Manipulating Trees with Hidden Labels

Luca Cardelli - Microsoft Research
Philippa Gardner - Imperial College London
Giorgio Ghelli - Università di Pisa

Abstract. We define an operational semantics and a type system for manipulating semistructured data that contains hidden information. The data model is simple labeled trees with a hiding operator. Data manipulation is based on pattern matching, with types that track the use of hidden labels.

1 Introduction

1.1 Languages for Semistructured Data

XML and semistructured data [1] are inspiring a new generation of programming and query languages based on more flexible type systems [26, 5, 6, 15]. Traditional type systems are grounded on mathematical constructions such as cartesian products, disjoint unions, function spaces, and recursive types. The type systems for semistructured data, in contrast, resemble grammars or logics, with untagged unions, associative products, and Kleene star operators. The theory of formal languages, for strings and trees, provides a wealth of ready results, but it does not account, in particular, for functions. Some integration of the two approaches to type systems is necessary [26, 5].

While investigating semistructured data models and associated languages, we became aware of the need for manipulating private data elements, such as XML identifiers, unique node identifiers in graph models [7], and even heap locations. Such private resources can be modeled using *names* and *name hiding* notions arising from the π-calculus [27]: during data manipulation, the identity of a private name is not important as long as the distinctions between it and other (public or private) names are preserved. Recent progress has been made in handling private resources in programming. FreshML [21] pioneers the *transposition* [30], or swapping, of names, within a type systems that prevents the disclosure of private names.

Other recent techniques can be useful for our purposes. The spatial logics of concurrency devised to cope with π-calculus restriction and scope extrusion [27], and the separation logics used to describe data structures [28,29], provide novel logical operators that can be used also in type systems. Moreover, the notion of dependent types, when the dependence is restricted to names, is tractable [25].

In this paper we bring together a few current threads of development: the effort to devise new languages, type systems, and logics for data structures, the logical operators that come from spatial and nominal logics for private resources, the techniques of transpositions, and the necessity to handle name-dependent types when manipulating private resources. We study these issues in the context of a simplified data model: simple labeled trees with hidden labels, and programs that manipulate such trees. The edges of such trees are labeled with *names*. Our basic techniques can be applied to related data models, such as graphs with hidden node and edge labels, which will be the subject of further work.

1.2 Data Model

The data model we investigate here has the following constructors. Essentially, we extend a simple tree model (such as XML) in a general and orthogonal way with a hiding operator.

0	the tree consisting of a single root node;
$n[P]$	the tree with a single edge from the root, labeled n, leading to P;
$P \mid Q$	the root-merge of two trees (commutative and associative);
$(\nu n)P$	the tree P where the label n is hidden/private/restricted. As in π-calculus, we call *restriction* the act of hiding a name.

Trees are inspected by pattern matching. For example, program (1) below inspects a tree t having shape $n[P] \mid Q$, for some P,Q, and produces $P \mid m[Q]$. Here n,m are constant (public) labels, x,y are pattern variables, and $\mathbf{T}$ is both the pattern that matches any tree and the type of all trees. It is easy to imagine that, when parameterized in t, this program should have the type indicated.

$$\textbf{\textit{match}}\ t\ \textbf{\textit{as}}\ (n[x{:}\mathbf{T}] \mid y{:}\mathbf{T})\ \textbf{\textit{then}}\ (x \mid m[y]) \qquad (1)$$
transforms a tree $t = n[P] \mid Q$ into $P \mid m[Q]$
expected typing: $(n[\mathbf{T}] \mid \mathbf{T}) \to (\mathbf{T} \mid m[\mathbf{T}])$

Using the same pattern match as in (1), let us now remove the public label n and insert a private one, p, that is created and bound to the program variable z at "run-time":

$$\textbf{\textit{match}}\ t\ \textbf{\textit{as}}\ (n[x{:}\mathbf{T}] \mid y{:}\mathbf{T})\ \textbf{\textit{then}}\ (\nu z)\ (x \mid z[y]) \qquad (2)$$
transforms $t = n[P] \mid Q$ into $(\nu p)\ (P \mid p[Q])$ for a fresh label p
expected typing: $(n[\mathbf{T}] \mid \mathbf{T}) \to (\mathsf{H}z.\ (\mathbf{T} \mid z[\mathbf{T}]))$

In our type system, the *hidden name quantifier* H is the type construct corresponding to the data construct ν [10]. More precisely, $\mathsf{H}z.\mathcal{A}$ means that there is a hidden label p denoted by the variable z, such that the data is described by $\mathcal{A}\{z{\leftarrow}p\}$. (Scope extrusion [27] makes the relationship non trivial, see Sections 2 and 4.) Because of the $\mathsf{H}z.\mathcal{A}$ construct, types contain name variables; that is, types are dependent on names.

The first two examples pattern match on the public name n. Suppose instead that we want to find and manipulate private names. The following example is similar to (2), except that now a private label p from the data is matched and bound to the variable z.

$$\textbf{\textit{match}}\ t\ \textbf{\textit{as}}\ ((\nu z)\ (z[x{:}\mathbf{T}] \mid y{:}\mathbf{T}))\ \textbf{\textit{then}}\ x \mid z[y] \qquad (3)$$
transforms $t = (\nu p)(p[P] \mid Q)$ into $(\nu p)(P \mid p[Q])$
expected typing: $(\mathsf{H}z.\ (z[\mathbf{T}] \mid \mathbf{T})) \to (\mathsf{H}z.\ (\mathbf{T} \mid z[\mathbf{T}]))$

Note that the restriction (νp) in the result is not apparent in the program: it is implicitly applied by a match that opens a restriction, so that the restricted name does not escape.

As the fourth and remaining case, we convert a private name in the data into a public one. The only change from (3) is a public name m instead of z in the result:

$$\textbf{\textit{match}}\ t\ \textbf{\textit{as}}\ ((\nu z)\ (z[x{:}\mathbf{T}] \mid y{:}\mathbf{T}))\ \textbf{\textit{then}}\ x \mid m[y] \qquad (4)$$
transforms $t = (\nu p)\ (p[P] \mid Q)$ into $(\nu p)(P \mid m[Q])$
expected typing: $(\mathsf{H}z.\ (z[\mathbf{T}] \mid \mathbf{T})) \to (\mathsf{H}z.\ (\mathbf{T} \mid m[\mathbf{T}]))$

This program replaces only one occurrence of p: the residual restriction (νp) guarantees that any other occurrences inside P,Q remain bound. As a consequence, the binder $\mathsf{H}z$ has to remain in the result type. Note that, although we can replace a private name with a public one, we cannot "expose" a private name, because of the rebinding of the output.

As an example of an incorrectly typed program consider the following attempt to assign a simpler type to the result of example (4), via a typed ***let*** biding:

$$\textbf{\textit{let}}\ w : (\mathbf{T} \mid m[\mathbf{T}]) = \textbf{\textit{match}}\ t\ \textbf{\textit{as}}\ ((\nu z)\ (z[x{:}\mathbf{T}] \mid y{:}\mathbf{T}))\ \textbf{\textit{then}}\ x \mid m[y]$$

Here we would have to check that $\mathsf{H}z.\ (\mathbf{T} \mid m[\mathbf{T}])$ is compatible with $(\mathbf{T} \mid m[\mathbf{T}])$. This

would work if we could first show that $\mathsf{H}z.\ (\mathbf{T} \mid m[\mathbf{T}])$ is a subtype of $(\mathsf{H}z.\ \mathbf{T}) \mid (\mathsf{H}z.\ m[\mathbf{T}])$, and then simplify. But such a subtyping does not hold since, e.g., $(\nu p)(p[0] \mid m[p[0]])$ matches the former type but not the latter, because the restriction (νp) cannot be distributed.

1.3 Transpositions

So far, we have illustrated the manipulation of individual private or public names by pattern matching and data constructors. However, we may want to replace throughout a whole data structure a public name with another, or a public one with a private one, or vice versa. We could do this by recursive analysis, but it would be very difficult to reflect what has happened in the type structure, likely resulting in programs of type $\mathbf{T}\rightarrow\mathbf{T}$. So, we introduce a *transposition* facility as a primitive operation, and as a corresponding type operator. In the simplest case, if we want to transpose (exchange) a name n with a name m in a data structure t we write $t(n\leftrightarrow m)$. If t has type $\mathcal{A}$, then $t(n\leftrightarrow m)$ has type $\mathcal{A}(n\leftrightarrow m)$. We define rules to manipulate type level transpositions; for example we derive that, as types, $n[\mathbf{0}](n\leftrightarrow m) = m[\mathbf{0}]$.

Transposition types are interesting when exchanging public and private labels. Consider the following program and its initial syntax-driven type:

$$\lambda x{:}n[\mathbf{T}].\ (\nu z)\ x(m\leftrightarrow z) \quad :\quad n[\mathbf{T}]\rightarrow\mathsf{H}z.n[\mathbf{T}](m\leftrightarrow z) \quad (=\ n[\mathbf{T}]\rightarrow n[\mathbf{T}]\ ) \qquad (5)$$

This program takes data of the form $n[P]$, creates a fresh label p denoted by z, and swaps the public m with the fresh p in $n[P]$, to yield $(\nu p)n[P](m\leftrightarrow p)$, where the fresh p has been hidden in the result. Since n is a constant different from m, and p is fresh, the result is in fact $(\nu p)n[P(m\leftrightarrow p)]$. The result type can be similarly simplified to $\mathsf{H}z.n[\mathbf{T}(m\leftrightarrow z)]$. Now, swapping two names in the set of all trees, $\mathbf{T}$, produces again the set of all trees. Therefore, the result type can be further simplified to $\mathsf{H}z.n[\mathbf{T}]$. We then have that $\mathsf{H}z.n[\mathbf{T}] = n[\mathbf{T}]$, since a restriction can be pushed through a public label, where it is absorbed by $\mathbf{T}$. Therefore, the type of our program is $n[\mathbf{T}]\rightarrow n[\mathbf{T}]$.

Since we already need to handle name-dependent types, we can introduce, without much additional complexity, a dependent function type $\Pi w.\ \mathcal{A}$. This is the type of functions $\lambda w{:}\mathbf{N}.t$ that take a name m (of type $\mathbf{N}$) as input, and return a result of type $\mathcal{A}\{w\leftarrow m\}$. We can then write a more parametric version of example (5), where the constant n is replaced by a name variable w which is a parameter:

$$\lambda w{:}\mathbf{N}.\ \lambda x{:}w[\mathbf{T}].\ (\nu z)\ x(m\leftrightarrow z) \quad :\quad \Pi w.\ (w[\mathbf{T}] \rightarrow \mathsf{H}z.\ w[\mathbf{T}](m\leftrightarrow z)) \qquad (6)$$

Now, the type $\mathsf{H}z.\ w[\mathbf{T}](m\leftrightarrow z)$ simplifies to $\mathsf{H}z.\ w(m\leftrightarrow z)[\mathbf{T}]$, but no further, since m can in fact be given for w, in which case it would be transposed to the private z.

Transpositions are emerging as a unifying and simplifying principle in the formal manipulation of binding operators [30], which is a main goal of this paper. If some type-level manipulation of names is of use, then transpositions seem a good starting point.

1.4 General Structure

The type system of our calculus has, at the basis, tree types. Function types are built on top of the tree types in standard higher-order style. The tree types, however, are unusual: they are the formulas of a spatial logic. Therefore, we can write types such as:

$$\mathbf{T} \rightarrow \neg\mathbf{0}$$
$$((\mathcal{A} \wedge \neg\mathbf{0}) \mid n[\mathcal{B}]) \rightarrow (n[\mathcal{A}] \mid \mathcal{B})$$

Note that logical operators can be applied only to tree types, not to higher-order types.

A subtyping relation is defined between types. On tree types, subtyping is defined as

a sound approximation to logical implication; that is, $\mathcal{A} <: \mathcal{B}$ implies $\mathcal{A} \Rightarrow \mathcal{B}$. Subtyping is then extended to function types by the usual contravariant rule. This means that a logical implication check needs to be used during static typechecking, whenever a subtyping check is required.

Tree data is manipulated via pattern matching constructs that perform "run-time type checks". Since tree types are formulas, we have the full power of the logic to express the pattern matching conditions. Those run-time type checks are executed as run-time satisfaction checks ($\models$). For example, one of our matching construct is a test to see whether the value denoted by expression t has type $\mathcal{A}$:

$t?(x{:}\mathcal{A}).u,v$

This construct first computes the tree P denoted by the expression t, and then performs a test $P \models \mathcal{A}$. If the test is successful, it binds P to x and executes u; otherwise it binds P to x and executes v. The variable x can be used both inside u and v, but in u it has type $\mathcal{A}$, while in v it has type $\neg\mathcal{A}$.

To summarize, our formulas are used as a very expressive type system for tree data, within a typed λ-calculus. A satisfaction algorithm is used to analyze data at run time, and an entailment algorithm is needed during static typechecking. In absence of polymorphism, type are ground, which facilitates matters. With name-dependent types (necessary to handle a restriction construct), types can contain variables of sort Name, but this can be handled without much difficulty. At run-time, all values and types are ground. As usual, the type system checks whether an *open* term has a type: it can do so without additional difficulties, even though the basic satisfaction test we have is for *closed* terms (i.e., values).

1.5 Related and Future Work

It should be clear from Section 1.2 that sophisticated type-level manipulations are required for our data model, involving transposition types (which seem to be unique to our work), hiding quantifiers, and dependent types. Furthermore, we work in the context of a data model and type system that is "non-structural", both in the sense of supporting grammar-like types (with $\wedge \vee \neg$) and in the sense of supporting π-calculus-style extruding scopes. In both these aspects we differ from FreshML [31], although we base much of our development on the same foundations [30]. Our technique of automatically rebinding restrictions sidesteps some complex issues in the FreshML type system, and yet seems to be practical for many examples. FreshML uses "apartness types" $\mathcal{A}\#w$, which can be used to say that a function takes a name denoted by w and a piece of data $\mathcal{A}$ that does not contain that name. We can express that idiom differently as $\Pi w.\ (\mathcal{A} \wedge \neg \copyright w) \rightarrow \mathcal{B}$, where the operator $\copyright w$ [10] means "contains free the name denoted by w".

Our calculus is based on a pattern matching construct that performs run-time type tests; in this respect, it is similar to the XML manipulation languages XDuce [26] and CDuce [5]. However, those languages do not deal with hidden names, whose study is our main goal. XDuce types are based on tree grammars: they are more restrictive than ours but are based on well-known algorithms. CDuce types are in some aspects richer than ours: they mix the logical and functional levels that we keep separate; such mixing would not easily extend to our $\mathsf{H}x.\mathcal{A}$ types. Other differences stem from the data model (our $P \mid Q$ is commutative), and from auxiliary programming constructs.

The database community has defined many languages to query semistructured data [1,3,6,8,15,17,19,20], but they do not deal with hidden names. The theme of hidden identifiers (OIDs) has been central in the field of object-oriented database languages [2, 4]. However, the debate there was between languages where OIDs are hidden to the user, and lower-level languages where OIDs are fully visible. The second approach is more ex-

pressive but has the severe problem that OIDs lose their meaning once they are exported outside their natural scope. We are not aware of any proposal with operators to define a scope for, reveal, and rehide private identifiers, as we do in our calculus.

In TQL [15], the semistructured query language closest to this work, a programmer writes a logical formula, and the system chooses a way to retrieve all pieces of data that satisfy that formula. In our calculus, such formulas are our tree types, but the programmer has to write the recursion patterns that collect the result (as in Section 8). The TQL approach is best suited to collecting the subtrees that satisfy a condition, but the approach we explore here is much more expressive; for example, we can apply transformations at an arbitrary depth, which is not possible in TQL. Other query-oriented languages, such as XQuery [6], support structural recursion as well, for expressiveness.

As a major area of needed future work, our subtyping relation is not prescribed in detail here (apart for the non-trivial subtypings coming from transposition equivalence). Our type system is parameterized by an unspecified set of ValidEntailments, which are simply assumed to be sound for typing purposes. The study of related subtyping relations (a.k.a. valid logical implications in spatial logics [11]) is in rapid development. The work in [12] provides a complete subtyping algorithm for ground types (i.e. not including $\mathsf{H}z.\mathcal{A}$), and other algorithms are being developed that include Kleene star [18]. Such theories and algorithms could be taken as the core of our ValidEntailments. But adding quantifiers is likely to lead to either undecidability or incompleteness. In the middle ground, there is a collection of sound and practical inclusion rules [10,11] that can be usefully added to the ground subtyping relation (e.g., $\mathsf{H}z.n[\mathcal{A}] <: n[\mathsf{H}z.\mathcal{A}]$ for example (5)). By parameterizing over the ValidEntailments, we show that these issues are relatively orthogonal to the handling of transpositions and hiding.

A precursor of this work handles a simpler data model, with no hiding but with a similarly rich type system based on spatial logic [12]. However, even the richer data model considered in this paper is not all one could wish for. For example, hiding makes better sense for graph nodes [14], or for addresses in heaps [29], than for tree labels. More sophisticated data models include graphs, and combinations of trees and graphical links as in practical uses of XML (see example in Section 8). In any case, the manipulation of hidden resources in data structures is fundamental.

2 Tree Values

Our programs manipulate *values*; either *name values* (from a countable set of names Λ), *tree values*, or *function values* (i.e., closures). Over the tree values, we define a structural congruence relation $\equiv$ that factors out the equivalence laws for $|$ and 0, and the scoping laws for restriction. Function values are triples of a term t (Section 3.1) with respect to an input variable x (essentially, $\lambda x.t$) and a stack for free variables ρ. A stack ρ is a list of bindings of variables to values. Name transpositions are defined on all values.

2-1 Definition: Tree Values

Λ	Names: a countable set of names *n, m, p, ...*		
$P,Q,R ::=$	Tree values	All names: $na(P)$	Free names: $fn(P)$
0	void	$na(0) \triangleq \{\}$	$fn(0) \triangleq \{\}$
$P \mid Q$	composition	$na(P \mid Q) \triangleq na(P) \cup na(Q)$	$fn(P \mid Q) \triangleq fn(P) \cup fn(Q)$
$n[P]$	location	$na(n[P]) \triangleq \{n\} \cup na(P)$	$fn(n[P]) \triangleq \{n\} \cup fn(P)$
$(\nu n)P$	restriction	$na((\nu n)P) \triangleq \{n\} \cup na(P)$	$fn((\nu n)P) \triangleq fn(P) - \{n\}$

We define an *actual transposition* operation on tree values, $P\bullet(m \leftrightarrow m')$, that blindly

swaps free and bound names m,m' within P. The interaction of transpositions with binders such as $(\nu n)P$ supports a general formal treatment of bound names [30].

2-2 Definition: Actual Transposition of Names and Tree Values.

$n\bullet(n\leftrightarrow m) = m$ $\qquad$ $0\bullet(m\leftrightarrow m') = 0$

$n\bullet(m\leftrightarrow n) = m$ $\qquad$ $(P \mid Q)\bullet(m\leftrightarrow m') = P\bullet(m\leftrightarrow m') \mid Q\bullet(m\leftrightarrow m')$

$n\bullet(m\leftrightarrow m') = n$ if $n\neq m$ and $n\neq m'$ $\qquad$ $n[P]\bullet(m\leftrightarrow m') = n\bullet(m\leftrightarrow m')[P\bullet(m\leftrightarrow m')]$

$((\nu n)P)\bullet(m\leftrightarrow m') = (\nu n\bullet(m\leftrightarrow m'))P\bullet(m\leftrightarrow m')$

2-3 Lemma: Distribution of Transpositions.

(1) $p\bullet(n\leftrightarrow n')\bullet(m\leftrightarrow m') = p\bullet(m\leftrightarrow m')\bullet(n\bullet(m\leftrightarrow m')\leftrightarrow n'\bullet(m\leftrightarrow m'))$

$p\bullet(n\leftrightarrow n')\bullet(m\leftrightarrow m') = p\bullet(m\bullet(n\leftrightarrow n')\leftrightarrow m'\bullet(n\leftrightarrow n'))\bullet(n\leftrightarrow n')$

(2) $P\bullet(n\leftrightarrow n')\bullet(m\leftrightarrow m') = P\bullet(m\leftrightarrow m')\bullet(n\bullet(m\leftrightarrow m')\leftrightarrow n'\bullet(m\leftrightarrow m'))$

$P\bullet(n\leftrightarrow n')\bullet(m\leftrightarrow m') = P\bullet(m\bullet(n\leftrightarrow n')\leftrightarrow m'\bullet(n\leftrightarrow n'))\bullet(n\leftrightarrow n')$

Transpositions are used in the definition of α-congruence and capture-avoiding substitution. Structural congruence is analogous to the standard definition for π-calculus [27]; the "scope extrusion" rule for ν over -|- is written in an equivalent equational style.

2-4 Definition: Congruence on Values.

A congruence relation on values is a binary relation c satisfying:

$P \mathsf{c} P$

$P \mathsf{c} Q \Rightarrow Q \mathsf{c} P$

$P \mathsf{c} Q \wedge Q \mathsf{c} R \Rightarrow P \mathsf{c} R$

$P \mathsf{c} P' \wedge Q \mathsf{c} Q' \Rightarrow (P \mid Q) \mathsf{c} (P' \mid Q')$

$P \mathsf{c} P' \Rightarrow n[P] \mathsf{c} n[P']$

$P \mathsf{c} P' \Rightarrow ((\nu n)P) \mathsf{c} ((\nu n)P')$

2-5 Definition: α-Congruence and Structural Congruence on Tree Values.

α-congruence, $\equiv_\alpha$, is the least congruence relation on tree values such that:

$(\nu n)P \equiv_\alpha (\nu m)(P\bullet(n\leftrightarrow m))$ $\qquad$ where $m\notin na(P)$

Structural congruence, $\equiv$, is the least congruence relation on tree values such that:

$P \equiv_\alpha Q \Rightarrow P \equiv Q$ $\qquad$ $(\nu n)0 \equiv 0$

$P \mid Q \equiv Q \mid P$ $\qquad$ $(\nu n)m[P] \equiv m[(\nu n)P]$ if $n\neq m$

$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$ $\qquad$ $(\nu n)(P \mid (\nu n)Q) \equiv ((\nu n)P) \mid ((\nu n)Q)$

$P \mid 0 \equiv P$ $\qquad$ $(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$

N.B.: This notion of α-congruence can be shown equivalent to the standard one.

2-6 Lemma: Structural Congruence Properties.

If $P \equiv Q$ then $fn(P)=fn(Q)$

If $P \equiv Q$ then $P\bullet(m\leftrightarrow m') \equiv Q\bullet(m\leftrightarrow m')$.

2-7 Definition: Free Name Substitution on Tree Values.

$0\{n\leftarrow m\} = 0$

$(P \mid Q)\{n\leftarrow m\} = P\{n\leftarrow m\} \mid Q\{n\leftarrow m\}$

$p[P]\{n\leftarrow m\} = p\{n\leftarrow m\}[P\{n\leftarrow m\}]$

$((\nu p)P)\{n\leftarrow m\} = (\nu q)((P\bullet(p\leftrightarrow q))\{n\leftarrow m\})$ $\qquad$ for $q\notin na((\nu p)P)\cup\{n,m\}$

N.B.: different choices of q in the last clause, lead to α-congruent results.

3 Terms and High Values

3.1 Syntax

Our λ-calculus is stratified in terms of *low types* and *high types*. The low types are the tree types and the type of names, **N**. (Basic data types such as integers could be added to low types.) The novel aspects of the type structure are the richness of the tree types, which come from the formulas of spatial logics [16, 10], and the presence of transposition types. We then have higher types over the low types: function types and name-dependent types. The precise meaning of types is given in Section 4.

The same stratification holds on terms, which can be of low or high type, as is more apparent in the operational semantics of Section 5 and in the type rules of Section 7.

3-1 Definition: Syntax

Var	Variables: a countable set of variables x, y, z, ...
$\mathcal{N},\mathcal{M}$ ::=	Name Expressions
x	name variable
n	name constant
$\mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')$	name transposition
$\mathcal{A},\mathcal{B}$ ::=	Tree Types
$\mathbf{0}$	void
$\mathcal{N}[\mathcal{A}]$	location
$\mathcal{A} \mid \mathcal{B}$	composition
$\mathsf{H}x.\mathcal{A}$	hiding quantifier (x:**N**)
$©\mathcal{N}$	occurence
$\mathbf{F}$	false
$\mathcal{A} \wedge \mathcal{B}$	conjunction
$\mathcal{A} \Rightarrow \mathcal{B}$	implication
$\mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}')$	transposition
$\mathcal{F},\mathcal{G},\mathcal{H}$ ::=	High Types
$\mathcal{A}$	tree types
$\mathbf{N}$	name type
$\mathcal{F}\rightarrow\mathcal{G}$	function types ($\mathcal{F}\neq\mathbf{N}$)
$\Pi x.\mathcal{G}$	name dependent function types (x:**N**)
t,u,v ::=	Terms
0	void
$\mathcal{N}[u]$	location
$t \mid u$	composition
$(\nu\underline{x})t$	restriction
$t(\mathcal{M}\leftrightarrow\mathcal{M}')$	term transposition
$t\div(\mathcal{N}[\underline{y}{:}\mathcal{A}]).u$	location match (binding y only)
$t\div(\underline{x}{:}\mathcal{A} \mid \underline{y}{:}\mathcal{B}).u$	composition match (binding x,y)
$t\div((\nu\underline{x})\underline{y}{:}\mathcal{A}).u$	restriction match (binding x,y) (auto-rebind u)
$t?(\underline{x}{:}\mathcal{A}).u,v$	tree type test (binding x)
x	high variable
$\mathcal{N}$	name expression (incl. name variables)

$\lambda\underline{x}{:}\mathcal{T}.t$	function
$t(u)$	application

Underlined variables indicate binding occurrences. The scoping rules should be clear, except that: in location match $\underline{y}$ scopes u; in composition match $\underline{x}$ and $\underline{y}$ scope u; in restriction match $\underline{x}$ scopes $\mathcal{A}$ and u, and $\underline{y}$ scopes u; in tree type test $\underline{x}$ scopes u and v. We define name sets, such as $na(\mathcal{A})$, and actual transpositions on all syntax, such as $t\bullet(n\leftrightarrow m)$, in the obvious way (there are no name binders in the syntax). We also define free-variable sets $fv(\text{-})$ on all syntax (based on the mentioned binding occurrences), and capture-avoiding substitutions of name expressions for variables.

3-2 Definition: Actual Transposition on All Syntax.
$\mathcal{N}\bullet(n\leftrightarrow m)$, $\mathcal{A}\bullet(n\leftrightarrow m)$, $\mathcal{T}\bullet(n\leftrightarrow m)$, and $t\bullet(n\leftrightarrow m)$ transpose the names n and m in $\mathcal{N}$, $\mathcal{A}$, $\mathcal{T}$, and t. N.B.: there are no name binders in the syntax.

3-3 Definition: Variables Substitution on Name and Type Expressions.
$\mathcal{N}\{x\leftarrow\mathcal{M}\}$, $\mathcal{A}\{x\leftarrow\mathcal{M}\}$, and $\mathcal{T}\{x\leftarrow\mathcal{M}\}$, are the capture-avoiding substitutions of $\mathcal{M}$ for x in $\mathcal{N}$, $\mathcal{A}$, and $\mathcal{T}$.

Name expressions, tree types, and terms all include *(formal) transposition* operations that are part of the syntax; they represent (actual) transpositions on data, indicated by the $\bullet$ symbol.

The tree types are formulas in a spatial logic, so we can derive the standard types (formulas) for negation $\neg\mathcal{A} \triangleq \mathcal{A}\Rightarrow\mathbf{F}$ and disjunction $\mathcal{A}\vee\mathcal{B} \triangleq \neg(\neg\mathcal{A}\wedge\neg\mathcal{B})$.

The terms include a standard λ-calculus fragment, the basic tree constructors, and some matching operators for analyzing tree data. The *tree type test* construct (distinguished by the character '?') performs a run-time check to see whether a tree has a given type: if tree t satisfies type $\mathcal{A}$ then u is run with x of type $\mathcal{A}$ bound to t; otherwise v is run with x of type $\neg\mathcal{A}$ bound to t. In addition, one needs matching constructs (distinguished by the character '÷') to decompose the tree: *composition match* splits a tree in two components, *location match* strips an edge from a tree, and *restriction match* inspects a hidden label in a tree. A *zero match* is redundant because of the tree type test construct. These multiple matching constructs are designed to simplify the operational semantics and the type rules. In practice, one would use a single case statement with patterns over the structure of trees, but this can be encoded.

In the quantifier $\text{H}x.\mathcal{A}$ and in the restriction match construct, the type $\mathcal{A}$ is dependent on variable x (denoting a hidden name). This induces the need for handling dependent types, and motivates the $\Pi x.\mathcal{G}$ dependent function type constructor. The type dependencies, however, are restricted to name variables, which may be replaced only by name expressions (that is, not by general computations on names). Because of this, these dependent types are relatively easy to handle.

3.2 High Values

High values are the results of evaluating terms. They can be either names, trees, or closures. Closures, in turn, are defined via terms and *stacks*:

3-4 Definition: High Values and Stacks

(1) A stack is a finite map ρ from variables in *Var* to high values F. We write $\emptyset$ for the empty map, and $\rho[x\leftarrow F]$ for the map that is the same as ρ except for mapping x to F.

(2) High values are defined as follows:

F,G,H ::=	High Values	All names: $na(F)$	Free names: $fn(F)$
n	name values	$na(n) \triangleq \{n\}$	$fn(n) \triangleq \{n\}$
P	tree values	$na(P)$: see tree values	$fn(P)$: see tree values
$\langle \rho, x, t \rangle$	function values	$na(\langle \rho, x, t \rangle) \triangleq na(t) \cup na(\rho)$	$fn(\langle \rho, x, t \rangle) \triangleq fn(t) \cup fn(\rho)$
		$na(\rho) \triangleq \bigcup_{x \in dom(\rho)} na(\rho(x))$	$fn(\rho) \triangleq \bigcup_{x \in dom(\rho)} fn(\rho(x))$

(3) Transpositions of function values and stacks are defined as follows:

$\langle \rho, x, t \rangle \bullet (n \leftrightarrow n') \triangleq \langle \rho \bullet (n \leftrightarrow n'), x, t \bullet (n \leftrightarrow n') \rangle$

$\emptyset \bullet (n \leftrightarrow n') \triangleq \emptyset$

$\rho[x \leftarrow F] \bullet (n \leftrightarrow n') \triangleq \rho \bullet (n \leftrightarrow n')[x \leftarrow F \bullet (n \leftrightarrow n')]$ $(x \notin dom(\rho))$

4 Satisfaction

The satisfaction relation, written $\models$, relates values to types, and thus provides the semantic meaning of typing that is enforced by the type system of Section 7. For type constructs such as $\wedge$ and $\Rightarrow$, this is related to the notion of satisfaction from modal logic. Over tree types we have essentially the relation studied in [16, 10], extended to hiding and transpositions. Satisfaction is then generalized to high types, where it depends on the operational semantics $\Downarrow_\rho$ of Section 5, which depends on satisfaction at tree types only.

4-1 Definition: Satisfaction

On Name Expressions: $n \models_N \mathcal{N}$, for $\mathcal{N}$ closed (no free variables), is defined by:

$n \models_N m$	iff $m = n$
$n \models_N \mathcal{N}(\mathcal{M} \leftrightarrow \mathcal{M}')$	iff $\exists m,m'.\ m \models_N \mathcal{M}$ and $m' \models_N \mathcal{M}'$ and $n \bullet (m \leftrightarrow m') \models_N \mathcal{N}$

On Tree Types: $P \models_T \mathcal{A}$, for $\mathcal{A}$ closed, is defined by:

$P \models_T \mathbf{0}$	iff $P \equiv 0$
$P \models_T \mathcal{N}[\mathcal{A}]$	iff $\exists n,P'.\ n \models_N \mathcal{N}$ and $P \equiv n[P']$ and $P' \models_T \mathcal{A}$
$P \models_T \mathcal{A} \mid \mathcal{B}$	iff $\exists P',P''.\ P \equiv P' \mid P''$ and $P' \models_T \mathcal{A}$ and $P'' \models_T \mathcal{B}$
$P \models_T \mathsf{H}x.\mathcal{A}$	iff $\exists n,P'.\ P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $P' \models_T \mathcal{A}\{x \leftarrow n\}$
$P \models_T \copyright\mathcal{N}$	iff $\exists n.\ n \models_N \mathcal{N}$ and $n \in fn(P)$
$P \models_T \mathbf{F}$	never
$P \models_T \mathcal{A} \wedge \mathcal{B}$	iff $P \models_T \mathcal{A}$ and $P \models_T \mathcal{B}$
$P \models_T \mathcal{A} \Rightarrow \mathcal{B}$	iff $P \models_T \mathcal{A}$ implies $P \models_T \mathcal{B}$
$P \models_T \mathcal{A}(\mathcal{M} \leftrightarrow \mathcal{M}')$	iff $\exists m,m'.\ m \models_N \mathcal{M}$ and $m' \models_N \mathcal{M}'$ and $P \bullet (m \leftrightarrow m') \models_T \mathcal{A}$

On High Types: $F \models_H \mathcal{F}$, for $\mathcal{F}$ closed, is defined by:

$F \models_H \mathbf{N}$	iff $F \models_N \mathcal{N}$ for some $\mathcal{N}$
$F \models_H \mathcal{A}$	iff $F \models_T \mathcal{A}$
$H \models_H \mathcal{F} \rightarrow \mathcal{G}$ $(\mathcal{F} \neq \mathbf{N})$	iff $H = \langle \rho, z, t \rangle$ and $\forall F,G.\ (F \models_H \mathcal{F} \wedge\ t \Downarrow_{\rho[z \leftarrow F]} G) \Rightarrow G \models_H \mathcal{G}$
$H \models_H \Pi x.\ \mathcal{G}$	iff $H = \langle \rho, z, t \rangle$ and $\forall n,G.\ t \Downarrow_{\rho[z \leftarrow n]} G\ \Rightarrow\ G \models_H \mathcal{G}\{x \leftarrow n\}$

(We will omit the subscripts on $\models$.) The constructs $\mathsf{H}x.\mathcal{A}$ and $\copyright\mathcal{N}$ are derived operators in [10], and are taken here as primitive, in the original spirit of [9]. In the definition of $\mathsf{H}x.\mathcal{A}$, the clause $P \equiv (\nu n)P'$ pulls a restriction (even a dummy one) from elsewhere in the data, via scope extrusion (Definition 2-5). The type $\mathsf{H}w.\copyright w$ is the type of non-redundant restrictions, with the quantifier $\mathsf{H}w$ revealing a restricted name n, and $\copyright w$ declaring that this n is used in the data. The meaning of formal transpositions relies on actual transpositions. At high types, a closure $\langle \rho, z, t \rangle$ satisfies a function type $\mathcal{F} \rightarrow \mathcal{G}$ if, on any input

satisfying $\mathcal{F}$, every output satisfies $\mathcal{G}$; similarly for $\Pi x.\, \mathcal{G}$.

4-2 Proposition: Tree Satisfaction Under Structural Congruence.
If $P \vDash \mathcal{A}$ and $P \equiv Q$ then $Q \vDash \mathcal{A}$.

4-3 Lemma: Name and Tree Satisfaction Under Actual Transposition.
If $n \vDash \mathcal{N}$ then $n\bullet(m{\leftrightarrow}m') \vDash \mathcal{N}\bullet(m{\leftrightarrow}m')$. If $P \vDash \mathcal{A}$ then $P\bullet(m{\leftrightarrow}m') \vDash \mathcal{A}\bullet(m{\leftrightarrow}m')$.

5 Operational Semantics

We give a *big step* operational semantics that is later used for a subject reduction result (Theorem 7-5). This style of semantics, namely a relation between a program and all its potential final results, is sufficient to clarify the intended behavior of our operations. It could be extended with error handling. Alternatively, a *small step* semantics could be given. In either case, one could go further and establish a type soundness theorem stating that well-typed programs (preserve types and) do not get stuck. All this is relatively routine, and we opt to give only the essential semantics.

The operational semantics is given by a relation $t \Downarrow_\rho F$ between terms t, stacks ρ, and values F, meaning that t can evaluate to F on stack ρ. An auxiliary relation, $\mathcal{N} \downarrow_\rho n$, deals with evaluation of name expressions. The semantics of run-time tests makes use of the satisfaction relation from Section 4. We use, $t \Downarrow_\rho P$ to indicate that t evaluates to a tree value. We use $t \Downarrow_\rho{\equiv} P$ as an abbreviation for $t \Downarrow_\rho Q$ and $Q \equiv P$, for some Q.

5-1 Definition: Operational Semantics

$$\frac{x{\in}dom(\rho) \quad \rho(x){\in}\Lambda}{x \downarrow_\rho \rho(x)}\ (\text{NRed } x) \qquad \frac{}{n \downarrow_\rho n}\ (\text{NRed } n) \qquad \frac{\mathcal{N} \downarrow_\rho n \quad \mathcal{M} \downarrow_\rho m \quad \mathcal{M}' \downarrow_\rho m'}{\mathcal{N}\{\mathcal{M}{\leftrightarrow}\mathcal{M}'\} \downarrow_\rho n\bullet(m{\leftrightarrow}m')}\ (\text{NRed} {\leftrightarrow})$$

$$\frac{}{0 \Downarrow_\rho 0}\ (\text{Red } 0) \qquad \frac{\mathcal{N} \downarrow_\rho n \quad t \Downarrow_\rho P}{\mathcal{N}[t] \Downarrow_\rho n[P]}\ (\text{Red } \mathcal{N}[]) \qquad \frac{t \Downarrow_\rho P \quad u \Downarrow_\rho Q}{t \mid u \Downarrow_\rho P \mid Q}\ (\text{Red} \mid) \qquad \frac{n{\notin}na(t,\,\rho) \quad t \Downarrow_{\rho[x\leftarrow n]} P}{(\nu x)t \Downarrow_\rho (\nu n)P}\ (\text{Red } \nu)$$

$$\frac{t \Downarrow_\rho P \quad \mathcal{M} \downarrow_\rho m \quad \mathcal{M}' \downarrow_\rho m'}{t\{\mathcal{M}{\leftrightarrow}\mathcal{M}'\} \Downarrow_\rho P\bullet(m{\leftrightarrow}m')}\ (\text{Red} {\leftrightarrow}) \qquad \frac{\mathcal{N} \downarrow_\rho n \quad t \Downarrow_\rho{\equiv} n[P] \quad P \vDash \rho(\mathcal{A}) \quad u \Downarrow_{\rho[y\leftarrow P]} F}{t{\div}(\mathcal{N}[y{:}\mathcal{A}]).u \Downarrow_\rho F}\ (\text{Red} \div\mathcal{N}[])$$

$$\frac{t \Downarrow_\rho{\equiv} P' \mid P'' \quad P' \vDash \rho(\mathcal{A}) \quad P'' \vDash \rho(\mathcal{B}) \quad x{\neq}y \quad u \Downarrow_{\rho[x\leftarrow P'][y\leftarrow P'']} F}{t{\div}(x{:}\mathcal{A} \mid y{:}\mathcal{B}).u \Downarrow_\rho F}\ (\text{Red} \div\mid) \qquad \frac{n{\notin}na(t,\mathcal{A},u,\rho) \quad t \Downarrow_\rho{\equiv} (\nu n)P \quad P \vDash \rho[x\leftarrow n](\mathcal{A}) \quad x{\neq}y \quad u \Downarrow_{\rho[x\leftarrow n][y\leftarrow P]} Q}{t{\div}((\nu x)y{:}\mathcal{A}).u \Downarrow_\rho (\nu n)Q}\ (\text{Red} \div\nu)$$

$$\frac{t \Downarrow_\rho P \quad P \vDash \rho(\mathcal{A}) \quad u \Downarrow_{\rho[x\leftarrow P]} F}{t?(x{:}\mathcal{A}).u,v \Downarrow_\rho F}\ (\text{Red } ?{\vDash}) \qquad \frac{t \Downarrow_\rho P \quad P \vDash \neg\rho(\mathcal{A}) \quad v \Downarrow_{\rho[x\leftarrow P]} F}{t?(x{:}\mathcal{A}).u,v \Downarrow_\rho F}\ (\text{Red } ?{\nvDash})$$

$$\frac{x \in dom(\rho)}{x \Downarrow_\rho \rho(x)}\ (\text{Red } x) \qquad \frac{\mathcal{N} \downarrow_\rho n}{\mathcal{N} \Downarrow_\rho n}\ (\text{Red } \mathcal{N}) \qquad \frac{}{\lambda x{:}\mathcal{F}.t \Downarrow_\rho \langle \rho, x, t\rangle}\ (\text{Red } \lambda) \qquad \frac{t \Downarrow_\rho \langle \rho', x, t'\rangle \quad u \Downarrow_\rho G \quad t' \Downarrow_{\rho'[x \leftarrow G]} H}{t(u) \Downarrow_\rho H}\ (\text{Red App})$$

The operations (Red ÷ -) and (Red ? -) can execute run-time type tests on dependent types that are run-time instantiated; e.g., note the role of x in $\lambda x{:}\mathbf{N}.\ t?(y{:}x[\mathbf{0}]).u,v$. In these rules, $\rho(\mathcal{A})$ replaces every free variable $x \in dom(\rho)$ in $\mathcal{A}$ with $\rho(x)$. The rules are applicable only if $\rho(\mathcal{A})$ is a well-formed type (otherwise, the rules are *stuck*): the type rules of Section 7 guarantee this well-formedness condition.

The matching reductions are nondeterministic. Hence we have a nondeterministic big step semantics, which gives no information on the existence of divergent reduction paths.

Reduction is not closed up to $\equiv$ (0 does not reduce to 0|0), nor up to $\equiv_\alpha$ (see (Red ν) and (Red ÷ν), which exclude some of the bound names that can be returned). But this is a matter of choice that has no effect on our results.

The following lemma is crucial in the subject reduction cases for (Red ν) (Theorem 7-5). Only this transposition lemma is needed there, not a harder substitution lemma.

5-2 Lemma: Reduction Under Transposition.
If $\mathcal{N} \downarrow_\rho m$ then $\mathcal{N}\bullet(n \leftrightarrow n') \downarrow_{\rho\bullet(n \leftrightarrow n')} m\bullet(n \leftrightarrow n')$.
If $t \Downarrow_\rho F$ then $t\bullet(n \leftrightarrow n') \Downarrow_{\rho\bullet(n \leftrightarrow n')} F\bullet(n \leftrightarrow n')$.

Proof See Appendix.

6 Transposition Equivalence and Apartness

We define a type equivalence relation on name expressions and tree types, which in particular allows any type transposition to be eliminated or pushed down to the name expressions that appear in the type. The main aim of this section is to establish the soundness of such an equivalence relation, which is inspired by [22,11]. A crucial equivalence rule, (EqN $\leftrightarrow$ Apart), requires the notion of *apartness* of name expressions, meaning that the names that those expressions denote are distinct. (*C.f.* examples (5) and (6) in Introduction.) Apartness of name expressions depends on apartness of variables and names; we keep track of such relationships via a *freshness signature*.

6-1 Definition: Freshness Signature
A *freshness signature* ϕ is an ordered list of names followed by an ordered list of distinct variables annotated with a quantifier Q that is either $\forall$ or H. (For example: $n,m,m,p,\forall x,\mathsf{H}y,\mathsf{H}z,\forall w$.) Notation: $dom(\phi)$ is the set of variables in ϕ; $na(\phi)$ is the set of names in ϕ; $\phi(x)$ is the symbol $\forall$ or H associated to x in ϕ. We write $x <_\phi y$ if $x \neq y$ and x precedes y in ϕ. We write $\phi \supseteq \mathcal{N}$ (ϕ covers $\mathcal{N}$) when $fv(\mathcal{N}) \subseteq dom(\phi)$ and $fn(\mathcal{N}) \subseteq na(\phi)$; similarly for $\phi \supseteq \mathcal{A}$ and $\phi \supseteq \mathcal{F}$. We write $S \# S'$ for disjointness of sets of names and variables.

Next we define three equivalence relations between name expressions, $\sim_N$, tree types, $\sim_T$, and high types, $\sim_H$ (often omitting the subscripts), and an apartness relation on name expressions, #. These relations are all indexed by a freshness signature that is understood to cover the free variables and names occurring in the expressions involved. Whether an equivalence or apartness holds, depends crucially on such freshness signature; consider the following examples:

$n \#_{n,m} m$	by (Apart Names)
$n \#_{\mathsf{H}x,n} x$	by (Apart Name Var)
$x \#_{\forall x,\mathsf{H}y} y$	by (Apart Vars)
$y \#_{\forall x,\mathsf{H}y,n} x,\ y \#_{\forall x,\mathsf{H}y,n} n \ \Rightarrow\ y\{x\leftrightarrow n\} \sim_{\forall x,\mathsf{H}y,n} y$	by (EqN $\leftrightarrow$ Apart)
$y \#_{\forall x,\mathsf{H}y,n} x \ \Rightarrow\ y\{y\leftrightarrow n\} \#_{\forall x,\mathsf{H}y,n} x\{y\leftrightarrow n\}$	by (Apart Congr)
$n \sim_{\forall x,\mathsf{H}y,n} y\{y\leftrightarrow n\},\ y\{y\leftrightarrow n\} \#_{\forall x,\mathsf{H}y,n} x\{y\leftrightarrow n\}$ $\Rightarrow\ n \#_{\forall x,\mathsf{H}y,n} x\{y\leftrightarrow n\}$	by (Apart Equiv)

In the following table, we highlight the most interesting rules with an arrow ➔.

6-2 Definition: Equivalence and Apartness

Name expression equivalence, $\mathcal{N} \sim_{\mathrm{N}\phi} \mathcal{M}$, (abbrev. $\mathcal{N} \sim_\phi \mathcal{M}$) and apartness, $\mathcal{N} \#_\phi \mathcal{M}$, are the least relations on name expressions such that $\phi \supseteq \mathcal{N},\mathcal{M}$, and:

$\mathcal{N} \#_\phi \mathcal{M} \ \Rightarrow\ \mathcal{M} \#_\phi \mathcal{N}$		(Apart Symm)
$n \neq m \ \Rightarrow\ n \#_\phi m$	➔	(Apart Names)
$\phi(x) = \mathsf{H} \ \Rightarrow\ n \#_\phi x$	➔	(Apart Name Var)
$x <_\phi y$ and $\phi(y) = \mathsf{H} \ \Rightarrow\ x \#_\phi y$	➔	(Apart Vars)
$\mathcal{N} \#_\phi \mathcal{M} \wedge \mathcal{P} \sim_\phi \mathcal{P}' \wedge \mathcal{Q} \sim_\phi \mathcal{Q}' \ \Rightarrow\ \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{Q}\} \#_\phi \mathcal{M}\{\mathcal{P}'\leftrightarrow\mathcal{Q}'\}$		(Apart Congr)
$\mathcal{N} \sim_\phi \mathcal{N}'$ and $\mathcal{N}' \#_\phi \mathcal{M}'$ and $\mathcal{M}' \sim_\phi \mathcal{M} \ \Rightarrow\ \mathcal{N} \#_\phi \mathcal{M}$		(Apart Equiv)
$\mathcal{N} \sim_\phi \mathcal{N}$		(EqN Refl)
$\mathcal{N} \sim_\phi \mathcal{M} \ \Rightarrow\ \mathcal{M} \sim_\phi \mathcal{N}$		(EqN Symm)
$\mathcal{N} \sim_\phi \mathcal{M}$ and $\mathcal{M} \sim_\phi \mathcal{P} \ \Rightarrow\ \mathcal{N} \sim_\phi \mathcal{P}$		(EqN Trans)
$\mathcal{N} \sim_\phi \mathcal{N}',\ \mathcal{M} \sim_\phi \mathcal{M}',\ \mathcal{P} \sim_\phi \mathcal{P}' \ \Rightarrow\ \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{P}\} \sim_\phi \mathcal{N}'\{\mathcal{M}'\leftrightarrow\mathcal{P}'\}$		(EqN $\leftrightarrow$ Congr)
$\mathcal{N}\{\mathcal{N}\leftrightarrow\mathcal{M}\} \sim_\phi \mathcal{M}$		(EqN $\leftrightarrow$ App)
$\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{N}\} \sim_\phi \mathcal{N}$		(EqN $\leftrightarrow$ Id)
$\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}\{\mathcal{M}'\leftrightarrow\mathcal{M}\}$		(EqN $\leftrightarrow$ Symm)
$\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}$		(EqN $\leftrightarrow$ Inv)
$\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{P}\leftrightarrow\mathcal{P}'\} \sim_\phi \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\{\mathcal{M}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\leftrightarrow\mathcal{M}'\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\}$		(EqN $\leftrightarrow$ $\leftrightarrow$)
$\mathcal{N} \#_\phi \mathcal{M}$ and $\mathcal{N} \#_\phi \mathcal{M}' \ \Rightarrow\ \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}$	➔	(EqN $\leftrightarrow$ Apart)

Tree type equivalence, $\mathcal{A} \sim_{\mathrm{T}\phi} \mathcal{B}$, (abbrev. $\mathcal{A} \sim_\phi \mathcal{B}$), are the least relations on tree types such that $\phi \supseteq \mathcal{A},\mathcal{B}$, and:

$\mathcal{A} \sim_\phi \mathcal{A}$		(EqT Refl)
$\mathcal{A} \sim_\phi \mathcal{B} \ \Rightarrow\ \mathcal{B} \sim_\phi \mathcal{A}$		(EqT Symm)
$\mathcal{A} \sim_\phi \mathcal{B}$ and $\mathcal{B} \sim_\phi \mathcal{C} \ \Rightarrow\ \mathcal{A} \sim_\phi \mathcal{C}$		(EqT Trans)
$\mathcal{N} \sim_{\mathrm{N}\phi} \mathcal{N}'$ and $\mathcal{A} \sim_\phi \mathcal{A}' \ \Rightarrow\ \mathcal{N}'[\mathcal{A}'] \sim_\phi \mathcal{N}'[\mathcal{A}']$	➔	(EqT $\mathcal{N}$[] Congr)
$\mathcal{A} \sim_\phi \mathcal{A}'$ and $\mathcal{B} \sim_\phi \mathcal{B}' \ \Rightarrow\ \mathcal{A} \mid \mathcal{B} \sim_\phi \mathcal{A}' \mid \mathcal{B}'$		(EqT | Congr)
$\mathcal{A} \sim_\phi \mathcal{A}'$ and $\mathcal{B} \sim_\phi \mathcal{B}' \ \Rightarrow\ \mathcal{A} \wedge \mathcal{B} \sim_\phi \mathcal{A}' \wedge \mathcal{B}'$		(EqT $\wedge$ Congr)
$\mathcal{A} \sim_\phi \mathcal{A}'$ and $\mathcal{B} \sim_\phi \mathcal{B}' \ \Rightarrow\ \mathcal{A} \Rightarrow \mathcal{B} \sim_\phi \mathcal{A}' \Rightarrow \mathcal{B}'$		(EqT $\Rightarrow$ Congr)
$\mathcal{A} \sim_{(\phi,\mathsf{H}x)} \mathcal{A}' \ \Rightarrow\ \mathsf{H}x.\,\mathcal{A} \sim_\phi \mathsf{H}x.\,\mathcal{A}'$	➔	(EqT H Congr)
$\mathcal{N} \sim_{\mathrm{N}\phi} \mathcal{N}' \ \Rightarrow\ ©\mathcal{N} \sim_\phi ©\mathcal{N}'$		(EqT © Congr)

$\mathbf{0}(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathbf{0}$		(EqT $\mathbf{0}$ $\leftrightarrow$)
$(\mathcal{N}[\mathcal{A}])(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')[\mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}')]$		(EqT $\mathcal{N}[]$ $\leftrightarrow$)
$(\mathcal{A} \mid \mathcal{B})(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}') \mid \mathcal{B}(\mathcal{M}\leftrightarrow\mathcal{M}')$		(EqT $\mid$ $\leftrightarrow$)
$\mathbf{F}(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathbf{F}$		(EqT $\mathbf{F}$ $\leftrightarrow$)
$(\mathcal{A} \wedge \mathcal{B})(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}') \wedge \mathcal{B}(\mathcal{M}\leftrightarrow\mathcal{M}')$		(EqT $\wedge$ $\leftrightarrow$)
$(\mathcal{A} \Rightarrow \mathcal{B})(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}') \Rightarrow \mathcal{B}(\mathcal{M}\leftrightarrow\mathcal{M}')$		(EqT $\Rightarrow$ $\leftrightarrow$)
$(\mathsf{H}x.\mathcal{A})(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi$	➔	(EqT H $\leftrightarrow$)
$\mathsf{H}x.(\mathcal{A}\{x\leftarrow x[\mathcal{M}\leftrightarrow\mathcal{M}']\}(\mathcal{M}\leftrightarrow\mathcal{M}'))$ with $x\notin fv(\mathcal{M},\mathcal{M}')$		
$(\copyright\mathcal{N})(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \copyright(\mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}'))$		(EqT $\copyright$ $\leftrightarrow$)
$\mathcal{A}(\mathcal{P}\leftrightarrow\mathcal{P}')(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{A}(\mathcal{M}\leftrightarrow\mathcal{M}')(\mathcal{P}(\mathcal{M}\leftrightarrow\mathcal{M}')\leftrightarrow\mathcal{P}'(\mathcal{M}\leftrightarrow\mathcal{M}'))$		(EqT $\leftrightarrow$ $\leftrightarrow$)
$\mathsf{H}x.\ \mathcal{A} \sim_\phi \mathsf{H}y.\ \mathcal{A}\{x\leftarrow y\}$ with $y\notin fv(\mathcal{A})$	➔	(EqT H-α)

High type equivalence, $\mathcal{F}\sim_{\mathrm{H}\phi}\mathcal{G}$, (abbrev. $\mathcal{F}\sim_\phi\mathcal{G}$),
are the least relations on high types such that $\phi\supseteq\mathcal{F},\mathcal{G}$, and:

$\mathcal{F}\sim_\phi\mathcal{F}$		(EqH Refl)
$\mathcal{F}\sim_\phi\mathcal{G} \ \Rightarrow\ \mathcal{G}\sim_\phi\mathcal{F}$		(EqH Symm)
$\mathcal{F}\sim_\phi\mathcal{G}$ and $\mathcal{G}\sim_\phi\mathcal{H} \ \Rightarrow\ \mathcal{F}\sim_\phi\mathcal{H}$		(EqH Trans)
$\mathcal{A}\sim_{\mathrm{T}\phi}\mathcal{B} \ \Rightarrow\ \mathcal{A}\sim_{\mathrm{H}\phi}\mathcal{B}$		(EqH $\mathcal{A}$ Congr)
$\mathcal{F}\sim_\phi\mathcal{F}'\wedge\mathcal{G}\sim_\phi\mathcal{G}' \ \Rightarrow\ \mathcal{F}\rightarrow\mathcal{G}\sim_\phi\mathcal{F}'\rightarrow\mathcal{G}'$		(EqH $\rightarrow$ Congr)
$\mathcal{F}\sim_{(\phi,\forall x)}\mathcal{G} \ \Rightarrow\ \Pi x.\ \mathcal{F}\sim_\phi\Pi x.\ \mathcal{G}$	➔	(EqH Π Congr)
$\Pi x.\ \mathcal{G}\sim_\phi\Pi y.\ \mathcal{G}\{x\leftarrow y\}$ with $y\notin fv(\mathcal{G})$	➔	(EqH Π-α)

A notion of apartness of names from types is not necessary, since transpositions on types can be distributed down to transpositions on name expressions.

6-3 Definition: Valuation.

(1) If $\varepsilon:Var\rightarrow\Lambda$ is a finite map, then we say that ε is a *valuation*.

(2) We indicate by $\varepsilon(\mathcal{N})$, $\varepsilon(\mathcal{A})$ the homomorphic extensions of ε to name expressions and tree types, with the understanding that in such extension $\varepsilon(x) = x$ for $x\notin dom(\varepsilon)$.

(3) If $fv(\mathcal{N}) \subseteq dom(\varepsilon)$ then we say that ε is a *ground valuation* for $\mathcal{N}$, and we write ε *grounds* $\mathcal{N}$; similarly for $\mathcal{A}$ and $\mathcal{F}$.

(4) We indicate by $\varepsilon(\phi)$ the freshness signature obtained by:

$\varepsilon(\emptyset) = \emptyset$

$\varepsilon(n,\phi) = n,\varepsilon(\phi)$

$\varepsilon(\phi,\mathsf{Q}x) = \varepsilon(x),\varepsilon(\phi)$ if $x\in dom(\varepsilon)$, and $\varepsilon(\phi),\mathsf{Q}x$ otherwise

N.B.: if $\phi \supseteq \mathcal{N}$ then $\varepsilon(\phi) \supseteq \varepsilon(\mathcal{N})$.

We say that a valuation ε satisfies a freshness signature ϕ if it respects the freshness constraints of ϕ, in the following sense:

6-4 Definition: Freshness Signature Satisfaction

$\varepsilon \vDash \phi$ iff $dom(\varepsilon) \subseteq dom(\phi)$

and $\forall x\in dom(\varepsilon).\ \phi(x)=\mathsf{H} \Rightarrow \varepsilon(x)\notin na(\phi)$

and $\forall y\in dom(\varepsilon).\ \forall x\in dom(\phi).\ (x<_\phi y \wedge \phi(y)=\mathsf{H}) \Rightarrow (x\in dom(\varepsilon) \wedge \varepsilon(x)\neq\varepsilon(y))$

In $\varepsilon \vDash \phi$ we do not require $dom(\phi) \subseteq dom(\varepsilon)$, to allow for partial valuations. But we require any partial valuation that instantiates an H variable to instantiate all the variables to the left of it (with distinct names).

6-5 Lemma: Composition of Signature Satisfaction

$\varepsilon_1 \vDash \phi \wedge \varepsilon_2 \vDash \varepsilon_1(\phi)$ implies $dom(\varepsilon_1)\#dom(\varepsilon_2) \wedge \varepsilon_2 \circ \varepsilon_1 \vDash \phi$

The proof of this composition Lemma is in Appendix. Note that if the third condition of Definition 6-4 is changed to the apparently more natural:

$\forall x,y \in dom(\varepsilon).\ (x <_\phi y \wedge \phi(y)=\mathsf{H}) \Rightarrow \varepsilon(x) \neq \varepsilon(y)$

(allowing uninstantiated variables to the left of H variables) then Lemma 6-5 fails. Consider $\phi = \forall x, \mathsf{H}y$, $\varepsilon_1 = y \mapsto n$, $\varepsilon_2 = x \mapsto n$; then $\varepsilon_1 \vDash \phi$, $\varepsilon_2 \vDash \varepsilon_1(\phi) = n, \forall x$, but $\varepsilon_2 \circ \varepsilon_1 = y \mapsto n, x \mapsto n \nvDash \phi$.

Equivalence and apartness are preserved by (partial) valuation (proof in Appendix):

6-6 Proposition: Instances of Equivalance and Apartness

(1) If $\mathcal{N} \#_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{N}) \#_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$.
(2) If $\mathcal{N} \sim_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{N}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$.
(3) If $\mathcal{A} \sim_\phi \mathcal{B}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{A}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{B})$.
(4) If $\mathcal{F} \sim_\phi \mathcal{G}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{F}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{G})$.

We now study the relationship between satisfaction and transposition equivalence. For example, situations like $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{B}$. Fortunately, here $\mathcal{A}$ must be closed, and we require $\mathcal{B}$ to be closed too, so ϕ does not play a significant role.

The main soundness result, Proposition 6-12, requires some careful build-up: results for instantiations of equivalence and apartness under partial valuations, for closure of satisfaction under closed equivalence, and substitution lemmas. The proofs are in Appendix.

6-7 Lemma: Facts about Name Satisfaction and Equivalence.

(1) If $n \vDash \mathcal{N}$ then $n \in na(\mathcal{N})$. If $n \notin na(\mathcal{N})$ and $m \vDash \mathcal{N}$ then $n \neq m$.
(2) $\mathcal{N}[\mathcal{M} \leftrightarrow \mathcal{M}'] \sim_\phi \mathcal{N}'$ iff $\mathcal{N} \sim_\phi \mathcal{N}'[\mathcal{M} \leftrightarrow \mathcal{M}']$.

6-8 Proposition: Soundness of Name Expression Equivalence and Apartness.

For $\mathcal{N}, \mathcal{N}'$ closed:

(1) $\forall \phi \supseteq n, \mathcal{N}.\ n \vDash \mathcal{N}$ iff $n \sim_\phi \mathcal{N}$.
(2) $n \vDash \mathcal{N}$ and $\mathcal{N} \sim_\phi \mathcal{N}'$ imply $n \vDash \mathcal{N}'$.
(3) $n \vDash \mathcal{N}$ and $\mathcal{N} \#_\phi \mathcal{N}'$ and $n' \vDash \mathcal{N}'$ imply $n \neq n'$.

6-9 Lemma: Satisfaction Under Name Expression Substitution.

(1) $n \vDash \mathcal{N}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $n \vDash \mathcal{N}\{x \leftarrow \mathcal{M}'\}$
(2) $P \vDash \mathcal{A}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $P \vDash \mathcal{A}\{x \leftarrow \mathcal{M}'\}$
(3) $F \vDash \mathcal{F}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $F \vDash \mathcal{F}\{x \leftarrow \mathcal{M}'\}$

6-10 Corollary: Alternative Characterization of $P \vDash \mathsf{H}x.\mathcal{A}$.

$P \vDash \mathsf{H}x.\mathcal{A}$ iff $\exists n, P'$, closed $\mathcal{N}$.
$P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $n \vDash \mathcal{N}$ and $P' \vDash \mathcal{A}\{x \leftarrow \mathcal{N}\}$

Proof

By definition, $P \vDash \mathsf{H}x.\mathcal{A}$ iff $\exists n, P'$. $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $P' \vDash \mathcal{A}\{x \leftarrow n\}$.

1) Assume exist $\exists n, P'$. $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $P' \vDash \mathcal{A}\{x \leftarrow n\}$.

Then there exists closed $\mathcal{N} = n$ such that $n \vDash \mathcal{N}$ and $P' \vDash \mathcal{A}\{x \leftarrow \mathcal{N}\}$.

2) Assume $\exists n, P'$, closed $\mathcal{N}$. $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $n \vDash \mathcal{N}$ and $P' \vDash \mathcal{A}\{x \leftarrow \mathcal{N}\}$

By Proposition 6-8, $n \vDash \mathcal{N}$ iff $n \sim_\phi \mathcal{N}$, and by Lemma 6-9, $P' \vDash \mathcal{A}\{x \leftarrow \mathcal{N}\}$
and $n \sim_\phi \mathcal{N}$ imply $P' \vDash \mathcal{A}\{x \leftarrow n\}$.

□

6-11 Proposition: Soundness of Closed Type Equivalence.
(1) If $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{B}$ ($\mathcal{B}$ closed) then $P \vDash \mathcal{B}$.
(2) If $F \vDash \mathcal{F}$ and $\mathcal{F} \sim_\phi \mathcal{G}$ ($\mathcal{G}$ closed) then $F \vDash \mathcal{G}$.

6-12 Proposition: Soundness of Equivalence and Apartness.
If $\mathcal{N} \#_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{N},\mathcal{M}) \Rightarrow \varepsilon(\mathcal{N}) \neq \varepsilon(\mathcal{M})$.
If $\mathcal{N} \sim_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{N},\mathcal{M}) \Rightarrow \varepsilon(\mathcal{N}) = \varepsilon(\mathcal{M})$.
If $\mathcal{A} \sim_\phi \mathcal{B}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{A},\mathcal{B}) \Rightarrow \forall P.\ P \vDash \varepsilon(\mathcal{A}) \Rightarrow P \vDash \varepsilon(\mathcal{B})$.
If $\mathcal{F} \sim_\phi \mathcal{G}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{F},\mathcal{G}) \Rightarrow \forall F.\ F \vDash \varepsilon(\mathcal{F}) \Rightarrow F \vDash \varepsilon(\mathcal{G})$.

7 Type System

We now present a type system that is sound for the operational semantics of Section 5. Subtyping includes the transposition equivalence of Section 6 (see rule (Sub Equiv)), and an unspecified collection of *ValidEntailments* that may capture aspects of logical implication. Apart from the flexibility given by subtyping through rule (Subsumption), the type rules for terms are remarkably straightforward and syntax-driven.

The type system uses environments E that have a slightly unusual structure. They are ordered lists of either names (covering all the names occurring in expressions; see rule (NExpr n)), or variables (covering all the free variables of expressions; see rule (Term x)). Variables have associated type and freshness information of the form $x{:}\mathcal{F}$ if $\mathcal{F} \neq \mathbf{N}$, and $\mathsf{Q}x{:}\mathcal{F}$ (either $\forall x{:}\mathcal{F}$ or $\mathsf{H}x{:}\mathcal{F}$) if $\mathcal{F} = \mathbf{N}$. We write $dom(E)$ and $na(E)$ for the set of variables and the set of names defined by E. We group names to the left and variables to the right; hence we write n,E for the extension of E with a name n, and we write $E,x{:}\mathcal{F}$ and $E,\mathsf{Q}x{:}\mathbf{N}$ for the extension of E with a new variable association (provided that $x \notin dom(E)$), where $\mathcal{F}$ may depend on $dom(E)$. We write $E(x)$ for the (open) type associated to $x \in dom(E)$ in E. A freshness signature (Definition 6-1) can be extracted as follows:

7-1 Definition: Freshness Signature of an Environment

$$\begin{array}{llll} fs(\emptyset) & \triangleq \emptyset & fs(E, \mathsf{Q}x{:}\mathbf{N}) & \triangleq fs(E), \mathsf{Q}x \\ fs(n, E) & \triangleq n, fs(E) & fs(E, x{:}\mathcal{F}) & \triangleq fs(E) \end{array}$$

Through $fs(E)$, in typing rule (Sub Equiv), typing environments are connected to the freshness signatures used in transposition equivalence.

7-2 Definition: Type Rules

Environments. Rules for $E \vdash \diamond$ (that is, E is well-formed).

$$\frac{}{\emptyset \vdash \diamond}\ (\text{Env } \emptyset) \qquad \frac{E \vdash \diamond}{n, E \vdash \diamond}\ (\text{Env } n) \qquad \frac{E \vdash \diamond \quad \mathsf{Q} \in \{\forall, \mathsf{H}\} \quad x \notin dom(E)}{E, \mathsf{Q}x{:}\mathbf{N} \vdash \diamond}\ (\text{Env } x \in \mathbf{N}) \qquad \frac{E \vdash \mathcal{F} \quad \mathcal{F} \neq \mathbf{N} \quad x \notin dom(E)}{E, x{:}\mathcal{F} \vdash \diamond}\ (\text{Env } x \notin \mathbf{N})$$

Names. Rules for $E \vdash_{\mathrm{N}} \mathcal{N}$ (that is, $\mathcal{N}$ is a name expression in E).

$$\frac{E \vdash \diamond \quad n \in na(E)}{E \vdash_{\mathrm{N}} n}\ (\text{NExpr } n) \qquad \frac{E \vdash \diamond \quad E(x) = \mathbf{N}}{E \vdash_{\mathrm{N}} x}\ (\text{NExpr } x) \qquad \frac{E \vdash_{\mathrm{N}} \mathcal{N} \quad E \vdash_{\mathrm{N}} \mathcal{M} \quad E \vdash_{\mathrm{N}} \mathcal{M}'}{E \vdash_{\mathrm{N}} \mathcal{N}[\mathcal{M} \leftrightarrow \mathcal{M}']}\ (\text{NExpr } \leftrightarrow)$$

Tree Types. Rules for $E \vdash_{\mathrm{T}} \mathcal{A}$ (that is, $\mathcal{A}$ is a tree type in E).

(Type **0**) $\dfrac{E \vdash \diamond}{E \vdash_{\mathrm{T}} \mathbf{0}}$ (Type $\mathcal{N}[]$) $\dfrac{E \vdash_{\mathrm{N}} \mathcal{N} \quad E \vdash_{\mathrm{T}} \mathcal{A}}{E \vdash_{\mathrm{T}} \mathcal{N}[\mathcal{A}]}$ (Type $\mid$) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A} \quad E \vdash_{\mathrm{T}} \mathcal{B}}{E \vdash_{\mathrm{T}} \mathcal{A} \mid \mathcal{B}}$

(Type **F**) $\dfrac{E \vdash \diamond}{E \vdash_{\mathrm{T}} \mathbf{F}}$ (Type $\wedge$) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A} \quad E \vdash_{\mathrm{T}} \mathcal{B}}{E \vdash_{\mathrm{T}} \mathcal{A} \wedge \mathcal{B}}$ (Type $\Rightarrow$) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A} \quad E \vdash_{\mathrm{T}} \mathcal{B}}{E \vdash_{\mathrm{T}} \mathcal{A} \Rightarrow \mathcal{B}}$

(Type H) $\dfrac{E, \mathsf{H}x{:}\mathbf{N} \vdash_{\mathrm{T}} \mathcal{A}}{E \vdash_{\mathrm{T}} \mathsf{H}x.\mathcal{A}}$ (Type ©) $\dfrac{E \vdash_{\mathrm{N}} \mathcal{N}}{E \vdash_{\mathrm{T}} ©\mathcal{N}}$ (Type $\leftrightarrow$) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A} \quad E \vdash_{\mathrm{N}} \mathcal{M} \quad E \vdash_{\mathrm{N}} \mathcal{M}'}{E \vdash_{\mathrm{T}} \mathcal{A}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}}$

Types. Rules for $E \vdash \mathcal{T}$ (that is, $\mathcal{T}$ is a type in E).

(Type Tree) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A}}{E \vdash \mathcal{A}}$ (Type **N**) $\dfrac{E \vdash \diamond}{E \vdash \mathbf{N}}$ (Type $\rightarrow$) $\dfrac{E \vdash \mathcal{T} \quad E \vdash \mathcal{G} \quad \mathcal{T} \neq \mathbf{N}}{E \vdash \mathcal{T} \rightarrow \mathcal{G}}$ (Type Π) $\dfrac{E, \forall x{:}\mathbf{N} \vdash \mathcal{G}}{E \vdash \Pi x.\, \mathcal{G}}$

Subtyping. Rules for $E \vdash \mathcal{T} <: \mathcal{G}$ (that is, $\mathcal{T}$ is a subtype of $\mathcal{G}$ in E).

(Sub Tree) $\dfrac{E \vdash_{\mathrm{T}} \mathcal{A} \quad E \vdash_{\mathrm{T}} \mathcal{B} \quad \langle \mathcal{A}, fs(E), \mathcal{B} \rangle \in \mathit{ValidEntailments}}{E \vdash \mathcal{A} <: \mathcal{B}}$ (Sub Equiv) $\dfrac{E \vdash \mathcal{T} \quad E \vdash \mathcal{G} \quad \mathcal{T} \sim_{fs(E)} \mathcal{G}}{E \vdash \mathcal{T} <: \mathcal{G}}$

(Sub **N**) $\dfrac{E \vdash \diamond}{E \vdash \mathbf{N} <: \mathbf{N}}$ (Sub $\rightarrow$) $\dfrac{E \vdash \mathcal{T}' <: \mathcal{T} \quad E \vdash \mathcal{G} <: \mathcal{G}' \quad \mathcal{T}, \mathcal{T}' \neq \mathbf{N}}{E \vdash \mathcal{T} \rightarrow \mathcal{G} <: \mathcal{T}' \rightarrow \mathcal{G}'}$ (Sub Π) $\dfrac{E, \forall x{:}\mathbf{N} \vdash \mathcal{G} <: \mathcal{G}'}{E \vdash \Pi x.\, \mathcal{G} <: \Pi x.\, \mathcal{G}'}$

Terms. Rules for $E \vdash t : \mathcal{T}$ (t has type $\mathcal{T}$ in E, with $E \vdash_{\mathrm{T}} t : \mathcal{A} \triangleq E \vdash_{\mathrm{T}} \mathcal{A} \wedge E \vdash t : \mathcal{A}$).

(Term 0) $\dfrac{E \vdash \diamond}{E \vdash 0 : \mathbf{0}}$ (Term $\mathcal{N}[]$) $\dfrac{E \vdash_{\mathrm{N}} \mathcal{N} \quad E \vdash_{\mathrm{T}} t : \mathcal{A}}{E \vdash \mathcal{N}[t] : \mathcal{N}[\mathcal{A}]}$ (Term $\mid$) $\dfrac{E \vdash_{\mathrm{T}} t : \mathcal{A} \quad E \vdash_{\mathrm{T}} u : \mathcal{B}}{E \vdash t \mid u : \mathcal{A} \mid \mathcal{B}}$

(Term ν) $\dfrac{E, \mathsf{H}x{:}\mathbf{N} \vdash_{\mathrm{T}} t : \mathcal{A}}{E \vdash (\nu x)t : \mathsf{H}x.\mathcal{A}}$ (Term $\leftrightarrow$) $\dfrac{E \vdash_{\mathrm{T}} t : \mathcal{A} \quad E \vdash_{\mathrm{N}} \mathcal{M} \quad E \vdash_{\mathrm{N}} \mathcal{M}'}{E \vdash t\{\mathcal{M}\leftrightarrow\mathcal{M}'\} : \mathcal{A}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}}$

(Term $\div\mathcal{N}[]$) $\dfrac{E \vdash_{\mathrm{T}} t : \mathcal{N}[\mathcal{A}] \quad E, y{:}\mathcal{A} \vdash u : \mathcal{T}}{E \vdash t \div (\mathcal{N}[y{:}\mathcal{A}]).u : \mathcal{T}}$ (Term $\div\mid$) $\dfrac{E \vdash_{\mathrm{T}} t : \mathcal{A} \mid \mathcal{B} \quad E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{T}}{E \vdash t \div (x{:}\mathcal{A} \mid y{:}\mathcal{B}).u : \mathcal{T}}$

(Term $\div\nu$) $\dfrac{E \vdash_{\mathrm{T}} t : \mathsf{H}x.\mathcal{A} \quad E, \mathsf{H}x{:}\mathbf{N}, y{:}\mathcal{A} \vdash_{\mathrm{T}} u : \mathcal{B}}{E \vdash t \div ((\nu x)y{:}\mathcal{A}).u : \mathsf{H}x.\mathcal{B}}$ (Term ?) $\dfrac{E \vdash_{\mathrm{T}} t : \mathcal{B} \quad E, x{:}\mathcal{A} \vdash u : \mathcal{T} \quad E, x{:}\neg\mathcal{A} \vdash v : \mathcal{T}}{E \vdash t?(x{:}\mathcal{A}).u, v : \mathcal{T}}$

(Term x) $\dfrac{E \vdash \diamond \quad x \in dom(E)}{E \vdash x : E(x)}$

(Term $\mathcal{N}$) $\dfrac{E \vdash_{\mathrm{N}} \mathcal{N}}{E \vdash \mathcal{N} : \mathbf{N}}$

(Term λ) $\dfrac{E,\ x{:}\mathcal{F} \vdash t : \mathcal{G} \quad \mathcal{F} \neq \mathbf{N}}{E \vdash \lambda x{:}\mathcal{F}.t : \mathcal{F} \to \mathcal{G}}$

(Term App) $\dfrac{E \vdash t : \mathcal{F} \to \mathcal{G} \quad E \vdash u : \mathcal{F}}{E \vdash t(u) : \mathcal{G}}$

(Term Depλ) $\dfrac{E,\ \forall x{:}\mathbf{N} \vdash t : \mathcal{G}}{E \vdash \lambda x{:}\mathbf{N}.t : \Pi x.\ \mathcal{G}}$

(Term DepApp) $\dfrac{E \vdash t : \Pi x.\ \mathcal{G} \quad E \vdash_{\mathrm{N}} \mathcal{N}}{E \vdash t(\mathcal{N}) : \mathcal{G}\{x \leftarrow \mathcal{N}\}}$

(Subsumption) $\dfrac{E \vdash t : \mathcal{F} \quad E \vdash \mathcal{F} <: \mathcal{G}}{E \vdash t : \mathcal{G}}$

Notes: • As we already mentioned, the type system includes dependent types, with binding operators $\mathsf{H}x.\mathcal{A}$ (the type of hiding in trees) and $\Pi x.\mathcal{F}$ (the type of those functions $\lambda x{:}\mathbf{N}.t$ such that the type $\mathcal{F}$ of t may depend on the input variable x).
• The subtyping relation is parameterized by a set *ValidEntailments*, assumed to consist of triples $\langle \mathcal{A}, \phi, \mathcal{B} \rangle$ that are sound ($\forall \varepsilon \vDash \phi.\ (\varepsilon\ grounds\ \mathcal{A},\mathcal{B}) \Rightarrow \forall P.\ P \vDash \varepsilon(\mathcal{A}) \Rightarrow P \vDash \varepsilon(\mathcal{B})$).
• The use of $x{:}\neg\mathcal{A}$ in (Term ?) means $x{:}\mathcal{A} \Rightarrow \mathbf{F}$, but this assumption is not very useful without a rich theory of subtyping: see discussion in Section 1.5. On the other hand, there are no significant problems in executing run-time type tests such as $P \vDash \neg\mathcal{A}$ (see Definition 4-1), e.g., resulting from $t?(x{:}\neg\mathcal{A}).u,v$. A more informative typing of x for the third assumption of this rule is $x{:}\mathcal{B} \wedge \neg\mathcal{A}$, but we lack a compelling use for it.
• In (Term $\div\mathcal{N}$[]) (and (Term $\div$|), (Term ?)) we do not need extra assumptions $E \vdash \mathcal{F}$ to avoid the escape of y (and x,y, and x) into $\mathcal{F}$, because these are not variables of type **N**, and $\mathcal{F}$ cannot depend on them. We do not need the extra assumption in (Term $\div\nu$) for x because there we rebind the result type.
• In (TermDepApp) we require the argument $\mathcal{N}$ to be a name expression, not an expression of type **N**, so we can do a substitution $\mathcal{G}\{x \leftarrow \mathcal{N}\}$ into the type. Note that $E \vdash t : \mathbf{N}$ means that t can be any computation of type **N**, unlike $E \vdash_{\mathrm{N}} \mathcal{N}$.

A stack satisfies an environment, $\rho \vDash E$, if $\rho(x) \vDash \rho(E(x))$ for all x's in $dom(E)$; note the extra $\rho(\text{-})$ used to bind the dependent variables in $E(x)$. Here $\rho(\mathcal{F})$ or $\rho(\mathcal{A})$ means that ρ is used as a valuation (Definition 6-3). In addition, though, we require ρ to satisfy the freshness requirements of the H-quantified variables in E.

7-3 Definition: Environment Satisfaction

$\rho \vDash E$, for a stack ρ and environment E, is defined as follows:

$\rho \vDash n_1, \ldots, n_k$ always
$\rho \vDash E, x{:}\mathcal{A}$ iff $\exists \rho', x, F.\ \rho = \rho'[x \leftarrow F] \wedge x \notin dom(\rho') \wedge \rho' \vDash E' \wedge F \vDash \rho'(\mathcal{A})$
$\rho \vDash E, \forall x{:}\mathbf{N}$ iff $\exists \rho', x, n.\ \rho = \rho'[x \leftarrow n] \wedge x \notin dom(\rho') \wedge \rho' \vDash E'$
$\rho \vDash E, \mathsf{H}x{:}\mathbf{N}$ iff $\exists \rho', x, n.\ \rho = \rho'[x \leftarrow n] \wedge x \notin dom(\rho') \wedge \rho' \vDash E' \wedge n \notin na(E') \wedge n \notin na(\rho')$

The following lemma connects environment satisfaction to freshness signature satisfaction, when a stack is reduced to a valuation and an environment is reduced to a freshness signature. The proof is in Appendix.

7-4 Lemma: Environemnt Satisfaction and Freshness Signatures

If $\rho \vDash E$ then $\rho|_{dom(fs(E))} \vDash fs(E)$
where $\rho|_S$ is the restriction of ρ to the domain S,
and $\vDash fs(E)$ is in the sense of Definition 6-4

7-5 Theorem: Subject Reduction.

(1) If $E \vdash \mathcal{F} <: \mathcal{G}$ and $\rho \vDash E$ and $F \vDash_H \rho(\mathcal{F})$ then $F \vDash_H \rho(\mathcal{G})$.

(2) If $E \vdash_N \mathcal{N}$ and $\rho \vDash E$ and $\mathcal{N} \downarrow_\rho n$ then $n \vDash_N \rho(\mathcal{N})$.

(3) If $E \vdash t : \mathcal{F}$ and $\rho \vDash E$ and $t \Downarrow_\rho F$, then $F \vDash_H \rho(\mathcal{F})$.

Proof

The full proof is in Appendix. Here we show the (Term ν) case of (3), which is by induction on the derivation of $E \vdash t : \mathcal{F}$.

We have $E \vdash (\nu x)t : \mathsf{H}x.\mathcal{A}$ and $\rho \vDash E$ and $(\nu x)t \Downarrow_\rho F$. We have from (Term ν) $E, \mathsf{H}x{:}\mathbf{N} \vdash_T t : \mathcal{A}$, and from (Red ν) $F = (\nu n)P$ and $t \Downarrow_{\rho[x\leftarrow n]} P$ for $n \notin na(t,\rho)$. Since n could appear in $\mathcal{A}$, blocking the last step of this proof, take $n' \notin na(t,\mathcal{A},\rho,P)$, so that $F \equiv_\alpha (\nu n')P\bullet\{n\leftrightarrow n'\}$. By Lemma 5-2 $t\bullet\{n\leftrightarrow n'\} \Downarrow_{\rho[x\leftarrow n]\bullet\{n\leftrightarrow n'\}} P\bullet\{n\leftrightarrow n'\}$, that is $t \Downarrow_{\rho[x\leftarrow n']} P\bullet\{n\leftrightarrow n'\}$. We have $\rho[x\leftarrow n'] \vDash E, \mathsf{H}x{:}\mathbf{N}$. By Ind Hyp, $P\bullet\{n\leftrightarrow n'\} \vDash \rho[x\leftarrow n'](\mathcal{A})$, that is, $P\bullet\{n\leftrightarrow n'\} \vDash \rho\backslash x(\mathcal{A})\{x\leftarrow n'\}$. Since $F \equiv (\nu n')P\bullet\{n\leftrightarrow n'\}$ and $n' \notin na(\rho\backslash x(\mathcal{A}))$, by Definition 4-1, $F \vDash \mathsf{H}x.\rho\backslash x(\mathcal{A})$. That is, $F \vDash \rho(\mathsf{H}x.\mathcal{A})$. □

8 Examples

We discuss some programming examples, using plausible (but not formally checked) extensions of the formal development of the previous sections. In particular, we use recursive types, ***rec*** $X.\ \mathcal{A}$, existential types $\exists x.\mathcal{A}$ where x ranges over names (these are simpler to handle than $\mathsf{H}x.\mathcal{A}$), and a variant of location matching, $t\div(\underline{x}[\underline{y}{:}\mathcal{A}]).u$, that binds labels x from the data in addition to contents y (its typing requires existential types). Examples of transposition types have been discussed in the Introduction; here we concentrate on pattern matching, using some abbreviations:

test *t* ***as*** *w*:$\mathcal{A}$ ***then*** *u* ***else*** *v*	for	$t?(w{:}\mathcal{A}).\ u,\ v$
match *t* ***as*** (*pattern*) ***then*** *u* ***else*** *v*	for	$t?(w{:}\mathcal{B}).\ (w\div(pattern).u),\ v$

where $\mathcal{B}$ is the type naturally extracted from *pattern*.

We also use nested patterns, in the examples, which can be defined in a similar way. We use standard notations for recursive function definitions. We sometimes underline binding occurrences of variables, for clarity. We explicitly list the subtypings, if any, that must be included in *ValidEntailments* for these examples to typecheck (none are needed for the examples in Section 1.2).

Basic. Duplicating a given label, and duplicating a hidden label:

$$\lambda x{:}\mathbf{N}.\ \lambda y{:}x[\mathbf{T}].\ x[y] \quad : \quad \Pi x.\ (x[\mathbf{T}] \to x[x[\mathbf{T}]])$$
$$\lambda z{:}(\mathsf{H}x.x[\mathbf{T}]).\ z\div((\nu x)y{:}x[\mathbf{T}]).\ x[y] \quad : \quad (\mathsf{H}x.x[\mathbf{T}]) \to (\mathsf{H}x.x[x[\mathbf{T}]])$$

Collect. Collect all the subtrees that satisfy $\mathcal{A}$, even under restrictions:

let type *Result* = ***rec*** $X.\ \mathbf{0} \vee \mathcal{A} \vee (X \mid X) \vee \mathsf{H}x.X$

let rec *collect*($\underline{x}$: **T**): *Result* =
(***test*** *x* ***as*** $\underline{w}{:}\mathcal{A}$ ***then*** *w* ***else*** 0) |
(***test*** *x* ***as*** $\underline{w}{:}\mathbf{0}$ ***then*** 0 ***else***
match *x* ***as*** ($\underline{y}{:}\neg\mathbf{0} \mid \underline{w}{:}\neg\mathbf{0}$) ***then*** *collect*(*y*) | *collect*(*w*) ***else***
match *x* ***as*** ($\underline{y}[\underline{w}{:}\mathbf{T}]$) ***then*** *collect*($\underline{w}$) ***else***
match *x* ***as*** ($(\nu\underline{y})\underline{w}{:}\copyright y$) ***then*** *collect*(*w*) ***else*** 0)

Recall that, in the last match, a (νy) is automatically wrapped around the result; hence the

$\mu X.X$ in the definition of *Result*. The typing $\underline{w}$:©y (instead of $\underline{w}$:**T**) is used to reduce nondeterminism by forcing the analysis of non-redundant restrictions. Similarly, the pattern $\neg\mathbf{0} \mid \neg\mathbf{0}$ is used to avoid vacuous splits where one component is 0. In general, the splitting of composition is nondeterministic; in this case the result may or may not be uniquely determined depending on the shape of $\mathcal{A}$. The subtypings needed here are $\mathcal{A} <: \mathbf{T}$, $\mathcal{A} <: \mathcal{A} \vee \mathcal{B}$ and ***rec*** fold/unfold.

Removing Dangling Pointers. We can encode addresses and pointers in the same style as in XML. An address definition is encoded as *addr*[*n*[0]], where *addr* is a conventional name, and *n* is the name of a particular address. A pointer to an address is encoded as *ptr*[*n*[0]], where *ptr* is another conventional name. Addresses may be global, like URLs, or local, like XML's IDs; local addresses are represented by restriction: (ν*n*) ... *addr*[*n*[0]] ... *ptr*[*n*[0]] ... . A tree should not contain two address definitions for the same name, but this assumption is not important in our example.

We write a function that copies a tree, including both public and private addresses, but deletes all the pointers that do not have a corresponding address in the tree. Every time a *ptr*[*n*[0]] is found, we need to see if there is an *addr*[*n*[0]] somewhere in the tree. But we cannot search for *addr*[*n*[0]] in the original tree, because *n* may be a restricted address we have come across. So, we first open all the (non-trivial) restrictions, and then we proceed as above, passing the root of the restriction-free tree as an additional parameter. The search for *addr*[*n*[0]] can be done by a single type test for *Somewhere*(*addr*[*n*[**0**]]), where *Somewhere*($\mathcal{A}$) ≜ ***rec*** *X*. ($\mathcal{A} \mid \mathbf{T}$) ∨ ∃*y*.(*y*[*X*] | **T**).

```
let rec deDangle(x: T): T =
    match x as ((νy)w:©y) then deDangle(w) else f (x, x)
and f (x: T, root: T): T =
    test x as w:0 then 0 else
    match x as (y:¬0 | w:¬0) then f (y, root) | f (w, root) else
    match x as (ptr[y[0]]) then
        test root as w:Somewhere(addr[y[0]]) then ptr[y[0]] else 0 else
    match x as (z[w:T]) then z[f (w, root)] else 0
```

Note that *deDangle* automatically recloses, in the result, all the restrictions that it opens. The subtypings needed here are just $\mathcal{A} <: \mathbf{T}$.

9 Conclusions and Acknowledgments

We have introduced a language and a rich type system for manipulating semistructured data with hidden labels and scope extrusion, via pattern matching and transpositions.

As advocated in [23,30], our formal development could be carried out within a metatheory with transpositions; then, Lemmas 4-3 and 5-2 would fall out of the metatheory, and one could be less exposed to mistakes in α-conversion issues. We have not gone that far, but we should seriously consider this option in the future.

We are not aware of previous uses of transpositions in structural operational semantics, although this falls within the general framework of [30]. We believe ours is the first calculus or language with explicit transpositions in the syntax of terms and types.

Thanks to Murdoch J. Gabbay for illuminating discussions on transpositions, and to Luís Caires who indirectly influenced this paper through earlier work with the first author. Moreover, Gabbay and Caires helped simplify the technical presentation.

References

[1] S. Abiteboul, P. Buneman, D. Suciu.: **Data on the Web**. Morgan Kaufmann Publishers, 2000.

[2] S. Abiteboul, P. Kanellakis: **Object identity as a query language primitive.** Journal of the ACM, 45(5):798-842, 1998. A first version appeared in SIGMOD'89.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. **The Lorel Query Language for Semistructured Data.** International Journal on Digital Libraries, 1(1), pp. 68-88, April 1997.

[4] M.P. Atkinson, F. Bancilhon, et al.: **The Object-Oriented Database System Manifesto.** Building an Object-Oriented Database System, The Story of O2, 1992, pp. 3-20.

[5] V. Benzaken, G. Castagna, A. Frisch: **CDuce: a white paper.** PLAN-X: Programming Language Technologies for XML, Pittsburgh PA, Oct. 2002. http://www.cduce.org.

[6] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, J. Siméon: **XQuery 1.0: An XML Query Language,** W3C Working Draft, 2002, http://www.w3.org/TR/xquery.

[7] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler: **Extensible Markup Language (XML) 1.0 (Second Edition)**, W3C document, http://www.w3.org/TR/REC-xml.

[8] P. Buneman, S.B. Davidson, G.G. Hillebrand, D. Suciu: **A Query Language and Optimization Techniques for Unstructured Data.** SIGMOD Conference 1996, pp. 505-516.

[9] L. Caires: **A Model for Declarative Programming and Specification with Concurrency and Mobility**. Ph.D. Thesis, Dept. de Informática, FTC, Universidade Nove de Lisboa, 1999.

[10] L. Caires, L Cardelli: **A Spatial Logic for Concurrency: Part I**. Proc. TACS 2001, Naoki Kobayashi and Benjamin C. Pierce (Eds.). LNCS. 2215. Springer, 2001, pp 1-37. To appear in I&C.

[11] L. Caires, L. Cardelli: **A Spatial Logic for Concurrency: Part II**. Proc. CONCUR'02, 2002.

[12] C. Calcagno, L. Cardelli, A.D. Gordon, **Deciding Validity in a Spatial Logic for Trees**. Draft.

[13] C. Calcagno, H. Yang, P.W. O'Hearn: **Computability and Complexity Results for a Spatial Assertion Language for Data Structures**. Proc. FSTTCS 2001, pp. 108-119.

[14] L. Cardelli, P. Gardner, G. Ghelli, **A Spatial Logic for Querying Graphs**. Proc. ICALP'02, Peter Widmayer et al. (Eds.). LNCS 2380, Springer, 2002. pp 597-610.

[15] L. Cardelli, G. Ghelli, **A Query Language Based on the Ambient Logic**. Proc. ESOP'01, David Sands (Ed.). LNCS 2028, Springer, 2001, pp. 1-22.

[16] L. Cardelli, A.D. Gordon, **Anytime, Anywhere. Modal Logics for Mobile Ambients.** Proc. of the 27th ACM Symposium on Principles of Programming Languages, 2000, pp. 365-377.

[17] S. Cluet, S. Jacqmin, and J. Simeon. **The New YATL: Design and Specifications.** INRIA, 1999.

[18] E. Cohen: **Validity and Model Checking for Logics of Finite Multisets.** Draft.

[19] D. Florescu, A. Deutsch, A. Levy, D. Suciu, M. Fernandez: **A Query Language for XML**. In Proc. of Eighth International World Wide Web Conference, 1999.

[20] D. Florescu, A. Levy, M. Fernandez, D. Suciu, **A Query Language for a Web-Site Management System.** SIGMOD Record , vol. 26 , no. 3 , pp. 4-11 , September, 1997.

[21] M.J. Gabbay: **A Theory of Inductive Definitions with a-Equivalence: Semantics, Implementation, Programming Language.** Ph.D. Thesis, University of Cambridge, 2000.

[22] M.J. Gabbay, A.M. Pitts, **A New Approach to Abstract Syntax Involving Binders**. Proc. LICS1999. IEEE Computer Society Press, 1999. pp 214-224.

[23] M.J. Gabbay: **FM-HOL, A Higher-Order Theory of Names.** In Thirty Five years of Automath, Heriot-Watt University, Edinburgh, April 2002. Inforal Proc., 2002.

[24] A.D. Gordon: **Notes on Nominal Calculi for Security and Mobility.** R.Focardi, R.Gorrieri (Eds.): Foundations of Security Analysis and Design. LNCS 2171. Springer, 1998.

[25] A.D. Gordon, A. Jeffrey: **Typing Correspondence Assertions for Communication Protocols.** MFPS 17, Elsevier Electronic Notes in Theoretical Computer Science, Vol 45, 2001.

[26] H. Hosoya, B. C. Pierce: **XDuce: A Typed XML Processing Language (Preliminary Report)**. WebDB (Selected Papers) 2000, pp: 226-244

[27] R. Milner: **Communicating and Mobile Systems: the π-Calculus**. Cambridge U. Press, 1999.

[28] P.W. O'Hearn, D. Pym: **Logic of Bunched Implication.** Bulletin of Symbolic Logic 5(2), pp 215-244, 1999.

[29] P.W. O'Hearn, J.C. Reynolds, H. Yang: **Local Reasoning about Programs that Alter Data Structures**. Proc. CSL 2001, pp. 1-19.

[30] A.M. Pitts: **Nominal Logic, A First Order Theory of Names and Binding.** Proc. TACS 2001, Naoki Kobayashi and Benjamin C. Pierce (Eds.). LNCS 2215. Springer, 2001, pp 219-242.

[31] A.M. Pitts, M.J. Gabbay: **A Metalanguage for Programming with Bound Names Modulo Renaming.** R. Backhouse and J.N. Oliveira (Eds.): MPC 2000, LNCS 1837, Springer, pp. 230-255.

10 Appendix: Proofs

Proof of 4-2 Proposition: Tree Satisfaction Under Structural Congruence. (p.10)
If $P \vDash \mathcal{A}$ and $P \equiv Q$ then $Q \vDash \mathcal{A}$.

Proof

By induction on the structure of $\mathcal{A}$. Standard [16], except for the new cases:

($\mathcal{N}[\mathcal{A}]$) We have $P \vDash \mathcal{N}[\mathcal{A}]$ iff $n \vDash \mathcal{N}$ and $P \equiv n[P']$ and $P' \vDash \mathcal{A}$. Let $Q \equiv P$. Then $Q \equiv n[P']$. That is, $Q \vDash \mathcal{N}[\mathcal{A}]$.

($\mathsf{H}x.\mathcal{B}$) We have $P \vDash \mathsf{H}x.\mathcal{A}$ iff $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A})$ and $P' \vDash \mathcal{A}\{x \leftarrow n\}$. Let $Q \equiv P$. Then $Q \equiv (\nu n)P'$. That is, $Q \vDash \mathsf{H}x.\mathcal{A}$.

($\copyright\mathcal{N}$) We have $P \vDash \copyright\mathcal{N}$ iff $n \vDash \mathcal{N}$ and $n \in fn(P)$. Let $Q \equiv P$. Then $fn(Q) = fn(P)$ by Lemma 2-6. That is, $Q \vDash \copyright\mathcal{N}$.

($\mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$) We have $P \vDash \mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$ iff $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $P \bullet (m \leftrightarrow m') \vDash \mathcal{A}$. Let $Q \equiv P$. Then by Lemma 2-6, $Q \bullet (m \leftrightarrow m') \equiv P \bullet (m \leftrightarrow m')$. By Ind. Hyp. $Q \bullet (m \leftrightarrow m') \vDash \mathcal{A}$. That is, $Q \vDash \mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.

□

10-1 Lemma: Uniqueness of Name Satisfaction.
If $n \vDash \mathcal{N}$ and $m \vDash \mathcal{N}$ then $n = m$.

Proof

Induction on the structure of $\mathcal{N}$.

If $\mathcal{N} = p$, then $n = p = m$.

If $\mathcal{N} = \mathcal{R}\{\mathcal{P} \leftrightarrow \mathcal{Q}\}$, then for some p,q, we have $n \bullet \{p \leftrightarrow q\} \vDash \mathcal{R}$ and $p \vDash \mathcal{P}$ and $q \vDash \mathcal{Q}$, and for some p',q', we have $m \bullet \{p' \leftrightarrow q'\} \vDash \mathcal{R}$ and $p' \vDash \mathcal{P}$ and $q' \vDash \mathcal{Q}$. By Ind Hyp $p = p'$ and $q = q'$. By Ind Hyp also $n \bullet \{p \leftrightarrow q\} = m \bullet \{p \leftrightarrow q\}$. By properties of transpositions, we must have $n = m$. (E.g. apply $\bullet\{p \leftrightarrow q\}$ to both sides and simplify.)

□

10-2 Lemma: Name and Tree Satisfaction Under Transposition.
(1) If $n \vDash \mathcal{N}$ and $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$, then $n \bullet (m \leftrightarrow m') \vDash \mathcal{N}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.
(2) If $P \vDash \mathcal{A}$ and $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$, then $P \bullet (m \leftrightarrow m') \vDash \mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.

Proof

(1) By the properties of transpositions, $n = n \bullet (m \leftrightarrow m') \bullet (m \leftrightarrow m')$. Hence, by assumption, $n \bullet (m \leftrightarrow m') \bullet (m \leftrightarrow m') \vDash \mathcal{N}$, and by definition of satisfaction, $n \bullet (m \leftrightarrow m') \vDash \mathcal{N}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.

(2) Similar.

□

N.B. The following lemma is used to do alpha conversions on trees in restriction cases. Hence, a third statement for high-value satisfaciton is not needed, and therefore the lemma is not dependent on the operational semantics of section 5

Proof of 4-3 Lemma: Name and Tree Satisfaction Under Actual Transposition. (p.10)
(1) If $n \vDash \mathcal{N}$ then $n \bullet (m \leftrightarrow m') \vDash \mathcal{N} \bullet (m \leftrightarrow m')$.
(2) If $P \vDash \mathcal{A}$ then $P \bullet (m \leftrightarrow m') \vDash \mathcal{A} \bullet (m \leftrightarrow m')$.

Proof

(1) Induction on the structure of $\mathcal{N}$.

Case x**.** Impossible, since $n \vDash \mathcal{N}$ means that $\mathcal{N}$ is closed.

Case n**.** Then we have $n \vDash n$. By definition of satisfaction, $n\bullet(m\leftrightarrow m') \vDash n\bullet(m\leftrightarrow m')$.

Case $\mathcal{N}_0(\mathcal{N}_1\leftrightarrow\mathcal{N}_2)$**.** Then we have $n \vDash \mathcal{N}_0(\mathcal{N}_1\leftrightarrow\mathcal{N}_2)$. By definition of satisfaction, $\exists n_1,n_2.\ n_1 \vDash \mathcal{N}_1$ and $n_2 \vDash \mathcal{N}_2$ and $n\bullet(n_1\leftrightarrow n_2) \vDash \mathcal{N}_0$.
By Ind Hyp $n_1\bullet(m\leftrightarrow m') \vDash \mathcal{N}_1\bullet(m\leftrightarrow m')$ and $n_2\bullet(m\leftrightarrow m') \vDash \mathcal{N}_2\bullet(m\leftrightarrow m')$
and $n\bullet(n_1\leftrightarrow n_2)\bullet(m\leftrightarrow m') \vDash \mathcal{N}_0\bullet(m\leftrightarrow m')$.
Moreover, $n\bullet(m\leftrightarrow m')\bullet(n_1\bullet(m\leftrightarrow m')\leftrightarrow n_2\bullet(m\leftrightarrow m')) = n\bullet(n_1\leftrightarrow n_2)\bullet(m\leftrightarrow m')$.
By taking $n'_1 = n_1\bullet(m\leftrightarrow m')$ and $n'_2 = n_2\bullet(m\leftrightarrow m')$ we have shown that
$\exists n'_1,n'_2.\ n'_1 \vDash \mathcal{N}_1\bullet(m\leftrightarrow m')$ and $n'_2 \vDash \mathcal{N}_2\bullet(m\leftrightarrow m')$
and $n\bullet(m\leftrightarrow m')\bullet(n'_1\leftrightarrow n_2') \vDash \mathcal{N}_0\bullet(m\leftrightarrow m')$.
Therefore, $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}_0\bullet(m\leftrightarrow m')(\mathcal{N}_1\bullet(m\leftrightarrow m')\leftrightarrow\mathcal{N}_2\bullet(m\leftrightarrow m'))$
that is, $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}_0(\mathcal{N}_1\leftrightarrow\mathcal{N}_2)\bullet(m\leftrightarrow m')$.

(2) Induction on the structure of $\mathcal{A}$, the interesting cases are:

Case $\mathcal{N}[\mathcal{A}]$**.** Assume $P \vDash \mathcal{N}[\mathcal{A}]$, then $n \vDash \mathcal{N}$ and $P \equiv n[P']$ and $P' \vDash \mathcal{A}$.
By (1), $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}\bullet(m\leftrightarrow m')$, and by Ind Hyp $P'\bullet(m\leftrightarrow m') \vDash \mathcal{A}\bullet(m\leftrightarrow m')$.
Hence $n\bullet(m\leftrightarrow m')[P'\bullet(m\leftrightarrow m')] \vDash \mathcal{N}\bullet(m\leftrightarrow m')[\mathcal{A}\bullet(m\leftrightarrow m')]$,
that is $n[P']\bullet(m\leftrightarrow m') \vDash \mathcal{N}[\mathcal{A}]\bullet(m\leftrightarrow m')$.

Case $\mathsf{H}x.\mathcal{A}$**.** Assume $P \vDash \mathsf{H}x.\mathcal{A}$, then $P\equiv(\nu n)P'$ and $n\notin na(\mathcal{A})$ and $P' \vDash \mathcal{A}\{x\leftarrow n\}$.
Take a fresh $q\notin na((\nu n)P', \mathcal{A}, n, m, m')$. ($q$ is needed in case $n=m$ or $n=m'$.)
By Ind Hyp $P'\bullet(n\leftrightarrow q) \vDash \mathcal{A}\{x\leftarrow n\}\bullet(n\leftrightarrow q)$;
since $n,q\notin na(\mathcal{A})$, this means $P'\bullet(n\leftrightarrow q) \vDash \mathcal{A}\{x\leftarrow q\}$.
By Ind Hyp, $P'\bullet(n\leftrightarrow q)\bullet(m\leftrightarrow m') \vDash \mathcal{A}\{x\leftarrow q\}\bullet(m\leftrightarrow m')$
That is, $P'\bullet(n\leftrightarrow q)\bullet(m\leftrightarrow m') \vDash \mathcal{A}\bullet(m\leftrightarrow m')\{x\leftarrow q\}$.
Since $P \equiv (\nu n)P' = (\nu q)P'\bullet(n\leftrightarrow q)$
we have that $P\bullet(m\leftrightarrow m') \equiv ((\nu q)P'\bullet(n\leftrightarrow q))\bullet(m\leftrightarrow m') = (\nu q)P'\bullet(n\leftrightarrow q)\bullet(m\leftrightarrow m')$.
Therefore, we have that $P\bullet(m\leftrightarrow m') \equiv (\nu q)P'\bullet(n\leftrightarrow q)\bullet(m\leftrightarrow m')$
with $q \notin na(\mathcal{A}\bullet(m\leftrightarrow m'))$ and $P'\bullet(n\leftrightarrow q)\bullet(m\leftrightarrow m') \vDash \mathcal{A}\bullet(m\leftrightarrow m')\{x\leftarrow q\}$;
by definition of satisfaction we conclude $P\bullet(m\leftrightarrow m') \vDash \mathsf{H}x.\mathcal{A}\bullet(m\leftrightarrow m')$,
that is $P\bullet(m\leftrightarrow m') \vDash (\mathsf{H}x.\mathcal{A})\bullet(m\leftrightarrow m')$.

Case $\copyright\mathcal{N}$**.** Assume $P \vDash \copyright\mathcal{N}$, then $\exists n.\ n \vDash \mathcal{N}$ and $n\in fn(P)$.
By (1), $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}\bullet(m\leftrightarrow m')$.
Since $n\in fn(P)$, we have $n\bullet(m\leftrightarrow m')\in fn(P\bullet(m\leftrightarrow m'))$.
Take $n' = n\bullet(m\leftrightarrow m')$; we have shown that:
$\exists n'.\ n' \vDash \mathcal{N}\bullet(m\leftrightarrow m')$ and $n'\in fn(P\bullet(m\leftrightarrow m'))$.
By definition of satisfaction, $P\bullet(m\leftrightarrow m') \vDash \copyright(\mathcal{N}\bullet(m\leftrightarrow m'))$
That is, $P\bullet(m\leftrightarrow m') \vDash (\copyright\mathcal{N})\bullet(m\leftrightarrow m')$.

Case $\mathcal{A}(\mathcal{N}\leftrightarrow\mathcal{N}')$**.** Assume $P \vDash \mathcal{A}(\mathcal{N}\leftrightarrow\mathcal{N}')$,
then $n \vDash \mathcal{N}$ and $n' \vDash \mathcal{N}'$ and $P\bullet(n\leftrightarrow n') \vDash \mathcal{A}$.
By Ind Hyp, $P\bullet(n\leftrightarrow n')\bullet(m\leftrightarrow m') \vDash \mathcal{A}\bullet(m\leftrightarrow m')$
that is, $P\bullet(m\leftrightarrow m')\bullet(n\bullet(m\leftrightarrow m')\leftrightarrow n'\bullet(m\leftrightarrow m')) \vDash \mathcal{A}\bullet(m\leftrightarrow m')$ by Lemma .
that is, $P\bullet(m\leftrightarrow m') \vDash \mathcal{A}\bullet(m\leftrightarrow m')(\mathcal{N}\bullet(m\leftrightarrow m')\leftrightarrow\mathcal{N}'\bullet(m\leftrightarrow m'))$ by (1)

that is, $P\bullet(m\leftrightarrow m')\vDash \mathcal{A}(\mathcal{N}\leftrightarrow\mathcal{N}')\bullet(m\leftrightarrow m')$ by definition.

□

Proof of 5-2 Lemma: Reduction Under Transposition. (p.11)

(1) If $\mathcal{M}\downarrow_\rho m$ then $\mathcal{M}\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')$.

(2) If $t\Downarrow_\rho F$ then $t\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')} F\bullet(n\leftrightarrow n')$.

Proof

This lemma should be a completely routine application of a metatheory of transpositions [30]. Nonetheless, we show some selected interesting cases.

1) Induction on the derivation of $\mathcal{M}\downarrow_\rho m$.

Case (NRed x). We have $x\downarrow_\rho \rho(x)$ with $x\in dom(\rho)$ and $\rho(x)=m\in\Lambda$.
Then $x = x\bullet(n\leftrightarrow n')$ and $\rho\bullet(n\leftrightarrow n')(x)=m\bullet(n\leftrightarrow n')$,
and by (NRed x) $x\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')$.

Case (NRed n). We have $m\downarrow_\rho m$.
By (NRed n) we have $m\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')$.

Case (NRed $\leftrightarrow$). We have $\mathcal{M}_0(\mathcal{M}_1\leftrightarrow\mathcal{M}_2)\downarrow_\rho m_0\bullet(m_1\leftrightarrow m_2)$,
with $\mathcal{M}_0\downarrow_\rho m_0$ and $\mathcal{M}_1\downarrow_\rho m_1$ and $\mathcal{M}_2\downarrow_\rho m_2$.
By Ind Hyp, $\mathcal{M}_0\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m_0\bullet(n\leftrightarrow n')$ and $\mathcal{M}_1\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m_1\bullet(n\leftrightarrow n')$
and $\mathcal{M}_2\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m_2\bullet(n\leftrightarrow n')$.
By (NRed $\leftrightarrow$) $\mathcal{M}_0\bullet(n\leftrightarrow n')(\mathcal{M}_1\bullet(n\leftrightarrow n')\leftrightarrow\mathcal{M}_2\bullet(n\leftrightarrow n'))$
$\downarrow_{\rho\bullet(n\leftrightarrow n')} m_0\bullet(n\leftrightarrow n')\bullet(m_1\bullet(n\leftrightarrow n')\leftrightarrow m_2\bullet(n\leftrightarrow n'))$,
which by Definition 3-2 means
$\mathcal{M}_0(\mathcal{M}_1\leftrightarrow\mathcal{M}_2)\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m_0\bullet(m_1\leftrightarrow m_2)\bullet(n\leftrightarrow n')$.

2) Induction on the derivation of $t\Downarrow_\rho P$.

Case (Red 0). We have $0\Downarrow_\rho 0$. By (Red 0), $0\bullet(n\leftrightarrow n') = 0\Downarrow_{\rho\bullet(n\leftrightarrow n')} 0 = 0\bullet(n\leftrightarrow n')$.

Case (Red $\mathcal{N}[]$). We have $\mathcal{M}[t]\Downarrow_\rho m[P]$ with $\mathcal{M}\downarrow_\rho m$ and $t\Downarrow_\rho P$.
By (1), $\mathcal{M}\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')$,
and by Ind Hyp $t\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')} P\bullet(n\leftrightarrow n')$.
By (Red $\mathcal{N}[]$) $\mathcal{M}\bullet(n\leftrightarrow n')[t\bullet(n\leftrightarrow n')]\Downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')[P\bullet(n\leftrightarrow n')]$
that is, $\mathcal{M}[t]\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')} m[P]\bullet(n\leftrightarrow n')$

Case (Red ν). We have $(\nu x)t\Downarrow_\rho(\nu p)P$ with $t\Downarrow_{\rho[x\leftarrow p]} P$ and $p\notin na(t,\rho)$.
By Ind Hyp $t\bullet(n\leftrightarrow n')\Downarrow_{\rho[x\leftarrow p]\bullet(n\leftrightarrow n')} P\bullet(n\leftrightarrow n')$,
that is $t\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')[x\leftarrow p\bullet(n\leftrightarrow n')]} P\bullet(n\leftrightarrow n')$.
We have that $p\bullet(n\leftrightarrow n')\notin na(t\bullet(n\leftrightarrow n'),\rho\bullet(n\leftrightarrow n'))$,
hence by (Red ν), $(\nu x)t\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')}(\nu p\bullet(n\leftrightarrow n'))P\bullet(n\leftrightarrow n')$,
that is $((\nu x)t)\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')}((\nu p)P)\bullet(n\leftrightarrow n')$.

Case (Red $\leftrightarrow$). We have $t(\mathcal{M}\leftrightarrow\mathcal{M}')\Downarrow_\rho P\bullet(m\leftrightarrow m')$
with $t\Downarrow_\rho P$ and $\mathcal{M}\downarrow_\rho m$ and $\mathcal{M}'\downarrow_\rho m'$.
By (1) $\mathcal{M}\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m\bullet(n\leftrightarrow n')$ and $\mathcal{M}'\bullet(n\leftrightarrow n')\downarrow_{\rho\bullet(n\leftrightarrow n')} m'\bullet(n\leftrightarrow n')$
and by Ind Hyp $t\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')} P\bullet(n\leftrightarrow n')$.
Hence by (Red $\leftrightarrow$), $t\bullet(n\leftrightarrow n')(\mathcal{M}\bullet(n\leftrightarrow n')\leftrightarrow\mathcal{M}'\bullet(n\leftrightarrow n'))$
$\Downarrow_{\rho\bullet(n\leftrightarrow n')} P\bullet(n\leftrightarrow n')\bullet(m\bullet(n\leftrightarrow n')\leftrightarrow m'\bullet(n\leftrightarrow n'))$
that is, $t(\mathcal{M}\leftrightarrow\mathcal{M}')\bullet(n\leftrightarrow n')\Downarrow_{\rho\bullet(n\leftrightarrow n')} P\bullet(m\leftrightarrow m')\bullet(n\leftrightarrow n')$.

Case (Red ÷ν). We have $t \div ((\nu x) y{:}\mathcal{A}).u \Downarrow_\rho (\nu p)Q$, with $p \notin na(t, \mathcal{A}, u, \rho)$ and $t \Downarrow_\rho \equiv (\nu p)P$ and $P \vDash \rho[x \leftarrow p](\mathcal{A})$ and $x \neq y$ and $u \Downarrow_{\rho[x \leftarrow p][y \leftarrow P]} Q$.

By Ind Hyp and Lemma 2-6, $t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \equiv ((\nu p)P) \bullet (n \leftrightarrow n')$,

that is $t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \equiv (\nu p \bullet (n \leftrightarrow n'))P \bullet (n \leftrightarrow n')$.

and by Ind Hyp, $u \bullet (n \leftrightarrow n') \Downarrow_{\rho[x \leftarrow p][y \leftarrow P] \bullet (n \leftrightarrow n')} Q \bullet (n \leftrightarrow n')$,

that is $u \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')[x \leftarrow p \bullet (n \leftrightarrow n')][y \leftarrow P \bullet (n \leftrightarrow n')]} Q \bullet (n \leftrightarrow n')$.

We have that $p \bullet (n \leftrightarrow n') \notin na(t \bullet (n \leftrightarrow n'), \mathcal{A} \bullet (n \leftrightarrow n'), u \bullet (n \leftrightarrow n'), \rho \bullet (n \leftrightarrow n'))$.

Since $P \vDash \rho[x \leftarrow p](\mathcal{A})$, by Lemma 4-3, $P \bullet (n \leftrightarrow n') \vDash \rho[x \leftarrow p](\mathcal{A}) \bullet (n \leftrightarrow n')$,

that is, $P \bullet (n \leftrightarrow n') \vDash \rho \bullet (n \leftrightarrow n')[x \leftarrow p \bullet (n \leftrightarrow n')](\mathcal{A} \bullet (n \leftrightarrow n'))$. Hence by (Red ÷ν),

$t \bullet (n \leftrightarrow n') \div ((\nu x) y{:}\mathcal{A} \bullet (n \leftrightarrow n')).u \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} (\nu p \bullet (n \leftrightarrow n'))Q \bullet (n \leftrightarrow n')$,

that is $(t \div ((\nu x) y{:}\mathcal{A}).u) \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} ((\nu p)Q) \bullet (n \leftrightarrow n')$.

Case (Red ?⊨). We have $t?(x{:}\mathcal{A}).u,v \Downarrow_\rho F$ with $t \Downarrow_\rho P$ and $P \vDash \rho(\mathcal{A})$ and $u \Downarrow_{\rho[x \leftarrow P]} F$.

By Ind Hyp $t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} P \bullet (n \leftrightarrow n')$,

and $u \bullet (n \leftrightarrow n') \Downarrow_{\rho[x \leftarrow P] \bullet (n \leftrightarrow n')} F \bullet (n \leftrightarrow n')$,

that is $u \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')[x \leftarrow P \bullet (n \leftrightarrow n')]} F \bullet (n \leftrightarrow n')$.

By Lemma 4-3, $P \bullet (n \leftrightarrow n') \vDash \rho(\mathcal{A}) \bullet (n \leftrightarrow n') = \rho \bullet (n \leftrightarrow n')(\mathcal{A} \bullet (n \leftrightarrow n'))$.

Hence by (Red ?⊨), we have

$t \bullet (n \leftrightarrow n')?(x{:}\mathcal{A} \bullet (n \leftrightarrow n')).u \bullet (n \leftrightarrow n'), v \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} F \bullet (n \leftrightarrow n')$,

that is $(t?(x{:}\mathcal{A}).u,v) \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} F \bullet (n \leftrightarrow n')$.

Case (Red ?⊭). Similar.

Case (Red x). We have $x \Downarrow_\rho \rho(x)$ with $x \in dom(\rho)$.

Then $x \in dom(\rho \bullet (n \leftrightarrow n'))$, and by (Red x), $x \Downarrow_{\rho \bullet (n \leftrightarrow n')} \rho \bullet (n \leftrightarrow n')(x)$,

that is $x \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \rho(x) \bullet (n \leftrightarrow n')$.

Case (Red $\mathcal{N}$). We have $\mathcal{M} \Downarrow_\rho m$, with $\mathcal{M} \downarrow_\rho m$.

By (1), $\mathcal{M} \bullet (n \leftrightarrow n') \downarrow_{\rho \bullet (n \leftrightarrow n')} m \bullet (n \leftrightarrow n')$,

and by (Red $\mathcal{N}$), $\mathcal{M} \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} m \bullet (n \leftrightarrow n')$.

Case (Red λ). We have $\lambda x{:}\mathcal{T}.t \Downarrow_\rho \langle \rho, x, t \rangle$.

By (Red λ) we have $\lambda x{:}\mathcal{T} \bullet (n \leftrightarrow n').t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \langle \rho \bullet (n \leftrightarrow n'), x, t \bullet (n \leftrightarrow n') \rangle$

That is $(\lambda x{:}\mathcal{T}.t) \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \langle \rho, x, t \rangle \bullet (n \leftrightarrow n')$.

Case (Red App). We have $t(u) \Downarrow_\rho H$, with $t \Downarrow_\rho \langle \rho', x, t' \rangle$ and $u \Downarrow_\rho G$ and $t' \Downarrow_{\rho'[x \leftarrow G]} H$.

By Ind Hyp, $t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \langle \rho', x, t' \rangle \bullet (n \leftrightarrow n')$

that is, $t \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} \langle \rho' \bullet (n \leftrightarrow n'), x, t' \bullet (n \leftrightarrow n') \rangle$

and $u \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} G \bullet (n \leftrightarrow n')$

and $t' \bullet (n \leftrightarrow n') \Downarrow_{\rho'[x \leftarrow G] \bullet (n \leftrightarrow n')} H \bullet (n \leftrightarrow n')$

that is $t' \bullet (n \leftrightarrow n') \Downarrow_{\rho' \bullet (n \leftrightarrow n')[x \leftarrow G \bullet (n \leftrightarrow n')]} H \bullet (n \leftrightarrow n')$.

By (Red App), $t \bullet (n \leftrightarrow n')(u \bullet (n \leftrightarrow n')) \Downarrow_{\rho \bullet (n \leftrightarrow n')} H \bullet (n \leftrightarrow n')$

that is $t(u) \bullet (n \leftrightarrow n') \Downarrow_{\rho \bullet (n \leftrightarrow n')} H \bullet (n \leftrightarrow n')$.

□

Proof of 6-5 Lemma: Composition of Signature Satisfaction (p.14)

$\varepsilon_1 \vDash \phi \wedge \varepsilon_2 \vDash \varepsilon_1(\phi)$ implies $dom(\varepsilon_1) \# dom(\varepsilon_2) \wedge \varepsilon_2 \circ \varepsilon_1 \vDash \phi$

Proof

Assume $\varepsilon_1 \vDash \phi \wedge \varepsilon_2 \vDash \varepsilon_1(\phi)$; that is, by definition:

(a_1) $dom(\varepsilon_1) \subseteq dom(\phi)$
(b_1) $\forall x \in dom(\varepsilon_1).\ \phi(x)=\mathsf{H} \Rightarrow \varepsilon_1(x) \notin na(\phi)$
(c_1) $\forall y \in dom(\varepsilon_1).\ \forall x \in dom(\phi).\ (x <_\phi y \wedge \phi(y)=\mathsf{H}) \Rightarrow (x \in dom(\varepsilon_1) \wedge \varepsilon_1(x) \neq \varepsilon_1(y))$
(a_2) $dom(\varepsilon_2) \subseteq dom(\varepsilon_1(\phi))$
(b_2) $\forall y \in dom(\varepsilon_2).\ \varepsilon_1(\phi)(y)=\mathsf{H} \Rightarrow \varepsilon_2(y) \notin na(\varepsilon_1(\phi))$
(c_2) $\forall y \in dom(\varepsilon_2).\forall x \in dom(\varepsilon_1(\phi)).\ (x <_{\varepsilon_1(\phi)} y \wedge \varepsilon_1(\phi)(y)=\mathsf{H}) \Rightarrow (x \in dom(\varepsilon_2) \wedge \varepsilon_2(x) \neq \varepsilon_2(y))$

Case (#) We show that $dom(\varepsilon_1) \# dom(\varepsilon_2)$.

Suppose $x \in dom(\varepsilon_1)$, then by (a_1) $x \in dom(\phi)$ and by Definition 6-3, $x \notin dom(\varepsilon_1(\phi))$.

Suppose also $x \in dom(\varepsilon_2)$, then by (a_2) $x \in dom(\varepsilon_1(\phi))$. Contradiction.

Case ($\circ$) We derive the three conditions for $\varepsilon_2 \circ \varepsilon_1 \vDash \phi$ from Definition 6-4.

(a) By (a_1) $dom(\varepsilon_1) \subseteq dom(\phi)$. By ($a_2$) $dom(\varepsilon_2) \subseteq dom(\varepsilon_1(\phi)) = dom(\phi) - dom(\varepsilon_1) \subseteq dom(\phi)$. Hence $dom(\varepsilon_2 \circ \varepsilon_1) = dom(\varepsilon_1) \cup dom(\varepsilon_2) \subseteq dom(\phi)$.

(b) Take any $x \in dom(\varepsilon_2 \circ \varepsilon_1) = dom(\varepsilon_1) \cup dom(\varepsilon_2)$; by (#) either $x \in dom(\varepsilon_1)$ or $x \in dom(\varepsilon_2)$.

Assume $\phi(x)=\mathsf{H}$.

(i) Suppose $x \in dom(\varepsilon_1)$, by (b_1) we have $\varepsilon_1(x) \notin na(\phi)$. Hence also $\varepsilon_2 \circ \varepsilon_1(x) \notin na(\phi)$.

(ii) Suppose $x \in dom(\varepsilon_2)$, then $\varepsilon_1(\phi)(x)=\phi(x)=\mathsf{H}$, and by ($b_2$) we have $\varepsilon_2(x) \notin na(\varepsilon_1(\phi))$. Since $na(\phi) \subseteq na(\varepsilon_1(\phi))$, we have that $\varepsilon_2 \circ \varepsilon_1(x) = \varepsilon_2(x) \notin na(\phi)$.

In both cases, we have shown that $\forall x \in dom(\varepsilon_2 \circ \varepsilon_1).\ \phi(x)=\mathsf{H} \Rightarrow \varepsilon_2 \circ \varepsilon_1(x) \notin na(\phi)$.

(c) Take any $y \in dom(\varepsilon_2 \circ \varepsilon_1) = dom(\varepsilon_1) \cup dom(\varepsilon_2)$; by (#) either $y \in dom(\varepsilon_1)$ or $y \in dom(\varepsilon_2)$.

Take any $x \in dom(\phi)$ and assume $x <_\phi y \wedge \phi(y)=\mathsf{H}$.

(i) Suppose $y \in dom(\varepsilon_1)$, by (c_1) we have $x \in dom(\varepsilon_1) \wedge \varepsilon_1(x) \neq \varepsilon_1(y)$.

Hence also $x \in dom(\varepsilon_2 \circ \varepsilon_1)$ and $\varepsilon_2 \circ \varepsilon_1(x) = \varepsilon_1(x) \neq \varepsilon_1(y) = \varepsilon_2 \circ \varepsilon_1(y)$.

(ii) Suppose $y \in dom(\varepsilon_2)$. Suppose that $x \notin dom(\varepsilon_1(\phi))$, which means that $x \in dom(\varepsilon_1)$. Hence also $x \in dom(\varepsilon_2 \circ \varepsilon_1)$. Since $\varepsilon_1(\phi)(y)=\phi(y)=\mathsf{H}$, by ($b_2$) we have $\varepsilon_2(y) \notin na(\varepsilon_1(\phi))$, that is $\varepsilon_2(y) \neq \varepsilon_1(x)$. Hence $\varepsilon_2 \circ \varepsilon_1(x) = \varepsilon_1(x) \neq \varepsilon_2(y) = \varepsilon_2 \circ \varepsilon_1(y)$.

Suppose instead that $x \in dom(\varepsilon_1(\phi))$. Since $x <_\phi y$ we have also $x <_{\varepsilon_1(\phi)} y$. Moreover $\varepsilon_1(\phi)(y)=\phi(y)=\mathsf{H}$. Then by ($c_2$) we have $x \in dom(\varepsilon_2) \wedge \varepsilon_2(x) \neq \varepsilon_2(y)$. Hence $x \in dom(\varepsilon_2 \circ \varepsilon_1)$ and $\varepsilon_2 \circ \varepsilon_1(x) = \varepsilon_2(x) \neq \varepsilon_2(y) = \varepsilon_2 \circ \varepsilon_1(y)$.

Therefore, in all cases we have shown that $\forall y \in dom(\varepsilon_2 \circ \varepsilon_1).\ \forall x \in dom(\phi).\ (x <_\phi y \wedge \phi(y)=\mathsf{H}) \Rightarrow (x \in dom(\varepsilon_2 \circ \varepsilon_1) \wedge \varepsilon_2 \circ \varepsilon_1(x) \neq \varepsilon_2 \circ \varepsilon_1(y))$.

□

Proof of 6-6 Proposition: Instances of Equivalance and Apartness (p.14)

(1) If $\mathcal{N} \#_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{N}) \#_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$.
(N.B.: for ground valuations we obtain $\varepsilon(\mathcal{N}) \neq \varepsilon(\mathcal{M})$)

(2) If $\mathcal{N} \sim_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{N}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$.
(N.B.: for ground valuations we obtain $\varepsilon(\mathcal{N}) = \varepsilon(\mathcal{M})$)

(3) If $\mathcal{A} \sim_\phi \mathcal{B}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{A}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{B})$.

(4) If $\mathcal{F} \sim_\phi \mathcal{G}$ then $\forall \varepsilon \vDash \phi.\ \varepsilon(\mathcal{F}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{G})$.

Proof

By induction on the derivations. We check some intersting cases: transitivity (because of the lack of a subformula property) and quantifiers (because of bound variables), and the apartness cases. N.B.: we must deal with partial (non-ground) instantiations to get through the induction in the congruence rules for quantifiers.

(1) Cases (1) and (2) are by mutual induction.

Case (Apart Symm) $\mathcal{N}\#_\phi\mathcal{N} \Rightarrow \mathcal{N}\#_\phi\mathcal{M}$.

By Ind Hyp, $\forall\varepsilon\vDash\phi.\ \varepsilon(\mathcal{M})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{N})$. Take any $\varepsilon\vDash\phi$, then $\varepsilon(\mathcal{M})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{N})$, and by (Apart Symm) $\varepsilon(\mathcal{N})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{M})$. That is, $\forall\varepsilon\vDash\phi.\ \varepsilon(\mathcal{N})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{M})$.

Case (Apart Names) $n\neq m \Rightarrow n\ \#_\phi\ m$, where $\phi\supseteq n,m$.

Take any $\varepsilon\vDash\phi$; we have $\varepsilon(n) = n$ and $\varepsilon(m) = m$; by assumption $\varepsilon(n)\ \#_\phi\ \varepsilon(m)$.

Case (Apart Name Var) $\phi(x)=\mathsf{H} \Rightarrow n\ \#_\phi\ x$, where $\phi\supseteq n,x$.

Take any $\varepsilon\vDash\phi$.

If $x\in dom(\varepsilon)$ then, since $\phi(x)=\mathsf{H}$, by Definition 6-4 we have $\varepsilon(x)\notin na(\phi)$. Hence $\varepsilon(n) = n \neq \varepsilon(x)$. We have $\varepsilon(\phi)\supseteq\varepsilon(n),\varepsilon(x)$, and by (Apart Names) we obtain $\varepsilon(n)\ \#_\phi\ \varepsilon(x)$.

If $x\notin dom(\varepsilon)$ then $\varepsilon(\phi)\supseteq\varepsilon(n),\varepsilon(x)$ and $\varepsilon(\phi)(x)=\mathsf{H}$ and $\varepsilon(n) = n$, and $\varepsilon(x) = x$; by (Apart Name Var), $\varepsilon(n)\ \#_{\varepsilon(\phi)}\ \varepsilon(x)$.

Case (Apart Vars) $x<_\phi y$ and $\phi(y)=\mathsf{H} \Rightarrow x\ \#_\phi\ y$, where $\phi\supseteq x,y$.

Take any $\varepsilon\vDash\phi$.

If $y\in dom(\varepsilon)$, since $x<_\phi y$ and $\phi(y)=\mathsf{H}$, by Definition 6-4 we have $x\in dom(\varepsilon)$ and $\varepsilon(x)\neq\varepsilon(y)$. We have $\varepsilon(\phi)\supseteq\varepsilon(x),\varepsilon(y)$ and by (Apart Names) $\varepsilon(x)\ \#_{\varepsilon(\phi)}\ \varepsilon(y)$.

If $y\notin dom(\varepsilon)$ and $x\in dom(\varepsilon)$ then $\varepsilon(x) = n$ for some n. Since $\phi(y)=\mathsf{H}$, we have $\varepsilon(\phi)(y)=\mathsf{H}$. We also have $\varepsilon(\phi)\supseteq\varepsilon(x),\varepsilon(y)$. Hence, by (Apart Name Var) we obtain $\varepsilon(x)\ \#_{\varepsilon(\phi)}\ \varepsilon(y)$.

If $y,x\notin dom(\varepsilon)$ then $\varepsilon(\phi)\supseteq\varepsilon(x),\varepsilon(y)$ and $x<_{\varepsilon(\phi)}y$ and $\varepsilon(\phi)(y)=\mathsf{H}$ and $\varepsilon(x) = x$, and $\varepsilon(y) = y$; by (Apart Vars) $\varepsilon(x)\ \#_{\varepsilon(\phi)}\ \varepsilon(y)$.

Cases (Apart Congr), (Apart Equiv) By Ind Hyp (1) and (2) and respective rules.

(2) Cases (1) and (2) are by mutual induction.

Cases (EqN Refl)...(EqN $\leftrightarrow\leftrightarrow$) All directly by the respective rules, possibly using IndHyp (2).

Case (EqN $\leftrightarrow$ Apart) $\mathcal{N}\#_\phi\mathcal{M}$ and $\mathcal{N}\#_\phi\mathcal{M}' \Rightarrow \mathcal{N}[\mathcal{M}\leftrightarrow\mathcal{M}'] \sim_\phi \mathcal{N}$, where $\phi\supseteq\mathcal{N}[\mathcal{M}\leftrightarrow\mathcal{M}']$. Take any $\varepsilon\vDash\phi$; by Ind Hyp (1), $\varepsilon(\mathcal{N})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{M})$ and $\varepsilon(\mathcal{N})\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{M}')$.
We have $\varepsilon(\phi)\supseteq\varepsilon(\mathcal{N}[\mathcal{M}\leftrightarrow\mathcal{M}'])$, hence by (EqN $\leftrightarrow$ Apart),
$\varepsilon(\mathcal{N})[\varepsilon(\mathcal{M})\leftrightarrow\varepsilon(\mathcal{M}')]\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{N})$, that is $\varepsilon(\mathcal{N}[\mathcal{M}\leftrightarrow\mathcal{M}'])\ \#_{\varepsilon(\phi)}\ \varepsilon(\mathcal{M})$.

(3) If $\mathcal{A}\sim_\phi\mathcal{B}$ then $\forall\varepsilon\vDash\phi.\ \varepsilon(\mathcal{A})\sim_{\varepsilon(\phi)}\varepsilon(\mathcal{B})$.

Cases (EqT Refl) (EqT Symm) (EqT Trans). By Ind Hyp (3) and respective rules.

N.B. (EqT Trans) does not allow us to forget the part of ϕ that covers only the cut formula $\mathcal{B}$.

Cases (EqT $\mathcal{N}$[] Congr) ... (EqT $\Rightarrow$ Congr), (EqT © Congr). By Ind Hyp (2-3) and respective rules.

Case (EqT H Congr) $\mathcal{A}\sim_{(\phi,\mathsf{H}x)}\mathcal{B} \Rightarrow \mathsf{H}x.\ \mathcal{A}\sim_\phi\mathsf{H}x.\ \mathcal{B}$, where $\phi\supseteq\mathsf{H}x.\mathcal{A},\mathsf{H}x.\ \mathcal{B}$.

By the shape of the assumption $(\phi,\mathsf{H}x)$, we have $x\notin dom(\phi)$.

Take any $\varepsilon\vDash\phi$, we have $\varepsilon\backslash x\vDash(\phi,\mathsf{H}x)$ and $\varepsilon\backslash x(\phi) = \varepsilon(\phi)$.

By Ind Hyp, $\forall\varepsilon'\vDash(\phi,\mathsf{H}x).\ \varepsilon'(\mathcal{A})\sim_{\varepsilon'(\phi,\mathsf{H}x)}\varepsilon'(\mathcal{B})$.

Hence, taking $\varepsilon'=\varepsilon\backslash x$, we get $\varepsilon\backslash x(\mathcal{A})\sim_{(\varepsilon(\phi),\mathsf{H}x)}\varepsilon\backslash x(\mathcal{B})$.

We also have $\varepsilon(\phi)\supseteq\varepsilon(\mathsf{H}x.\mathcal{A}),\varepsilon(\mathsf{H}x.\ \mathcal{B})$, that is $\varepsilon(\phi)\supseteq\mathsf{H}x.\varepsilon\backslash x(\mathcal{A}),\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{B})$.

By (EqT H Congr), $\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{A})\sim_{\varepsilon(\phi)}\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{B})$, that is, $\varepsilon(\mathsf{H}x.\ \mathcal{A})\sim_{\varepsilon(\phi)}\varepsilon(\mathsf{H}x.\ \mathcal{B})$.

Cases (EqT 0 $\leftrightarrow$) ... (EqT $\Rightarrow\leftrightarrow$), (EqT © $\leftrightarrow$), (EqT $\leftrightarrow\leftrightarrow$).

By the respective rules, noting that $\varepsilon(\phi)$ covers the appropriate formulas.

Case (EqT H ↔) $(\mathsf{H}x.\mathcal{A})\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathsf{H}x.\ (\mathcal{A}\{x\leftarrow x\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\})$
with $x\notin fv(\mathcal{M},\mathcal{M}')$, where $\phi\supseteq\mathsf{H}x.\mathcal{A},\mathcal{M},\mathcal{M}'$. Then, $\varepsilon(\phi)\supseteq\varepsilon(\mathsf{H}x.\mathcal{A}),\varepsilon(\mathcal{M}),\varepsilon(\mathcal{M}')$.
Take any $\varepsilon\vDash\phi$. Since $x\notin fv(\mathcal{M},\mathcal{M}')$:
$\varepsilon((\mathsf{H}x.\mathcal{A})\{\mathcal{M}\leftrightarrow\mathcal{M}'\}) = \varepsilon\backslash x((\mathsf{H}x.\mathcal{A})\{\mathcal{M}\leftrightarrow\mathcal{M}'\}) = (\mathsf{H}x.\varepsilon\backslash x(\mathcal{A}))\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}$,
and $\varepsilon(\mathsf{H}x.\mathcal{A}\{x\leftarrow x\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\}) = \varepsilon\backslash x(\mathsf{H}x.\mathcal{A}\{x\leftarrow x\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\})$
$= \mathsf{H}x.\varepsilon\backslash x(\mathcal{A})\{x\leftarrow x\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}\}\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}$.
Now, $x\notin fv(\varepsilon\backslash x(\mathcal{M}),\varepsilon\backslash x(\mathcal{M}))$,
and $\varepsilon(\phi)\supseteq\mathsf{H}x.\varepsilon\backslash x(\mathcal{A}),\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{A})\{x\leftarrow x\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}\}\varepsilon\backslash x(\mathcal{M}),\varepsilon\backslash x(\mathcal{M})$,
hence by (EqT H ↔), $(\mathsf{H}x.\varepsilon\backslash x(\mathcal{A}))\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\} \sim_{\varepsilon(\phi)}$
$\mathsf{H}x.\varepsilon\backslash x(\mathcal{A})\{x\leftarrow x\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}\}\{\varepsilon\backslash x(\mathcal{M})\leftrightarrow\varepsilon\backslash x(\mathcal{M}')\}$
that is, $\varepsilon((\mathsf{H}x.\mathcal{A})\{\mathcal{M}\leftrightarrow\mathcal{M}'\}) \sim_{\varepsilon(\phi)} \varepsilon(\mathsf{H}x.\ (\mathcal{A}\{x\leftarrow x\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\}))$.

Case (EqT H-α) $\mathsf{H}x.\mathcal{A} \sim_\phi \mathsf{H}y.\mathcal{A}\{x\leftarrow y\}$ with $y\notin fv(\mathcal{A})$, where $\phi\supseteq\mathsf{H}x.\mathcal{A},\mathsf{H}y.\mathcal{A}\{x\leftarrow y\}$.
Take any $\varepsilon\vDash\phi$, then $\varepsilon(\phi)\supseteq\varepsilon(\mathsf{H}x.\mathcal{A}),\varepsilon(\mathsf{H}y.\mathcal{A}\{x\leftarrow y\})$.
Since, $y\notin fv(\mathcal{A})$, also $y\notin fv(\varepsilon\backslash x(\mathcal{A}))$.
We have $\varepsilon(\mathsf{H}x.\mathcal{A}) = \varepsilon\backslash x(\mathsf{H}x.\mathcal{A}) = \mathsf{H}x.\ \varepsilon\backslash x(\mathcal{A})$
and $\varepsilon(\mathsf{H}y.\mathcal{A}\{x\leftarrow y\}) = \varepsilon\backslash y(\mathsf{H}y.\mathcal{A}\{x\leftarrow y\}) = \mathsf{H}y.\ \varepsilon\backslash y(\mathcal{A}\{x\leftarrow y\}) = \mathsf{H}y.\ \varepsilon\backslash x(\mathcal{A})\{x\leftarrow y\}$.
Hence $\varepsilon(\phi)\supseteq\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{A}),\ \mathsf{H}y.\ \varepsilon\backslash x(\mathcal{A})\{x\leftarrow y\}$, so
by (EqT H-α) $\mathsf{H}x.\ \varepsilon\backslash x(\mathcal{A}) \sim_{\varepsilon(\phi)} \mathsf{H}y.\ \varepsilon\backslash x(\mathcal{A})\{x\leftarrow y\}$,
that is $\varepsilon(\mathsf{H}x.\mathcal{A}) \sim_{\varepsilon(\phi)} \varepsilon(\mathsf{H}y.\mathcal{A}\{x\leftarrow y\})$.

(4)

Cases (EqH Refl) (EqH Symm) (EqH Trans). By Ind Hyp (4) and respective rules.
Cases (EqH $\mathcal{A}$ Congr), (EqH → Congr). By Ind Hyp (3-4) and respective rules.
Case (EqH Π Congr) Similar to case (EqT H Congr).
Case (EqH Π-α) Similar to case (EqT H-α).

□

Proof of 6-7 Lemma: Facts about Name Satisfaction and Equivalence. (p.14)
(1) If $n\vDash\mathcal{N}$ then $n\in na(\mathcal{N})$. If $n\notin na(\mathcal{N})$ and $m\vDash\mathcal{N}$ then $n\neq m$.
(2) $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}'$ iff $\mathcal{N} \sim_\phi \mathcal{N}'\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$

Proof

(1) The first part is by induction on $\mathcal{N}$. The base case is trivial. If $\mathcal{N} = \mathcal{N}_0\{\mathcal{N}_1\leftrightarrow\mathcal{N}_2\}$ then $n = n_0\bullet\{n_1\leftrightarrow n_2\}$ with $n_i\vDash\mathcal{N}_i$; then by cases on n_0, n_1, n_2, we see that $n\in na(\mathcal{N})$. The second part is because, if $m\vDash\mathcal{N}$, then by the first part we have $m\in na(\mathcal{N})$. Hence if $n\notin na(\mathcal{N})$ we must have $n\neq m$.

(2) Assume $\mathcal{N} \sim_\phi \mathcal{N}'\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$. By (EqN ↔ Congr), $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}'\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$. By (EqN ↔ Inv) and (EqN ↔ Trans), $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}'$. The converse is similar.

□

Proof of 6-8 Proposition: Soundness of Name Expression Equivalence and Apartness. (p.14)

For $\mathcal{N}$, $\mathcal{N}'$ closed:
(1) $\forall\phi\supseteq n,\mathcal{N}.\ n\vDash\mathcal{N}$ iff $n \sim_\phi \mathcal{N}$.
(2) $n\vDash\mathcal{N}$ and $\mathcal{N} \sim_\phi \mathcal{N}'$ imply $n\vDash\mathcal{N}'$.
(3) $n\vDash\mathcal{N}$ and $\mathcal{N}\ \#_\phi\ \mathcal{N}'$ and $n'\vDash\mathcal{N}'$ imply $n\neq n'$.

Proof

We prove **(1⇒)** first.

The we prove the following cases by mutual induction on the ~ and # derivations:

(1⇐ a) $n \sim_\phi \mathcal{N} \Rightarrow n \vDash \mathcal{N}$,

(1⇐ b) $\mathcal{N} \sim_\phi n \Rightarrow n \vDash \mathcal{N}$,

(2a) $n \vDash \mathcal{M}$ and $\mathcal{M} \sim_\phi \mathcal{N} \Rightarrow n \vDash \mathcal{N}$.

(2b) $n \vDash \mathcal{M}$ and $\mathcal{N} \sim_\phi \mathcal{M} \Rightarrow n \vDash \mathcal{N}$.

(3a) $n \vDash \mathcal{N}$ and $\mathcal{N} \#_\phi \mathcal{M}$ and $m \vDash \mathcal{M}$ imply $n \neq m$.

(3b) $n \vDash \mathcal{N}$ and $\mathcal{M} \#_\phi \mathcal{N}$ and $m \vDash \mathcal{M}$ imply $n \neq m$.

(1⇒) For $\mathcal{N}$ closed, $\forall \phi \supseteq n, \mathcal{N}.\ n \vDash \mathcal{N} \Rightarrow n \sim_\phi \mathcal{N}$.

Induction on the structure of $\mathcal{N}$.

Case m**.** Then $n \vDash m \Rightarrow n = m \Rightarrow n \sim_\phi m$ by (EqN Refl).

Case $\mathcal{N}_0\{\mathcal{N}_1 \leftrightarrow \mathcal{N}_2\}$**.** Then $n \vDash \mathcal{N}_0\{\mathcal{N}_1 \leftrightarrow \mathcal{N}_2\}$ implies, for some n_1, n_2, that $n_1 \vDash \mathcal{N}_1$ and $n_2 \vDash \mathcal{N}_2$ and $n \bullet \{n_1 \leftrightarrow n_2\} \vDash \mathcal{N}_0$. By Lemma 6-7(1) we have $\phi \supseteq n \bullet \{n_1 \leftrightarrow n_2\}$. By Ind Hyp $n \bullet \{n_1 \leftrightarrow n_2\} \sim_\phi \mathcal{N}_0$ and $n_1 \sim_\phi \mathcal{N}_1$ and $n_2 \sim_\phi \mathcal{N}_2$.

Then, by congruence $(n \bullet \{n_1 \leftrightarrow n_2\})\{n_1 \leftrightarrow n_2\} \sim_\phi \mathcal{N}_0\{\mathcal{N}_1 \leftrightarrow \mathcal{N}_2\}$.

Suppose $n = n_1$; then $n \bullet \{n_1 \leftrightarrow n_2\} = n_2$, and $n_2\{n_1 \leftrightarrow n_2\} \sim_\phi n_1 = n$.

That is, $n \sim_\phi (n \bullet \{n_1 \leftrightarrow n_2\})\{n_1 \leftrightarrow n_2\}$.

Suppose $n = n_2$; similarly, $n \sim_\phi (n \bullet \{n_1 \leftrightarrow n_2\})\{n_1 \leftrightarrow n_2\}$.

Suppose $n \neq n_1$ and $n \neq n_2$; then $n \bullet \{n_1 \leftrightarrow n_2\} = n$, and $n\{n_1 \leftrightarrow n_2\} \sim_\phi n$.

That is, $n \sim_\phi (n \bullet \{n_1 \leftrightarrow n_2\})\{n_1 \leftrightarrow n_2\}$.

So, in all cases $n \sim_\phi (n \bullet \{n_1 \leftrightarrow n_2\})\{n_1 \leftrightarrow n_2\}$, and by transitivity $n \sim_\phi \mathcal{N}_0\{\mathcal{N}_1 \leftrightarrow \mathcal{N}_2\}$.

(1⇐ a) $n \sim_\phi \mathcal{N} \Rightarrow n \vDash \mathcal{N}$; induction on the derivation of $n \sim_\phi \mathcal{N}$.

(EqN Refl) Then $\mathcal{N} = n$, hence $n \vDash \mathcal{N}$.

(EqN Symm) Then $\mathcal{N} \sim_\phi n$; by Ind Hyp (**1⇐ b**), $n \vDash \mathcal{N}$.

(EqN Trans) Then $n \sim_\phi \mathcal{M}$ and $\mathcal{M} \sim_\phi \mathcal{N}$;
by Ind Hyp (**1⇐ a**) $n \vDash \mathcal{M}$, and by Ind Hyp (**2a**) $n \vDash \mathcal{N}$.

Other No other cases apply to the form $n \sim_\phi \mathcal{N}$.

(1⇐ b) $\mathcal{N} \sim_\phi n \Rightarrow n \vDash \mathcal{N}$; induction on the derivation of $\mathcal{N} \sim_\phi n$.

(EqN Refl) Then $\mathcal{N} = n$, hence $n \vDash \mathcal{N}$.

(EqN Symm) Then $n \sim_\phi \mathcal{N}$; by Ind Hyp (**1⇐ a**), $n \vDash \mathcal{N}$.

(EqN Trans) Then $\mathcal{N} \sim_\phi \mathcal{M}$ and $\mathcal{M} \sim_\phi n$;
by Ind Hyp (**1⇐ b**) $n \vDash \mathcal{M}$, and by Ind Hyp (**2b**) $n \vDash \mathcal{N}$.

(EqN ↔ App) Then $\mathcal{N}\{\mathcal{N} \leftrightarrow n\} \sim_\phi n$. Assume $n_0 \vDash \mathcal{N}$ (an n_0 always exists for closed $\mathcal{N}$). Then $n \bullet \{n_0 \leftrightarrow n\} = n_0 \vDash \mathcal{N}$, and by definition of satisfaction, $n \vDash \mathcal{N}\{\mathcal{N} \leftrightarrow n\}$.

(EqN ↔ Id) Then $n\{\mathcal{N} \leftrightarrow \mathcal{N}\} \sim_\phi n$. Assume $n_0 \vDash \mathcal{N}$ (an n_0 always exists for closed $\mathcal{N}$). Then (whether $n = n_0$ or not) we have $n \bullet \{n_0 \leftrightarrow n_0\} = n \vDash n$, and by definition of satisfaction, $n \vDash n\{\mathcal{N} \leftrightarrow \mathcal{N}\}$.

(EqN ↔ Inv) Then $n\{\mathcal{M} \leftrightarrow \mathcal{M}'\}\{\mathcal{M} \leftrightarrow \mathcal{M}'\} \sim_\phi n$. Assume $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ (they always exist for closed $\mathcal{M}$, $\mathcal{M}'$). Then (by cases on n, m, m') we have $n \bullet \{m \leftrightarrow m'\} \bullet \{m \leftrightarrow m'\} = n \vDash n$. By definition of satisfaction, $n \bullet \{m \leftrightarrow m'\} \vDash n\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$ and then again, $n \vDash n\{\mathcal{M} \leftrightarrow \mathcal{M}'\}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.

(EqN ↔ Apart) Then $n\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi n$ with $n\#\mathcal{M}$ and $n\#\mathcal{M}'$. Assume $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ (they always exist for closed $\mathcal{M}$, $\mathcal{M}'$). By Ind Hyp (3b), $m \neq n$ and $m' \neq n$. Then we have that $n\bullet(m\leftrightarrow m') = n \vDash n$, and by definition of satisfaction, $n \vDash n\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$.

Other No other cases apply to the form $\mathcal{N} \sim_\phi n$.

(2a) $n \vDash \mathcal{M}$ and $\mathcal{M} \sim_\phi \mathcal{N} \Rightarrow n \vDash \mathcal{N}$; induction on the derivation of $\mathcal{M} \sim_\phi \mathcal{N}$.

(EqN Refl) Then $n \vDash \mathcal{N}$ and $\mathcal{N} \sim_\phi \mathcal{N}$, hence $n \vDash \mathcal{N}$.

(EqN Symm) Then $n \vDash \mathcal{M}$ and $\mathcal{N} \sim_\phi \mathcal{M}$; and by Ind Hyp (**2b**) $n \vDash \mathcal{N}$.

(EqN Trans) Then $n \vDash \mathcal{M}$ and $\mathcal{M} \sim_\phi \mathcal{P}$ and $\mathcal{P} \sim_\phi \mathcal{N} \Rightarrow n \vDash \mathcal{N}$;
by Ind Hyp (**2a**) $n \vDash \mathcal{P}$, and by Ind Hyp (**2a**) $n \vDash \mathcal{N}$.

(EqN ↔ Congr) Then $n \vDash \mathcal{M}_0\{\mathcal{M}_1\leftrightarrow\mathcal{M}_2\}$ and $\mathcal{M}_0\{\mathcal{M}_1\leftrightarrow\mathcal{M}_2\} \sim_\phi \mathcal{N}_0\{\mathcal{N}_1\leftrightarrow\mathcal{N}_2\}$, with $\mathcal{M}_0 \sim_\phi \mathcal{N}_0$ and $\mathcal{M}_1 \sim_\phi \mathcal{N}_1$ and $\mathcal{M}_2 \sim_\phi \mathcal{N}_2$. By definition, $n_1 \vDash \mathcal{M}_1$ and $n_2 \vDash \mathcal{M}_2$ and $n\bullet(n_1\leftrightarrow n_2\} \vDash \mathcal{M}_0$. By Ind Hyp (**2a**) $n\bullet(n_1\leftrightarrow n_2\} \vDash \mathcal{N}_0$ and $n_1 \vDash \mathcal{N}_1$ and $n_2 \vDash \mathcal{N}_2$. Hence $n \vDash \mathcal{N}_0\{\mathcal{N}_1\leftrightarrow\mathcal{N}_2\}$ by definition.

(EqN ↔ App) Then $n \vDash \mathcal{N}\{\mathcal{N}\leftrightarrow\mathcal{M}\}$ and $\mathcal{N}\{\mathcal{N}\leftrightarrow\mathcal{M}\} \sim_\phi \mathcal{M}$.
By definition, $n_1 \vDash \mathcal{N}$ and $n_2 \vDash \mathcal{M}$ and $n\bullet(n_1\leftrightarrow n_2\} \vDash \mathcal{N}$. By Lemma 10-1, $n_1 = n\bullet(n_1\leftrightarrow n_2\}$, hence $n_1\bullet(n_1\leftrightarrow n_2\} = n$, that is, $n_2 = n$. So, $n \vDash \mathcal{M}$.

(EqN ↔ Id) Then $n \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}\}$ and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}\} \sim_\phi \mathcal{N}$.
By definition, $n_1 \vDash \mathcal{M}$ and $n_2 \vDash \mathcal{M}$ and $n\bullet(n_1\leftrightarrow n_2\} \vDash \mathcal{N}$. By Lemma 10-1, $n_1 = n_2$, hence $n\bullet(n_1\leftrightarrow n_2\} = n$. So, $n \vDash \mathcal{N}$.

(EqN ↔ Symm) Then $n \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$ and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}\{\mathcal{M}'\leftrightarrow\mathcal{M}\}$.
By definition, $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}$. Then also $n\bullet(m'\leftrightarrow m\} \vDash \mathcal{N}$, and by definition, $n \vDash \mathcal{N}\{\mathcal{M}'\leftrightarrow\mathcal{M}\}$.

(EqN ↔ Inv) Then $n \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$ and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}$.
By definition, $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$. Then again, $m_1 \vDash \mathcal{M}$ and $m_1' \vDash \mathcal{M}'$ and $n\bullet(m\leftrightarrow m')\bullet(m_1\leftrightarrow m_1'\} \vDash \mathcal{N}$. By Lemma 10-1, $m = m_1$, and $m' = m_1'$. So, $n\bullet(m\leftrightarrow m')\bullet(m\leftrightarrow m') = n \vDash \mathcal{N}$.

(EqN ↔ ↔) Then $n \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}$
and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}\{\mathcal{P}\leftrightarrow\mathcal{P}'\} \sim_\phi \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\{\mathcal{M}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\leftrightarrow\mathcal{M}'\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\}$.
By definition, for some p,p', $p \vDash \mathcal{P}$ and $p' \vDash \mathcal{P}'$ and $n\bullet(p\leftrightarrow p') \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$. Then again for some m,m', $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $n\bullet(p\leftrightarrow p')\bullet(m\leftrightarrow m') \vDash \mathcal{N}$. By Lemma 2-3, $n\bullet(p\leftrightarrow p')\bullet(m\leftrightarrow m') = n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\bullet(p\leftrightarrow p') \vDash \mathcal{N}$. Hence, by definition, $n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\} \vDash \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}$. Now, $m\bullet(p\leftrightarrow p') \vDash \mathcal{M}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}$ and $m'\bullet(p\leftrightarrow p') \vDash \mathcal{M}'\{\mathcal{P}\leftrightarrow\mathcal{P}'\}$, hence, by definition, $n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\} \vDash \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\{\mathcal{M}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\leftrightarrow\mathcal{M}'\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\}$. We have $n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\} = n$. So we conclude $n \vDash \mathcal{N}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\{\mathcal{M}\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\leftrightarrow\mathcal{M}'\{\mathcal{P}\leftrightarrow\mathcal{P}'\}\}$.

(EqN ↔ Apart) Then $n \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$ and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \sim_\phi \mathcal{N}$, with $n\#\mathcal{M}$ and $n\#\mathcal{M}'$.
By definition, $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$. Since $n \vDash n$, by Ind Hyp (3a) we have $n\neq m$ and $n\neq m'$. Then, $n\bullet(m\leftrightarrow m') = n$, and $n \vDash \mathcal{N}$.

(2b) $n \vDash \mathcal{M}$ and $\mathcal{N} \sim \mathcal{M} \Rightarrow n \vDash \mathcal{N}$; induction on the derivation of $\mathcal{N} \sim_\phi \mathcal{M}$.

(EqN ↔ App) Then $n \vDash \mathcal{M}$ and $\mathcal{N}\{\mathcal{N}\leftrightarrow\mathcal{M}\} \sim_\phi \mathcal{M}$.
Take $n_1 \vDash \mathcal{N}$ (one always exists for $\mathcal{N}$ closed). Since $n_1 = n\bullet(n_1\leftrightarrow n)$, we have $n\bullet(n_1\leftrightarrow n\} \vDash \mathcal{N}$. By definition, $n \vDash \mathcal{N}\{\mathcal{N}\leftrightarrow\mathcal{M}\}$.

(EqN ↔ Id) Then $n \vDash \mathcal{N}$ and $\mathcal{N}\{\mathcal{M}\leftrightarrow\mathcal{M}\} \sim_\phi \mathcal{N}$.

Take $n_1 \vDash \mathcal{M}$ (one always exists for $\mathcal{M}$ closed). Since $n\bullet(n_1\leftrightarrow n_1) = n$, we have $n\bullet(n_1\leftrightarrow n_1) \vDash \mathcal{N}$. By definition, $n \vDash \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M})$.

(EqN ↔ Inv) Then $n \vDash \mathcal{N}$ and $\mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{N}$.

Take $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ (they always exists for $\mathcal{M},\mathcal{M}'$ closed). Since $n\bullet(m\leftrightarrow m')\bullet(m\leftrightarrow m') = n$, we have $n\bullet(m\leftrightarrow m')\bullet(m\leftrightarrow m') \vDash \mathcal{N}$. By definition, $n \vDash \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')(\mathcal{M}\leftrightarrow\mathcal{M}')$.

(EqN ↔ ↔) Then $n \vDash \mathcal{N}(\mathcal{P}\leftrightarrow\mathcal{P}')(\mathcal{M}(\mathcal{P}\leftrightarrow\mathcal{P}')\leftrightarrow\mathcal{M}'(\mathcal{P}\leftrightarrow\mathcal{P}')]$
and $\mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')(\mathcal{P}\leftrightarrow\mathcal{P}') \sim_\phi \mathcal{N}(\mathcal{P}\leftrightarrow\mathcal{P}')(\mathcal{M}(\mathcal{P}\leftrightarrow\mathcal{P}')\leftrightarrow\mathcal{M}'(\mathcal{P}\leftrightarrow\mathcal{P}')]$.

By definition, for some q,q', $q \vDash \mathcal{M}(\mathcal{P}\leftrightarrow\mathcal{P}')$ and $q' \vDash \mathcal{M}'(\mathcal{P}\leftrightarrow\mathcal{P}')$ and $n\bullet(q\leftrightarrow q') \vDash \mathcal{N}(\mathcal{P}\leftrightarrow\mathcal{P}')$. Then for some p_1,p_1', $p_1 \vDash \mathcal{P}$ and $p_1' \vDash \mathcal{P}'$ and $m \triangleq q\bullet(p_1\leftrightarrow p_1') \vDash \mathcal{M}$. Similarly, for some p_2,p_2', $p_2 \vDash \mathcal{P}$ and $p_2' \vDash \mathcal{P}'$ and $m' \triangleq q'\bullet(p_2\leftrightarrow p_2') \vDash \mathcal{M}'$. Then again for some p,p', $p \vDash \mathcal{P}$ and $p' \vDash \mathcal{P}'$ and $n\bullet(q\leftrightarrow q')\bullet(p\leftrightarrow p') \vDash \mathcal{N}$. By Lemma 10-1, $p = p_1 = p_2$, and $p' = p_1' = p_2'$. Since $q = m\bullet(p\leftrightarrow p')$ and $q' = m'\bullet(p\leftrightarrow p')$, we have $n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\bullet(p\leftrightarrow p') \vDash \mathcal{N}$.

By Lemma 2-3, $n\bullet(m\bullet(p\leftrightarrow p')\leftrightarrow m'\bullet(p\leftrightarrow p'))\bullet(p\leftrightarrow p') = n\bullet(p\leftrightarrow p')\bullet(m\leftrightarrow m') \vDash \mathcal{N}$. Hence, by definition $n\bullet(p\leftrightarrow p') \vDash \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')$, and again by definition $n \vDash \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')(\mathcal{P}\leftrightarrow\mathcal{P}')$.

(EqN ↔ Apart) Then $n \vDash \mathcal{N}$ and $\mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}') \sim_\phi \mathcal{N}$, with $n\#\mathcal{M}$ and $n\#\mathcal{M}'$.

Take $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ (they always exists for $\mathcal{M},\mathcal{M}'$ closed). Since $n \vDash n$, by Ind Hyp (3a) we have $n\neq m$ and $n\neq m'$. Then, $n = n\bullet(m\leftrightarrow m')$, and $n\bullet(m\leftrightarrow m') \vDash \mathcal{N}$. By definition, $n \vDash \mathcal{N}(\mathcal{M}\leftrightarrow\mathcal{M}')$.

Others Symmetric to (**2a**).

(3a) $n \vDash \mathcal{N}$ and $\mathcal{N} \#_\phi \mathcal{M}$ and $m \vDash \mathcal{M}$ imply $n\neq m$; induction on the derivation of $\mathcal{N} \#_\phi \mathcal{M}$.

(Apart Symm) We have $n \vDash \mathcal{N}$ and $\mathcal{N} \#_\phi \mathcal{M}$ and $m \vDash \mathcal{M}$, with $\mathcal{M} \#_\phi \mathcal{N}$.

By Ind Hyp (3b), $n\neq m$.

(Apart Names) We have $n \vDash p$ and $p \#_\phi q$ and $m \vDash q$, with $p \neq q$.

Then $n = p$ and $m = q$; hence, $n \neq m$.

(Apart Name Var) Impossible because $\mathcal{M}$ must be closed.

(Apart Vars) Impossible because $\mathcal{M}$ and $\mathcal{N}$ must be closed.

(Apart Congr) We have $n \vDash \mathcal{N}(\mathcal{P}\leftrightarrow\mathcal{Q})$ and $\mathcal{N}(\mathcal{P}\leftrightarrow\mathcal{Q}) \#_\phi \mathcal{M}(\mathcal{P}'\leftrightarrow\mathcal{Q}')$ and $m \vDash \mathcal{M}(\mathcal{P}'\leftrightarrow\mathcal{Q}')$, with $\mathcal{N} \#_\phi \mathcal{M}$ and $\mathcal{P}\sim_\phi\mathcal{P}'$ and $\mathcal{Q}\sim_\phi\mathcal{Q}'$. By definition of satisfaction, for some p,q, we have $n\bullet(p\leftrightarrow q) \vDash \mathcal{N}$ and $p \vDash \mathcal{P}$ and $q \vDash \mathcal{Q}$, and for some p',q', we have $m\bullet(p'\leftrightarrow q') \vDash \mathcal{M}$ and $p' \vDash \mathcal{P}'$ and $q' \vDash \mathcal{Q}'$.

By Ind Hyp (2) we have $p \vDash \mathcal{P}'$ and $q \vDash \mathcal{Q}'$. By Lemma 10-1, $p=p'$ and $q=q'$. By Ind Hyp, $n\bullet(p\leftrightarrow q) \neq m\bullet(p\leftrightarrow q)$, which implies $n\neq m$.

(Apart Equiv) We have $n \vDash \mathcal{N}$ and $\mathcal{N} \#_\phi \mathcal{M}$ and $m \vDash \mathcal{M}$,
with $\mathcal{N} \sim_\phi \mathcal{N}'$ and $\mathcal{N}' \#_\phi \mathcal{M}'$ and $\mathcal{M}' \#_\phi \mathcal{M}$.

By Ind Hyp (2a) $n \vDash \mathcal{N}'$. By Ind Hyp (2b) $m \vDash \mathcal{M}'$. By Ind Hyp (3a) $n\neq m$.

(3b) $n \vDash \mathcal{N}$ and $\mathcal{M} \#_\phi \mathcal{N}$ and $m \vDash \mathcal{M}$ imply $n\neq m$; induction on the derivation of $\mathcal{M} \#_\phi \mathcal{N}$.

All Symmetric to (**3a**).

□

Proof of 6-9 Lemma: Satisfaction Under Name Expression Substitution. (p.14)

(1) $n \vDash \mathcal{N}\{x\leftarrow\mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $n \vDash \mathcal{N}\{x\leftarrow\mathcal{M}'\}$

(2) $P \vDash \mathcal{A}\{x\leftarrow\mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $P \vDash \mathcal{A}\{x\leftarrow\mathcal{M}'\}$

(3) $F \models \mathcal{F}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $F \models \mathcal{F}\{x \leftarrow \mathcal{M}'\}$

Proof

(1) $n \models \mathcal{N}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ imply $n \models \mathcal{N}\{x \leftarrow \mathcal{M}'\}$.
Induction on the structure of $\mathcal{N}$.

Case x**.** We have $n \models x\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$;
by Proposition 6-8 we have $n \models x\{x \leftarrow \mathcal{M}'\}$.

Case $y \neq x$**.** Impossible because $y\{x \leftarrow \mathcal{M}\}$ is not closed.

Case n**.** We have $n\{x \leftarrow \mathcal{M}\} = n = n\{x \leftarrow \mathcal{M}'\}$.

Case $m \neq n$**.** Impossible because $n \models m$ does not hold.

Case $\mathcal{N}_0(\mathcal{N}_1 \leftrightarrow \mathcal{N}_2)$**.** We have $n \models \mathcal{N}_0(\mathcal{N}_1 \leftrightarrow \mathcal{N}_2)\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$.
That is $n \models \mathcal{N}_0\{x \leftarrow \mathcal{M}\}(\mathcal{N}_1\{x \leftarrow \mathcal{M}\} \leftrightarrow \mathcal{N}_2\{x \leftarrow \mathcal{M}\})$. By def. of satisfaction, $n = n_0 \bullet (n_1 \leftrightarrow n_2)$ and $n_0 \models \mathcal{N}_0\{x \leftarrow \mathcal{M}\}$ and $n_1 \models \mathcal{N}_1\{x \leftarrow \mathcal{M}\}$ and $n_2 \models \mathcal{N}_2\{x \leftarrow \mathcal{M}\}$. By Ind Hyp $n_0 \models \mathcal{N}_0\{x \leftarrow \mathcal{M}'\}$ and $n_1 \models \mathcal{N}_1\{x \leftarrow \mathcal{M}'\}$ and $n_2 \models \mathcal{N}_2\{x \leftarrow \mathcal{M}'\}$.
Hence $n \models \mathcal{N}_0\{x \leftarrow \mathcal{M}'\}(\mathcal{N}_1\{x \leftarrow \mathcal{M}'\} \leftrightarrow \mathcal{N}_2\{x \leftarrow \mathcal{M}'\})$,
that is $n \models \mathcal{N}_0(\mathcal{N}_1 \leftrightarrow \mathcal{N}_2)\{x \leftarrow \mathcal{M}'\}$.

(2) $P \models \mathcal{A}\{x \leftarrow \mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) implies $P \models \mathcal{A}\{x \leftarrow \mathcal{M}'\}$.
Induction on the structure of $\mathcal{A}$.

Case 0. $\mathbf{0}\{x \leftarrow \mathcal{M}\} = \mathbf{0} = \{x \leftarrow \mathcal{M}'\}$.

Case $\mathcal{N}[\mathcal{A}]$**.** We have $P \models \mathcal{N}[\mathcal{A}]\{x \leftarrow \mathcal{M}\}$, that is $P \models \mathcal{N}\{x \leftarrow \mathcal{M}\}[\mathcal{A}\{x \leftarrow \mathcal{M}\}]$,
that is $P \equiv n[P']$ and $n \models \mathcal{N}\{x \leftarrow \mathcal{M}\}$ and $P' \models \mathcal{A}\{x \leftarrow \mathcal{M}\}$.
By (1) we have $n \models \mathcal{N}\{x \leftarrow \mathcal{M}'\}$. By Ind Hyp we have $P' \models \mathcal{A}\{x \leftarrow \mathcal{M}'\}$.
Hence $P \models \mathcal{N}\{x \leftarrow \mathcal{M}'\}[\mathcal{A}\{x \leftarrow \mathcal{M}'\}]$, that is $P \models \mathcal{N}[\mathcal{A}]\{x \leftarrow \mathcal{M}'\}$.

Case $\mathsf{H}y.\mathcal{A}$**.** We have $P \models (\mathsf{H}y.\mathcal{A})\{x \leftarrow \mathcal{M}\}$, choose $y' \notin fv(y,\mathcal{A},x,\mathcal{M},\mathcal{M}')$,
by Definition 3-3 $P \models \mathsf{H}y'.\mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\}$,
that is $P \equiv (\nu n_0)P_0$ and $n_0 \notin na(\mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\})$
and $P_0 \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\}\{y' \leftarrow n_0\}$.
Since n_0 might occur in $\mathcal{M}$ (if $x \notin fv(\mathcal{A})$) or in $\mathcal{M}'$, choose $n \notin na(\mathcal{A},\mathcal{M},\mathcal{M}',n_0,P_0)$;
By Lemma 4-3, $P_0 \bullet (n_0 \leftrightarrow n) \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\}\{y' \leftarrow n_0\} \bullet (n_0 \leftrightarrow n)$.
Let $P' = P_0 \bullet (n_0 \leftrightarrow n)$, so that $(\nu n)P' \equiv_\alpha (\nu n_0)P_0$;
then $P' \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\}\{y' \leftarrow n_0\} \bullet (n_0 \leftrightarrow n)$ from above,
that is $P' \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\}\{y' \leftarrow n\}$ since $n, n_0 \notin na(\mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}\})$,
that is $P' \models \mathcal{A}\{y \leftarrow y'\}\{y' \leftarrow n\}\{x \leftarrow \mathcal{M}\}$ since $n, \mathcal{M}$ are closed and $y' \neq x$.
By Ind Hyp (since $\mathcal{A}\{y \leftarrow y'\}\{y' \leftarrow n\}$ has the same size as $\mathcal{A}$)
we obtain $P' \models \mathcal{A}\{y \leftarrow y'\}\{y' \leftarrow n\}\{x \leftarrow \mathcal{M}'\}$,
that is $P' \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}'\}\{y' \leftarrow n\}$ since $n, \mathcal{M}'$ are closed and $y' \neq x$.
We have now obtained that $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}'\})$
and $P' \models \mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}'\}\{y' \leftarrow n\}$;
therefore $P \models \mathsf{H}y'.\mathcal{A}\{y \leftarrow y'\}\{x \leftarrow \mathcal{M}'\}$ by definition of satisfaction,
that is $P \models (\mathsf{H}y.\mathcal{A})\{x \leftarrow \mathcal{M}'\}$ by Definition 3-3.

Case $\copyright\mathcal{N}$**.** We have $P \models (\copyright\mathcal{N})\{x \leftarrow \mathcal{M}\}$, that is $P \models \copyright(\mathcal{N}\{x \leftarrow \mathcal{M}\})$,
that is $n \models \mathcal{N}\{x \leftarrow \mathcal{M}\}$ and $n \in fn(P)$.
By (1), $n \models \mathcal{N}\{x \leftarrow \mathcal{M}'\}$, hence $P \models \copyright(\mathcal{N}\{x \leftarrow \mathcal{M}'\})$, that is $P \models (\copyright\mathcal{N})\{x \leftarrow \mathcal{M}'\}$.

Case $\mathcal{A}[\mathcal{N} \leftrightarrow \mathcal{N}']$**.** We have $P \models (\mathcal{A}[\mathcal{N} \leftrightarrow \mathcal{N}'])\{x \leftarrow \mathcal{M}\}$,

that is $P \vDash (\mathcal{A}\{x\leftarrow\mathcal{M}\}\{\mathcal{N}\{x\leftarrow\mathcal{M}\}\leftrightarrow\mathcal{N}'\{x\leftarrow\mathcal{M}\}\})$,
that is $n \vDash \mathcal{N}\{x\leftarrow\mathcal{M}\}$ and $n' \vDash \mathcal{N}'\{x\leftarrow\mathcal{M}\}$ and $P\bullet(n\leftrightarrow n') \vDash \mathcal{A}\{x\leftarrow\mathcal{M}\}$.
By (1) we have $n \vDash \mathcal{N}\{x\leftarrow\mathcal{M}'\}$ have $n' \vDash \mathcal{N}'\{x\leftarrow\mathcal{M}'\}$.
By Ind Hyp we have $P\bullet(n\leftrightarrow n') \vDash \mathcal{A}\{x\leftarrow\mathcal{M}'\}$.
Hence $P \vDash (\mathcal{A}\{x\leftarrow\mathcal{M}'\}\{\mathcal{N}\{x\leftarrow\mathcal{M}'\}\leftrightarrow\mathcal{N}'\{x\leftarrow\mathcal{M}'\}\})$,
that is $P \vDash (\mathcal{A}\{\mathcal{N}\leftrightarrow\mathcal{N}'\})\{x\leftarrow\mathcal{M}'\}$.

Cases $(\mathcal{A} \mid \mathcal{B})$, **F**, $(\mathcal{A} \wedge \mathcal{B})$, $(\mathcal{A} \Rightarrow \mathcal{B})$ Routine.

(3) $F \vDash \mathcal{T}\{x\leftarrow\mathcal{M}\}$ and $\mathcal{M} \sim_\phi \mathcal{M}'$ ($\mathcal{M}'$ closed) imply $F \vDash \mathcal{T}\{x\leftarrow\mathcal{M}'\}$.
Induction on the structure of $\mathcal{T}$.

Case $\mathcal{A}$**.** From (2).

Case N. $F \vDash \mathbf{N}\{x\leftarrow\mathcal{M}\} = \mathbf{N} = \{x\leftarrow\mathcal{M}'\}$.

Case $\mathcal{T}\rightarrow\mathcal{G}$ with $\mathcal{T}\neq$**N.** We have $H \vDash (\mathcal{T}\rightarrow\mathcal{G})\{x\leftarrow\mathcal{M}\}$,
that is $H \vDash \mathcal{T}\{x\leftarrow\mathcal{M}\}\rightarrow\mathcal{G}\{x\leftarrow\mathcal{M}\}$.
This means that $H = \langle\rho, x, t\rangle$
and $\forall F,G.\ (F \vDash_H \mathcal{T}\{x\leftarrow\mathcal{M}\} \wedge t \Downarrow_{\rho[x\leftarrow F]} G) \Rightarrow G \vDash_H \mathcal{G}\{x\leftarrow\mathcal{M}\}$.
Take any F,G, and assume $F \vDash_H \mathcal{T}\{x\leftarrow\mathcal{M}'\} \wedge t \Downarrow_{\rho[x\leftarrow F]} G$; by Ind Hyp (which is qantified over all $\mathcal{M}',\mathcal{M}$), $F \vDash_H \mathcal{T}\{x\leftarrow\mathcal{M}\}$. Then by assumption, $G \vDash_H \mathcal{G}\{x\leftarrow\mathcal{M}\}$, and by Ind Hyp, $G \vDash_H \mathcal{G}\{x\leftarrow\mathcal{M}'\}$.
We have shown that $\forall F,G.\ (F \vDash_H \mathcal{T}\{x\leftarrow\mathcal{M}'\} \wedge t \Downarrow_{\rho[x\leftarrow F]} G) \Rightarrow G \vDash_H \mathcal{G}\{x\leftarrow\mathcal{M}'\}$,
that is $H \vDash \mathcal{T}\{x\leftarrow\mathcal{M}'\}\rightarrow\mathcal{G}\{x\leftarrow\mathcal{M}'\}$, that is $H \vDash (\mathcal{T}\rightarrow\mathcal{G})\{x\leftarrow\mathcal{M}'\}$.

Case $\Pi y.\ \mathcal{G}$ **.** We have $H \vDash (\Pi y.\mathcal{G})\{x\leftarrow\mathcal{M}\}$. Choose $y' \notin fv(y,\mathcal{G},x,\mathcal{M},\mathcal{M}')$.
Then $H \vDash \Pi y'.\mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}\}$ for $y' \notin fv(y,\mathcal{G},x,\mathcal{M})$.
This means that $H = \langle\rho, z, t\rangle$
and $\forall n,G.\ t \Downarrow_{\rho[z\leftarrow n]} G \Rightarrow G \vDash_H \mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}\}\{y'\leftarrow n\}$.
Take any n,G, and assume $t \Downarrow_{\rho[z\leftarrow n]} G$;
by assumption, $G \vDash_H \mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}\}\{y'\leftarrow n\}$, that is $G \vDash_H \mathcal{G}\{y\leftarrow n\}\{x\leftarrow\mathcal{M}\}$.
By Ind Hyp (since $\mathcal{G}\{y\leftarrow n\}$ has the same size as $\mathcal{G}$) $G \vDash_H \mathcal{G}\{y\leftarrow n\}\{x\leftarrow\mathcal{M}'\}$,
that is $G \vDash_H \mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}'\}\{y'\leftarrow n\}$.
We have shown that $\forall n,G.\ t \Downarrow_{\rho[z\leftarrow n]} G \Rightarrow G \vDash_H \mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}'\}\{y'\leftarrow n\}$,
that is $H \vDash \Pi y'.\mathcal{G}\{y\leftarrow y'\}\{x\leftarrow\mathcal{M}'\}$, that is $H \vDash (\Pi y.\mathcal{G})\{x\leftarrow\mathcal{M}'\}$.

□

Proof of 6-11 Proposition: Soundness of Closed Type Equivalence. (p.15) (Part I)
If $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{B}$ ($\mathcal{B}$ closed) then $P \vDash \mathcal{B}$.

Proof

We actually prove:
(1) $\forall\phi.\ P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{B}$ ($\mathcal{B}$ closed) then $P \vDash \mathcal{B}$
(2) $\forall\phi.\ P \vDash \mathcal{A}$ and $\mathcal{B} \sim_\phi \mathcal{A}$ ($\mathcal{B}$ closed) then $P \vDash \mathcal{B}$
by mutual induction on the size of derivations of $\mathcal{A} \sim_\phi \mathcal{B}$ and $\mathcal{B} \sim_\phi \mathcal{A}$,
where we ignore the size of name expression derivations (see case (EqT ⊟ Congr)).

(1)

Case (EqT Refl) Then $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{A}$; trivial.

Case (EqT Symm) Then $P \vDash \mathcal{A}$ and $\mathcal{B} \sim_\phi \mathcal{A}$; by Ind Hyp (2), $P \vDash \mathcal{B}$.

Case (EqT Trans) Then $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi C$ and $C \sim_\phi \mathcal{B}$;
by Ind Hyp (1), $P \vDash C$, and by Ind Hyp (1), $P \vDash \mathcal{B}$.

Case (EqT $\mathcal{N}$[] Congr) Then $P \vDash \mathcal{N}[\mathcal{A}]$ and $\mathcal{N}[\mathcal{A}] \sim_\phi \mathcal{N}'[\mathcal{B}]$, with $\mathcal{N} \sim_\phi \mathcal{N}'$ and $\mathcal{A} \sim_\phi \mathcal{B}$. Then $P \vDash \mathcal{N}[\mathcal{A}]$ holds iff $\exists n,P'.\ n \vDash \mathcal{N}$ and $P \equiv n[P']$ and $P' \vDash \mathcal{A}$. By Ind Hyp (1), $P' \vDash \mathcal{B}$. By Proposition 6-8, $n \vDash \mathcal{N}'$. Hence $P \vDash \mathcal{N}'[\mathcal{B}]$.

Case (EqT | Congr) ... (EqT $\Rightarrow$ Congr), (EqT © Congr) Similar.

Case (EqT H Congr) $\forall \phi.\ P \vDash \mathsf{H}x.\mathcal{A}$ and $\mathsf{H}x.\mathcal{A} \sim_\phi \mathsf{H}x.\mathcal{B}$ ($\mathsf{H}x.\mathcal{B}$ closed) then $P \vDash \mathsf{H}x.\mathcal{B}$, where $\mathsf{H}x.\mathcal{A} \sim_\phi \mathsf{H}x.\mathcal{B}$ comes from $\mathcal{A} \sim_{(\phi,\mathsf{H}x)} \mathcal{B}$.
Then $P \vDash \mathsf{H}x.\mathcal{A}$ holds iff $P \equiv (\nu n_0)P_0$ and $n_0 \notin na(\mathcal{A})$ and $P_0 \vDash \mathcal{A}\{x \leftarrow n_0\}$.
Since n_0 might occur in $\mathcal{B}$, choose $n \notin na(\mathcal{A},\mathcal{B},n_0,P_0,\phi)$;
by Lemma 4-3 $P_0 \bullet (n_0 \leftrightarrow n] \vDash \mathcal{A}\{x \leftarrow n_0\} \bullet (n_0 \leftrightarrow n]$.
Let $P' = P_0 \bullet (n_0 \leftrightarrow n]$, so that $(\nu n)P' \equiv_\alpha (\nu n_0)P_0 \equiv P$, and $P' \vDash \mathcal{A}\{x \leftarrow n_0\} \bullet (n_0 \leftrightarrow n]$, that is $P' \vDash \mathcal{A}\{x \leftarrow n\}$, since $n,n_0 \notin na(\mathcal{A})$.
Take any $\varepsilon \vDash \phi$; since $x \notin dom(\phi)$ and $n \notin na(\phi)$ we have $\varepsilon[x \leftarrow n] \vDash \phi, \mathsf{H}x$, by Definition 6-4. By Proposition 6-6, $\mathcal{A} \sim_{(\phi,\mathsf{H}x)} \mathcal{B}$ implies $\varepsilon[x \leftarrow n](\mathcal{A}) \sim_{\varepsilon[x \leftarrow n](\phi,\mathsf{H}x)} \varepsilon[x \leftarrow n](\mathcal{B})$, which is the same as $\mathcal{A}\{x \leftarrow n\} \sim_{(\varepsilon(\phi),n)} \mathcal{B}\{x \leftarrow n\}$.
By Ind Hyp (since $\mathcal{A}\{x \leftarrow n\} \sim_{(\varepsilon(\phi),n)} \mathcal{B}\{x \leftarrow n\}$ has the same size as $\mathcal{A} \sim_{(\phi,\mathsf{H}x)} \mathcal{B}$), we obtain $P' \vDash \mathcal{B}\{x \leftarrow n\}$.
Since $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{B})$, we conclude $P \vDash \mathsf{H}x.\mathcal{B}$.

Case (EqT 0 $\leftrightarrow$) $\mathbf{0}\{\mathcal{M} \leftrightarrow \mathcal{M}'\} \sim_\phi \mathbf{0}$. Then $P \vDash \mathbf{0}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.
$P \vDash \mathbf{0}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$ holds iff $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $P \bullet (m \leftrightarrow m'] \vDash \mathbf{0}$,
that is iff $P \bullet (m \leftrightarrow m'] \equiv 0$ iff $P \equiv 0 \bullet (m \leftrightarrow m']$ iff $P \equiv 0$, hence $P \vDash \mathbf{0}$.

Case (EqT $\mathcal{N}$[] $\leftrightarrow$) $\mathcal{N}[\mathcal{A}]\{\mathcal{M} \leftrightarrow \mathcal{M}'\} \sim_\phi \mathcal{N}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}[\mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}]$.
Then $P \vDash \mathcal{N}[\mathcal{A}]\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.
$P \vDash \mathcal{N}[\mathcal{A}]\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$ holds iff $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $P \bullet (m \leftrightarrow m'] \vDash \mathcal{N}[\mathcal{A}]$,
that is iff $P \bullet (m \leftrightarrow m'] \equiv n[P']$ and $n \vDash \mathcal{N}$ and $P' \vDash \mathcal{A}$.
We obtain that $n \bullet (m \leftrightarrow m'] \vDash \mathcal{N}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.
Since $P' \vDash \mathcal{A}$, we have $P' \bullet (m \leftrightarrow m'] \vDash \mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$ by Lemma 10-2.
Since $P \bullet (m \leftrightarrow m'] \equiv n[P']$ we have $P \equiv n[P'] \bullet (m \leftrightarrow m']$,
that is $P \equiv n \bullet (m \leftrightarrow m')[P' \bullet (m \leftrightarrow m')]$.
Therefore, $P \vDash \mathcal{N}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}[\mathcal{A}\{\mathcal{M} \leftrightarrow \mathcal{M}'\}]$ by definition of satisfaction.

Case (EqT | $\leftrightarrow$) ... (EqT $\Rightarrow$ Congr), (EqT © $\leftrightarrow$) Similar.

Case (EqT $\leftrightarrow$ $\leftrightarrow$) Similar; see also case (EqN $\leftrightarrow$ $\leftrightarrow$) in Proposition 6-8.

Case (EqT H $\leftrightarrow$) $(\mathsf{H}x.\mathcal{A})\{\mathcal{M} \leftrightarrow \mathcal{M}'\} \sim_\phi \mathsf{H}x.(\mathcal{A}\{x \leftarrow x\{\mathcal{M} \leftrightarrow \mathcal{M}'\}\}\{\mathcal{M} \leftrightarrow \mathcal{M}'\})$, with $x \notin fv(\mathcal{M},\mathcal{M}')$. (N.B.: $\mathsf{H}x.(\mathcal{A}\{x \leftarrow x\{\mathcal{M} \leftrightarrow \mathcal{M}'\}\})$ closed and $x \notin fv(\mathcal{M},\mathcal{M}')$ imply $\mathcal{M},\mathcal{M}'$ closed.) Then $P \vDash (\mathsf{H}x.\mathcal{A})\{\mathcal{M} \leftrightarrow \mathcal{M}'\}$.
This holds iff some $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $P \bullet (m \leftrightarrow m'] \vDash \mathsf{H}x.\mathcal{A}$
which holds iff $P \bullet (m \leftrightarrow m'] \equiv (\nu n_0)P_0$ and $n_0 \notin na(\mathcal{A})$ and $P_0 \vDash \mathcal{A}\{x \leftarrow n_0\}$.
Since n_0 might occur in $\mathcal{M},\mathcal{M}'$, choose $n \notin na(\mathcal{A},\mathcal{M},\mathcal{M}',n_0,P_0,m,m')$;
by Lemma 4-3 $P_0 \bullet (n_0 \leftrightarrow n] \vDash \mathcal{A}\{x \leftarrow n_0\} \bullet (n_0 \leftrightarrow n]$.
Let $P' = P_0 \bullet (n_0 \leftrightarrow n]$, so that $(\nu n)P' \equiv_\alpha (\nu n_0)P_0 \equiv P \bullet (m \leftrightarrow m']$,
and $P' \vDash \mathcal{A}\{x \leftarrow n\}$ since $n,n_0 \notin na(\mathcal{A})$.
By Proposition 6-8 (and Lemma 6-7(1)), we have $m \sim_\phi \mathcal{M}$ and $m' \sim_\phi \mathcal{M}'$.

Since $n \notin na(\mathcal{M},\mathcal{N})$, by Lemma 6-7 we have that $n \neq m,m'$.
Then $n \sim_\phi n(m \leftrightarrow m') \sim_\phi n(\mathcal{M} \leftrightarrow \mathcal{N}')$ by congruence.
Hence, by Lemma 6-9, $P' \vDash \mathcal{A}\{x \leftarrow n(\mathcal{M} \leftrightarrow \mathcal{N}')\}$.
That is, since $x \notin fv(\mathcal{M},\mathcal{N}')$, $P' \vDash \mathcal{A}\{x \leftarrow x(\mathcal{M} \leftrightarrow \mathcal{N}')\}\{x \leftarrow n\}$.
Let $\mathcal{A}' = \mathcal{A}\{x \leftarrow x(\mathcal{M} \leftrightarrow \mathcal{N}')\}$, so that $P' \vDash \mathcal{A}'\{x \leftarrow n\}$.
By definition of satisfaction, $P' \bullet (m \leftrightarrow m') \vDash \mathcal{A}'\{x \leftarrow n\}(\mathcal{M} \leftrightarrow \mathcal{N}')$,
and since $x \notin fv(\mathcal{M},\mathcal{N}')$, this is the same as $P' \bullet (m \leftrightarrow m') \vDash \mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}')\{x \leftarrow n\}$.
Since $P \bullet (m \leftrightarrow m') \equiv (\nu n)P'$, we have that $P \equiv ((\nu n)P') \bullet (m \leftrightarrow m')$,
and since $n \neq m,m'$, we have that $P \equiv (\nu n)P' \bullet (m \leftrightarrow m')$.
Moreover, $n \notin na(\mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}'))$.
Therefore, by definition of satisfaction, we obtain that $P \vDash \mathsf{H}x.\mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}')$.

Case (EqT H-α) $\mathsf{H}x.\ \mathcal{A} \sim_\phi \mathsf{H}y.\ \mathcal{A}\{x \leftarrow y\}$ with $y \notin fv(\mathcal{A})$.
Then $P \vDash \mathsf{H}x.\ \mathcal{A}$, which holds iff $P \equiv (\nu n_0)P_0$ and $n_0 \notin na(\mathcal{A})$ and $P_0 \vDash \mathcal{A}\{x \leftarrow n_0\}$.
Since $y \notin fv(\mathcal{A})$ we have that $\mathcal{A}\{x \leftarrow n_0\} = \mathcal{A}\{x \leftarrow y\}\{y \leftarrow n_0\}$, so $P_0 \vDash \mathcal{A}\{x \leftarrow y\}\{y \leftarrow n_0\}$.
Moreover, $n_0 \notin na(\mathcal{A}\{x \leftarrow y\})$, hence $P \vDash \mathsf{H}y.\ \mathcal{A}\{x \leftarrow y\}$.

(2)

Case (EqT Refl) Then $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{A}$; trivial.
Case (EqT Symm) Then $P \vDash \mathcal{A}$ and $\mathcal{A} \sim_\phi \mathcal{B}$; by Ind Hyp (1), $P \vDash \mathcal{B}$.
Case (EqT Trans) Then $P \vDash \mathcal{A}$ and $\mathcal{B} \sim_\phi C$ and $C \sim_\phi \mathcal{A}$;
by Ind Hyp (2), $P \vDash C$, and by Ind Hyp (2), $P \vDash \mathcal{B}$.
Case (EqT $\mathcal{N}$[] Congr) Symmetrical to (1).
Case (EqT | Congr) ... (EqT ⇒ Congr), (EqT © Congr) Similar.
Case (EqT H Congr) $\mathcal{B} \sim_{(\phi,\mathsf{H}x)} \mathcal{A} \Rightarrow \mathsf{H}x.\ \mathcal{B} \sim_\phi \mathsf{H}x.\ \mathcal{A}$**.** Symmetrical to (1).
Case (EqT 0 ↔) $\mathbf{0}(\mathcal{M} \leftrightarrow \mathcal{N}') \sim_\phi \mathbf{0}$. Then $P \vDash \mathbf{0}$.
Assume $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{N}'$ (such m,m' always exist for closed $\mathcal{M},\mathcal{N}'$).
$P \vDash \mathbf{0}$ holds iff $P \equiv 0$, hence $P \bullet (m \leftrightarrow m') \equiv 0 \bullet (m \leftrightarrow m')$, that is $P \bullet (m \leftrightarrow m') \equiv 0$.
Hence $P \bullet (m \leftrightarrow m') \vDash \mathbf{0}$, and by definition of satisfaction, $P \vDash \mathbf{0}(\mathcal{M} \leftrightarrow \mathcal{N}')$.
Case (EqT $\mathcal{N}$[] ↔) $\mathcal{N}[\mathcal{A}](\mathcal{M} \leftrightarrow \mathcal{N}') \sim_\phi \mathcal{N}(\mathcal{M} \leftrightarrow \mathcal{N}')[\mathcal{A}(\mathcal{M} \leftrightarrow \mathcal{N}')]$.
Then $P \vDash \mathcal{N}(\mathcal{M} \leftrightarrow \mathcal{N}')[\mathcal{A}(\mathcal{M} \leftrightarrow \mathcal{N}')]$.
This holds iff $P \equiv n[P']$ and $n \vDash \mathcal{N}(\mathcal{M} \leftrightarrow \mathcal{N}')$ and $P' \vDash \mathcal{A}(\mathcal{M} \leftrightarrow \mathcal{N}')$.
The former holds iff some $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{N}'$ and $n \bullet (m \leftrightarrow m') \vDash \mathcal{N}$.
That latter holds iff some $p \vDash \mathcal{M}$ and $p' \vDash \mathcal{N}'$ and $P' \bullet (p \leftrightarrow p') \vDash \mathcal{A}$.
By Lemma 10-1, $p = m$ and $p' = m'$.
Since $P \equiv n[P']$, we have that $P \bullet (m \leftrightarrow m') \equiv n[P'] \bullet (m \leftrightarrow m')$,
that is $P \bullet (m \leftrightarrow m') \equiv n \bullet (m \leftrightarrow m')[P' \bullet (m \leftrightarrow m')]$.
Then, by definition of satisfaction, $P \bullet (m \leftrightarrow m') \vDash \mathcal{N}[\mathcal{A}]$.
Hence, again by defnition, $P \vDash \mathcal{N}[\mathcal{A}](\mathcal{M} \leftrightarrow \mathcal{N}')$.
Case (EqT | ↔) ... (EqT ⇒ Congr), (EqT © ↔) Similar.
Case (EqT ↔ ↔) Similar; see also case (EqN ↔ ↔) in Proposition 6-8.
Case (EqT H ↔) $(\mathsf{H}x.\mathcal{A})(\mathcal{M} \leftrightarrow \mathcal{N}') \sim_\phi \mathsf{H}x.(\mathcal{A}\{x \leftarrow x(\mathcal{M} \leftrightarrow \mathcal{N}')\}(\mathcal{M} \leftrightarrow \mathcal{N}'))$,
with $x \notin fv(\mathcal{M},\mathcal{N}')$. (N.B.: $\mathsf{H}x.\ (\mathcal{A}\{x \leftarrow x(\mathcal{M} \leftrightarrow \mathcal{N}')\})$ closed and $x \notin fv(\mathcal{M},\mathcal{N}')$ imply $\mathcal{M},\mathcal{N}'$ closed.) Let $\mathcal{A}' = \mathcal{A}\{x \leftarrow x(\mathcal{M} \leftrightarrow \mathcal{N}')\}$; then $P \vDash \mathsf{H}x.\mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}')$.
This holds iff $P \equiv (\nu n)P'$ and $n \notin na(\mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}'))$ and $P' \vDash \mathcal{A}'(\mathcal{M} \leftrightarrow \mathcal{N}')\{x \leftarrow n\}$

that is, since $x \notin fv(\mathcal{M},\mathcal{M}')$, $P' \vDash \mathcal{A}'\{x \leftarrow n\}(\mathcal{M} \leftrightarrow \mathcal{M}')$,
which holds if some $m \vDash \mathcal{M}$ and $m' \vDash \mathcal{M}'$ and $P' \bullet (m \leftrightarrow m') \vDash \mathcal{A}'\{x \leftarrow n\}$,
that is, since $x \notin fv(\mathcal{M},\mathcal{M}')$, $P' \bullet (m \leftrightarrow m') \vDash \mathcal{A}\{x \leftarrow n(\mathcal{M} \leftrightarrow \mathcal{M}')\}$.
Since $n \notin na(\mathcal{M},\mathcal{M}')$ we have $n \neq m,m'$ by Lemma 6-7.
Then $n \sim_\phi n(m \leftrightarrow m') \sim_\phi n(\mathcal{M} \leftrightarrow \mathcal{M}')$ by congruence.
Hence, by Lemma 6-9, $P' \vDash \mathcal{A}\{x \leftarrow n\}$.
Since $P \equiv (\nu n)P'$, we have $P \bullet (m \leftrightarrow m') \equiv ((\nu n)P') \bullet (m \leftrightarrow m')$,
that is $P \bullet (m \leftrightarrow m') \equiv (\nu n)P' \bullet (m \leftrightarrow m')$. Moreover, $n \notin na(\mathcal{A})$.
So, by definition of satisfaction $P \bullet (m \leftrightarrow m') \vDash \mathsf{H}x.\mathcal{A}$.
And again by definition of satisfaction, $P \vDash (\mathsf{H}x.\mathcal{A})(\mathcal{M} \leftrightarrow \mathcal{M}')$.

Case (EqT H-α) $\mathsf{H}x.\ \mathcal{A} \sim_\phi \mathsf{H}y.\ \mathcal{A}\{x \leftarrow y\}$ with $y \notin fv(\mathcal{A})$.
Then $P \vDash \mathsf{H}y.\ \mathcal{A}\{x \leftarrow y\}$, that is $P \equiv (\nu n_0)P_0$ and $n_0 \notin na(\mathcal{A}\{x \leftarrow y\})$
and $P_0 \vDash \mathcal{A}\{x \leftarrow y\}\{y \leftarrow n_0\}$.
Since $y \notin fv(\mathcal{A})$ we have that $\mathcal{A}\{x \leftarrow y\}\{y \leftarrow n_0\} = \mathcal{A}\{x \leftarrow n_0\}$, so $P_0 \vDash \mathcal{A}\{x \leftarrow n_0\}$.
Moreover, $n_0 \notin na(\mathcal{A})$, hence $P \vDash \mathsf{H}x.\ \mathcal{A}$.

□

Proof of 6-11 Proposition: Soundness of Closed Type Equivalence. (p.15) (Part II)
If $F \vDash \mathcal{F}$ and $\mathcal{F} \sim_\phi \mathcal{G}$ ($\mathcal{G}$ closed) then $F \vDash \mathcal{G}$.

Proof

We actually prove:
(1) $\forall\phi$. If $F \vDash \mathcal{F}$ and $\mathcal{F} \sim_\phi \mathcal{G}$ ($\mathcal{G}$ closed) then $F \vDash \mathcal{G}$
(2) $\forall\phi$. If $F \vDash \mathcal{F}$ and $\mathcal{G} \sim_\phi \mathcal{F}$ ($\mathcal{G}$ closed) then $F \vDash \mathcal{G}$
by mutual induction on the derivations of $\mathcal{F} \sim_\phi \mathcal{G}$ and $\mathcal{G} \sim_\phi \mathcal{F}$.

(1)

Cases (EqH Refl) (EqH Symm) (EqH Trans). As in Proposition 6-11.

Case (EqH $\mathcal{A}$ Congr) From Proposition 6-11.

Case (EqH → Congr) $\mathcal{F} \sim_\phi \mathcal{F}' \wedge \mathcal{G} \sim_\phi \mathcal{G}' \Rightarrow \mathcal{F} \rightarrow \mathcal{G} \sim_\phi \mathcal{F}' \rightarrow \mathcal{G}'$
Then $H \vDash \mathcal{F} \rightarrow \mathcal{G}$; this holds iff $H = \langle \rho, x, t \rangle$ and $\forall F,G.\ (F \vDash \mathcal{F} \wedge t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \vDash \mathcal{G}$.
Take any F,G and assume $F \vDash \mathcal{F}' \wedge t \Downarrow_{\rho[x \leftarrow F]} G$. By Ind Hyp, $F \vDash \mathcal{F}$. Then, by assumption, $G \vDash \mathcal{G}$. By Ind Hyp, we obtain $G \vDash \mathcal{G}'$. We have shown that
$\forall F,G.\ (F \vDash \mathcal{F}' \wedge t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \vDash \mathcal{G}'$, that is $\langle \rho, x, t \rangle \vDash \mathcal{F}' \rightarrow \mathcal{G}'$.

Case (EqH Π Congr) $\mathcal{G} \sim_{(\phi,\forall x)} \mathcal{G}' \Rightarrow \Pi x.\ \mathcal{G} \sim_\phi \Pi x.\ \mathcal{G}'$
Then $F \vDash \Pi x.\ \mathcal{G}$; this holds iff $F = \langle \rho, z, t \rangle$ and $\forall n,G.\ t \Downarrow_{\rho[z \leftarrow n]} G \Rightarrow G \vDash \mathcal{G}\{x \leftarrow n\}$.
Take any n,G and assume $t \Downarrow_{\rho[z \leftarrow n]} G$. By assumption, $G \vDash \mathcal{G}\{x \leftarrow n\}$.
Take any $\varepsilon \vDash \phi$; since $x \notin dom(\phi)$, we have $\varepsilon[x \leftarrow n] \vDash (\phi, \forall x)$ by Definition 6-4.
By Proposition 6-6, $\mathcal{G} \sim_{(\phi,\forall x)} \mathcal{G}'$ implies $\varepsilon[x \leftarrow n](\mathcal{G}) \sim_{\varepsilon[x \leftarrow n](\phi,\forall x)} \varepsilon[x \leftarrow n](\mathcal{G}')$, which is the same as $\mathcal{G}\{x \leftarrow n\} \sim_{(\varepsilon(\phi),n)} \mathcal{G}'\{x \leftarrow n\}$. By Ind Hyp (since $\mathcal{G}\{x \leftarrow n\} \sim_{(\varepsilon(\phi),n)} \mathcal{G}'\{x \leftarrow n\}$ has the same size as $\mathcal{G} \sim_{(\phi,\forall x)} \mathcal{G}'$), we obtain $G \vDash \mathcal{G}'\{x \leftarrow n\}$. We have shown that $\forall n,G.\ t \Downarrow_{\rho[z \leftarrow n]} G \Rightarrow G \vDash \mathcal{G}'\{x \leftarrow n\}$, that is $\langle \rho, z, t \rangle \vDash \Pi x.\ \mathcal{G}$.

Case (EqH Π-α) $\Pi x.\ \mathcal{G} \sim_\phi \Pi y.\ \mathcal{G}\{x \leftarrow y\}$ with $y \notin fv(\mathcal{G})$**.**
Then $F \vDash \Pi x.\ \mathcal{G}$; this holds iff $F = \langle \rho, z, t \rangle$ and $\forall n,G.\ t \Downarrow_{\rho[z \leftarrow n]} G \Rightarrow G \vDash \mathcal{G}\{x \leftarrow n\}$.
Since $y \notin fv(\mathcal{G})$, this is the same as $\forall n,G.\ t \Downarrow_{\rho[z \leftarrow n]} G \Rightarrow G \vDash \mathcal{G}\{x \leftarrow y\}\{y \leftarrow n\}$,

that is $\langle \rho, z, t\rangle \vDash \Pi y.\, \mathcal{G}\{x \leftarrow y\}$.

(2)

Cases (EqH Refl) (EqH Symm) (EqH Trans). As in Proposition 6-11.

Case (EqH $\mathcal{A}$ Congr) From Proposition 6-11.

Case (EqH $\rightarrow$ Congr) Symmetrical to (1).

Case (EqH Π Congr) Symmetrical to (1).

Case (EqH Π-α) $\Pi x.\, \mathcal{G} \sim_\phi \Pi y.\, \mathcal{G}\{x \leftarrow y\}$ with $y \notin fv(\mathcal{G})$**.**

Then $F \vDash \Pi y.\, \mathcal{G}\{x \leftarrow y\}$; this holds iff $F = \langle \rho, z, t\rangle$

and $\forall n, G.\ t \Downarrow_{\rho[z \leftarrow n]} G \ \Rightarrow\ G \vDash \mathcal{G}\{x \leftarrow y\}\{y \leftarrow n\}$. Since $y \notin fv(\mathcal{G})$, this is the same as $\forall n, G.\ t \Downarrow_{\rho[z \leftarrow n]} G \ \Rightarrow\ G \vDash \mathcal{G}\{x \leftarrow n\}$, that is $\langle \rho, z, t\rangle \vDash \Pi x.\, \mathcal{G}$.

□

Proof of 6-12 Proposition: Soundness of Equivalence and Apartness. (p.15)

(1) If $\mathcal{N} \#_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{N},\mathcal{M}) \Rightarrow \varepsilon(\mathcal{N}) \neq \varepsilon(\mathcal{M})$.

(2) If $\mathcal{N} \sim_\phi \mathcal{M}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{N},\mathcal{M}) \Rightarrow \varepsilon(\mathcal{N}) = \varepsilon(\mathcal{M})$.

(3) If $\mathcal{A} \sim_\phi \mathcal{B}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{A},\mathcal{B}) \Rightarrow \forall P.\ P \vDash \varepsilon(\mathcal{A}) \Rightarrow P \vDash \varepsilon(\mathcal{B})$.

(4) If $\mathcal{F} \sim_\phi \mathcal{G}$ then $\forall \varepsilon \vDash \phi.\ (\varepsilon \textit{ grounds } \mathcal{F},\mathcal{G}) \Rightarrow \forall F.\ F \vDash \varepsilon(\mathcal{F}) \Rightarrow F \vDash \varepsilon(\mathcal{G})$.

Proof

(1) Assume $\mathcal{N} \#_\phi \mathcal{M}$. Take any $\varepsilon \vDash \phi$; by Proposition 6-6 we obtain $\varepsilon(\mathcal{N}) \#_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$. Assume ε *grounds* $\mathcal{N},\mathcal{M}$; that is, $\varepsilon(\mathcal{N})$ and $\varepsilon(\mathcal{M})$ are closed, so that $\varepsilon(\mathcal{N})=n$ and $\varepsilon(\mathcal{M})=m$ for some n,m and $n \#_{\varepsilon(\phi)} m$. Take any p, and assume $p \vDash n$; by Proposition 6-8 we obtain $p \nvDash m$. By definition of satisfaction we must have $n = p \neq m$

(2) Assume $\mathcal{N} \sim_\phi \mathcal{M}$. Take any $\varepsilon \vDash \phi$; by Proposition 6-6 we obtain $\varepsilon(\mathcal{N}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{M})$. Assume ε *grounds* $\mathcal{N},\mathcal{M}$; that is, $\varepsilon(\mathcal{N})$ and $\varepsilon(\mathcal{M})$ are closed, so that $\varepsilon(\mathcal{N})=n$ and $\varepsilon(\mathcal{M})=m$ for some n,m and $n \sim_{\varepsilon(\phi)} m$. Take any p, and assume $p \vDash n$; by Proposition 6-8 we obtain $p \vDash m$. By definition of satisfaction we must have $n = p = m$.

(3) Assume $\mathcal{A} \sim_\phi \mathcal{B}$. Take any $\varepsilon \vDash \phi$; by Proposition 6-6 we obtain $\varepsilon(\mathcal{A}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{B})$. Assume ε *grounds* $\mathcal{A},\mathcal{B}$; that is, $\varepsilon(\mathcal{A})$ and $\varepsilon(\mathcal{B})$ are closed. Take any P, and assume $P \vDash \varepsilon(\mathcal{A})$; by Proposition 6-11 we obtain $P \vDash \varepsilon(\mathcal{B})$.

(4) Assume $\mathcal{F} \sim_\phi \mathcal{G}$. Take any $\varepsilon \vDash \phi$; by Proposition 6-6 we obtain $\varepsilon(\mathcal{F}) \sim_{\varepsilon(\phi)} \varepsilon(\mathcal{G})$. Assume ε *grounds* $\mathcal{F},\mathcal{G}$; that is, $\varepsilon(\mathcal{F})$ and $\varepsilon(\mathcal{G})$ are closed. Take any F, and assume $F \vDash \varepsilon(\mathcal{F})$; by Proposition 6-11 we obtain $F \vDash \varepsilon(\mathcal{G})$.

□

Facts:

If $\rho \vDash E$ then $dom(\rho) \supseteq dom(E)$

$\rho \vDash E$ iff $\rho|_{dom(E)} \vDash E$

Proof of 7-4 Lemma: Environemnt Satisfaction and Freshness Signatures (p.17)

If $\rho \vDash E$ then $\rho|_{dom(fs(E))} \vDash fs(E)$

where $\rho|_S$ is the restriction of ρ to the domain S,

and $\vDash fs(E)$ is in the sense of Definition 6-4

Proof

By cases on the structure of E.

Case $E = n_1, ..., n_k$. Then $fs(E) = n_1, ..., n_k$ and $\rho|_{dom(fs(E))} = \emptyset$.

By Definition 6-4, $\emptyset \vDash n_1, ..., n_k$. That is, $\rho|_{dom(fs(E))} \vDash fs(E)$.

Case $E = E', x{:}\mathcal{A}$. We have $\rho = \rho'[x \leftarrow F] \wedge x \notin dom(\rho') \wedge \rho' \vDash E'$.

Then, $fs(E) = fs(E')$ and $\rho|_{dom(fs(E))} = \rho'[x \leftarrow F]|_{dom(fs(E'))} = \rho'|_{dom(fs(E'))}$.

By Ind Hyp, $\rho'|_{dom(fs(E'))} \vDash fs(E')$. That is, $\rho|_{dom(fs(E))} \vDash fs(E)$.

Case $E = E', \forall x{:}\mathbf{N}$. We have $\rho = \rho'[x \leftarrow n] \wedge x \notin dom(\rho') \wedge \rho' \vDash E'$.

Then, $fs(E) = fs(E'), \forall x$ and $\rho|_{dom(fs(E))} = \rho'[x \leftarrow n]|_{dom(fs(E'), \forall x)} = \rho'|_{dom(fs(E'))}[x \leftarrow n]$.

By Ind Hyp, $\rho'|_{dom(fs(E'))} \vDash fs(E')$.

By Definition 6-4, $\rho'|_{dom(fs(E'))}[x \leftarrow n] \vDash fs(E'), \forall x$. That is, $\rho|_{dom(fs(E))} \vDash fs(E)$.

Case $E = E', \mathsf{H}z{:}\mathbf{N}$.

We have $\rho = \rho'[z \leftarrow n] \wedge z \notin dom(\rho') \wedge \rho' \vDash E' \wedge n \notin na(E') \wedge n \notin na(\rho')$.

Then, $fs(E) = fs(E'), \mathsf{H}z$ and $\rho|_{dom(fs(E))} = \rho'[z \leftarrow n]|_{dom(fs(E'), \mathsf{H}z)} = \rho'|_{dom(fs(E'))}[z \leftarrow n]$.

By Ind Hyp, $\rho'|_{dom(fs(E'))} \vDash fs(E')$, hence, by Definition 6-4, we have:

(a) $\forall x \in dom(fs(E'))$. $fs(E')(x) = \mathsf{H} \Rightarrow \rho'(x) \notin na(fs(E'))$, i.e., $\rho'(x) \notin na(E')$.

(b) $\forall y \in dom(fs(E'))$. $\forall x \in dom(fs(E'))$. $(x <_{fs(E')} y \wedge fs(E')(y) = \mathsf{H}) \Rightarrow \rho'(x) \neq \rho'(y)$.

Let $\varepsilon = \rho'|_{dom(fs(E'))}[z \leftarrow n]$ and $\phi = fs(E'), \mathsf{H}z$.

We need to show $\varepsilon \vDash \phi$ according to Definition 6-4, that is:

- $dom(\varepsilon) \subseteq dom(\phi)$, trivially.
- $\forall x \in dom(\varepsilon)$. $\phi(x) = \mathsf{H} \Rightarrow \varepsilon(x) \notin na(\phi)$.

  If $x \in dom(fs(E'))$, apply (a). Else, if $x = z$, we have $\phi(z) = \mathsf{H}$ and $\varepsilon(z) = n$.

  By assumption, $n \notin na(E') = na(\phi)$.
- $\forall y \in dom(\varepsilon)$. $\forall x \in dom(\phi)$. $(x <_\phi y \wedge \phi(y) = \mathsf{H}) \Rightarrow (x \in dom(\varepsilon) \wedge \varepsilon(x) \neq \varepsilon(y))$.

  If $y \in dom(fs(E'))$ then if $x \in dom(fs(E'))$ apply (b), else if $x = z$, we do *not* have $z <_\phi y$.

  If $y = z$, we have $\phi(z) = \mathsf{H}$; take $x \in dom(\phi)$. If $x = z$ then we do *not* have $x <_\phi x$.

  Else $x \in dom(fs(E'))$ and $x <_\phi z$; we have $x \in dom(\varepsilon)$, and $\varepsilon(x) = \rho'(x) \neq n = \varepsilon(z)$ because of the assumption $n \notin na(\rho')$.

Hence, we have shown $\rho'|_{dom(fs(E'))}[z \leftarrow n] \vDash fs(E'), \mathsf{H}z$. That is, $\rho|_{dom(fs(E))} \vDash fs(E)$.

□

10-3 Proposition: Subsumption.

If $E \vdash \mathcal{F} <: \mathcal{G}$ and $\varepsilon \vDash fs(E)$ and $H \vDash_H \varepsilon(\mathcal{F})$ then $H \vDash_H \varepsilon(\mathcal{G})$.

Proof

Induction on the derivation of $E \vdash \mathcal{F} <: \mathcal{G}$.

(Sub Tree) We have $E \vdash \mathcal{A} <: \mathcal{B}$ and $\varepsilon \vDash fs(E)$ and $P \vDash \varepsilon(\mathcal{A})$.

From (Sub Tree) we must have $E \vdash_T \mathcal{A}$ and $E \vdash_T \mathcal{B}$, and $\langle \mathcal{A}, fs(E), \mathcal{B} \rangle \in ValidEntailments$, which means $\forall \varepsilon \vDash fs(E)$. ($\varepsilon$ *grounds* $\mathcal{A}, \mathcal{B}$) $\Rightarrow \forall P$. $P \vDash \varepsilon(\mathcal{A}) \Rightarrow P \vDash \varepsilon(\mathcal{B})$.

Since $E \vdash_T \mathcal{A}$ and $\varepsilon \vDash fs(E)$, we have $fv(\mathcal{A}) \subseteq dom(E) \subseteq dom(\varepsilon)$.

Since $E \vdash_T \mathcal{B}$ and $\varepsilon \vDash fs(E)$, we have $fv(\mathcal{B}) \subseteq dom(E) \subseteq dom(\varepsilon)$.

Hence ε is a ground valuation for $\mathcal{A}$ and $\mathcal{B}$.

Since $\varepsilon \vDash fs(E)$, we obtain that $\forall P$. $P \vDash \varepsilon(\mathcal{A}) \Rightarrow P \vDash \varepsilon(\mathcal{B})$.

We also have $P \vDash \varepsilon(\mathcal{A})$, therefore $P \vDash \varepsilon(\mathcal{B})$.

(Sub Equiv) We have $E \vdash \mathcal{F} <: \mathcal{G}$ and $\varepsilon \vDash fs(E)$ and $H \vDash \varepsilon(\mathcal{F})$

From (Sub Equiv) we must have $E \vdash \mathcal{F}$ and $E \vdash \mathcal{G}$ and $\mathcal{F} \sim_{fs(E)} \mathcal{G}$.

Since $E \vdash \mathcal{F}$ and $\varepsilon \vDash fs(E)$, we have $fv(\mathcal{F}) \subseteq dom(E) \subseteq dom(\varepsilon)$.

Since $E \vdash \mathcal{G}$ and $\varepsilon \vDash fs(E)$ we also have $fv(\mathcal{G}) \subseteq dom(E) \subseteq dom(\varepsilon)$.
Hence ε is a ground valuation for $\mathcal{F}$ and $\mathcal{G}$.
Then, since $\varepsilon \vDash fs(E)$ and $\mathcal{F} \sim_{fs(E)} \mathcal{G}$, by Proposition 6-12 we obtain $H \vDash \varepsilon(\mathcal{G})$.

(Sub N) We have $E \vdash \mathbf{N} <: \mathbf{N}$ and $\varepsilon \vDash fs(E)$ and $H \vDash \varepsilon(\mathbf{N}) = \mathbf{N}$; trivially, $H \vDash \varepsilon(\mathbf{N})$.

(Sub $\rightarrow$) We have $E \vdash \mathcal{F} \rightarrow \mathcal{G} <: \mathcal{F}' \rightarrow \mathcal{G}'$ and $\varepsilon \vDash fs(E)$ and $H \vDash \varepsilon(\mathcal{F} \rightarrow \mathcal{G}) = \varepsilon(\mathcal{F}) \rightarrow \varepsilon(\mathcal{G})$.
From (Sub $\rightarrow$) we must have $E \vdash \mathcal{F}' <: \mathcal{F}$ and $E \vdash \mathcal{G} <: \mathcal{G}'$.
By definition, $H = \langle \rho, x, t \rangle$ and $\forall F, G.\ (F \vDash \varepsilon(\mathcal{F}) \ \wedge\ t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \vDash \varepsilon(\mathcal{G})$.
Take any $F \vDash \varepsilon(\mathcal{F}')$; by Ind Hyp $F \vDash \varepsilon(\mathcal{F})$. Assume $t \Downarrow_{\rho[x \leftarrow F]} G$, then $G \vDash \varepsilon(\mathcal{G})$, and by Ind Hyp $G \vDash \varepsilon(\mathcal{G}')$. We have shown that $\forall F, G.\ (F \vDash \varepsilon(\mathcal{F}') \wedge\ t \Downarrow_{\rho[x \leftarrow F]} G) \Rightarrow G \vDash \varepsilon(\mathcal{G}')$. That is, we have shown that $\langle \rho, x, t \rangle \vDash \varepsilon(\mathcal{F}') \rightarrow \varepsilon(\mathcal{G}')$, i.e., $H \vDash \varepsilon(\mathcal{F}' \rightarrow \mathcal{G}')$.

(Sub Π) We have $E \vdash \Pi x.\ \mathcal{G} <: \Pi x.\ \mathcal{G}'$ and $\varepsilon \vDash fs(E)$ and $H \vDash \varepsilon(\Pi x.\ \mathcal{G}) = \Pi x.\ \varepsilon \backslash x(\mathcal{G})$.
From (Sub Π) we must have $E,\ \forall x{:}\mathbf{N} \vdash \mathcal{G} <: \mathcal{G}'$. By definition, $H = \langle \rho, z, t \rangle$ and $\forall n, G.\ t \Downarrow_{\rho[z \leftarrow n]} G \ \Rightarrow\ G \vDash (\varepsilon \backslash x(\mathcal{G}))\{x \leftarrow n\} = \varepsilon[x \leftarrow n](\mathcal{G})$.
Take any n and assume $t \Downarrow_{\rho[z \leftarrow n]} G$, then $G \vDash \varepsilon[x \leftarrow n](\mathcal{G})$.
By Ind Hyp, since $\varepsilon[x \leftarrow n] \vDash fs(E,\ \forall x{:}\mathbf{N})$, we get $G \vDash \varepsilon[x \leftarrow n](\mathcal{G}') = (\varepsilon \backslash x(\mathcal{G}'))\{x \leftarrow n\}$.
We have shown that $\forall n, G.\ (t \Downarrow_{\rho[z \leftarrow n]} G) \Rightarrow G \vDash (\varepsilon \backslash x(\mathcal{G}'))\{x \leftarrow n\}$.
That is, we have shown that $\langle \rho, z, t \rangle \vDash \Pi x.\ \varepsilon \backslash x(\mathcal{G}')$, i.e., $H \vDash \varepsilon(\Pi x.\ \mathcal{G}')$.

□

Note: in the following proof we often have expressions like $\rho(\mathcal{A})$ and $\rho(\mathcal{F})$. Under the assumptions that $\rho \vDash E$ and $E \vdash t : ...\mathcal{A}...$, or $E \vdash ...\mathcal{A}... : \mathcal{F}$, or $E \vdash u : \mathcal{F}$ we can infer that $\rho(\mathcal{A})$ and $\rho(\mathcal{F})$ are well formed and that the relevant reduction rules actually fire. This is not essential to the proof, because if $\rho(\mathcal{A})$ and $\rho(\mathcal{F})$ were ill-formed, and the reduction rules were stuck, subject reduction would still vacuously hold. However, this form of stuckness does not actually arise.

Proof of 7-5 Theorem: Subject Reduction. (p.18)

(1) If $E \vdash_N \mathcal{N}$ and $\rho \vDash E$ and $\mathcal{N} \downarrow_\rho n$ then $n \vDash_N \rho(\mathcal{N})$.
(2) If $E \vdash t : \mathcal{F}$ and $\rho \vDash E$ and $t \Downarrow_\rho F$, then $F \vDash_H \rho(\mathcal{F})$.

Proof

1) If $E \vdash_N \mathcal{N}$ and $\rho \vDash E$ and $\mathcal{N} \downarrow_\rho n$ then $n \vDash \rho(\mathcal{N})$.
Induction on the derivation of $E \vdash_N \mathcal{N}$.

(NExpr *n*) We have $E \vdash_N m$ and $\rho \vDash E$ and $m \downarrow_\rho n$.
We must have from (NRed *n*) that m is a name and $m{=}n$.
By definition of satisfaction, $n \vDash n = \rho(m)$.

(NExpr *x*) We have $E \vdash_N x$ and $\rho \vDash E$ and $x \downarrow_\rho n$.
We must have from (NExpr *x*) that $E(x) = \mathbf{N}$, so $x \in dom(E)$.
We must have from (NRed *x*) that $x \in dom(\rho)$ and $n = \rho(x)$.
Since $\rho \vDash E$, we have that $\rho(x) \vDash \rho(E(x)) = \mathbf{N}$, that is $\rho(x) \in \Lambda$, so $n \in \Lambda$.
By definition of satisfaction, $n \vDash n = \rho(x)$.

(NExpr $\leftrightarrow$) We have $E \vdash_N \mathcal{M}_0\{\mathcal{M}_1 \leftrightarrow \mathcal{M}_2\}$ and $\rho \vDash E$ and $\mathcal{M}_0\{\mathcal{M}_1 \leftrightarrow \mathcal{M}_2\} \downarrow_\rho n$.
We must have from (NExpr $\leftrightarrow$) that $E \vdash_N \mathcal{M}_0$ and $E \vdash_N \mathcal{M}_1$ and $E \vdash_N \mathcal{M}_2$.
We must have from (NRed $\leftrightarrow$) that $\mathcal{M}_0 \downarrow_\rho m_0$ and $\mathcal{M}_1 \downarrow_\rho m_1$ and $\mathcal{M}_2 \downarrow_\rho m_2$, with $n = m_0 \bullet \{m_1 \leftrightarrow m_2\}$. That is, $n \bullet \{m_1 \leftrightarrow m_2\} = m_0$.
By Ind Hyp, $m_0 \vDash \rho(\mathcal{M}_0)$ and $m_1 \vDash \rho(\mathcal{M}_1)$ and $m_2 \vDash \rho(\mathcal{M}_2)$.

By definition of satisfaction, $n \vDash \rho(\mathcal{M}_0)\{\rho(\mathcal{M}_1)\leftrightarrow\rho(\mathcal{M}_2)\}$,
that is, $n \vDash \rho(\mathcal{M}_0\{\mathcal{M}_1\leftrightarrow\mathcal{M}_2\})$.

2) If $E \vdash t : \mathcal{T}$ and $\rho \vDash E$ and $t \Downarrow_\rho F$, then $F \vDash \rho(\mathcal{T})$.
Induction on the derivation of $E \vdash t : \mathcal{T}$.

(Term *x*) We have $E \vdash x : E(x)$ and $\rho \vDash E$ and $x \Downarrow_\rho F$.
We must have from (Red x) that $x \in dom(\rho)$ and $F = \rho(x)$.
Since $\rho \vDash E$, we have that $\rho(x) \vDash \rho(E(x))$, i.e. $F \vDash \rho(E(x))$.

(Term 0) We have $E \vdash 0 : \mathbf{0}$ and $\rho \vDash E$ and $0 \Downarrow_\rho 0$. By definition, $0 \vDash \rho(\mathbf{0})$.

(Term $\mathcal{N}$[]) We have $E \vdash \mathcal{N}[t] : \mathcal{N}[\mathcal{A}]$ and $\rho \vDash E$ and $\mathcal{N}[t] \Downarrow_\rho P$.
We must have from (Term $\mathcal{N}$[]) that $E \vdash_N \mathcal{N}$ and $E \vdash_T t : \mathcal{A}$.
We must have from (Red $\mathcal{N}$[]) that $\mathcal{N} \downarrow_\rho n$ and $t \Downarrow_\rho P'$ and $P = n[P']$.
By (1), $n \vDash \rho(\mathcal{N})$, and by Ind Hyp, $P' \vDash \rho(\mathcal{A})$,
hence by definition of satisfaction $n[P'] \vDash \rho(\mathcal{N})[\rho(\mathcal{A})] = \rho(\mathcal{N}[\mathcal{A}])$.

(Term |) We have $E \vdash t \mid u : \mathcal{A} \mid \mathcal{B}$ and $\rho \vDash E$ and $t \mid u \Downarrow_\rho R$.
We must have from (Term |) that $E \vdash_T t : \mathcal{A}$ and $E \vdash_T u : \mathcal{B}$.
We must have from (Red |) that $R = P \mid Q$ and $t \Downarrow_\rho P$ and $u \Downarrow_\rho Q$.
By Ind Hyp, $P \vDash \rho(\mathcal{A})$ and $Q \vDash \rho(\mathcal{B})$, hence by definition $P \mid Q \vDash \rho(\mathcal{A} \mid \mathcal{B})$.

(Term ν) We have $E \vdash (\nu x)t : \mathsf{H}x.\mathcal{A}$ and $\rho \vDash E$ and $(\nu x)t \Downarrow_\rho F$.
We must have from (Term ν) that $E, \mathsf{H}x{:}\mathbf{N} \vdash_T t : \mathcal{A}$.
We must have from (Red ν) that $F = (\nu n)P$ and $t \Downarrow_{\rho[x\leftarrow n]} P$ for $n \notin na(t,\rho)$.
Since n could appear in $\mathcal{A}$, thus blocking the last step of this proof,
take $n' \notin na(t,\mathcal{A},\rho,P)$, so that $F \equiv_\alpha (\nu n')P\bullet\{n\leftrightarrow n'\}$.
By Lemma 5-2 $t\bullet\{n\leftrightarrow n'\} \Downarrow_{\rho[x\leftarrow n]\bullet\{n\leftrightarrow n'\}} P\bullet\{n\leftrightarrow n'\}$, that is $t \Downarrow_{\rho[x\leftarrow n']} P\bullet\{n\leftrightarrow n'\}$.
We have that $\rho[x\leftarrow n'] \vDash E, \mathsf{H}x{:}\mathbf{N}$. By Ind Hyp, $P\bullet\{n\leftrightarrow n'\} \vDash \rho[x\leftarrow n'](\mathcal{A})$.
That is, $P\bullet\{n\leftrightarrow n'\} \vDash \rho\backslash x(\mathcal{A})\{x\leftarrow n'\}$.
Since $F \equiv (\nu n')P\bullet\{n\leftrightarrow n'\}$ and $n' \notin na(\rho\backslash x(\mathcal{A}))$,
by definition, $F \vDash \mathsf{H}x.\rho\backslash x(\mathcal{A})$. That is, $F \vDash \rho(\mathsf{H}x.\mathcal{A})$.

(Term ↔) We have $E \vdash t\{\mathcal{M}\leftrightarrow\mathcal{M}'\} : \mathcal{A}\{\mathcal{M}\leftrightarrow\mathcal{M}'\}$ and $\rho \vDash E$ and $t\{\mathcal{M}\leftrightarrow\mathcal{M}'\} \Downarrow_\rho P$.
We must have from (Term ↔) that $E \vdash_T t : \mathcal{A}$ and $E \vdash_N \mathcal{M}$ and $E \vdash_N \mathcal{M}'$.
We must have from (Red ↔) that $P = P'\bullet\{m\leftrightarrow m'\}$
with $t \Downarrow_\rho P'$ and $\mathcal{M} \downarrow_\rho m$ and $\mathcal{M}' \downarrow_\rho m'$.
By (1) and Ind Hyp, $P' \vDash \rho(\mathcal{A})$ and $m \vDash \rho(\mathcal{M})$ and $m' \vDash \rho(\mathcal{M}')$.
By Lemma 10-2, $P'\bullet\{m\leftrightarrow m'\} \vDash \rho(\mathcal{A})\{\rho(\mathcal{M})\leftrightarrow\rho(\mathcal{M}')\}$,
that is $P \vDash \rho(\mathcal{A}\{\mathcal{M}\leftrightarrow\mathcal{M}'\})$.

(Term ÷$\mathcal{N}$[]) We have $E \vdash t \div (\mathcal{N}[y{:}\mathcal{A}]).u : \mathcal{T}$ and $\rho \vDash E$ and $t \div (n[y{:}\mathcal{A}]).u \Downarrow_\rho F$.
We must have from (Term ÷$\mathcal{N}$[]) that $E \vdash_T t : \mathcal{N}[\mathcal{A}]$ and $E, y{:}\mathcal{A} \vdash u : \mathcal{T}$.
We must have from (Red ÷$\mathcal{N}$[]) that $\mathcal{N} \downarrow_\rho n$ and $t \Downarrow_\rho \equiv n[P]$ and $P \vDash \rho(\mathcal{A})$
and $u \Downarrow_{\rho[y\leftarrow P]} F$. We have that $\rho[y\leftarrow P] \vDash E, y{:}\mathcal{A}$.
By Ind Hyp $E, y{:}\mathcal{A} \vdash u : \mathcal{T}$, $\rho[y\leftarrow P] \vDash E, y{:}\mathcal{A}$, and $u \Downarrow_{\rho[y\leftarrow P]} F$ imply $F \vDash \rho[y\leftarrow P](\mathcal{T})$.
But since $y{:}\mathcal{A}$ (is not a name), we have $\rho[y\leftarrow P](\mathcal{T}) = \rho(\mathcal{T})$.

(Term ÷|) We have $E \vdash t \div (x{:}\mathcal{A} \mid y{:}\mathcal{B}).u : \mathcal{T}$ and $\rho \vDash E$ and $t \div (x{:}\mathcal{A} \mid y{:}\mathcal{B}).u \Downarrow_\rho F$.
We must have from (Term ÷|) that $E \vdash_T t : \mathcal{A} \mid \mathcal{B}$ and $E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{T}$.

We must have from (Red ÷|) that $t \Downarrow_\rho \equiv P' \mid P''$ and $P' \vDash \rho(\mathcal{A})$ and $P'' \vDash \rho(\mathcal{B})$
and $u \Downarrow_{\rho[x\leftarrow P'][y\leftarrow P'']} F$ and $x \neq y$.
Since $E \vdash_T t : \mathcal{A} \mid \mathcal{B}$ and $E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{T}$ (and hence $E, x{:}\mathcal{A}, y{:}\mathcal{B}$ is well-formed),
it follows that $\mathcal{B}$ does not have x free. Therefore $\rho[x\leftarrow P'](\mathcal{B}) = \rho(\mathcal{B})$, and hence
$\rho[x\leftarrow P'][y\leftarrow P''] \vDash E, x{:}\mathcal{A}, y{:}\mathcal{B}$.
By Ind Hyp $E, x{:}\mathcal{A}, y{:}\mathcal{B} \vdash u : \mathcal{T}$ and $\rho[x\leftarrow P'][y\leftarrow P''] \vDash E, x{:}\mathcal{A}, y{:}\mathcal{B}$
and $u \Downarrow_{\rho[x\leftarrow P'][y\leftarrow P'']} F$ imply $F \vDash \rho[x\leftarrow P'][y\leftarrow P''](\mathcal{T})$.
But since $x{:}\mathcal{A}, y{:}\mathcal{B}$ (are not names), we have $\rho[x\leftarrow P'][y\leftarrow P''](\mathcal{T}) = \rho(\mathcal{T})$.

(Term ÷ν) We have $E \vdash t \div ((\nu x)y{:}\mathcal{A}).u : \mathsf{H}x.\mathcal{B}$ and $\rho \vDash E$ and $t \div ((\nu x)y{:}\mathcal{A}).u \Downarrow_\rho F$.
We must have from (Term ÷ν) that $E \vdash_T t : \mathsf{H}x.\mathcal{A}$ and $E, x{:}\mathbf{N}, y{:}\mathcal{A} \vdash_T u : \mathcal{B}$.
We must have from (Red ÷ν) that $n \notin na(t, \mathcal{A}, u, \rho)$ and $t \Downarrow_\rho \equiv (\nu n)P$
and $P \vDash \rho[x\leftarrow n](\mathcal{A})$ and $u \Downarrow_{\rho[x\leftarrow n][y\leftarrow P]} Q$ and $F = (\nu n)Q$ and $x \neq y$.
Take $n' \notin na(t, u, \mathcal{A}, \mathcal{B}, \rho, Q)$, so that $F \equiv_\alpha (\nu n')Q\bullet(n\leftrightarrow n')$.
By Lemma 5-2 $u\bullet(n\leftrightarrow n') \Downarrow_{\rho[x\leftarrow n][y\leftarrow P]\bullet(n\leftrightarrow n')} Q\bullet(n\leftrightarrow n')$.
that is $u \Downarrow_{\rho[x\leftarrow n'][y\leftarrow P\bullet(n\leftrightarrow n')]} Q\bullet(n\leftrightarrow n')$.
By Lemma 4-3 $P\bullet(n\leftrightarrow n') \vDash \rho[x\leftarrow n](\mathcal{A})\bullet(n\leftrightarrow n')$, that is $P\bullet(n\leftrightarrow n') \vDash \rho[x\leftarrow n'](\mathcal{A})$,
so we have that $\rho[x\leftarrow n'][y\leftarrow P\bullet(n\leftrightarrow n')] \vDash E, x{:}\mathbf{N}, y{:}\mathcal{A}$.
By Ind Hyp $Q\bullet(n\leftrightarrow n') \vDash \rho[x\leftarrow n'][y\leftarrow P\bullet(n\leftrightarrow n')](\mathcal{B})$.
Since $y{:}\mathcal{A}$ (is not a name) and $y \notin dom(E, x{:}\mathbf{N})$
we have $\rho[x\leftarrow n'][y\leftarrow P\bullet(n\leftrightarrow n')](\mathcal{B}) = \rho[x\leftarrow n'](\mathcal{B})$.
That is, $Q\bullet(n\leftrightarrow n') \vDash \rho[x\leftarrow n'](\mathcal{B})$, that is $Q\bullet(n\leftrightarrow n') \vDash \rho\backslash x(\mathcal{B})\{x\leftarrow n'\}$
Since $F \equiv (\nu n')Q\bullet(n\leftrightarrow n')$ and $n' \notin na(\rho\backslash x(\mathcal{B}))$,
by definition of satisfaction, $F \vDash \mathsf{H}x.\rho\backslash x(\mathcal{B})$. That is, $F \vDash \rho(\mathsf{H}x.\mathcal{B})$.

(Term ?) We have $E \vdash t?(x{:}\mathcal{A}).u, v : \mathcal{T}$ and $\rho \vDash E$ and $t?(x{:}\mathcal{A}).u, v \Downarrow_\rho F$.
We have from (Term ?) that $E \vdash_T t : \mathcal{B}$ and $E, x{:}\mathcal{A} \vdash u : \mathcal{T}$ and $E, x{:}\neg\mathcal{A} \vdash v : \mathcal{T}$.
The reduction may come from (Red ?⊨); then $t \Downarrow_\rho P$ and $P \vDash \rho(\mathcal{A})$ and $u \Downarrow_{\rho[x\leftarrow P]} F$.
We have that $\rho[x\leftarrow P] \vDash E, x{:}\mathcal{A}$.
By Ind Hyp $E, x{:}\mathcal{A} \vdash u : \mathcal{T}$ and $\rho[x\leftarrow P] \vDash E, x{:}\mathcal{A}$ and $u \Downarrow_{\rho[x\leftarrow P]} F$
imply $F \vDash \rho[x\leftarrow P](\mathcal{T})$. But since $x{:}\mathcal{A}$ (is not a name), we have $\rho[x\leftarrow P](\mathcal{T}) = \rho(\mathcal{T})$.
Else the reduction must be from (Red ?⊭);
then $t \Downarrow_\rho P$ and $P \vDash \rho(\neg\mathcal{A})$ and $v \Downarrow_{\rho[x\leftarrow P]} F$.
We have that $\rho[x\leftarrow P] \vDash E, x{:}\neg\mathcal{A}$.
By Ind Hyp $E, x{:}\neg\mathcal{A} \vdash v : \mathcal{T}$ and $\rho[x\leftarrow P] \vDash E, x{:}\neg\mathcal{A}$ and $v \Downarrow_{\rho[x\leftarrow P]} F$
imply $F \vDash \rho[x\leftarrow P](\mathcal{T})$. But since $x{:}\neg\mathcal{A}$ (is not a name), we have $\rho[x\leftarrow P](\mathcal{T}) = \rho(\mathcal{T})$.

(Term $\mathcal{N}$) We $E \vdash \mathcal{N} : \mathbf{N}$ and $\rho \vDash E$ and $\mathcal{N} \Downarrow_\rho F$.
We must have from (Term $\mathcal{N}$) that $E \vdash_N \mathcal{N}$.
We must have from (Red $\mathcal{N}$) that $F = n$ and $\mathcal{N} \downarrow_\rho n$.
By (1), $n \vDash_N \rho(\mathcal{N})$. Hence $n \vDash_H \mathbf{N}$.

(Term λ) We have $E \vdash \lambda x{:}\mathcal{T}.t : \mathcal{T}\rightarrow\mathcal{G}$ and $\rho \vDash E$ and $\lambda x{:}\mathcal{T}.t \Downarrow_\rho F$.
We must have from (Term λ) that $E, x{:}\mathcal{T} \vdash t : \mathcal{G}$ and $\mathcal{T} \neq \mathbf{N}$.
We must have from (Red λ) that $F = \langle\rho, x, t\rangle$.
We need to show that $\langle\rho, x, t\rangle \vDash \rho(\mathcal{T}\rightarrow\mathcal{G})$,
that is that $\forall F,G.\ (F \vDash \rho(\mathcal{T}) \ \wedge\ t \Downarrow_{\rho[x\leftarrow F]} G) \Rightarrow G \vDash \rho(\mathcal{G})$.

Take any $F \vDash \rho(\mathcal{T})$, then $\rho[x \leftarrow F] \vDash E, x{:}\mathcal{T}$.
Assuming that $t \Downarrow_{\rho[x \leftarrow F]} G$ we need to show that $G \vDash \rho(\mathcal{G})$.
By Ind Hyp if $E, x{:}\mathcal{T} \vdash t : \mathcal{G}$ and $\rho[x \leftarrow F] \vDash E, x{:}\mathcal{T}$ and $t \Downarrow_{\rho[x \leftarrow F]} G$, then $G \vDash \rho(\mathcal{G})$.

(Term App) We have $E \vdash t(u) : \mathcal{T}$ and $\rho \vDash E$ and $t(u) \Downarrow_\rho F$.
We must have from (Term App) that $E \vdash t : \mathcal{G} \rightarrow \mathcal{T}$ and $E \vdash u : \mathcal{G}$ (where $\mathcal{G} \neq \mathbf{N}$)
We must have from (Red App) that $t \Downarrow_\rho \langle \rho', x, t' \rangle$ and $u \Downarrow_\rho G$ and $t' \Downarrow_{\rho'[x \leftarrow G]} F$.
By Ind Hyp if $E \vdash t : \mathcal{G} \rightarrow \mathcal{T}$ and $\rho \vDash E$ and $t \Downarrow_\rho \langle \rho', x, t' \rangle$ then $\langle \rho', x, t' \rangle \vDash \rho(\mathcal{G} \rightarrow \mathcal{T})$.
That means that $\forall G', F'.\ (G' \vDash \rho(\mathcal{G}) \ \wedge \ t' \Downarrow_{\rho'[x \leftarrow G']} F') \Rightarrow F' \vDash \rho(\mathcal{T})$.
By Ind Hyp if $E \vdash u : \mathcal{G}$ and $\rho \vDash E$ and $u \Downarrow_\rho G$ then $G \vDash \rho(\mathcal{G})$.
Hence, by taking $G'=G$ and $F'=F$, we conclude $F \vDash \rho(\mathcal{T})$.

(Term Depλ) We have $E \vdash \lambda x{:}\mathbf{N}.t : \Pi x.\ \mathcal{G}$ and $\rho \vDash E$ and $\lambda x{:}\mathbf{N}.t \Downarrow_\rho F$.
We must have from (Term Depλ) that $E, x{:}\mathbf{N} \vdash t : \mathcal{G}$.
We must have from (Red λ) that $F = \langle \rho, x, t \rangle$.
We need to show that $\langle \rho, x, t \rangle \vDash \rho(\Pi x.\ \mathcal{G})$,
that is that $\forall n, G.\ t \Downarrow_{\rho[x \leftarrow n]} G \Rightarrow G \vDash \rho(\mathcal{G}\{x \leftarrow n\})$.
Take any n, then $\rho[x \leftarrow n] \vDash E, x{:}\mathbf{N}$.
Assuming that $t \Downarrow_{\rho[x \leftarrow n]} G$ we need to show that $G \vDash \rho(\mathcal{G}\{x \leftarrow n\})$.
By Ind Hyp if $E, x{:}\mathbf{N} \vdash t : \mathcal{G}$ and $\rho[x \leftarrow n] \vDash E, x{:}\mathbf{N}$ and $t \Downarrow_{\rho[x \leftarrow n]} G$, then $G \vDash \rho[x \leftarrow n](\mathcal{G})$.
That is, $G \vDash \rho(\mathcal{G}\{x \leftarrow n\})$.

(Term DepApp) We have $E \vdash t(\mathcal{N}) : \mathcal{T}\{x \leftarrow \mathcal{N}\}$ and $\rho \vDash E$ and $t(\mathcal{N}) \Downarrow_\rho F$.
We must have from (Term DepApp) that $E \vdash t : \Pi x.\ \mathcal{T}$ and $E \vdash_N \mathcal{N}$.
We must have from (Red App) that $t \Downarrow_\rho \langle \rho', z, t' \rangle$ and $\mathcal{N} \Downarrow_\rho G$ and $t' \Downarrow_{\rho'[z \leftarrow G]} F$.
But $\mathcal{N} \Downarrow_\rho G$ must come from (Red $\mathcal{N}$), hence $\mathcal{N} \downarrow_\rho n$ and $G = n$.
By Ind Hyp if $E \vdash t : \Pi x.\ \mathcal{T}$ and $\rho \vDash E$ and $t \Downarrow_\rho \langle \rho', z, t' \rangle$ then $\langle \rho', z, t' \rangle \vDash \rho(\Pi x.\ \mathcal{T})$.
That means that $\forall n', F'.\ t' \Downarrow_{\rho'[z \leftarrow n']} F' \Rightarrow F' \vDash \rho(\mathcal{T}\{x \leftarrow n'\})$.
By (1) if $E \vdash_N \mathcal{N}$ and $\rho \vDash E$ and $\mathcal{N} \Downarrow_\rho n$ then $n \vDash_N \rho(\mathcal{N})$.
Hence, by taking $n'=n$ and $F'=F$, we conclude $F \vDash \rho(\mathcal{T}\{x \leftarrow n\})$,
that is $F \vDash \rho(\mathcal{T})\{x \leftarrow n\}$. By Proposition 6-8, $n \sim \rho(\mathcal{N})$,
and by Lemma 6-9, $F \vDash \rho(\mathcal{T})\{x \leftarrow \rho(\mathcal{N})\}$, that is $F \vDash \rho(\mathcal{T}\{x \leftarrow \mathcal{N}\})$.

(Subsumption) We have $E \vdash t : \mathcal{T}$ and $\rho \vDash E$ and $t \Downarrow_\rho F$.
We must have from (Subsumption) that $E \vdash t : \mathcal{G}$ and $E \vdash \mathcal{G} <: \mathcal{T}$.
By Ind Hyp, $F \vDash \rho(\mathcal{G})$, where $\rho(\mathcal{G}) = \rho|_{dom(fs(E))}(\mathcal{G})$.
By Lemma 7-4, since $\rho \vDash E$, we have $\rho|_{dom(fs(E))} \vDash fs(E)$.
Then by Proposition 10-3 (Subsumption), we have $F \vDash \rho|_{dom(fs(E))}(\mathcal{T})$,
where $\rho|_{dom(fs(E))}(\mathcal{T}) = \rho(\mathcal{T})$. Hence $F \vDash \rho(\mathcal{T})$.

□