The role of indirections in lazy natural semantics (extended version)

Technical Report 13/13
Lidia Sánchez-Gil[1], Mercedes Hidalgo-Herrero[2], and Yolanda Ortega-Mallén[1]

[1] Facultad de CC. Matemáticas, Universidad Complutense de Madrid, Spain
[2] Facultad de Educación, Universidad Complutense de Madrid, Spain

Abstract. Launchbury defines a natural semantics for lazy evaluation and proposes an alternative call-by-name version which introduces indirections and does not update closures. These changes in the semantic rules are not so innocuous as they seem, so that the equivalence of both semantics is not straightforward. We separate the two modifications and define two intermediate semantics: one with indirections and the other without update. In the present work we focus on the introduction of indirections during β-reduction and study how the heaps, i.e., the sets of bindings, obtained with this kind of evaluation do relate with the heaps produced by substitution. As a heap represents the context of evaluation for a term, we first define an equivalence that identifies terms with the same meaning under a given context. This notion of *context* equivalence is extended to heaps. Finally, we define a relation between heap/term pairs to establish the equivalence between the alternative natural semantics and its corresponding version without indirections.

1 Motivation

Twenty years have elapsed since Launchbury first presented in [8] his natural semantics for lazy evaluation, a key contribution to the semantic foundation for non-strict functional programming languages like Haskell or Clean. Launchbury defines in [8] a natural semantics for lazy evaluation (*call-by-need*) where the set of *bindings*, i.e., (variable, expression) pairs, is explicitly managed to make possible their sharing. Throughout these years, Launchbury's natural semantics has been frequently cited and has inspired many further works as well as several extensions like in [2, 9, 15, 17]. In [12] the authors of this paper have extended the lambda calculus with a new expression that introduces parallelism when performing functional applications. *Parallel application* creates new processes to distribute the computation, and these distributed processes exchange values through communication channels. For that reason, we have presented an extension of Launchbury's natural semantics with parallel application. The success of Lanchbury's proposal lies in its simplicity. Expressions are evaluated with respect to a *context*, which is represented by a heap of *bindings*. This heap is explicitly managed to make possible the sharing of bindings, thus, modeling laziness.

In order to prove that this lazy (operational) semantics is *correct* and *computationally adequate* with respect to a standard denotational semantics, Launchbury introduces some variations in the operational semantics. On the one hand, the update of bindings with their computed values is an operational notion without counterpart in the standard denotational semantics, so that the alternative natural semantics does no longer update bindings and becomes a *call-by-name* semantics. Moreover, in the alternative semantics self-references yield infinite reductions, while in the original semantics the reduction of a self-reference gets blocked. On the other hand, functional application is modeled denotationally by extending the environment with a variable bound to a value. This new variable represents the formal parameter of the function, while the value corresponds to the actual argument. For a closer approach to this mechanism, applications are carried out in the alternative operational semantics by introducing *indirections*, i.e., variables bound to variables, instead of by performing the β-reduction through substitution.

Unfortunately, the proof of the equivalence between the natural semantics and its alternative version is detailed nowhere, and a simple induction turns out to be insufficient. Intuitively, both reduction systems

should lead to the same results. However, the *context-heap* semantics is too sensitive to the changes introduced by the alternative rules. Consequently, the equivalence cannot be directly established since final values may contain free variables that are dependent on the context of evaluation, which is represented by the heap of bindings. No updating leads to the duplication of bindings, and although these duplicated bindings, as well as the indirections, do not add relevant information to the context, it is awkward to prove this fact. Therefore, our challenge is to establish a way of relating the heaps obtained with each reduction system, and to prove that at the end the semantics are equivalent, so that any reduction of a term in one of the systems has its counterpart in the other. To facilitate this task we consider separately the no updating and the introduction of indirections, and we define two intermediate semantics.

In this paper we investigate the effect of introducing indirections in a setting without updates, and we analyze the similarities and differences between the reductions proofs obtained with and without indirections. This analysis provides a deep insight on the behavior of a context-heap semantics like Launchbury's.

We want to identify terms up to α-conversion, but dealing with α-equated terms usually implies the use of Barendregt's variable convention [3] to avoid the renaming of bound variables. However, the use of the variable convention is sometimes dubious and may lead to *faulty* results (as it is shown by Urban et al. in [16]). Moreover, we intend to formalize our results with the help of the Coq [4] proof assistant. These reasons have led us to look for a system of binding amenable to formalization and we have chosen a *locally nameless* representation (as presented by Charguéraud in [6]). This is a mixed notation where bound variables names are replaced by de Bruijn indices [7], while free variables preserve their names. Although de Bruijn indices solve the problem with α-conversion, they make it very complicated to deal with heaps. Moreover, the formalization becomes unreadable. Hence, the mixed notation of naming variables is more convenient. A locally nameless version of Launchbury's natural semantics has been presented by the authors in [14] and [13].

The main contributions of the present work are:

1. An equivalence relation that identifies heaps that define the same free variables but whose corresponding closures may differ on *undefined free variables*;
2. A preorder that relates two heaps whenever the first can be transformed into the second by *eliminating indirections*;
3. An extension of the preorder relation for heap/term pairs expressing that two terms are equivalent if they have the same structure and their free variables, defined in the context of the respective heaps, are the same except for some indirections.
4. An equivalence theorem for Launchbury's alternative semantics and a version without indirections (and without update).

The paper is structured as follows: In Section 2, we review the lambda calculus and the two natural semantics described by Launchbury in [8]. We also introduce two intermediate semantics, each introducing just one alternative rule. In Section 3 we give a locally nameless representation of the calculus and the semantics presented in Section 2. In Section 4 we define equivalence and preorder relations on terms, heaps and also on heap/term pairs. We include a number of interesting results concerning these relations and, finally, we prove the equivalence of Launchbury's alternative semantics and the intermediate semantics without update and without indirections. In the last section we draw conclusions and outline our future work.

The proofs of theorems, propositions and lemmas are detailed in the Appendix.

2 Lazy natural semantics

In this section we review the natural semantics defined by Launchbury in [8] for lazy evaluation, together with the alternative rules for application and variables which transform the lazy semantics in a call-by-name semantics. In order to facilitate the comparison of these semantics, we focus independently on each change and define two intermediate semantics.

$$x \in \mathit{Var}$$
$$e \in \mathit{Exp} ::= x \mid \lambda x.e \mid (e\ x) \mid \texttt{let}\ \{x_i = e_i\}_{i=1}^{n}\ \texttt{in}\ e$$

Fig. 1. Restricted syntax of the extended λ-calculus

$$\textsc{Lam}\quad \Gamma : \lambda x.e \Downarrow \Gamma : \lambda x.e \qquad \textsc{App}\quad \frac{\Gamma : e \Downarrow \Theta : \lambda y.e' \qquad \Theta : e'[x/y] \Downarrow \Delta : w}{\Gamma : (e\ x) \Downarrow \Delta : w}$$

$$\textsc{Var}\quad \frac{\Gamma : e \Downarrow \Delta : w}{(\Gamma, x \mapsto e) : x \Downarrow (\Delta, x \mapsto w) : \hat{w}} \qquad \textsc{Let}\quad \frac{(\Gamma, \{x_i \mapsto e_i\}_{i=1}^{n}) : e \Downarrow \Delta : w}{\Gamma : \texttt{let}\ \{x_i = e_i\}_{i=1}^{n}\ \texttt{in}\ e \Downarrow \Delta : w}$$

Fig. 2. Natural semantics

2.1 Natural semantics

The language described in [8] is a normalized lambda calculus extended with recursive local declarations. The (restricted) abstract syntax appears in Figure 1. Normalization is achieved in two steps: First an α-conversion is performed so that bound variables have distinct names; in a second phase, arguments for applications are enforced to be variables. These static transformations simplify the definition of the operational rules.

The natural semantics defined by Launchbury in [8] follows a call-by-need strategy. Judgements are of the form $\Gamma : e \Downarrow \Delta : w$, that is, the expression e in the context of the heap Γ reduces to the value w in the context of the heap Δ. *Heaps* are partial functions from variables into expressions. Each pair (variable, expression) is called a *binding* and it is represented by $x \mapsto e$. During evaluation, new bindings may be added to the heap, and bindings may be updated to their corresponding computed values. *Values* ($w \in \mathit{Val}$) are expressions in weak-head-normal-form (*whnf*). The semantic rules are shown in Figure 2. Despite of the normalization, in the Var rule an α-conversion of the final value, represented as $\hat{w}$, is still needed to avoid name clashing. This renaming is justified by Barendregt's variable convention [3].

2.2 Alternative natural semantics

In order to prove the computational adequacy of the natural semantics (Figure 2) with respect to a standard denotational semantics, Launchbury introduces the alternative rules for application and variables shown in Figure 3. The AVar rule removes update from the semantics. The effect of AApp is the addition of an indirection $(y \mapsto x)$[1] instead of performing the β-reduction by substitution as in $e'[x/y]$ in App. This increases the number of bindings in the heap.

In the following, the natural semantics (rules in Figure 2) is referred as the NS, and the alternative semantics (rules Lam, Let and those in Figure 3) as the ANS. We write $\Downarrow^{A}$ for reductions in the ANS.

Launchbury proves in [8] the correctness of the NS with respect to a standard denotational semantics, and a similar result for the ANS is easily obtained (as the authors of this paper have done in [11]). Therefore, the NS and the ANS are "denotationally" equivalent in the sense that if an expression is reducible (in some heap context) by both semantics then the obtained values have the same denotation. But this is insufficient for our purposes, because we want to ensure that if for some (heap : term) pair a reduction exists in any of the semantics, then there must exist a reduction in the other too and the final heaps must be related.

[1] Thanks to the normalization and the α-conversion $\hat{e}$ in AVar, it is guaranteed that y is fresh in Θ.

$$\textsc{AVar}\quad \frac{(\Gamma, x \mapsto e) : \hat{e} \Downarrow \Delta : w}{(\Gamma, x \mapsto e) : x \Downarrow \Delta : w} \qquad \textsc{AApp}\quad \frac{\Gamma : e \Downarrow \Theta : \lambda y.e' \qquad (\Theta, y \mapsto x) : e' \Downarrow \Delta : w}{\Gamma : (e\ x) \Downarrow \Delta : w}$$

Fig. 3. Alternative rules

2.3 Two intermediate semantics

The changes introduced by the ANS might seem to involve no serious difficulties to prove that any reduction proof from a (heap : term) pair with the NS implies some reduction with the ANS, such that the final (heap : value) pairs are "equivalent", and vice versa. Unfortunately things are not so easy. On the one hand, the alternative rule for variables transforms the original call-by-need semantics into a call-by-name semantics because bindings are not updated and computed values are no longer shared. On the other hand, the addition of indirections also complicates the task of comparing the (heap : value) pairs obtained by each reduction system. Notice that the introduction of indirections produces larger heaps and the final values may depend on these aditional names.

To deal separately with these difficulties we introduce two intermediate semantics, each corresponding to the inclusion of just one of the modifications into the NS. Thus, the rules of the *Indirection Natural Semantics* (INS) are those of the NS (Figure 2) except for the application rule, that corresponds to the one in the alternative version, i.e., AApp in Figure 3. Analogously, the rules of the *No-update Natural Semantics* (NNS) are those of the NS but for the variable rule, that corresponds to the alternative AVar rule in Figure 3. We use $\Downarrow^I$ to represent reductions of the INS and $\Downarrow^N$ for those of the NNS. The following table summarizes the characteristics of the four reduction systems defined so far:

	NS	**INS**	**NNS**	**ANS**
Indirections	✗	✓	✗	✓
Update	✓	✓	✗	✗

3 A locally nameless representation

The syntax given in Figure 1 includes two name binders: λ-abstraction and `let`-declaration. The *named representation* requires a quotient structure respect to α-equivalence. To avoid this in our analysis, we opt for a *locally nameless representation* [6] that uses de Bruijn indices [7] to represent bound variables while names are retained for free variables. We have chosen such a mixed notation because heaps collect free variables whose names we are interested in preserving in order to identify them more easily.

As mentioned above, our locally nameless representation has already been presented in [14] and [13]. A complete definition with detailed explanations can be found there. Here we just show what is indispensable to understand the present work.

3.1 Locally nameless syntax

The locally nameless syntax corresponding to the lambda calculus of Figure 1 is shown in Figure 4. *Var* stands now for the set of *variables*, where *bound variables* and *free variables* are distinguished. When the language includes multibinders, Charguéraud proposes in [6] the use of two natural numbers to represent bound variables. We choose this notation to designate bound variables from `let`-declarations, which are in fact multibinders. The first number is a de Bruijn index that counts how many binders (abstraction or `let`) one needs to cross to the left to reach the corresponding binder for the variable, while the second refers to the position of the variable inside that binder. Abstractions are seen as multi-binders that bind only one variable, so that the second number should be always zero to represent a correct term. In the following, a list like $\{t_i\}_{i=1}^n$ is represented as $\overline{t}$, with length $|\overline{t}| = n$.

$$
\begin{array}{lll}
x \in \mathit{Id} & & i, j \in \mathbb{N} \\
v \in \mathit{Var} & ::= & \texttt{bvar}\ i\ j \mid \texttt{fvar}\ x \\
t \in \mathit{LNExp} & ::= & v \mid \texttt{abs}\ t \mid \texttt{app}\ t\ v \mid \texttt{let}\ \{t_i\}_{i=1}^{n}\ \texttt{in}\ t
\end{array}
$$

Fig. 4. Locally nameless syntax

Computing the *free variables* of a term t, denoted by $\texttt{fv}(t)$, is very easy when using the locally nameless representation, since bound and free variables are syntactically different:

$$
\begin{array}{lll@{\qquad}lll}
\texttt{fv}(\texttt{bvar}\ i\ j) & = & \emptyset & \texttt{fv}(\texttt{fvar}\ x) & = & \{x\} \\
\texttt{fv}(\texttt{abs}\ t) & = & \texttt{fv}(t) & \texttt{fv}(\texttt{app}\ t\ v) & = & \texttt{fv}(t) \cup \texttt{fv}(v) \\
\texttt{fv}(\texttt{let}\ \overline{t}\ \texttt{in}\ t) & = & \texttt{fv}(\overline{t}) \cup \texttt{fv}(t) & & &
\end{array}
$$

where $\texttt{fv}(\overline{t})$ collects the free variables of all the terms in a list $\overline{t}$.

A name $x \in \mathit{Id}$ is *fresh in a term* $t \in \mathit{LNExp}$, written $\texttt{fresh}\ x\ \texttt{in}\ t$, if x does not belong to the set of free variables of t, i.e., $x \notin \texttt{fv}(t)$. This is extended to a list of names: $\texttt{fresh}\ \overline{x}\ \texttt{in}\ t \stackrel{\texttt{def}}{=} \overline{x} \notin \texttt{fv}(t)$.

Definition 1. *Two terms $t, t' \in \mathit{LNExp}$ have the* same structure*, written $t \sim_S t'$, if they differ only in the names of their free variables:*

$$
\text{SS\_BVAR}\ \frac{}{(\texttt{bvar}\ i\ j) \sim_S (\texttt{bvar}\ i\ j)} \qquad \text{SS\_FVAR}\ \frac{}{(\texttt{fvar}\ x) \sim_S (\texttt{fvar}\ y)}
$$

$$
\text{SS\_ABS}\ \frac{t \sim_S t'}{(\texttt{abs}\ t) \sim_S (\texttt{abs}\ t')} \qquad \text{SS\_APP}\ \frac{t \sim_S t' \quad v \sim_S v'}{(\texttt{app}\ t\ v) \sim_S (\texttt{app}\ t'\ v')}
$$

$$
\text{SS\_LET}\ \frac{|\overline{t}| = |\overline{t'}| \quad \overline{t} \sim_S \overline{t'} \quad t \sim_S t'}{(\texttt{let}\ \overline{t}\ \texttt{in}\ t) \sim_S (\texttt{let}\ \overline{t'}\ \texttt{in}\ t')}
$$

where $\overline{t} \sim_S \overline{t'} = \texttt{List.forall2}\ (\cdot \sim_S \cdot)\ \overline{t}\ \overline{t'}$. [2]

Next proposition states that $\sim_S$ is an equivalence relation on *LNExp*:

Proposition 1.

SS_REF $\quad t \sim_S t$

SS_SIM $\quad t \sim_S t' \Rightarrow t' \sim_S t$

SS_TRANS $\quad t \sim_S t' \wedge t' \sim_S t'' \Rightarrow t \sim_S t''$

Since there is no danger of name capture, *substitution* of variable names in a term is trivial in the locally nameless representation. We write $t[y/x]$ for replacing the occurrences of x by y in the term t.

Name substitution preserves the structure of a term:

Lemma 1.

SS_SUBST $\quad t[y/x] \sim_S t$

A *variable opening* operation is needed to manipulate locally nameless terms. This operation turns the outermost bound variables into free variables. The operation is defined in terms of a more general function with an extra parameter representing the nesting level of the binder to be open.

A term $t \in \mathit{LNExp}$ is *open with a list of names* $\overline{x} \subseteq \mathit{Id}$, written $t^{\overline{x}}$, as follows:

[2] We use an ML-like notation for operations on lists. For instance, `List.map` applies a function to every component in a list and `List.nth` refers to the nth-component. Elements in lists are numbered starting with 0 to match bound variables indices.

$$t^{\overline{x}} = \{0 \to \overline{x}\}t$$

where

$$\begin{array}{ll}
\{k \to \overline{x}\}(\texttt{bvar}\ i\ j) & = \begin{cases} \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{x}) & \text{if } i = k \wedge j < |\overline{x}| \\ \texttt{bvar}\ i\ j & \text{otherwise} \end{cases} \\
\{k \to \overline{x}\}(\texttt{fvar}\ x) & = \texttt{fvar}\ x \\
\{k \to \overline{x}\}(\texttt{abs}\ t) & = \texttt{abs}\ (\{k+1 \to \overline{x}\}\ t) \\
\{k \to \overline{x}\}(\texttt{app}\ t\ v) & = \texttt{app}\ (\{k \to \overline{x}\}\ t)\ (\{k \to \overline{x}\}\ v) \\
\{k \to \overline{x}\}(\texttt{let}\ \overline{t}\ \texttt{in}\ t) & = \texttt{let}\ (\{k+1 \to \overline{x}\}\ \overline{t})\ \texttt{in}\ (\{k+1 \to \overline{x}\}\ t) \\
\text{and } \{k \to \overline{x}\}\ \overline{t} & = \texttt{List.map}\ (\{k \to \overline{x}\}\cdot)\ \overline{t}.
\end{array}$$

For simplicity, we write t^x for the variable opening with a unitary list $[x]$. We illustrate this concept and its use with an example:

Example 1. Let $t \equiv \texttt{abs}\ (\texttt{let bvar 0 1}, \texttt{bvar 1 0 in app}\ (\texttt{abs bvar 2 0})\ (\texttt{bvar 0 1}))$. Hence, the body of the abstraction is:

$$u \equiv \texttt{let bvar 0 1}, \boxed{\texttt{bvar 1 0}}\ \texttt{in app}\ (\texttt{abs}\ \boxed{\texttt{bvar 2 0}})\ (\texttt{bvar 0 1}).$$

But then in u the bound variables referring to the outermost abstraction of t (shown squared) point to nowhere. The opening of u with variable x replaces with x the bound variables referring to an hypothetical binder with body u:
$u^x = \texttt{let bvar 0 1}, \texttt{fvar}\ x\ \texttt{in app}\ (\texttt{abs fvar}\ x)\ (\texttt{bvar 0 1})$. □

In some occasions we are interested in applying the opening operation to a list of terms, so that we define $\overline{t}^{\overline{x}} = \texttt{List.map}\ (\cdot^{\overline{x}})\ \overline{t}$. From now on, $\overline{x}$ represents a list of pairwise-distinct names in Id.

The structure of a term that has been opened does not depend on the names chosen for the opening:

Lemma 2.
SS_OP $\quad |\overline{x}| = |\overline{y}| \Rightarrow t^{\overline{x}} \sim_S t^{\overline{y}}$

Inversely to variable opening, there is an operation to transform free names into bound variables. The *variable closing* of a term is represented by ${}^{\backslash\overline{x}}t$, where $\overline{x}$ is the list of names to be bound (recall that the names in $\overline{x}$ are distinct):

$${}^{\backslash\overline{x}}t = \{0 \leftarrow \overline{x}\}t$$

where

$$\begin{array}{ll}
\{k \leftarrow \overline{x}\}(\texttt{bvar}\ i\ j) & = \texttt{bvar}\ i\ j \\
\{k \leftarrow \overline{x}\}(\texttt{fvar}\ x) & = \begin{cases} \texttt{bvar}\ k\ j & \text{if } \exists j : 0 \le j < |\overline{x}|.x = \texttt{List.nth}\ j\ \overline{x} \\ \texttt{fvar}\ x & \text{otherwise} \end{cases} \\
\{k \leftarrow \overline{x}\}(\texttt{abs}\ t) & = \texttt{abs}\ (\{k+1 \leftarrow \overline{x}\}\ t) \\
\{k \leftarrow \overline{x}\}(\texttt{app}\ t\ v) & = \texttt{app}\ (\{k \leftarrow \overline{x}\}\ t)\ (\{k \leftarrow \overline{x}\}\ v) \\
\{k \leftarrow \overline{x}\}(\texttt{let}\ \overline{t}\ \texttt{in}\ t) & = \texttt{let}\ (\{k+1 \leftarrow \overline{x}\}\ \overline{t})\ \texttt{in}\ (\{k+1 \leftarrow \overline{x}\}\ t) \\
\text{and } \{k \leftarrow \overline{x}\}\ \overline{t} & = \texttt{List.map}\ (\{k \leftarrow \overline{x}\}\ \cdot)\ \overline{t}.
\end{array}$$

Under some conditions variable closing and variable opening are inverse operations:

Lemma 3.
CLOSE_OPEN $\quad \texttt{fresh}\ \overline{x}\ \texttt{in}\ t \Leftrightarrow {}^{\backslash\overline{x}}(t^{\overline{x}}) = t$

For the other way around a condition on terms is required. The locally nameless syntax given in Figure 4 allows to build terms that have no corresponding expression in *Exp* (Figure 1). For instance, when bound variables indices are out of range. Those terms in *LNExp* that match expressions in *Exp* are called *locally-closed*, written $\texttt{lc}\ t$. The predicate lc is defined by the following rules:

$$\text{LC\_VAR}\ \frac{}{\texttt{lc}\ (\texttt{fvar}\ x)} \qquad \text{LC\_ABS}\ \frac{\forall x \notin L \subseteq Id \quad \texttt{lc}\ t^x}{\texttt{lc}\ (\texttt{abs}\ t)}$$

$$\text{LC\_APP}\ \frac{\texttt{lc}\ t \quad \texttt{lc}\ v}{\texttt{lc}\ (\texttt{app}\ t\ v)} \qquad \text{LC\_LET}\ \frac{\forall \overline{x}^{|\overline{t}|} \notin L \subseteq Id \quad \texttt{lc}\ [t : \overline{t}]^{\overline{x}}}{\texttt{lc}\ (\texttt{let}\ \overline{t}\ \texttt{in}\ t)}$$

where $\texttt{lc}\ \overline{t} = \texttt{List.forall}\ (\texttt{lc}\ \cdot)\ \overline{t}$ and $\overline{x}^n \notin L$ indicates that $\overline{x}$ is a list of n pairwise distinct names not belonging to the *(finite)* set L.

In the rules LC_ABS and LC_LET we use *cofinite quantification* as described in [1]. Cofinite quantification is an elegant alternative to "exist-fresh" conditions and provides stronger induction and inversion principles. We use the notation $[t : \overline{t}]$ to represent the list with head t and tail $\overline{t}$. Later on the empty list is represented as $[\,]$, a unitary list as $[t]$, and $+\!\!+$ stands for the concatenation of lists.

Now we can write down the property complementary of Lemma 3:

Lemma 4.
OPEN_CLOSE $\quad \texttt{lc}\ t \Rightarrow (\backslash^{\overline{x}} t)^{\overline{x}} = t$

The locally-closure condition of a term is independent of the names of its free variables:

Lemma 5.
SS_LC $\quad t \sim_S t' \wedge \texttt{lc}\ t \Rightarrow \texttt{lc}\ t'$

Moreover, any locally closed term can be expressed as the variable opening of another term that does not contain the names chosen for the opening:

Lemma 6.
LC_OP_VARS $\quad \texttt{lc}\ t \wedge \overline{x} \subseteq Id \Rightarrow \exists s \in \mathit{LNExp}.\, (\texttt{fresh}\ \overline{x}\ \texttt{in}\ s \wedge s^{\overline{x}} = t)$

This result is useful to express that a term depends on some set of names.

3.2 Locally nameless semantics

Bindings in a heap associate expressions to free variables, therefore bindings are now pairs $(\texttt{fvar}\ x, t)$ with $x \in Id$ and $t \in \mathit{LNExp}$. To simplify, we just write $x \mapsto t$. In the following, we represent a heap $\{x_i \mapsto t_i\}_{i=1}^n$ as $(\overline{x} \mapsto \overline{t})$, with $|\overline{x}| = |\overline{t}| = n$. The set of the locally-nameless-heaps is denoted as *LNHeap*.

The *domain* of a heap Γ, written $\texttt{dom}(\Gamma)$, collects the set of names that are defined in the heap:

$$\texttt{dom}(\emptyset) = \emptyset \qquad \texttt{dom}(\Gamma, x \mapsto t) = \texttt{dom}(\Gamma) \cup \{x\}$$

In a *well-formed* heap names are defined at most once and terms are locally closed. The predicate ok expresses that a heap is well-formed:

$$\text{OK-EMPTY}\ \frac{}{\texttt{ok}\ \emptyset} \qquad \text{OK-CONS}\ \frac{\texttt{ok}\ \Gamma \qquad x \notin \texttt{dom}(\Gamma) \qquad \texttt{lc}\ t}{\texttt{ok}\ (\Gamma, x \mapsto t)}$$

The function names returns the set of names that appear in a heap, i.e., the names occurring in the domain or in the terms in the right-hand side:

$$\texttt{names}(\emptyset) = \emptyset \qquad \texttt{names}(\Gamma, x \mapsto t) = \texttt{names}(\Gamma) \cup \{x\} \cup \texttt{fv}(t)$$

and this is used to define the freshness predicate of a list of names in a heap:

$$\texttt{fresh}\ \overline{x}\ \texttt{in}\ \Gamma = \overline{x} \notin \texttt{names}(\Gamma).$$

$$\text{LNLam}\quad \frac{\{\texttt{ok}\ \Gamma\} \qquad \{\texttt{lc}\ (\texttt{abs}\ t)\}}{\Gamma : \texttt{abs}\ t \Downarrow \Gamma : \texttt{abs}\ t}$$

$$\text{LNVar}\quad \frac{\Gamma : t \Downarrow \Delta : w \qquad \{x \notin \texttt{dom}(\Gamma) \cup \texttt{dom}(\Delta)\}}{(\Gamma, x \mapsto t) : \texttt{fvar}\ x \Downarrow (\Delta, x \mapsto w) : w}$$

$$\text{LNApp}\quad \frac{\Gamma : t \Downarrow \Theta : \texttt{abs}\ u \qquad \Theta : u^x \Downarrow \Delta : w \qquad \{x \notin \texttt{dom}(\Gamma) \Rightarrow x \notin \texttt{dom}(\Delta)\}}{\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x) \Downarrow \Delta : w}$$

$$\text{LNLet}\quad \frac{\forall \overline{x}^{|\overline{t}|} \notin L \subseteq Id\ \ (\Gamma, \overline{x} \mapsto \overline{t}^{\overline{x}}) : t^{\overline{x}} \Downarrow (\overline{x} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{x}}) : w^{\overline{x}} \wedge {}^{\backslash \overline{x}}(\overline{s}^{\overline{x}}) = \overline{s} \wedge {}^{\backslash \overline{x}}(w^{\overline{x}}) = w \quad \{\overline{y}^{|\overline{t}|} \notin L\}}{\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t \Downarrow (\overline{y} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{y}}) : w^{\overline{y}}}$$

Fig. 5. Natural semantics with locally nameless representation

$$\text{ALNVar}\quad \frac{(\Gamma, x \mapsto t) : t \Downarrow \Delta : w}{(\Gamma, x \mapsto t) : \texttt{fvar}\ x \Downarrow \Delta : w}$$

$$\text{ALNApp}\quad \frac{\begin{array}{c}\Gamma : t \Downarrow \Theta : \texttt{abs}\ u \\ \forall y \notin L \subseteq Id \quad (\Theta, y \mapsto \texttt{fvar}\ x) : u^y \Downarrow ([y : \overline{z}] \mapsto \overline{s}^y) : w^y \ \wedge\ {}^{\backslash y}(\overline{s}^y) = \overline{s} \wedge {}^{\backslash y}(w^y) = w \\ \{x \notin \texttt{dom}(\Gamma) \Rightarrow x \notin [z : \overline{z}]\} \qquad \{z \notin L\}\end{array}}{\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x) \Downarrow ([z : \overline{z}] \mapsto \overline{s}^z) : w^z}$$

Fig. 6. Alternative natural semantics with locally nameless representation

These definitions are extended to (heap : term) pairs:

$$\texttt{names}(\Gamma : t) = \texttt{names}(\Gamma) \cup \texttt{fv}(t) \qquad\qquad \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma : t) = \overline{x} \notin \texttt{names}(\Gamma : t)$$

Substitution of variable names is extended to heaps as follows:

$$\emptyset[z/y] = \emptyset \qquad\qquad (\Gamma, x \mapsto t)[z/y] = (\Gamma[z/y], x[z/y] \mapsto t[z/y])$$
$$\text{where } x[z/y] = \begin{cases} z & \text{if } x = y \\ x & \text{otherwise} \end{cases}$$

It is required that $z \notin \texttt{dom}(\Gamma)$ to guarantee that name substitution preserves well-formedness.

The rules for the natural semantics (Figure 2) using the locally nameless representation are shown in Figure 5 and the alternative rules (Figure 3) in Figure 6. For clarity, side-conditions in the rules are written within braces to distinguish them from judgements.

Notice that we introduce cofinite quantification in rules LNLet and ALNApp. As it is explained in [6], the advantage of the cofinite rules over existential and universal ones is that the freshness side-conditions are not explicit. Nevertheless, the finite set L represents somehow the names that should be avoided during a reduction proof. We use the variable opening to express that the final heap and value may depend on the chosen names. For instance, in LNLet, $w^{\overline{x}}$ indicates that it depends on the names $\overline{x}$, but there is a common basis w (Lemma 6 is helpful to deal with this notation in proofs). Moreover, it is required that this basis does not contain occurrences of $\overline{x}$; this is is expressed by ${}^{\backslash \overline{x}}(w^{\overline{x}}) = w$. By contrast to the rules for the predicate `lc` (in Section 3.1), in the cofinite rules LNLet and ALNApp the names introduced in the (infinite) premises do appear in the conclusion too. Therefore, the conclusion has to be particularized to some selected names not belonging to L.

A more detailed explanation of the rules of Figure 5 can be found in [14]. New here are the alternative rules of Figure 6. The rule ALNVar is a direct translation of the rule AVar. Notice that the renaming of the term in the premise is no longer needed. The rule ALNApp follows a similar pattern to LNLet. The

cofinite quantification indicates that the name introduced in the heap for the indirection should be fresh. Among infinite valid candidates, the name z is chosen for the reduction. The list $\overline{z}$ represents the rest of names defined in the heap which is obtained after the reduction. Since the rhs terms of this final heap may refer to the name of the indirection, they are represented as $\overline{s}^z$, i.e., each rhs term is open with the name z (this can be done by Lemma 6). The condition $^{\backslash y}(\overline{s}^y) = \overline{s}$ ensures that s does not further depend on z. Similar considerations are valid for the final value.

3.3 Properties

The reduction systems corresponding to the rules given in Figures 5 and 6 verify a number of interesting properties. Since some of these properties are true for the four reduction systems defined in Section 2, in the following (and when not indicated otherwise) we use $\Downarrow^K$ to represent $\Downarrow$, $\Downarrow^A$, $\Downarrow^I$, and $\Downarrow^N$.

The first property is a *regularity* lemma that ensures that the judgements produced by the locally nameless rules involve only well-formed heaps and locally closed terms:

Lemma 7.
REGULARITY $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \wedge \mathtt{ok}\ \Delta \wedge \mathtt{lc}\ w$

Another general property is that definitions are not lost during reduction, i.e., heaps only can grow with new names:

Lemma 8.
DEF_NOT_LOST $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Delta).$

Moreover, in the case of no update (NNS and ANS), the bindings in the initial heap are preserved during the whole reduction:

Lemma 9.
NO_UPDATE $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \Gamma \subseteq \Delta$
where $\Downarrow^K$ *represents* $\Downarrow^N$ *and* $\Downarrow^A$

During reduction, names might be added to the heap by the rules LNLet and ALNApp. However, there is no "spontaneous generation" of names, i.e., any name occurring in a final (heap : value) pair must either appear already in the initial (heap : term) pair or be defined in the final heap:

Lemma 10.
ADD_NAMES $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{names}(\Delta : w) \subseteq \mathtt{names}(\Gamma : t) \cup \mathtt{dom}(\Delta).$

The freshness of the names introduced by the rules LNLet and ALNApp is determined as follows:

Lemma 11.
NEW_NAMES1 $\quad \Gamma : t \Downarrow^N \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ \Gamma$
NEW_NAMES2 $\quad \Gamma : t \Downarrow^A \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ (\Gamma : t)$

The following *renaming* lemma ensures that the evaluation of a term is "independent" of the fresh names chosen during the reduction process. Furthermore, any name defined in the context heap can be replaced by a fresh one without changing the meaning of the terms evaluated in that context. In fact, reduction proofs for (heap : term) pairs are unique up to α-conversion of the names defined in the heap.

Lemma 12.
RENAMING1 $\quad \Gamma : t \Downarrow^K \Delta : w \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : t) \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Delta : w)$
$\Rightarrow \Gamma[y/x] : t[y/x] \Downarrow^K \Delta[y/x] : w[y/x]$

RENAMING2 $\quad \Gamma : t \Downarrow^K \Delta : w \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : t) \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Delta : w) \wedge x \notin \mathtt{dom}(\Gamma) \wedge x \in \mathtt{dom}(\Delta)$
$\Rightarrow \Gamma : t \Downarrow^K \Delta[y/x] : w[y/x]$

Some of these properties are proved in [14] for $\Downarrow$.[3] The proofs are done by rule induction. In the Appendix we extend the proofs for the rules ALNVar and ALNApp.

[3] Actually, in [14] only RENAMING1 appears.

4 Indirections

The aim of this section is to prove the equivalence of NNS and ANS. Each semantics yields a different heap after evaluating the same (heap : term) pair, and it is necessary to analyze their differences, which lie in the introduced indirections. Recall that a heap represents the context for the evaluation of terms, so that we need to determine when two heaps represent "equivalent" evaluation contexts. More precisely, we show how *indirections*, i.e., bindings that just redirect to another variable name, can be removed while preserving the context represented by the heap.

An *indirection* is a binding of the form $x \mapsto \texttt{fvar}\ y$, that is, it just redirects to another variable name. The set of indirections of a given heap is defined as follows:

$$\texttt{Ind}(\emptyset) = \emptyset \qquad\qquad \texttt{Ind}(\Gamma, x \mapsto t) = \begin{cases} \texttt{Ind}(\Gamma) \cup \{x\} & \text{if } t \equiv \texttt{fvar}\ y \\ \texttt{Ind}(\Gamma) & \text{otherwise} \end{cases}$$

Obviously, $\texttt{Ind}(\Gamma) \subseteq \texttt{dom}(\Gamma)$.

The next example illustrates how these indirections are introduced in the heap during the evaluation of terms.

Example 2. Let us evaluate the term $t \equiv \texttt{app}\ (\texttt{abs}\ (\texttt{bvar}\ 0\ 0))\ x$, in the context $\Gamma = \{x \mapsto \texttt{abs}\ (\texttt{bvar}\ 0\ 0)\}$. Reductions obtained with the NNS and the ANS are:

$$\Gamma : t \Downarrow^N \{x \mapsto \texttt{abs}\ (\texttt{bvar}\ 0\ 0)\} : \texttt{abs}\ (\texttt{bvar}\ 0\ 0)$$
$$\Gamma : t \Downarrow^A \{x \mapsto \texttt{abs}\ (\texttt{bvar}\ 0\ 0), y \mapsto \texttt{fvar}\ x\} : \texttt{abs}\ (\texttt{bvar}\ 0\ 0)$$

The value produced is exactly the same in both cases. However, when comparing the final heap in $\Downarrow^N$ with the final heap in $\Downarrow^A$, the latter contains an extra indirection, $y \mapsto \texttt{fvar}\ x$. This indirection corresponds to the binding introduced by the ALNApp rule to reduce the application in the term t.

The previous example gives us a hint of how to establish a relation between the heaps that are obtained with the NNS and those produced by the ANS. We want two heaps to be related if one can be obtained from the other by just eliminating some indirections. For this purpose we define how to remove indirections from a heap, while preserving the evaluation context represented by this heap. Erasing an indirection $x \mapsto \texttt{fvar}\ y$ from a heap implies not only the elimination of the binding itself, but also the substitution of y for x in the rest of bindings:

$$(\emptyset, x \mapsto \texttt{fvar}\ y) \ominus x = \emptyset \qquad\qquad ((\Gamma, z \mapsto t), x \mapsto \texttt{fvar}\ y) \ominus x = ((\Gamma, x \mapsto \texttt{fvar}\ y) \ominus x, z \mapsto t[y/x])$$

Since substitution preserves the structure of terms (Lemma 1), when erasing an indirection from a heap, the rest of indirections in the heap remain as indirections. Therefore, we generalize our definition to remove a list of indirections from a heap:

$$\Gamma \ominus [\,] = \Gamma \qquad\qquad \Gamma \ominus [x : \overline{x}] = (\Gamma \ominus x) \ominus \overline{x}$$

4.1 Context equivalence

In order to identify heaps, an equivalence relation in *LNHeap* must be defined. This relation is based on the equivalence of terms. The meaning of a term depends on the meaning of its free variables. However, if a free variable is undefined in the context of evaluation of a term, then the name of this free variable is irrelevant. Therefore, we consider that two terms are equivalent in a given context if they only differ in the names of the free variables that do not belong to the context.

Definition 2. *Let $V \subseteq Id$, and $t, t' \in LNExp$. We say that t and t' are* context-equivalent in V*, written $t \approx^V t'$, when:*

$$\text{CE-BVAR}\quad \frac{}{(\texttt{bvar}\ i\ j) \approx^V (\texttt{bvar}\ i\ j)} \qquad \text{CE-FVAR}\quad \frac{x, x' \notin V \vee x = x'}{(\texttt{fvar}\ x) \approx^V (\texttt{fvar}\ x')}$$

$$\text{CE-ABS}\quad \frac{t \approx^V t'}{(\texttt{abs}\ t) \approx^V (\texttt{abs}\ t')} \qquad \text{CE-APP}\quad \frac{t \approx^V t' \quad v \approx^V v'}{(\texttt{app}\ t\ v) \approx^V (\texttt{app}\ t'\ v')}$$

$$\text{CE-LET}\quad \frac{|\overline{t}| = |\overline{t'}| \quad \overline{t} \approx^V \overline{t'} \quad t \approx^V t'}{(\texttt{let}\ \overline{t}\ \texttt{in}\ t) \approx^V (\texttt{let}\ \overline{t'}\ \texttt{in}\ t')}$$

where $\overline{t} \approx^V \overline{t'} = \texttt{List.forall2}\ (\cdot \approx^V \cdot)\ \overline{t}\ \overline{t'}$.

Fixed a set of names V, $\approx^V$ is an equivalence relation on *LNExp*:

Proposition 2.

CE_REF	$t \approx^V t$
CE_SYM	$t \approx^V t' \Rightarrow t' \approx^V t$
CE_TRANS	$t \approx^V t' \wedge t' \approx^V t'' \Rightarrow t \approx^V t''$

If two terms are context-equivalent then they must have the same structure:

Lemma 13.

CE_SS	$t \approx^V t' \Rightarrow t \sim_S t'$

Context-equivalence is preserved in smaller contexts, but also in contexts extended with fresh names:

Lemma 14.

CE_SUB	$t \approx^V t' \wedge V' \subseteq V \Rightarrow t \approx^{V'} t'$
CE_ADD	$t \approx^V t' \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ t \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ t' \Rightarrow t \approx^{V \cup \overline{x}} t'$

Under some conditions context-equivalence is preserved by variable opening and substitution:

Lemma 15.

CE_SUBS1	$t \approx^V t' \wedge x, y \notin V \Rightarrow t[y/x] \approx^V t'$
CE_SUBS2	$t \approx^V t' \wedge (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \wedge x \in V \Rightarrow t[y/x] \approx^V t'[y'/x]$
CE_SUBS3	$t \approx^V t' \wedge y \notin V \wedge \texttt{fresh}\ y\ \texttt{in}\ t \wedge \texttt{fresh}\ y\ \texttt{in}\ t' \Rightarrow t[y/x] \approx^{V[y/x]} t'[y/x]$
CE_OP1	$t \approx^V t' \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{x}}$
CE_OP2	$t \approx^V t' \wedge \overline{x}, \overline{y} \notin V \wedge \lvert\overline{x}\rvert = \lvert\overline{y}\rvert \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{y}}$
CE_OP3	$t \approx^V t' \wedge (\texttt{fvar}\ x) \approx^V (\texttt{fvar}\ y) \Rightarrow t^x \approx^V t'^y$

Based on this equivalence on terms, we define a family of relations on (well-formed) heaps:

Definition 3. *Let $V \subseteq Id$, and $\Gamma, \Gamma' \in LNHeap$. We say that Γ and Γ' are* heap-context-equivalent in V*, written $\Gamma \approx^V \Gamma'$, when:*

$$\text{HCE-EMPTY}\ \frac{}{\emptyset \approx^V \emptyset} \qquad \text{HCE-CONS}\ \frac{\Gamma \approx^V \Gamma' \quad t \approx^V t' \quad \texttt{lc}\ t \quad x \notin \texttt{dom}(\Gamma)}{(\Gamma, x \mapsto t) \approx^V (\Gamma', x \mapsto t')}$$

These relations are in fact equivalences for well-formed heaps:

Proposition 3.

HCE_REF	$\texttt{ok}\ \Gamma \Rightarrow \Gamma \approx^V \Gamma$
HCE_SYM	$\Gamma \approx^V \Gamma' \Rightarrow \Gamma' \approx^V \Gamma$
HCE_TRANS	$\Gamma \approx^V \Gamma' \wedge \Gamma' \approx^V \Gamma'' \Rightarrow \Gamma \approx^V \Gamma''$

Moreover, when two heaps are heap-context-equivalent in some set of identifiers, then both are well-formed, have the same domain, and have the same indirections.

Lemma 16.
HCE_DOM $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma')$
HCE_IND $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{Ind}(\Gamma) = \mathtt{Ind}(\Gamma')$
HCE_OK $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma'$

There is an alternative characterization for heap-context-equivalence which expresses that two heaps are context-equivalent whenever they have the same domain and each pair of corresponding bound terms is context-equivalent.

Lemma 17.
HCE_ALT $\quad \Gamma \approx^V \Gamma' \Leftrightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma' \wedge \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \wedge (x \mapsto t \in \Gamma \wedge x \mapsto t' \in \Gamma' \Rightarrow t \approx^V t')$

Next results guarantee uniqueness up to permutations of sequence of indirections that makes two heaps be equivalent. The order in which two indirections are removed from a heap can be exchanged and the produced heaps are context-equivalent.

Lemma 18.
HCE_SWAP $\quad \mathtt{ok}\ \Gamma \wedge x, y \in \mathtt{Ind}(\Gamma) \wedge x \neq y \Rightarrow \Gamma \ominus [x, y] \approx^{V-\{x,y\}} \Gamma \ominus [y, x]$

Considering context-equivalence on heaps, we are particularly interested in the case where the context coincides with the domain of the heaps:

Definition 4. *Let $\Gamma, \Gamma' \in LNHeap$. We say that Γ and Γ' are* heap-equivalent*, written $\Gamma \approx \Gamma'$, if they are heap-context-equivalent in* $\mathtt{dom}(\Gamma)$*:*

$$\text{HE}\quad \frac{\Gamma \approx^{\mathtt{dom}(\Gamma)} \Gamma'}{\Gamma \approx \Gamma'}$$

Lemma 18 can be generalized to a list of indirections and its permutations.

Lemma 19.
HE_PERM $\quad \mathtt{ok}\ \Gamma \wedge \overline{x}, \overline{y} \subseteq \mathtt{Ind}(\Gamma) \Rightarrow (\Gamma \ominus \overline{x} \approx \Gamma \ominus \overline{y} \Leftrightarrow \overline{y} \in \mathcal{S}(\overline{x}))$

where $\mathcal{S}(\overline{x})$ denotes the set of all permutations of $\overline{x}$.

4.2 Indirection Relation

Coming back to the idea, shown in Example 2, that a heap can be obtained from another by just removing some indirections, we define the following relation on heaps:

Definition 5. *Let $\Gamma, \Gamma' \in LNHeap$. Γ is* indirection-related *to Γ', written $\Gamma \gtrsim_I \Gamma'$, when:*

$$\text{IR-HE}\ \frac{\Gamma \approx \Gamma'}{\Gamma \gtrsim_I \Gamma'} \qquad \text{IR-IR}\ \frac{\mathtt{ok}\ \Gamma \quad \Gamma \ominus x \gtrsim_I \Gamma' \quad x \in \mathtt{Ind}(\Gamma)}{\Gamma \gtrsim_I \Gamma'}$$

There is an alternative characterization for the relation $\gtrsim_I$ which expresses that a heap is indirection-related to another whenever the later can be obtained from the former by erasing a sequence of indirections.

Proposition 4.
IR_ALT $\quad \Gamma \gtrsim_I \Gamma' \Leftrightarrow \mathtt{ok}\ \Gamma \wedge \exists\ \overline{x} \subseteq \mathtt{Ind}(\Gamma)\ .\ \Gamma \ominus \overline{x} \approx \Gamma'$

Furthermore, this sequence of indirections is unique up to permutations (by Lemma 19), and it corresponds to the difference between the domains of the related heaps:

Corollary 1.
IR_DOM_DOM $\quad \Gamma \gtrsim_I \Gamma' \Rightarrow \Gamma \ominus (\mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')) \approx \Gamma'$ [4]

The *indirection-relation* is a preorder on the set of well-formed heaps:

Proposition 5.

IR_REF $\quad \mathtt{ok}\ \Gamma \Rightarrow \Gamma \succsim_I \Gamma$

IR_TRANS $\quad \Gamma \succsim_I \Gamma' \wedge \Gamma' \succsim_I \Gamma'' \Rightarrow \Gamma \succsim_I \Gamma''$

Additionally, the *indirection-relation* satisfies the following properties:

Lemma 20.

IR_DOM $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{dom}(\Gamma') \subseteq \mathtt{dom}(\Gamma)$

IR_IND $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{Ind}(\Gamma') \subseteq \mathtt{Ind}(\Gamma)$

IR_OK $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma'$

IR_DOM_HE $\quad \Gamma \succsim_I \Gamma' \wedge \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \Rightarrow \Gamma \approx \Gamma'$

IR_IR_HE $\quad (\Gamma \succsim_I \Gamma' \wedge \Gamma' \succsim_I \Gamma) \Leftrightarrow \Gamma \approx \Gamma'$

Since $\approx$ is an equivalence relation (Proposition 3), the set of well-formed heaps can be partitioned into mutually exclusive equivalence classes: $[\Gamma] = \{\Gamma' \in \mathit{LNHeap} \mid \Gamma \approx \Gamma'\}$. The *quotien set* is $\mathit{LNHeap}/\approx = \{[\Gamma] \mid \Gamma \in \mathit{LNHeap}\}$. The indirection-relation over heap-equivalence classes is defined as $[\Gamma] \succsim_I [\Gamma'] = \Gamma \succsim_I \Gamma'$. This definition is correct, i.e., it does not depend on the chosen representative of the class, and defines a partial order in $\mathit{LNHeap}/\approx$.

Lemma 21.

IREQ_HE_IREQ1 $\quad [\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma \Rightarrow [\Delta] \succsim_I [\Gamma']$

IREQ_HE_IREQ2 $\quad [\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma' \Rightarrow [\Gamma] \succsim_I [\Delta]$

Proposition 6.

IREQ_REF $\quad \mathtt{ok}\ \Gamma \Rightarrow [\Gamma] \succsim_I [\Gamma]$

IREQ_ANTSYM $\quad [\Gamma] \succsim_I [\Gamma'] \wedge [\Gamma'] \succsim_I [\Gamma] \Rightarrow [\Gamma] = [\Gamma']$

IREQ_TRANS $\quad [\Gamma] \succsim_I [\Gamma'] \wedge [\Gamma'] \succsim_I [\Gamma''] \Rightarrow [\Gamma] \succsim_I [\Gamma'']$

The *indirection-relation* can be extended to (heap : term) pairs.

Definition 6. *Let $\Gamma, \Gamma' \in \mathit{LNHeap}$, and $t, t' \in \mathit{LNExp}$. We say that $(\Gamma : t)$ is* indirection-related *to $(\Gamma' : t')$, written $(\Gamma : t) \succsim_I (\Gamma' : t')$, when*

$$\frac{\forall z \notin L \subseteq \mathit{Id} \quad (\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t')}{(\Gamma : t) \succsim_I (\Gamma' : t')}$$

Example 3. Let us consider the following heap and term:

$$\Gamma = \{x_0 \mapsto \mathtt{fvar}\ x_1, x_1 \mapsto \mathtt{abs}\ (\mathtt{bvar}\ 0\ 0), x_2 \mapsto \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ (\mathtt{bvar}\ 0\ 0)), y_0 \mapsto \mathtt{fvar}\ x_2\}$$
$$t\ = \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ \mathtt{bvar}\ 0\ 0)$$

The (heap : term) pairs related with $(\Gamma : t)$ by removing the sequences of indirections $[\,]$, $[y_0]$, $[x_0]$, and $[x_0, y_0]$ are, respectively, the following:

(*a*) $\{x_0 \mapsto \mathtt{fvar}\ x_1, x_1 \mapsto \mathtt{abs}\ (\mathtt{bvar}\ 0\ 0), x_2 \mapsto \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ (\mathtt{bvar}\ 0\ 0)), y_0 \mapsto \mathtt{fvar}\ x_2\}$
$\quad : \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ (\mathtt{bvar}\ 0\ 0))$

(*b*) $\{x_0 \mapsto \mathtt{fvar}\ x_1, x_1 \mapsto \mathtt{abs}\ (\mathtt{bvar}\ 0\ 0), x_2 \mapsto \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ (\mathtt{bvar}\ 0\ 0))\}$
$\quad : \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_0)\ (\mathtt{bvar}\ 0\ 0))$

(*c*) $\{x_1 \mapsto \mathtt{abs}\ (\mathtt{bvar}\ 0\ 0), x_2 \mapsto \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_1)\ (\mathtt{bvar}\ 0\ 0)), y_0 \mapsto \mathtt{fvar}\ x_2\}$
$\quad : \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_1)\ (\mathtt{bvar}\ 0\ 0))$

(*d*) $\{x_1 \mapsto \mathtt{abs}\ (\mathtt{bvar}\ 0\ 0), x_2 \mapsto \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_1)\ (\mathtt{bvar}\ 0\ 0))\}$
$\quad : \mathtt{abs}\ (\mathtt{app}\ (\mathtt{fvar}\ x_1)\ (\mathtt{bvar}\ 0\ 0))$

Notice that in Example 1 the (heap : term) pair obtained with the ANS is indirection-related to the pair obtained with the NNS, by just removing the indirection $y \mapsto \texttt{fvar}\ x$.

The indirection-relation over (heap : term) pairs satisfies the following properties:

Lemma 22.

IRHT_IRH $(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \Gamma \succsim_I \Gamma'$

IRHT_SS $(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow t \sim_S t'$

IRHT_LC $(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \texttt{lc}\ t \wedge \texttt{lc}\ t'$

4.3 Equivalence

Now we are ready to establish the desired equivalence between the NNS and the ANS in the sense that if a reduction proof can be obtained with the ANS for some term in a given context heap, then there must exist a reduction proof in the NNS for that same (heap : term) pair such that the final (heap : value) is indirection-related to the final (heap : value) obtained with the ANS, and vice versa.

Theorem 1.

EQ_AN $\Gamma : t \Downarrow^A \Delta_A : w_A \Rightarrow$
$\exists \Delta_N \in \mathit{LNHeap} \,.\, \exists w_N \in \mathit{LNVal} \,.\, \Gamma : t \Downarrow^N \Delta_N : w_N \wedge (\Delta_A : w_A) \succsim_I (\Delta_N : w_N)$

EQ_NA $\Gamma : t \Downarrow^N \Delta_N : w_N \Rightarrow$
$\exists \Delta_A \in \mathit{LNHeap} \,.\, \exists w_A \in \mathit{LNVal} \,.\, \exists \overline{x} \subseteq \texttt{dom}(\Delta_N) - \texttt{dom}(\Gamma) \,.\, \exists \overline{y} \subseteq \mathit{Id} \,.\, |\overline{x}| = |\overline{y}| \wedge$
$\Gamma : t \Downarrow^A \Delta_A : w_A \wedge (\Delta_A : w_A) \succsim_I (\Delta_N[\overline{y}/\overline{x}] : w_N[\overline{y}/\overline{x}])$

Notice that in the second part of the theorem, i.e., from NNS to ANS, a renaming may be needed. This renaming only affects names that are added to the heap during the reduction process. This is due to the fact that in the NNS names occurring in the evaluation term (that is t in the theorem) may disappear during the evaluation and, thus, may be chosen on some application of the rule LNLet and added to the final heap. This cannot happen in the ANS (new_names2 in Lemma 11).

To prove this theorem by rule induction, a generalization is needed. Instead of evaluating the same term in the same initial heap, we consider indirection-related initial (heap : term) pairs:

Proposition 7.

EQ_IR_AN $(\Gamma_A : t_A) \succsim_I (\Gamma_N : t_N) \wedge$
$\forall \overline{x} \notin L \subseteq \mathit{Id} \,.\, \Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w_A{}^{\overline{x}} \wedge \ {}^{\backslash \overline{x}}(\overline{s}_A{}^{\overline{x}}) = \overline{s}_A \wedge \ {}^{\backslash \overline{x}}(w_A{}^{\overline{x}}) = w_A$
$\Rightarrow \exists \overline{y} \notin L \,.\, \exists \overline{s}_N \subset \mathit{LNExp} \,.\, \exists w_N \in \mathit{LNVal} \,.$
$\Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}} \wedge \ {}^{\backslash \overline{z}}(\overline{s}_N{}^{\overline{z}}) = \overline{s}_N \wedge \ {}^{\backslash \overline{z}}(w_N{}^{\overline{z}}) = w_N \wedge \overline{z} \subseteq \overline{y}$
$\wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$

EQ_IR_NA $(\Gamma_A : t_A) \succsim_I (\Gamma_N : t_N) \wedge$
$\forall \overline{x} \notin L \subseteq \mathit{Id} \,.\, \Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : w_N{}^{\overline{x}} \wedge \ {}^{\backslash \overline{x}}(\overline{s}_N{}^{\overline{x}}) = \overline{s}_N \wedge \ {}^{\backslash \overline{x}}(w_N{}^{\overline{x}}) = w_N$
$\Rightarrow \exists \overline{z} \notin L \,.\, \exists \overline{s}_A \subset \mathit{LNExp} \,.\, \exists w_A \in \mathit{LNVal} \,.$
$\Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}} \wedge \ {}^{\backslash \overline{y}}(\overline{s}_A{}^{\overline{y}}) = \overline{s}_A \wedge \ {}^{\backslash \overline{y}}(w_A{}^{\overline{y}}) = w_A \wedge \overline{z} \subseteq \overline{y}$
$\wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$

Once again, cofinite quantification replaces freshness conditions. For instance, in EQ_IR_NA it is required that the names introduced during the reduction for the NNS do not collide with names that are already defined in the initial heap for the ANS. The cofinite quantification expresses that if there is an infinite number of "similar" reduction proofs for $(\Gamma_N : t_N)$, each introducing different names in the heap, one can chose a reduction proof such that the new bindings do not interfere with $(\Gamma_A : t_A)$.

Since there is no update in both ANS and NNS (see Lemma 9), a final heap is expressed as the initial heap plus some set of bindings, such as $(\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}})$. In this case, $\overline{x}$ represents the list of new names, i.e., those that have been added during the reduction of local declarations. Since the terms bound to these new

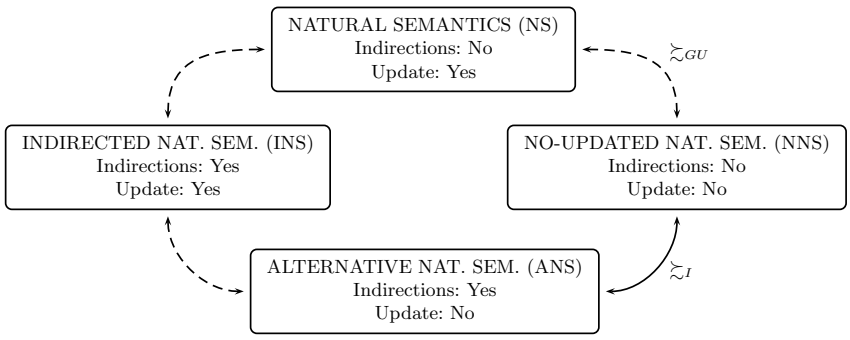

Fig. 7. The relations between the semantics

names are dependent on $\overline{x}$, they are represented as ${\overline{s}_A}^{\overline{x}}$. Similarly for the final value ${w_A}^{\overline{x}}$. The proposition indicates that it is possible to construct reductions for the NNS whose set of new defined names is a subset of the set of new names of the corresponding ANS reduction (which introduces new names when local declarations are evaluated by the rule LNLet, and also when indirections are created by the rule ALNApp).

5 Conclusions and Future Work

Launchbury natural semantics (NS) has turned out to be too much sensitive to the changes introduced by the alternative semantics (ANS), i.e., indirections and no-update. Intuitively, these changes should lead to the same values. However this cannot be directly established since values may contain free variables which are dependent on the context of evaluation, represented by the heap. And, precisely, the changes introduced by the ANS do affect deeply the heaps. In fact, the equivalence of the values produced by the NS and the ANS is based on their correctness with respect to a denotational semantics. Although indirections and duplicated bindings (consequence of the no-update) do not add new information to the heap, it is not straightforward to prove it formally.

Since the variations introduced by Launchbury in the ANS do affect two rules, i.e. the variable rule (no update) and the application rule (indirections), we have defined two intermediate semantics to deal separately with the effect of each modification: The NNS (without update) and the INS (with indirections). A schema of the semantics and how to related them is included in Figure 7.

In the present work we have compared the NNS with the ANS, that is, substitution vs. indirections. We have started by defining an equivalence $\approx$ such that two heaps are considered equivalent when they have the same domain and the corresponding closures may differ only in the free variables not defined in the heaps. We have used this equivalence to define a preorder $\succsim_I$ expressing that a heap can be transformed into another by eliminating indirections. Furthermore, the relation has been extended to (heap : terms) pairs, expressing that two terms can be considered equivalent when they have the same structure and their free variables (only those defined in the context of the corresponding heap) are the same except for some indirections. We have used this extended relation to establish the equivalence between the NNS and the ANS (Theorem 1).

Presently we are working on the equivalence of the NS and the NNS, which will close the path from the NS to the ANS. In order to compare the NS with the NNS, that is, update vs. no update, new relations on heaps and terms have to be defined. The no update of bindings in the heap corresponds in fact to a call-by-name strategy, and implies the duplication of evaluation work, that leads to the generation of duplicated bindings. These duplicated bindings come from several evaluations of the same `let`-declarations, so that they form *groups* of equivalent bindings. Therefore, we first define a preorder $\succsim_G$ that relates two

heaps whenever the first can be transformed into the second by eliminating duplicated groups of bindings. Afterwards, we define a relation $\sim_U$ that establishes when a heap is an updated version of another heap. Finally, both relations must be combined to obtain the *group-update* relation $\precsim_{GU}$, which extended for (heap : terms) will allow us to formulate an equivalence theorem for the NS and the NNS, similar to Theorem 1.

Although the relations $\precsim_I$ and $\precsim_{GU}$ are sufficient for proving the equivalence of the NS and the ANS, it would be interesting to complete the picture by comparing the NS with the INS, and then the INS with the AN. For the first step, we have to define a preorder similar to $\precsim_I$, but taking into account that extra indirections may now be updated, thus leading to "redundant" bindings. For the second step, some version of the group-update relation is needed. Dashed lines indicate this future work.

We have used a locally nameless representation to avoid problems with α-equivalence, while keeping a readable formalization of the syntax and semantics. This representation allow us to treat with heaps in a convenient and easy way, avoiding the problems that arise when using the de Bruijn notation. We have also introduced cofinite quantification (in the style of [1]) in the evaluation rules that introduce fresh names, namely the rule for local declarations (LNLet) and for the alternative application (ALNApp). Moreover, this representation is more amenable to formalization in proof assistants. In fact we have started to implement the semantic rules given in Section 3.2 using Coq [4], with the intention of obtaining a formal checking of our proofs. At this point we should mention the work of Joachim Breitner who is working on the formalization of the correctness and adequacy of Launchbury's semantics by using Isabelle [10]. So far he has been able to mechanically verify the correctness of the operational semantics with respect to the denotational one [5].

6 Acknowledgments

This work is partially supported by the projects: TIN2012-39391-C04-04 and S2009/TIC-1465.

References

1. B. E. Aydemir, A. Charguéraud, B. C. Pierce, R. Pollack, and S. Weirich. Engineering formal metatheory. In *Proceedings of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, pages 3–15. ACM Press, 2008.
2. C. Baker-Finch, D. King, and P. W. Trinder. An operational semantics for parallel lazy evaluation. In *ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pages 162–173. ACM Press, 2000.
3. H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.
4. Y. Bertot. Coq in a hurry. *CoRR*, abs/cs/0603118, 2006.
5. J. Breitner. The correctness of Launchbury's natural semantics for lazy evaluation. Archive of formal proofs, `http://afp.sf.net/entries/Launchbury`, 2013. Formal proof development.
6. A. Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, 49(3):363–408, 2012.
7. N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 75(5):381–392, 1972.
8. J. Launchbury. A natural semantics for lazy evaluation. In *Proceedings of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'93)*, pages 144–154. ACM Press, 1993.
9. K. Nakata and M. Hasegawa. Small-step and big-step semantics for call-by-need. *Journal of Functional Programming*, 19(6):699–722, 2009.
10. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. `http://isabelle.in.tum.de`.
11. L. Sánchez-Gil, M. Hidalgo-Herrero, and Y. Ortega-Mallén. Call-by-need, call-by-name, and natural semantics. Technical Report UU-CS-2010-020, Departament of Information and Computing Sciences, Utrech University, 2010.
12. L. Sánchez-Gil, M. Hidalgo-Herrero, and Y. Ortega-Mallén. *Trends in Functional Programming*, volume 10, chapter An Operational Semantics for Distributed Lazy Evaluation, pages 65–80. Intellect, 2010.

13. L. Sánchez-Gil, M. Hidalgo-Herrero, and Y. Ortega-Mallén. A locally nameless representation for a natural semantics for lazy evaluation. In *Theoretical Aspects of Computing (ICTAC 2012)*, pages 105–119. LNCS 7521, Springer, 2012.
14. L. Sánchez-Gil, M. Hidalgo-Herrero, and Y. Ortega-Mallén. A locally nameless representation for a natural semantics for lazy evaluation (extended version). Technical Report 01/12, Departamento de Sistemas Informáticos y Computación. Universidad Complutense de Madrid, 2012. http://federwin.sip.ucm.es/sic/investigacion/publicaciones/pdfs/SIC-1-12.pdf.
15. P. Sestoft. Deriving a lazy abstract machine. *Journal of Functional Programming*, 7(3):231–264, 1997.
16. C. Urban, S. Berghofer, and M. Norrish. Barendregt's variable convention in rule inductions. In *Proceedings of the 21st International Conference on Automated Deduction (CADE-21)*, pages 35–50. LNCS 4603, Springer-Verlag, 2007.
17. M. van Eekelen and M. de Mol. *Reflections on Type Theory, λ-calculus, and the Mind. Essays dedicated to Henk Barendregt on the Occasion of his 60th Birthday*, chapter Proving Lazy Folklore with Mixed Lazy/Strict Semantics, pages 87–101. Radboud University Nijmegen, 2007.

7 Appendix

Some technical lemmas appear only in this appendix; since they are introduced before the results that need them, numeration is not consecutive. An ordered list can be found at the end of the appendix to facilitate the reading.

7.1 Results in Section 3.1

Proposition 1.

SS_REF $t \sim_S t$

SS_SIM $t \sim_S t' \Rightarrow t' \sim_S t$

SS_TRANS $t \sim_S t' \wedge t \sim_S t'' \Rightarrow t \sim_S t''$

Proof. *Reflexivity*, *symmetry* and *transitivity* are easily proved by induction on the structure of the terms. □

Lemma 1.

SS_SUBST $t[y/x] \sim_S t$

Proof. An easy structural induction on t. □

To prove Lemma 2 we need to prove first a more general result for terms that are opened at any level.

Lemma 23.

SS_OPK $t \sim_S t' \wedge |\overline{x}| = |\overline{y}| \Rightarrow \{k \to \overline{x}\}t \sim_S \{k \to \overline{y}\}t'$

Proof. By structural induction on t:

– $t \equiv \texttt{bvar}\ i\ j$:
$t \sim_S t' \Rightarrow t' \equiv \texttt{bvar}\ i\ j$.
$\{k \mapsto \overline{x}\}(\texttt{bvar}\ i\ j) = \begin{cases} \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{x}) & \text{if } i = k \wedge j < |\overline{x}| \\ \texttt{bvar}\ i\ j & \text{otherwise} \end{cases}$
$\{k \mapsto \overline{y}\}(\texttt{bvar}\ i\ j) = \begin{cases} \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{y}) & \text{if } i = k \wedge j < |\overline{y}| \\ \texttt{bvar}\ i\ j & \text{otherwise} \end{cases}$
If $i = k$ and $j < |\overline{x}| = |\overline{y}|$, then $(\texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{x})) \sim_S (\texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{y}))$, otherwise $(\texttt{bvar}\ i\ j) \sim_S (\texttt{bvar}\ i\ j)$.

- $t \equiv \texttt{fvar}\ x$:
  $t \sim_S t' \Rightarrow t' \equiv \texttt{fvar}\ y$.
  $\{k \mapsto \overline{x}\}(\texttt{fvar}\ x) = \texttt{fvar}\ x \sim_S \texttt{fvar}\ y = \{k \mapsto \overline{y}\}(\texttt{fvar}\ y)$.

- $t \equiv \texttt{abs}\ u$:
  $t \sim_S t' \Rightarrow t' \equiv \texttt{abs}\ u' \wedge u \sim_S u'$
  $\overset{I.H.}{\Rightarrow} \{k+1 \to \overline{x}\}u \sim_S \{k+1 \to \overline{y}\}u'$
  $\Rightarrow \texttt{abs}\ (\{k+1 \to \overline{x}\}u) \sim_S \texttt{abs}\ (\{k+1 \to \overline{y}\}u')$
  $\Rightarrow \{k \to \overline{x}\}(\texttt{abs}\ u) \sim_S \{k \to \overline{y}\}(\texttt{abs}\ u')$.
- $t \equiv \texttt{app}\ u\ v$:
  $t \sim_S t' \Rightarrow t' \equiv \texttt{app}\ u'\ v' \wedge u \sim_S u' \wedge v \sim_S v'$
  $\overset{I.H.}{\Rightarrow} \{k \to \overline{x}\}u \sim_S \{k \to \overline{y}\}u' \wedge \{k \to \overline{x}\}v \sim_S \{k \to \overline{y}\}v$
  $\Rightarrow \texttt{app}\ (\{k \to \overline{x}\}u)\ (\{k \to \overline{x}\}v) \sim_S \texttt{app}\ (\{k \to \overline{y}\}u')\ (\{k \to \overline{y}\}v')$
  $\Rightarrow \{k \to \overline{x}\}(\texttt{app}\ u\ v) \sim_S \{k \to \overline{y}\}(\texttt{app}\ u'\ v')$.
- $t \equiv \texttt{let}\ \overline{t}\ \texttt{in}\ u$:
  $t \sim_S t' \Rightarrow t' \equiv \texttt{let}\ \overline{t'}\ \texttt{in}\ u' \wedge |\overline{t}| = |\overline{t'}| \wedge \overline{t} \sim_S \overline{t'} \wedge u \sim_S u'$
  $\overset{I.H.}{\Rightarrow} |\overline{t}| = |\overline{t'}| \wedge \{k+1 \to \overline{x}\}\overline{t} \sim_S \{k+1 \to \overline{y}\}\overline{t'} \wedge \{k+1 \to \overline{x}\}u \sim_S \{k+1 \to \overline{y}\}u'$
  $\Rightarrow \texttt{let}\ (\{k+1 \to \overline{x}\}\overline{t})\ \texttt{in}\ (\{k+1 \to \overline{x}\}u) \sim_S \texttt{let}\ (\{k+1 \to \overline{y}\}\overline{t'})\ \texttt{in}\ (\{k+1 \to \overline{y}\}u')$
  $\Rightarrow \{k \to \overline{x}\}(\texttt{let}\ \overline{t}\ \texttt{in}\ u) \sim_S \{k \to \overline{y}\}(\texttt{let}\ \overline{t'}\ \texttt{in}\ u')$.

□

Lemma 2.
SS_OP $\quad |\overline{x}| = |\overline{y}| \Rightarrow t^{\overline{x}} \sim_S t^{\overline{y}}$

Proof. By Proposition 1, $t \sim_S t$. A direct consequence of Lemma 23 (take $k = 0$). □

A proof for **Lemma 4** can be found in [14]. In that same technical report there is a proof for a part of **Lemma 3**, that is, $\texttt{fresh}\ \overline{x}\ \texttt{in}\ t \Rightarrow {}^{\backslash\overline{x}}(t^{\overline{x}}) = t$. We prove here the remaining result:

$$^{\backslash\overline{x}}(t^{\overline{x}}) = t \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ t.$$

For this we need first to generalize to terms opened at any level.

Lemma 24.
OPK_CLK_FRESH $\quad \{k \leftarrow \overline{x}\}(\{k \to \overline{x}\}t) = t \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ t$

Proof. By structural induction on t:

- $t \equiv \texttt{bvar}\ i\ j$: Immediate.
- $t \equiv \texttt{fvar}\ z$:
  $\{k \leftarrow \overline{x}\}(\{k \to \overline{x}\}(\texttt{fvar}\ z)) = \texttt{fvar}\ z$ only if $\forall j : 0 \leq j < |\overline{x}|.z \neq \texttt{List.nth}\ j\ \overline{x} \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\texttt{fvar}\ z)$.
- $t \equiv \texttt{abs}\ u$:
  $\{k \leftarrow \overline{x}\}(\{k \to \overline{x}\}(\texttt{abs}\ u)) = \{k \leftarrow \overline{x}\}(\texttt{abs}\ (\{k+1 \to \overline{x}\}u))$
  $= \texttt{abs}\ (\{k+1 \leftarrow \overline{x}\}(\{k+1 \to \overline{x}\}u)) = \texttt{abs}\ u$
  $\Rightarrow \{k+1 \leftarrow \overline{x}\}(\{k+1 \to \overline{x}\}u) = u$
  $\overset{I.H.}{\Rightarrow} \texttt{fresh}\ \overline{x}\ \texttt{in}\ u \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\texttt{abs}\ u)$

The rest of cases are similar. □

Lemma 5.
SS_LC $\quad t \sim_S t' \wedge \texttt{lc}\ t \Rightarrow \texttt{lc}\ t'$

Proof. By structural induction on t:

– $t \equiv \texttt{abs}\ u$:
$\texttt{lc}\ t \quad \Rightarrow\ \forall x \notin L \subseteq Id.\texttt{lc}\ u^x$.
$t \sim_S t' \ \Rightarrow\ t' \equiv \texttt{abs}\ u' \wedge u \sim_S u' \overset{L23}{\Rightarrow} u^x \sim_S u'^x$ for any $x \in Id$ (by taking $\overline{x} = \overline{y} = [x]$ and $k = 0$)
$\overset{I.H.}{\Rightarrow} \forall x \notin L \subseteq Id.\texttt{lc}\ u'^x \Rightarrow \texttt{lc}\ (\texttt{abs}\ u')$

The rest of cases are either immediate or similar to the one shown above. □

To prove Lemma 6 we need a more general result for opening at an arbitrary level. The definition of *locally close at level k* can be found in [13].

Lemma 25.
LC_OPK_VARS $\quad \texttt{lc\_at}\ k\ \overline{n}\ t \wedge \overline{x} \subseteq Id \Rightarrow \exists s \in LNExp.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ s \wedge \{k \to \overline{x}\}s = t)$

Proof. By structural induction on t:

– $t \equiv \texttt{bvar}\ i\ j$:
$\texttt{lc\_at}\ k\ \overline{n}\ \texttt{bvar}\ i\ j \Rightarrow i < k \Rightarrow \{k \mapsto \overline{x}\}(\texttt{bvar}\ i\ j) = \texttt{bvar}\ i\ j$,
and $\texttt{fresh}\ \overline{x}\ \texttt{in}\ \texttt{bvar}\ i\ j$.
Hence, take $s \equiv \texttt{bvar}\ i\ j$. Notice that $s \in Var$.
– $t \equiv \texttt{fvar}\ y$:
- $y \notin \overline{x}$: Take $s \equiv \texttt{fvar}\ y$.
- $y \in \overline{x}$: Take $s \equiv \texttt{bvar}\ k\ j$.

Notice that in both cases $s \in Var$.
– $t \equiv \texttt{abs}\ t'$:
$\texttt{lc\_at}\ k\ \overline{n}\ \texttt{abs}\ t' \Rightarrow \texttt{lc\_at}\ (k+1)\ [1 : \overline{n}]\ t' \overset{I.H.}{\Rightarrow} \exists s'\,.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ s' \wedge \{k+1 \to \overline{x}\}s' = t')$.
Take $s \equiv \texttt{abs}\ t'$, then $\{k \to \overline{x}\}s = \texttt{abs}\ (\{k+1 \to \overline{x}\}s') = \texttt{abs}\ t' = t$.
– $t \equiv \texttt{app}\ t'\ v$:
$\texttt{lc\_at}\ k\ \overline{n}\ \texttt{app}\ t'\ v \Rightarrow \texttt{lc\_at}\ k\ \overline{n}\ t' \wedge \texttt{lc\_at}\ k\ \overline{n}\ v$
$\overset{I.H.}{\Rightarrow} \exists s'\,.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ s' \wedge \{k \to \overline{x}\}s' = t') \wedge \exists v'\,.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ v' \wedge \{k \to \overline{x}\}v' = v)$.
Take $s \equiv \texttt{app}\ t'\ v'$, then $\{k \to \overline{x}\}s = \texttt{app}\ (\{k \to \overline{x}\}s')\ (\{k \to \overline{x}\}v') = \texttt{app}\ t'\ v = t$.
– $t \equiv \texttt{let}\ \overline{t}\ \texttt{in}\ t'$:
$\texttt{lc\_at}\ k\ \overline{n}\ \texttt{let}\ \overline{t}\ \texttt{in}\ t' \Rightarrow \texttt{lc\_at}\ (k+1)\ [|\overline{t}| : \overline{n}]\ [t' : \overline{t}]$
$\overset{I.H.}{\Rightarrow} \exists \overline{s}\,.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ \overline{s} \wedge \{k+1 \to \overline{x}\}\overline{s} = \overline{t}) \wedge \exists s'\,.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ s' \wedge \{k+1 \to \overline{x}\}s' = t')$.
Take $s \equiv \texttt{let}\ \overline{s}\ \texttt{in}\ s'$, then $\{k \to \overline{x}\}s = \texttt{let}\ (\{k+1 \to \overline{x}\}\overline{s})\ \texttt{in}\ (\{k+1 \to \overline{x}\}s') = \texttt{let}\ \overline{t}\ \texttt{in}\ t' = t$.

□

Lemma 6.
LC_OP_VARS $\quad \texttt{lc}\ t \wedge \overline{x} \subseteq Id \Rightarrow \exists s \in LNExp.\,(\texttt{fresh}\ \overline{x}\ \texttt{in}\ s \wedge s^{\overline{x}} = t)$

Proof. Take $k = 0$ in Lemma 25. □

7.2 Results in Section 3.3

Remember that if not indicated otherwise $\Downarrow^K$ represents $\Downarrow$, $\Downarrow^A$, $\Downarrow^I$ and $\Downarrow^N$.

Lemma 7.
REGULARITY $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \texttt{ok}\ \Gamma \wedge \texttt{lc}\ t \wedge \texttt{ok}\ \Delta \wedge \texttt{lc}\ w$

Proof. This has been proved for $\Downarrow$ in [14] by rule induction. We just need to extend the induction proof for the alternative rules ALNVar and ALNApp:

– ALNVar
By induction hypothesis, $\mathtt{ok}\ (\Gamma, x \mapsto t) \wedge \mathtt{lc}\ t \wedge \mathtt{ok}\ \Delta \wedge \mathtt{lc}\ w$.
By definition $\mathtt{lc}\ (\mathtt{fvar}\ x)$.

– ALNApp
By induction hypothesis, $\mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \wedge \mathtt{ok}\ \Theta \wedge \mathtt{lc}\ (\mathtt{abs}\ u)$ and
$\forall y \notin L.\mathtt{ok}\ (\Theta, y \mapsto \mathtt{fvar}\ x) \wedge \mathtt{lc}\ u^y \wedge \mathtt{ok}\ ([y : \overline{z}] \mapsto \overline{s}^y) \wedge \mathtt{lc}\ w^y$.
Particularly for $z \notin L$, $\mathtt{ok}\ ([z : \overline{z}] \mapsto \overline{s}^z) \wedge \mathtt{lc}\ w^z$.
Since $\mathtt{lc}\ t$ and $\mathtt{lc}\ (\mathtt{fvar}\ x)$, we have $\mathtt{lc}\ (\mathtt{app}\ t\ (\mathtt{fvar}\ x))$ too.

□

Lemma 8.
DEF_NOT_LOST $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Delta)$

Proof. This has been proved for $\Downarrow$ in [14] by rule induction. We extend the proof for the alternative rules ALNVar and ALNApp:

– ALNVar:
By induction hypothesis, $\mathtt{dom}(\Gamma, x \mapsto t) \subseteq \mathtt{dom}(\Delta)$.

– ALNApp:
By induction hypothesis, $\mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Theta)$ and $\forall y \notin L.\mathtt{dom}(\Theta, y \mapsto \mathtt{fvar}\ x) \subseteq \mathtt{dom}([y : \overline{z}] \mapsto \overline{s}^y)$.
Particularly for $z \notin L$, $\mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Theta) \subseteq \mathtt{dom}(\Theta, z \mapsto \mathtt{fvar}\ x) \subseteq \mathtt{dom}([z : \overline{z}] \mapsto \overline{s}^z)$.

□

Lemma 9.
NO_UPDATE $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \Gamma \subseteq \Delta$
where $\Downarrow^K$ represents $\Downarrow^N$ and $\Downarrow^A$

Proof. A very easy rule induction. □

Lemma 10.
ADD_NAMES $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{names}(\Delta : w) \subseteq \mathtt{names}(\Gamma : t) \cup \mathtt{dom}(\Delta)$

Proof. This has been proved for $\Downarrow$ in [14] by rule induction. We extend the proof for the alternative rules ALNVar and ALNApp:

– ALNVar:
$$\begin{aligned}\mathtt{names}(\Delta : w) &\overset{I.H.}{\subseteq} \mathtt{names}((\Gamma, x \mapsto t) : t) \cup \mathtt{dom}(\Delta)\\ &= \mathtt{names}(\Gamma, x \mapsto t) \cup \mathtt{fv}(t) \cup \mathtt{dom}(\Delta)\\ &= \mathtt{names}(\Gamma, x \mapsto t) \cup \mathtt{fv}(t) \cup \{x\} \cup \mathtt{dom}(\Delta)\\ &= \mathtt{names}((\Gamma, x \mapsto t) : \mathtt{fvar}\ x) \cup \mathtt{dom}(\Delta).\end{aligned}$$

– ALNApp:
For any $y \notin L$:
$$\begin{aligned}\mathtt{names}(([y : \overline{z}] \mapsto \overline{s}^y) : w^y) &\overset{I.H.}{\subseteq} \mathtt{names}((\Theta, y \mapsto \mathtt{fvar}\ x) : u^y) \cup \mathtt{dom}([y : \overline{z}] \mapsto \overline{s}^y)\\ &\overset{I.H.}{\subseteq} \mathtt{names}(\Gamma : t) \cup \mathtt{dom}(\Theta) \cup \{x\} \cup \mathtt{dom}([y : \overline{z}] \mapsto \overline{s}^y)\\ &\overset{L8}{\subseteq} \mathtt{names}(\Gamma : t) \cup \{x\} \cup \mathtt{dom}([y : \overline{z}] \mapsto \overline{s}^y)\end{aligned}$$
Particularly for $z \notin L$, $\mathtt{names}(([z : \overline{z}] \mapsto \overline{s}^z) : w^z) \subseteq \mathtt{names}(\Gamma : \mathtt{app}\ t\ (\mathtt{fvar}\ x)) \cup \mathtt{dom}([z : \overline{z}] \mapsto \overline{s}^z)$.

□

To prove Lemma 11 we prove first a result concerning reductions of the ANS:

Lemma 26.
KEEP_NAMES $\quad \Gamma : t \Downarrow^A \Delta : w \Rightarrow \mathtt{fv}(t) \subseteq \mathtt{names}(\Delta : w)$

Proof. By rule induction:

- LNLam: Immediate.
- ALNVar: $\mathtt{fv}(\mathtt{fvar}\ x) = \{x\} \subseteq \mathtt{dom}(\Gamma, x \mapsto t) \overset{L8}{\subseteq} \mathtt{dom}(\Delta) \subseteq \mathtt{names}(\Delta : w)$.
- ALNApp:
  On the one hand, $\Gamma : t \Downarrow^A \Theta : \mathtt{abs}\ u \overset{I.H.}{\Rightarrow} \mathtt{fv}(t) \subseteq \mathtt{names}(\Theta : \mathtt{abs}\ u)$.
  But $\mathtt{names}(\Theta) \overset{L9}{\subseteq} \mathtt{names}([z : \overline{z}] \mapsto \overline{s}^z)$, and
  $\mathtt{fv}(\mathtt{abs}\ u) = \mathtt{fv}(u) \subseteq \mathtt{fv}(u^z) \overset{I.H.}{\subseteq} \mathtt{names}(([z : \overline{z}] \mapsto \overline{s}^z) : w^z)$.
  On the other hand, $x \in \mathtt{names}(\Theta, z \mapsto \mathtt{fvar}\ x) \overset{L9}{\subseteq} \mathtt{names}([z : \overline{z}] \mapsto \overline{s}^z)$.
  Therefore, $\mathtt{fv}(\mathtt{app}\ t\ (\mathtt{fvar}\ x)) \subseteq \mathtt{names}(([z : \overline{z}] \mapsto \overline{s}^z) : w^z)$.
- LNLet:
  $\mathtt{fv}(\overline{t}) \subseteq \mathtt{fv}(\overline{t}^{\overline{y}}) \subseteq \mathtt{names}(\Gamma, \overline{y} \mapsto \overline{t}^{\overline{y}}) \overset{L9}{\subseteq} \mathtt{names}(\overline{y} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{y}})$, and
  $\mathtt{fv}(t) \subseteq \mathtt{fv}(t^{\overline{y}}) \overset{I.H.}{\subseteq} \mathtt{names}((\overline{y} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{y}}) : w^{\overline{y}})$.

□

Lemma 11.
NEW_NAMES1 $\quad \Gamma : t \Downarrow^N \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ \Gamma$
NEW_NAMES2 $\quad \Gamma : t \Downarrow^A \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ (\Gamma : t)$

Proof.
NEW_NAMES1
By rule induction:

- LNLam: Since $\Gamma = \Delta$, the result is immediate.
- ALNVar: Let $y \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma, x \mapsto t) \overset{I.H.}{\Rightarrow} \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma, x \mapsto t)$.
- LNApp: Let $y \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma)$:
  - $y \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Theta) \overset{I.H.}{\Rightarrow} \mathtt{fresh}\ y\ \mathtt{in}\ \Theta \overset{L9}{\Rightarrow} \mathtt{fresh}\ y\ \mathtt{in}\ \Gamma$.
  - $y \in \mathtt{dom}(\Theta) - \mathtt{dom}(\Gamma) \overset{I.H.}{\Rightarrow} \mathtt{fresh}\ y\ \mathtt{in}\ \Gamma$.
- LNLet: Let $x \in \mathtt{dom}(\overline{y} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{y}}) - \mathtt{dom}(\Gamma)$:
  - $x \notin \overline{y} \Rightarrow x \notin \mathtt{dom}(\Gamma, \overline{y} \mapsto \overline{t}^{\overline{y}}) \overset{I.H.}{\Rightarrow} \mathtt{fresh}\ x\ \mathtt{in}\ (\Gamma, \overline{y} \mapsto \overline{t}^{\overline{y}}) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ \Gamma$.
  - $x \in \overline{y} \Rightarrow x \notin L$.
    Suppose that $x \in \mathtt{names}(\Gamma)$, hence, exists $z \mapsto t_z \in \Gamma$ such that $x \in \mathtt{fv}(t_z)$ for some $z \in \mathtt{dom}(\Gamma)$.
    Thus, $\forall \overline{x}^{|\overline{t}|} \notin L\,.\, {s_z}^{\overline{x}} = t_z$ and ${}^{\backslash \overline{x}}({s_z}^{\overline{x}}) = s_z$ for some fixed s_z.
    By Lemma 9, $s_z \in \overline{s}$ such that ${s_z}^{\overline{y}} = t_z$.
    If $x \notin \overline{x} \Rightarrow x \in \mathtt{fv}(s_z)$.
    If $x \in \overline{x} \overset{L3}{\Rightarrow} x \notin \mathtt{fv}(s_z)$.
    But this cannot be, since s_z is fixed.

NEW_NAMES2
By rule induction:

- LNLam: Since $\Gamma = \Delta$, the result is immediate.
- ALNVar: Let $y \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma, x \mapsto t) \overset{I.H.}{\Rightarrow} \mathtt{fresh}\ y\ \mathtt{in}\ ((\Gamma, x \mapsto t) : t)$.
  Moreover, $y \neq x$. Hence, $\mathtt{fresh}\ y\ \mathtt{in}\ ((\Gamma, x \mapsto t) : \mathtt{fvar}\ x)$.

– ALNApp: $t \equiv \texttt{app}\ t'\ (\texttt{fvar}\ x)$, and $\Delta = ([z : \overline{z}] \mapsto \overline{s}^z)$, which by Lemma 9 can be also expressed as $(\Gamma, z \mapsto \texttt{fvar}\ x, \overline{z}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s}_1 +\!\!+ \overline{s}_2)$, with $\Gamma : t' \Downarrow^A (\Gamma, \overline{z}_1 \mapsto \overline{s}_1) : \texttt{abs}\ u$ and $(\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z \mapsto \texttt{fvar}\ x) : u^z \Downarrow^A (\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z \mapsto \texttt{fvar}\ x, \overline{z}_2 \mapsto \overline{s}_2) : w^z$.
Let $y \in \texttt{dom}(\Delta) - \texttt{dom}(\Gamma)$.
case 1: $y = z$.
Therefore, ${}^{\backslash y}(\overline{s}^y) = \overline{s} \wedge {}^{\backslash y}(w^y) = w \overset{L3}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ \overline{s} \wedge \texttt{fresh}\ y\ \texttt{in}\ w$
$\overset{y \notin \overline{z}}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ (([z' : \overline{z}] \mapsto \overline{s}^{z'}) : w^{z'}) = ((\Gamma, z' \mapsto \texttt{fvar}\ x, \overline{z}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s}_1 +\!\!+ \overline{s}_2) : w^{z'})$ for all $z' \notin L \cup \{y\}$, and thus, $\texttt{fresh}\ y\ \texttt{in}\ \Gamma$.
Let us denote as $(\Delta' : w')$ the later (heap : value) pair,
$(\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z' \mapsto \texttt{fvar}\ x) : u^{z'} \Downarrow^A \Delta' : w' \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Delta' : w') \overset{L26}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ u \wedge y \neq x$.
Furthermore, $\Gamma : t' \Downarrow^A (\Gamma, \overline{z}_1 \mapsto \overline{s}_1) : \texttt{abs}\ u \wedge \texttt{fresh}\ y\ \texttt{in}\ ((\Gamma, \overline{z}_1 \mapsto \overline{s}_1) : \texttt{abs}\ u) \overset{L26}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ t' \overset{y \neq x}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ \texttt{app}\ t'\ (\texttt{fvar}\ x)$.
case 2: $y \in \overline{z}_1$.
By induction hypothesis, $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t')$.
On the one hand, $x \in \texttt{dom}(\Gamma) \Rightarrow x \notin \overline{z_1} \Rightarrow x \neq y$;
on the other hand, by the rule $x \notin \texttt{dom}(\Gamma) \Rightarrow x \notin [z : \overline{z}] \Rightarrow x \neq y$.
Therefore, $y \neq x$ at any case and $\texttt{fresh}\ y\ \texttt{in}\ \texttt{app}\ t'\ (\texttt{fvar}\ x)$.
case 3: $y \in \overline{z}_2$.
By induction hypothesis, $\texttt{fresh}\ y\ \texttt{in}\ ((\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z \mapsto \texttt{fvar}\ x) : u^z) \Rightarrow \texttt{fresh}\ y\ \texttt{in}\ \Gamma$.
Also, $\texttt{fresh}\ y\ \texttt{in}\ ((\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z \mapsto \texttt{fvar}\ x) : u^z) \Rightarrow y \neq x \wedge y \notin \texttt{names}((\Gamma, \overline{z}_1 \mapsto \overline{s}_1, z \mapsto \texttt{fvar}\ x) : \texttt{abs}\ u)$
$\overset{L26}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ t'$.
Therefore, $\texttt{fresh}\ y\ \texttt{in}\ \texttt{app}\ t'\ (\texttt{fvar}\ x)$.
– LNLet: $t \equiv \texttt{let}\ \overline{t}\ \texttt{in}\ t'$, and let $x \in \texttt{dom}(\overline{y} +\!\!+ \overline{z} \mapsto \overline{s^y}) - \texttt{dom}(\Gamma)$.
- $x \notin \overline{y} \Rightarrow x \notin \texttt{dom}(\Gamma, \overline{y} \mapsto \overline{t}^{\overline{y}}) \overset{I.H.}{\Rightarrow} \texttt{fresh}\ x\ \texttt{in}\ ((\Gamma, \overline{y} \mapsto \overline{t}^{\overline{y}}) : t'^{\overline{y}}) \Rightarrow \texttt{fresh}\ x\ \texttt{in}\ (\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t')$.
- $x \in \overline{y} \Rightarrow x \notin L \wedge x \notin \overline{z}$.
  ${}^{\backslash \overline{y}}(\overline{s}^{\overline{y}}) = \overline{s} \wedge {}^{\backslash \overline{y}}(w^{\overline{y}}) = w \overset{L3}{\Rightarrow} \texttt{fresh}\ \overline{y}\ \texttt{in}\ \overline{s} \wedge \texttt{fresh}\ \overline{y}\ \texttt{in}\ w$
  $\Rightarrow \texttt{fresh}\ x\ \texttt{in}\ \overline{s} \wedge \texttt{fresh}\ x\ \texttt{in}\ w$
  $\Rightarrow \texttt{fresh}\ x\ \texttt{in}\ ((\overline{x} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{x}}) : w^{\overline{x}})$, for all $\overline{x} \notin L \cup \{x\}$.
  Let us take any $\overline{x} \notin L \cup \{x\}$:
  On the one hand, $\texttt{names}(\Gamma) \subseteq \texttt{names}(\Gamma, \overline{x} \mapsto \overline{t}^{\overline{x}}) \overset{L9}{\subseteq} \texttt{names}(\overline{x} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{x}})$, and thus, $\texttt{fresh}\ x\ \texttt{in}\ \Gamma$.
  On the other hand, $(\Gamma, \overline{x} \mapsto \overline{t}^{\overline{x}}) : t'^{\overline{x}} \Downarrow^A (\overline{x} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{x}}) : w^{\overline{x}} \wedge \texttt{fresh}\ x\ \texttt{in}\ ((\overline{x} +\!\!+ \overline{z} \mapsto \overline{s}^{\overline{x}}) : w^{\overline{x}})$
  $\overset{L26}{\Rightarrow} \texttt{fresh}\ x\ \texttt{in}\ \texttt{let}\ \overline{t}\ \texttt{in}\ t'$;

□

Lemma 12.

renaming1	$\Gamma : t \Downarrow^K \Delta : w \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Delta : w)$ $\Rightarrow \Gamma[y/x] : t[y/x] \Downarrow^K \Delta[y/x] : w[y/x]$
renaming2	$\Gamma : t \Downarrow^K \Delta : w \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Delta : w) \wedge x \notin \texttt{dom}(\Gamma) \wedge x \in \texttt{dom}(\Delta)$ $\Rightarrow \Gamma : t \Downarrow^K \Delta[y/x] : w[y/x]$

Proof.
In [14] only renaming1 appears, but the proof of renaming2 is similar to the one of renaming1. We show here the proof cases (in the rule induction) for the alternative rules.

renaming1
We assume that $y \neq x$, otherwise the result is immediate.

– ALNVar:
$(\Gamma, z \mapsto t) : \texttt{fvar}\ z \Downarrow \Delta : w \Rightarrow (\Gamma, z \mapsto t) : t \Downarrow \Delta : w$;
$\texttt{fresh}\ y\ \texttt{in}\ ((\Gamma, z \mapsto t) : \texttt{fvar}\ z) \Rightarrow \texttt{fresh}\ y\ \texttt{in}\ ((\Gamma, z \mapsto t) : t)$,
so that by induction hypothesis, $(\Gamma[y/x], z[y/x] \mapsto t[y/x]) : t[y/x] \Downarrow \Delta[y/x] : w[y/x]$,
that is, $(\Gamma[y/x], z' \mapsto t[y/x]) : t[y/x] \Downarrow \Delta[y/x] : w[y/x]$, where $z' = z[y/x]$.
By rule ALNVar, $(\Gamma[y/x], z' \mapsto t[y/x]) : \texttt{fvar}\ z' \Downarrow \Delta[y/x] : w[y/x]$
$\Rightarrow (\Gamma, z \mapsto t)[y/x] : (\texttt{fvar}\ z)[y/x] \Downarrow \Delta[y/x] : w[y/x]$.

– ALNApp:
$\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([z : \overline{z}] \mapsto \overline{s}^z) : w^z$
$\Rightarrow (x' \notin \texttt{dom}(\Gamma) \Rightarrow x' \notin [z : \overline{z}]) \wedge\ \Gamma : t \Downarrow \Theta : \texttt{abs}\ u$
$\wedge\ \forall y' \notin L.\ (\Theta, y' \mapsto \texttt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'} \wedge\ {\backslash^y}(\overline{s}^y) = \overline{s} \wedge\ {\backslash^y}(w^y) = w$,
for some finite $L \subseteq Id$ such that $z \notin L$.

$$\begin{array}{ll} \texttt{names}(\Gamma : t) & \subseteq\ \texttt{names}(\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x')) \\ \texttt{names}(\Theta : \texttt{abs}\ u) & \overset{L10}{\subseteq}\ \texttt{dom}(\Theta) \cup \texttt{names}(\Gamma : t) \subseteq \texttt{dom}(\Theta, z \mapsto \texttt{fvar}\ x') \cup \texttt{names}(\Gamma : t) \\ & \overset{L8}{\subseteq}\ \texttt{dom}([z : \overline{z}] \mapsto \overline{s}^z) \cup \texttt{names}(\Gamma : t) \\ & \subseteq\ \texttt{names}(([z : \overline{z}] \mapsto \overline{s}^z) : w^z) \cup \texttt{names}(\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x')). \end{array}$$

By hypothesis, $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x')) \wedge \texttt{fresh}\ y\ \texttt{in}\ (([z : \overline{z}] \mapsto \overline{s}^z) : w^z)$,
so that $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t)\ \wedge\ \texttt{fresh}\ y\ \texttt{in}\ (\Theta : \texttt{abs}\ u)$.
Therefore, by induction hypothesis,

$$\Gamma[y/x] : t[y/x] \Downarrow \Theta[y/x] : \underbrace{(\texttt{abs}\ u)[y/x]}_{\texttt{abs}\ u[y/x]} \tag{1}$$

Also, for any $y' \neq y$ we have $\texttt{fresh}\ y\ \texttt{in}\ ((\Theta, y' \mapsto \texttt{fvar}\ x') : u^{y'})\ \wedge\ \texttt{fresh}\ y\ \texttt{in}\ (([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'})$,
and thus, by induction hypothesis,
$\forall y' \notin L \cup \{y\}.(\Theta, y' \mapsto \texttt{fvar}\ x')[y/x] : (u^{y'})[y/x] \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'})[y/x] : (w^{y'})[y/x]$.
This can be rewritten as

$$\forall y' \notin L'.(\Theta[y/x], y' \mapsto (\texttt{fvar}\ x')[y/x]) : u[y/x]^{y'} \Downarrow ([y' : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^{y'}) : w[y/x]^{y'} \tag{2}$$

where $L' = \begin{cases} L \cup \{y\} & \text{if } x \in L \\ L \cup \{x\} - \{y\} & \text{if } x \notin L \end{cases}$ so that $z[y/x] \notin L'$.

Now we have to check that $\forall y' \notin L'.{\backslash^y}(\overline{s}[y/x]^y) = \overline{s}[y/x] \wedge\ {\backslash^y}(w^y) = w$.
Let $y' \notin L'$:

- $y' = y \Rightarrow x \notin L \Rightarrow {\backslash^x}(\overline{s}^x) = \overline{s} \overset{L3}{\Rightarrow} \texttt{fresh}\ x\ \texttt{in}\ \overline{s} \Rightarrow \overline{s}[y/x] = \overline{s}$;
  but $\texttt{fresh}\ y\ \texttt{in}\ \overline{s} \overset{L3}{\Rightarrow} {\backslash^y}(\overline{s}^y) = \overline{s}$.
- $y' \neq y \Rightarrow y' \notin L \Rightarrow {\backslash^{y'}}(\overline{s}^{y'}) = \overline{s} \overset{L3}{\Rightarrow} \texttt{fresh}\ y'\ \texttt{in}\ \overline{s} \Rightarrow \texttt{fresh}\ y'\ \texttt{in}\ \overline{s}[y/x] \overset{L3}{\Rightarrow} {\backslash^{y'}}(\overline{s}[y/x]^{y'}) = \overline{s}[y/x]$.

Similarly for w.

In order to apply ALNApp and obtain
$\Gamma[y/x] : \texttt{app}\ t[y/x]\ (\texttt{fvar}\ x'[y/x]) \Downarrow ([z[y/x] : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^{z[y/x]}) : w[y/x]^{z[y/x]}$
from (1) and (2), we have to prove that if $x'[y/x] \notin \texttt{dom}(\Gamma[y/x])$ then $x'[y/x] \notin [z : \overline{z}][y/x]$:

- $x' = x \Rightarrow x'[y/x] = y$.
  Assume $y \notin \texttt{dom}(\Gamma[y/x]) \Rightarrow x' = x \notin \texttt{dom}(\Gamma) \overset{\text{hip.}}{\Rightarrow} x' = x \notin [z : \overline{z}]$.
  But $\texttt{fresh}\ y\ \texttt{in}\ ([z : \overline{z}] \mapsto \overline{s}^z)$, hence $y \notin [z : \overline{z}][y/x]$.
- $x' \neq x \Rightarrow x'[y/x] = x'$.
  Assume $x' \notin \texttt{dom}(\Gamma[y/x]) \Rightarrow x' \notin \texttt{dom}(\Gamma) \overset{\text{hip.}}{\Rightarrow} x' \notin [z : \overline{z}]$.
  Since $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x'))$ we have that $y \neq x'$. Therefore, $x' \notin [z : \overline{z}][y/x]$.

RENAMING2
We assume that $y \neq x$, otherwise the result is immediate.
By rule induction:

- ALNVAR:
  The proof is similar to the corresponding case in RENAMING1.

- ALNAPP:
  $\Rightarrow (x' \notin \mathtt{dom}(\Gamma) \Rightarrow x' \notin [z : \overline{z}]) \wedge \ \Gamma : t \Downarrow \Theta : \mathtt{abs}\ u$
  $\wedge\ \forall y' \notin L.\ (\Theta, y' \mapsto \mathtt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'} \wedge {}^{\backslash y}(\overline{s}^{y}) = \overline{s} \wedge {}^{\backslash y}(w^{y}) = w,$
  for some finite $L \subseteq Id$ such that $z \notin L$.

  Following the same steps as in RENAMING1 we obtain that $\mathtt{fresh}\ y\ \mathtt{in}\ (\Theta : \mathtt{abs}\ u)$, and that for any $y' \neq y$ it is $\mathtt{fresh}\ y\ \mathtt{in}\ ((\Theta, y' \mapsto \mathtt{fvar}\ x') : u^{y'}) \ \wedge \ \mathtt{fresh}\ y\ \mathtt{in}\ (([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'})$.
  Moreover, $\mathtt{fresh}\ y\ \mathtt{in}\ (([z : \overline{z}] \mapsto \overline{s}^{z}) : w^{z}) \Rightarrow y \neq z \wedge y \notin \overline{z}$,
  $\mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : \mathtt{app}\ t\ (\mathtt{fvar}\ x')) \Rightarrow y \neq x'$, and
  $x \notin \mathtt{dom}(\Gamma) \wedge x \in \mathtt{dom}([z : \overline{z}] \mapsto \overline{s}^{z}) \Rightarrow x \neq x'$.

  CASE 1: $x \in \mathtt{dom}(\Theta)$
  On the one hand, by hypothesis $x \notin \mathtt{dom}(\Gamma)$, so that by induction hypothesis,

$$\Gamma : t \Downarrow \Theta[y/x] : \underbrace{(\mathtt{abs}\ u)[y/x]}_{\mathtt{abs}\ u[y/x]}$$

  On the other hand, if we define $L' = L \cup \{y\}$ we obtain

$$\forall y' \notin L'.(\Theta, y' \mapsto \mathtt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'}$$

  and by using RENAMING1 we have

$$\forall y' \notin L'.(\Theta, y' \mapsto \mathtt{fvar}\ x')[y/x] : u^{y'}[y/x] \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'})[y/x] : w^{y'}[y/x].$$

  As we are assuming $x \in \mathtt{dom}(\Theta)$, by Lemma 7 we have $\forall y' \notin L'.y' \neq x$; particularly, $z \neq x$. This fact, together with $x \neq x'$ allows to rewrite the last equation as

$$\forall y' \notin L'.(\Theta[y/x], y' \mapsto \mathtt{fvar}\ x') : u[y/x]^{y'} \Downarrow ([y' : \overline{z}][y/x] \mapsto \overline{s}[y/x]^{y'}) : w[y/x]^{y'}.$$

  Take any $y' \notin L' \Rightarrow y' \notin L \Rightarrow {}^{\backslash y}(\overline{s}^{y}) = \overline{s} \overset{L3}{\Rightarrow} \mathtt{fresh}\ y'\ \mathtt{in}\ \overline{s} \overset{y' \neq y}{\Rightarrow} \mathtt{fresh}\ y'\ \mathtt{in}\ \overline{s}[y/x] \overset{L3}{\Rightarrow} {}^{\backslash y}(\overline{s}[y/x]^{y}) = \overline{s}[y/x]$ (similarly for w).

  Furthermore, $z \notin L'$, and $x' \notin \mathtt{dom}(\Gamma) \Rightarrow x' \notin \mathtt{dom}([z : \overline{z}] \mapsto \overline{s}^{z}) \overset{y \neq x'}{\Rightarrow} x' \notin \mathtt{dom}([z : \overline{z}][y/x] \mapsto \overline{s}[y/x]^{z})$.
  Then by rule ALNAPP,

$$\Gamma : \mathtt{app}\ t\ (\mathtt{fvar}\ x') \Downarrow ([z : \overline{z}][y/x] \mapsto \overline{s}[y/x]^{z}) : w[y/x]^{z}$$

  which, as $z \neq x$, can be rewritten to

$$\Gamma : \mathtt{app}\ t\ (\mathtt{fvar}\ x') \Downarrow ([z : \overline{z}] \mapsto \overline{s}^{z})[y/x] : w^{z}[y/x].$$

  CASE 2: $x \notin \mathtt{dom}(\Theta)$
  By hypothesis, $\Gamma : t \Downarrow \Theta : \mathtt{abs}\ u$.
  We have to find a finite set $L' \subseteq Id$ such that $z[y/x] \notin L'$,
  $x' \notin \mathtt{dom}(\Gamma) \Rightarrow x' \notin \mathtt{dom}([z : \overline{z}][y/x] \mapsto \overline{s}^{z}[y/x])$, and

$$\forall y' \notin L'.(\Theta, y' \mapsto \mathtt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^{y'}) : w[y/x]^{y'}$$

  and then apply the rule ALNAPP.

- SUBCASE 2.1: $x \neq z$.
  Let $L' = L \cup \{y\} \cup \{x\}$, then by induction hypothesis,

  $$\forall y' \notin L'.(\Theta, y' \mapsto \texttt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'})[y/x] : w^{y'}[y/x]$$

  which, as $y' \neq x$, can be rewritten to

  $$\forall y' \notin L'.(\Theta, y' \mapsto \texttt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^{y'}) : w[y/x]^{y'}.$$

  Similarly as done in CASE 1, we check that ${}^{\backslash y'}(\overline{s}^{y'}) = \overline{s} \wedge {}^{\backslash y'}(w^{y'}) = w$, for all $y' \notin L'$.
  Furthermore, $z[y/x] \overset{z \neq x}{=} z \notin L \overset{z \neq y}{\Rightarrow} z[y/x] \notin L'$, and
  $x' \notin \texttt{dom}(\Gamma) \Rightarrow x' \notin \texttt{dom}([z : \overline{z}] \mapsto \overline{s}^z) \overset{x' \neq y}{\Rightarrow} x' \notin \texttt{dom}([z : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^z)$.
  Hence, by rule ALNAPP,

  $$\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([z[y/x] : \overline{z}[y/x]] \mapsto \overline{s}[y/x]^{z[y/x]}) : w[y/x]^{z[y/x]}$$

  that is $\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([z : \overline{z}] \mapsto \overline{s}^z)[y/x] : w^z[y/x]$.
- SUBCASE 2.2: $x = z$.
  Therefore, ${}^{\backslash x}(\overline{s}^x) = \overline{s} \wedge {}^{\backslash x}(w^x) = w \overset{L3}{\Rightarrow} \texttt{fresh}\ x\ \texttt{in}\ \overline{s} \wedge \texttt{fresh}\ x\ \texttt{in}\ w$
  $\Rightarrow ([z : \overline{z}] \mapsto \overline{s}^z)[y/x] = ([x : \overline{z}] \mapsto \overline{s}^x)[y/x] = [y : \overline{z}] \mapsto \overline{s}^y$, and $w^z[y/x] = w^x[y/x] = w^y$.
  Furthermore, $x' \notin \texttt{dom}(\Gamma) \Rightarrow x' \notin \texttt{dom}([z : \overline{z}] \mapsto \overline{s}^z) \overset{x' \neq y}{\Rightarrow} x' \notin \texttt{dom}([y : \overline{z}] \mapsto \overline{s}^y)$.
  - $y \notin L$
    Then we are done, because by rule ALNAPP: $\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([y : \overline{z}] \mapsto \overline{s}^y) : w^y$.
  - $y \in L$
    Let $L' = L \cup \{x\} \cup \{x'\} \cup \texttt{names}(\Gamma : t)$.
    Since $L' \subseteq L$ we have $\forall y' \notin L'.(\Theta, y' \mapsto \texttt{fvar}\ x') : u^{y'} \Downarrow ([y' : \overline{z}] \mapsto \overline{s}^{y'}) : w^{y'}$.
    Let us choose some $z' \notin L'$ such as $z' \neq y$.
    Hence, if $x' \notin \texttt{dom}(\Gamma) \Rightarrow x' \notin \texttt{dom}([z : \overline{z}] \mapsto \overline{s}^z) \overset{x' \neq z'}{\Rightarrow} x' \notin \texttt{dom}([z' : \overline{z}] \mapsto \overline{s}^{z'})$.
    Thus, by rule ALNAPP,

    $$\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([z' : \overline{z}] \mapsto \overline{s}^{z'}) : w^{z'}.$$

    By hypothesis, $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x'))$, and
    $\texttt{fresh}\ y\ \texttt{in}\ (([z : \overline{z}] \mapsto \overline{s}^z) : w^z) \overset{y \neq z'}{\Rightarrow} \texttt{fresh}\ y\ \texttt{in}\ (([z' : \overline{z}] \mapsto \overline{s}^{z'}) : w^{z'})$
    So by RENAMING1 we obtain

    $$\Gamma[y/z'] : (\texttt{app}\ t\ (\texttt{fvar}\ x'))[y/z'] \Downarrow ([z' : \overline{z}] \mapsto \overline{s}^{z'})[y/z'] : w^{z'}[y/z'].$$

    Since $z' \notin \texttt{names}(\Gamma : t) \cup \{x'\}$, this is rewritten to $\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x') \Downarrow ([y : \overline{z}] \mapsto \overline{s}^y) : w^y$.

□

The following version of renaming will be useful in some cases.

Corollary 2.
RENAMING3 $\quad \Gamma : t \Downarrow^K \Delta : w \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Delta : w) \wedge x \notin \texttt{names}(\Gamma : t)$
$\Rightarrow \Gamma : t \Downarrow^K \Delta[y/x] : w[y/x]$

Proof. If $x \notin \texttt{names}(\Gamma : t)$ then $(\Gamma[y/x] : t[y/x]) = (\Gamma : t)$. Thus, it is a corollary of RENAMING1. □

7.3 Results on context equivalence (Section 4.1)

Proposition 2.
CE_REF $t \approx^V t$
CE_SYM $t \approx^V t' \Rightarrow t' \approx^V t$
CE_TRANS $t \approx^V t' \wedge t' \approx^V t'' \Rightarrow t \approx^V t''$

Proof.
CE_REF and CE_SYM are proved easily by rule induction.

CE_TRANS
By structural induction on t:

- $t \equiv \texttt{fvar}\ y$.
  $t \approx^V t' \Rightarrow t' \equiv \texttt{fvar}\ y' \wedge (y, y' \notin V \vee y = y')$.
  - $y, y' \notin V$.
    $t' \approx^V t'' \Rightarrow t'' \equiv \texttt{fvar}\ y'' \wedge (y', y'' \notin V \vee y' = y'')$.
    * $y', y'' \notin V \Rightarrow y, y'' \notin V \Rightarrow (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y'')$.
    * $y' = y'' \Rightarrow y'' \notin V \Rightarrow (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y'')$.
  - $y = y'$. Immediate.

The rest of cases are very easy. □

Lemma 13.
CE_SS $t \approx^V t' \Rightarrow t \sim_S t'$

Proof. An easy structural induction on t. □

We add a corollary of the previous lemma which will be useful in forthcoming proofs.

Corollary 3.
CE_LC $t \approx^V t' \wedge \texttt{lc}\ t \Rightarrow \texttt{lc}\ t'$

Proof. By Lemmas 13 and 5. □

Lemma 14.
CE_SUB $t \approx^V t' \wedge V' \subseteq V \Rightarrow t \approx^{V'} t'$
CE_ADD $t \approx^V t' \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ t \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ t' \Rightarrow t \approx^{V \cup \overline{x}} t'$

Proof.

CE_SUB
By rule induction.
The only interesting case is CE-FVAR: $(\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \Rightarrow y, y' \notin V \vee y = y'$.

- $y, y' \notin V \Rightarrow y, y' \notin V' \Rightarrow (\texttt{fvar}\ y) \approx^{V'} (\texttt{fvar}\ y')$.
- $y = y' \Rightarrow (\texttt{fvar}\ y) \approx^{V'} (\texttt{fvar}\ y')$.

CE_ADD
By rule induction.
The only interesting case is CE-FVAR: $(\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \Rightarrow (y, y' \notin V \vee y = y')$.
$\texttt{fresh}\ \overline{x}\ \texttt{in}\ (\texttt{fvar}\ y) \Rightarrow y \notin \overline{x}$, and similarly, $y' \notin \overline{x}$.
Therefore, $(\texttt{fvar}\ y) \approx^{V \cup \overline{x}} (\texttt{fvar}\ y')$. □

To prove Lemma 15 we prove first some results involving the operation of variable opening at level k.

Lemma 27.

CE_OPK1 $t \approx^V t' \Rightarrow \{k \to \overline{x}\}t \approx^V \{k \to \overline{x}\}t'$

CE_OPK2 $t \approx^V t' \wedge \overline{x}, \overline{y} \notin V \wedge |\overline{x}| = |\overline{y}| \Rightarrow \{k \to \overline{x}\}t \approx^V \{k \to \overline{y}\}t'$

Proof.

CE_OPK1
By rule induction:

- CE-ABS : $(\texttt{abs}\ t) \approx^V (\texttt{abs}\ t') \Leftrightarrow t \approx^V t'$.
  $\{k \to \overline{x}\}(\texttt{abs}\ t) = \texttt{abs}\ (\{k+1 \to \overline{x}\}t)$.
  $\{k \to \overline{x}\}(\texttt{abs}\ t') = \texttt{abs}\ (\{k+1 \to \overline{x}\}t')$.
  By induction hypothesis, $\{k+1 \to \overline{x}\}t \approx^V \{k+1 \to \overline{x}\}t'$.
  Hence, $\{k \to \overline{x}\}(\texttt{abs}\ t) \approx^V \{k \to \overline{x}\}(\texttt{abs}\ t')$.

The rest of cases are very easy.

CE_OPK2
By rule induction. The only interesting cases are the rules for bound and free variables:

- CE-BVAR : $(\texttt{bvar}\ i\ j) \approx^V (\texttt{bvar}\ i\ j)$.
  $\{k \to \overline{x}\}(\texttt{bvar}\ i\ j) = \begin{cases} \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{x}) & \text{if } i = k \wedge j < |\overline{x}| \\ \texttt{bvar}\ i\ j & \text{otherwise} \end{cases}$
  $\{k \to \overline{y}\}(\texttt{bvar}\ i\ j) = \begin{cases} \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{y}) & \text{if } i = k \wedge j < |\overline{y}| \\ \texttt{bvar}\ i\ j & \text{otherwise} \end{cases}$
  Since $|\overline{x}| = |\overline{y}|$
  - either $\{k \to \overline{x}\}(\texttt{bvar}\ i\ j) = \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{x})$ and $\{k \to \overline{y}\}(\texttt{bvar}\ i\ j) = \texttt{fvar}\ (\texttt{List.nth}\ j\ \overline{y})$,
  - or $\{k \to \overline{x}\}(\texttt{bvar}\ i\ j) = (\texttt{bvar}\ i\ j)$ and $\{k \to \overline{y}\}(\texttt{bvar}\ i\ j) = (\texttt{bvar}\ i\ j)$.

  Since $\overline{x}, \overline{y} \notin V$, the resulting terms are context equivalent in both cases.
- CE-FVAR : $(\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \Leftrightarrow (y, y' \notin V \vee y = y')$.
  It is immediate, since $\{k \to \overline{x}\}(\texttt{fvar}\ z) = \texttt{fvar}\ z$.

□

Lemma 15.

CE_SUBS1 $t \approx^V t' \wedge x, y \notin V \Rightarrow t[y/x] \approx^V t'$

CE_SUBS2 $t \approx^V t' \wedge (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \wedge x \in V \Rightarrow t[y/x] \approx^V t'[y'/x]$

CE_SUBS3 $t \approx^V t' \wedge y \notin V \wedge \texttt{fresh}\ y\ \texttt{in}\ t \wedge \texttt{fresh}\ y\ \texttt{in}\ t' \Rightarrow t[y/x] \approx^{V[y/x]} t'[y/x]$

CE_OP1 $t \approx^V t' \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{x}}$

CE_OP2 $t \approx^V t' \wedge \overline{x}, \overline{y} \notin V \wedge |\overline{x}| = |\overline{y}| \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{y}}$

CE_OP3 $t \approx^V t' \wedge (\texttt{fvar}\ x) \approx^V (\texttt{fvar}\ y) \Rightarrow t^x \approx^V t'^y$

Proof.

CE_SUBS1
An easy rule induction.

CE_SUBS2
By rule induction:

- CE-FVAR : $(\texttt{fvar}\ z) \approx^V (\texttt{fvar}\ z') \Rightarrow (z, z' \notin V \vee z = z')$.
  CASE 1: $z, z' \notin V \Rightarrow z \neq x \wedge z' \neq x$.
  Hence, $(\texttt{fvar}\ z)[y/x] = (\texttt{fvar}\ z)$ and $(\texttt{fvar}\ z')[y/x] = (\texttt{fvar}\ z')$.
  CASE 2: $z = z'$
  - $z = x \Rightarrow z' = x$
    $(\texttt{fvar}\ z)[y/x] = (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') = (\texttt{fvar}\ z')[y'/x]$.

- $z \neq x \Rightarrow z' \neq x$. Similar to CASE 1.

The rest of cases are immediate.

CE_SUBS3
By rule induction:

- CE-FVAR : $(\texttt{fvar}\ z) \approx^V (\texttt{fvar}\ z') \Rightarrow (z, z' \notin V \vee z = z')$.
  CASE 1: $z, z' \notin V$
  - $z = x \Rightarrow V[y/x] = V \wedge (\texttt{fvar}\ z)[y/x] = \texttt{fvar}\ y \wedge y \notin V \Rightarrow (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ z')$.
  - $z' = x$. Analogous.
  - $z \neq x \wedge z' \neq x \Rightarrow (\texttt{fvar}\ z)[y/x] = \texttt{fvar}\ z \wedge (\texttt{fvar}\ z')[y/x] = \texttt{fvar}\ z'$.
    And $z, z' \notin V \wedge y \neq z \wedge y \neq z' \Rightarrow z, z' \notin V[y/x]$.

  CASE 2: $z = z'$. Immediate.

The rest of cases are immediate.

CE_OP1
Take $k = 0$ in CE_OPK1 (Lemma 27).

CE_OP2
Take $k = 0$ in CE_OPK2 (Lemma 27).

CE_OP3
Since $(\texttt{fvar}\ x) \approx^V (\texttt{fvar}\ x')$, either

- $x = y$, and we use CE_OP1; or,
- $x, y \notin V$, and we use CE_OP2.

□

Lemma 16.

HCE_DOM $\quad \Gamma \approx^V \Gamma' \Rightarrow \texttt{dom}(\Gamma) = \texttt{dom}(\Gamma')$
HCE_IND $\quad \Gamma \approx^V \Gamma' \Rightarrow \texttt{Ind}(\Gamma) = \texttt{Ind}(\Gamma')$
HCE_OK $\quad \Gamma \approx^V \Gamma' \Rightarrow \texttt{ok}\ \Gamma \wedge \texttt{ok}\ \Gamma'$

Proof.
The three properties are proved by rule induction, where the case of the empty heap is immediate.

HCE_DOM
$\Gamma = (\Theta, y \mapsto t) \approx^V \Gamma' = (\Theta', y \mapsto t')$ with $\Theta \approx^V \Theta' \wedge y \notin \texttt{dom}(\Theta)$.
$\texttt{dom}(\Gamma) = \texttt{dom}(\Theta, y \mapsto t) = \texttt{dom}(\Theta) \cup \{y\} \overset{I.H.}{=} \texttt{dom}(\Theta') \cup \{y\} = \texttt{dom}(\Theta', y \mapsto t') = \texttt{dom}(\Gamma')$.

HCE_IND
$\Gamma = (\Theta, y \mapsto t) \approx^V \Gamma' = (\Theta', y \mapsto t')$ with $\Theta \approx^V \Theta' \wedge t \approx^V t' \wedge y \notin \texttt{dom}(\Theta)$.

$$\texttt{Ind}(\Theta, y \mapsto t) = \begin{cases} \texttt{Ind}(\Theta) \cup \{y\} & \text{if } t \equiv \texttt{fvar}\ z \\ \texttt{Ind}(\Theta) & \text{otherwise} \end{cases}$$

$$\texttt{Ind}(\Theta', y \mapsto t') = \begin{cases} \texttt{Ind}(\Theta') \cup \{y\} & \text{if } t' \equiv \texttt{fvar}\ z' \\ \texttt{Ind}(\Theta') & \text{otherwise} \end{cases}$$

But $t \approx^V t' \overset{L13}{\Rightarrow} t \sim_S t'$, therefore, $t \equiv \texttt{fvar}\ z \Leftrightarrow t' \equiv \texttt{fvar}\ z'$.

HCE_OK
$\Gamma = (\Theta, y \mapsto t) \approx^V \Gamma' = (\Theta', y \mapsto t')$ with $\Theta \approx^V \Theta' \wedge t \approx^V t' \wedge \texttt{lc}\ t \wedge y \notin \texttt{dom}(\Theta)$.

$\Theta \approx^V \Theta' \overset{I.H.}{\Rightarrow} \mathtt{ok}\ \Theta \wedge \mathtt{ok}\ \Theta'$.
$\mathtt{lc}\ t \overset{C3}{\Rightarrow} \mathtt{lc}\ t'$.
$y \notin \mathtt{dom}(\Theta) \overset{\text{HCE\_DOM}}{=} \mathtt{dom}(\Theta')$.
Thus, $\mathtt{ok}\ (\Theta, y \mapsto t)$ and $\mathtt{ok}\ (\Theta', y \mapsto t')$. □

To prove that $\approx^V$ is an equivalence on heaps, we prove first that the relation is independent of the order in which the bindings of the heaps are considered. So that when two heaps are related (in some context), all the names defined in those heaps are related.

Lemma 28.
HCE_BIND $\quad (\Gamma, x \mapsto t) \approx^V (\Gamma', x \mapsto t') \Rightarrow \Gamma \approx^V \Gamma' \wedge t \approx^V t'$

Proof. By induction on the size of Γ.

- $\Gamma = \emptyset$.
  $(\emptyset, x \mapsto t) \approx^V (\Gamma', x \mapsto t') \Rightarrow \Gamma = \emptyset \approx^V \Gamma' \wedge t \approx^V t' \wedge \mathtt{lc}\ t$.

- $\Gamma \neq \emptyset$.
  Let $\Delta = (\Gamma, x \mapsto t)$ and $\Delta' = (\Gamma', x \mapsto t')$.
  $\Delta \approx^V \Delta' \Rightarrow \exists y.\ \Delta = (\Theta, y \mapsto t_y) \wedge \Delta' = (\Theta', y \mapsto t'_y) \wedge \Theta \approx^V \Theta' \wedge t_y \approx^V t'_y \wedge \mathtt{lc}\ t_y$.
  If $y = x$ then by HCE_OK in Lemma 16 there is a unique x in Δ and Δ', so that $\Gamma = \Theta \wedge \Gamma' = \Theta'$, and we are done.
  Otherwise, $\Theta = (\Theta_x, x \mapsto t)$ and $\Theta' = (\Theta'_x, x \mapsto t')$, being $(\Theta_x, y \mapsto t_y) = \Gamma$, and hence $\Theta_x \subsetneq \Gamma$.
  By induction hypotesis, $\Theta_x \approx^V \Theta'_x \wedge t \approx^V t' \wedge \mathtt{lc}\ t$.
  Furthermore, $\Theta_x \approx^V \Theta'_x \wedge t_y \approx^V t'_y \wedge \mathtt{lc}\ t_y \Rightarrow \Gamma = (\Theta_x, y \mapsto t_y) \approx^V (\Theta'_x, y \mapsto t'_y) = \Gamma'$.

□

Proposition 3
HCE_REF $\quad \mathtt{ok}\ \Gamma \Rightarrow \Gamma \approx^V \Gamma$
HCE_SYM $\quad \Gamma \approx^V \Gamma' \Rightarrow \Gamma' \approx^V \Gamma$
HCE_TRANS $\quad \Gamma \approx^V \Gamma' \wedge \Gamma' \approx^V \Gamma'' \Rightarrow \Gamma \approx^V \Gamma''$

Proof.
HCE_REF and HCE_SYM are immediate.

HCE_TRANS
By rule induction on $\Gamma \approx^V \Gamma'$:

- HCE_EMPTY. $\emptyset \approx^V \emptyset \Rightarrow \Gamma'' = \emptyset$. Immediate.

- HCE_CONS. $\Gamma = (\Theta, y \mapsto t) \approx^V \Gamma' = (\Theta', y \mapsto t') \Rightarrow \Theta \approx^V \Theta' \wedge t \approx^V t' \wedge \mathtt{lc}\ t \wedge y \notin \mathtt{dom}(\Theta)$
  $\Gamma' \approx^V \Gamma'' \overset{L16}{\Rightarrow} \mathtt{dom}(\Gamma') = \mathtt{dom}(\Gamma'') \Rightarrow \Gamma'' = (\Theta'', y \mapsto t'') \overset{L28}{\Rightarrow} \Theta' \approx^V \Theta'' \wedge t' \approx^V t''$.
  By induction hypothesis, $\Theta \approx^V \Theta''$.
  By CE_TRANS in Proposition 2, $t \approx^V t''$.
  Hence, $\Gamma = (\Theta, y \mapsto t) \approx^V (\Theta'', y \mapsto t'') = \Gamma''$.

□

Lemma 17
HCE_ALT $\quad \Gamma \approx^V \Gamma' \Leftrightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma' \wedge \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \wedge (x \mapsto t \in \Gamma \wedge x \mapsto t' \in \Gamma' \Rightarrow t \approx^V t')$

Proof.
$\Rightarrow$ By HCE_DOM and HCE_OK (Lemma 16) and HCE_BIND (Lemma 28).

$\Leftarrow$ By induction on the size of Γ:

– $\Gamma = \emptyset$. Immediate.
– $\Gamma = (\Theta, y \mapsto t_y) \Rightarrow \Gamma' = (\Theta', y \mapsto t'_y) \wedge \mathtt{dom}(\Theta) = \mathtt{dom}(\Theta') \wedge t_y \approx^V t'_y$.
$\mathtt{ok}\ \Gamma \Rightarrow \mathtt{ok}\ \Theta \wedge y \notin \mathtt{dom}(\Theta) \wedge \mathtt{lc}\ t_y$,
$\mathtt{ok}\ \Gamma' \Rightarrow \mathtt{ok}\ \Theta' \wedge y \notin \mathtt{dom}(\Theta') \wedge \mathtt{lc}\ t'_y$.
If $x \mapsto t \in \Theta \subseteq \Gamma \wedge x \mapsto t' \in \Theta' \subseteq \Gamma'$ then $t \approx^V t'$.
Hence $\mathtt{ok}\ \Theta \wedge \mathtt{ok}\ \Theta' \wedge \mathtt{dom}(\Theta) = \mathtt{dom}(\Theta') \wedge (x \mapsto t \in \Theta \wedge x \mapsto t' \in \Theta' \Rightarrow t \approx^V t')$.
By induction hypothesis, $\Theta \approx^V \Theta'$.
Thus, $\Gamma = (\Theta, y \mapsto t_y) \approx^V (\Theta', x \mapsto t'_y) = \Gamma'$.

□

The following auxiliary result is used in the proofs of the Lemmas 18 and 19:

Lemma 29.
IND_DOM $\quad \mathtt{ok}\ \Gamma \wedge x \in \mathtt{Ind}(\Gamma) \Rightarrow \mathtt{dom}(\Gamma \ominus x) = \mathtt{dom}(\Gamma) - \{x\} \wedge \mathtt{Ind}(\Gamma \ominus x) = \mathtt{Ind}(\Gamma) - \{x\}$

Proof.
An easy induction on the size of Γ. □

We also give here some other properties related to the deletion of indirections. These will be used in forthcoming proofs.

Lemma 30.
IND_OK $\quad \mathtt{ok}\ \Gamma \wedge x \in \mathtt{Ind}(\Gamma) \Rightarrow \mathtt{ok}\ (\Gamma \ominus x)$

Proof. $\Gamma = (\Theta, x \mapsto \mathtt{fvar}\ y)$.
$\mathtt{ok}\ \Gamma \Rightarrow \mathtt{ok}\ \Theta \wedge x \notin \mathtt{dom}(\Theta)$.
By induction on the size of Θ:

– $\Theta = \emptyset$.
$\Gamma \ominus x = \emptyset$. Immediate.
– $\Theta = (\Delta, z \mapsto t)$.
$\Gamma \ominus x = ((\Delta, x \mapsto \mathtt{fvar}\ y) \ominus x, z \mapsto t[y/x])$.
$\mathtt{ok}\ \Theta \Rightarrow \mathtt{ok}\ \Delta \wedge z \notin \mathtt{dom}(\Delta) \wedge \mathtt{lc}\ t$.
$z \notin \mathtt{dom}(\Delta) \overset{L29}{=} \mathtt{dom}((\Delta, x \mapsto \mathtt{fvar}\ y) \ominus x)$.
$\mathtt{lc}\ t \Rightarrow \mathtt{lc}\ t[y/x]$.
$\left.\begin{array}{l} \mathtt{ok}\ \Delta \\ x \notin \mathtt{dom}(\Theta) \Rightarrow x \notin \mathtt{dom}(\Delta) \\ \mathtt{lc}\ (\mathtt{fvar}\ y) \end{array}\right\} \Rightarrow \mathtt{ok}\ (\Delta, x \mapsto \mathtt{fvar}\ y) \overset{I.H.}{\Rightarrow} \mathtt{ok}\ ((\Delta, x \mapsto \mathtt{fvar}\ y) \ominus x)$.
Therefore $\mathtt{ok}\ ((\Delta, x \mapsto \mathtt{fvar}\ y)) \ominus x, z \mapsto t[y/x]) \Rightarrow \mathtt{ok}\ (\Gamma \ominus x)$.

□

Lemma 31.
IND_SUBS $\quad x \notin \mathtt{dom}(\Gamma) \Rightarrow (\Gamma, x \mapsto \mathtt{fvar}\ y) \ominus x = \Gamma[y/x]$

Proof.
An easy induction on the size of Γ. □

Lemma 18
HCE_SWAP $\quad \mathtt{ok}\ \Gamma \wedge x, y \in \mathtt{Ind}(\Gamma) \wedge x \neq y \Rightarrow \Gamma \ominus [x, y] \approx^{V - \{x,y\}} \Gamma \ominus [y, x]$

Proof.
We have $\Gamma = (\Theta, x \mapsto \mathtt{fvar}\ x', y \mapsto \mathtt{fvar}\ y')$.
By induction on the size of Θ:

– $\Theta = \emptyset \Rightarrow \Gamma \ominus [x,y] = \emptyset = \Gamma \ominus [y,x]$
– $\Theta = (\Delta, z \mapsto t) \Rightarrow$
$\Gamma \ominus [x,y] = (\Gamma \ominus x) \ominus y$
$= ((\Delta, x \mapsto \texttt{fvar}\ x', y \mapsto \texttt{fvar}\ y') \ominus x, z \mapsto t[x'/x]) \ominus y$
$= (((\Delta, x \mapsto \texttt{fvar}\ x', y \mapsto \texttt{fvar}\ y') \ominus x) \ominus y, z \mapsto (t[x'/x])[y'[x'/x]/y])$
$= ((\Delta, x \mapsto \texttt{fvar}\ x', y \mapsto \texttt{fvar}\ y') \ominus [x,y], z \mapsto (t[x'/x])[y'[x'/x]/y])$
$= (\Delta' \ominus [x,y], z \mapsto t')$
where $\Delta' = (\Delta, x \mapsto \texttt{fvar}\ x', y \mapsto \texttt{fvar}\ y')$ and $t' = (t[x'/x])[y'[x'/x]/y]$.
Similarly, $\Gamma \ominus [y,x] = (\Delta' \ominus [y,x], z \mapsto t'')$ with $t'' = (t[y'/y])[x'[y'/y]/x]$.
$\texttt{ok}\ \Gamma \Rightarrow \texttt{ok}\ \Delta' \wedge z \notin \texttt{dom}(\Delta') \wedge \texttt{lc}\ t \overset{L29}{\Rightarrow} z \notin \texttt{dom}(\Delta' \ominus [x,y]) \wedge \texttt{lc}\ t'$.

$$\left.\begin{array}{l}\texttt{ok}\ \Delta' \\ x, y \in \texttt{Ind}(\Delta')\end{array}\right\} \overset{I.H.}{\Rightarrow} \Delta' \ominus [x,y] \approx^{V-\{x,y\}} \Delta' \ominus [y,x]$$

To prove: $t' \approx^{V-\{x,y\}} t''$.
- $x \neq y' \wedge y \neq x'$
$t' = (t[x'/x])[y'[x'/x]/y] = (t[x'/x])[y'/y] \overset{x \neq y}{=} (t[y'/y])[x'/x] = (t[y'/y])[x'[y'/y]/x] = t''$.
- $x \neq y' \wedge y = x'$
$t' = (t[x'/x])[y'[x'/x]/y] = (t[y/x])[y'/y] = (t[y'/x])[y'/y] \overset{x \neq y}{=} (t[y'/y])[y'/x] = (t[y'/y])[x'[y'/y]/x] = t''$.
- $x = y' \wedge y \neq x'$
$t' = (t[x'/x])[y'[x'/x]/y] = (t[x'/x])[x'/y] \overset{x \neq y}{=} (t[x'/y])[x'/x] = (t[x/y])[x'/x] = (t[x/y])[x'[x/y]/x] = (t[y'/y])[x'[y'/y]/x] = t''$.
- $x = y' \wedge y = x'$
$t' = (t[x'/x])[y'[x'/x]/y] = (t[y/x])[y/y] = t[y/x]$
$t'' = (t[y'/y])[x'[y'/y]/x] = (t[x/y])[x/x] = t[x/y]$
By CE_REF (Proposition 2),
$t \approx^{V-\{x,y\}} t \overset{L15}{\Rightarrow} t[y/x] \approx^{V-\{x,y\}} t \overset{P2}{\Rightarrow} t \approx^{V-\{x,y\}} t[y/x] \overset{L15}{\Rightarrow} t[x/y] \approx^{V-\{x,y\}} t[y/x]$.

$$\left.\begin{array}{l}\Delta' \ominus [x,y] \approx^{V-\{x,y\}} \Delta' \ominus [y,x] \\ t' \approx^{V-\{x,y\}} t'' \\ z \notin \texttt{dom}(\Delta' \ominus [x,y]) \wedge \texttt{lc}\ t'\end{array}\right\} \Rightarrow \Gamma \ominus [x,y] \approx^{V-\{x,y\}} \Gamma \ominus [y,x]$$

□

Lemma 19
HE_PERM $\quad \texttt{ok}\ \Gamma \wedge \overline{x}, \overline{y} \in \texttt{Ind}(\Gamma) \Rightarrow (\Gamma \ominus \overline{x} \approx \Gamma \ominus \overline{y} \Leftrightarrow \overline{y} \in \mathcal{S}(\overline{x}))$

Proof.
$\Rightarrow$ $\Gamma \ominus \overline{x} \approx \Gamma \ominus \overline{y} \overset{L16}{\Rightarrow} \texttt{dom}(\Gamma \ominus \overline{x}) = \texttt{dom}(\Gamma \ominus \overline{y}) \overset{L29}{\Rightarrow} \texttt{dom}(\Gamma) - \{\overline{x}\} = \texttt{dom}(\Gamma) - \{\overline{y}\}$.
Since $\overline{x}, \overline{y} \subseteq \texttt{dom}(\Gamma)$ and have pairwise-distinct names, we infer that $\overline{y} \in \mathcal{S}(\overline{x})$.

$\Leftarrow$ From Lemma 18 and Abstract Algebra results, i.e., any permutation can be written as a product of transpositions.

□

7.4 Results on indirection relation (Section 4.2)

Proposition 4
IR_ALT $\quad \Gamma \gtrsim_I \Gamma' \Leftrightarrow \texttt{ok}\ \Gamma \wedge \exists\ \overline{x} \subseteq \texttt{Ind}(\Gamma)\ .\ \Gamma \ominus \overline{x} \approx \Gamma'$

Proof.
$\Rightarrow$ By rule induction:

– IR_HE: $\Gamma \approx \Gamma'$.
  By HCE_OK (Lemma 16), $\mathtt{ok}\ \Gamma$.
  Let $\overline{x} = [\,]$, then $\Gamma \ominus [\,] = \Gamma \approx \Gamma'$.
– IR_IR: $\mathtt{ok}\ \Gamma \wedge \Gamma \ominus x \gtrsim_I \Gamma' \wedge x \in \mathtt{Ind}(\Gamma)$.
  By induction hypothesis,
  $\exists\ \overline{x} \subseteq \mathtt{Ind}(\Gamma \ominus x) \overset{L29}{=} \mathtt{Ind}(\Gamma) - \{x\}\ .\ (\Gamma \ominus x) \ominus \overline{x} \approx \Gamma' \Rightarrow \Gamma \ominus [x : \overline{x}] \approx \Gamma'$.

$\Leftarrow$ By induction on the length of $\overline{x}$:

– $\overline{x} = [\,]$
  $\Gamma = \Gamma \ominus [\,] \approx \Gamma' \Rightarrow \Gamma \gtrsim_I \Gamma'$.
– $\overline{x} = [y : \overline{y}]$
  $$\left.\begin{array}{l} \mathtt{ok}\ \Gamma \wedge y \in \mathtt{Ind}(\Gamma) \overset{L30}{\Rightarrow} \mathtt{ok}\ (\Gamma \ominus y) \\ y \notin \overline{y} \wedge \overline{y} \subseteq \mathtt{Ind}(\Gamma) \Rightarrow \overline{y} \subseteq \mathtt{Ind}(\Gamma) - \{y\} \overset{L29}{=} \mathtt{Ind}(\Gamma \ominus y) \\ \Gamma \ominus [y : \overline{y}] = (\Gamma \ominus y) \ominus \overline{y} \end{array}\right\} \overset{I.H.}{\Rightarrow} (\Gamma \ominus y) \gtrsim_I \Gamma' \Rightarrow \Gamma \gtrsim_I \Gamma'.$$

□

Corollary 1.
IR_DOM_DOM $\quad \Gamma \gtrsim_I \Gamma' \Rightarrow \Gamma \ominus (\mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')) \approx \Gamma'$

Proof.
$\Gamma \gtrsim_I \Gamma' \overset{P4}{\Rightarrow} \mathtt{ok}\ \Gamma \wedge \exists\ \overline{x} \subseteq \mathtt{Ind}(\Gamma)\ .\ \Gamma \ominus \overline{x} \approx \Gamma'$.
$\mathtt{dom}(\Gamma') \overset{L16}{=} \mathtt{dom}(\Gamma \ominus \overline{x}) \overset{L29}{=} \mathtt{dom}(\Gamma) - \overline{x} \Rightarrow \overline{x} = \mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')$. □

To prove Lemma 5 we extend some results on context-equivalence (for terms) to heap-context-equivalence.

Lemma 32.
HCE_SUB $\quad \Gamma \approx^V \Gamma' \wedge V' \subseteq V \Rightarrow \Gamma \approx^{V'} \Gamma'$
HCE_ADD $\quad \Gamma \approx^V \Gamma' \wedge \overline{x} \notin \mathtt{names}(\Gamma) \cup \mathtt{names}(\Gamma') \Rightarrow \Gamma \approx^{V \cup \overline{x}} \Gamma'$

Proof.

HCE_SUB
An easy induction using CE_SUB in Lemma 14.

HCE_ADD
An easy induction using CE_ADD in Lemma 14. □

Lemma 33.
HCE_DEL_IND $\quad \Gamma \approx^V \Gamma' \wedge \mathtt{dom}(\Gamma) \subseteq V \wedge x \in \mathtt{Ind}(\Gamma) \Rightarrow \Gamma \ominus x \approx^V \Gamma' \ominus x$
HE_DEL_IND $\quad \Gamma \approx \Gamma' \wedge x \in \mathtt{Ind}(\Gamma) \Rightarrow \Gamma \ominus x \approx \Gamma' \ominus x$

Proof.
HCE_DEL_IND
$x \in \mathtt{Ind}(\Gamma) \overset{L16}{=} \mathtt{Ind}(\Gamma') \Rightarrow \Gamma = (\Theta, x \mapsto \mathtt{fvar}\ y) \wedge \Gamma' = (\Theta', x \mapsto \mathtt{fvar}\ y')$
$\overset{L28}{\Rightarrow} \Theta \approx^V \Theta' \wedge (\mathtt{fvar}\ y) \approx^V (\mathtt{fvar}\ y')$.
Moreover, by HCE_OK (Lemma 16) $x \notin \mathtt{dom}(\Theta) \cup \mathtt{dom}(\Theta')$.
In addition, $\mathtt{dom}(\Gamma) \subseteq V \Rightarrow x \in V$.
We proceed by induction on the size of Θ:

– $\Theta = \emptyset \overset{L16}{\Rightarrow} \Theta' = \emptyset$.

– $\Theta = (\Delta, z \mapsto t)$ with $z \neq x$.
  By Lemma 16, $\Theta' = (\Delta', z \mapsto t')$.
  By Lemma 28, $\Delta \approx^V \Delta' \wedge t \approx^V t'$.
  Therefore, $(\Delta, x \mapsto \texttt{fvar}\ y) \approx^V (\Delta', x \mapsto \texttt{fvar}\ y') \overset{I.H.}{\Rightarrow} (\Delta, x \mapsto \texttt{fvar}\ y) \ominus x \approx^V (\Delta', x \mapsto \texttt{fvar}\ y') \ominus x$.
  Moreover, $t \approx^V t' \wedge (\texttt{fvar}\ y) \approx^V (\texttt{fvar}\ y') \wedge x \in V \overset{L15}{\Rightarrow} t[y/x] \approx^V t'[y'/x]$, $\texttt{lc}\ t \Rightarrow \texttt{lc}\ t[y/x]$, and $z \notin \texttt{dom}(\Delta, x \mapsto \texttt{fvar}\ y)$.
  Hence, $((\Delta, x \mapsto \texttt{fvar}\ y) \ominus x, z \mapsto t[y/x]) \approx^V ((\Delta', x \mapsto \texttt{fvar}\ y') \ominus x, z \mapsto t'[y'/x])$.
  But $\Gamma \ominus x = ((\Delta, x \mapsto \texttt{fvar}\ y) \ominus x, z \mapsto t[y/x])$ and $\Gamma' \ominus x = ((\Delta', x \mapsto \texttt{fvar}\ y') \ominus x, z \mapsto t'[y'/x])$.

HE_DEL_IND
It is a direct consequence of HCE_DEL_IND and Lemma 32. □

Proposition 5.

IR_REF	$\texttt{ok}\ \Gamma \Rightarrow \Gamma \succsim_I \Gamma$
IR_TRANS	$\Gamma \succsim_I \Gamma' \wedge \Gamma' \succsim_I \Gamma'' \Rightarrow \Gamma \succsim_I \Gamma''$

Proof.
IR_REF
By HCE_REF in Proposition 3.

IR_TRANS
By rule induction on $\Gamma \succsim_I \Gamma'$:

– $\Gamma \approx \Gamma'$.
  $\Gamma' \succsim_I \Gamma'' \overset{P4}{\Rightarrow} \texttt{ok}\ \Gamma' \wedge \exists\ \overline{x} \subseteq \texttt{Ind}(\Gamma')\ .\ \Gamma' \ominus \overline{x} \approx \Gamma''$.
  By HCE_IND (Lemma 16), $\overline{x} \subseteq \texttt{Ind}(\Gamma)$.
  By HE_DEL_IND (Lemma 33), $\Gamma \ominus \overline{x} \approx \Gamma' \ominus \overline{x}$ and, by transitivity of $\approx$ (Proposition 3), $\Gamma \ominus \overline{x} \approx \Gamma''$.
  By Proposition 4 (in the other direction) we get $\Gamma \succsim_I \Gamma''$.
– $\Gamma \ominus y \succsim_I \Gamma' \wedge y \in \texttt{Ind}(\Gamma) \wedge \texttt{ok}\ \Gamma$.
  By induction hypothesis $\Gamma \ominus y \succsim_I \Gamma'' \Rightarrow \Gamma \succsim_I \Gamma''$.

□

Lemma 20

IR_DOM	$\Gamma \succsim_I \Gamma' \Rightarrow \texttt{dom}(\Gamma') \subseteq \texttt{dom}(\Gamma)$
IR_IND	$\Gamma \succsim_I \Gamma' \Rightarrow \texttt{Ind}(\Gamma') \subseteq \texttt{Ind}(\Gamma)$
IR_OK	$\Gamma \succsim_I \Gamma' \Rightarrow \texttt{ok}\ \Gamma \wedge \texttt{ok}\ \Gamma'$
IR_DOM_HE	$\Gamma \succsim_I \Gamma' \wedge \texttt{dom}(\Gamma) = \texttt{dom}(\Gamma') \Rightarrow \Gamma \approx \Gamma'$
IR_IR_HE	$(\Gamma \succsim_I \Gamma' \wedge \Gamma' \succsim_I \Gamma) \Leftrightarrow \Gamma \approx \Gamma'$

Proof.

The proofs of IR_DOM, IR_IND and IR_OK are easy rule inductions.

IR_DOM_HE
We prove by contrapositive: Assume $\Gamma \succsim_I \Gamma' \wedge \Gamma \not\approx \Gamma'$.
$\Gamma \succsim_I \Gamma' \Rightarrow \texttt{ok}\ \Gamma \wedge \Gamma \ominus x \succsim_I \Gamma' \wedge x \in \texttt{Ind}(\Gamma)$.
By IR_DOM, $\texttt{dom}(\Gamma') \subseteq \texttt{dom}(\Gamma \ominus x) \overset{L29}{=} \texttt{dom}(\Gamma) - \{x\} \subsetneq \texttt{dom}(\Gamma)$.
Hence $\texttt{dom}(\Gamma) \neq \texttt{dom}(\Gamma')$.

IR_IR_HE

$\Rightarrow$ By IR_DOM,
$\left.\begin{array}{l} \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{dom}(\Gamma') \subseteq \mathtt{dom}(\Gamma) \\ \Gamma' \succsim_I \Gamma \Rightarrow \mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Gamma') \end{array}\right\} \Rightarrow \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma')$.
By IR_DOM_HE, $\Gamma \approx \Gamma'$.

$\Leftarrow$ $\Gamma \approx \Gamma' \Rightarrow \Gamma \succsim_I \Gamma'$.
$\Gamma \approx \Gamma' \overset{P3}{\Rightarrow} \Gamma' \approx \Gamma \Rightarrow \Gamma' \succsim_I \Gamma$.

□

Lemma 21

IREQ_HE_IREQ1	$[\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma \Rightarrow [\Delta] \succsim_I [\Gamma']$
IREQ_HE_IREQ2	$[\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma' \Rightarrow [\Gamma] \succsim_I [\Delta]$

Proof.
IREQ_HE_IREQ1
$[\Gamma] \succsim_I [\Gamma'] \Rightarrow \Gamma \succsim_I \Gamma'$.

- $\Gamma \approx \Gamma'$.
  $\Delta \approx \Gamma \overset{L16}{\Rightarrow} \mathtt{dom}(\Delta) = \mathtt{dom}(\Gamma) \overset{P3}{\Rightarrow} \Delta \approx \Gamma' \Rightarrow \Delta \succsim_I \Gamma' \Rightarrow [\Delta] \succsim_I [\Gamma']$.
- $\mathtt{ok}\ \Gamma \wedge (\Gamma \ominus x) \succsim_I \Gamma' \wedge x \in \mathtt{Ind}(\Gamma)$.
  $\Delta \approx \Gamma \overset{L16}{\Rightarrow} x \in \mathtt{Ind}(\Delta) \wedge \mathtt{ok}\ \Delta \overset{L33}{\Rightarrow} (\Delta \ominus x) \approx (\Gamma \ominus x) \Rightarrow (\Delta \ominus x) \succsim_I (\Gamma \ominus x) \overset{P5}{\Rightarrow} (\Delta \ominus x) \succsim_I \Gamma'$.
  Thus, $\Delta \succsim_I \Gamma' \Rightarrow [\Delta] \succsim_I [\Gamma']$.

IREQ_HE_IREQ2
$[\Gamma] \succsim_I [\Gamma'] \Rightarrow \Gamma \succsim_I \Gamma'$.

- $\Gamma \approx \Gamma' \overset{L16}{\Rightarrow} \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma')$.
  $\Delta \approx \Gamma' \overset{P3}{\Rightarrow} \Gamma' \approx \Delta \overset{P3}{\Rightarrow} \Gamma \approx \Delta \Rightarrow \Gamma \succsim_I \Delta \Rightarrow [\Gamma] \succsim_I [\Delta]$.
- $\mathtt{ok}\ \Gamma \wedge (\Gamma \ominus x) \succsim_I \Gamma' \wedge x \in \mathtt{Ind}(\Gamma)$.
  $\Delta \approx \Gamma' \overset{P3}{\Rightarrow} \Gamma' \approx \Delta \Rightarrow \Gamma' \succsim_I \Delta \overset{P5}{\Rightarrow} (\Gamma \ominus x) \succsim_I \Delta \Rightarrow \Gamma \succsim_I \Delta \Rightarrow [\Gamma] \succsim_I [\Delta]\cdot$

□

Proposition 6

IREQ_REF	$\mathtt{ok}\ \Gamma \Rightarrow [\Gamma] \succsim_I [\Gamma]$
IREQ_ANTSYM	$[\Gamma] \succsim_I [\Gamma'] \wedge [\Gamma'] \succsim_I [\Gamma] \Rightarrow [\Gamma] = [\Gamma']$
IREQ_TRANS	$[\Gamma] \succsim_I [\Gamma'] \wedge [\Gamma'] \succsim_I [\Gamma''] \Rightarrow [\Gamma] \succsim_I [\Gamma'']$

Proof.
Reflexivity and transitivity are immediate because $\succsim_I$ is a preorder for heaps (Proposition 5).
Antisymmetry is a consequence of IR_IR_HE (Lemma 20). □

Lemma 22

IRHT_IRH	$(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \Gamma \succsim_I \Gamma'$
IRHT_SS	$(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow t \sim_S t'$
IRHT_LC	$(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \mathtt{lc}\ t \wedge \mathtt{lc}\ t'$

Proof.
$(\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \forall z \notin L\,.\,(\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t')$, for some finite $L \subseteq Id$.

IRHT_IRH
$(\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t') \overset{C1}{\Rightarrow} (\Gamma, z \mapsto t) \ominus \overline{x} \approx (\Gamma', z \mapsto t')$ with $\overline{x} = \mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')$.
$(\Gamma, z \mapsto t) \ominus \overline{x} = (\Gamma \ominus \overline{x}, z \mapsto t'')$ being t'' the transformation of t by $\overline{x}$
$\overset{L28}{\Rightarrow} \Gamma \ominus \overline{x} \approx \Gamma' \overset{P4}{\Rightarrow} \Gamma \succsim_I \Gamma'$.

IRHT_SS
$(\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t') \overset{C1}{\Rightarrow} (\Gamma, z \mapsto t) \ominus \overline{x} \approx (\Gamma', z \mapsto t')$ with $\overline{x} = \mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')$.
We proceed by induction on the length of $\overline{x}$:

- $\overline{x} = [\,] \Rightarrow (\Gamma, z \mapsto t) \approx (\Gamma', z \mapsto t') \overset{L28}{\Rightarrow} t \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} t' \overset{L13}{\Rightarrow} t \sim_S t'$.
- $\overline{x} = [y : \overline{y}]$ for some $y \mapsto \mathtt{fvar}\ y' \in \Gamma$.
  $$\begin{aligned}(\Gamma, z \mapsto t) \ominus \overline{x} &= ((\Gamma, z \mapsto t) \ominus y) \ominus \overline{y} \\ &= (\Gamma \ominus y, z \mapsto t[y'/y]) \ominus \overline{y} \approx (\Gamma', z \mapsto t') \\ &\overset{I.H.}{\Rightarrow} t[y'/y] \sim_S t' \overset{L1\&P1}{\Rightarrow} t \sim_S t'.\end{aligned}$$

IRHT_LC
$(\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t') \overset{L20}{\Rightarrow} \mathtt{ok}\ (\Gamma, z \mapsto t) \wedge \mathtt{ok}\ (\Gamma', z \mapsto t') \Rightarrow \mathtt{lc}\ t \wedge \mathtt{lc}\ t'$. □

7.5 Equivalence (Section 4.3)

Several auxiliary results are needed to prove Proposition 7.

If $(\Gamma : \mathtt{fvar}\ x)$ and $(\Gamma' : \mathtt{fvar}\ x')$ are related, and there is a binding for x' in Γ', then there must be a binding for x in Γ too. Furthermore, there exists in Γ a sequence of indirections leading from x to x'.

Lemma 34.
IR_FVAR
$$\begin{aligned}&(\Gamma : \mathtt{fvar}\ x) \succsim_I (\Gamma' : \mathtt{fvar}\ x') \\ &\Rightarrow (x \notin \mathtt{dom}(\Gamma) \wedge x' \notin \mathtt{dom}(\Gamma')) \\ &\vee (x \in \mathtt{dom}(\Gamma) \wedge x' \in \mathtt{dom}(\Gamma') \wedge \\ &\quad \exists x_0, \ldots, x_n \in Id\,.\, x_0 = x \wedge x_n = x' \wedge \forall i : 0 \le i < n\,.\, x_i \mapsto \mathtt{fvar}\ x_{i+1} \in \Gamma)\end{aligned}$$

Proof.
$(\Gamma : \mathtt{fvar}\ x) \succsim_I (\Gamma' : \mathtt{fvar}\ x') \Rightarrow \forall z \notin L\,.\,(\Gamma, z \mapsto \mathtt{fvar}\ x) \succsim_I (\Gamma', z \mapsto \mathtt{fvar}\ x')$
$\overset{C1}{\Rightarrow} (\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus \overline{x} \approx (\Gamma', z \mapsto \mathtt{fvar}\ x')$ with $\overline{x} = \mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')$.
Take $z \notin L$ such that $z \neq x \wedge z \neq x'$. Now we proceed by induction on the length of $\overline{x}$:

- $\overline{x} = [\,] \Rightarrow \mathtt{dom}(\Gamma') = \mathtt{dom}(\Gamma) \wedge (\Gamma, z \mapsto \mathtt{fvar}\ x) \approx (\Gamma', z \mapsto \mathtt{fvar}\ x') \overset{L28}{\Rightarrow} \mathtt{fvar}\ x \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} \mathtt{fvar}\ x'$.
  - $x \notin \mathtt{dom}(\Gamma) \Rightarrow x' \notin \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma')$.
  - $x \in \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \Rightarrow x = x' \Rightarrow x' \in \mathtt{dom}(\Gamma')$,
    and the second part of the result is immediate (take $n = 0$).
- $\overline{x} \neq [\,]$
  - $x \notin \overline{x} \Rightarrow (\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus \overline{x} = (\Gamma \ominus \overline{x}, z \mapsto \mathtt{fvar}\ x) \approx (\Gamma', z \mapsto \mathtt{fvar}\ x')$.
    With a reasoning similar to the empty list case, we infer that either $x \notin \mathtt{dom}(\Gamma) \wedge x' \notin \mathtt{dom}(\Gamma')$, or $x \in \mathtt{dom}(\Gamma) \cap \mathtt{dom}(\Gamma')$ and $x = x'$.
  - $x \in \overline{x} \Rightarrow x \in \mathtt{Ind}(\Gamma)$ and $x \mapsto \mathtt{fvar}\ y \in \Gamma$ for some $y \in Id$.
    Moreover, we have $[x : \overline{y}] \in \mathcal{S}(\overline{x}) \overset{L19}{\Rightarrow} (\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus [x : \overline{y}] \approx (\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus \overline{x}$.
    Hence $(\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus [x : \overline{y}] = (\Gamma \ominus x, z \mapsto \mathtt{fvar}\ y) \ominus \overline{y}$
    $\overset{I.H.}{\Rightarrow} x' \in \mathtt{dom}(\Gamma') \wedge \exists x_0, \ldots, x_n \in Id\,.\, x_0 = y \wedge x_n = x' \wedge \forall i : 0 \le i < n\,.\, x_i \mapsto \mathtt{fvar}\ x_{i+1} \in \Gamma \ominus x$.
    Since $\forall i : 0 < i < n\,.\, x_i \neq y$ we have $\forall i : 0 \le i < n\,.\, x_i \mapsto \mathtt{fvar}\ x_{i+1} \in \Gamma$ too, and we can extend the sequence with $x \mapsto \mathtt{fvar}\ y$.

□

Lemma 35.
IR_IRHT $(\Gamma, x \mapsto t) \succsim_I (\Gamma', x \mapsto t') \Rightarrow ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x \mapsto t') : t')$

Proof. By rule induction:

– IR_HE
$(\Gamma, x \mapsto t) \approx (\Gamma', x \mapsto t') \Rightarrow (\Gamma, x \mapsto t) \approx^{\mathtt{dom}(\Gamma, x \mapsto t)} (\Gamma', x \mapsto t') \overset{L28}{\Rightarrow} t \approx^{\mathtt{dom}(\Gamma, x \mapsto t)} t' \wedge \mathtt{lc}\ t$.
Let $L = \mathtt{names}(\Gamma, x \mapsto t) \cup \mathtt{names}(\Gamma', x \mapsto t')$, then by CE_ADD (in Lemma 14) and HCE_ADD (in Lemma 32), $\forall z \notin L \,.\, (\Gamma, x \mapsto t) \approx^{\mathtt{dom}(\Gamma, x \mapsto t) \cup \{z\}} (\Gamma', x \mapsto t') \wedge t \approx^{\mathtt{dom}(\Gamma, x \mapsto t) \cup \{z\}} t'$
$\Rightarrow \forall z \notin L \,.\, (\Gamma, x \mapsto t, z \mapsto t) \approx^{\mathtt{dom}(\Gamma, x \mapsto t) \cup \{z\}} (\Gamma', x \mapsto t', z \mapsto t')$
$\Rightarrow \forall z \notin L \,.\, (\Gamma, x \mapsto t, z \mapsto t) \succsim_I (\Gamma', x \mapsto t', z \mapsto t')$
$\Rightarrow ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x \mapsto t') : t')$.
– IR_IR
$(\Gamma, x \mapsto t) \ominus y \succsim_I (\Gamma', x \mapsto t')$ for some $y \mapsto \mathtt{fvar}\ y' \in \Gamma$
$\Rightarrow (\Gamma \ominus y, x \mapsto t[y'/y]) \succsim_I (\Gamma', x \mapsto t')$
$\overset{I.H.}{\Rightarrow} ((\Gamma \ominus y, x \mapsto t[y'/y]) : t[y'/y]) \succsim_I ((\Gamma', x \mapsto t') : t')$
$\Rightarrow \forall z \notin L \,.\, (\Gamma \ominus y, x \mapsto t[y'/y], z \mapsto t[y'/y]) \succsim_I (\Gamma', x \mapsto t', z \mapsto t')$, for some $L \subseteq Id$
$\Rightarrow \forall z \notin L \cup \mathtt{dom}(\Gamma) \,.\, (\Gamma, x \mapsto t, z \mapsto t) \ominus y \succsim_I (\Gamma', x \mapsto t', z \mapsto t')$
$\Rightarrow ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x \mapsto t') : t')$.

□

Lemma 36.
IR_FVAR_IRHT $((\Gamma, x \mapsto t) : \mathtt{fvar}\ x) \succsim_I ((\Gamma', x' \mapsto t') : \mathtt{fvar}\ x')$
$\Rightarrow ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x' \mapsto t') : t') \vee ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x' \mapsto t') : \mathtt{fvar}\ x')$.

Proof.
$((\Gamma, x \mapsto t) : \mathtt{fvar}\ x) \succsim_I ((\Gamma', x' \mapsto t') : \mathtt{fvar}\ x')$
$\overset{L34}{\Rightarrow} \exists x_0, \ldots, x_n \in Id \,.\, x_0 = x \wedge x_n = x' \wedge \forall i : 0 \leq i < n \,.\, x_i \mapsto \mathtt{fvar}\ x_{i+1} \in (\Gamma, x \mapsto t)$.

– $n = 0 \Rightarrow x = x'$
$\Rightarrow ((\Gamma, x \mapsto t) : \mathtt{fvar}\ x) \succsim_I ((\Gamma', x \mapsto t') : \mathtt{fvar}\ x)$
$\overset{L22}{\Rightarrow} (\Gamma, x \mapsto t) \succsim_I (\Gamma', x \mapsto t')$
$\overset{L35}{\Rightarrow} ((\Gamma, x \mapsto t) : t) \succsim_I ((\Gamma', x \mapsto t') : t')$.
– $n > 0 \Rightarrow t \equiv \mathtt{fvar}\ x_1 \Rightarrow ((\Gamma, x \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x) \succsim_I ((\Gamma', x' \mapsto t') : \mathtt{fvar}\ x')$
$\Rightarrow \forall z \notin L \,.\, (\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x) \succsim_I (\Gamma', x' \mapsto t', z \mapsto \mathtt{fvar}\ x')$ for some $L \subseteq Id$
$\overset{C1}{\Rightarrow} (\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x) \ominus \overline{y} \approx (\Gamma', x' \mapsto t', z \mapsto \mathtt{fvar}\ x')$
with $\overline{y} = \mathtt{dom}(\Gamma) \cup \{x\} - (\mathtt{dom}(\Gamma') \cup \{x'\})$.
  - $x \notin \overline{y} \Rightarrow x \in \mathtt{dom}((\Gamma, x \mapsto \mathtt{fvar}\ x_1) \ominus \overline{y})$.
  $(\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x) \ominus \overline{y} = ((\Gamma, x \mapsto \mathtt{fvar}\ x_1) \ominus \overline{y}, z \mapsto \mathtt{fvar}\ x)$
  $\overset{L28}{\Rightarrow} \mathtt{fvar}\ x \approx^{\mathtt{dom}((\Gamma, x \mapsto \mathtt{fvar}\ x_1) \ominus \overline{y}) \cup \{z\}} \mathtt{fvar}\ x'$.
  Hence, $x = x'$ and we proceed like in the case $n = 0$.
  - $x \in \overline{y} \Rightarrow [x : \overline{x}] \in \mathcal{S}(\overline{y})$ for some $\overline{x}$.
  $\overset{L19}{\Rightarrow} (\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x) \ominus [x : \overline{x}] \approx (\Gamma', x' \mapsto t', z \mapsto \mathtt{fvar}\ x')$.
  But $(\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x) \ominus [x : \overline{x}] = (\Gamma, z \mapsto \mathtt{fvar}\ x_1) \ominus \overline{x} =$
  $(\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x_1) \ominus [x : \overline{x}]$
  $\overset{P4}{\Rightarrow} (\Gamma, x \mapsto \mathtt{fvar}\ x_1, z \mapsto \mathtt{fvar}\ x_1) \succsim_I (\Gamma', x' \mapsto t', z \mapsto \mathtt{fvar}\ x')$.
  This can be obtained for any $z \notin L$, so that
  $((\Gamma, x \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x_1) \succsim_I ((\Gamma', x' \mapsto t') : \mathtt{fvar}\ x')$.

□

If two heap/application pairs are indirection related then the bodies of the applications and their arguments are also related.

Lemma 37.
IRHT_APP $(\Gamma : \mathtt{app}\ t\ (\mathtt{fvar}\ x)) \succsim_I (\Gamma' : \mathtt{app}\ t'\ (\mathtt{fvar}\ x'))$
$\Rightarrow (\Gamma : t) \succsim_I (\Gamma' : t') \wedge (\Gamma : \mathtt{fvar}\ x) \succsim_I (\Gamma' : \mathtt{fvar}\ x')$.

Proof.
$(\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x)) \gtrsim_I (\Gamma' : \texttt{app}\ t'\ (\texttt{fvar}\ x'))$
$\Rightarrow \forall z \notin L\,.\,(\Gamma, z \mapsto \texttt{app}\ t\ (\texttt{fvar}\ x)) \gtrsim_I (\Gamma', z \mapsto \texttt{app}\ t'\ (\texttt{fvar}\ x'))$ for some finite $L \subseteq Id$
$\overset{C1}{\Rightarrow} (\Gamma, z \mapsto \texttt{app}\ t\ (\texttt{fvar}\ x)) \ominus \overline{x} \approx (\Gamma', z \mapsto \texttt{app}\ t'\ (\texttt{fvar}\ x'))$ with $\overline{x} = \texttt{dom}(\Gamma) - \texttt{dom}(\Gamma')$.
Now we prove by induction on the length of $\overline{x}$ that

$$(\Gamma, z \mapsto \texttt{app}\ t\ (\texttt{fvar}\ x)) \ominus \overline{x} \approx (\Gamma', z \mapsto \texttt{app}\ t'\ (\texttt{fvar}\ x'))$$
$$\Rightarrow (\Gamma, z \mapsto t) \ominus \overline{x} \approx (\Gamma', z \mapsto t') \wedge (\Gamma, z \mapsto \texttt{fvar}\ x) \ominus \overline{x} \approx (\Gamma', z \mapsto \texttt{fvar}\ x')$$

- $\overline{x} = [\,]$
  $\Rightarrow (\Gamma, z \mapsto \texttt{app}\ t\ (\texttt{fvar}\ x)) \approx (\Gamma', z \mapsto \texttt{app}\ t'\ (\texttt{fvar}\ x'))$
  $\overset{L28}{\Rightarrow} \Gamma \approx^{\texttt{dom}(\Gamma)\cup\{z\}} \Gamma' \wedge (\texttt{app}\ t\ (\texttt{fvar}\ x)) \approx^{\texttt{dom}(\Gamma)\cup\{z\}} (\texttt{app}\ t'\ (\texttt{fvar}\ x')) \wedge \texttt{lc}\ (\texttt{app}\ t\ (\texttt{fvar}\ x)) \wedge z \notin \texttt{dom}(\Gamma)$
  $(\texttt{app}\ t\ (\texttt{fvar}\ x)) \approx^{\texttt{dom}(\Gamma)\cup\{z\}} (\texttt{app}\ t'\ (\texttt{fvar}\ x')) \Rightarrow t \approx^{\texttt{dom}(\Gamma)\cup\{z\}} t' \wedge (\texttt{fvar}\ x) \approx^{\texttt{dom}(\Gamma)\cup\{z\}} (\texttt{fvar}\ x')$
  $\texttt{lc}\ (\texttt{app}\ t\ (\texttt{fvar}\ x)) \Rightarrow \texttt{lc}\ t \wedge \texttt{lc}\ (\texttt{fvar}\ x)$
  $\Rightarrow (\Gamma, z \mapsto t) \approx^{\texttt{dom}(\Gamma)\cup\{z\}} (\Gamma', z \mapsto t') \wedge (\Gamma, z \mapsto \texttt{fvar}\ x) \approx^{\texttt{dom}(\Gamma)\cup\{z\}} (\Gamma', z \mapsto \texttt{fvar}\ x')$.
- $\overline{x} = [y : \overline{y}]$ with $y \mapsto \texttt{fvar}\ y' \in \Gamma \wedge y \neq z$.
  $(\Gamma, z \mapsto \texttt{app}\ t\ (\texttt{fvar}\ x)) \ominus \overline{x} = (\Gamma \ominus y, z \mapsto \texttt{app}\ t[y'/y]\ (\texttt{fvar}\ x[y'/y])) \ominus \overline{y} \approx (\Gamma', z \mapsto \texttt{app}\ t'\ (\texttt{fvar}\ x'))$
  $\overset{I.H.}{\Rightarrow} (\Gamma \ominus y, z \mapsto t[y'/y]) \ominus \overline{y} \approx (\Gamma', z \mapsto t') \wedge (\Gamma \ominus y, z \mapsto \texttt{fvar}\ x[y'/y]) \ominus \overline{y} \approx (\Gamma', z \mapsto \texttt{fvar}\ x')$
  $\Rightarrow (\Gamma, z \mapsto t) \ominus [y : \overline{y}] \approx (\Gamma', z \mapsto t') \wedge (\Gamma, z \mapsto \texttt{fvar}\ x) \ominus [y : \overline{y}] \approx (\Gamma', z \mapsto \texttt{fvar}\ x')$.

By Proposition 4 we have $(\Gamma, z \mapsto t) \gtrsim_I (\Gamma', z \mapsto t') \wedge (\Gamma, z \mapsto \texttt{fvar}\ x) \gtrsim_I (\Gamma', z \mapsto \texttt{fvar}\ x')$,
and the result is valid for any $z \notin L$, so that $(\Gamma : t) \gtrsim_I (\Gamma' : t')$ and $(\Gamma : \texttt{fvar}\ x) \gtrsim_I (\Gamma' : \texttt{fvar}\ x')$. □

The next lemma is useful to show that the indirection relation between variables can be preserved through evaluation.

Lemma 38.
IRHT_FV
$$(\Gamma : \texttt{fvar}\ x) \gtrsim_I (\Gamma' : \texttt{fvar}\ x') \wedge \Gamma \subseteq \Delta \wedge \Gamma' \subseteq \Delta' \wedge \Delta \gtrsim_I \Delta' \wedge$$
$$(x \notin \texttt{dom}(\Gamma) \Rightarrow x \notin \texttt{dom}(\Delta)) \wedge (x' \notin \texttt{dom}(\Gamma') \Rightarrow x' \notin \texttt{dom}(\Delta')) \wedge$$
$$(y \in \texttt{dom}(\Gamma) \wedge y \notin \texttt{dom}(\Gamma') \Rightarrow y \notin \texttt{dom}(\Delta'))$$
$$\Rightarrow (\Delta : \texttt{fvar}\ x) \gtrsim_I (\Delta' : \texttt{fvar}\ x').$$

Proof.
$(\Gamma : \texttt{fvar}\ x) \gtrsim_I (\Gamma' : \texttt{fvar}\ x') \Rightarrow \forall z \notin L\,.\,(\Gamma, z \mapsto \texttt{fvar}\ x) \gtrsim_I (\Gamma', z \mapsto \texttt{fvar}\ x')$ for some finite $L \subseteq Id$
$\overset{C1}{\Rightarrow} \forall z \notin L\,.\,(\Gamma, z \mapsto \texttt{fvar}\ x) \ominus \overline{x} \approx (\Gamma', z \mapsto \texttt{fvar}\ x')$ with $\overline{x} = \texttt{dom}(\Gamma) - \texttt{dom}(\Gamma')$.
Furthermore, $\Delta \gtrsim_I \Delta' \overset{C1}{\Rightarrow} \Delta \ominus \overline{y} \approx \Delta'$ with $\overline{y} = \texttt{dom}(\Delta) - \texttt{dom}(\Delta')$.
Notice that, $\Gamma \subseteq \Delta \Rightarrow \overline{x} \subseteq \texttt{dom}(\Delta)$;
and by hypothesis, $y \in \texttt{dom}(\Gamma) - \texttt{dom}(\Gamma') \Rightarrow y \notin \texttt{dom}(\Delta')$, so that we can write $\overline{y} = \overline{x} +\!\!+ \overline{z}$.
Let $L' = L \cup \{x, x'\} \cup \texttt{dom}(\Delta) \cup \overline{y}$, we prove:

$$\forall z \notin L'\,.\,(\Delta, z \mapsto \texttt{fvar}\ x) \ominus \overline{y} \approx (\Delta', z \mapsto \texttt{fvar}\ x').$$

By induction on the length of $\overline{x}$:

- $\overline{x} = [\,] \Rightarrow \overline{y} = \overline{z}$.
  On the one hand, $z \notin L' \Rightarrow z \notin L$ so that $(\Gamma, z \mapsto \texttt{fvar}\ x) \approx (\Gamma', z \mapsto \texttt{fvar}\ x')$
  $\overset{L28}{\Rightarrow} \texttt{fvar}\ x \approx^{\texttt{dom}(\Gamma)\cup\{z\}} \texttt{fvar}\ x'$
  $\Rightarrow x = x' \vee x, x' \notin \texttt{dom}(\Gamma) \cup \{z\} = \texttt{dom}(\Gamma') \cup \{z\}$.
  On the other hand, $\Delta \ominus \overline{z} \approx \Delta'$.
  We show that $\texttt{fvar}\ x \approx^{\texttt{dom}(\Delta\ominus\overline{z})\cup\{z\}} \texttt{fvar}\ x'$:
  - If $x = x'$ then the result is trivial.

- If $x, x' \notin \mathtt{dom}(\Gamma) \cup \{z\} = \mathtt{dom}(\Gamma') \cup \{z\}$ then, by hypothesis,
  $x \notin \mathtt{dom}(\Delta) \cup \{z\} \Rightarrow x \notin \mathtt{dom}(\Delta \ominus \overline{z}) \cup \{z\}$
  $x' \notin \mathtt{dom}(\Delta') \cup \{z\} = \mathtt{dom}(\Delta \ominus \overline{z}) \cup \{z\}$

  We know that $\mathtt{lc}\ (\mathtt{fvar}\ x)$ and $z \notin L' \Rightarrow z \notin \mathtt{dom}(\Delta \ominus \overline{z})$.
  Therefore, $\forall z \notin L'.(\Delta, z \mapsto \mathtt{fvar}\ x) \ominus \overline{z} = (\Delta \ominus \overline{z}, z \mapsto \mathtt{fvar}\ x) \approx (\Delta', z \mapsto \mathtt{fvar}\ x')$.

– $\overline{x} = [y : \overline{x'}]$ with $y \mapsto \mathtt{fvar}\ y' \in \Gamma$.
  We know that:
  1. $(\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus [y : \overline{x'}] = (\Gamma \ominus y, z \mapsto \mathtt{fvar}\ x[y'/y]) \ominus \overline{x'} \approx (\Gamma', z \mapsto \mathtt{fvar}\ x')$, for all $z \notin L' \subseteq L$.
  2. $\Gamma \subseteq \Delta \Rightarrow \Gamma \ominus y \subseteq \Delta \ominus y$.
  3. $\Delta \ominus ([y : \overline{x'}] +\!\!+ \overline{z}) = (\Delta \ominus y) \ominus (\overline{x'} +\!\!+ \overline{z}) \approx \Delta' \overset{P4}{\Rightarrow} \Delta \ominus y \gtrsim_I \Delta'$.
  4. By Lemma 29, $\mathtt{dom}(\Gamma \ominus y) = \mathtt{dom}(\Gamma) - \{y\} \wedge \mathtt{dom}(\Delta \ominus y) = \mathtt{dom}(\Delta) - \{y\}$.
     Therefore, $x \notin \mathtt{dom}(\Gamma \ominus y) \Rightarrow x \notin \mathtt{dom}(\Gamma) \vee x = y \Rightarrow x \notin \mathtt{dom}(\Delta) \vee x = y \Rightarrow x \notin \mathtt{dom}(\Delta \ominus y)$.
  5. $y'' \in \mathtt{dom}(\Gamma \ominus y) \wedge y'' \notin \mathtt{dom}(\Gamma') \Rightarrow y'' \in \mathtt{dom}(\Gamma) \wedge y'' \neq y \wedge y'' \notin \mathtt{dom}(\Gamma') \Rightarrow y'' \notin \mathtt{dom}(\Delta')$.

  Then by induction hypothesis:
  $\forall z \notin L'.(\Delta \ominus y, z \mapsto (\mathtt{fvar}\ x)[y'/y]) \ominus (\overline{x'} +\!\!+ \overline{z}) \approx (\Delta', z \mapsto \mathtt{fvar}\ x')$
  $\Rightarrow \forall z \notin L'.(\Delta, z \mapsto \mathtt{fvar}\ x) \ominus \overline{y} \approx (\Delta', z \mapsto \mathtt{fvar}\ x')$.

Now, by Proposition 4, $\forall z \notin L'.(\Delta, z \mapsto \mathtt{fvar}\ x) \gtrsim_I (\Delta', z \mapsto \mathtt{fvar}\ x') \Rightarrow (\Delta : \mathtt{fvar}\ x) \gtrsim_I (\Delta' : \mathtt{fvar}\ x')$. □

The introduction of indirections at functional application preserves the indirection-relation.

Lemma 39.
IRHT_RED_IND $\quad \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : t) \wedge y \neq x \wedge (\Gamma : \mathtt{abs}\ t) \gtrsim_I (\Gamma' : \mathtt{abs}\ t') \wedge (\Gamma : \mathtt{fvar}\ x) \gtrsim_I (\Gamma' : \mathtt{fvar}\ x')$
$\Rightarrow ((\Gamma, y \mapsto \mathtt{fvar}\ x) : t^y) \gtrsim_I (\Gamma' : t'^{x'})$.

Proof.
$(\Gamma : \mathtt{abs}\ t) \gtrsim_I (\Gamma' : \mathtt{abs}\ t') \Rightarrow \forall z \notin L'.(\Gamma, z \mapsto \mathtt{abs}\ t) \gtrsim_I (\Gamma', z \mapsto \mathtt{abs}\ t')$, for some finite $L' \subseteq Id$
$\overset{C1}{\Rightarrow} \forall z \notin L'.(\Gamma, z \mapsto \mathtt{abs}\ t) \ominus \overline{x} \approx (\Gamma', z \mapsto \mathtt{abs}\ t')$ with $\overline{x} = \mathtt{dom}(\Gamma) - \mathtt{dom}(\Gamma')$.
Similarly, $(\Gamma : \mathtt{fvar}\ x) \gtrsim_I (\Gamma' : \mathtt{fvar}\ x') \Rightarrow \forall z \notin L''.(\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus \overline{x} \approx (\Gamma', z \mapsto \mathtt{fvar}\ x')$ for some finite $L'' \subseteq Id$.
Let $L = L' \cup L'' \cup \{y\} \cup \mathtt{names}(\Gamma) \cup \mathtt{names}(\Gamma') \cup \mathtt{fv}(t) \cup \mathtt{fv}(t') \cup \{x, x'\}$,
we will prove that $\forall z \notin L.(\Gamma, y \mapsto \mathtt{fvar}\ x, z \mapsto t^y) \ominus [y : \overline{x}] \approx (\Gamma', z \mapsto t'^{x'})$.
By induction on the length of $\overline{x}$:

– $\overline{x} = [\,]$.
  Let $z \notin L$, $(\Gamma, y \mapsto \mathtt{fvar}\ x, z \mapsto t^y) \ominus y = ((\Gamma, y \mapsto \mathtt{fvar}\ x) \ominus y, z \mapsto t^x) \overset{L31}{=} (\Gamma, z \mapsto t^x)$.
  $z \notin L \Rightarrow z \notin L' \Rightarrow (\Gamma, z \mapsto \mathtt{abs}\ t) \approx (\Gamma', z \mapsto \mathtt{abs}\ t')$
  $\overset{L28}{\Rightarrow} (\mathtt{abs}\ t) \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} (\mathtt{abs}\ t') \Rightarrow t \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} t'$.
  $z \notin L \Rightarrow z \notin L'' \Rightarrow (\Gamma, z \mapsto \mathtt{fvar}\ x) \approx (\Gamma', z \mapsto \mathtt{fvar}\ x') \Rightarrow (\mathtt{fvar}\ x) \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} (\mathtt{fvar}\ x')$.
  Therefore, for any $z \notin L$ we have $t \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} t' \wedge (\mathtt{fvar}\ x) \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} (\mathtt{fvar}\ x')$,
  and by CE_OP3 (Lemma 15), $t^x \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} t'^{x'}$.
  Furthermore, $z \notin L \Rightarrow z \notin \mathtt{dom}(\Gamma) \cup \mathtt{dom}(\Gamma')$, so that $\Gamma \approx \Gamma' \overset{L32}{\Rightarrow} \Gamma \approx^{\mathtt{dom}(\Gamma) \cup \{z\}} \Gamma'$.
  Finally, $\mathtt{lc}\ (\mathtt{abs}\ t) \Rightarrow \mathtt{lc}\ t^x$.
  We can then conclude that $\forall z \notin L.(\Gamma, z \mapsto t^x) \approx (\Gamma', z \mapsto t'^{x'})$.
– $\overline{x} = [y' : \overline{x'}]$ with $y' \mapsto \mathtt{fvar}\ y'' \in \Gamma \wedge y' \neq y$.
  Let $z \notin L$,
  $z \notin L' \Rightarrow (\Gamma, z \mapsto \mathtt{abs}\ t) \ominus \overline{x} = (\Gamma \ominus y', z \mapsto \mathtt{abs}\ t[y''/y']) \ominus \overline{x'} \approx (\Gamma', z \mapsto \mathtt{abs}\ t')$;
  $z \notin L'' \Rightarrow (\Gamma, z \mapsto \mathtt{fvar}\ x) \ominus \overline{x} = (\Gamma \ominus y', z \mapsto (\mathtt{fvar}\ x)[y''/y']) \ominus \overline{x'} \approx (\Gamma', z \mapsto \mathtt{fvar}\ x')$.

By induction hypothesis:
$(\Gamma \ominus y', y \mapsto (\texttt{fvar}\ x)[y''/y'], z \mapsto t[y''/y']^y) \ominus [y : \overline{x'}] \approx (\Gamma', z \mapsto t'^{x'})$
$\overset{y \neq y'}{\Rightarrow} (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y) \ominus [y' : y : \overline{x'}] \approx (\Gamma', z \mapsto t'^{x'})$
$\overset{L19}{\Rightarrow} (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y) \ominus [y : y' : \overline{x'}] \approx (\Gamma', z \mapsto t'^{x'})$
$\Rightarrow (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y) \ominus [y : \overline{x}] \approx (\Gamma', z \mapsto t'^{x'})$.

Now we check that $\texttt{ok}\ (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y)$:

$$\left.\begin{array}{l} \Gamma \gtrsim_I \Gamma'' \overset{L20}{\Rightarrow} \texttt{ok}\ \Gamma \wedge \texttt{ok}\ \Gamma' \\ y \notin \texttt{dom}(\Gamma) \\ \texttt{lc}\ (\texttt{fvar}\ x) \end{array}\right\} \Rightarrow \texttt{ok}\ (\Gamma, y \mapsto \texttt{fvar}\ x).$$

$$\left.\begin{array}{l} \texttt{ok}\ (\Gamma, y \mapsto \texttt{fvar}\ x) \\ z \notin L \supseteq \texttt{dom}(\Gamma) \cup \{y\} \\ \texttt{lc}\ (\texttt{abs}\ t) \Rightarrow \texttt{lc}\ t^y \end{array}\right\} \Rightarrow \texttt{ok}\ (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y).$$

By Proposition 4,
$\forall z \notin L. (\Gamma, y \mapsto \texttt{fvar}\ x, z \mapsto t^y) \ominus [y : \overline{x}] \approx (\Gamma', z \mapsto t'^{x'}) \Rightarrow ((\Gamma, y \mapsto \texttt{fvar}\ x) : t^y) \gtrsim_I (\Gamma' : t'^{x'})$. □

If two let expressions are indirection related, then their body terms are also related with respect to the heaps extended with the local declarations.

Lemma 40.
INTR_VARS $\quad \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \wedge$
$(\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \gtrsim_I (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \Rightarrow ((\Gamma, \overline{x} \mapsto \overline{t^x}) : t^{\overline{x}}) \gtrsim_I ((\Gamma', \overline{x} \mapsto \overline{t'^x}) : t'^{\overline{x}})$.

Proof.
$(\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \gtrsim_I (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t')$
$\Rightarrow \forall z \notin L\,.\,(\Gamma, z \mapsto \texttt{let}\ \overline{t}\ \texttt{in}\ t) \gtrsim_I (\Gamma', z \mapsto \texttt{let}\ \overline{t'}\ \texttt{in}\ t')$ for some finite $L \subseteq Id$
$\overset{C1}{\Rightarrow} (\Gamma, z \mapsto \texttt{let}\ \overline{t}\ \texttt{in}\ t) \ominus \overline{y} \approx (\Gamma', z \mapsto \texttt{let}\ \overline{t'}\ \texttt{in}\ t')$ with $\overline{y} = \texttt{dom}(\Gamma) - \texttt{dom}(\Gamma')$.
Consider any $z \notin L$ such that $\texttt{fresh}\ z\ \texttt{in}\ (\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \wedge \texttt{fresh}\ z\ \texttt{in}\ (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \wedge z \notin \overline{x}$.
Now we prove by induction on the length of $\overline{y}$ that

$$(\Gamma, z \mapsto \texttt{let}\ \overline{t}\ \texttt{in}\ t) \ominus \overline{y} \approx (\Gamma', z \mapsto \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \Rightarrow (\Gamma, \overline{x} \mapsto \overline{t^x}, z \mapsto t^{\overline{x}}) \ominus \overline{y} \approx (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$$

– $\overline{y} = [\,]$
$\Rightarrow (\Gamma, z \mapsto \texttt{let}\ \overline{t}\ \texttt{in}\ t) \approx (\Gamma', z \mapsto \texttt{let}\ \overline{t'}\ \texttt{in}\ t')$
$\overset{L28}{\Rightarrow} \Gamma \approx^{\texttt{dom}(\Gamma) \cup \{z\}} \Gamma' \wedge \texttt{let}\ \overline{t}\ \texttt{in}\ t \approx^{\texttt{dom}(\Gamma) \cup \{z\}} \texttt{let}\ \overline{t'}\ \texttt{in}\ t' \wedge \texttt{lc}\ (\texttt{let}\ \overline{t'}\ \texttt{in}\ t') \wedge z \notin \texttt{dom}(\Gamma)$
$\texttt{let}\ \overline{t}\ \texttt{in}\ t \approx^{\texttt{dom}(\Gamma) \cup \{z\}} \texttt{let}\ \overline{t'}\ \texttt{in}\ t' \Rightarrow |\overline{t}| = |\overline{t'}| \wedge \overline{t} \approx^{\texttt{dom}(\Gamma) \cup \{z\}} \overline{t'} \wedge t \approx^{\texttt{dom}(\Gamma) \cup \{z\}} t'$
$\overset{L14}{\Rightarrow} \overline{t} \approx^{\texttt{dom}(\Gamma) \cup \overline{x} \cup \{z\}} \overline{t'} \wedge t \approx^{\texttt{dom}(\Gamma) \cup \overline{x} \cup \{z\}} t'$
$\overset{L15}{\Rightarrow} \overline{t^x} \approx^{\texttt{dom}(\Gamma) \cup \overline{x} \cup \{z\}} \overline{t'^x} \wedge t^{\overline{x}} \approx^{\texttt{dom}(\Gamma) \cup \overline{x} \cup \{z\}} t'^{\overline{x}}$.
$\texttt{lc}\ \texttt{let}\ \overline{t'}\ \texttt{in}\ t' \Rightarrow \texttt{lc}\ t^{\overline{x}} \wedge \texttt{lc}\ \overline{t^x} \Rightarrow \texttt{ok}\ (\Gamma, \overline{x} \mapsto \overline{t^x}, z \mapsto t^{\overline{x}}) \wedge \texttt{ok}\ (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$
$\overset{L17}{\Rightarrow} (\Gamma, \overline{x} \mapsto \overline{t^x}, z \mapsto t^{\overline{x}}) \approx (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$.

– $\overline{y} = [y : \overline{z}]$ with $y \mapsto \texttt{fvar}\ y' \in \Gamma \wedge y \neq z$.
$\texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ \Gamma \ominus y : \texttt{let}\ \overline{t}[y'/y]\ \texttt{in}\ t[y'/y]$.
$(\Gamma, z \mapsto \texttt{let}\ \overline{t}\ \texttt{in}\ t) \ominus \overline{y} = (\Gamma \ominus y, z \mapsto \texttt{let}\ \overline{t}[y'/y]\ \texttt{in}\ t[y'/y]) \ominus \overline{z} \approx (\Gamma', z \mapsto \texttt{let}\ \overline{t'}\ \texttt{in}\ t')$.
$\overset{I.H.}{\Rightarrow} (\Gamma \ominus y, \overline{x} \mapsto \overline{t}[y'/y]^{\overline{x}}, z \mapsto t[y'/y]^{\overline{x}}) \ominus \overline{z} \approx (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$
$\Rightarrow (\Gamma, \overline{x} \mapsto \overline{t^x}, z \mapsto t^{\overline{x}}) \ominus \overline{y} \approx (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$.

By Proposition 4 we have $(\Gamma, \overline{x} \mapsto \overline{t^x}, z \mapsto t^{\overline{x}}) \gtrsim_I (\Gamma', \overline{x} \mapsto \overline{t'^x}, z \mapsto t'^{\overline{x}})$ and the result is valid for any $z \notin L$ and sufficiently fresh, so that $((\Gamma, \overline{x} \mapsto \overline{t^x}) : t^{\overline{x}}) \gtrsim_I ((\Gamma', \overline{x} \mapsto \overline{t'^x}) : t'^{\overline{x}})$. □

We also need some auxiliary results for heap-context-equivalence.

Lemma 41.

HCE_SUBS $\quad \Gamma \approx^V \Gamma' \wedge y \notin V \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma' \Rightarrow \Gamma[y/x] \approx^{V[y/x]} \Gamma'[y/x]$

IR_SUBS $\quad \Gamma \succsim_I \Gamma' \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma' \Rightarrow \Gamma[y/x] \succsim_I \Gamma'[y/x]$

IR_HT_SUBS $\quad (\Gamma : t) \succsim_I (\Gamma' : t') \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma' : t')$
$\Rightarrow (\Gamma[y/x] : t[y/x]) \succsim_I (\Gamma'[y/x] : t'[y/x])$

Proof.

HCE_SUBS

By rule induction, where the case of the empty heap is immediate.

$$(\Theta, z \mapsto t) \approx^V (\Theta', z \mapsto t') \begin{cases} \Theta \approx^V \Theta' \overset{I.H.}{\Rightarrow} \Theta[y/x] \approx^{V[y/x]} \Theta'[y/x] \\ t \approx^V t' \overset{L15}{\Rightarrow} t[y/x] \approx^{V[y/x]} t'[y/x] \\ \texttt{lc}\ t \Rightarrow \texttt{lc}\ t[y/x] \end{cases}$$

- $x \neq z \Rightarrow z[y/x] = z \notin \texttt{dom}(\Theta) \overset{z \neq y}{\Rightarrow} z[y/x] \notin \texttt{dom}(\Theta[y/x])$.
- $x = z \Rightarrow z[y/x] = y \notin \texttt{dom}(\Theta) \overset{x \notin \texttt{dom}(\Theta)}{\Rightarrow} z[y/x] \notin \texttt{dom}(\Theta[y/x])$.

Therefore, $(\Theta[y/x], z[y/x] \mapsto t[y/x]) \approx^{V[y/x]} (\Theta'[y/x], z[y/x] \mapsto t'[y/x])$.
Thus, $\Gamma[y/x] \approx^{V[y/x]} \Gamma'[y/x]$.

IR_SUBS

By rule induction.

- $\Gamma \approx \Gamma' \Rightarrow \Gamma \approx^{\texttt{dom}(\Gamma)} \Gamma' \overset{\text{HCE\_SUBS}}{\Rightarrow} \Gamma[y/x] \approx^{\texttt{dom}(\Gamma[y/x])} \Gamma'[y/x] \Rightarrow \Gamma[y/x] \succsim_I \Gamma'[y/x]$.
- $\texttt{ok}\ \Gamma \wedge \Gamma \ominus z \succsim_I \Gamma' \wedge z \in \texttt{Ind}(\Gamma)$
  $\texttt{ok}\ \Gamma \overset{y \notin \texttt{dom}(\Gamma)}{\Rightarrow} \texttt{ok}\ \Gamma[y/x]$
  $\Gamma \ominus z \succsim_I \Gamma' \overset{I.H.}{\Rightarrow} (\Gamma \ominus z)[y/x] \succsim_I \Gamma'[y/x]$
  - $x \neq z \Rightarrow z \in \texttt{Ind}(\Gamma[y/x]) \wedge (\Gamma \ominus z)[y/x] = \Gamma[y/x] \ominus z \Rightarrow \Gamma[y/x] \succsim_I \Gamma'[y/x]$.
  - $x = z \Rightarrow y \in \texttt{Ind}(\Gamma[y/x]) \wedge (\Gamma \ominus z)[y/x] = \Gamma[y/x] \ominus y \Rightarrow \Gamma[y/x] \succsim_I \Gamma'[y/x]$.

IR_HT_SUBS

$$\begin{array}{rcl} (\Gamma : t) \succsim_I (\Gamma' : t') & \Rightarrow & \forall z \notin L\ .\ (\Gamma, z \mapsto t) \succsim_I (\Gamma', z \mapsto t'), \text{for some finite } L \subset Id \\ & \overset{\text{IR\_SUBS}}{\Rightarrow} & \forall z \notin L \cup \{y\}\ .\ (\Gamma[y/x], \underbrace{z[y/x]}_{z'} \mapsto t[y/x]) \succsim_I (\Gamma'[y/x], \underbrace{z[y/x]}_{z'} \mapsto t'[y/x]) \\ & \Rightarrow & (\Gamma[y/x] : t[y/x]) \succsim_I (\Gamma'[y/x] : t'[y/x]) \end{array}$$

□

Proposition 7

EQ_IR_AN
$$\begin{array}{l} (\Gamma_A : t_A) \succsim_I (\Gamma_N : t_N) \wedge \\ \forall \overline{x} \notin L \subseteq Id\ .\ \Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : {w_A}^{\overline{x}} \wedge {}^{\backslash \overline{x}}(\overline{s}_A{}^{\overline{x}}) = \overline{s}_A \wedge {}^{\backslash \overline{x}}({w_A}^{\overline{x}}) = w_A \\ \Rightarrow \exists \overline{y} \notin L\ .\ \exists \overline{s}_N \subset LNExp\ .\ \exists w_N \in LNVal\ . \\ \Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : {w_N}^{\overline{z}} \wedge {}^{\backslash \overline{z}}(\overline{s}_N{}^{\overline{z}}) = \overline{s}_N \wedge {}^{\backslash \overline{z}}({w_N}^{\overline{z}}) = w_N \wedge \overline{z} \subseteq \overline{y} \\ \wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : {w_A}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : {w_N}^{\overline{z}}) \end{array}$$

EQ_IR_NA
$$\begin{array}{l} (\Gamma_A : t_A) \succsim_I (\Gamma_N : t_N) \wedge \\ \forall \overline{x} \notin L \subseteq Id\ .\ \Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : {w_N}^{\overline{x}} \wedge {}^{\backslash \overline{x}}(\overline{s}_N{}^{\overline{x}}) = \overline{s}_N \wedge {}^{\backslash \overline{x}}({w_N}^{\overline{x}}) = w_N \\ \Rightarrow \exists \overline{z} \notin L\ .\ \exists \overline{s}_A \subset LNExp\ .\ \exists w_A \in LNVal\ . \\ \Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : {w_A}^{\overline{y}} \wedge {}^{\backslash \overline{y}}(\overline{s}_A{}^{\overline{y}}) = \overline{s}_A \wedge {}^{\backslash \overline{y}}({w_A}^{\overline{y}}) = w_A \wedge \overline{z} \subseteq \overline{y} \\ \wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : {w_A}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : {w_N}^{\overline{z}}) \end{array}$$

Proof.

$$(\Gamma_A : t_A) \precsim_I (\Gamma_N : t_N) \overset{L22}{\Rightarrow} t_A \sim_S t_N.$$

EQ_IR_AN: By rule induction:

- Rule LNLam
  $t_A \equiv \texttt{abs}\ u_A \Rightarrow t_N \equiv \texttt{abs}\ u_N$, and in this case $\overline{x} = [\,]$ (so that $\overline{y} = [\,]$ too).
  It is easy to prove that for any $t \in \mathit{LNExp}$ it is $t^{[\,]} = t = {}^{\backslash [\,]}t$.
  $$\begin{array}{rcl}(\Gamma_A : \texttt{abs}\ u_A) \precsim_I (\Gamma_N : \texttt{abs}\ u_N) & \Rightarrow & \forall z \notin L' \,.\, (\Gamma_A, z \mapsto \texttt{abs}\ u_A) \precsim_I (\Gamma_N, z \mapsto \texttt{abs}\ u_N) \\ & \overset{L20}{\Rightarrow} & \texttt{ok}\ (\Gamma_N, z \mapsto \texttt{abs}\ u_N) \\ & \Rightarrow & \texttt{ok}\ \Gamma_N \wedge \texttt{lc}\ (\texttt{abs}\ u_N) \\ & \overset{\text{LNLam}}{\Rightarrow} & \Gamma_N : \texttt{abs}\ u_N \Downarrow^N \Gamma_N : \texttt{abs}\ u_N, \text{and take } \overline{z} = [\,].\end{array}$$
- Rule ALNVar.
  $t_A \equiv \texttt{fvar}\ x_A \wedge \Gamma_A = (\Gamma'_A, x_A \mapsto u_A)$.
  On the one hand, $\forall \overline{x} \notin L.(\Gamma'_A, x_A \mapsto u_A) : \texttt{fvar}\ x_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w_A{}^{\overline{x}}$
  $\Rightarrow \forall \overline{x} \notin L.(\Gamma'_A, x_A \mapsto u_A) : u_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w_A{}^{\overline{x}}$.
  On the other hand, $t_N \equiv \texttt{fvar}\ x_N$ and, by hypothesis, $((\Gamma'_A, x_A \mapsto u_A) : \texttt{fvar}\ x_A) \precsim_I (\Gamma_N : \texttt{fvar}\ x_N)$
  $\overset{L34}{\Rightarrow} \Gamma_N = (\Gamma'_N, x_N \mapsto u_N)$
  $\overset{L36}{\Rightarrow} ((\Gamma'_A, x_A \mapsto u_A) : u_A) \precsim_I ((\Gamma'_N, x_N \mapsto u_N) : u_N) \vee$
  $((\Gamma'_A, x_A \mapsto u_A) : u_A) \precsim_I ((\Gamma'_N, x_N \mapsto u_N) : \texttt{fvar}\ x_N)$.
  In the first case, by induction hypothesis we have $(\Gamma'_N, x_N \mapsto u_N) : u_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}}$
  and $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \precsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$, for some $\overline{z} \subseteq \overline{y} \notin L$, $\overline{s}_N$ and w_N.
  By applying rule ALNVar to this result we obtain $\Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}}$.
  In the second case, by induction hypothesis we obtain directly
  $(\Gamma'_N, x_N \mapsto u_N) : \texttt{fvar}\ x_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}}$,
  with $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \precsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$ for some $\overline{z} \subseteq \overline{y} \notin L$, $\overline{s}_N$ and w_N.
- Rule ALNApp.
  $t_A \equiv \texttt{app}\ t'_A\ (\texttt{fvar}\ x_A) \Rightarrow t_N \equiv \texttt{app}\ t'_N\ (\texttt{fvar}\ x_N)$.
  By hypothesis, $\forall \overline{x} \notin L \,.\, \Gamma_A : \texttt{app}\ t'_A\ (\texttt{fvar}\ x_A) \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w_A{}^{\overline{x}}$.
  Let us choose a particular $\overline{x} \notin L$ such that $\texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma_A : t_A)$;
  $\overline{x}$ can be decomposed as $\overline{x} = [z : \overline{x}_1 +\!\!+ \overline{x}_2]$ with

  $$\Gamma_A : t'_A \Downarrow^A (\Gamma_A, \overline{x}_1 \mapsto \overline{s}_{1A}{}^{\overline{x}_1}) : \texttt{abs}\ u'_A{}^{\overline{x}_1} \tag{3}$$

  and

  $$(\Gamma_A, \overline{x}_1 \mapsto \overline{s}_{1A}{}^{\overline{x}_1}, z \mapsto \texttt{fvar}\ x_A) : (u'_A{}^{\overline{x}_1})^z \Downarrow^A (\Gamma_A, \overline{x}_1 \mapsto \overline{s}_{1A}{}^{\overline{x}_1}, z \mapsto \texttt{fvar}\ x_A, \overline{x}_2 \mapsto \overline{s}_{2A}{}^{\overline{x}_2}) : w'_A{}^z \tag{4}$$

  where $w'_A{}^z = w_A{}^{\overline{x}}$.
  Let $L' = L \cup \texttt{names}(\Gamma_A : t_A) \cup \texttt{fv}(\overline{s}_{1A}) \cup \texttt{fv}(u'_A) \cup \{x_N\}$; by applying RENAMING2 (Lemma 12) to (3) we obtain
  $$\forall \overline{x}_1 \notin L' \,.\, \Gamma_A : t'_A \Downarrow^A (\Gamma_A, \overline{x}_1 \mapsto \overline{s}_{1A}{}^{\overline{x}_1}) : \texttt{abs}\ u'_A{}^{\overline{x}_1}.$$
  By hypothesis, $(\Gamma_A : \texttt{app}\ t'_A\ (\texttt{fvar}\ x_A)) \precsim_I (\Gamma_N : \texttt{app}\ t'_N\ (\texttt{fvar}\ x_N)) \overset{L37}{\Rightarrow} (\Gamma_A : t'_A) \precsim_I (\Gamma_N : t'_N)$, so that by induction hypothesis
  $$\Gamma_N : t'_N \Downarrow^N (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : w'_N{}^{\overline{z}_1} \ \wedge\ ((\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}) : \texttt{abs}\ u'_A{}^{\overline{y}_1}) \precsim_I ((\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : w'_N{}^{\overline{z}_1})$$
  for some $\overline{y}_1 \notin L'$, and with $\overline{z}_1 \subseteq \overline{y}_1$. It must be that $w'_N{}^{\overline{z}_1} \equiv \texttt{abs}\ u_N$ for some term u_N.
  Now we apply RENAMING1 (Lemma 12) to (4) to obtain
  $$(\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \texttt{fvar}\ x_A) : (u'_A{}^{\overline{y}_1})^z \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \texttt{fvar}\ x_A, \overline{x}_2 \mapsto \overline{s}_{2A}{}^{\overline{x}_2}) : w''_A{}^z$$

where ${w''_A}^z = {w_A}^{[z:\overline{y}_1 +\!\!+ \overline{x}_2]}$.
Let $L'' = L' \cup \mathtt{fv}(\overline{s}_{2A}) \cup \mathtt{fv}(w''_A) \cup \overline{y}_1 \cup \{z\}$; by RENAMING2 (Lemma 12)

$$\forall \overline{x}_2 \notin L'' \,.\, (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \mathtt{fvar}\ x_A) : ({u'_A}^{\overline{y}_1})^z \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \mathtt{fvar}\ x_A, \overline{x}_2 \mapsto \overline{s}_{2A}{}^{\overline{x}_2}) : {w''_A}^z$$

By Lemma 37 we also have $(\Gamma_A : \mathtt{fvar}\ x_A) \gtrsim_I (\Gamma_N : \mathtt{fvar}\ x_N)$.
If $\Delta_A = (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1})$ and $\Delta_N = (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1})$ then $\Gamma_A \subseteq \Delta_A \wedge \Gamma_N \subseteq \Delta_N \wedge \Delta_A \gtrsim_I \Delta_N$, by Lemmas 9 and 22.
Moreover, since $x_A \notin \overline{y}_1$ it is verified that $(x_A \notin \mathtt{dom}(\Gamma_A) \Rightarrow x_A \notin \mathtt{dom}(\Delta_A))$;
likewise $x_N \notin \overline{y}_1 \Rightarrow x_N \notin \overline{z}_1$ so that it is verified also that $(x_N \notin \mathtt{dom}(\Gamma_N) \Rightarrow x_N \notin \mathtt{dom}(\Delta_N))$.
Furthermore, $y \in \mathtt{dom}(\Gamma_A) \Rightarrow y \notin \overline{y}_1 \Rightarrow y \notin \overline{z}_1$, thus, $(y \in \mathtt{dom}(\Gamma_A) \wedge y \notin \mathtt{dom}(\Gamma_N) \Rightarrow y \notin \mathtt{dom}(\Delta_N))$.
Therefore, all the conditions of Lemma 38 are satisfied and we infer that
$(\Delta_A : \mathtt{fvar}\ x_A) \gtrsim_I (\Delta_N : \mathtt{fvar}\ x_N) \overset{L39}{\Rightarrow} ((\Delta_A, z \mapsto \mathtt{fvar}\ x_A) : ({u'_A}^{\overline{y}_1})^z) \gtrsim_I (\Delta_N : {u_N}^{x_N})$.
By induction hypothesis

$$(\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : {u_N}^{x_N} \Downarrow^N (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}, \overline{z}_2 \mapsto \overline{s}_{2N}{}^{\overline{z}_2}) : {w''_N}^{\overline{z}_2} \tag{5}$$

and

$$((\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \mathtt{fvar}\ x_A, \overline{y}_2 \mapsto \overline{s}_{2A}{}^{\overline{y}_2}) : {w_A}^{[z:\overline{y}_1 +\!\!+ \overline{y}_2]}) \gtrsim_I ((\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}, \overline{z}_2 \mapsto \overline{s}_{2N}{}^{\overline{z}_2}) : {w''_N}^{\overline{z}_2})$$

for some $\overline{y}_2 \notin L''$, and with $\overline{z}_2 \subseteq \overline{y}_2$.
Let $\overline{y} = [z : \overline{y}_1 +\!\!+ \overline{y}_2]$ and $\overline{z} = \overline{z}_1 +\!\!+ \overline{z}_2$ (notice that $\overline{z} \subseteq \overline{y}$), then we can rewrite (5) as

$$(\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : {u_N}^{x_N} \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : {w_N}^{\overline{z}}$$

with $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : {w_A}^{\overline{y}}) \gtrsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : {w_N}^{\overline{z}})$.
Finally, by the rule LNAPP the required result is obtained.

– Rule LNLET.
$t_A \equiv \mathtt{let}\ \overline{t}_A\ \mathtt{in}\ t'_A \Rightarrow t_N \equiv \mathtt{let}\ \overline{t}_N\ \mathtt{in}\ t'_N \wedge |\overline{t}_A| = |\overline{t}_N|$.
By hypothesis, $\forall \overline{x} \notin L \,.\, \Gamma_A : \mathtt{let}\ \overline{t}_A\ \mathtt{in}\ t'_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : {w_A}^{\overline{x}}$.
Let us choose a particular $\overline{x} \notin L$ such that $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ (\Gamma_A : \mathtt{let}\ \overline{t}_A\ \mathtt{in}\ t'_A)$, and we decompose the set of names as $\overline{x} = \overline{x}_1 +\!\!+ \overline{x}_2$, with $|\overline{x}_1| = |\overline{t}_A|$.
Then, by the LNLET rule we have that

$$\forall \overline{y_1}^{|\overline{t}_A|} \notin L'.(\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}) : {t'_A}^{\overline{y}_1} \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}, \overline{x}_2 \mapsto \overline{s'}_A{}^{\overline{y}_1}) : {w'_A}^{\overline{y}_1},$$

for some finite $L' \subseteq Id$, with $\overline{x}_1 \notin L'$ and $\overline{s}_A{}^{\overline{x}} = \overline{t}_A{}^{\overline{y}_1} +\!\!+ \overline{s'}_A{}^{\overline{y}_1} \wedge {w_A}^{\overline{x}} = {w'_A}^{\overline{y}_1}$.
Let $L'' = L \cup L' \cup \mathtt{names}(\Gamma_A : t_A) \cup \mathtt{names}(\Gamma_N : t_N)$, and fix a particular $\overline{y}_1 \notin L''$; thanks to Lemma 6, $\overline{s'}_A{}^{\overline{y}_1}$ can be expressed as $\overline{s''}_A{}^{\overline{x}_2}$ and ${w'_A}^{\overline{y}_1}$ as ${w''_A}^{\overline{x}_2}$, with $\backslash^{\overline{x}_2}(\overline{s''}_A{}^{\overline{x}_2}) = \overline{s''}_A \wedge \backslash^{\overline{x}_2}({w''_A}^{\overline{x}_2}) = w''_A$, so that

$$(\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}) : {t'_A}^{\overline{y}_1} \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}, \overline{x}_2 \mapsto \overline{s''}_A{}^{\overline{x}_2}) : {w''_A}^{\overline{x}_2}.$$

By RENAMING2 (Lemma 12) we can obtain

$$\forall \overline{y}_2 \notin L'' \cup \overline{y}_1.(\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}) : {t'_A}^{\overline{y}_1} \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}, \overline{y}_2 \mapsto \overline{s''}_A{}^{\overline{y}_2}) : {w''_A}^{\overline{y}_2}.$$

Moreover, by Lemma 40, $((\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}) : {t'_A}^{\overline{y}_1}) \gtrsim_I ((\Gamma_N, \overline{y}_1 \mapsto \overline{t}_N{}^{\overline{y}_1}) : {t'_N}^{\overline{y}_1})$.
Hence, by induction hypothesis there exists $\overline{y}_2 \notin L'' \cup \overline{y}_1$, and $\overline{s''}_N$ and w''_N such that

$$(\Gamma_N, \overline{y}_1 \mapsto \overline{t}_N{}^{\overline{y}_1}) : {t'_N}^{\overline{y}_1} \Downarrow^N (\Gamma_N, \overline{y}_1 \mapsto \overline{t}_N{}^{\overline{y}_1}, \overline{z}_2 \mapsto \overline{s''}_N{}^{\overline{z}_2}) : {w''_N}^{\overline{z}_2}$$

with ${}^{\backslash\overline{z}_2}(\overline{s''}_N{}^{\overline{z}_2}) = \overline{s''}_N \wedge {}^{\backslash\overline{z}_2}(w''_N{}^{\overline{z}_2}) = w''_N \wedge \overline{z}_2 \subseteq \overline{y}_2$,
and $(\Gamma_A, \overline{y}_1 \mapsto \overline{t}_A{}^{\overline{y}_1}, \overline{y}_2 \mapsto \overline{s''}_A{}^{\overline{y}_2}) : w''_A{}^{\overline{y}_2}) \succsim_I ((\Gamma_N, \overline{y}_1 \mapsto \overline{t}_N{}^{\overline{y}_1}, \overline{z}_2 \mapsto \overline{s''}_N{}^{\overline{z}_2}) : w''_N{}^{\overline{z}_2})$.
Notice that $\overline{y}_1 +\!\!+ \overline{y}_2 \notin L$.
By Lemma 6, $\overline{t}_N{}^{\overline{y}_1} +\!\!+ \overline{s''}_N{}^{\overline{z}_2}$ can be expressed as $\overline{s'}_N{}^{\overline{y}_1}$ and $w''_N{}^{\overline{z}_2}$ as $w'_N{}^{\overline{y}_1}$ with ${}^{\backslash\overline{y}_1}(\overline{s'}_N{}^{\overline{y}_1}) = \overline{s'}_N \wedge {}^{\backslash\overline{y}_1}(w'_N{}^{\overline{y}_1}) = w'_N$.
Let $L''' = L'' \cup \mathtt{fv}(\overline{s}_A) \cup \mathtt{fv}(w_A) \cup \mathtt{fv}(\overline{s'}_N) \cup \mathtt{fv}(w'_N) \cup \overline{y}_1 +\!\!+ \overline{y}_2 \cup \overline{z}_2$;
by RENAMING1 (Lemma 12) we obtain

$$\forall \overline{z}_1 \notin L'''.(\Gamma_N, \overline{z}_1 \mapsto \overline{t}_N{}^{\overline{z}_1}) : t'_N{}^{\overline{z}_1} \Downarrow^N (\Gamma_N, \overline{z}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s'}_N{}^{\overline{z}_1}) : w'_N{}^{\overline{z}_1}.$$

Then, by the LNLET rule we infer that

$$\Gamma_N : \mathtt{let}\ \overline{t}_N\ \mathtt{in}\ t'_N \Downarrow^N (\Gamma_N, \overline{z}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s'}_N{}^{\overline{z}_1}) : w'_N{}^{\overline{z}_1}, \text{for some } \overline{z}_1 \notin L'''.$$

Once again, by Lemma 6, we rewrite $\overline{s'}_N{}^{\overline{z}_1} = \overline{s}_N{}^{\overline{z}_1 +\!\!+ \overline{z}_2}$ and $w'_N{}^{\overline{z}_1} = w_N{}^{\overline{z}_1 +\!\!+ \overline{z}_2}$.
Besides, notice that $\overline{t}_A{}^{\overline{y}_1} +\!\!+ \overline{s''}_A{}^{\overline{y}_2} = \overline{s}_A{}^{\overline{z}_1 +\!\!+ \overline{y}_2}$ and $w''_A{}^{\overline{y}_2} = w_A{}^{\overline{y}_1 +\!\!+ \overline{y}_2}$; hence
$((\Gamma_A, \overline{y}_1 +\!\!+ \overline{y}_2 \mapsto \overline{s}_A{}^{\overline{y}_1 +\!\!+ \overline{y}_2}) : w_A{}^{\overline{y}_1 +\!\!+ \overline{y}_2}) \succsim_I ((\Gamma_N, \overline{y}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s}_N{}^{\overline{y}_1 +\!\!+ \overline{z}_2}) : w_N{}^{\overline{y}_1 +\!\!+ \overline{z}_2})$
$\overset{L41}{\Rightarrow} (\Gamma_A, \overline{z}_1 +\!\!+ \overline{y}_2 \mapsto \overline{s}_A{}^{\overline{z}_1 +\!\!+ \overline{y}_2}) : w_A{}^{\overline{z}_1 +\!\!+ \overline{y}_2}) \succsim_I ((\Gamma_N, \overline{z}_1 +\!\!+ \overline{z}_2 \mapsto \overline{s}_N{}^{\overline{z}_1 +\!\!+ \overline{z}_2}) : w_N{}^{\overline{z}_1 +\!\!+ \overline{z}_2})$,
and notice that $\overline{z}_1 +\!\!+ \overline{z}_2 \subseteq \overline{z}_1 +\!\!+ \overline{y}_2 \notin L$.

EQ_IR_NA: By rule induction.

For rules LNLAM and LNLET we follow similar reasonings as in EQ_IR_AN. Therefore, we detail only the cases of rules ALNVAR and LNAPP:

– Rule ALNVAR.
$t_N \equiv \mathtt{fvar}\ x_N \wedge \Gamma_N = (\Gamma'_N, x_N \mapsto u_N)$.
On the one hand, $\forall \overline{x} \notin L.(\Gamma'_N, x_N \mapsto u_N) : \mathtt{fvar}\ x_N \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : w_N{}^{\overline{x}}$
$\Rightarrow \forall \overline{x} \notin L.(\Gamma'_N, x_N \mapsto u_N) : u_N \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : w_N{}^{\overline{x}}$.
On the other hand, $t_A \equiv \mathtt{fvar}\ x_A$ and, by hypothesis, $(\Gamma_A : \mathtt{fvar}\ x_A) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : \mathtt{fvar}\ x_N)$
$\overset{L34}{\Rightarrow} \Gamma_A = (\Gamma'_A, x_A \mapsto u_A) \wedge$
$\exists x_0, \ldots, x_n \in Id\,.\, x_0 = x_A \wedge x_n = x_N \wedge \forall i : 0 \leq i < n\,.\, x_i \mapsto \mathtt{fvar}\ x_{i+1} \in (\Gamma'_A, x_A \mapsto u_A)$.
Now we proceed by induction on n to prove that $(\Gamma'_A, x_A \mapsto u_A) : u_A \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$
and $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$ for some $\overline{z} \subseteq \overline{y} \notin L$.
By applying rule ALNVAR to this result we obtain $\Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$.
- $n = 0 \Rightarrow x_A = x_N$
$\Rightarrow ((\Gamma'_A, x_A \mapsto u_A) : \mathtt{fvar}\ x_A) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : \mathtt{fvar}\ x_A)$
$\overset{L22}{\Rightarrow} (\Gamma'_A, x_A \mapsto u_A) \succsim_I (\Gamma'_N, x_N \mapsto u_N)$
$\overset{L35}{\Rightarrow} ((\Gamma'_A, x_A \mapsto u_A) : u_A) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : u_N)$.
By (rule) induction hypothesis we are done.
- $n > 0 \Rightarrow ((\Gamma'_A, x_A \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x_A) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : \mathtt{fvar}\ x_N)$
$\overset{L36}{\Rightarrow} ((\Gamma'_A, x_A \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x_1) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : u_N) \vee$
$((\Gamma'_A, x_A \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x_1) \succsim_I ((\Gamma'_N, x_N \mapsto u_N) : \mathtt{fvar}\ x_N)$.
In the first case, by (rule) induction hypothesis we are done.
In the second case, by Lemma 34, we have $x_1 \in \mathtt{dom}(\Gamma_A)$, so that $\Gamma_A = (\Gamma''_A, x_1 \mapsto u_1)$.
Since the path from x_1 to x_N is of length $n-1$, the induction hypothesis indicates that there exists $\overline{z} \subseteq \overline{y} \notin L$, such that
$(\Gamma''_A, x_1 \mapsto u_1) : u_1 \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$ and $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$.
By applying rule ALNVAR, we obtain $(\Gamma''_A, x_1 \mapsto u_1) : \mathtt{fvar}\ x_1 \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$, i.e.,
$(\Gamma'_A, x_A \mapsto \mathtt{fvar}\ x_1) : \mathtt{fvar}\ x_1 \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$.

– Rule LNApp.
$t_N \equiv \mathtt{app}\ t'_N\ (\mathtt{fvar}\ x_N) \Rightarrow t_A \equiv \mathtt{app}\ t'_A\ (\mathtt{fvar}\ x_A)$.
By hypothesis, $\forall \overline{x} \notin L\,.\,\Gamma_N : \mathtt{app}\ t'_N\ (\mathtt{fvar}\ x_N) \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : w_N{}^{\overline{x}}$.
Let us choose a particular $\overline{x} \notin L$ such that $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ (\Gamma_N : t_N)$;
$\overline{x}$ can be decomposed as $\overline{x} = \overline{x}_1 +\!\!+ \overline{x}_2$ with

$$\Gamma_N : t'_N \Downarrow^N (\Gamma_N, \overline{x}_1 \mapsto \overline{s}_{1N}{}^{\overline{x}_1}) : \mathtt{abs}\ u'_N{}^{\overline{x}_1} \tag{6}$$

and

$$(\Gamma_N, \overline{x}_1 \mapsto \overline{s}_{1N}{}^{\overline{x}_1}) : (u'_N{}^{\overline{x}_1})^{x_N} \Downarrow^N (\Gamma_N, \overline{x}_1 \mapsto \overline{s}_{1N}{}^{\overline{x}_1}, \overline{x}_2 \mapsto \overline{s}_{2N}{}^{\overline{x}_2}) : w'_N{}^{\overline{x}_2} \tag{7}$$

where $w'_N{}^{\overline{x}_2} = w_N{}^{\overline{x}}$.
Let $L' = L \cup \mathtt{names}(\Gamma_N : t_N) \cup \mathtt{fv}(\overline{s}_{1N}) \cup \mathtt{fv}(u'_N)$; by applying RENAMING2 (Lemma 12) to (6) we obtain

$$\forall \overline{x}_1 \notin L'\,.\,\Gamma_N : t'_N \Downarrow^N (\Gamma_N, \overline{x}_1 \mapsto \overline{s}_{1N}{}^{\overline{x}_1}) : \mathtt{abs}\ u'_N{}^{\overline{x}_1}.$$

By hypothesis, $(\Gamma_A : \mathtt{app}\ t'_A\ (\mathtt{fvar}\ x_A)) \precsim_I (\Gamma_N : \mathtt{app}\ t'_N\ (\mathtt{fvar}\ x_N)) \overset{L37}{\Rightarrow} (\Gamma_A : t'_A) \precsim_I (\Gamma_N : t'_N)$, so that by induction hypothesis

$$\Gamma_A : t'_A \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}) : w'_A{}^{\overline{y}_1} \wedge ((\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}) : w'_A{}^{\overline{y}_1}) \precsim_I ((\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : \mathtt{abs}\ u'_N{}^{\overline{z}_1})$$

for some $\overline{z}_1 \notin L'$, and with $\overline{z}_1 \subseteq \overline{y}_1$. It must be that $w'_A{}^{\overline{y}_1} \equiv \mathtt{abs}\ u_A$ for some term u_A.
Now we apply RENAMING1 (Lemma 12) to (7) to obtain

$$(\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : (u'_N{}^{\overline{z}_1})^{x_N} \Downarrow^N (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}, \overline{x}_2 \mapsto \overline{s}_{2N}{}^{\overline{x}_2}) : w''_N{}^{\overline{x}_2}$$

where $w''_N{}^{\overline{x}_2} = w_N{}^{\overline{z}_1 +\!\!+ \overline{x}_2}$.
Let $L'' = L' \cup \mathtt{fv}(\overline{s}_{2N}) \cup \mathtt{fv}(w''_N) \cup \overline{z}_1$; by RENAMING2 (Lemma 12)

$$\forall \overline{x}_2 \notin L''\,.\,(\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}) : (u'_N{}^{\overline{z}_1})^{x_N} \Downarrow^N (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}, \overline{x}_2 \mapsto \overline{s}_{2N}{}^{\overline{x}_2}) : w''_N{}^{\overline{x}_2}$$

By Lemma 37 we also have $(\Gamma_A : \mathtt{fvar}\ x_A) \precsim_I (\Gamma_N : \mathtt{fvar}\ x_N)$.
If $\Delta_A = (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1})$ and $\Delta_N = (\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1})$ then $\Gamma_A \subseteq \Delta_A \wedge \Gamma_N \subseteq \Delta_N \wedge \Delta_A \precsim_I \Delta_N$, by Lemmas 9 and 22.
On the one hand, $x_A \notin \mathtt{dom}(\Gamma_A) \overset{L11}{\Rightarrow} x_A \notin \mathtt{dom}(\Delta_A)$; on the other hand, since $x_N \notin \overline{z}_1$ it is verified that $(x_N \notin \mathtt{dom}(\Gamma_N) \Rightarrow x_N \notin \mathtt{dom}(\Delta_N))$.
Furthermore, $y \in \mathtt{dom}(\Gamma_A) \Rightarrow y \notin \overline{y}_1 \Rightarrow y \notin \overline{z}_1$, thus, $(y \in \mathtt{dom}(\Gamma_A) \wedge y \notin \mathtt{dom}(\Gamma_N) \Rightarrow y \notin \mathtt{dom}(\Delta_N))$.
Therefore, all the conditions of Lemma 38 are satisfied and we infer that
$(\Delta_A : \mathtt{fvar}\ x_A) \precsim_I (\Delta_N : \mathtt{fvar}\ x_N)$
$\overset{L39}{\Rightarrow} ((\Delta_A, y \mapsto \mathtt{fvar}\ x_A) : u_A{}^y) \precsim_I (\Delta_N : (u'_N{}^{\overline{z}_1})^{x_N})$,
for any $y \in Id$ such that $\mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma_A : u_A) \wedge y \neq x_A$.
By induction hypothesis

$$(\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, y \mapsto \mathtt{fvar}\ x_A) : u_A{}^y \Downarrow^A (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, y \mapsto \mathtt{fvar}\ x_A, \overline{y}_2 \mapsto \overline{s}_{2A}{}^{\overline{y}_2}) : w''_A{}^{\overline{y}_2} \tag{8}$$

and

$$((\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, z \mapsto \mathtt{fvar}\ x_A, \overline{y}_2 \mapsto \overline{s}_{2A}{}^{\overline{y}_2}) : w''_A{}^{\overline{y}_2}) \precsim_I ((\Gamma_N, \overline{z}_1 \mapsto \overline{s}_{1N}{}^{\overline{z}_1}, \overline{z}_2 \mapsto \overline{s}_{2N}{}^{\overline{z}_2}) : w''_N{}^{\overline{z}_2})$$

for some $\overline{z}_2 \notin L''$, and with $\overline{z}_2 \subseteq \overline{y}_2$.
Let $\overline{y} = [y : \overline{y}_1 +\!\!+ \overline{y}_2]$ and $\overline{z} = \overline{z}_1 +\!\!+ \overline{z}_2$ (notice that $\overline{z} \subseteq \overline{y}$), then we can rewrite (8) as

$$(\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, y \mapsto \mathtt{fvar}\ x_A) : u_A{}^y \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$$

with $((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \succsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$.
Finally, let $L''' = \mathtt{names}(\Gamma_A) \cup \mathtt{fv}(\overline{s}_A) \cup \mathtt{fv}(u_A) \cup \mathtt{fv}(w_A) \cup \{x_A\} \cup \overline{y}$, we apply RENAMING1 (Lemma 12) to (8) to obtain

$$\forall y \notin L''' \,.\, (\Gamma_A, \overline{y}_1 \mapsto \overline{s}_{1A}{}^{\overline{y}_1}, y \mapsto \mathtt{fvar}\ x_A) : u_A{}^y \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}$$

so that the required result is obtained by the rule ALNAPP.

□

Lemma 42.

IRHT_REF $\quad \mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \Rightarrow (\Gamma : t) \succsim_I (\Gamma : t)$

Proof.

$\mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \Rightarrow \forall z \notin \mathtt{dom}(\Gamma) \,.\, \mathtt{ok}\ (\Gamma, z \mapsto t) \overset{P5}{\Rightarrow} \forall z \notin \mathtt{dom}(\Gamma) \,.\, (\Gamma, z \mapsto t) \succsim_I (\Gamma, z \mapsto t) \Rightarrow (\Gamma : t) \succsim_I (\Gamma : t)$.

□

Theorem 1 (Equivalence)

EQ_AN $\quad \Gamma : t \Downarrow^A \Delta_A : w_A \Rightarrow$
$\exists \Delta_N \in LNHeap \,.\, \exists w_N \in LNVal \,.\, \Gamma : t \Downarrow^N \Delta_N : w_N \wedge (\Delta_A : w_A) \succsim_I (\Delta_N : w_N)$

EQ_NA $\quad \Gamma : t \Downarrow^N \Delta_N : w_N \Rightarrow$
$\exists \Delta_A \in LNHeap \,.\, \exists w_A \in LNVal \,.\, \exists \overline{x} \subseteq \mathtt{dom}(\Delta_N) - \mathtt{dom}(\Gamma) \,.\, \exists \overline{y} \subseteq Id \,.\, |\overline{x}| = |\overline{y}| \wedge$
$\Gamma : t \Downarrow^A \Delta_A : w_A \wedge (\Delta_A : w_A) \succsim_I (\Delta_N[\overline{y}/\overline{x}] : w_N[\overline{y}/\overline{x}])$

Proof.

EQ_AN

Assume $\Gamma : t \Downarrow^A \Delta_A : w_A$, then by Lemmas 9 and 6 the final heap and value can be written as $\Delta_A = (\Gamma, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}})$ and $w_A = w'_A{}^{\overline{x}}$ with $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ \overline{s}_A$ and $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ w'_A$.
Let $L = \mathtt{names}(\Gamma : t) \cup \mathtt{names}(\Delta_A : w_A) = \mathtt{names}(\Gamma : t) \cup \overline{x} \cup \mathtt{fv}(\overline{s}_A) \cup \mathtt{fv}(w'_A)$,
then by RENAMING2 (Lemma 12):

$$\forall \overline{y} \notin L.\Gamma : t \Downarrow^A \Delta_A[\overline{y}/\overline{x}] : w_A[\overline{y}/\overline{x}]$$

that is

$$\forall \overline{y} \notin L.\Gamma : t \Downarrow^A (\Gamma, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}})[\overline{y}/\overline{x}] : w'_A{}^{\overline{x}}[\overline{y}/\overline{x}].$$

But $\overline{x} = \mathtt{dom}(\Delta_A) - \mathtt{dom}(\Gamma) \overset{L11}{\Rightarrow} \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ (\Gamma : t)$, so that $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ \Gamma, \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ \overline{s}_A$ and $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ w'_A$.
Therefore,

$$\forall \overline{y} \notin L.\Gamma : t \Downarrow^A (\Gamma, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w'_A{}^{\overline{y}},$$

with $\backslash^{\overline{y}}(\overline{s}_A{}^{\overline{y}}) = \overline{s}_A \wedge \backslash^{\overline{y}}(w'_A{}^{\overline{y}}) = w'_A$ (by Lemma 3).
Moreover, by REGULARITY (Lemma 7) we have $\mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \overset{L42}{\Rightarrow} (\Gamma : t) \succsim_I (\Gamma : t)$.
By Proposition 7,

$$\exists \Delta_N \,.\, \exists w_N \,.\, \Gamma : t \Downarrow^N \Delta_N : w_N \wedge ((\Gamma, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w'_A{}^{\overline{y}}) \succsim_I (\Delta_N : w_N), \text{for some } \overline{y} \notin L,$$

with $(\Delta_N : w_N) = ((\Delta, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w'_N{}^{\overline{z}})$, where $\mathtt{fresh}\ \overline{z}\ \mathtt{in}\ \overline{s}_N, \mathtt{fresh}\ \overline{z}\ \mathtt{in}\ w'_N$ and $\overline{z} \subseteq \overline{y}$.
From $\overline{y} \notin L$ and $\overline{x} \subseteq L$ we infer that $\overline{y}$ and $\overline{x}$ are disjoint. $\overline{z}$ and $\overline{x}$ are disjoint too.
Hence, $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ ((\Gamma, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w'_A{}^{\overline{y}}) \overset{L20}{\Rightarrow} \overline{x} \notin \mathtt{dom}(\Delta_N)$.
The later result, together with $\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ (\Gamma : t) \overset{L10}{\Rightarrow} \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ (\Delta_N : w_N)$.
Now by RENAMING3 (Corollary 2): $\Gamma : t \Downarrow^N \Delta_N[\overline{x}/\overline{y}] : w_N[\overline{x}/\overline{y}]$.
Besides, by Lemma 41,

$(\Delta_A : w_A) = ((\Gamma, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w'_A{}^{\overline{x}}) = ((\Gamma, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}})[\overline{x}/\overline{y}] : w'_A{}^{\overline{y}}[\overline{x}/\overline{y}]) \succsim_I (\Delta_N[\overline{x}/\overline{y}] : w_N[\overline{x}/\overline{y}])$.
Therefore, for $(\Delta'_N : w'_N) = (\Delta_N[\overline{x}/\overline{y}] : w_N[\overline{x}/\overline{y}])$ we have:

$$\Gamma : t \Downarrow^N \Delta'_N : w'_N \wedge (\Delta_A : w_A) \succsim_I (\Delta'_N : w'_N).$$

EQ_NA
Assume $\Gamma : t \Downarrow^N \Delta_N : w_N$, then by Lemmas 9 and 6 the final heap and value can be written as $\Delta_N = (\Gamma, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}})$ and $w_N = w'_N{}^{\overline{x}}$ with $\texttt{fresh}\ \overline{x}\ \texttt{in}\ \overline{s}_N$ and $\texttt{fresh}\ \overline{x}\ \texttt{in}\ w'_N$.
Let $L = \texttt{names}(\Gamma : t) \cup \texttt{names}(\Delta_N : w_N) = \texttt{names}(\Gamma : t) \cup \overline{x} \cup \texttt{fv}(\overline{s}_N) \cup \texttt{fv}(w'_N)$,
then by RENAMING2 (Lemma 12):

$$\forall \overline{y} \notin L. \Gamma : t \Downarrow^N \Delta_N[\overline{y}/\overline{x}] : w_N[\overline{y}/\overline{x}].$$

But $\overline{x} = \texttt{dom}(\Delta_N) - \texttt{dom}(\Gamma) \overset{L11}{\Rightarrow} \texttt{fresh}\ \overline{x}\ \texttt{in}\ \Gamma$, so that $\texttt{fresh}\ \overline{x}\ \texttt{in}\ \Gamma, \texttt{fresh}\ \overline{x}\ \texttt{in}\ \overline{s}_N$ and $\texttt{fresh}\ \overline{x}\ \texttt{in}\ w'_N$.
Therefore,

$$\forall \overline{y} \notin L. \Gamma : t \Downarrow^N (\Gamma, \overline{y} \mapsto \overline{s}_N{}^{\overline{y}}) : w'_N{}^{\overline{y}},$$

with $\backslash^{\overline{y}}(\overline{s}_N{}^{\overline{y}}) = \overline{s}_N \wedge \backslash^{\overline{y}}(w'_N{}^{\overline{y}}) = w'_N$ (by Lemma 3).
Moreover, by REGULARITY (Lemma 7) we have $\texttt{ok}\ \Gamma \wedge \texttt{lc}\ t \overset{L42}{\Rightarrow} (\Gamma : t) \succsim_I (\Gamma : t)$.
By Proposition 7,

$$\exists \Delta_A\,.\,\exists w_A\,.\,\Gamma : t \Downarrow^A \Delta_A : w_A \wedge (\Delta_A : w_A) \succsim_I ((\Gamma, \overline{y} \mapsto \overline{s}_N{}^{\overline{y}}) : w'_N{}^{\overline{y}}), \text{for some } \overline{y} \notin L.$$

But $((\Gamma, \overline{y} \mapsto \overline{s}_N{}^{\overline{y}}) : w'_N{}^{\overline{y}}) = ((\Gamma, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}})[\overline{y}/\overline{x}] : w'_N{}^{\overline{x}}[\overline{y}/\overline{x}]) = (\Delta_N[\overline{y}/\overline{x}] : w_N[\overline{y}/\overline{x}])$.

□

7.6 List of Theorems, Propositions, Lemmas and Corollaries

Theorem 1.

EQ_AN $\Gamma : t \Downarrow^A \Delta_A : w_A \Rightarrow$
$\exists \Delta_N \in \mathit{LNHeap}\,.\,\exists w_N \in \mathit{LNVal}\,.\,\Gamma : t \Downarrow^N \Delta_N : w_N \wedge (\Delta_A : w_A) \succsim_I (\Delta_N : w_N)$

EQ_NA $\Gamma : t \Downarrow^N \Delta_N : w_N \Rightarrow$
$\exists \Delta_A \in \mathit{LNHeap}\,.\,\exists w_A \in \mathit{LNVal}\,.\,\exists \overline{x} \subseteq \texttt{dom}(\Delta_N) - \texttt{dom}(\Gamma)\,.\,\exists \overline{y} \subseteq \mathit{Id}\,.\,|\overline{x}| = |\overline{y}| \wedge$
$\Gamma : t \Downarrow^A \Delta_A : w_A \wedge (\Delta_A : w_A) \succsim_I (\Delta_N[\overline{y}/\overline{x}] : w_N[\overline{y}/\overline{x}])$

Proposition 1.

SS_REF $t \sim_S t$
SS_SIM $t \sim_S t' \Rightarrow t' \sim_S t$
SS_TRANS $t \sim_S t' \wedge t \sim_S t'' \Rightarrow t \sim_S t''$

Proposition 2.

CE_REF $t \approx^V t$
CE_SYM $t \approx^V t' \Rightarrow t' \approx^V t$
CE_TRANS $t \approx^V t' \wedge t' \approx^V t'' \Rightarrow t \approx^V t''$

Proposition 3.

HCE_REF $\texttt{ok}\ \Gamma \Rightarrow \Gamma \approx^V \Gamma$
HCE_SYM $\Gamma \approx^V \Gamma' \Rightarrow \Gamma' \approx^V \Gamma$
HCE_TRANS $\Gamma \approx^V \Gamma' \wedge \Gamma' \approx^V \Gamma'' \Rightarrow \Gamma \approx^V \Gamma''$

Proposition 4.

IR_ALT $\Gamma \succsim_I \Gamma' \Leftrightarrow \texttt{ok}\ \Gamma \wedge \exists\, \overline{x} \subseteq \texttt{Ind}(\Gamma)\ .\ \Gamma \ominus \overline{x} \approx \Gamma'$

Proposition 5.

IR_REF $\quad \mathtt{ok}\ \Gamma \Rightarrow \Gamma \gtrsim_I \Gamma$

IR_TRANS $\quad \Gamma \gtrsim_I \Gamma' \wedge \Gamma' \gtrsim_I \Gamma'' \Rightarrow \Gamma \gtrsim_I \Gamma''$

Proposition 6.

IREQ_REF $\quad \mathtt{ok}\ \Gamma \Rightarrow [\Gamma] \gtrsim_I [\Gamma]$

IREQ_ANTSYM $\quad [\Gamma] \gtrsim_I [\Gamma'] \wedge [\Gamma'] \gtrsim_I [\Gamma] \Rightarrow [\Gamma] = [\Gamma']$

IREQ_TRANS $\quad [\Gamma] \gtrsim_I [\Gamma'] \wedge [\Gamma'] \gtrsim_I [\Gamma''] \Rightarrow [\Gamma] \gtrsim_I [\Gamma'']$

Proposition 7.

EQ_IR_AN
$$(\Gamma_A : t_A) \gtrsim_I (\Gamma_N : t_N) \wedge$$
$$\forall \overline{x} \notin L \subseteq Id\,.\,\Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{x} \mapsto \overline{s}_A{}^{\overline{x}}) : w_A{}^{\overline{x}} \wedge {}^{\backslash \overline{x}}(\overline{s}_A{}^{\overline{x}}) = \overline{s}_A \wedge {}^{\backslash \overline{x}}(w_A{}^{\overline{x}}) = w_A$$
$$\Rightarrow \exists \overline{y} \notin L\,.\,\exists \overline{s}_N \subset \mathit{LNExp}\,.\,\exists w_N \in \mathit{LNVal}\,.$$
$$\Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}} \wedge {}^{\backslash \overline{z}}(\overline{s}_N{}^{\overline{z}}) = \overline{s}_N \wedge {}^{\backslash \overline{z}}(w_N{}^{\overline{z}}) = w_N \wedge \overline{z} \subseteq \overline{y}$$
$$\wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \gtrsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$$

EQ_IR_NA
$$(\Gamma_A : t_A) \gtrsim_I (\Gamma_N : t_N) \wedge$$
$$\forall \overline{x} \notin L \subseteq Id\,.\,\Gamma_N : t_N \Downarrow^N (\Gamma_N, \overline{x} \mapsto \overline{s}_N{}^{\overline{x}}) : w_N{}^{\overline{x}} \wedge {}^{\backslash \overline{x}}(\overline{s}_N{}^{\overline{x}}) = \overline{s}_N \wedge {}^{\backslash \overline{x}}(w_N{}^{\overline{x}}) = w_N$$
$$\Rightarrow \exists \overline{z} \notin L\,.\,\exists \overline{s}_A \subset \mathit{LNExp}\,.\,\exists w_A \in \mathit{LNVal}\,.$$
$$\Gamma_A : t_A \Downarrow^A (\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}} \wedge {}^{\backslash \overline{y}}(\overline{s}_A{}^{\overline{y}}) = \overline{s}_A \wedge {}^{\backslash \overline{y}}(w_A{}^{\overline{y}}) = w_A \wedge \overline{z} \subseteq \overline{y}$$
$$\wedge ((\Gamma_A, \overline{y} \mapsto \overline{s}_A{}^{\overline{y}}) : w_A{}^{\overline{y}}) \gtrsim_I ((\Gamma_N, \overline{z} \mapsto \overline{s}_N{}^{\overline{z}}) : w_N{}^{\overline{z}})$$

Lemma 1.

SS_SUBST $\quad t[y/x] \sim_S t$

Lemma 2.

SS_OP $\quad |\overline{x}| = |\overline{y}| \Rightarrow t^{\overline{x}} \sim_S t^{\overline{y}}$

Lemma 3.

CLOSE_OPEN $\quad \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ t \Leftrightarrow {}^{\backslash \overline{x}}(t^{\overline{x}}) = t$

Lemma 4.

OPEN_CLOSE $\quad \mathtt{lc}\ t \Rightarrow ({}^{\backslash \overline{x}}t)^{\overline{x}} = t$

Lemma 5.

SS_LC $\quad t \sim_S t' \wedge \mathtt{lc}\ t \Rightarrow \mathtt{lc}\ t'$

Lemma 6.

LC_OP_VARS $\quad \mathtt{lc}\ t \wedge \overline{x} \subseteq Id \Rightarrow \exists s \in \mathit{LNExp}.\,(\mathtt{fresh}\ \overline{x}\ \mathtt{in}\ s \wedge s^{\overline{x}} = t)$

Lemma 7.

REGULARITY $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{lc}\ t \wedge \mathtt{ok}\ \Delta \wedge \mathtt{lc}\ w$

Lemma 8.

DEF_NOT_LOST $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{dom}(\Gamma) \subseteq \mathtt{dom}(\Delta)$

Lemma 9.

NO_UPDATE $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \Gamma \subseteq \Delta$
where $\Downarrow^K$ represents $\Downarrow^N$ and $\Downarrow^A$

Lemma 10.

ADD_NAMES $\quad \Gamma : t \Downarrow^K \Delta : w \Rightarrow \mathtt{names}(\Delta : w) \subseteq \mathtt{names}(\Gamma : t) \cup \mathtt{dom}(\Delta)$

Lemma 11.

NEW_NAMES1 $\quad \Gamma : t \Downarrow^N \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ \Gamma$

NEW_NAMES2 $\quad \Gamma : t \Downarrow^A \Delta : w \wedge x \in \mathtt{dom}(\Delta) - \mathtt{dom}(\Gamma) \Rightarrow \mathtt{fresh}\ x\ \mathtt{in}\ (\Gamma : t)$

Lemma 12.

RENAMING1 $\quad \Gamma : t \Downarrow^K \Delta : w \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : t) \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Delta : w)$
$\Rightarrow \Gamma[y/x] : t[y/x] \Downarrow^K \Delta[y/x] : w[y/x]$

RENAMING2 $\quad \Gamma : t \Downarrow^K \Delta : w \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Gamma : t) \wedge \mathtt{fresh}\ y\ \mathtt{in}\ (\Delta : w) \wedge x \notin \mathtt{dom}(\Gamma) \wedge x \in \mathtt{dom}(\Delta)$
$\Rightarrow \Gamma : t \Downarrow^K \Delta[y/x] : w[y/x]$

Lemma 13.

CE_SS $\quad t \approx^V t' \Rightarrow t \sim_S t'$

Lemma 14.

CE_SUB $\quad t \approx^V t' \wedge V' \subseteq V \Rightarrow t \approx^{V'} t'$

CE_ADD $\quad t \approx^V t' \wedge \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ t \wedge \mathtt{fresh}\ \overline{x}\ \mathtt{in}\ t' \Rightarrow t \approx^{V \cup \overline{x}} t'$

Lemma 15.

CE_SUBS1 $\quad t \approx^V t' \wedge x, y \notin V \Rightarrow t[y/x] \approx^V t'$

CE_SUBS2 $\quad t \approx^V t' \wedge (\mathtt{fvar}\ y) \approx^V (\mathtt{fvar}\ y') \wedge x \in V \Rightarrow t[y/x] \approx^V t'[y'/x]$

CE_SUBS3 $\quad t \approx^V t' \wedge y \notin V \wedge \mathtt{fresh}\ y\ \mathtt{in}\ t \wedge \mathtt{fresh}\ y\ \mathtt{in}\ t' \Rightarrow t[y/x] \approx^{V[y/x]} t'[y/x]$

CE_OP1 $\quad t \approx^V t' \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{x}}$

CE_OP2 $\quad t \approx^V t' \wedge \overline{x}, \overline{y} \notin V \wedge |\overline{x}| = |\overline{y}| \Rightarrow t^{\overline{x}} \approx^V t'^{\overline{y}}$

CE_OP3 $\quad t \approx^V t' \wedge (\mathtt{fvar}\ x) \approx^V (\mathtt{fvar}\ y) \Rightarrow t^x \approx^V t'^y$

Lemma 16.

HCE_DOM $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma')$

HCE_IND $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{Ind}(\Gamma) = \mathtt{Ind}(\Gamma')$

HCE_OK $\quad \Gamma \approx^V \Gamma' \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma'$

Lemma 17.

HCE_ALT $\quad \Gamma \approx^V \Gamma' \Leftrightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma' \wedge \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \wedge (x \mapsto t \in \Gamma \wedge x \mapsto t' \in \Gamma' \Rightarrow t \approx^V t')$

Lemma 18.

HCE_SWAP $\quad \mathtt{ok}\ \Gamma \wedge x, y \in \mathtt{Ind}(\Gamma) \wedge x \neq y \Rightarrow \Gamma \ominus [x, y] \approx^{V - \{x,y\}} \Gamma \ominus [y, x]$

Lemma 19.

HE_PERM $\quad \mathtt{ok}\ \Gamma \wedge \overline{x}, \overline{y} \in \mathtt{Ind}(\Gamma) \Rightarrow (\Gamma \ominus \overline{x} \approx \Gamma \ominus \overline{y} \Leftrightarrow \overline{y} \in \mathcal{S}(\overline{x}))$

Lemma 20.

IR_DOM $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{dom}(\Gamma') \subseteq \mathtt{dom}(\Gamma)$

IR_IND $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{Ind}(\Gamma') \subseteq \mathtt{Ind}(\Gamma)$

IR_OK $\quad \Gamma \succsim_I \Gamma' \Rightarrow \mathtt{ok}\ \Gamma \wedge \mathtt{ok}\ \Gamma'$

IR_DOM_HE $\quad \Gamma \succsim_I \Gamma' \wedge \mathtt{dom}(\Gamma) = \mathtt{dom}(\Gamma') \Rightarrow \Gamma \approx \Gamma'$

IR_IR_HE $\quad (\Gamma \succsim_I \Gamma' \wedge \Gamma' \succsim_I \Gamma) \Leftrightarrow \Gamma \approx \Gamma'$

Lemma 21.

IREQ_HE_IREQ1 $\quad [\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma \Rightarrow [\Delta] \succsim_I [\Gamma']$

IREQ_HE_IREQ2 $\quad [\Gamma] \succsim_I [\Gamma'] \wedge \Delta \approx \Gamma' \Rightarrow [\Gamma] \succsim_I [\Delta]$

Lemma 22.

IRHT_IRH $\quad (\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \Gamma \succsim_I \Gamma'$

IRHT_SS $\quad (\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow t \sim_S t'$

IRHT_LC $\quad (\Gamma : t) \succsim_I (\Gamma' : t') \Rightarrow \mathtt{lc}\ t \wedge \mathtt{lc}\ t'$

Lemma 23.
SS_OPK $\quad t \sim_S t' \wedge |\overline{x}| = |\overline{y}| \Rightarrow \{k \to \overline{x}\}t \sim_S \{k \to \overline{y}\}t'$

Lemma 24.
OPK_CLK_FRESH $\quad \{k \leftarrow \overline{x}\}(\{k \to \overline{x}\}t) = t \Rightarrow \texttt{fresh}\ \overline{x}\ \texttt{in}\ t$

Lemma 25.
LC_OPK_VARS $\quad \texttt{lc\_at}\ k\ \overline{n}\ t \wedge \overline{x} \subseteq Id \Rightarrow \exists s \in \mathit{LNExp}.\, (\texttt{fresh}\ \overline{x}\ \texttt{in}\ s \wedge \{k \to \overline{x}\}s = t)$

Lemma 26.
KEEP_NAMES $\quad \Gamma : t \Downarrow^A \Delta : w \Rightarrow \texttt{fv}(t) \subseteq \texttt{names}(\Delta : w)$

Lemma 27.
CE_OPK1 $\quad t \approx^V t' \Rightarrow \{k \to \overline{x}\}t \approx^V \{k \to \overline{x}\}t'$
CE_OPK2 $\quad t \approx^V t' \wedge \overline{x}, \overline{y} \notin V \wedge |\overline{x}| = |\overline{y}| \Rightarrow \{k \to \overline{x}\}t \approx^V \{k \to \overline{y}\}t'$

Lemma 28.
HCE_BIND $\quad (\Gamma, x \mapsto t) \approx^V (\Gamma', x \mapsto t') \Rightarrow \Gamma \approx^V \Gamma' \wedge t \approx^V t'$

Lemma 29.
IND_DOM $\quad \texttt{ok}\ \Gamma \wedge x \in \texttt{Ind}(\Gamma) \Rightarrow \texttt{dom}(\Gamma \ominus x) = \texttt{dom}(\Gamma) - \{x\} \wedge \texttt{Ind}(\Gamma \ominus x) = \texttt{Ind}(\Gamma) - \{x\}$

Lemma 30.
IND_OK $\quad \texttt{ok}\ \Gamma \wedge x \in \texttt{Ind}(\Gamma) \Rightarrow \texttt{ok}\ (\Gamma \ominus x)$

Lemma 31.
IND_SUBS $\quad x \notin \texttt{dom}(\Gamma) \Rightarrow (\Gamma, x \mapsto \texttt{fvar}\ y) \ominus x = \Gamma[y/x]$

Lemma 32.
HCE_SUB $\quad \Gamma \approx^V \Gamma' \wedge V' \subseteq V \Rightarrow \Gamma \approx^{V'} \Gamma'$
HCE_ADD $\quad \Gamma \approx^V \Gamma' \wedge \overline{x} \notin \texttt{names}(\Gamma) \cup \texttt{names}(\Gamma') \Rightarrow \Gamma \approx^{V \cup \overline{x}} \Gamma'$

Lemma 33.
HCE_DEL_IND $\quad \Gamma \approx^V \Gamma' \wedge \texttt{dom}(\Gamma) \subseteq V \wedge x \in \texttt{Ind}(\Gamma) \Rightarrow \Gamma \ominus x \approx^V \Gamma' \ominus x$
HE_DEL_IND $\quad \Gamma \approx \Gamma' \wedge x \in \texttt{Ind}(\Gamma) \Rightarrow \Gamma \ominus x \approx \Gamma' \ominus x$

Lemma 34.
IR_FVAR $\quad (\Gamma : \texttt{fvar}\ x) \gtrsim_I (\Gamma' : \texttt{fvar}\ x')$
$\Rightarrow (x \notin \texttt{dom}(\Gamma)\ \wedge\ x' \notin \texttt{dom}(\Gamma'))$
$\vee\, (x \in \texttt{dom}(\Gamma)\ \wedge\ x' \in \texttt{dom}(\Gamma') \wedge$
$\exists\, x_0, \ldots, x_n \in Id\,.\, x_0 = x \wedge x_n = x'\ \wedge\ \forall i : 0 \le i < n\,.\, x_i \mapsto \texttt{fvar}\ x_{i+1} \in \Gamma)$

Lemma 35.
IR_IRHT $\quad (\Gamma, x \mapsto t) \gtrsim_I (\Gamma', x \mapsto t') \Rightarrow ((\Gamma, x \mapsto t) : t) \gtrsim_I ((\Gamma', x \mapsto t') : t')$.

Lemma 36.
IR_FVAR_IRHT $\quad ((\Gamma, x \mapsto t) : \texttt{fvar}\ x) \gtrsim_I ((\Gamma', x' \mapsto t') : \texttt{fvar}\ x')$
$\Rightarrow ((\Gamma, x \mapsto t) : t) \gtrsim_I ((\Gamma', x' \mapsto t') : t') \vee ((\Gamma, x \mapsto t) : t) \gtrsim_I ((\Gamma', x' \mapsto t') : \texttt{fvar}\ x')$.

Lemma 37.
IRHT_APP $\quad (\Gamma : \texttt{app}\ t\ (\texttt{fvar}\ x)) \gtrsim_I (\Gamma' : \texttt{app}\ t'\ (\texttt{fvar}\ x'))$
$\Rightarrow (\Gamma : t) \gtrsim_I (\Gamma' : t') \wedge (\Gamma : \texttt{fvar}\ x) \gtrsim_I (\Gamma' : \texttt{fvar}\ x')$.

Lemma 38.
IRHT_FV $(\Gamma : \texttt{fvar}\ x) \succsim_I (\Gamma' : \texttt{fvar}\ x') \wedge \Gamma \subseteq \Delta \wedge \Gamma' \subseteq \Delta' \wedge \Delta \succsim_I \Delta' \wedge$
$(x \notin \texttt{dom}(\Gamma) \Rightarrow x \notin \texttt{dom}(\Delta)) \wedge (x' \notin \texttt{dom}(\Gamma') \Rightarrow x' \notin \texttt{dom}(\Delta')) \wedge$
$(y \in \texttt{dom}(\Gamma) \wedge y \notin \texttt{dom}(\Gamma') \Rightarrow y \notin \texttt{dom}(\Delta'))$
$\Rightarrow (\Delta : \texttt{fvar}\ x) \succsim_I (\Delta' : \texttt{fvar}\ x')$.

Lemma 39.
IRHT_RED_IND $\texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge y \neq x \wedge (\Gamma : \texttt{abs}\ t) \succsim_I (\Gamma' : \texttt{abs}\ t') \wedge (\Gamma : \texttt{fvar}\ x) \succsim_I (\Gamma' : \texttt{fvar}\ x')$
$\Rightarrow ((\Gamma, y \mapsto \texttt{fvar}\ x) : t^y) \succsim_I (\Gamma' : t'^{x'})$.

Lemma 40.
INTR_VARS $\texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \wedge \texttt{fresh}\ \overline{x}\ \texttt{in}\ (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \wedge$
$(\Gamma : \texttt{let}\ \overline{t}\ \texttt{in}\ t) \succsim_I (\Gamma' : \texttt{let}\ \overline{t'}\ \texttt{in}\ t') \Rightarrow ((\Gamma, \overline{x} \mapsto \overline{t}^{\overline{x}}) : t^{\overline{x}}) \succsim_I ((\Gamma', \overline{x} \mapsto \overline{t'}^{\overline{x}}) : t'^{\overline{x}})$.

Lemma 41.
HCE_SUBS $\Gamma \approx^V \Gamma' \wedge y \notin V \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma' \Rightarrow \Gamma[y/x] \approx^{V[y/x]} \Gamma'[y/x]$
IR_SUBS $\Gamma \succsim_I \Gamma' \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma \wedge \texttt{fresh}\ y\ \texttt{in}\ \Gamma' \Rightarrow \Gamma[y/x] \succsim_I \Gamma'[y/x]$
IR_HT_SUBS $(\Gamma : t) \succsim_I (\Gamma' : t') \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma' : t')$
$\Rightarrow (\Gamma[y/x] : t[y/x]) \succsim_I (\Gamma'[y/x] : t'[y/x])$

Lemma 42.
IRHT_REF $\texttt{ok}\ \Gamma \wedge \texttt{lc}\ t \Rightarrow (\Gamma : t) \succsim_I (\Gamma : t)$

Corollary 1.
IR_DOM_DOM $\Gamma \succsim_I \Gamma' \Rightarrow \Gamma \ominus (\texttt{dom}(\Gamma) - \texttt{dom}(\Gamma')) \approx \Gamma'$

Corollary 2.
RENAMING3 $\Gamma : t \Downarrow^K \Delta : w \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Gamma : t) \wedge \texttt{fresh}\ y\ \texttt{in}\ (\Delta : w) \wedge x \notin \texttt{names}(\Gamma : t)$
$\Rightarrow \Gamma : t \Downarrow^K \Delta[y/x] : w[y/x]$

Corollary 3.
CE_LC $t \approx^V t' \wedge \texttt{lc}\ t \Rightarrow \texttt{lc}\ t'$