and *completeness*. For our CBN-to-CBV translation, these properties take the following form:

Soundness:

(i) $[\![e]\!] \Downarrow_{\text{cbv}} t \;\Rightarrow\; \exists v\; t \approx [\![v]\!] \wedge e \Downarrow_{\text{cbn}} v$

(ii) $[\![e]\!] \Uparrow_{\text{cbv}} \Rightarrow\; e \Uparrow_{\text{cbn}}$

Completeness:

(i) $e \Downarrow_{\text{cbn}} v \;\Rightarrow\; \exists t\; t \approx [\![v]\!] \wedge [\![e]\!] \Downarrow_{\text{cbv}} t$

(ii) $e \Uparrow_{\text{cbn}} \Rightarrow\; [\![e]\!] \Uparrow_{\text{cbv}}$.

where $\approx$ is some notion of target term equivalence that is preserved by evaluation.

Soundness says that the computation in the CBV domain starting from the image $[\![e]\!]$ of a CBN program e accurately simulates the computation starting from e in the CBN domain. Thus if the CBV process terminates in a value, then so must the CBN process, and the values must be related in the sense described formally above; and if the CBV computation diverges, then so does the CBN computation. Completeness says the opposite: the computation in the CBN domain starting from e is accurately simulated by the computation in the CBV domain starting from $[\![e]\!]$.

Adequacy is the combination of soundness and completeness.

[1] This cannot happen with our CBN-to-CBV translation, but we will see some examples soon enough.

1 Proving Adequacy

We would like to show that evaluation commutes with our translation $[\![\cdot]\!]$ from CBN to CBV. To do this, we first need a notion of target term equivalence ($\approx$) that is preserved by evaluation. This is made more challenging because as evaluation takes place in the target language, intermediate terms are generated that are not the translation of any source term. For some translations (but not this one), the reverse may also happen. Therefore, equivalence needs to allow for some extra β-redexes that appear during translation. We can define this equivalence by structural induction on CBV target terms according to the following rules:

$$x \approx x \qquad \frac{t \approx t'}{\lambda x.\, t \approx \lambda x.\, t'} \qquad \frac{t_0 \approx t_0' \qquad t_1 \approx t_1'}{t_0\, t_1 \approx t_0'\, t_1'} \qquad t \approx (\lambda z.\, t)\,\mathsf{id}, \text{ where } z \notin FV(t)$$

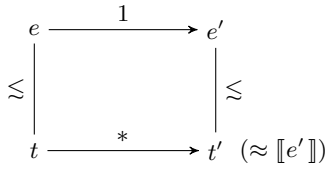
Here, t represents target terms, to keep them distinct from source terms e. We also include rules so that the relation $\approx$ is reflexive, symmetric, and transitive. One can show easily that if two terms are equivalent with respect to this relation, then they have the same β-normal form.

To show adequacy, we show that each evaluation step in the source term is mirrored by a sequence of evaluation steps in the corresponding target term, and vice versa. So we define a correspondence $\lesssim$ between source and target terms that is more general than the translation $[\![\cdot]\!]$ and is preserved during evaluation of both source and target.

We write $e \lesssim t$ to mean that CBN term e corresponds to CBV term t. The following proposition captures the idea that CBV evaluation simulates CBN evaluation at the level of individual steps:

$$e \lesssim t \ \wedge\ e \to e' \quad \Rightarrow \quad \exists t'\ t \xrightarrow{*} t' \ \wedge\ e' \lesssim t' \tag{1}$$

This can be visualized as a commutative diagram:



In fact, since in this case the source language cannot get stuck during evaluation, and both languages have deterministic evaluation, (1) ensures that evaluation in each language corresponds to the other.

We define the relation $\lesssim$ in such a way that $e \lesssim [\![e]\!]$. Then, using (1), we can show that any trace in the source language produces a corresponding trace in the target by induction on the number of source-language steps.

We define the relation $\lesssim$ by the following rules:

$$x \lesssim x\,\mathsf{id} \qquad \frac{e \lesssim t}{\lambda x.\, e \lesssim \lambda x.\, t} \qquad \frac{e_0 \lesssim t_0 \qquad e_1 \lesssim t_1}{e_0\, e_1 \lesssim t_0\, (\lambda\, .\, t_1)} \qquad \frac{e \lesssim t}{e \lesssim (\lambda\, .\, t)\,\mathsf{id}} \tag{2}$$

For simplicity, we ignore the fresh variable that would be used in the new lambda abstraction in the last two rules.

The first three rules of (2) ensure that a source term corresponds to its translation. The last rule is different; it takes care of the extra β-reductions that may arise during evaluation. Because the right-hand side of the $\lesssim$

relation becomes structurally smaller in this rule's premise, the definition of the relation is still well-founded. The first three rules are well-founded based on the structure of e; the last is well-founded based on the structure of t. If we were proving a more complex translation correct, we would need more rules like the last rule for other meaning-preserving target-language reductions.

First, let us warm up by showing that a term corresponds to its translation.

Lemma 1. $e \lesssim [\![e]\!]$.

Proof. An easy structural induction on e.

- Case x: $x \lesssim x\,\mathsf{id}$ by definition.
- Case $\lambda x.\, e'$: We have $[\![e]\!] = \lambda x.\, [\![e']\!]$. By the induction hypothesis, $e' \lesssim [\![e']\!]$, so $\lambda x.\, e' \lesssim \lambda x.\, [\![e']\!]$ by the second rule of (2).
- Case $e_0\, e_1$: We have $[\![e]\!] = [\![e_0]\!]\,(\lambda\,.\,[\![e_1]\!])$. By the induction hypothesis, $e_0 \lesssim [\![e_0]\!]$ and $e_1 \lesssim [\![e_1]\!]$. Therefore by the third rule of (2), $e_0\, e_1 \lesssim [\![e_0]\!]\,[\![e_1]\!]$.

□

Next, let us show that if e corresponds to t, its translation is equivalent to t.

Lemma 2. $e \lesssim t \;\Rightarrow\; [\![e]\!] \approx t$.

Proof. Induction on the derivation of $e \lesssim t$.

- Case $x \lesssim x\,\mathsf{id}$:
  This case is trivial: $[\![x]\!] = x\,\mathsf{id}$.
- Case $\lambda x.\, e' \lesssim \lambda x.\, t'$ where $e' \lesssim t'$:
  Here, $[\![e]\!] = \lambda x.\, [\![e']\!]$. By the induction hypothesis, $[\![e']\!] \approx t'$, therefore $\lambda x.\, [\![e']\!] \approx \lambda x.\, t'$ as required.
- Case $e_0\, e_1 \lesssim t_0\,(\lambda\,.\,t_1)$ where $e_0 \lesssim t_0$ and $e_1 \lesssim t_1$:
  Here, $[\![e_0\, e_1]\!] = [\![e_0]\!](\lambda\,.\,[\![e_1]\!])$, and by the induction hypothesis, $[\![e_0]\!] \approx t_0$ and $[\![e_1]\!] \approx t_1$. From the definition of $\approx$, we have $[\![e_0]\!](\lambda\,.\,[\![e_1]\!]) \approx t_0\,(\lambda\,.\,t_1)$.
- Case $e \lesssim (\lambda\,.\,t)\,\mathsf{id}$ where $e \lesssim t$:
  The induction hypothesis is $[\![e]\!] \approx t$. But $t \approx (\lambda\,.\,t)\,\mathsf{id}$, and $\approx$ is transitive.

□

Given these definitions, we can prove (1) by induction on the derivation of $e \lesssim t$. We will need two useful lemmas. The first is a substitution lemma that says substituting corresponding terms into corresponding terms produces corresponding terms:

Lemma 3. $e_1 \lesssim t_1 \wedge e_2 \lesssim t_2 \Rightarrow e_2\{e_1/x\} \lesssim t_2\{\lambda.\, t_1/x\}$.

Proof. We proceed by induction on the derivation of $e_2 \lesssim t_2$.

- Case $x \lesssim x\,\mathsf{id}$:
  We have $e_2\{e_1/x\} = e_1$ and $t_2\{\lambda.\, t_1/x\} = (\lambda.\, t_1)\,\mathsf{id}$. By the fourth rule of (2), we have $e_1 \lesssim (\lambda.\, t_1)\,\mathsf{id}$.

- Case $y \lesssim y\,\mathsf{id}$ where $y \neq x$:
  This case is trivial, as the substitution has no effect.

- Case $\lambda x.\, e \lesssim \lambda x.\, t$ where $e \lesssim t$:
  Again, this case is trivial, as the substitutions into e_2 and t_2 have no effect.

- Case $\lambda y.\, e \lesssim \lambda y.\, t$ where $e \lesssim t$, $x \neq y$:
  Here $e_2\{e_1/x\} = \lambda y.\, e\{e_1/x\}$ and $t_2\{\lambda.\, t_1/x\} = \lambda y.\, t\{\lambda.\, t_1/x\}$. Since $e \lesssim t$, by the induction hypothesis we have $e\{e_1/x\} \lesssim t\{\lambda.\, t_1/x\}$. Therefore by definition, $\lambda y.\, e\{e_1/x\} \lesssim \lambda y.\, t\{\lambda.\, t_1/x\}$, as required.

- Case $e\, e' \lesssim t\,(\lambda.\, t')$, where $e \lesssim t$ and $e' \lesssim t'$:
  We have $e_2\{e_1/x\} = e\{e_1/x\}\, e'\{e_1/x\}$, and $t_2\{\lambda.\, t_1/x\} = t\{\lambda.\, t_1/x\}\,(\lambda.\, t'\{t_1/x\})$. From the induction hypothesis, $e\{e_1/x\} \lesssim t\{\lambda.\, t_1/x\}$ and $e'\{e_1/x\} \lesssim t'\{\lambda.\, t_1/x\}$. Therefore, by definition we have $e\{e_1/x\}\, e'\{e_1/x\} \lesssim t\{\lambda.\, t_1/x\}\,(\lambda.\, t'\{t_1/x\})$.

- Case $e_2 \lesssim (\lambda\,.\, t_2')\,\mathsf{id}$, where $e_2 \lesssim t_2'$:
  We need to show that $e_2\{e_1/x\} \lesssim ((\lambda\,.\, t_2')\,\mathsf{id})\{\lambda\,.\, t_1/x\}$; that is, $e_2\{e_1/x\} \lesssim ((\lambda\,.\, t_2'\{\lambda\,.\, t_1/x\})\,\mathsf{id})$. From the induction hypothesis, we have $e_2\{e_1/x\} \lesssim t_2'\{\lambda.\, t_1/x\}$. By definition, this means $e_2\{e_1/x\} \lesssim (\lambda\,.\, t_2'\{\lambda.\, t_1/x\})\,\mathsf{id}$.

□

The next lemma says that if a value $\lambda x.\, e$ corresponds to a term t, then t reduces to a corresponding λ-term $\lambda\,.\, t'$.

Lemma 4. $\lambda x.\, e \lesssim t \Rightarrow \exists t'\ t \to \lambda x.\, t' \ \wedge\ e \lesssim t'$.

Proof. By induction on the derivation of $\lambda x.\, e \lesssim t$.

- Case $y \lesssim y\,\mathsf{id}$: Impossible, as $y \neq \lambda x.\, e$.

- Case $\lambda x.\, e \lesssim \lambda x.\, t'$ where $e \lesssim t'$:
  Here, $t = \lambda x.\, t'$, and the result is immediate.

- Case $e_0\, e_1 \lesssim t_0\,(\lambda.\, t_1)$: Impossible, as $e_0\, e_1 \neq \lambda x.\, e$.

- Case $e_0 \lesssim (\lambda.\, t_0)\,\mathsf{id}$, where $e_0 \lesssim t_0$:
  In this case $e_0 = \lambda x.\, e$, and $t = ((\lambda.\, t_0)\,\mathsf{id})$. By the induction hypothesis, there is some t' such that $t_0 \to \lambda x.\, t'$ and $e \lesssim t'$. Since $t = ((\lambda.\, t_0)\,\mathsf{id}) \xrightarrow{1} t_0$ we have $t \to \lambda x.\, t'$, as required.

□

We are now ready to prove (1).

Proof. By induction on the derivation of $e \lesssim t$:

- Case $x \lesssim x\,\mathsf{id}$: Vacuously true, as there is no evaluation step $e \xrightarrow{1} e'$.

- Case $\lambda x.\, e \lesssim \lambda x.\, t$: A value: also vacuously true.

- Case $e_0\, e_1 \lesssim t_0\,(\lambda.\, t_1)$, where $e_0 \lesssim t_0$ and $e_1 \lesssim t_1$:
  We show this by cases on the derivation of $e \xrightarrow{1} e'$:

- Case $e_0\, e_1 \xrightarrow{1} e_0'\, e_1$, where $e_0 \xrightarrow{1} e_0'$:
  By the induction hypothesis, $\exists t_0'\ e_0' \lesssim t_0' \wedge t_0 \to t_0'$. It is easy to see that $t_0\,(\lambda.\, t_1) \to t_0'\,(\lambda.\, t_1)$. By the third rule of (2), $e_0'\, e_1 \lesssim t_0'\,(\lambda.\, t_1)$, as required.
- Case $(\lambda x.\, e_2)\, e_1 \xrightarrow{1} e_2\{e_1/x\}$:
  Here $\lambda x.\, e_2 \lesssim t_0$ and $e_1 \lesssim t_1$.
  By Lemma 4, there exists a t_2 such that $t_0 \to \lambda x.\, t_2$ and $e_2 \lesssim t_2$. Therefore, we have $t_0\,(\lambda\,.\, t_1) \to (\lambda x.\, t_2)\ (\lambda\,.\, t_1) \xrightarrow{1} t_2\{\lambda.\, t_1/x\}$. But from the substitution lemma above (Lemma 3), we know that $e_2\{e_1/x\} \lesssim t_2\{\lambda.\, t_1/x\}$, as required.

- Case $e_0 \lesssim (\lambda.\, t_0)\ \mathsf{id}$, where $e_0 \lesssim t_0$:
  By the induction hypothesis, $\exists t_0'\ e_0 \lesssim t_0' \wedge t_0 \to t_0'$. It is easy to see that therefore $((\lambda.\, t_0)\ \mathsf{id}) \xrightarrow{1} t_0 \to t_0'$, as required.

□

Having proved (1), we can show completeness of the translation. If we start with a source term e and its translation $[\![e]\!]$, we know from Lemma 1 that $e \lesssim [\![e]\!]$. From (1), we know that each step of evaluation of e is mirrored by execution on the target side that preserves $e \lesssim t$. If the evaluation of e diverges, so will the evaluation of $[\![e]\!]$. If the evaluation of e converges on a value v, then the evaluation of $[\![e]\!]$ will reach a convergent (by Lemma 4) term t such that $v \lesssim t$. And by Lemma 2, $[\![v]\!] \approx t$. This demonstrates completeness.

To show soundness of the translation, we need to show that every evaluation in the target language corresponds to some evaluation in the source language. Suppose we have a target-language evaluation $t \to v'$, where $t = [\![e]\!]$, but there is no corresponding source-language evaluation of e. There are three possibilities. First, the evaluation of e could get stuck. This cannot happen for this source language because all terms are either values or have a legal evaluation. Second, the evaluation of e could evaluate to a value v. But then v must correspond to v', because the target-language evaluation is deterministic. Third, the evaluation of e might diverge. But then (1) says there is a divergent target-language evaluation. The determinism of the target language ensures that cannot happen.