A Modal Language for the Safety of Mobile Values

Sungwoo Park
Computer Science Department
Carnegie Mellon University
gla@cs.cmu.edu

April 25, 2005
CMU-CS-05-124

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present a modal language for distributed computation which addresses the safety of mobile values as well as mobile code. The safety of mobile code is achieved with the modality $\Box$ which corresponds to necessity of modal logic. For the safety of mobile values, we introduce a new modality $\bigcirc$ which expresses that given code evaluates to a mobile value. We demonstrate the use of modal types with three communication constructs: remote evaluation, futures, and asynchronous channels.

Keywords: Modal language, Distributed computation, Type system

1 Introduction

A distributed computation is a cooperative process taking place in a network of nodes. Each node is capable of performing a stand-alone computation and also communicating with other nodes to distribute and collect code and data. Thus a distributed computation has the potential to make productive use of all the nodes in the network simultaneously.

Usually a distributed computation assumes a heterogeneous group of nodes with different *local resources*. A local resource can be either a permanent/physical object available at a particular node (*e.g.*, printer, database) or an ephemeral/semantic object created during a stand-alone computation (*e.g.*, heap cell, abstract data type). Local resources are accessed via their references (*e.g.*, handle for a database file, pointer to a heap cell).

Local resources, however, give rise to an issue not found in stand-alone computations: the safety of *mobile code*, or in our terminology, the safety of *mobile terms* where a term represents a piece of code. In essence, a node cannot access remote resources in the same way that it accesses its own local resources, but it may receive mobile terms in which references to remote resources are exposed. Therefore the safety of mobile terms is achieved either by supporting direct access to remote resources (*e.g.*, remote file access, remote memory access) or by preventing references to remote resources from being dereferenced. This paper focuses on the second case with the assumption that references to remote resources are allowed in mobile terms as long as they are never dereferenced.

One approach to the safety of mobile terms is to build a modal type system with the modality $\Box$ [1, 12, 9, 13]. The basic idea is that a value of modal type $\Box A$ contains a mobile term that can be evaluated at any node. An indexed modal type $\Box_\omega A$ is used for mobile terms that can be evaluated at node ω. By requiring that a mobile term be from a value of type $\Box A$ or $\Box_\omega A$, we ensure its safety without recourse to runtime checks.

A type system augmented with the modality $\Box$ is not, however, expressive enough for the safe communication of *values*, *i.e.*, the safety of *mobile values*. In other words, we cannot rely solely on modal types $\Box A$ and $\Box_\omega A$ to verify that a value communicated from one node to another is mobile (*e.g.*, when a remote procedure call returns, or when a value is written to a channel). The reason is that in general, a value of type $\Box A$ or $\Box_\omega A$ contains *not a mobile value but a mobile term*. The evaluation of such a mobile term (with the intention of obtaining a mobile value) may result in a value that is not necessarily mobile because of references to local resources created during the evaluation.

As an example, consider a term of type `int -> int` in an ML-like language:

```
let
  val new_reference = ref 0
  val f = fn x => x + !new_reference
in
  f
end
```

The above term may be used in building a mobile term of type $\Box$`(int -> int)`, since it can be evaluated at any node. The resultant value `f`, however, is not mobile because it accesses a local resource `new_reference`. In contrast, the following term, also of type `int -> int`, cannot be used in building a mobile term, but the resultant value is mobile because it does not access any local resource:

```
let
  val v = !some_existing_reference
  val f = fn x => x + v
in
  f
end
```

Hence the modality $\Box$ is irrelevant to the safety of mobile values, which should now be verified by programmers themselves.

This paper investigates a new modality $\bigcirc$ which expresses that a given term evaluates to a mobile value. The basic idea is that a term contained in a value of modal type $\bigcirc A$ evaluates to a value that is valid at any node. Similarly to $\Box_\omega A$, an indexed modal type $\bigcirc_\omega A$ is used if the resultant value is valid at node ω. To obtain a value to be communicated to other nodes, we evaluate a term contained in a value of type $\bigcirc A$ or $\bigcirc_\omega A$. In this way, we achieve the safety of mobile values.

Since the mobility of a term is independent of the mobility of the value to which it evaluates, the two modalities $\Box$ and $\bigcirc$ are developed in an orthogonal way:

$$
\begin{array}{ccccc}
 & & \Box A & & \\
 & & | & & \\
\bigcirc_\omega A & - & A & - & \bigcirc A \\
 & & | & & \\
 & & \Box_\omega A & &
\end{array}
$$

We use combinations of $\Box$ and $\bigcirc$ to express various properties of mobile terms:

- $\Box\bigcirc A$: evaluates at any node to a value valid at any node.
- $\Box\bigcirc_\omega A$: evaluates at any node to a value valid at node ω.
- $\Box_\omega\bigcirc A$: evaluates at node ω to a value valid at any node.
- $\Box_\omega\bigcirc_{\omega'} A$: evaluates at node ω to a value valid at node ω'.

We first develop a modal language $\lambda_{\Box\bigcirc}$ by extending the λ-calculus with the modalities $\Box$ and $\bigcirc$. We formulate its type system in the natural deduction style by giving introduction and elimination rules for each connective and modality. The modality $\bigcirc$ requires us to introduce a typing judgment differentiating values from terms. This typing judgment induces a substitution defined inductively on the structure of the term being substituted instead of the term being substituted into. We then develop another modal language $\lambda_{\Box\bigcirc}{}^{W}$ by extending $\lambda_{\Box\bigcirc}$ with the indexed modalities $\Box_\omega$ and $\bigcirc_\omega$.

We also present a network operational semantics for $\lambda_{\Box\bigcirc}{}^{W}$ which is capable of modeling distributed computations. We demonstrate the use of modal types in the network operational semantics with three communication constructs: *remote evaluation*, *futures*, and *asynchronous channels*. The safety of mobile terms and mobile values is shown by the type safety of the network operational semantics, *i.e.*, its type preservation and progress properties.

Depending on the degree of code mobility and data mobility, languages for distributed computation are classified into four paradigms: *client/server*, *remote evaluation*, *code on demand*, and *mobile agents* [4]. The client/server paradigm allows only data to be transmitted to remote nodes. The remote evaluation paradigm extends the client/server paradigm by allowing both code and data to be transmitted to remote nodes. The code on demand paradigm is similar to the remote evaluation paradigm, but both code and data are fetched from remote nodes. In the mobile agents paradigm, autonomous code migrates to remote nodes by itself and also carries its state. $\lambda_{\Box\bigcirc}{}^{W}$ belongs to the remote evaluation paradigm as its primary capability is to transmit and evaluate mobile terms at remote nodes. The two modalities $\Box$ and $\bigcirc$ deal with *name resolution* [5], a safety issue in languages for distributed computation.

This paper is organized as follows. In Section 2, we develop the modal language $\lambda_{\Box\bigcirc}$. In Section 3, we develop the modal language $\lambda_{\Box\bigcirc}{}^{W}$. In Section 4, we present the network operational semantics and prove its type safety. Section 5 discusses how to handle local resources in distributed computations and compares $\lambda_{\Box\bigcirc}{}^{W}$ with other modal languages for distributed computation. Section 6 concludes with future work. See Appendix for details of all proofs.

2 Modal Language $\lambda_{\Box\bigcirc}$

Since $\lambda_{\Box\bigcirc}$ is an extension of the λ-calculus, we first review the type system of the λ-calculus in the context of distributed computations.

The syntax of the λ-calculus is standard; we use metavariables A, B for types and M, N for terms:

$$\begin{array}{llcl}
\text{type} & A & ::= & A \supset A \\
\text{term} & M & ::= & x \mid \lambda x\!:\!A.\, M \mid M\ M \\
\text{value} & V & ::= & \lambda x\!:\!A.\, M \\
\text{typing context} & \Gamma & ::= & \cdot \mid \Gamma, x : A
\end{array}$$

A variable x with binding $x : A$ is assumed to hold a term and is not regarded as a value. We use a typing judgment $\Gamma \vdash M : A$ to mean that term M has type A under typing context Γ:

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \mathsf{Var} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\!:\!A.\, M : A \supset B}\ {\supset}\mathsf{I} \qquad \frac{\Gamma \vdash M : A \supset B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}\ {\supset}\mathsf{E}$$

The β-reduction rule for the connective $\supset$ uses a capture-avoiding substitution $[M/x]N$ defined in a standard way:

$$(\lambda x\!:\!A.\, N)\ M \quad \rightarrow_{\beta\supset} \quad [M/x]N$$

It may be seen as the reduction of a typing derivation in which the introduction rule $\supset$I is followed by the elimination rule $\supset$E. The following proposition shows that the reduction is indeed type-preserving:

Proposition 2.1. *If* $\Gamma \vdash M : A$ *and* $\Gamma, x : A \vdash N : B$*, then* $\Gamma \vdash [M/x]N : B$*.*

In the context of distributed computations, $x : A$ in a typing context Γ means that variable x holds a term of type A that is valid at a hypothetical node where typechecking takes place, which we call the *current node* throughout the paper. Then a typing judgment $\Gamma \vdash M : A$ means that if typing context Γ is satisfied, the evaluation of term M at the current node returns a value V of type A. It does not, however, tell us if M is a mobile term that can be evaluated at other nodes. Nor does it tell us if V is a mobile value that is valid at other nodes. Therefore the above type system is not expressive enough for the safety of mobile terms and mobile values in distributed computations.

We first develop a modal language $\lambda_{\Box}$ which extends the λ-calculus with the modality $\Box$ to ensure the safety of mobile terms. $\lambda_{\Box}$ is based upon the type system for necessity of modal logic by Pfenning and Davies [14]. Next we develop another modal language $\lambda_{\bigcirc}$ which extends the λ-calculus with the modality $\bigcirc$ to ensure the safety of mobile values. $\lambda_{\Box}$ and $\lambda_{\bigcirc}$ extend the λ-calculus in an orthogonal way: the modality $\Box$ is concerned with *where we can evaluate a given term* whereas the modality $\bigcirc$ is concerned with *where we can use the result of evaluating a given term.* Thus we merge $\lambda_{\Box}$ and $\lambda_{\bigcirc}$ to obtain the modal language $\lambda_{\Box\bigcirc}$, which ensures the safety of both mobile terms and mobile values.

2.1 $\lambda_{\Box}$ for term mobility

The idea behind the modality $\Box$ is that if a term M is well-typed under an empty typing context, *i.e.*, $\cdot \vdash M : A$, we can evaluate it at any node. Intuitively M is valid at any node, or *globally valid*, because it does not depend on any local resource. Thus we use M in building a value $\mathsf{box}\ M$ of modal type $\Box A$.

The syntax of $\lambda_{\Box}$ is as follows:

$$\begin{array}{llll} \text{type} & A & ::= & \cdots \mid \Box A \\ \text{term} & M & ::= & \cdots \mid \mathsf{box}\ M \mid \mathsf{letbox}\ x = M\ \mathsf{in}\ M \\ \text{value} & V & ::= & \cdots \mid \mathsf{box}\ M \end{array}$$

If M evaluates to $\mathsf{box}\ M'$, then $\mathsf{letbox}\ x = M\ \mathsf{in}\ N$ substitutes M', *without evaluating it*, for x in N.

Now a variable x can hold a term that is globally valid (*e.g.*, $\mathsf{letbox}\ x = \mathsf{box}\ M\ \mathsf{in}\ N$). Accordingly we introduce a *mobile typing context* Δ. Γ is now called a *local typing context*.

$$\begin{array}{llll} \text{mobile typing context} & \Delta & ::= & \cdot \mid \Delta, x :: A \\ \text{local typing context} & \Gamma & ::= & \cdot \mid \Gamma, x : A \end{array}$$

$x :: A$ in Δ means that variable x holds a globally valid term of type A; hence a mobile typing context does not affect the mobility of a term being typechecked.

We use a typing judgment $\Delta; \Gamma \vdash M : A$ to mean that under mobile typing context Δ and local typing context Γ, term M evaluates to a value of type A valid at the current node.

$$\frac{x :: A \in \Delta \quad \text{or} \quad x : A \in \Gamma}{\Delta; \Gamma \vdash x : A}\ \mathsf{Cvar} \qquad \frac{\Delta; \cdot \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\ M : \Box A}\ \Box\mathsf{I} \qquad \frac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N : B}\ \Box\mathsf{E}$$

The rule Cvar replaces the rule Var. The rule $\Box$I implies that M is globally valid if it is well-typed under an empty local typing context and thus no assumption is made on the current node. Therefore the premise of the rule $\Box$I implicitly uses an arbitrary node as the current node in typechecking term M.

The β-reduction rule for the modality $\Box$ uses a capture-avoiding substitution $[M/x]N$ extended in a standard way:

$$\mathsf{letbox}\ x = \mathsf{box}\ M\ \mathsf{in}\ N \quad \rightarrow_{\beta\Box} \quad [M/x]N$$

As with the connective $\supset$, this β-reduction rule may be seen as the reduction of a typing derivation in which the introduction rule $\Box$I is followed by the elimination rule $\Box$E. The following proposition shows that the reduction is indeed type-preserving:

Proposition 2.2. *If* $\Delta; \cdot \vdash M : A$ *and* $\Delta, x :: A; \Gamma \vdash N : B$*, then* $\Delta; \Gamma \vdash [M/x]N : B$.

2.2 $\lambda_{\bigcirc}$ for value mobility

The typing judgment of the λ-calculus determines if a term is valid at a given node; if the term is well-typed, it evaluates to a value valid at that node. In contrast, the type system of $\lambda_{\bigcirc}$ should be able to check if the value to which a term evaluates is valid at a given node. This is a property that cannot be verified by the type system of the λ-calculus. Therefore we need an additional typing judgment for the type system of $\lambda_{\bigcirc}$.

As in the type system of $\lambda_{\Box}$, we split a typing context into two parts. We also introduce a new form of binding $v \sim A$:

$$\begin{array}{llll} \text{mobile typing context} & \Delta & ::= & \cdot \mid \Delta, v \sim A \\ \text{local typing context} & \Gamma & ::= & \cdot \mid \Gamma, x : A \end{array}$$

v is called a *value variable* and holds a value; hence it itself is also regarded as a value. $v \sim A$ in Δ means that v holds a globally valid value of type A.

We use a typing judgment $\Delta; \Gamma \vdash M \sim A$ to mean that M evaluates to a globally valid value of type A. In order to express that the value is valid at the current node, we use an ordinary typing judgment $\Delta; \Gamma \vdash M : A$. For any language construct producing local resources, we can use only an ordinary typing judgment (*e.g.*, for a memory allocation construct which returns pointers to heap cells).

The following typing rules hold independently of the syntax of $\lambda_{\bigcirc}$:

$$\frac{v \sim A \in \Delta}{\Delta; \Gamma \vdash v : A}\ \mathsf{Vvar} \qquad \frac{\Delta; \cdot \vdash V : A}{\Delta; \Gamma \vdash V \sim A}\ \mathsf{Val}$$

The rule Vvar says that a value variable in $v \sim A$ is valid at the current node. The rule Val conforms to the definition of the new typing judgment: the premise of the rule Val checks if V is globally valid, in which case the conclusion holds because V is already a value.

The syntax of $\lambda_{\bigcirc}$ is as follows:

$$\begin{array}{llcl} \text{type} & A & ::= & \cdots \mid \bigcirc A \\ \text{term} & M & ::= & \cdots \mid v \mid \mathsf{cir}\ M \mid \mathsf{letcir}\ v = M\ \mathsf{in}\ M \\ \text{value} & V & ::= & \cdots \mid v \mid \mathsf{cir}\ M \end{array}$$

$\mathsf{cir}\ M$ has a modal type $\bigcirc A$, where M evaluates to a globally valid value. $\mathsf{letcir}\ v = M\ \mathsf{in}\ N$ expects M to evaluate to $\mathsf{cir}\ M'$; it conceptually finishes the evaluation of M' before substituting the resultant value for v in N, since v holds a value.

$\mathsf{cir}\ M$ corresponds to the introduction rule for the modality $\bigcirc$. Note that in $\mathsf{letcir}\ v = M\ \mathsf{in}\ N$, the type of M does not determine the form of the typing judgment for the whole term. That is, regardless of the type of M, there are two possibilities for where the result of evaluating N is valid: at the current node and at any node. Therefore each instance of the modality $\bigcirc$ has one introduction rule and two elimination rules:

$$\frac{\Delta; \Gamma \vdash M \sim A}{\Delta; \Gamma \vdash \mathsf{cir}\ M : \bigcirc A}\ \bigcirc\mathsf{I} \qquad \begin{array}{c} \dfrac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N : B}\ \bigcirc\mathsf{E} \\ \dfrac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B}\ \bigcirc\mathsf{E}' \end{array}$$

The β-reduction rule for the modality $\bigcirc$ reduces $\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N$. In this case, we analyze M instead of N. The reason is that only a value can be substituted for v, but M may not be a value; therefore we analyze M to decide how to transform the whole term so that v is eventually replaced by a value. Conceptually N should be replicated at those places within M where the evaluation of M is finished, so that M and N are evaluated exactly once and in that order. If M is already a value V, we reduce the whole term into $[V/v]N$. Thus we are led to define a new form of substitution $\langle M/v \rangle N$ which is defined inductively on the structure of M instead of N, and use it in the β-reduction rule for the modality $\bigcirc$:

$$\begin{array}{rcl} \langle V/v \rangle N & = & [V/v]N \\ \langle \mathsf{letcir}\ v' = M\ \mathsf{in}\ M'/v \rangle N & = & \mathsf{letcir}\ v' = M\ \mathsf{in}\ \langle M'/v \rangle N \end{array}$$

$$\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N \quad \to_{\beta\bigcirc} \quad \langle M/v \rangle N$$

Note that we do not define $\langle M\ M'/v \rangle N$ because $\mathsf{cir}\ M\ M'$ cannot be well-typed: there is no derivation of $\Delta; \Gamma \vdash M\ M' \sim A$, which would require us to refine types of lambda abstractions. In practice, ordinary type $A \supset \bigcirc B$ for M suffices in conjunction with $\mathsf{letcir}\ v = M\ M'\ \mathsf{in}\ v$ to simulate such a derivation.

As with the connective $\supset$, the β-reduction rule may be seen as the reduction of a typing derivation in which the introduction rule $\bigcirc\mathsf{I}$ is followed by the elimination rule $\bigcirc\mathsf{E}$. The following proposition shows that the reduction is indeed type-preserving:

Proposition 2.3.
If $\Delta; \Gamma \vdash M \sim A$ *and* $\Delta, v \sim A; \Gamma \vdash N : C$*, then* $\Delta; \Gamma \vdash \langle M/v \rangle N : C$.
If $\Delta; \Gamma \vdash M \sim A$ *and* $\Delta, v \sim A; \Gamma \vdash N \sim C$*, then* $\Delta; \Gamma \vdash \langle M/v \rangle N \sim C$.

Proof. By induction on the structure of M (not N). □

2.3 $\lambda_{\Box\bigcirc}$ for term mobility and value mobility

$\lambda_{\Box\bigcirc}$ is a modal language which incorporates both $\lambda_{\Box}$ and $\lambda_{\bigcirc}$. Since $\lambda_{\Box}$ and $\lambda_{\bigcirc}$ are orthogonal extensions of the λ-calculus, all their individual properties continue to hold in $\lambda_{\Box\bigcirc}$.

We decide to allow $\mathsf{letbox}\ x = M\ \mathsf{in}\ N$ in the typing judgment for value mobility. The decision is based upon the observation that a substitution of a mobile term for x does not prevent N from evaluating to a mobile value. For example, x may not appear in N at all. Therefore we introduce a new elimination rule for the modality $\Box$ as follows:

$$\frac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B}\ \Box\mathsf{E}'$$

Since $\mathsf{cir}\ \mathsf{letbox}\ x = M\ \mathsf{in}\ M'$ can now be well-typed, we define $\langle \mathsf{letbox}\ x = M\ \mathsf{in}\ M'/v \rangle N$:

$$\langle \mathsf{letbox}\ x = M\ \mathsf{in}\ M'/v \rangle N \quad = \quad \mathsf{letbox}\ x = M\ \mathsf{in}\ \langle M'/v \rangle N$$

An easy induction shows that Proposition 2.3 continues to hold. The following proposition shows that the β-reduction rule for the modality $\Box$ continues to be type-preserving:

Proposition 2.4. *If* $\Delta; \cdot \vdash M : A$ *and* $\Delta, x : A; \Gamma \vdash N \sim B$*, then* $\Delta; \Gamma \vdash [M/x]N \sim B$.

2.4 Primitive types

A primitive type is one for which value mobility is an inherent property. For example, a boolean value, of type bool, is atomic and does not contain references to local resources. Therefore boolean values are always globally valid and $\Delta; \Gamma \vdash M : \mathsf{bool}$ semantically implies $\Delta; \Gamma \vdash M \sim \mathsf{bool}$. Under the above type system, however, value mobility for primitive types should be expressed explicitly by programmers.

As an example, consider a primitive type nat for natural numbers:

$$\begin{array}{lrcl} \text{type} & A & ::= & \cdots \mid \mathsf{nat} \\ \text{term} & M & ::= & \cdots \mid \mathsf{zero} \mid \mathsf{succ}\ M \\ \text{value} & V & ::= & \cdots \mid \mathsf{zero} \mid \mathsf{succ}\ V \end{array}$$

We use the following construct for primitive recursion over nat:

$$\text{term} \quad M \quad ::= \quad \cdots \mid \mathsf{rec}\ M\ \mathsf{of}\ f(\mathsf{zero}) \Rightarrow M\ /\ f(\mathsf{succ}\ x) \Rightarrow M$$

$$\frac{\begin{array}{l}\Delta; \Gamma \vdash M : \mathsf{nat} \\ \Delta; \Gamma \vdash M_0 : A \\ \Delta; \Gamma, x : \mathsf{nat}, f(x) : A \vdash M_1 : A\end{array}}{\begin{array}{c}\Delta; \Gamma \vdash \mathsf{rec}\ M\ \mathsf{of}\ f(\mathsf{zero}) \Rightarrow M_0 \quad : A \\ /\ f(\mathsf{succ}\ x) \Rightarrow M_1\end{array}}\ \mathsf{Rec} \qquad \frac{\begin{array}{l}\Delta; \Gamma \vdash M : \mathsf{nat} \\ \Delta; \Gamma \vdash M_0 \sim A \\ \Delta, f(x) \sim A; \Gamma, x : \mathsf{nat} \vdash M_1 \sim A\end{array}}{\begin{array}{c}\Delta; \Gamma \vdash \mathsf{rec}\ M\ \mathsf{of}\ f(\mathsf{zero}) \Rightarrow M_0 \quad \sim A \\ /\ f(\mathsf{succ}\ x) \Rightarrow M_1\end{array}}\ \mathsf{Rec}'$$

Now, for any term M such that $\Delta; \Gamma \vdash M : \mathsf{nat}$, we explictly express its value mobility with the following term $M^{\sim}$, which evaluates to the same value as M and also satisfies $\Delta; \Gamma \vdash M^{\sim} \sim \mathsf{nat}$:

$$M^{\sim} \ = \ \mathsf{rec}\ M\ \mathsf{of}\ f(\mathsf{zero}) \Rightarrow \mathsf{zero}\ /\ f(\mathsf{succ}\ x) \Rightarrow \mathsf{letcir}\ v = \mathsf{cir}\ f(x)\ \mathsf{in}\ \mathsf{succ}\ v$$

$$
\begin{array}{llll}
\text{type} & A & ::= & A \supset A \mid \Box A \mid \bigcirc A \\
\text{term} & M & ::= & x \mid \lambda x\!:\!A.\, M \mid M\ M \mid \mathsf{box}\ M \mid \mathsf{letbox}\ x = M\ \mathsf{in}\ M \mid \\
& & & v \mid \mathsf{cir}\ M \mid \mathsf{letcir}\ v = M\ \mathsf{in}\ M \\
\text{value} & V & ::= & \lambda x\!:\!A.\, M \mid \mathsf{box}\ M \mid v \mid \mathsf{cir}\ M
\end{array}
$$

$$
\frac{x :: A \in \Delta \quad \text{or} \quad x : A \in \Gamma}{\Delta; \Gamma \vdash x : A}\ \mathsf{Cvar}
\qquad
\frac{v \sim A \in \Delta}{\Delta; \Gamma \vdash v : A}\ \mathsf{Vvar}
\qquad
\frac{\Delta; \cdot \vdash V : A}{\Delta; \Gamma \vdash V \sim A}\ \mathsf{Val}
$$

$$
\frac{\Delta; \Gamma, x : A \vdash M : B}{\Delta; \Gamma \vdash \lambda x\!:\!A.\, M : A \supset B}\ {\supset}\mathsf{I}
\qquad
\frac{\Delta; \Gamma \vdash M : A \supset B \quad \Delta; \Gamma \vdash N : A}{\Delta; \Gamma \vdash M\ N : B}\ {\supset}\mathsf{E}
$$

$$
\frac{\Delta; \cdot \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}
\qquad
\frac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N : B}\ \Box\mathsf{E}
$$

$$
\frac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B}\ \Box\mathsf{E}'
$$

$$
\frac{\Delta; \Gamma \vdash M \sim A}{\Delta; \Gamma \vdash \mathsf{cir}\ M : \bigcirc A}\ \bigcirc\mathsf{I}
\qquad
\frac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N : B}\ \bigcirc\mathsf{E}
$$

$$
\frac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B}\ \bigcirc\mathsf{E}'
$$

$$
\frac{\Delta; \Gamma \vdash M : A_{prim}}{\Delta; \Gamma \vdash M \sim A_{prim}}\ \mathsf{Prim}{\sim}
$$

Figure 1: Syntax and type system of $\lambda_{\Box\bigcirc}$.

We choose to take advantage of the fact that every term M of a primitive type can be converted into an equivalent term $M^{\sim}$ with value mobility as illustrated above, and introduce the following typing rule in which value mobility for primitive types is built-in:

$$
\frac{\Delta; \Gamma \vdash M : A_{prim}}{\Delta; \Gamma \vdash M \sim A_{prim}}\ \mathsf{Prim}{\sim}
$$

Here A_{prim} is a primitive type ($A \supset A$, $\Box A$, and $\bigcirc A$ cannot be a primitive type). With the rule Prim$\sim$ in the type system, we can easily express value mobility for primitive types.

The price we pay for the rule Prim$\sim$ is that β-reduction $\rightarrow_{\beta\bigcirc}$ is no longer valid: $\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N$ may typecheck while $\langle M/v \rangle N$ is not defined. For example, $M = M_1\ M_2$ of type nat satisfies $\Delta; \Gamma \vdash M \sim \mathsf{nat}$ by the rule Prim$\sim$, but $\langle M_1\ M_2/v \rangle N$ is not defined. Intuitively the rule Prim$\sim$ disguises an unanalyzable term of a primitive type as an analyzable term.

A quick fix is to reduce $\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N$ only if M is already a value V:

$$
\mathsf{letcir}\ v = \mathsf{cir}\ V\ \mathsf{in}\ N \quad \rightarrow_{\beta\bigcirc} \quad [V/v]N \qquad\qquad (\rightarrow_{\beta\bigcirc} \text{ redefined})
$$

Note that we write $[V/v]N$ for $\langle V/v \rangle N$. Thus, in order to reduce $\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N$, we are forced to reduce M into a value first, instead of analyzing M to transform the whole term. Such a reduction strategy is reflected in the operational semantics, as we will see in Section 4.

Now we have introduced all typing rules of $\lambda_{\Box\bigcirc}$ (See Figure 1.) All the previous propositions, except Proposition 2.3, continue to hold for the type system of $\lambda_{\Box\bigcirc}$. The following proposition proves that $\Delta; \Gamma \vdash M \sim A$ is stronger than $\Delta; \Gamma \vdash M : A$:

Proposition 2.5. *The following typing rule is admissible:*

$$
\frac{\Delta; \Gamma \vdash M \sim A}{\Delta; \Gamma \vdash M : A}\ {\sim}{:}
$$

Proof. By induction on the structure of $\Delta; \Gamma \vdash M \sim A$. $\square$

2.5 Example

To express term mobility and value mobility for each new construct M, we provide a rule for ordinary typing judgment $\Delta; \Gamma \vdash M : A$ and optionally another rule for typing judgment $\Delta; \Gamma \vdash M \sim A$. As an example, consider constructs for memory allocation. We regard a heap cell as a local resource; hence its pointer is assumed to be valid only at the node where it is allocated. We use type $\mathsf{ptr}\ A$ for pointers to heap cells containing values of type A. For the sake of brevity, we do not consider typing rules for pointers.

$$\begin{array}{lll} \text{type} & A & ::= & \mathsf{ptr}\ A \\ \text{term} & M & ::= & \mathsf{new}\ M \mid \mathsf{read}\ M \mid \mathsf{write}\ M\ M \end{array}$$

The three constructs work as follows:

- If M evaluates to a value V, then $\mathsf{new}\ M$ allocates a new heap cell containing V and returns its pointer l.
- If M evaluates to a pointer l, then $\mathsf{read}\ M$ returns the contents of the heap cell pointed to by l.
- If M evaluates to a pointer l and N evaluates to a value V, then $\mathsf{write}\ M\ N$ writes V to the heap cell pointed to by l and returns V.

The rules for the ordinary typing judgment reflect how these three constructs work:

$$\frac{\Delta; \Gamma \vdash M : A}{\Delta; \Gamma \vdash \mathsf{new}\ M : \mathsf{ptr}\ A}\ \mathsf{New} \quad \frac{\Delta; \Gamma \vdash M : \mathsf{ptr}\ A}{\Delta; \Gamma \vdash \mathsf{read}\ M : A}\ \mathsf{Read} \quad \frac{\Delta; \Gamma \vdash M : \mathsf{ptr}\ A \quad \Delta; \Gamma \vdash N : A}{\Delta; \Gamma \vdash \mathsf{write}\ M\ N : A}\ \mathsf{Write}$$

Thus any of these constructs is mobile if its argument is globally valid. For example, $\mathsf{box}\ \mathsf{new}\ M$ (of type $\Box\ \mathsf{ptr}\ A$) typechecks if M is globally valid, which means that allocating a new heap cell itself can be done at any node. Once we finish evaluating $\mathsf{new}\ M$, however, the result is no longer mobile (because it is a pointer), which implies that the following rule is not allowed:

$$\frac{\cdots}{\Delta; \Gamma \vdash \mathsf{new}\ M \sim \mathsf{ptr}\ A}\ (\mathit{wrong})$$

Since the value contained in a heap cell is not necessarily globally valid, we do not allow the following rule:

$$\frac{\cdots}{\Delta; \Gamma \vdash \mathsf{read}\ M \sim A}\ (\mathit{wrong})$$

The following rule is safe to use because $\mathsf{write}\ M\ N$ returns the value to which N evaluates:

$$\frac{\Delta; \Gamma \vdash M : \mathsf{ptr}\ A \quad \Delta; \Gamma \vdash N \sim A}{\Delta; \Gamma \vdash \mathsf{write}\ M\ N \sim A}\ \mathsf{Write}'$$

As an example involving primitive types, let us build a mobile term adding two natural numbers. The following term does not typecheck because variables x and y are not added to the mobile typing context:

$$\lambda x\!:\!\mathsf{nat}.\, \lambda y\!:\!\mathsf{nat}.\, \mathsf{box}\ (x + y)$$

We can make it typecheck by converting x and y into value variables v_x and v_y (using the rule Prim$\sim$):

$$\begin{array}{ll} \lambda x\!:\!\mathsf{nat}.\, \lambda y\!:\!\mathsf{nat}. & \mathsf{letcir}\ v_x = \mathsf{cir}\ x\ \mathsf{in} \\ & \mathsf{letcir}\ v_y = \mathsf{cir}\ y\ \mathsf{in} \\ & \mathsf{box}\ (v_x + v_y) \end{array}$$

The following term copies mobiles terms contained in variables x and y, and the evaluation of the resultant mobile term may take longer than adding two natural numbers:

$$\lambda x\!:\!\Box\mathsf{nat}.\,\lambda y\!:\!\Box\mathsf{nat}.\;\begin{array}{l}\mathsf{letbox}\;x' = x\;\mathsf{in}\\ \mathsf{letbox}\;y' = y\;\mathsf{in}\\ \mathsf{box}\;(x' + y')\end{array}$$

The following term first finishes evaluating mobile terms contained in variables x and y:

$$\lambda x\!:\!\Box\mathsf{nat}.\,\lambda y\!:\!\Box\mathsf{nat}.\;\begin{array}{l}\mathsf{letbox}\;x' = x\;\mathsf{in}\;\mathsf{letcir}\;v_x = \mathsf{cir}\;x'\;\mathsf{in}\\ \mathsf{letbox}\;y' = y\;\mathsf{in}\;\mathsf{letcir}\;v_y = \mathsf{cir}\;y'\;\mathsf{in}\\ \mathsf{box}\;(v_x + v_y)\end{array}$$

2.6 Logic for $\lambda_{\Box\bigcirc}$

Modal types $\Box A$ in $\lambda_{\Box\bigcirc}$ use the same type system for necessity of modal logic of Davies and Pfenning [6, 14]. A minor difference is that our interpretation of the modality $\Box$ is spatial ($\Box A$ means that A is true at every node), whereas their interpretation is temporal or proof-theoretic.

The type system for modal types $\bigcirc A$ is unusual in that it differentiates values (*i.e.*, terms in weak head normal form) from ordinary terms, as shown in the rule Val. This differentiation implies that the logic corresponding to the modality $\bigcirc$ requires a judgment that inspects not only hypotheses in a proof but also inferences rules in it. Thus the modality $\bigcirc$ sets itself apart from other modalities and is not found in any other logic.

A substitution $\langle M/v\rangle N$ for the modality $\bigcirc$ is similar to (and was inspired by) those substitutions for modal possibility and lax truth in [14] in that it is defined inductively on the structure of the term being substituted (*i.e.*, M) instead of the term being substituted into (*i.e.*, N). In fact, we may even think of $\langle M/v\rangle N$ as substituting N into M because conceptually N is replicated at those places within M where the evaluation of M is finished.

We close this section with a discussion of the properties of the modalities $\Box$ and $\bigcirc$. Note that the two modalities interact with each other, although they are developed in an orthogonal way.

- $\Box A \supset A$ $\qquad \lambda x\!:\!\Box A.\,\mathsf{letbox}\;y = x\;\mathsf{in}\;y$
  A mobile term is a special case of an ordinary term.
- $\Box A \supset \Box\Box A$ $\qquad \lambda x\!:\!\Box A.\,\mathsf{letbox}\;y = x\;\mathsf{in}\;\mathsf{box}\;\mathsf{box}\;y$
  A mobile term itself is mobile.
- $\Box(A \supset B) \supset \Box A \supset \Box B$ $\qquad \lambda x\!:\!\Box(A \supset B).\,\lambda y\!:\!\Box A.\,\mathsf{letbox}\;x' = x\;\mathsf{in}\;\mathsf{letbox}\;y' = y\;\mathsf{in}\;\mathsf{box}\;x'\;y'$
- $\bigcirc A \supset A$ $\qquad \lambda x\!:\!\bigcirc A.\,\mathsf{letcir}\;v = x\;\mathsf{in}\;v$
  A mobile value is a special case of an ordinary term.
- $\bigcirc A \supset \bigcirc\bigcirc A$ $\qquad \lambda x\!:\!\bigcirc A.\,\mathsf{letcir}\;v = x\;\mathsf{in}\;\mathsf{cir}\;\mathsf{cir}\;v$
  A mobile value itself is mobile.
- $\bigcirc A \supset \Box A$ $\qquad \lambda x\!:\!\bigcirc A.\,\mathsf{letcir}\;v = x\;\mathsf{in}\;\mathsf{box}\;v$
  A mobile value is a special case of a mobile term.
- $\Box A \supset \bigcirc\Box A$ $\qquad \lambda x\!:\!\Box A.\,\mathsf{letbox}\;y = x\;\mathsf{in}\;\mathsf{cir}\;\mathsf{box}\;y$
  box M is a mobile value.

- $\bigcirc A \supset \Box\bigcirc A$ $\qquad$ $\lambda x\!:\!\bigcirc A.\,\mathsf{letcir}\ v = x\ \mathsf{in}\ \mathsf{box}\ \mathsf{cir}\ v$
  $\mathsf{cir}\ V$ is a mobile term.

- $\Box\bigcirc A \supset \Box A$ $\qquad$ $\lambda x\!:\!\Box\bigcirc A.\,\mathsf{letbox}\ y = x\ \mathsf{in}\ \mathsf{box}\ \mathsf{letcir}\ v = y\ \mathsf{in}\ v$
  (derivable from $\Box\bigcirc A \supset \bigcirc A \supset \Box A$)

- $\bigcirc\Box A \not\supset \bigcirc A$
  If $\bigcirc\Box A \supset \bigcirc A$ held, $\Box A$ and $\bigcirc A$ would be equivalent because of $\bigcirc A \supset \Box A$ and $\Box A \supset \bigcirc\Box A \supset \bigcirc A$.

3 Modal language $\lambda_{\Box\bigcirc}{}^{W}$ with indexed modalities

In the definition of $\lambda_{\Box\bigcirc}$, "mobile" is synonymous with "globally valid": a mobile term or value is valid at any node in the network. Such a model for distributed computation is adequate if all participating nodes are assumed to be homogeneous and have the same permanent local resources. In a grid computing environment, for example, a mobile term valid at a particular remote node is also globally valid and can be evaluated at any other remote node. For a heterogenous group of nodes with different permanent local resources, however, $\lambda_{\Box\bigcirc}$ becomes inadequate because a mobile term or value is not always globally valid. For example, a client node may transmit to a printer server a "mobile" term for printing a document; such a mobile term can be evaluated only at printer servers and is not globally valid. Since this notion of restricted mobility is useful in practice, we extend $\lambda_{\Box\bigcirc}$ to allow terms and values valid only at specific nodes.

The main design issue is whether or not the type system specifies a node at which a mobile term or value is valid. As an example, consider a mobile term M that is valid only at printer servers (*e.g.*, for printing a document). There are two approaches to expressing its mobility with a type. In one approach, the type system does not specify the node at which M is to be evaluated; instead it only indicates that there exists a certain node at which M can be evaluated. In this case, it is the linker or the runtime system that decides where to evaluate such a mobile term. In the other approach, the type system specifies explicitly the node at which M is to be evaluated. In this case, it is the type system that decides where to evaluate such a mobile term.

The first approach is attractive because the type system abstracts from any particular network configuration. For example, new printer servers can be deployed into the network and old printer servers can be removed without changing the type system. The second approach is useful if the network configuration is static. For example, if the set of available printer servers is published and never changes, programmers can specify a printer server with an appropriate type involving its identifier. In this paper, we adopt the second approach to extend $\lambda_{\Box\bigcirc}$ and leave it as future work to apply the first approach.

We extend $\lambda_{\Box\bigcirc}$ with two indexed modalities $\Box_\omega$ and $\bigcirc_\omega$ with the following interpretation:

- A value $\mathsf{box}_\omega\ M$ of indexed modal type $\Box_\omega A$ contains term M which is valid at node ω.

- A value $\mathsf{cir}_\omega\ M$ of indexed modal type $\bigcirc_\omega A$ contains term M which evaluates to a value valid at node ω.

Since the type system of $\lambda_{\Box\bigcirc}$ is incapable of expressing properties of a term with respect to specific nodes, we replace the typing judgments of $\lambda_{\Box\bigcirc}$ by a new form of typing judgment $\Delta;\Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$:

- $\Delta;\Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$ means that under mobile typing context Δ and local typing context Γ, term M at node ω evaluates to a value of type A valid at node ω'.

- $\Delta;\Gamma \vdash_\omega M : A$ is a shorthand for $\Delta;\Gamma \vdash_\omega M \sim A \mathbin{@} \omega$, where ω may be thought of as the current node for typechecking M. Note that it is *not* a separate judgment.

A mobile typing context Δ is defined as before, but a local typing context Γ now contains only those binding relativized to a specific node:

$$\begin{array}{llcl} \text{mobile typing context} & \Delta & ::= & \cdot \mid \Delta, x :: A \mid \Delta, v \sim A \\ \text{local typing context} & \Gamma & ::= & \cdot \mid \Gamma, x : A \mathbin{@} \omega \mid \Gamma, v \sim A \mathbin{@} \omega \end{array}$$

- $x :: A$ in Δ means that x holds a globally valid term of type A.
- $v \sim A$ in Δ means that v holds a globally valid value of type A.
- $x : A \mathbin{@} \omega$ in Γ means that x holds a term valid at node ω.
- $v \sim A \mathbin{@} \omega$ in Γ means that v holds a value valid at node ω.

Note that the use of typing judgment $\Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$ implies that a term may evaluate to a value that is *not* valid at the node at which it is evaluated. For example, a term may scan a list of handles for remote files and select one; the evaluation is safe as long as the selected handle is not dereferenced. We refer to our new modal language with indexed modalities as $\lambda_{\Box\bigcirc}{}^W$.

The syntax of $\lambda_{\Box\bigcirc}{}^W$ is as follows:

$$\begin{array}{llcl} \text{type} & A & ::= & A \supset A \mid \Box A \mid \Box_\omega A \mid \bigcirc A \mid \bigcirc_\omega A \\ \text{term} & M & ::= & x \mid \lambda x \mathbin{:} A.\, M \mid M\ M \mid \mathsf{box}\ M \mid \mathsf{box}_\omega\ M \mid \mathsf{letbox}\ x = M\ \mathsf{in}\ M \mid \\ & & & v \mid \mathsf{cir}\ M \mid \mathsf{cir}_\omega\ M \mid \mathsf{letcir}\ v = M\ \mathsf{in}\ M \\ \text{value} & V & ::= & \lambda x \mathbin{:} A.\, M \mid \mathsf{box}\ M \mid \mathsf{box}_\omega\ M \mid v \mid \mathsf{cir}\ M \mid \mathsf{cir}_\omega\ M \end{array}$$

For the sake of simplicity, we reuse $\mathsf{letbox}\ x = M\ \mathsf{in}\ N$ and $\mathsf{letcir}\ v = M\ \mathsf{in}\ N$ to expose terms inside $\mathsf{box}_\omega\ M'$ and $\mathsf{cir}_\omega\ M'$ (as well as $\mathsf{box}\ M'$ and $\mathsf{cir}\ M'$). Thus both $\mathsf{letbox}\ x = \mathsf{box}\ M'\ \mathsf{in}\ N$ and $\mathsf{letbox}\ x = \mathsf{box}_\omega\ M'\ \mathsf{in}\ N$ substitute M' for x in N; similarly both $\mathsf{letcir}\ v = \mathsf{cir}\ M'\ \mathsf{in}\ N$ and $\mathsf{letcir}\ v = \mathsf{cir}_\omega\ M'\ \mathsf{in}\ N$ first reduce M' to a value, which is then substituted for v in N.

Figure 2 shows the typing rules of $\lambda_{\Box\bigcirc}{}^W$. All these typing rules look similar to those of $\lambda_{\Box\bigcirc}$, except that we explicitly annotate every typing judgment with a node at which the evaluation is to take place and another node at which its end result is valid. For each form V of value, we provide a typing rule for the judgment $\Delta; \Gamma \vdash_\omega V : A$ only; in order to decide where else V is valid, we use the rule Val_W. Note that in the rule $\Box\mathsf{I}_W$, the local typing context Γ of the conclusion is carried over to the premise (whereas in the rule $\Box\mathsf{I}$ of $\lambda_{\Box\bigcirc}$, it is replaced by an empty local typing context). This is safe because an arbitrary node ω' (instantiated by *fresh* ω') serves as the current node in the premise.

The rules Cvar_W and Vvar_W prevent references to local resources from being dereferenced at remote nodes. Suppose $x : A \mathbin{@} \omega \in \Gamma$, $v \sim A \mathbin{@} \omega \in \Gamma$, and $\omega' \neq \omega$. In order to "evaluate" the term in x (which perhaps contains references to local resources belonging to ω) at ω', we should be able to derive $\Delta; \Gamma \vdash_{\omega'} x \sim A \mathbin{@} \omega''$ for a certain node ω'', which is impossible because of the rule Cvar_W; in order to "use" the value in v (which is perhaps a reference to a local resource belonging to ω) at ω', we should be able to derive $\Delta; \Gamma \vdash_{\omega'} v : A$, which is impossible because of the rule Vvar_W. Note, however, that we can derive $\Delta; \Gamma \vdash_{\omega'} v \sim A \mathbin{@} \omega$, which implies that a reference to a local resource may be present at remote nodes as long as it is not dereferenced.

As value mobility for primitive types is built-in in the rule $\mathsf{Prim}{\sim}_W$, we reduce $\mathsf{letcir}\ v = \mathsf{cir}\ M\ \mathsf{in}\ N$ and $\mathsf{letcir}\ v = \mathsf{cir}_\omega\ M\ \mathsf{in}\ N$ only if M is already a value, as in $\lambda_{\Box\bigcirc}$. Thus all β-reduction rules are defined in terms of an ordinary substitution $[M/x]N$ or $[V/v]N$:

$$\begin{array}{rcl} (\lambda x \mathbin{:} A.\, N)\ M & \rightarrow_{\beta\supset} & [M/x]N \\ \mathsf{letbox}\ x = \mathsf{box}\ M\ \mathsf{in}\ N & \rightarrow_{\beta\Box} & [M/x]N \\ \mathsf{letbox}\ x = \mathsf{box}_\omega\ M\ \mathsf{in}\ N & \rightarrow_{\beta\Box'} & [M/x]N \\ \mathsf{letcir}\ v = \mathsf{cir}\ V\ \mathsf{in}\ N & \rightarrow_{\beta\bigcirc} & [V/v]N \\ \mathsf{letcir}\ v = \mathsf{cir}_\omega\ V\ \mathsf{in}\ N & \rightarrow_{\beta\bigcirc'} & [V/v]N \end{array}$$

$$\frac{x :: A \in \Delta \quad \text{or} \quad x : A \mathbin{@} \omega \in \Gamma}{\Delta; \Gamma \vdash_\omega x : A}\ \mathsf{Cvar}_W \qquad \frac{v \sim A \in \Delta \quad \text{or} \quad v \sim A \mathbin{@} \omega \in \Gamma}{\Delta; \Gamma \vdash_\omega v : A}\ \mathsf{Vvar}_W$$

$$\frac{\Delta; \Gamma \vdash_{\omega'} V : A}{\Delta; \Gamma \vdash_\omega V \sim A \mathbin{@} \omega'}\ \mathsf{Val}_W\ (\omega \neq \omega')$$

$$\frac{\Delta; \Gamma, x : A \mathbin{@} \omega \vdash_\omega M : B}{\Delta; \Gamma \vdash_\omega \lambda x\!:\!A.\, M : A \supset B}\ {\supset}\mathsf{I}_W \qquad \frac{\Delta; \Gamma \vdash_\omega M : A \supset B \quad \Delta; \Gamma \vdash_\omega N : A}{\Delta; \Gamma \vdash_\omega M\ N : B}\ {\supset}\mathsf{E}_W$$

$$\frac{\mathit{fresh}\ \omega' \quad \Delta; \Gamma \vdash_{\omega'} M : A}{\Delta; \Gamma \vdash_\omega \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}_W \qquad \frac{\Delta; \Gamma \vdash_\omega M : \Box A \quad \Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \Box\mathsf{E}_W$$

$$\frac{\Delta; \Gamma \vdash_{\omega'} M : A}{\Delta; \Gamma \vdash_\omega \mathsf{box}_{\omega'}\ M : \Box_{\omega'} A}\ \Box\mathsf{I}'_W \qquad \frac{\Delta; \Gamma \vdash_\omega M : \Box_{\omega''} A \quad \Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \Box\mathsf{E}'_W$$

$$\frac{\mathit{fresh}\ \omega' \quad \Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{cir}\ M : \bigcirc A}\ \bigcirc\mathsf{I}_W \qquad \frac{\Delta; \Gamma \vdash_\omega M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \bigcirc\mathsf{E}_W$$

$$\frac{\Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{cir}_{\omega'}\ M : \bigcirc_{\omega'} A}\ \bigcirc\mathsf{I}'_W \qquad \frac{\Delta; \Gamma \vdash_\omega M : \bigcirc_{\omega''} A \quad \Delta; \Gamma, v \sim A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'}{\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \bigcirc\mathsf{E}'_W$$

$$\frac{\Delta; \Gamma \vdash_\omega M : A_{prim}}{\Delta; \Gamma \vdash_\omega M \sim A_{prim} \mathbin{@} \omega'}\ \mathsf{Prim}{\sim}_W\ (\omega \neq \omega')$$

Figure 2: Typing rules of $\lambda_{\Box\bigcirc}^{W}$.

The following propositions imply that all these β-reductions are type-preserving:

Proposition 3.1. *If* $\Delta; \Gamma \vdash_{\omega''} M : A$ *and* $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Proposition 3.2. *If* $\Delta; \Gamma \vdash_{\omega''} M : A$ *for any node* ω'' *and* $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Proposition 3.3. *If* $\Delta; \Gamma \vdash_{\omega''} V : A$ *and* $\Delta; \Gamma, v \sim A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \mathbin{@} \omega'$.

Proposition 3.4. *If* $\Delta; \Gamma \vdash_{\omega''} V : A$ *for any node* ω'' *and* $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \mathbin{@} \omega'$.

3.1 $\lambda_{\Box\bigcirc}^{W}$ as an extension of $\lambda_{\Box\bigcirc}$

Since all the β-reduction rules of $\lambda_{\Box\bigcirc}$ are included in $\lambda_{\Box\bigcirc}^{W}$, any reduction sequence in $\lambda_{\Box\bigcirc}$ is also valid in $\lambda_{\Box\bigcirc}^{W}$. All the typing rules of $\lambda_{\Box\bigcirc}$ can also be rewritten in terms of typing judgments in $\lambda_{\Box\bigcirc}^{W}$. Intuitively $\Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$ is more expressive than $\Delta; \Gamma \vdash M : A$ and $\Delta; \Gamma \vdash M \sim A$ because ω and ω' can be instantated into arbitrary nodes. Given a local typing context Γ in $\lambda_{\Box\bigcirc}$, we write $[\Gamma]^\omega$ for a local typing context in $\lambda_{\Box\bigcirc}^{W}$ that attaches $@\ \omega$ to every binding $x : A$ in Γ:

$$[\Gamma]^\omega \ = \ \{x : A \mathbin{@} \omega \mid x : A \in \Gamma\}$$

The following proposition shows how to interpret typing judgments in $\lambda_{\Box\bigcirc}$ in terms of those in $\lambda_{\Box\bigcirc}^{W}$:

Proposition 3.5.
If $\Delta; \Gamma \vdash M : A$*, then* $\Delta; [\Gamma]^\omega \vdash_\omega M : A$ *for any node* ω.
If $\Delta; \Gamma \vdash M \sim A$*, then* $\Delta; [\Gamma]^\omega \vdash_\omega M \sim A \mathbin{@} \omega'$ *for any nodes* ω *and* ω'.

3.2 Logic for $\lambda_{\Box\bigcirc}{}^{W}$

As every typing judgment in $\lambda_{\Box\bigcirc}{}^{W}$ is relative to a certain node, the logic for $\lambda_{\Box\bigcirc}{}^{W}$ requires judgments relativized to nodes. For example, $x : A \mathbin{@} \omega$ in a local typing context corresponds to a judgment that A is true at node ω. Since the indexed modalities $\Box_\omega$ and $\bigcirc_\omega$ directly internalize nodes within propositions, the logic for $\lambda_{\Box\bigcirc}{}^{W}$ is a restricted form of hybrid logic [2].

The notion of judgment relativized to nodes is also a suitable basis for the semantics of modal logic. For example, Simpson [15] provides a natural deduction system for intuitionistic modal logic based upon relative truth. The fragment of $\lambda_{\Box\bigcirc}{}^{W}$ without the indexed modalities can be explained in a similar way, with the assumption that all nodes are visible (or accessible) from each other. This assumption is justified because in a distribution computation, all nodes can communicate with each other.

The type system presented in this section is appropriate for understanding the roles of the modalities $\Box$ and $\bigcirc$ and the indexed modalities $\Box_\omega$ and $\bigcirc_\omega$. It is not, however, expressive enough for distributed computations in which communication constructs may generate terms whose type is determined by *remote nodes*. For example, a synchronization variable produced by a future construct (to be explained in the next section) is essentially a pointer to a remote node, which determines its type. In the next section, we extend the type system of $\lambda_{\Box\bigcirc}{}^{W}$ so that we can typecheck such terms, and also develop a network operational semantics which is capable of modeling distributed computations.

4 $\lambda_{\Box\bigcirc}{}^{W}$ for distributed computation

In this section, we develop an extended type system and a network operational semantics for $\lambda_{\Box\bigcirc}{}^{W}$. We demonstrate the use of modal types with three communication constructs: remote evaluation, futures, and asynchronous channels. We prove the type safety of the network operational semantics, *i.e.*, its type preservation and progress properties, in the presence of these communication constructs. The type safety implies the safety of mobile terms and mobile values.

4.1 Physical nodes and logical nodes

So far, we have restricted ourselves to physical nodes by interpreting ω as an identifier of a physical node. For example, ω may refer to a printer server or a database server. While appropriate for the type system, this interpretation poses a problem when we model distributed computations. For example, if a database server initiates a stand-alone computation for each query it receives, we cannot distinguish between these stand-alone computations with different node identifiers. Therefore there arises a need for *logical nodes*, each of which performs a single stand-alone computation. In order for a physical node to perform multiple stand-alone computations concurrently, it spawns the same number of logical nodes.

We distinguish between physical nodes and logical nodes as separate syntactic categories:

$$\begin{array}{ll} \text{physical node} & \omega \\ \text{logical node} & \gamma \end{array}$$

A logical node on physical node ω inherits all permanent local resources belonging to ω. Therefore a term valid at physical node ω is valid at every logical node on ω.

We assume two primitives, $new\ \gamma$ and $new\ \gamma \mathbin{@} \omega$, for creating logical nodes. $\mathcal{P}(\gamma)$ stands for the physical node with which logical node γ is associated, as defined below. Note that it is not defined as the actual physical node where logical node γ resides:

- $new\ \gamma$ creates a new logical node γ which may reside at an arbitrary physical node (including the physical node invoking $new\ \gamma$ itself). If γ is created with $new\ \gamma$, then $\mathcal{P}(\gamma)$ is a fresh physical node

ω (which is different from any existing physical node).
Example: $new\ \gamma$ searches for an idle computer in the network and establishes a logical node γ on it.

- $new\ \gamma \mathbin{@} \omega$ creates a new logical node γ at physical node ω. If γ is created with $new\ \gamma \mathbin{@} \omega$, then $\mathcal{P}(\gamma) = \omega$.
  Example: $new\ \gamma \mathbin{@} \omega$ contacts a database server ω and requests a logical node γ on it.

We assume that every physical node ω publishes a local typing context $\Gamma^{\mathsf{perm}}_{\omega}$ which records the type of its permanent local resources with bindings $v \sim A \mathbin{@} \omega$, where v may be thought of as a reference to a permanent local resource. We require that A not be a primitive type (to ensure the progress property in Theorem 4.5). We write Γ^{perm} for the union of all known local typing contexts $\Gamma^{\mathsf{perm}}_{\omega}$.

4.2 Configuration

We represent the state of a network with a *configuration* C which records the term being evaluated at each logical node. A *configuration type* Λ records the type of the term and the mobility of the resultant value. We assume that no logical node appears more than once in C and consider C as an unordered set.

$$
\begin{array}{lcll}
\text{configuration} & C & ::= & \cdot \mid C, M \ \mathbf{at}\ \gamma \\
\text{configuration type} & \Lambda & ::= & \cdot \mid \Lambda, \gamma \sim A \mathbin{@} \omega \mid \Lambda, \gamma \sim A \mathbin{@} \star
\end{array}
$$

- $M\ \mathbf{at}\ \gamma$ in C means that logical node γ is currently evaluating term M.
- $\gamma \sim A \mathbin{@} \omega$ in Λ means that the term at logical node γ evaluates to a value of type A valid at physical node ω.
- $\gamma \sim A \mathbin{@} \star$ in Λ means that the term at logical node γ evaluates to a globally valid value of type A.

The extended type system is formulated with a *configuration typing judgment* $C :: \Lambda$, which means that configuration C has configuration type Λ. The network operational semantics is formulated with a *configuration transition judgment* $C \Longrightarrow C'$, which means that configuration C reduces or evolves to configuration C'. We first consider the extended type system and then the network operational semantics.

4.3 Extended type system

In order to be able to typecheck those terms whose type is determined by remote nodes, we introduce an *extended typing judgment* which includes a configuration type as part of its typing context:

- An extended typing judgment $\Lambda; \Delta; \Gamma \vdash_{\omega} M \sim A \mathbin{@} \omega'$ means that under configuration type Λ, mobile typing context Δ, and local typing context Γ, term M at any logical node on physical node ω evaluates to a value of type A valid at physical node ω'. We assume $\Gamma^{\mathsf{perm}} \subset \Gamma$, which means that all references to permanent local resources are public.
- $\Lambda; \Delta; \Gamma \vdash_{\omega} M : A$ is a shorthand for $\Lambda; \Delta; \Gamma \vdash_{\omega} M \sim A \mathbin{@} \omega$.

The rules for extended typing judgments are derived from (and given the same name as) those in Figure 2 by prepending a configuration type Λ to every judgment $\Delta; \Gamma \vdash_{\omega} M \sim A \mathbin{@} \omega'$.

The configuration typing judgment is defined in terms of extended typing judgments. It has only one inference rule, which may be regarded as its definition:

$$\frac{\begin{array}{l} \textit{for each } M \textbf{ at } \gamma \in C, \\ \quad \gamma \sim A \mathbin{@} \omega \in \Lambda \textit{ and } \Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} M \sim A \mathbin{@} \omega, \textit{or} \\ \quad \gamma \sim A \mathbin{@} \star \in \Lambda \textit{ and } \Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} M \sim A \mathbin{@} \omega \textit{ for a fresh node } \omega. \end{array}}{C :: \Lambda} \;\mathsf{Tcfg}$$

We assume $|C| = |\Lambda|$ to maintain a one-to-one correspondence between C and Λ; hence Λ contains exactly one element for each logical node in C.

4.4 Network operational semantics

The configuration transition judgment uses evaluation contexts in a call-by-name style; we could equally choose a call-by-value style with another case $(\lambda x\!:\!A.\, M)\ \kappa$ for evaluation contexts:

$$\begin{array}{lcl} \text{evaluation context} \quad \kappa & ::= & [] \mid \kappa\ M \mid \mathsf{letbox}\ x = \kappa\ \mathsf{in}\ M \mid \\ & & \mathsf{letcir}\ v = \kappa\ \mathsf{in}\ M \mid \mathsf{letcir}\ v = \mathsf{cir}\ \kappa\ \mathsf{in}\ M \mid \mathsf{letcir}\ v = \mathsf{cir}_\omega\ \kappa\ \mathsf{in}\ M \end{array}$$

An evaluation context κ is a term with a hole $[]$ in it, where the hole indicates the position where a reduction may occur. The following rule shows how to use the β-reduction rules of $\lambda_{\Box\bigcirc^W}$ in the network operational semantics; $\longrightarrow$ refers to the one of the β-reduction rules $\rightarrow_{\beta\supset}$, $\rightarrow_{\beta\Box}$, $\rightarrow_{\beta\Box'}$, $\rightarrow_{\beta\bigcirc}$, $\rightarrow_{\beta\bigcirc'}$ of $\lambda_{\Box\bigcirc^W}$:

$$\frac{M \longrightarrow N}{C, \kappa[M] \textbf{ at } \gamma \Longrightarrow C, \kappa[N] \textbf{ at } \gamma} \;\mathsf{Rcfg}$$

Note that a configuration transition is nondeterministic, since the rule Rcfg can choose an arbitrary logical node γ from a given configuration.

We also need another configuration transition rule to deal with value variables in Γ^{perm}. Suppose that a value variable v is a reference to a permanent local resource V of a physical node ω (hence $v \sim A \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$). For example, V could be a printing function at a printer server ω. At a logical node γ such that $\mathcal{P}(\gamma) \neq \omega$, v does not need to reduce to V because V is not valid at γ anyway. If $\mathcal{P}(\gamma) = \omega$, however, v reduces to V by accessing the local resource. Thus, for each binding $v \sim A \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$, we define a reduction

$$v \rightarrow_{\mathsf{perm}} V$$

such that V is not another value variable and $\cdot; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega V : A$ holds. The following rule specifies that a reference to a permanent local resource reduces to a value only at the node to which it belongs:

$$\frac{v \sim A \mathbin{@} \omega \in \Gamma^{\mathsf{perm}} \quad v \rightarrow_{\mathsf{perm}} V \quad \mathcal{P}(\gamma) = \omega}{C, \kappa[v] \textbf{ at } \gamma \Longrightarrow C, \kappa[V] \textbf{ at } \gamma} \;\mathsf{Rvalvar}$$

Thus the rule Rvalvar ensures that references to permanent local resources are never dereferenced at remote nodes.

4.5 Communication constructs

The network operational semantics becomes interesting only with communication constructs; without communication constructs, all logical nodes perform stand-alone computations independently of each other and the type safety holds trivially. Below we give three examples of communication constructs. Each construct is defined with extended typing rules and configuration transition rules.

$$
\begin{array}{llcl}
\text{type} & A & ::= & \cdots \mid \mathsf{unit} \\
\text{term} & M & ::= & \cdots \mid () \mid \mathsf{eval}\ M \\
\text{value} & V & ::= & \cdots \mid () \\
\text{evaluation context} & \kappa & ::= & \cdots \mid \mathsf{eval}\ \kappa
\end{array}
$$

$$
\frac{}{\Lambda; \Delta; \Gamma \vdash_\omega () : \mathsf{unit}}\ \mathsf{T()} \quad \frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{eval}\ M : \mathsf{unit}}\ \mathsf{Teval} \quad \frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box_{\omega'} A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{eval}\ M : \mathsf{unit}}\ \mathsf{Teval@}
$$

$$
\frac{\mathit{new}\ \gamma'}{C, \kappa[\mathsf{eval\ box}\ M]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[()]\ \mathbf{at}\ \gamma, M\ \mathbf{at}\ \gamma'}\ \mathsf{Reval}
$$

$$
\frac{\mathit{new}\ \gamma'\ @\ \omega'}{C, \kappa[\mathsf{eval\ box}_{\omega'}\ M]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[()]\ \mathbf{at}\ \gamma, M\ \mathbf{at}\ \gamma'}\ \mathsf{Reval@}
$$

Figure 3: Definition of the remote evaluation construct.

4.5.1 Remote evaluation

In order to be able to evaluate a mobile term at a remote node, we introduce a remote evaluation construct $\mathsf{eval}\ M$. It expects M to evaluate to $\mathsf{box}\ N$ or $\mathsf{box}_\omega\ N$ and transmits N to a remote node. Unlike a remote procedure call, it does not expect the result of evaluating N and immediately returns a value $()$ of type unit.

Figure 3 shows the definition of the remote evaluation construct. The rule Reval creates a new logical node γ' with $\mathit{new}\ \gamma'$ because M may be evaluated at any node. In contrast, the rule Reval@ creates a new logical node γ' with $\mathit{new}\ \gamma'\ @\ \omega'$ because M may be evaluated only at node ω'.[1]

4.5.2 Futures

A future construct [8] is similar to a remote procedure call in that it initiates a stand-alone computation at a remote node and also expects the result. The difference is that it does not wait for the result and immediately returns a *synchronization variable* which points to the remote node. When the result is needed, it is requested through a synchronization operation. If the remote node has finished the computation, the result is returned; otherwise the synchronization operation is suspended until the result becomes ready. We can simulate a remote procedure call by performing a synchronization operation immediately after evaluating a future construct.

Figure 4 shows the definition of the future construct $\mathsf{future}\ M$. It expects M to be of type $\Box\bigcirc A$, $\Box_\omega\bigcirc A$, $\Box\bigcirc_{\omega'} A$, or $\Box_\omega\bigcirc_{\omega'} A$. If M evaluates to $\mathsf{box}\ N$, it initiates a stand-alone computation of $\mathsf{letcir}\ v = N\ \mathsf{in}\ v$ at a new logical node γ created with $\mathit{new}\ \gamma$ and returns a synchronization variable $\mathsf{syncvar}\ \gamma$ of type $A\ \mathsf{sync}$; if M evaluates to $\mathsf{box}_\omega\ N$, it initiates the same stand-alone computation at a new logical node γ created with $\mathit{new}\ \gamma\ @\ \omega$ and returns a synchronization variable $\mathsf{syncvar}\ \gamma$ of type $A\ \mathsf{sync}_\omega$. Since N has type $\bigcirc A$ or $\bigcirc_{\omega'} A$, $\mathsf{letcir}\ v = N\ \mathsf{in}\ v$ evaluates to a mobile value of type A that is valid either at any node or at node ω'. The result is requested through a synchronization operation $\mathsf{syncwith\ syncvar}\ \gamma$.

Note that a synchronization variable itself is inherently mobile and we can synchronize with it *at any node*. Intuitively it is just a pointer to a certain logical node and hence is globally valid. The result of a synchronization operation may not be valid at the node where it takes place, but the typing system correctly

[1] A remote evaluation construct can be simulated by a future construct; we present the remote evaluation construct only as a simple example of using modal types $\Box A$ and $\Box_\omega A$. As we will see below, $\mathsf{eval}\ M$ is simulated as $\mathsf{let}\ \_ = \mathsf{future}\ (\mathsf{letbox}\ x = M\ \mathsf{in\ box\ let}\ \_ = x\ \mathsf{in\ cir}\ ())\ \mathsf{in}\ ()$ where $\mathsf{let}\ x = M\ \mathsf{in}\ N$ is standard let-binding and $\_$ is a wildcard pattern.

indicates the mobility of the result. For example, in the rule Tswith', the result of evaluating $\mathsf{syncwith}\ M$ is valid only at node ω', which is correctly indicated by $@\ \omega'$ in the typing judgment of the conclusion.

The rules Tsvar and Tsvar' show that a configuration type Λ is necessary in extended typing judgments in order to typecheck synchronization variables. Since synchronization variables are created only by the future construct and do not appear in a source program, we need these rules only for proving the type safety.

$$\begin{array}{llll}
\text{type} & A & ::= & \cdots \mid A\ \mathsf{sync} \mid A\ \mathsf{sync}_\omega \\
\text{term} & M & ::= & \cdots \mid \mathsf{future}\ M \mid \mathsf{syncvar}\ \gamma \mid \mathsf{syncwith}\ M \\
\text{value} & V & ::= & \cdots \mid \mathsf{syncvar}\ \gamma \\
\text{evaluation context} & \kappa & ::= & \cdots \mid \mathsf{future}\ \kappa \mid \mathsf{syncwith}\ \kappa
\end{array}$$

$$\frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box\bigcirc A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{future}\ M \sim A\ \mathsf{sync}\ @\ \omega^*}\ \mathsf{Tfuture} \qquad \frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box_{\omega'}\bigcirc A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{future}\ M \sim A\ \mathsf{sync}\ @\ \omega^*}\ \mathsf{Tfuture@}$$

$$\frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box\bigcirc_{\omega''} A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{future}\ M \sim A\ \mathsf{sync}_{\omega''}\ @\ \omega^*}\ \mathsf{Tfuture}' \qquad \frac{\Lambda; \Delta; \Gamma \vdash_\omega M : \Box_{\omega'}\bigcirc_{\omega''} A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{future}\ M \sim A\ \mathsf{sync}_{\omega''}\ @\ \omega^*}\ \mathsf{Tfuture@}'$$

$$\frac{\gamma \sim A\ @\ \star \in \Lambda}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{syncvar}\ \gamma : A\ \mathsf{sync}}\ \mathsf{Tsvar} \qquad \frac{\gamma \sim A\ @\ \omega' \in \Lambda}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{syncvar}\ \gamma : A\ \mathsf{sync}_{\omega'}}\ \mathsf{Tsvar}'$$

$$\frac{\Lambda; \Delta; \Gamma \vdash_\omega M : A\ \mathsf{sync}}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{syncwith}\ M \sim A\ @\ \omega^*}\ \mathsf{Tswith} \qquad \frac{\Lambda; \Delta; \Gamma \vdash_\omega M : A\ \mathsf{sync}_{\omega'}}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{syncwith}\ M \sim A\ @\ \omega'}\ \mathsf{Tswith}'$$

$$\frac{\mathit{new}\ \gamma'}{C, \kappa[\mathsf{future}\ \mathsf{box}\ M]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[\mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v\ \mathbf{at}\ \gamma'}\ \mathsf{Rfuture}$$

$$\frac{\mathit{new}\ \gamma'\ @\ \omega'}{C, \kappa[\mathsf{future}\ \mathsf{box}_{\omega'}\ M]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[\mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v\ \mathbf{at}\ \gamma'}\ \mathsf{Rfuture@}$$

$$\frac{}{C, \kappa[\mathsf{syncwith}\ \mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' \Longrightarrow C, \kappa[V]\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma'}\ \mathsf{Rswith}$$

Figure 4: Definition of the future construct. ω^* may be read as "any node."

4.5.3 Asynchronous channels

An asynchronous channel is a first-in-first-out buffer containing values communicated among nodes. A write operation adds a value to the buffer and always succeeds. A read operation removes the oldest value from the buffer; if the buffer is empty, it waits until a new value is written. We assume that an asynchronous channel is accessible to every node. This means that a value written to it must be globally valid, which in turn means that a value read from it is also globally valid. A similar idea can be used to implement *shared variables*, for which a write operation overwrites a single-entry buffer and a read operation leaves the buffer intact.

We implement an asynchronous channel for type A as a special node holding a list of values of type A. The node updates the list when a read or write operation is performed on the channel. It maintains the invariant that every value in the list is globally valid.

Figure 5 shows the definition of asynchronous channels. nil and $V_h :: V_t$, both of type A vlist, are constructs for lists. $\mathsf{newchan}_A$ creates a new logical node γ to implement an asynchronous channel for type A, and returns a *channel variable* $\mathsf{chanvar}\ \gamma$ of type A chan. A channel variable points to an asynchronous channel and is globally valid. The rules Rreadc and $\mathsf{Rwritec}$ show how read and write operations manipulate the node associated with an asynchronous channel.

Like synchronization variables for future constructs, channel variables are created only by $\mathsf{newchan}_A$ and do not appear in a source program. Therefore we need the rule Tchanv only for proving the type safety.

$$
\begin{array}{llcl}
\text{type} & A & ::= & \cdots \mid A\ \mathsf{chan} \mid A\ \mathsf{vlist} \\
\text{term} & M & ::= & \cdots \mid \mathsf{nil} \mid V :: V \mid \mathsf{chanvar}\ \gamma \mid \mathsf{newchan}_A \mid \mathsf{readchan}\ M \mid \mathsf{writechan}\ M\ M \\
\text{value} & V & ::= & \cdots \mid \mathsf{nil} \mid V :: V \mid \mathsf{chanvar}\ \gamma \\
\text{evaluation context} & \kappa & ::= & \cdots \mid \mathsf{readchan}\ \kappa \mid \mathsf{writechan}\ \kappa\ M \mid \mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ \kappa
\end{array}
$$

$$
\frac{}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{nil} : A\ \mathsf{vlist}}\ \mathsf{Tvnil}
\qquad
\frac{\Lambda; \Delta; \Gamma \vdash_\omega V_h : A \quad \Lambda; \Delta; \Gamma \vdash_\omega V_t : A\ \mathsf{vlist}}{\Lambda; \Delta; \Gamma \vdash_\omega V_h :: V_t : A\ \mathsf{vlist}}\ \mathsf{Tvcon}
$$

$$
\frac{\gamma \sim A\ \mathsf{vlist} \ @ \star \in \Lambda}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{chanvar}\ \gamma : A\ \mathsf{chan}}\ \mathsf{Tchanv}
\qquad
\frac{}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{newchan}_A \sim A\ \mathsf{chan} \ @\ \omega^*}\ \mathsf{Tnewc}
$$

$$
\frac{\Lambda; \Delta; \Gamma \vdash_\omega M : A\ \mathsf{chan}}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{readchan}\ M \sim A \ @\ \omega^*}\ \mathsf{Treadc}
$$

$$
\frac{\Lambda; \Delta; \Gamma \vdash_\omega M : A\ \mathsf{chan} \qquad \textit{fresh}\ \omega' \quad \Lambda; \Delta; \Gamma \vdash_\omega N \sim A \ @\ \omega'}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{writechan}\ M\ N \sim A \ @\ \omega^*}\ \mathsf{Twritec}
$$

$$
\frac{\textit{new}\ \gamma'}{C, \kappa[\mathsf{newchan}_A]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[\mathsf{chanvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{nil}\ \mathbf{at}\ \gamma'}\ \mathsf{Rnewc}
$$

$$
\frac{}{C, \kappa[\mathsf{readchan}\ \mathsf{chanvar}\ \gamma']\ \mathbf{at}\ \gamma, V_h :: V_t\ \mathbf{at}\ \gamma' \Longrightarrow C, \kappa[V_h]\ \mathbf{at}\ \gamma, V_t\ \mathbf{at}\ \gamma'}\ \mathsf{Rreadc}
$$

$$
\frac{}{\begin{array}{l} C, \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' \Longrightarrow \\ \qquad\qquad C, \kappa[V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil}\ \mathbf{at}\ \gamma' \end{array}}\ \mathsf{Rwritec}
$$

Figure 5: Definition of asynchronous channels. ω^* may be read as "any node."

4.6 Type safety

The type safety of the network operational semantics consists of two properties: configuration type preservation (Theorem 4.1) and configuration progress (Theorem 4.5). Configuration type preservation states that a configuration transition does not alter the type and mobility of the term being evaluated at each node. Configuration progress states that we can apply a configuration transition rule until every node has finished its stand-alone computation or waits for a result from another node (by the rules Rswith, Rreadc, and Rwritec).

Theorem 4.1 (configuration type preservation).
If $C :: \Lambda$ *and* $C \Longrightarrow C'$, *then* $C' :: \Lambda'$ *such that* $\Lambda \subset \Lambda'$.

Proof. By case analysis on $C \Longrightarrow C'$. There are three cases:
1) $C_0, \kappa[M]\ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[N]\ \mathbf{at}\ \gamma$
2) $C_0, \kappa[M]\ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[N]\ \mathbf{at}\ \gamma, N'\ \mathbf{at}\ \gamma'$
3) $C_0, \kappa[M]\ \mathbf{at}\ \gamma, M'\ \mathbf{at}\ \gamma' \Longrightarrow C_0, \kappa[N]\ \mathbf{at}\ \gamma, N'\ \mathbf{at}\ \gamma'$

In each case, we show that N preserves the type and mobility of M. In case 3), we also show that N' preserves the type and mobility of M'. □

Lemma 4.2 (Canonical forms). *If* $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega V \sim A \ @\ \omega'$, *then*
$V = v$,

A is a primitive type,
$A = A_1 \supset A_2$ and $V = \lambda x\!:\!A_1.\, M$,
$A = \Box B$ and $V = \mathsf{box}\ M$,
$A = \Box_{\omega''} B$ and $V = \mathsf{box}_{\omega''}\ M$,
$A = \bigcirc B$ and $V = \mathsf{cir}\ M$,
$A = \bigcirc_{\omega''} B$ and $V = \mathsf{cir}_{\omega''}\ M$,
$A = \mathsf{unit}$ and $V = ()$,
$A = B\ \mathsf{sync}$ and $V = \mathsf{syncvar}\ \gamma$,
$A = B\ \mathsf{sync}_{\omega''}$ and $V = \mathsf{syncvar}\ \gamma$,
$A = B\ \mathsf{chan}$ and $V = \mathsf{chanvar}\ \gamma$,
$A = B\ \mathsf{vlist}$ and $V = \mathsf{nil}$,
or $A = B\ \mathsf{vlist}$ and $V = V_h :: V_t$.

Proof. Suppose that $V \neq v$ and A is not a primitive type.

If $A = A_1 \supset A_2$, then $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega V \sim A @ \omega'$ is derived by the rule $\supset\mathsf{I}_W$, optionally followed by the rule Val_W. Hence $V = \lambda x\!:\!A_1.\, M$.

All the other cases are analogous. □

Lemma 4.3. *If $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A @ \omega'$, then*

$M = V \neq v$,
$M = v$ and $v \sim A @ \omega' \in \Gamma^{\mathsf{perm}}$,
$M = \kappa[v]$ and $v \sim B @ \omega \in \Gamma^{\mathsf{perm}}$,
$M = \kappa[N]$ where $N \longrightarrow N'$,
$M = \kappa[\mathsf{eval\ box}\ N]$,
$M = \kappa[\mathsf{eval\ box}_{\omega''}\ N]$,
$M = \kappa[\mathsf{future\ box}\ N]$,
$M = \kappa[\mathsf{future\ box}_{\omega''}\ N]$,
$M = \kappa[\mathsf{syncwith\ syncvar}\ \gamma]$,
$M = \kappa[\mathsf{newchan}_B]$,
$M = \kappa[\mathsf{readchan\ chanvar}\ \gamma]$,
or $M = \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V]$.

Proof. By induction on the structure of $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A @ \omega'$. We present one case.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M : A_{prim}}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A_{prim} @ \omega'}\ \mathsf{Prim}{\sim}_W\ (\omega \neq \omega')$:

If $M = V \neq v$ by induction hypothesis, we are done.

$M = v$ and $v \sim A_{prim} @ \omega \in \Gamma^{\mathsf{perm}}$ cannot happen by induction hypothesis, since the assumption on Γ^{perm} requires that permanent local resources not be of a primitive type.

If $M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim B @ \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_B$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V$,

then we are done. □

Lemma 4.4. *If $\Lambda; \Delta; \Gamma \vdash_\omega \kappa[M] \sim A @ \omega'$, then there exist B and ω'' such that $\Lambda; \Delta; \Gamma \vdash_\omega M \sim B @ \omega''$.*

Proof. By induction on the structure of κ. □

Theorem 4.5 (configuration progress).
If $C :: \Lambda$, then either there exists C' such that $C \Longrightarrow C'$, or C consists only of the following:
V **at** γ,
$\kappa[\mathsf{syncwith\ syncvar}\ \gamma']$ **at** γ,
$\kappa[\mathsf{readchan\ chanvar}\ \gamma']$ **at** γ,
$\kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]$ **at** γ.

Proof. Suppose $C = C_0, M \textbf{ at } \gamma$. By the rule Tcfg, we have $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A \,@\, \omega'$ for $\mathcal{P}(\gamma) = \omega$ and a certain node ω'. We do case analysis according to Lemma 4.3. We present one case.
Case $M = \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]$:

By Lemma 4.4, we have $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V \sim B \,@\, \omega''$.

By the rule $\mathsf{Twritec}$ (optionally preceded by the rule $\mathsf{Prim}\sim_W$ if B is a primitive type), we have $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{chanvar}\ \gamma' : B\ \mathsf{chan}$.

By the rule Tchanv, we have $\gamma' \sim B\ \mathsf{vlist} \,@\, \star \in \Lambda$.

Since $C :: \Lambda$, we have $C = C_0', M \textbf{ at } \gamma, N \textbf{ at } \gamma'$ and $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N \sim B\ \mathsf{vlist} \,@\, \omega^*$ for a fresh node ω^*.

If $N = V_1 :: \cdots :: V_n :: \mathsf{nil}$ (where $0 \leq n$), then

$$\frac{}{\begin{array}{l} C_0', \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V] \textbf{ at } \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil} \textbf{ at } \gamma' \Longrightarrow \\ \qquad\qquad\qquad\qquad C_0', \kappa[V] \textbf{ at } \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil} \textbf{ at } \gamma' \end{array}} \mathsf{Rwritec} \;.$$

Otherwise $N \neq V_1 :: \cdots :: V_n :: \mathsf{nil}$ and M is not further reduced. $\square$

The two cases $\kappa[\mathsf{syncwith\ syncvar}\ \gamma'] \textbf{ at } \gamma$ and $\kappa[\mathsf{readchan\ chanvar}\ \gamma'] \textbf{ at } \gamma$ in Theorem 4.5 can occur during a distributed computation. Here is an example of a configuration whose transition gives rise to the two cases:

$$\begin{array}{ll} & \mathsf{syncwith\ future\ box\ cir}\ (\mathsf{readchan\ newchan}_A) \textbf{ at } \gamma \\ \Longrightarrow & \mathsf{syncwith\ syncvar}\ \gamma' \textbf{ at } \gamma, \mathsf{letcir}\ v = \mathsf{cir}\ (\mathsf{readchan\ newchan}_A)\ \mathsf{in}\ v \textbf{ at } \gamma' \\ \Longrightarrow & \mathsf{syncwith\ syncvar}\ \gamma' \textbf{ at } \gamma, \mathsf{letcir}\ v = \mathsf{cir}\ (\mathsf{readchan\ chanvar}\ \gamma'')\ \mathsf{in}\ v \textbf{ at } \gamma', \mathsf{nil} \textbf{ at } \gamma'' \end{array}$$

Here node γ waits for a result from node γ', which in turns waits for a value to be written to node γ''. Since no value can be written to node γ'', the last configuration is stuck. The case $\kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V] \textbf{ at } \gamma$ in Theorem 4.5 occurs only when the term being evaluated at node γ' cannot be reduced to a list of values (whether empty or not), as clarified in the proof above. This case, however, does not actually occur because an asynchronous channel is always initialized as nil by the rule Rnewc and never holds a term that is not a list.

The type safety of the network operational semantics implies that mobile terms and mobile values are both safe to use: well-typed terms never go wrong even in the presence of mobile terms and mobile values.

4.7 Example

Consider a network of two nodes $\mathbf{S}$ (server) and $\mathbf{C}$ (client). Node $\mathbf{S}$ has a printer attached to it, and provides a function *print* for printing pdf files of type pdf. The printer accepts pdf files written only with local fonts, and provides a function $convert_{\mathbf{S}}$ for converting ordinary pdf files into a suitable format. Node $\mathbf{C}$ has its own conversion function $convert_{\mathbf{C}}$.

$$\begin{array}{lcl} \Gamma^{\mathsf{perm}}_{\mathbf{S}} & = & \mathit{file}_{\mathbf{S}} \sim \mathsf{pdf} \,@\, \mathbf{S}, \mathit{convert}_{\mathbf{S}} \sim \bigcirc(\mathsf{pdf} \supset \bigcirc_{\mathbf{S}}\mathsf{pdf}) \,@\, \mathbf{S}, \mathit{print} \sim \mathsf{pdf} \supset \mathsf{unit} \,@\, \mathbf{S} \\ \Gamma^{\mathsf{perm}}_{\mathbf{C}} & = & \mathit{file}_{\mathbf{C}} \sim \mathsf{pdf} \,@\, \mathbf{C}, \mathit{convert}_{\mathbf{C}} \sim \mathsf{pdf} \supset \bigcirc_{\mathbf{S}}\mathsf{pdf} \,@\, \mathbf{C} \end{array}$$

We give three examples (similar to those in [9]) to illustrate how to describe tasks in $\lambda_{\Box\bigcirc}{}^{W}$. All terms below have type $\Box_{\mathbf{S}}\mathsf{unit}$ and typecheck at any node. We use syntactic sugar $\mathsf{rpc}\ M$ for $\mathsf{syncwith\ future}\ M$.

Printing a pdf file $\mathit{file}_{\mathbf{S}}$ of node $\mathbf{S}$:

$$\mathsf{box}_{\mathbf{S}}\ (\mathit{print}\ \mathit{file}_{\mathbf{S}})$$

Printing a pdf file $\mathit{file}_{\mathbf{C}}$ of node $\mathbf{C}$ after converting it with $\mathit{convert}_{\mathbf{C}}$:

$$\begin{array}{l} \mathsf{letcir}\ v = \mathsf{cir}_{\mathbf{S}}\ \mathsf{rpc}\ \mathsf{box}_{\mathbf{C}}\ (\mathit{convert}_{\mathbf{C}}\ \mathit{file}_{\mathbf{C}})\ \mathsf{in} \\ \mathsf{box}_{\mathbf{S}}\ (\mathit{print}\ v) \end{array}$$

Printing a pdf file $\mathit{file}_{\mathbf{C}}$ of node $\mathbf{C}$ after converting it with $\mathit{convert}_{\mathbf{S}}$:

$$\begin{array}{ll}\mathsf{box}_{\mathbf{S}} & \mathsf{letcir}\ v = \mathit{convert}_{\mathbf{S}}\ \mathsf{in} \\ & \mathsf{letcir}\ v' = \mathsf{cir}_{\mathbf{S}}\ \mathsf{rpc}\ \mathsf{box}_{\mathbf{C}}\ (v\ \mathit{file}_{\mathbf{C}})\ \mathsf{in} \\ & \mathit{print}\ v'\end{array}$$

5 Related work

5.1 Local resources in distributed computations

In designing a distributed system, there are several ways to handle references to local resources when they are transmitted (as part of a mobile term) to a remote node. If the underlying system supports direct access to remote resources, such a reference can be replaced in the remote node by a proxy which automatically redirects all requests for the resource to the originating node. Obliq [3] adopts such a computation model, in which *local references* are replaced by *network references* in a remote node.

$\lambda_{\Box\bigcirc}{}^{W}$ allows references to remote resources in mobile terms, but it also ensures that they are never dereferenced. In essence, references to local resources become invalid when they are transmitted to remote nodes, but their validity is restored when they are brought back to the original node. For example, if a term M accesses local resources of node ω and returns a globally valid value of type A, then

$$\mathsf{syncwith}\ \mathsf{future}\ \mathsf{box}_{\omega}\ \mathsf{cir}\ M$$

can be evaluated *at any node*: wherever the above term is evaluated, it calls back with the same term M to node ω, where all references in M again point to their corresponding local resources. The same computation model is used by Mascolo *et al.* [11] in their treatment of references.

References to remote resources, as used in the above two computation models, are suitable for persistent resources such as printers and databases, but they can be problematic for ephemeral resources which are eventually destroyed. For example, the presence of references to remote heap cells incurs the problem of distributed garbage collection [7]. An alternative computation model is one that permits no references to remote resources either by rejecting mobile terms containing such references or by transmitting copies of local resources along with mobile terms. Facile [10] supports such a computation model, in which local resources are copied whenever their references (called *singular values*) are transmitted to a remote node. Thus the problem with ephemeral resources is resolved at an increased cost of transmitting mobile terms.

5.2 Modal languages for distributed computation

Borghuis and Feijs [1] present a typed λ-calculus *MTSN* (Modal Type System for Networks). It assumes stationary services (*i.e.*, stationary code) and mobile data, and belongs to the client/server paradigm. An indexed modal type $\Box^{\omega}(A \to B)$ represents services transforming data of type A into data of type B at node ω (similarly to $\Box_{\omega}(A \supset B)$ in $\lambda_{\Box\bigcirc}{}^{W}$). MTSN is a task description language rather than a programming language, since services are all "black boxes" whose inner workings are unknown. For example, terms of type $\mathit{tex} \to \mathit{dvi}$ all describe procedures to convert tex files to dvi files. Thus reduction on terms is tantamount to simplifying procedures to achieve a certain task.

Jia and Walker [9] present a modal language λ_{rpc} which belongs to the remote evaluation paradigm. It is based upon hybrid logic [2], and every typing judgment explicitly specifies the current node where typechecking takes place. The modalities $\Box$ and $\Diamond$ are used for mobile terms that can be evaluated at any node and at a certain node, respectively.

Murphy *et al.* [13] present a modal language *Lambda 5* which addresses both code mobility and resource locality. It also belongs to the remote evaluation paradigm, and is based upon modal logic S5 where all judgments are relativized to nodes. A value of type $\Box A$ contains a mobile term that can be evaluated at any node, and a value of type $\Diamond A$ contains a *label*, a reference to a local resource. A label may appear at remote nodes, but the type system guarantees that it is dereferenced only at the node where it is valid.

Although the intuition behind the modality $\Box$ is the same, λ_{rpc} and Lambda 5 are fundamentally different from $\lambda_{\Box\bigcirc}{}^{W}$ in their use of modal types $\Box A$ in remote procedure calls. In both languages, a remote procedure call, by the pull construct in λ_{rpc} and by the fetch construct in Lambda 5, is given a specific node where the evaluation is to occur, and therefore *does not expect a term contained in a value of type* $\Box A$. Instead it expects just a term of type $\Box A$, which itself may not be mobile but eventually produces a mobile term valid at any node including the caller node. The resultant mobile term is delivered to (*i.e.*, pulled or fetched by) the caller node, which needs to further evaluate it to obtain a value. As such, neither language needs to address the issue of value mobility. In contrast, a remote procedure call in $\lambda_{\Box\bigcirc}{}^{W}$ (by the eval or future construct) transmits a term *contained in a value of type* $\Box A$ and relies on the modality $\bigcirc$ for return values. Such use of the modality $\Box$ is natural in $\lambda_{\Box\bigcirc}{}^{W}$, since it supports remote procedure calls to unknown nodes.

Moody [12] presents a system which is based upon modal logic S4 and belongs to the remote evaluation paradigm. The modality $\Box$ is used for mobile terms that can be evaluated at any node, and the modality $\Diamond$ is used for terms located at some node. As in $\lambda_{\Box\bigcirc}{}^{W}$, remote procedure calls use modal types $\Box A$ to transmit mobile terms to unknown remote nodes. Moody's system uses the elimination rules for the modalities $\Box$ and $\Diamond$ to send mobile terms to remote nodes, and does not provide a separate construct for remote procedure calls.

6 Conclusion and future work

We present a modal language $\lambda_{\Box\bigcirc}{}^{W}$ which ensures the safety of both mobile terms and mobile values. It provides a flexible programming environment for various kinds of distributed computations. For example, if the network evolves dynamically and no permanent local resources are known in advance, only modal types $\Box A$ and $\bigcirc A$ are necessary; if the network is static and every node publishes its permanent local resources, we can program exclusively with indexed modal types $\Box_\omega A$ and $\bigcirc_\omega A$.

The modality $\bigcirc$ is useful in $\lambda_{\Box\bigcirc}{}^{W}$ only because the unit of communication includes a value. That is, if the unit of communication was just a term and did not include a value, the modality $\bigcirc$ would be unnecessary. Then, however, the future construct would have to be redefined in a similar way to the pull construct of λ_{rpc} and the fetch construct of Lambda 5, and asynchronous channels would be difficult to implement.

The three communication constructs of $\lambda_{\Box\bigcirc}{}^{W}$ are all defined separately. A better approach would be to introduce a few primitive operations and then implement various communication constructs using these primitive operations. For example, we could introduce a send operation for the modality $\Box$ and a receive operation for the modality $\bigcirc$, and then implement the future construct using these operations. Because of technical difficulties arising from asynchronous channels, however, we do not adopt this approach and define all communication constructs separately.

A drawback of $\lambda_{\Box\bigcirc}{}^{W}$ is that in general, references to ephemeral local resources cannot be transmitted to remote nodes. As an example, consider a pointer v of type ptr A at a logical node γ created with $new\ \gamma$. Node γ wishes to use v as a shared pointer among all its child nodes, *i.e.*, those nodes created with the eval and future constructs. No child node, however, even knows the existence of v because the physical node ω in a binding $v \sim A @ \omega$ is not known statically. (If node γ was created with $new\ \gamma @ \omega$, then v could be transmitted to remote nodes.)

To overcome this drawback, we are currently investigating how to augment $\lambda_{\Box\bigcirc}$ (not $\lambda_{\Box\bigcirc}{}^{W}$) with a modality $\Diamond$ similar to that of Jia and Walker [9]. The idea is that a term M in dia M of type $\Diamond A$ can be evaluated at a certain node, which is unknown to the type system but known to the runtime system. The use of the modality $\Diamond$ will allow us to dispense with indexed modalies $\Box_\omega$ and $\bigcirc_\omega$.

Acknowledgment

I am grateful to Tom Murphy and Jonathan Moody for their helpful comments on an earlier draft of this paper, and Karl Crary for his helpful comments on the type system.

References

[1] T. Borghuis and L. Feijs. A constructive logic for services and information flow in computer networks. *The Computer Journal*, 43(4):275–289, 2000.

[2] T. Braüner. Natural deduction for hybrid logic. *Journal of Logic and Computation*, 14(3):329–353, 2004.

[3] L. Cardelli. A language with distributed scope. In *Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 286–297. ACM Press, 1995.

[4] A. Carzaniga, G. P. Picco, and G. Vigna. Designing distributed applications with mobile code paradigms. In *Proceedings of the 19th international conference on Software engineering*, pages 22–32. ACM Press, 1997.

[5] G. Cugola, C. Ghezzi, G. P. Picco, and G. Vigna. Analyzing mobile code languages. In *Selected Presentations and Invited Papers Second International Workshop on Mobile Object Systems - Towards the Programmable Internet*, pages 93–110. Springer-Verlag, 1997.

[6] R. Davies and F. Pfenning. A modal analysis of staged computation. *Journal of the ACM*, 48(3):555–604, 2001.

[7] F. L. Fessant, I. Piumarta, and M. Shapiro. An implementation of complete, asynchronous, distributed garbage collection. In *Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation*, pages 152–161. ACM Press, 1998.

[8] R. H. Halstead, Jr. Multilisp: a language for concurrent symbolic computation. *ACM Transactions on Programming Languages and Systems*, 7(4):501–538, 1985.

[9] L. Jia and D. Walker. Modal proofs as distributed programs (extended abstract). In D. Schmidt, editor, *Proceedings of the European Symposium on Programming, LNCS 2986*, pages 219–233. Springer, Apr. 2004.

[10] F. C. Knabe. *Language Support for Mobile Agents*. PhD thesis, Department of Computer Science, Carnegie Mellon University, 1995.

[11] C. Mascolo, G. P. Picco, and G.-C. Roman. A fine-grained model for code mobility. In *Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 39–56. Springer-Verlag, 1999.

[12] J. Moody. Modal logic as a basis for distributed computation. Technical Report CMU-CS-03-194, Carnegie Mellon University, Oct. 2003.

[13] T. Murphy, VII, K. Crary, R. Harper, and F. Pfenning. A symmetric modal lambda calculus for distributed computing. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*. IEEE Press, July 2004.

[14] F. Pfenning and R. Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.

[15] A. K. Simpson. *The Proof Theory and Semantics of Intuitionistic Modal Logic*. PhD thesis, Department of Philosophy, University of Edinburgh, 1994.

A Proofs of the properties of $\lambda_{\Box\bigcirc}$

Proposition A.1.
If $\Delta;\Gamma \vdash M : A$ *and* $\Delta;\Gamma, x : A \vdash N : B$*, then* $\Delta;\Gamma \vdash [M/x]N : B$.
If $\Delta;\Gamma \vdash M : A$ *and* $\Delta;\Gamma, x : A \vdash N \sim B$*, then* $\Delta;\Gamma \vdash [M/x]N \sim B$.

Proof. By simulataneous induction on the structure of of the derivation of $\Delta;\Gamma, x : A \vdash N : B$ and $\Delta;\Gamma, x : A \vdash N \sim B$.
Proof of the first clause:
Case $N = x$: $[M/x]N = M$
By the rule Cvar, $\Delta;\Gamma, x : A \vdash N : B$ implies $A = B$.
$\Delta;\Gamma \vdash M : A$ implies $\Delta;\Gamma \vdash [M/x]N : A$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = y$, $y \neq x$: $[M/x]N = y$
By the rule Cvar, $\Delta;\Gamma, x : A \vdash N : B$ implies $y :: B \in \Delta$ or $y : B \in \Gamma, x : A$.
Since $y \neq x$, we have $y :: B \in \Delta$ or $y : B \in \Gamma$.
By the rule Cvar, $\Delta;\Gamma \vdash y : B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = v$: $[M/x]N = v$
By the rule Vvar, $\Delta;\Gamma, x : A \vdash N : B$ implies $v \sim B \in \Delta$.
By the rule Vvar, $\Delta;\Gamma \vdash v : B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = \lambda y\!:\!B'.\,N'$, $y \neq x$, y not a free variable of M: $[M/x]N = \lambda y\!:\!B'.\,[M/x]N'$
By the rule $\supset\!\mathsf{I}$, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\Gamma, x : A, y : B' \vdash N' : B''$ and $B = B' \supset B''$.
By weakening, $\Delta;\Gamma \vdash M : A$ implies $\Delta;\Gamma, y : B' \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma, y : B' \vdash [M/x]N' : B''$.
By the rule $\supset\!\mathsf{I}$, $\Delta;\Gamma \vdash \lambda y\!:\!B'.\,[M/x]N' : B' \supset B''$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = N_1\ N_2$: $[M/x]N = [M/x]N_1\ [M/x]N_2$
By the rule $\supset\!\mathsf{E}$, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\Gamma, x : A \vdash N_1 : B' \supset B$ and $\Delta;\Gamma, x : A \vdash N_2 : B'$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : B' \supset B$ and $\Delta;\Gamma \vdash [M/x]N_2 : B'$.
By the rule $\supset\!\mathsf{E}$, $\Delta;\Gamma \vdash [M/x]N_1\ [M/x]N_2 : B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = \mathsf{box}\ N'$: $[M/x]N = \mathsf{box}\ [M/x]N'$
By the rule $\Box\mathsf{I}$, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\cdot \vdash N' : B'$ and $B = \Box B'$.
Since x is not a free variable of N', we have $[M/x]N' = N'$.
By the rule $\Box\mathsf{I}$, $\Delta;\Gamma \vdash \mathsf{box}\ [M/x]N' : \Box B'$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:
$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\Box\mathsf{E}$, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\Gamma, x : A \vdash N_1 : \Box B_1$ and $\Delta, y :: B_1;\Gamma, x : A \vdash N_2 : B$.
By weakening, $\Delta;\Gamma \vdash M : A$ implies $\Delta, y :: B_1;\Gamma \vdash M : A$.

By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1;\Gamma \vdash [M/x]N_2 : B$.
By the rule $\Box$E, $\Delta;\Gamma \vdash \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 : B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = \mathsf{cir}\ N'$: $\quad [M/x]N = \mathsf{cir}\ [M/x]N'$
By the rule $\bigcirc$I, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\Gamma, x : A \vdash N' \sim B'$ and $B = \bigcirc B'$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N' \sim B'$.
By the rule $\bigcirc$I, $\Delta;\Gamma \vdash \mathsf{cir}\ [M/x]N' : \bigcirc B'$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $\quad [M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\bigcirc$E, $\Delta;\Gamma, x : A \vdash N : B$ implies $\Delta;\Gamma, x : A \vdash N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1;\Gamma, x : A \vdash N_2 : B$.
By weakening, $\Delta;\Gamma \vdash M : A$ implies $\Delta, v \sim B_1;\Gamma \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1;\Gamma \vdash [M/x]N_2 : B$.
By the rule $\bigcirc$E, $\Delta;\Gamma \vdash \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 : B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Proof of the second clause:
If the rule Prim$\sim$ is used to deduce $\Delta;\Gamma, x : A \vdash N \sim B$:
$\Delta;\Gamma, x : A \vdash N : B$ and B is a primitive type.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N : B$.
By the rule Prim$\sim$, $\Delta;\Gamma \vdash [M/x]N \sim B$.
Now N cannot be an application $N_1\ N_2$ or a variable y.
Case $N = V$:
By the rule Val, $\Delta;\Gamma, x : A \vdash N \sim B$ implies $\Delta;\cdot \vdash N : B$.
Since x is not a free variable of N, we have $[M/x]N = N$.
By the rule Val, $\Delta;\Gamma \vdash [M/x]N \sim B$.
Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:
$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\Box$E$'$, $\Delta;\Gamma, x : A \vdash N \sim B$ implies $\Delta;\Gamma, x : A \vdash N_1 : \Box B_1$ and $\Delta, y :: B_1;\Gamma, x : A \vdash N_2 \sim B$.
By weakening, $\Delta;\Gamma \vdash M : A$ implies $\Delta, y :: B_1;\Gamma \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1;\Gamma \vdash [M/x]N_2 \sim B$.
By the rule $\Box$E$'$, $\Delta;\Gamma \vdash \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B$.
Therefore $\Delta;\Gamma \vdash [M/x]N \sim B$.
Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $\quad [M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\bigcirc$E$'$, $\Delta;\Gamma, x : A \vdash N \sim B$ implies $\Delta;\Gamma, x : A \vdash N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1;\Gamma, x : A \vdash N_2 \sim B$.
By weakening, $\Delta;\Gamma \vdash M : A$ implies $\Delta, v \sim B_1;\Gamma \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1;\Gamma \vdash [M/x]N_2 \sim B$.
By the rule $\bigcirc$E$'$, $\Delta;\Gamma \vdash \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B$.
Therefore $\Delta;\Gamma \vdash [M/x]N \sim B$. $\Box$

Proof of Proposition 2.2 and Proposition2.4:

Proof. By simulataneous induction on the structure of the derivation of $\Delta, x :: A;\Gamma \vdash N : B$ and $\Delta, x :: A;\Gamma \vdash N \sim B$.
Proof of Proposition 2.2:
Case $N = x$: $\quad [M/x]N = M$
$\Delta;\cdot \vdash M : A$ implies $\Delta;\cdot \vdash [M/x]N : A$.
By weakening, $\Delta;\cdot \vdash [M/x]N : A$ implies $\Delta;\Gamma \vdash [M/x]N : A$.
$\Delta, x :: A;\Gamma \vdash N : B$ implies $A = B$.
Therefore $\Delta;\Gamma \vdash [M/x]N : B$.
Case $N = y$, $y \neq x$: $\quad [M/x]N = y$

By the rule Cvar, $\Delta, x :: A; \Gamma \vdash N : B$ implies $y :: B \in \Delta, x :: A$ or $y : B \in \Gamma$.
Since $y \neq x$, we have $y :: B \in \Delta$ or $y : B \in \Gamma$.
By the rule Cvar, $\Delta; \Gamma \vdash y : B$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = v$: $\quad [M/x]N = v$
By the rule Vvar, $\Delta, x :: A; \Gamma \vdash N : B$ implies $v \sim B \in \Delta, x :: A$, which means $v \sim B \in \Delta$.
By the rule Vvar, $\Delta; \Gamma \vdash v : B$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = \lambda y\!:\!B'.\, N'$, $y \neq x$, y not a free variable of M: $\quad [M/x]N = \lambda y\!:\!B'.\, [M/x]N'$
By the rule $\supset\mathsf{I}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \Gamma, y : B' \vdash N' : B''$ and $B = B' \supset B''$.
By induction hypothesis, $\Delta; \Gamma, y : B' \vdash [M/x]N' : B''$.
By the rule $\supset\mathsf{I}$, $\Delta; \Gamma \vdash \lambda y\!:\!B'.\, [M/x]N' : B' \supset B''$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = N_1\ N_2$: $\quad [M/x]N = [M/x]N_1\ [M/x]N_2$
By the rule $\supset\mathsf{E}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \Gamma \vdash N_1 : B' \supset B$ and $\Delta, x :: A; \Gamma \vdash N_2 : B'$.
By induction hypothesis, $\Delta; \Gamma \vdash [M/x]N_1 : B' \supset B$ and $\Delta; \Gamma \vdash [M/x]N_2 : B'$.
By the rule $\supset\mathsf{E}$, $\Delta; \Gamma \vdash [M/x]N_1\ [M/x]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = \mathsf{box}\ N'$: $\quad [M/x]N = \mathsf{box}\ [M/x]N'$
By the rule $\Box\mathsf{I}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \cdot \vdash N' : B'$ and $B = \Box B'$.
By induction hypothesis, $\Delta; \cdot \vdash [M/x]N' : B'$.
By the rule $\Box\mathsf{I}$, $\Delta; \Gamma \vdash \mathsf{box}\ [M/x]N' : \Box B'$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:
$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\Box\mathsf{E}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \Gamma \vdash N_1 : \Box B_1$ and $\Delta, x :: A, y :: B_1; \Gamma \vdash N_2 : B$.
By weakening, $\Delta; \cdot \vdash M : A$ implies $\Delta, y :: B_1; \cdot \vdash M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1; \Gamma \vdash [M/x]N_2 : B$.
By the rule $\Box\mathsf{E}$, $\Delta; \Gamma \vdash \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = \mathsf{cir}\ N'$: $\quad [M/x]N = \mathsf{cir}\ [M/x]N'$
By the rule $\bigcirc\mathsf{I}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \Gamma \vdash N' \sim B'$ and $B = \bigcirc B'$.
By induction hypothesis, $\Delta; \Gamma \vdash [M/x]N' \sim B'$.
By the rule $\bigcirc\mathsf{I}$, $\Delta; \Gamma \vdash \mathsf{cir}\ [M/x]N' : \bigcirc B'$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $\quad [M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\bigcirc\mathsf{E}$, $\Delta, x :: A; \Gamma \vdash N : B$ implies $\Delta, x :: A; \Gamma \vdash N_1 : \bigcirc B_1$ and $\Delta, x :: A, v \sim B_1; \Gamma \vdash N_2 : B$.
By weakening, $\Delta; \cdot \vdash M : A$ implies $\Delta, v \sim B_1; \cdot \vdash M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1; \Gamma \vdash [M/x]N_2 : B$.
By the rule $\bigcirc\mathsf{E}$, $\Delta; \Gamma \vdash \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [M/x]N : B$.

Proof of Proposition 2.4:
If the rule $\mathsf{Prim}{\sim}$ is used to deduce $\Delta, x :: A; \Gamma \vdash N \sim B$:
$\Delta, x :: A; \Gamma \vdash N : B$ and B is a primitive type.
By induction hypothesis, $\Delta; \Gamma \vdash [M/x]N : B$.

By the rule $\mathsf{Prim}\!\sim$, $\Delta;\Gamma \vdash [M/x]N \sim B$.
Now N cannot be an application $N_1\ N_2$ or a variable y.
Case $N = V$:
By the rule Val, $\Delta, x :: A; \Gamma \vdash N \sim B$ implies $\Delta, x :: A; \cdot \vdash N : B$.
By induction hypothesis, $\Delta; \cdot \vdash [M/x]N : B$.
By the rule Val, $\Delta;\Gamma \vdash [M/x]N \sim B$.
Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:
$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\Box\mathsf{E}'$, $\Delta, x :: A; \Gamma \vdash N \sim B$ implies $\Delta, x :: A; \Gamma \vdash N_1 : \Box B_1$ and $\Delta, x :: A, y :: B_1; \Gamma \vdash N_2 \sim B$.
By weakening, $\Delta; \cdot \vdash M : A$ implies $\Delta, y :: B_1; \cdot \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1; \Gamma \vdash [M/x]N_2 \sim B$.
By the rule $\Box\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B$.
Therefore $\Delta;\Gamma \vdash [M/x]N \sim B$.
Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $[M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
By the rule $\bigcirc\mathsf{E}'$, $\Delta, x :: A; \Gamma \vdash N \sim B$ implies $\Delta, x :: A; \Gamma \vdash N_1 : \bigcirc B_1$ and $\Delta, x :: A, v \sim B_1; \Gamma \vdash N_2 \sim B$.
By weakening, $\Delta; \cdot \vdash M : A$ implies $\Delta, v \sim B_1; \cdot \vdash M : A$.
By induction hypothesis, $\Delta;\Gamma \vdash [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1; \Gamma \vdash [M/x]N_2 \sim B$.
By the rule $\bigcirc\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B$.
Therefore $\Delta;\Gamma \vdash [M/x]N \sim B$. □

Lemma A.2.
If $\Delta; \cdot \vdash V : A$ *and* $\Delta, v \sim A; \Gamma \vdash N : B$, *then* $\Delta;\Gamma \vdash [V/v]N : B$.
If $\Delta; \cdot \vdash V : A$ *and* $\Delta, v \sim A; \Gamma \vdash N \sim B$, *then* $\Delta;\Gamma \vdash [V/v]N \sim B$.

Proof. By simulataneous induction on the structure of the derivation of $\Delta, v \sim A; \Gamma \vdash N : B$ and $\Delta, v \sim A; \Gamma \vdash N \sim B$.
Proof of the first clause:
Case $N = x$: $[V/v]N = x$
By the rule Cvar, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $x :: B \in \Delta, v \sim A$ or $x : B \in \Gamma$, which means $x :: B \in \Delta$ or $x : B \in \Gamma$.
By the rule Cvar, $\Delta;\Gamma \vdash x : B$.
Therefore $\Delta;\Gamma \vdash [V/v]N : B$.
Case $N = v$: $[V/v]N = V$
$\Delta; \cdot \vdash V : A$ implies $\Delta; \cdot \vdash [V/v]N : A$.
By weakening, $\Delta; \cdot \vdash [V/v]N : A$ implies $\Delta;\Gamma \vdash [V/v]N : A$.
$\Delta, v \sim A; \Gamma \vdash N : B$ implies $A = B$.
Therefore $\Delta;\Gamma \vdash [V/v]N : B$.
Case $N = w$, $w \neq v$: $[V/v]N = w$
By the rule Vvar, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $w \sim B \in \Delta, v \sim A$, which means $w \sim B \in \Delta$.
By the rule Vvar, $\Delta;\Gamma \vdash w : B$.
Therefore $\Delta;\Gamma \vdash [V/v]N : B$.
Case $N = \lambda x\!:\!B'.\,N'$, x not a free variable of V: $[V/v]N = \lambda x\!:\!B'.\,[V/v]N'$
By the rule $\supset\!\mathsf{I}$, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \Gamma, x : B' \vdash N' : B''$ and $B = B' \supset B''$.
By induction hypothesis, $\Delta; \Gamma, x : B' \vdash [V/v]N' : B''$.
By the rule $\supset\!\mathsf{I}$, $\Delta;\Gamma \vdash \lambda x\!:\!B'.\,[V/v]N' : B' \supset B''$.
Therefore $\Delta;\Gamma \vdash [V/v]N : B$.
Case $N = N_1\ N_2$: $[V/v]N = [V/v]N_1\ [V/v]N_2$
By the rule $\supset\!\mathsf{E}$, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \Gamma \vdash N_1 : B' \supset B$ and $\Delta, v \sim A; \Gamma \vdash N_2 : B'$.
By induction hypothesis, $\Delta;\Gamma \vdash [V/v]N_1 : B' \supset B$ and $\Delta;\Gamma \vdash [V/v]N_2 : B'$.

By the rule $\supset$E, $\Delta; \Gamma \vdash [V/v]N_1\ [V/v]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [V/v]N : B$.

Case $N = \mathsf{box}\ N'$: $\quad [V/v]N = \mathsf{box}\ [V/v]N'$
By the rule $\Box$I, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \cdot \vdash N' : B'$ and $B = \Box B'$.
By induction hypothesis, $\Delta; \cdot \vdash [V/v]N' : B'$.
By the rule $\Box$I, $\Delta; \Gamma \vdash \mathsf{box}\ [V/v]N' : \Box B'$.
Therefore $\Delta; \Gamma \vdash [V/v]N : B$.

Case $N = \mathsf{letbox}\ x = N_1\ \mathsf{in}\ N_2$, x not a free variable of V: $\quad [V/v]N = \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$
By the rule $\Box$E, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \Gamma \vdash N_1 : \Box B_1$ and $\Delta, v \sim A, x :: B_1; \Gamma \vdash N_2 : B$.
By weakening, $\Delta; \cdot \vdash V : A$ implies $\Delta, x :: B_1; \cdot \vdash V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N_1 : \Box B_1$ and $\Delta, x :: B_1; \Gamma \vdash [V/v]N_2 : B$.
By the rule $\Box$E, $\Delta; \Gamma \vdash \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [V/v]N : B$.

Case $N = \mathsf{cir}\ N'$: $\quad [V/v]N = \mathsf{cir}\ [V/v]N'$
By the rule $\bigcirc$I, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \Gamma \vdash N' \sim B'$ and $B = \bigcirc B'$.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N' \sim B'$.
By the rule $\bigcirc$I, $\Delta; \Gamma \vdash \mathsf{cir}\ [V/v]N' : \bigcirc B'$.
Therefore $\Delta; \Gamma \vdash [V/v]N : B$.

Case $N = \mathsf{letcir}\ w = N_1\ \mathsf{in}\ N_2, w \neq v, w$ not a free variable of V: $\quad [V/v]N = \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$
By the rule $\bigcirc$E, $\Delta, v \sim A; \Gamma \vdash N : B$ implies $\Delta, v \sim A; \Gamma \vdash N_1 : \bigcirc B_1$ and $\Delta, v \sim A, w \sim B_1; \Gamma \vdash N_2 : B$.
By weakening, $\Delta; \cdot \vdash V : A$ implies $\Delta, w \sim B_1; \cdot \vdash V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N_1 : \bigcirc B_1$. and $\Delta, w \sim B_1; \Gamma \vdash [V/v]N_2 : B$.
By the rule $\bigcirc$E, $\Delta; \Gamma \vdash \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 : B$.
Therefore $\Delta; \Gamma \vdash [V/v]N : B$.

Proof of the second clause:
If the rule Prim$\sim$ is used to deduce $\Delta, v \sim A; \Gamma \vdash N \sim B$:
$\Delta, v \sim A; \Gamma \vdash N : B$ and B is a primitive type.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N : B$.
By the rule Prim$\sim$, $\Delta; \Gamma \vdash [V/v]N \sim B$.
Now N cannot be an application $N_1\ N_2$ or a variable x.

Case $N = V'$:
By the rule Val, $\Delta, v \sim A; \Gamma \vdash N \sim B$ implies $\Delta, v \sim A; \cdot \vdash N : B$.
By induction hypothesis, $\Delta; \cdot \vdash [V/v]N : B$.
By the rule Val, $\Delta; \Gamma \vdash [V/v]N \sim B$.

Case $N = \mathsf{letbox}\ x = N_1\ \mathsf{in}\ N_2$, x not a free variable of V:
$[V/v]N = \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$
By the rule $\Box$E$'$, $\Delta, v \sim A; \Gamma \vdash N \sim B$ implies $\Delta, v \sim A; \Gamma \vdash N_1 : \Box B_1$ and $\Delta, v \sim A, x :: B_1; \Gamma \vdash N_2 \sim B$.
By weakening, $\Delta; \cdot \vdash V : A$ implies $\Delta, x :: B_1; \cdot \vdash V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N_1 : \Box B_1$ and $\Delta, x :: B_1; \Gamma \vdash [V/v]N_2 \sim B$.
By the rule $\Box$E$'$, $\Delta; \Gamma \vdash \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B$.
Therefore $\Delta; \Gamma \vdash [V/v]N \sim B$.

Case $N = \mathsf{letcir}\ w = N_1\ \mathsf{in}\ N_2, w \neq v, w$ not a free variable of V: $\quad [V/v]N = \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$
By the rule $\bigcirc$E$'$, $\Delta, v \sim A; \Gamma \vdash N \sim B$ implies $\Delta, v \sim A; \Gamma \vdash N_1 : \bigcirc B_1$ and $\Delta, v \sim A, w \sim B_1; \Gamma \vdash N_2 \sim B$.
By weakening, $\Delta; \cdot \vdash V : A$ implies $\Delta, w \sim B_1; \cdot \vdash V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash [V/v]N_1 : \bigcirc B_1$ and $\Delta, w \sim B_1; \Gamma \vdash [V/v]N_2 \sim B$.
By the rule $\bigcirc$E$'$, $\Delta; \Gamma \vdash \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B$.
Therefore $\Delta; \Gamma \vdash [V/v]N \sim B$. $\Box$

Proof of Proposition 2.3:

Proof. By induction on the structure of M.
Proof of the first clause:
Case $M = V$: $\quad \langle M/v\rangle N = [M/v]N$
By the rule Val, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\cdot \vdash M : A$.
By Lemma A.2, we have $\Delta;\Gamma \vdash [M/v]N : B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N : B$.
Case $M = \mathsf{letbox}\ x = M_1\ \mathsf{in}\ M_2$: $\quad \langle M/v\rangle N = \mathsf{letbox}\ x = M_1\ \mathsf{in}\ \langle M_2/v\rangle N$
By the rule $\Box\mathsf{E}'$, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\Gamma \vdash M_1 : \Box A_1$ and $\Delta, x :: A_1;\Gamma \vdash M_2 \sim A$.
By weakening, $\Delta, v \sim A;\Gamma \vdash N : B$ implies $\Delta, v \sim A, x :: A_1;\Gamma \vdash N : B$.
By induction hypothesis on M_2, $\Delta, x :: A_1;\Gamma \vdash \langle M_2/v\rangle N : B$.
By the rule $\Box\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letbox}\ x = M_1\ \mathsf{in}\ \langle M_2/v\rangle N : B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N : B$.
Case $M = \mathsf{letcir}\ w = M_1\ \mathsf{in}\ M_2$: $\quad \langle M/v\rangle N = \mathsf{letcir}\ w = M_1\ \mathsf{in}\ \langle M_2/v\rangle N$
By the rule $\bigcirc\mathsf{E}'$, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\Gamma \vdash M_1 : \bigcirc A_1$ and $\Delta, w \sim A_1;\Gamma \vdash M_2 \sim A$.
By weakening, $\Delta, v \sim A;\Gamma \vdash N : B$ implies $\Delta, v \sim A, w \sim A_1;\Gamma \vdash N : B$.
By induction hypothesis on M_2, $\Delta, w \sim A_1;\Gamma \vdash \langle M_2/v\rangle N : B$.
By the rule $\bigcirc\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letcir}\ w = M_1\ \mathsf{in}\ \langle M_2/v\rangle N : B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N : B$.
Proof of the second clause:
Case $M = V$: $\quad \langle M/v\rangle N = [M/v]N$
By the rule Val, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\cdot \vdash M : A$.
By Lemma A.2, we have $\Delta;\Gamma \vdash [M/v]N \sim B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N \sim B$.
Case $M = \mathsf{letbox}\ x = M_1\ \mathsf{in}\ M_2$: $\quad \langle M/v\rangle N = \mathsf{letbox}\ x = M_1\ \mathsf{in}\ \langle M_2/v\rangle N$
By the rule $\Box\mathsf{E}'$, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\Gamma \vdash M_1 : \Box A_1$ and $\Delta, x :: A_1;\Gamma \vdash M_2 \sim A$.
By weakening, $\Delta, v \sim A;\Gamma \vdash N \sim B$ implies $\Delta, v \sim A, x :: A_1;\Gamma \vdash N \sim B$.
By induction hypothesis on M_2, $\Delta, x :: A_1;\Gamma \vdash \langle M_2/v\rangle N \sim B$.
By the rule $\Box\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letbox}\ x = M_1\ \mathsf{in}\ \langle M_2/v\rangle N \sim B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N \sim B$.
Case $M = \mathsf{letcir}\ w = M_1\ \mathsf{in}\ M_2$: $\quad \langle M/v\rangle N = \mathsf{letcir}\ w = M_1\ \mathsf{in}\ \langle M_2/v\rangle N$
By the rule $\bigcirc\mathsf{E}'$, $\Delta;\Gamma \vdash M \sim A$ implies $\Delta;\Gamma \vdash M_1 : \bigcirc A_1$ and $\Delta, w \sim A_1;\Gamma \vdash M_2 \sim A$.
By weakening, $\Delta, v \sim A;\Gamma \vdash N \sim B$ implies $\Delta, v \sim A, w \sim A_1;\Gamma \vdash N \sim B$.
By induction hypothesis on M_2, $\Delta, w \sim A_1;\Gamma \vdash \langle M_2/v\rangle N \sim B$.
By the rule $\bigcirc\mathsf{E}'$, $\Delta;\Gamma \vdash \mathsf{letcir}\ w = M_1\ \mathsf{in}\ \langle M_2/v\rangle N \sim B$.
Therefore $\Delta;\Gamma \vdash \langle M/v\rangle N \sim B$.

□

Proof of Proposition 2.5:

Proof. By induction on the structure of the derivation of $\Delta;\Gamma \vdash M \sim A$.
Case $\dfrac{\Delta;\cdot \vdash V : A}{\Delta;\Gamma \vdash V \sim A}\ \mathsf{Val}$ and $M = V$:
By weakening, $\Delta;\cdot \vdash V : A$ implies $\Delta;\Gamma \vdash V : A$.
Therefore $\Delta;\Gamma \vdash M : A$.
Case $\dfrac{\Delta;\Gamma \vdash M_1 : \Box A_1 \quad \Delta, x :: A_1;\Gamma \vdash M_2 \sim A}{\Delta;\Gamma \vdash \mathsf{letbox}\ x = M_1\ \mathsf{in}\ M_2 \sim A}\ \Box\mathsf{E}'$ and $M = \mathsf{letbox}\ x = M_1\ \mathsf{in}\ M_2$:
By induction hypothesis on $\Delta, x :: A_1;\Gamma \vdash M_2 \sim A$, we have $\Delta, x :: A_1;\Gamma \vdash M_2 : A$.

By the rule $\Box$E, $\Delta; \Gamma \vdash \mathsf{letbox}\ x = M_1\ \mathsf{in}\ M_2 : A$

Therefore $\Delta; \Gamma \vdash M : A$.

Case $\dfrac{\Delta; \Gamma \vdash M_1 : \bigcirc A_1 \quad \Delta, v \sim A_1; \Gamma \vdash M_2 \sim A}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M_1\ \mathsf{in}\ M_2 \sim A}\ \bigcirc\mathsf{E}'$ and $M = \mathsf{letcir}\ v = M_1\ \mathsf{in}\ M_2$:

By induction hypothesis on $\Delta, v \sim A_1; \Gamma \vdash M_2 \sim A$, we have $\Delta, v \sim A_1; \Gamma \vdash M_2 : A$.

By the rule $\bigcirc$E, $\Delta; \Gamma \vdash \mathsf{letcir}\ v = M_1\ \mathsf{in}\ M_2 : A$.

Therefore $\Delta; \Gamma \vdash M : A$.

Case $\dfrac{\Delta; \Gamma \vdash M : A}{\Delta; \Gamma \vdash M \sim A}\ \mathsf{Prim}{\sim}$

The premise gives $\Delta; \Gamma \vdash M : A$. □

B Proofs of the properties of $\lambda_{\Box\bigcirc}{}^{W}$

Proof of Proposition 3.1:

Lemma B.1. $[M/x]V$ *is a value.*

Proof. By case analysis of V. □

Proof. By induction on the structure of the derivation of $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$.

If $N = V$ and the rule Val_W is used to deduce $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$:

$\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_{\omega'} N : B$.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega'} [M/x]N : B$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$ because $[M/x]N$ is a value by Lemma B.1.

If the rule $\mathsf{Prim}{\sim}_W$ is used to deduce $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$:

$\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N : B$ and B is a primitive type.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N : B$.

By the rule $\mathsf{Prim}{\sim}_W$, $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Now we assume that the rules Cvar_W and Cvar_W are not used to deduce $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$.

Case $N = x$: $\quad [M/x]N = M$

By the rule Cvar_W, $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $A = B$ and $\omega = \omega' = \omega''$.

$\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma \vdash_{\omega''} [M/x]N : A$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = y$, $y \neq x$: $\quad [M/x]N = y$

By the rule Cvar_W, $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $y :: B \in \Delta$ or $y : B \mathbin{@} \omega \in \Gamma, x : A \mathbin{@} \omega''$, and $\omega = \omega'$.

Since $y \neq x$, we have $y :: B \in \Delta$ or $y : B \mathbin{@} \omega \in \Gamma$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega y : B$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = v$: $\quad [M/x]N = v$

By the rule Vvar_W, $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $v \sim B \in \Delta$ or $v \sim B \mathbin{@} \omega \in \Gamma, x : A \mathbin{@} \omega''$, and $\omega = \omega'$.

Since $v \neq x$, we have $v \sim B \in \Delta$ or $v \sim B \mathbin{@} \omega \in \Gamma$.

By the rule Vvar_W, $\Delta; \Gamma \vdash_\omega v : B$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \lambda y{:}B'.\, N'$, $y \neq x$, y not a free variable of M: $\quad [M/x]N = \lambda y{:}B'.\, [M/x]N'$

By the rule $\supset\mathsf{I}_W$, $\Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta; \Gamma, x : A \mathbin{@} \omega'', y : B' \mathbin{@} \omega \vdash_\omega N' : B''$, $B = B' \supset B''$, and $\omega = \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, y : B' \text{ @ } \omega \vdash_{\omega''} M : A$.
By induction hypothesis, $\Delta; \Gamma, y : B' \text{ @ } \omega \vdash_{\omega} [M/x]N' : B''$.
By the rule $\supset\mathsf{I}_W$, $\Delta; \Gamma \vdash_{\omega} \lambda y{:}B'.\,[M/x]N' : B' \supset B''$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = N_1\ N_2$: $\quad [M/x]N = [M/x]N_1\ [M/x]N_2$
By the rule $\supset\mathsf{E}_W$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N_1 : B' \supset B$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N_2 : B'$, and $\omega = \omega'$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega} [M/x]N_1 : B' \supset B$ and $\Delta; \Gamma \vdash_{\omega} [M/x]N_2 : B'$.
By the rule $\supset\mathsf{E}_W$, $\Delta; \Gamma \vdash_{\omega} [M/x]N_1\ [M/x]N_2 : B$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{box}\ N'$: $\quad [M/x]N = \mathsf{box}\ [M/x]N'$
By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega^*} N' : B'$, $B = \Box B'$, and $\omega = \omega'$ where ω^* is a fresh node.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [M/x]N' : B'$.
By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma \vdash_{\omega} \mathsf{box}\ [M/x]N' : \Box B'$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{box}_{\omega^*}\ N'$: $\quad [M/x]N = \mathsf{box}_{\omega^*}\ [M/x]N'$
By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega^*} N' : B'$, $B = \Box_{\omega^*} B'$, and $\omega = \omega'$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [M/x]N' : B'$.
By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma \vdash_{\omega} \mathsf{box}_{\omega^*}\ [M/x]N' : \Box_{\omega^*} B'$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:
$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$
If the rule $\Box\mathsf{E}_W$ is used to deduce $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$,
$\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N_1 : \Box B_1$ and $\Delta, y :: B_1; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N_2 \sim B \text{ @ } \omega'$.
By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta, y :: B_1; \Gamma \vdash_{\omega''} M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega} [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1; \Gamma \vdash_{\omega} [M/x]N_2 \sim B \text{ @ } \omega'$.
By the rule $\Box\mathsf{E}_W$, $\Delta; \Gamma \vdash_{\omega} \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \text{ @ } \omega'$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.
If the rule $\Box\mathsf{E}'_W$ is used to deduce $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$,
$\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, x : A \text{ @ } \omega'', y : B_1 \text{ @ } \omega^* \vdash_{\omega} N_2 \sim B \text{ @ } \omega'$.
By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, y : B_1 \text{ @ } \omega^* \vdash_{\omega''} M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega} [M/x]N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, y : B_1 \text{ @ } \omega^* \vdash_{\omega} [M/x]N_2 \sim B \text{ @ } \omega'$.
By the rule $\Box\mathsf{E}'_W$, $\Delta; \Gamma \vdash_{\omega} \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \text{ @ } \omega'$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{cir}\ N'$: $\quad [M/x]N = \mathsf{cir}\ [M/x]N'$
By the rule $\bigcirc\mathsf{I}_W$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N' \sim B' \text{ @ } \omega^*$, $B = \bigcirc B'$, and $\omega = \omega'$ where ω^* is a fresh node.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega} [M/x]N' \sim B' \text{ @ } \omega^*$.
By the rule $\bigcirc\mathsf{I}_W$, $\Delta; \Gamma \vdash_{\omega} \mathsf{cir}\ [M/x]N' : \bigcirc B'$.
Therefore $\Delta; \Gamma \vdash_{\omega} [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{cir}_{\omega^*}\ N'$: $\quad [M/x]N = \mathsf{cir}_{\omega^*}\ [M/x]N'$
By the rule $\bigcirc\mathsf{I}'_W$, $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N \sim B \text{ @ } \omega'$ implies $\Delta; \Gamma, x : A \text{ @ } \omega'' \vdash_{\omega} N' \sim B' \text{ @ } \omega^*$, $B = \bigcirc_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N' \sim B' @ \omega^*$.
By the rule $\bigcirc\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [M/x]N' : \bigcirc_{\omega^*} B'$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $[M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$

If the rule $\bigcirc\mathsf{E}_W$ is used to deduce $\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N \sim B @ \omega'$,
$\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1; \Gamma, x : A @ \omega'' \vdash_\omega N_2 \sim B @ \omega'$.
By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta, v \sim B_1; \Gamma \vdash_{\omega''} M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1; \Gamma \vdash_\omega [M/x]N_2 \sim B @ \omega'$.
By the rule $\bigcirc\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B @ \omega'$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

If the rule $\bigcirc\mathsf{E}'_W$ is used to deduce $\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N \sim B @ \omega'$,
$\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, x : A @ \omega'' \vdash_\omega N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, x : A @ \omega'', v \sim B_1 @ \omega^* \vdash_\omega N_2 \sim B @ \omega'$.
By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, v \sim B_1 @ \omega^* \vdash_{\omega''} M : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, v \sim B_1 @ \omega^* \vdash_\omega [M/x]N_2 \sim B @ \omega'$.
By the rule $\bigcirc\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B @ \omega'$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$. □

Proof of Proposition 3.2:

Proof. By induction on the structure of the derivation of $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$.

If $N = V$ and the rule Val_W is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$:
$\Delta, x :: A; \Gamma \vdash_{\omega'} N : B$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega'} [M/x]N : B$.
By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$ because $[M/x]N$ is a value by Lemma B.1.

If the rule $\mathsf{Prim}{\sim}_W$ is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$:
$\Delta, x :: A; \Gamma \vdash_\omega N : B$ and B is a primitive type.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N : B$.
By the rule $\mathsf{Prim}{\sim}_W$, $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

Now we assume that the rules Cvar_W and Cvar_W are not used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$.

Case $N = x$: $[M/x]N = M$
By the rule Cvar_W, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $A = B$ and $\omega = \omega'$.
$\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma \vdash_\omega M : A$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

Case $N = y, y \neq x$: $[M/x]N = y$
By the rule Cvar_W, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $y :: B \in \Delta, x :: A$ or $y : B @ \omega \in \Gamma$, and $\omega = \omega'$.
Since $y \neq x$, we have $y :: B \in \Delta$ or $y : B @ \omega \in \Gamma$.
By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega y : B$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

Case $N = v$: $[M/x]N = v$
By the rule Vvar_W, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $v \sim B \in \Delta, x :: A$ or $v \sim B @ \omega \in \Gamma$, and $\omega = \omega'$.
Since $v \neq x$, we have $v \sim B \in \Delta$ or $v \sim B @ \omega \in \Gamma$.
By the rule Vvar_W, $\Delta; \Gamma \vdash_\omega v : B$.
Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B @ \omega'$.

Case $N = \lambda y{:}B'.\,N'$, $y \neq x$, y not a free variable of M: $[M/x]N = \lambda y{:}B'.\,[M/x]N'$

By the rule $\supset\mathsf{I}_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma, y : B' \mathbin{@} \omega \vdash_\omega N' : B''$, $B = B' \supset B''$, and $\omega = \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, y : B' \mathbin{@} \omega \vdash_{\omega''} M : A$.

By induction hypothesis, $\Delta; \Gamma, y : B' \mathbin{@} \omega \vdash_\omega [M/x]N' : B''$.

By the rule $\supset\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \lambda y{:}B'.\,[M/x]N' : B' \supset B''$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = N_1\ N_2$: $\quad [M/x]N = [M/x]N_1\ [M/x]N_2$

By the rule $\supset\mathsf{E}_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N_1 : B' \supset B$, $\Delta, x :: A; \Gamma \vdash_\omega N_2 : B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : B' \supset B$ and $\Delta; \Gamma \vdash_\omega [M/x]N_2 : B'$.

By the rule $\supset\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega [M/x]N_1\ [M/x]N_2 : B$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \mathsf{box}\ N'$: $\quad [M/x]N = \mathsf{box}\ [M/x]N'$

By the rule $\Box\mathsf{I}_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_{\omega^*} N' : B'$, $B = \Box B'$, and $\omega = \omega'$ where ω^* is a fresh node.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [M/x]N' : B'$.

By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}\ [M/x]N' : \Box B'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \mathsf{box}_{\omega^*}\ N'$: $\quad [M/x]N = \mathsf{box}_{\omega^*}\ [M/x]N'$

By the rule $\Box\mathsf{I}'_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_{\omega^*} N' : B'$, $B = \Box_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [M/x]N' : B'$.

By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}_{\omega^*}\ [M/x]N' : \Box_{\omega^*} B'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \mathsf{letbox}\ y = N_1\ \mathsf{in}\ N_2$, $y \neq x$, y not a free variable of M:

$[M/x]N = \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$

If the rule $\Box\mathsf{E}_W$ is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$,

$\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N_1 : \Box B_1$ and $\Delta, y :: B_1, x :: A; \Gamma \vdash_\omega N_2 \sim B \mathbin{@} \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta, y :: B_1; \Gamma \vdash_{\omega''} M : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \Box B_1$ and $\Delta, y :: B_1; \Gamma \vdash_\omega [M/x]N_2 \sim B \mathbin{@} \omega'$.

By the rule $\Box\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \mathbin{@} \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

If the rule $\Box\mathsf{E}'_W$ is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$,

$\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N_1 : \Box_{\omega^*} B_1$ and $\Delta, x :: A; \Gamma, y : B_1 \mathbin{@} \omega^* \vdash_\omega N_2 \sim B \mathbin{@} \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, y : B_1 \mathbin{@} \omega^* \vdash_{\omega''} M : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, y : B_1 \mathbin{@} \omega^* \vdash_\omega [M/x]N_2 \sim B \mathbin{@} \omega'$.

By the rule $\Box\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ y = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \mathbin{@} \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \mathsf{cir}\ N'$: $\quad [M/x]N = \mathsf{cir}\ [M/x]N'$

By the rule $\bigcirc\mathsf{I}_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N' \sim B' \mathbin{@} \omega^*$, $B = \bigcirc B'$, and $\omega = \omega'$ where ω^* is a fresh node.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N' \sim B' \mathbin{@} \omega^*$.

By the rule $\bigcirc\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [M/x]N' : \bigcirc B'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $N = \mathsf{cir}_{\omega^*}\ N'$: $\quad [M/x]N = \mathsf{cir}_{\omega^*}\ [M/x]N'$

By the rule $\bigcirc\mathsf{I}'_W$, $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \text{ @ } \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N' \sim B' \text{ @ } \omega^*$, $B = \bigcirc_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N' \sim B' \text{ @ } \omega^*$.

By the rule $\bigcirc\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [M/x]N' : \bigcirc_{\omega^*} B'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \text{ @ } \omega'$.

Case $N = \mathsf{letcir}\ v = N_1\ \mathsf{in}\ N_2$, v not a free variable of M: $\quad [M/x]N = \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2$

If the rule $\bigcirc\mathsf{E}_W$ is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \text{ @ } \omega'$,

$\Delta, x :: A; \Gamma \vdash_\omega N \sim B \text{ @ } \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1, x :: A; \Gamma \vdash_\omega N_2 \sim B \text{ @ } \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta, v \sim B_1; \Gamma \vdash_{\omega''} M : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \bigcirc B_1$ and $\Delta, v \sim B_1; \Gamma \vdash_\omega [M/x]N_2 \sim B \text{ @ } \omega'$.

By the rule $\bigcirc\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \text{ @ } \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \text{ @ } \omega'$.

If the rule $\bigcirc\mathsf{E}'_W$ is used to deduce $\Delta, x :: A; \Gamma \vdash_\omega N \sim B \text{ @ } \omega'$,

$\Delta, x :: A; \Gamma \vdash_\omega N \sim B \text{ @ } \omega'$ implies $\Delta, x :: A; \Gamma \vdash_\omega N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta, x :: A; \Gamma, v \sim B_1 \text{ @ } \omega^* \vdash_\omega N_2 \sim B \text{ @ } \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} M : A$ implies $\Delta; \Gamma, v \sim B_1 \text{ @ } \omega^* \vdash_{\omega''} M : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [M/x]N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, v \sim B_1 \text{ @ } \omega^* \vdash_\omega [M/x]N_2 \sim B \text{ @ } \omega'$.

By the rule $\bigcirc\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ v = [M/x]N_1\ \mathsf{in}\ [M/x]N_2 \sim B \text{ @ } \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [M/x]N \sim B \text{ @ } \omega'$. □

Proof of Proposition 3.3:

Proof. By induction on the structure of the derivation of $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$.

If N is a value and the rule Val_W is used to deduce $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$:

$\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_{\omega'} N : B$.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega'} [V/v]N : B$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \text{ @ } \omega'$ because $[V/v]N$ is a value by Lemma B.1.

If the rule $\mathsf{Prim}{\sim}_W$ is used to deduce $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$:

$\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N : B$ and B is a primitive type.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N : B$.

By the rule $\mathsf{Prim}{\sim}_W$, $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \text{ @ } \omega'$.

Now we assume that the rules Cvar_W and Cvar_W are not used to deduce $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$.

Case $N = x$: $\quad [V/v]N = x$

By the rule Cvar_W, $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$ implies $x :: B \in \Delta$ or $x : B \text{ @ } \omega \in \Gamma, v \sim A \text{ @ } \omega''$, and $\omega = \omega'$.

Since $x \neq v$, we have $x :: B \in \Delta$ or $x : B \text{ @ } \omega \in \Gamma$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega x : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \text{ @ } \omega'$.

Case $N = v$: $\quad [V/v]N = V$

By the rule Vvar_W, $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$ implies $A = B$ and $\omega = \omega' = \omega''$.

$\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma \vdash_\omega V : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \text{ @ } \omega'$.

Case $N = w$, $w \neq v$: $\quad [V/v]N = w$

By the rule Vvar_W, $\Delta; \Gamma, v \sim A \text{ @ } \omega'' \vdash_\omega N \sim B \text{ @ } \omega'$ implies $w \sim B \in \Delta$ or $w \sim B \text{ @ } \omega \in \Gamma, v \sim A \text{ @ } \omega''$, and $\omega = \omega'$.

Since $w \neq v$, we have $w \sim B \in \Delta$ or $w \sim B \text{ @ } \omega \in \Gamma$.

By the rule Vvar_W, $\Delta; \Gamma \vdash_\omega w : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \lambda x\!:\!B'.\,N'$, x not a free variable of V: $[V/v]N = \lambda x\!:\!B'.\,[V/v]N'$

By the rule $\supset\mathsf{I}_W$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'', x : B' @ \omega \vdash_\omega N' : B''$, $B = B' \supset B''$, and $\omega = \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, x : B' @ \omega \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma, x : B' @ \omega \vdash_\omega [V/v]N' : B''$.

By the rule $\supset\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \lambda x\!:\!B'.\,[V/v]N' : B' \supset B''$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = N_1\ N_2$: $[V/v]N = [V/v]N_1\ [V/v]N_2$

By the rule $\supset\mathsf{E}_W$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N_1 : B' \supset B$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N_2 : B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : B' \supset B$ and $\Delta; \Gamma \vdash_\omega [V/v]N_2 : B'$.

By the rule $\supset\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega [V/v]N_1\ [V/v]N_2 : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{box}\ N'$: $[V/v]N = \mathsf{box}\ [V/v]N'$

By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_{\omega^*} N' : B'$, $B = \Box B'$, and $\omega = \omega'$ where ω^* is a fresh node.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [V/v]N' : B'$.

By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}\ [V/v]N' : \Box B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{box}_{\omega^*}\ N'$: $[V/v]N = \mathsf{box}_{\omega^*}\ [V/v]N'$

By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_{\omega^*} N' : B'$, $B = \Box_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [V/v]N' : B'$.

By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}_{\omega^*}\ [V/v]N' : \Box_{\omega^*} B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{letbox}\ x = N_1\ \mathsf{in}\ N_2$, x not a free variable of V:

$[V/v]N = \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$

If the rule $\Box\mathsf{E}_W$ is used to deduce $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$,

$\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N_1 : \Box B_1$ and $\Delta, x :: B_1; \Gamma, v \sim A @ \omega'' \vdash_\omega N_2 \sim B @ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta, x :: B_1; \Gamma \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \Box B_1$ and $\Delta, x :: B_1; \Gamma \vdash_\omega [V/v]N_2 \sim B @ \omega'$.

By the rule $\Box\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B @ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

If the rule $\Box\mathsf{E}'_W$ is used to deduce $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$,

$\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, v \sim A @ \omega'', x : B_1 @ \omega^* \vdash_\omega N_2 \sim B @ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, x : B_1 @ \omega^* \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, x : B_1 @ \omega^* \vdash_\omega [V/v]N_2 \sim B @ \omega'$.

By the rule $\Box\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B @ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{cir}\ N'$: $[V/v]N = \mathsf{cir}\ [V/v]N'$

By the rule $\bigcirc\mathsf{I}_W$, $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N \sim B @ \omega'$ implies $\Delta; \Gamma, v \sim A @ \omega'' \vdash_\omega N' \sim B' @ \omega^*$, $B = \bigcirc B'$, and $\omega = \omega'$ where ω^* is a fresh node.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N' \sim B' @ \omega^*$.

By the rule $\bigcirc\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [V/v]N' : \bigcirc B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

Case $N = \mathsf{cir}_{\omega^*} \ N'$: $\quad [V/v]N = \mathsf{cir}_{\omega^*} \ [V/v]N'$

By the rule $\bigcirc\mathsf{I}'_W$, $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N \sim B \ @ \ \omega'$ implies $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N' \sim B' \ @ \ \omega^*$, $B = \bigcirc_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N' \sim B' \ @ \ \omega^*$.

By the rule $\bigcirc\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir} \ [V/v]N' : \bigcirc_{\omega^*} B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

Case $N = \mathsf{letcir} \ w = N_1 \ \mathsf{in} \ N_2, w \neq v, w$ not a free variable of V: $\quad [V/v]N = \mathsf{letcir} \ w = [V/v]N_1 \ \mathsf{in} \ [V/v]N_2$

If the rule $\bigcirc\mathsf{E}_W$ is used to deduce $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N \sim B \ @ \ \omega'$,

$\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N \sim B \ @ \ \omega'$ implies $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N_1 : \bigcirc B_1$ and $\Delta, w \sim B_1; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N_2 \sim B \ @ \ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta, w \sim B_1; \Gamma \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \bigcirc B_1$ and $\Delta, w \sim B_1; \Gamma \vdash_\omega [V/v]N_2 \sim B \ @ \ \omega'$.

By the rule $\bigcirc\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir} \ w = [V/v]N_1 \ \mathsf{in} \ [V/v]N_2 \sim B \ @ \ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

If the rule $\bigcirc\mathsf{E}'_W$ is used to deduce $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N \sim B \ @ \ \omega'$,

$\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N \sim B \ @ \ \omega'$ implies $\Delta; \Gamma, v \sim A \ @ \ \omega'' \vdash_\omega N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, v \sim A \ @ \ \omega'', w \sim B_1 \ @ \ \omega^* \vdash_\omega N_2 \sim B \ @ \ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, w \sim B_1 \ @ \ \omega^* \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, w \sim B_1 \ @ \ \omega^* \vdash_\omega [V/v]N_2 \sim B \ @ \ \omega'$.

By the rule $\bigcirc\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir} \ w = [V/v]N_1 \ \mathsf{in} \ [V/v]N_2 \sim B \ @ \ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$. $\square$

Proof of Proposition 3.4:

Proof. By induction on the structure of the derivation of $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$.

If N is a value and the rule Val_W is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$:

$\Delta, v \sim A; \Gamma \vdash_{\omega'} N : B$.

By induction hypothesis, $\Delta; \Gamma \vdash_{\omega'} [V/v]N : B$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$ because $[V/v]N$ is a value by Lemma B.1.

If the rule $\mathsf{Prim}{\sim}_W$ is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$:

$\Delta, v \sim A; \Gamma \vdash_\omega N : B$ and B is a primitive type.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N : B$.

By the rule $\mathsf{Prim}{\sim}_W$, $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

Now we assume that the rules Cvar_W and Cvar_W are not used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$.

Case $N = x$: $\quad [V/v]N = x$

By the rule Cvar_W, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$ implies $x :: B \in \Delta, v \sim A$ or $x : B \ @ \ \omega \in \Gamma$, and $\omega = \omega'$.

Since $x \neq v$, we have $x :: B \in \Delta$ or $x : B \ @ \ \omega \in \Gamma$.

By the rule Cvar_W, $\Delta; \Gamma \vdash_\omega x : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

Case $N = v$: $\quad [V/v]N = V$

By the rule Vvar_W, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$ implies $A = B$ and $\omega = \omega'$.

$\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma \vdash_\omega V : B$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B \ @ \ \omega'$.

Case $N = w, w \neq v$: $\quad [V/v]N = w$

By the rule Vvar_W, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B \ @ \ \omega'$ implies $w \sim B \in \Delta, v \sim A$ or $w \sim B \ @ \ \omega \in \Gamma$, and $\omega = \omega'$.

Since $w \neq v$, we have $w \sim B \in \Delta$ or $w \sim B$ @ $\omega \in \Gamma$.
By the rule Vvar_W, $\Delta; \Gamma \vdash_\omega w : B$.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = \lambda x\!:\!B'.\, N'$, x not a free variable of V: $\quad [V/v]N = \lambda x\!:\!B'.\, [V/v]N'$

By the rule $\supset\!\mathsf{I}_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma, x : B'$ @ $\omega \vdash_\omega N' : B''$, $B = B' \supset B''$, and $\omega = \omega'$.
By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, x : B'$ @ $\omega \vdash_{\omega''} V : A$.
By induction hypothesis, $\Delta; \Gamma, x : B'$ @ $\omega \vdash_\omega [V/v]N' : B''$.
By the rule $\supset\!\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \lambda x\!:\!B'.\, [V/v]N' : B' \supset B''$.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = N_1\ N_2$: $\quad [V/v]N = [V/v]N_1\ [V/v]N_2$

By the rule $\supset\!\mathsf{E}_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_\omega N_1 : B' \supset B$, $\Delta, v \sim A; \Gamma \vdash_\omega N_2 : B'$, and $\omega = \omega'$.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : B' \supset B$ and $\Delta; \Gamma \vdash_\omega [V/v]N_2 : B'$.
By the rule $\supset\!\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega [V/v]N_1\ [V/v]N_2 : B$.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = \mathsf{box}\ N'$: $\quad [V/v]N = \mathsf{box}\ [V/v]N'$

By the rule $\Box\mathsf{I}_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_{\omega^*} N' : B'$, $B = \Box B'$, and $\omega = \omega'$ where ω^* is a fresh node.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [V/v]N' : B'$.
By the rule $\Box\mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}\ [V/v]N' : \Box B'$.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = \mathsf{box}_{\omega^*}\ N'$: $\quad [V/v]N = \mathsf{box}_{\omega^*}\ [V/v]N'$

By the rule $\Box\mathsf{I}'_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_{\omega^*} N' : B'$, $B = \Box_{\omega^*} B'$, and $\omega = \omega'$.
By induction hypothesis, $\Delta; \Gamma \vdash_{\omega^*} [V/v]N' : B'$.
By the rule $\Box\mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{box}_{\omega^*}\ [V/v]N' : \Box_{\omega^*} B'$.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = \mathsf{letbox}\ x = N_1\ \mathsf{in}\ N_2$, x not a free variable of V:
$[V/v]N = \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$
If the rule $\Box\mathsf{E}_W$ is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω',
$\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_\omega N_1 : \Box B_1$ and $\Delta, x :: B_1, v \sim A; \Gamma \vdash_\omega N_2 \sim B$ @ ω'.
By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta, x :: B_1; \Gamma \vdash_{\omega''} V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \Box B_1$ and $\Delta, x :: B_1; \Gamma \vdash_\omega [V/v]N_2 \sim B$ @ ω'.
By the rule $\Box\mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B$ @ ω'.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.
If the rule $\Box\mathsf{E}'_W$ is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω',
$\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_\omega N_1 : \Box_{\omega^*} B_1$ and $\Delta, v \sim A; \Gamma, x : B_1$ @ $\omega^* \vdash_\omega N_2 \sim B$ @ ω'.
By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, x : B_1$ @ $\omega^* \vdash_{\omega''} V : A$.
By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \Box_{\omega^*} B_1$ and $\Delta; \Gamma, x : B_1$ @ $\omega^* \vdash_\omega [V/v]N_2 \sim B$ @ ω'.
By the rule $\Box\mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B$ @ ω'.
Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B$ @ ω'.

Case $N = \mathsf{cir}\ N'$: $\quad [V/v]N = \mathsf{cir}\ [V/v]N'$

By the rule $\bigcirc\mathsf{I}_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B$ @ ω' implies $\Delta, v \sim A; \Gamma \vdash_\omega N' \sim B'$ @ ω^*, $B = \bigcirc B'$, and $\omega = \omega'$ where ω^* is a fresh node.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N' \sim B' @ \omega^*$.

By the rule $\bigcirc \mathsf{I}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [V/v]N' : \bigcirc B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{cir}_{\omega^*}\ N'$: $\quad [V/v]N = \mathsf{cir}_{\omega^*}\ [V/v]N'$

By the rule $\bigcirc \mathsf{I}'_W$, $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $\Delta, v \sim A; \Gamma \vdash_\omega N' \sim B' @ \omega^*$, $B = \bigcirc_{\omega^*} B'$, and $\omega = \omega'$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N' \sim B' @ \omega^*$.

By the rule $\bigcirc \mathsf{I}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{cir}\ [V/v]N' : \bigcirc_{\omega^*} B'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

Case $N = \mathsf{letcir}\ w = N_1\ \mathsf{in}\ N_2, w \neq v, w$ not a free variable of V: $\quad [V/v]N = \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2$

If the rule $\bigcirc \mathsf{E}_W$ is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B @ \omega'$,

$\Delta, v \sim A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $\Delta, v \sim A; \Gamma \vdash_\omega N_1 : \bigcirc B_1$ and $\Delta, w \sim B_1, v \sim A; \Gamma \vdash_\omega N_2 \sim B @ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta, w \sim B_1; \Gamma \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \bigcirc B_1$ and $\Delta, w \sim B_1; \Gamma \vdash_\omega [V/v]N_2 \sim B @ \omega'$.

By the rule $\bigcirc \mathsf{E}_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B @ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$.

If the rule $\bigcirc \mathsf{E}'_W$ is used to deduce $\Delta, v \sim A; \Gamma \vdash_\omega N \sim B @ \omega'$,

$\Delta, v \sim A; \Gamma \vdash_\omega N \sim B @ \omega'$ implies $\Delta, v \sim A; \Gamma \vdash_\omega N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta, v \sim A; \Gamma, w \sim B_1 @ \omega^* \vdash_\omega N_2 \sim B @ \omega'$.

By weakening, $\Delta; \Gamma \vdash_{\omega''} V : A$ implies $\Delta; \Gamma, w \sim B_1 @ \omega^* \vdash_{\omega''} V : A$.

By induction hypothesis, $\Delta; \Gamma \vdash_\omega [V/v]N_1 : \bigcirc_{\omega^*} B_1$ and $\Delta; \Gamma, w \sim B_1 @ \omega^* \vdash_\omega [V/v]N_2 \sim B @ \omega'$.

By the rule $\bigcirc \mathsf{E}'_W$, $\Delta; \Gamma \vdash_\omega \mathsf{letcir}\ w = [V/v]N_1\ \mathsf{in}\ [V/v]N_2 \sim B @ \omega'$.

Therefore $\Delta; \Gamma \vdash_\omega [V/v]N \sim B @ \omega'$. □

Proof of Proposition 3.5:

Proof. By simultaneous induction on the structure of the derivation of $\Delta; \Gamma \vdash M : A$ and $\Delta; \Gamma \vdash M \sim A$. (Below we reuse metavarible M and type A.)

Case $\dfrac{x :: A \in \Delta \quad \text{or} \quad x : A \in \Gamma}{\Delta; \Gamma \vdash x : A}\ \mathsf{Cvar}$:

$x :: A \in \Delta$ or $x : A \in \Gamma$ implies $x :: A \in \Delta$ or $x : A @ \omega \in [\Gamma]^\omega$.

Then,

$$\frac{x :: A \in \Delta \quad \text{or} \quad x : A @ \omega \in [\Gamma]^\omega}{\Delta; [\Gamma]^\omega \vdash_\omega x : A}\ \mathsf{Cvar}_W$$

Case $\dfrac{v \sim A \in \Delta}{\Delta; \Gamma \vdash v : A}\ \mathsf{Vvar}$:

$v \sim A \in \Delta$ implies $v \sim A \in \Delta$ or $v \sim A @ \omega \in [\Gamma]^\omega$.

Then,

$$\frac{v \sim A \in \Delta \quad \text{or} \quad v \sim A @ \omega \in [\Gamma]^\omega}{\Delta; [\Gamma]^\omega \vdash_\omega v : A}\ \mathsf{Vvar}_W$$

Case $\dfrac{\Delta; \cdot \vdash V : A}{\Delta; \Gamma \vdash V \sim A}\ \mathsf{Val}$:

By induction hypothesis on $\Delta; \cdot \vdash V : A$, we have $\Delta; \cdot \vdash_{\omega'} V : A$.

By weakening, $\Delta; \cdot \vdash_{\omega'} V : A$ implies $\Delta; [\Gamma]^\omega \vdash_{\omega'} V : A$.

Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_{\omega'} V : A}{\Delta; [\Gamma]^\omega \vdash_\omega V \sim A \ @\ \omega'}\ \mathsf{Val}_W$$

Case $\dfrac{\Delta; \Gamma, x : A \vdash M : B}{\Delta; \Gamma \vdash \lambda x{:}A.\, M : A \supset B}\ \supset\mathsf{I}$:

By induction hypothesis on $\Delta; \Gamma, x : A \vdash M : B$, we have $\Delta; [\Gamma]^\omega, x : A \ @\ \omega \vdash_\omega M : B$.
Then,

$$\frac{\Delta; [\Gamma]^\omega, x : A \ @\ \omega \vdash_\omega M : B}{\Delta; [\Gamma]^\omega \vdash_\omega \lambda x{:}A.\, M : A \supset B}\ \supset\mathsf{I}_W$$

Case $\dfrac{\Delta; \Gamma \vdash M : A \supset B \quad \Delta; \Gamma \vdash N : A}{\Delta; \Gamma \vdash M\ N : B}\ \supset\mathsf{E}$:

By induction hypothesis on $\Delta; \Gamma \vdash M : A \supset B$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : A \supset B$.
By induction hypothesis on $\Delta; \Gamma \vdash N : A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega N : A$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : A \supset B \quad \Delta; [\Gamma]^\omega \vdash_\omega N : A}{\Delta; [\Gamma]^\omega \vdash_\omega M\ N : B}\ \supset\mathsf{E}_W$$

Case $\dfrac{\Delta; \cdot \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}$:

By induction hypothesis on $\Delta; \cdot \vdash M : A$, we have $\Delta; \cdot \vdash_{\omega'} M : A$.
By weakening, $\Delta; \cdot \vdash_{\omega'} M : A$ implies $\Delta; [\Gamma]^\omega \vdash_{\omega'} M : A$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_{\omega'} M : A}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}_W$$

Case $\dfrac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N : B}\ \Box\mathsf{E}$:

By induction hypothesis on $\Delta; \Gamma \vdash M : \Box A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : \Box A$.
By induction hypothesis on $\Delta, x :: A; \Gamma \vdash N : B$, we have $\Delta, x :: A; [\Gamma]^\omega \vdash_\omega N : B$.
$\Delta, x :: A; [\Gamma]^\omega \vdash_\omega N : B$ is equivalent to $\Delta, x :: A; [\Gamma]^\omega \vdash_\omega N \sim B \ @\ \omega$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : \Box A \quad \Delta, x :: A; [\Gamma]^\omega \vdash_\omega N \sim B \ @\ \omega}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \ @\ \omega}\ \Box\mathsf{E}_W$$

$\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \ @\ \omega$ is equivalent to $\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N : B$.

Case $\dfrac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, x :: A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B}\ \Box\mathsf{E}'$:

By induction hypothesis on $\Delta; \Gamma \vdash M : \Box A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : \Box A$.
By induction hypothesis on $\Delta, x :: A; \Gamma \vdash N \sim B$, we have $\Delta, x :: A; [\Gamma]^\omega \vdash_\omega N \sim B \ @\ \omega'$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : \Box A \quad \Delta, x :: A; [\Gamma]^\omega \vdash_\omega N \sim B \ @\ \omega'}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \ @\ \omega'}\ \Box\mathsf{E}_W$$

Case $\dfrac{\Delta; \Gamma \vdash M \sim A}{\Delta; \Gamma \vdash \mathsf{cir}\ M : \bigcirc A}\ \bigcirc\mathsf{I}$:

By induction hypotheis on $\Delta; \Gamma \vdash M \sim A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M \sim A \ @\ \omega'$.

Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M \sim A \mathbin{@} \omega'}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{cir}\ M : \bigcirc A}\ \bigcirc\mathsf{I}_W$$

Case $\dfrac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N : B}\ \bigcirc\mathsf{E}$:

By induction hypothesis on $\Delta; \Gamma \vdash M : \bigcirc A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : \bigcirc A$.
By induction hypothesis on $\Delta, v \sim A; \Gamma \vdash N : B$, we have $\Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N : B$.
$\Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N : B$ is equivalent to $\Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N \sim B \mathbin{@} \omega$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : \bigcirc A \quad \Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N \sim B \mathbin{@} \omega}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega}\ \bigcirc\mathsf{E}_W$$

$\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega$ is equivalent to $\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N : B$.

Case $\dfrac{\Delta; \Gamma \vdash M : \bigcirc A \quad \Delta, v \sim A; \Gamma \vdash N \sim B}{\Delta; \Gamma \vdash \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B}\ \bigcirc\mathsf{E}'$:

By induction hypothesis on $\Delta; \Gamma \vdash M : \bigcirc A$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : \bigcirc A$.
By induction hypothesis on $\Delta, v \sim A; \Gamma \vdash N \sim B$, we have $\Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N \sim B \mathbin{@} \omega'$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : \bigcirc A \quad \Delta, v \sim A; [\Gamma]^\omega \vdash_\omega N \sim B \mathbin{@} \omega'}{\Delta; [\Gamma]^\omega \vdash_\omega \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \bigcirc\mathsf{E}_W$$

Case $\dfrac{\Delta; \Gamma \vdash M : A_{prim}}{\Delta; \Gamma \vdash M \sim A_{prim}}\ \mathsf{Prim}{\sim}$:

By induction hypothesis on $\Delta; \Gamma \vdash M : A_{prim}$, we have $\Delta; [\Gamma]^\omega \vdash_\omega M : A_{prim}$.
Then,

$$\frac{\Delta; [\Gamma]^\omega \vdash_\omega M : A_{prim}}{\Delta; [\Gamma]^\omega \vdash_\omega M \sim A_{prim} \mathbin{@} \omega'}\ \mathsf{Prim}{\sim}_W$$

□

C Proofs of the type safety of $\lambda_{\Box\bigcirc}{}^W$

Proposition C.1.
If $\Lambda; \Delta; \Gamma \vdash_{\omega''} M : A$ *and* $\Lambda; \Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Lambda; \Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Proof. By induction on the structure of the derivation of $\Lambda; \Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$. □

Proposition C.2.
If $\Lambda; \Delta; \Gamma \vdash_{\omega''} M : A$ *for any node* ω'' *and* $\Lambda; \Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Lambda; \Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Proof. By induction on the structure of the derivation of $\Lambda; \Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$. □

Proposition C.3.
If $\Lambda; \Delta; \Gamma \vdash_{\omega''} V : A$ *and* $\Lambda; \Delta; \Gamma, v \sim A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Lambda; \Delta; \Gamma \vdash_\omega [V/v]N \sim B \mathbin{@} \omega'$.

Proof. By induction on the structure of the derivation of $\Lambda; \Delta; \Gamma, v \sim A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'$. □

Proposition C.4.
If $\Lambda; \Delta; \Gamma \vdash_{\omega''} V : A$ *for any node* ω'' *and* $\Lambda; \Delta, v \sim A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$*, then* $\Lambda; \Delta; \Gamma \vdash_\omega [V/v]N \sim B \mathbin{@} \omega'$.

Proof. By induction on the structure of the derivation of $\Lambda; \Delta, v \sim A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'$. $\square$

Proofs of Propositions C.1 to C.4 are similar to those of Propositions 3.1 to 3.4. Cases for communication constructs are also straightforward, as substitutions on communication constructs are all structural:

$$
\begin{array}{rcl}
[M/x]() & = & () \\
[M/x]\mathsf{eval}\ N & = & \mathsf{eval}\ [M/x]N \\
[M/x]\mathsf{future}\ N & = & \mathsf{future}\ [M/x]N \\
[M/x]\mathsf{syncvar}\ \gamma & = & \mathsf{syncvar}\ \gamma \\
[M/x]\mathsf{syncwith}\ N & = & \mathsf{syncwith}\ [M/x]N \\
[M/x]\mathsf{nil} & = & \mathsf{nil} \\
[M/x]V_1 :: V_2 & = & [M/x]V_1 :: [M/x]V_2 \\
[M/x]\mathsf{chanvar}\ \gamma & = & \mathsf{chanvar}\ \gamma \\
[M/x]\mathsf{newchan}_A & = & \mathsf{newchan}_A \\
[M/x]\mathsf{readchan}\ N & = & \mathsf{readchan}\ [M/x]N \\
[M/x]\mathsf{writechan}\ N_1\ N_2 & = & \mathsf{writechan}\ [M/x]N_1\ [M/x]N_2
\end{array}
$$

Lemma C.5. *If* $\Lambda; \Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$ *and* $M \longrightarrow N$*, then* $\Lambda; \Delta; \Gamma \vdash_\omega N \sim A \mathbin{@} \omega'$.

Proof. By induction on the structure of the derivation of $\Lambda; \Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$. (Below we reuse metavariable M and type A.)

Case $\dfrac{\Lambda; \Delta; \Gamma \vdash_\omega M : A_{prim}}{\Lambda; \Delta; \Gamma \vdash_\omega M \sim A_{prim} \mathbin{@} \omega'}\ \mathsf{Prim}{\sim}_W\ (\omega \neq \omega')$:

By induction hypothesis, $\Lambda; \Delta; \Gamma \vdash_\omega N : A_{prim}$.

By the rule $\mathsf{Prim}{\sim}_W$, $\Lambda; \Delta; \Gamma \vdash_\omega N \sim A_{prim} \mathbin{@} \omega'$.

Now we now assume that the rule $\mathsf{Prim}{\sim}_W$ is not used to derive $\Lambda; \Delta; \Gamma \vdash_\omega M \sim A \mathbin{@} \omega'$.

Case $(\lambda x\!:\!A.\, N)\ M \rightarrow_{\beta\supset} [M/x]N$:

The only possible derivation is:

$$
\dfrac{\dfrac{\Lambda; \Delta; \Gamma, x : A \mathbin{@} \omega \vdash_\omega N : B}{\Lambda; \Delta; \Gamma \vdash_\omega \lambda x\!:\!A.\, N : A \supset B}\ {\supset}\mathsf{I}_W \quad \Lambda; \Delta; \Gamma \vdash_\omega M : A}{\Lambda; \Delta; \Gamma \vdash_\omega (\lambda x\!:\!A.\, N)\ M : B}\ {\supset}\mathsf{E}_W
$$

By Proposition C.1, $\Lambda; \Delta; \Gamma \vdash_\omega [M/x]N : B$.

Case $\mathsf{letbox}\ x = \mathsf{box}\ M\ \mathsf{in}\ N \rightarrow_{\beta\Box} [M/x]N$:

The only possible derivation is:

$$
\dfrac{\dfrac{\mathit{fresh}\ \omega'' \quad \Lambda; \Delta; \Gamma \vdash_{\omega''} M : A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}_W \quad \Lambda; \Delta, x :: A; \Gamma \vdash_\omega N \sim B \mathbin{@} \omega'}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = \mathsf{box}\ M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \Box\mathsf{E}_W
$$

By Proposition C.2, $\Lambda; \Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $\mathsf{letbox}\ x = \mathsf{box}_{\omega''}\ M\ \mathsf{in}\ N \rightarrow_{\beta\Box'} [M/x]N$:

The only possible derivation is:

$$
\dfrac{\dfrac{\Lambda; \Delta; \Gamma \vdash_{\omega''} M : A}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{box}_{\omega''}\ M : \Box_{\omega''} A}\ \Box\mathsf{I}'_W \quad \Lambda; \Delta; \Gamma, x : A \mathbin{@} \omega'' \vdash_\omega N \sim B \mathbin{@} \omega'}{\Lambda; \Delta; \Gamma \vdash_\omega \mathsf{letbox}\ x = \mathsf{box}_{\omega''}\ M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \Box\mathsf{E}'_W
$$

By Proposition C.1, $\Lambda; \Delta; \Gamma \vdash_\omega [M/x]N \sim B \mathbin{@} \omega'$.

Case $\mathsf{letcir}\ v = \mathsf{cir}\ V\ \mathsf{in}\ N {\rightarrow_{\beta\bigcirc}} [V/v]N$:

The only possible derivation is:

$$\dfrac{fresh\ \omega'' \quad \dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega''} V : A}{\Lambda;\Delta;\Gamma\vdash_{\omega} V \sim A\ @\ \omega''}\ \mathsf{Val}_W}{\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{cir}\ V : \bigcirc A \quad \mathsf{\bigcirc I}_W \qquad \Lambda;\Delta, v\sim A;\Gamma\vdash_{\omega} N \sim B\ @\ \omega'}{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letcir}\ v = \mathsf{cir}\ V\ \mathsf{in}\ N \sim B\ @\ \omega'}\ \mathsf{\bigcirc E}_W}$$

By Proposition C.4, $\Lambda;\Delta;\Gamma\vdash_{\omega} [V/v]N \sim B\ @\ \omega'$.

Case $\mathsf{letcir}\ v = \mathsf{cir}_{\omega''}\ V\ \mathsf{in}\ N \rightarrow_{\beta\bigcirc'} [V/v]N$:

The only possible derivation is:

$$\dfrac{\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} V \sim A\ @\ \omega''}{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{cir}_{\omega''}\ V : \bigcirc_{\omega''} A}\ \mathsf{\bigcirc I}'_W \qquad \Lambda;\Delta;\Gamma, v\sim A\ @\ \omega''\vdash_{\omega} N \sim B\ @\ \omega'}{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letcir}\ v = \mathsf{cir}_{\omega''}\ V\ \mathsf{in}\ N \sim B\ @\ \omega'}\ \mathsf{\bigcirc E}'_W$$

From $\Lambda;\Delta;\Gamma\vdash_{\omega} V \sim A\ @\ \omega''$, we have $\Lambda;\Delta;\Gamma\vdash_{\omega''} V : A$, whether $\omega = \omega''$ or $\omega \neq \omega''$.

By Proposition C.3, $\Lambda;\Delta;\Gamma\vdash_{\omega} [V/v]N \sim B\ @\ \omega'$. □

Lemma C.6.

Consider two terms M_0 and N_0 such that $\Lambda;\Delta;\Gamma\vdash_{\omega} M_0 \sim A_0\ @\ \omega_0$ implies $\Lambda;\Delta;\Gamma\vdash_{\omega} N_0 \sim A_0\ @\ \omega_0$ for any A_0 and ω_0.
If $\Lambda;\Delta;\Gamma\vdash_{\omega} M \sim A\ @\ \omega'$, then for any κ such that $M = \kappa[M_0]$, it holds $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa[N_0] \sim A\ @\ \omega'$.

Proof. If $\kappa = []$, then $M = M_0$ and $\kappa[N_0] = N_0$. Hence $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa[N_0] \sim A\ @\ \omega'$ holds by the assumption on M_0 and N_0.

Suppose $\kappa \neq []$, which means that $M \neq x$, $M \neq v$, and $M \neq V$.

Now we apply induction on the structure of $\Lambda;\Delta;\Gamma\vdash_{\omega} M \sim A\ @\ \omega'$. (Below we reuse metavarible M and type A.)

Case $\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} M : A_{prim}}{\Lambda;\Delta;\Gamma\vdash_{\omega} M \sim A_{prim}\ @\ \omega'}\ \mathsf{Prim}{\sim}_W\ (\omega\neq\omega')$, $M = \kappa[M_0]$:

By induction hypothesis, $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa[N_0] : A_{prim}$.

By the rule $\mathsf{Prim}{\sim}_W$, $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa[N_0] \sim A_{prim}\ @\ \omega'$.

Case $\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} M : A\supset B \quad \Lambda;\Delta;\Gamma\vdash_{\omega} N : A}{\Lambda;\Delta;\Gamma\vdash_{\omega} M\ N : B}\ \mathsf{\supset E}_W$, $M\ N = \kappa[M_0] = \kappa'[M_0]\ N$:

By induction hypothesis on $\Lambda;\Delta;\Gamma\vdash_{\omega} M : A\supset B$, we have $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa'[N_0] : A\supset B$.

By the rule $\mathsf{\supset E}_W$, $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa'[N_0]\ N : B$, and $\kappa'[N_0]\ N = \kappa[N_0]$.

Case $\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} M : \Box A \quad \Lambda;\Delta, x :: A;\Gamma\vdash_{\omega} N \sim B\ @\ \omega'}{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B\ @\ \omega'}\ \mathsf{\Box E}_W$,

$\mathsf{letbox}\ x = M\ \mathsf{in}\ N = \kappa[M_0] = \mathsf{letbox}\ x = \kappa'[M_0]\ \mathsf{in}\ N$:

By induction hypothesis on $\Lambda;\Delta;\Gamma\vdash_{\omega} M : \Box A$, we have $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa'[N_0] : \Box A$.

By the rule $\mathsf{\Box E}_W$, $\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letbox}\ x = \kappa'[N_0]\ \mathsf{in}\ N \sim B\ @\ \omega'$, and $\mathsf{letbox}\ x = \kappa'[N_0]\ \mathsf{in}\ N = \kappa[N_0]$.

Case $\mathsf{\Box E}'_W$ is similar to Case $\mathsf{\Box E}_W$.

Case $\dfrac{\Lambda;\Delta;\Gamma\vdash_{\omega} M : \bigcirc A \quad \Lambda;\Delta, v\sim A;\Gamma\vdash_{\omega} N \sim B\ @\ \omega'}{\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B\ @\ \omega'}\ \mathsf{\bigcirc E}_W$

If $\mathsf{letcir}\ v = M\ \mathsf{in}\ N = \kappa[M_0] = \mathsf{letcir}\ v = \kappa'[M_0]\ \mathsf{in}\ N$ and $M = \kappa'[M_0]$,

By induction hypothesis on $\Lambda;\Delta;\Gamma\vdash_{\omega} M : \bigcirc A$, we have $\Lambda;\Delta;\Gamma\vdash_{\omega} \kappa'[N_0] : \bigcirc A$.

By the rule $\mathsf{\bigcirc E}_W$, $\Lambda;\Delta;\Gamma\vdash_{\omega} \mathsf{letcir}\ v = \kappa'[N_0]\ \mathsf{in}\ N \sim B\ @\ \omega'$, and $\mathsf{letcir}\ v = \kappa'[N_0]\ \mathsf{in}\ N = \kappa[N_0]$.

If $\mathsf{letcir}\ v = M\ \mathsf{in}\ N = \kappa[M_0] = \mathsf{letcir}\ v = \mathsf{cir}\ \kappa'[M_0]\ \mathsf{in}\ N$ and $M = \mathsf{cir}\ \kappa'[M_0]$, We have $\dfrac{\textit{fresh}\ \omega'' \quad \Lambda;\Delta;\Gamma \vdash_\omega \kappa'[M_0] \sim A \ @\ \omega''}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{cir}\ \kappa'[M_0] : \bigcirc A}\ \bigcirc\mathsf{I}_W$.

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[M_0] \sim A \ @\ \omega''$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] \sim A \ @\ \omega''$. Then,

$$\dfrac{\dfrac{\textit{fresh}\ \omega'' \quad \Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] \sim A \ @\ \omega''}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{cir}\ \kappa'[N_0] : \bigcirc A}\ \bigcirc\mathsf{I}_W \quad \Lambda;\Delta, v \sim A;\Gamma \vdash_\omega N \sim B \ @\ \omega'}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{letcir}\ v = \mathsf{cir}\ \kappa'[N_0]\ \mathsf{in}\ N \sim B \ @\ \omega'}\ \bigcirc\mathsf{E}_W$$

and $\mathsf{letcir}\ v = \mathsf{cir}\ \kappa'[N_0]\ \mathsf{in}\ N = \kappa[N_0]$.

If $\mathsf{letcir}\ v = M\ \mathsf{in}\ N = \kappa[M_0] = \mathsf{letcir}\ v = \mathsf{cir}_{\omega''}\ \kappa'[M_0]\ \mathsf{in}\ N$ and $M = \mathsf{cir}_{\omega''}\ \kappa'[M_0]$,

There is no rule for deriving $\Lambda;\Delta;\Gamma \vdash_\omega M : \bigcirc A$.

Case $\bigcirc\mathsf{E}'_W$ is similar to Case $\bigcirc\mathsf{E}_W$.

Case $\dfrac{\Lambda;\Delta;\Gamma \vdash_\omega M : \Box A}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{eval}\ M : \mathsf{unit}}\ \mathsf{Teval}$, $\mathsf{eval}\ M = \kappa[M_0] = \mathsf{eval}\ \kappa'[M_0]$:

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega M : \Box A$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] : \Box A$

By the rule Teval, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{eval}\ \kappa'[N_0] : \mathsf{unit}$, and $\mathsf{eval}\ \kappa'[N_0] = \kappa[N_0]$.

Case Teval@ is similar to Case Teval.

Case $\dfrac{\Lambda;\Delta;\Gamma \vdash_\omega M : \Box\bigcirc A}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{future}\ M \sim A\ \mathsf{sync}\ @\ \omega^*}\ \mathsf{Tfuture}$, $\mathsf{future}\ M = \kappa[M_0] = \mathsf{future}\ \kappa'[M_0]$:

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega M : \Box\bigcirc A$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] : \Box\bigcirc A$.

By the rule Tfuture, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{future}\ \kappa'[N_0] \sim A\ \mathsf{sync}\ @\ \omega^*$, and $\mathsf{future}\ \kappa'[N_0] = \kappa[N_0]$.

Cases Tfuture@, Tfuture$'$, Tfuture@$'$ are similar to Case Tfuture.

Case $\dfrac{\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{sync}}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{syncwith}\ M \sim A\ @\ \omega^*}\ \mathsf{Tswith}$, $\mathsf{syncwith}\ M = \kappa[M_0] = \mathsf{syncwith}\ \kappa'[M_0]$:

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{sync}$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] : A\ \mathsf{sync}$.

By the rule Tswith, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{syncwith}\ \kappa'[N_0] \sim A\ @\ \omega^*$, and $\mathsf{syncwith}\ \kappa'[N_0] = \kappa[N_0]$.

Case Tswith$'$ is similar to Case Tswith.

Case $\dfrac{}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{newchan}_A \sim A\ \mathsf{chan}\ @\ \omega^*}\ \mathsf{Tnewc}$:

There is no κ such that $\mathsf{newchan}_A = \kappa[M_0]$ and $\kappa \neq []$.

Case $\dfrac{\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{chan}}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{readchan}\ M \sim A\ @\ \omega^*}\ \mathsf{Treadc}$, $\mathsf{readchan}\ M = \kappa[M_0] = \mathsf{readchan}\ \kappa'[M_0]$:

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{chan}$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] : A\ \mathsf{chan}$.

By the rule Treadc, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{readchan}\ \kappa'[N_0] \sim A\ @\ \omega^*$, and $\mathsf{readchan}\ \kappa'[N_0] = \kappa[N_0]$.

Case $\dfrac{\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{chan} \qquad \textit{fresh}\ \omega' \quad \Lambda;\Delta;\Gamma \vdash_\omega N \sim A\ @\ \omega'}{\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{writechan}\ M\ N \sim A\ @\ \omega^*}\ \mathsf{Twritec}$:

If $\mathsf{writechan}\ M\ N = \kappa[M_0] = \mathsf{writechan}\ \kappa'[M_0]\ N$ and $M = \kappa'[M_0]$,

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega M : A\ \mathsf{chan}$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] : A\ \mathsf{chan}$.

By the rule Twritec, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{writechan}\ \kappa'[N_0]\ N \sim A\ @\ \omega^*$, and $\mathsf{writechan}\ \kappa'[N_0]\ N = \kappa[N_0]$.

If $\mathsf{writechan}\ M\ N = \kappa[M_0] = \mathsf{writechan}\ M\ \kappa'[M_0]$ and $N = \kappa'[M_0]$ where $M = \mathsf{chanvar}\ \gamma$,

By induction hypothesis on $\Lambda;\Delta;\Gamma \vdash_\omega N \sim A\ @\ \omega'$, we have $\Lambda;\Delta;\Gamma \vdash_\omega \kappa'[N_0] \sim A\ @\ \omega'$.

By the rule Twritec, $\Lambda;\Delta;\Gamma \vdash_\omega \mathsf{writechan}\ M\ \kappa'[N_0] \sim A\ @\ \omega^*$, and $\mathsf{writechan}\ M\ \kappa'[N_0] = \kappa[N_0]$.

□

Lemma C.7.
If C, M **at** $\gamma :: \Lambda, \gamma \sim A\ @\ \omega$ *and* $\Lambda, \gamma \sim A\ @\ \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A\ @\ \omega$,
then C, N **at** $\gamma :: \Lambda, \gamma \sim A\ @\ \omega$.

Proof. C, M **at** $\gamma :: \Lambda, \gamma \sim A\ @\ \omega$ implies that for each M' **at** $\gamma' \in C$,
$\gamma' \sim A'\ @\ \omega' \in \Lambda, \gamma \sim A\ @\ \omega$ and $\Lambda, \gamma \sim A\ @\ \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A'\ @\ \omega'$, or
$\gamma' \sim A'\ @\ \star \in \Lambda, \gamma \sim A\ @\ \omega$ and $\Lambda, \gamma \sim A\ @\ \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A'\ @\ \omega''$ for a fresh node ω''.

By the rule Tcfg and $\Lambda, \gamma \sim A \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A \mathbin{@} \omega$, we have $C, N \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \omega$. □

Lemma C.8.
If $C, M \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \star$ and $\Lambda, \gamma \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A \mathbin{@} \omega$ for a fresh node ω, then $C, N \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \star$.

Proof. $C, M \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \star$ implies that for each $M' \ \mathbf{at}\ \gamma' \in C$,
$\gamma' \sim A' \mathbin{@} \omega' \in \Lambda, \gamma \sim A \mathbin{@} \star$ and $\Lambda, \gamma \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega'$, or
$\gamma' \sim A' \mathbin{@} \star \in \Lambda, \gamma \sim A \mathbin{@} \star$ and $\Lambda, \gamma \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega''$ for a fresh node ω''.

By the rule Tcfg and $\Lambda, \gamma \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A \mathbin{@} \omega$, we have $C, N \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \star$. □

Proof of Lemma 4.4:

Proof. By induction on the structure of κ.
Case $\kappa = []$:
$B = A$ and $\omega'' = \omega'$.
If $\kappa \neq []$, it suffices to consider those cases in which the rule $\mathsf{Prim}{\sim}_W$ is not used to deduce $\Lambda; \Delta; \Gamma \vdash_\omega \kappa[M] \sim A \mathbin{@} \omega'$; if the rule $\mathsf{Prim}{\sim}_W$ is used, we repeat the same case analysis on the premise of the rule.
Case $\kappa = \kappa_0\ M_0$:
By the rule $\supset\!\mathsf{E}_W$ and induction hypothesis on κ_0.
Case $\kappa = \mathsf{letbox}\ x = \kappa_0\ \mathsf{in}\ M_0$:
By the rule $\Box\mathsf{E}_W$ or $\Box\mathsf{E}'_W$ and induction hypothesis on κ_0.
Case $\kappa = \mathsf{letcir}\ v = \kappa_0\ \mathsf{in}\ M_0$:
By the rule $\bigcirc\mathsf{E}_W$ or $\bigcirc\mathsf{E}'_W$ and induction hypothesis on κ_0.
Case $\kappa = \mathsf{letcir}\ v = \mathsf{cir}\ \kappa_0\ \mathsf{in}\ M_0$:
By the rules $\bigcirc\mathsf{E}_W$ and $\bigcirc\mathsf{I}_W$ and induction hypothesis on κ_0.
Case $\kappa = \mathsf{letcir}\ v = \mathsf{cir}_{\omega_0}\ \kappa_0\ \mathsf{in}\ M_0$:
By the rules $\bigcirc\mathsf{E}'_W$ and $\bigcirc\mathsf{I}'_W$ and induction hypothesis on κ_0.
Case $\mathsf{eval}\ \kappa_0$:
By the rule Teval or $\mathsf{Teval@}$ and induction hypothesis on κ_0.
Case $\mathsf{future}\ \kappa_0$:
By the rule $\mathsf{Tfuture}$, $\mathsf{Tfuture@}$, $\mathsf{Tfuture}'$, or $\mathsf{Tfuture@}'$, and induction hypothesis on κ_0.
Case $\mathsf{syncwith}\ \kappa_0$:
By the rule Tswith or Tswith' and induction hypothesis on κ_0.
Case $\mathsf{readchan}\ \kappa_0$:
By the rule Treadc and induction hypothesis on κ_0.
Case $\mathsf{writechan}\ \kappa_0\ M_0$:
By the rule $\mathsf{Twritec}$ and induction hypothesis on κ_0.
Case $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ \kappa_0$:
By the rule $\mathsf{Twritec}$ and induction hypothesis on κ_0. □

Proposition C.9 (Weakening).
Suppose
$C :: \Lambda$,
$\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M : A$,
$\omega = \mathcal{P}(\gamma)$, *where γ is not found in Λ.*
Then $C, M \ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A \mathbin{@} \omega$.

Proof.
If M' **at** $\gamma' \in C$ and $\gamma' \sim A' \mathbin{@} \omega' \in \Lambda$,
By the rule Tcfg, $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega'$
By weakening on Λ, we have $\Lambda, \gamma \sim A \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega'$
If M' **at** $\gamma' \in C$ and $\gamma' \sim A' \mathbin{@} \star \in \Lambda$,
By the rule Tcfg, $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega'$ for a fresh node ω'.
By weakening on Λ, we have $\Lambda, \gamma \sim A \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} M' \sim A' \mathbin{@} \omega'$
For M **at** γ,
By weakening $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M : A$, we have $\Lambda, \gamma \sim A \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M : A$.
That is, $\Lambda, \gamma \sim A \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} M \sim A \mathbin{@} \omega$.
Therefore C, M **at** $\gamma :: \Lambda, \gamma \sim A \mathbin{@} \omega$ by the rule Tcfg. ☐

Lemma C.10.
If
C, M **at** $\gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N' \sim A_{\gamma'} \mathbin{@} \omega^*$ *for an arbitrary node* ω^*,
then
C, N **at** γ, N' **at** $\gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star$.

Proof. From C, M **at** $\gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
for each M_0 **at** $\gamma_0 \in C$,
$\gamma_0 \sim A_0 \mathbin{@} \omega_0 \in \Lambda$ and $\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma_0)} M_0 \sim A_0 \mathbin{@} \omega_0$, or
$\gamma_0 \sim A_0 \mathbin{@} \star \in \Lambda$ and $\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma_0)} M_0 \sim A_0 \mathbin{@} \omega_0$ for an arbitrary node ω_0.
By weakening on $\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma_0)} M_0 \sim A_0 \mathbin{@} \omega_0$, or
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma_0)} M_0 \sim A_0 \mathbin{@} \omega_0$ for an arbitrary node ω_0.
By the rule Tcfg, we have C, N **at** γ, N' **at** $\gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star$. ☐

Lemma C.11.
If
C, M **at** $\gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N' \sim A_{\gamma'} \mathbin{@} \omega'$,
then
C, N **at** γ, N' **at** $\gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'$.

Lemma C.12.
If
C, M **at** $\gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \star$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \star, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A_\gamma \mathbin{@} \omega^*$ *for an arbitrary node* ω^*,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \star, \gamma' \sim A_{\gamma'} \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N' \sim A_{\gamma'} \mathbin{@} \omega^*$ *for an arbitrary node* ω^*,
then
C, N **at** γ, N' **at** $\gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \star, \gamma' \sim A_{\gamma'} \mathbin{@} \star$.

Lemma C.13.
If
C, M **at** $\gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \star$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \star, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} N \sim A_\gamma \mathbin{@} \omega^*$ *for an arbitrary node* ω^*,

$\Lambda, \gamma \sim A_\gamma \,@\, \star, \gamma' \sim A_{\gamma'} \,@\, \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N' \sim A_{\gamma'} \,@\, \omega'$,

then

$C, N \;\mathbf{at}\; \gamma, N' \;\mathbf{at}\; \gamma' :: \Lambda, \gamma \sim A_\gamma \,@\, \star, \gamma' \sim A_{\gamma'} \,@\, \omega'$.

Proof. Similar to the proof of Lemma C.10. □

Proof of Theorem 4.1:

Proof. By case analysis of $C \Longrightarrow C'$. (Below we reuse all metavariables.)

Case $\dfrac{M \longrightarrow N}{C, \kappa[M] \;\mathbf{at}\; \gamma \Longrightarrow C, \kappa[N] \;\mathbf{at}\; \gamma}\ \mathsf{Rcfg}$:

If $C, \kappa[M] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$, then $\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[M] \sim A_\gamma \,@\, \omega$.

Since $M \longrightarrow N$, Lemmas C.5 and C.6 imply $\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[N] \sim A_\gamma \,@\, \omega$.

By Lemma C.7, we have $C, \kappa[N] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$.

If $C, \kappa[M] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \star$, then $\Lambda, \gamma \sim A_\gamma \,@\, \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[M] \sim A_\gamma \,@\, \omega$ for a fresh node ω.

Since $M \longrightarrow N$, Lemmas C.5 and C.6 imply $\Lambda, \gamma \sim A_\gamma \,@\, \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[N] \sim A_\gamma \,@\, \omega$.

By Lemma C.8, we have $C, \kappa[N] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \star$.

Case $\dfrac{\mathit{new}\ \gamma'}{C, \kappa[\mathsf{eval\ box}\ M] \;\mathbf{at}\; \gamma \Longrightarrow C, \kappa[()] \;\mathbf{at}\; \gamma, M \;\mathbf{at}\; \gamma'}\ \mathsf{Reval}$:

If $C, \kappa[\mathsf{eval\ box}\ M] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$, then $\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{eval\ box}\ M] \sim A_\gamma \,@\, \omega$.

By Lemma 4.4, eval box M typechecks:

$$\dfrac{\dfrac{\mathit{fresh}\ \omega' \quad \Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} M : A}{\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{box}\ M : \Box A}\ \Box\mathsf{I}_W}{\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{eval\ box}\ M : \mathsf{unit}}\ \mathsf{Teval}$$

Since $\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} () : \mathsf{unit}$,

$\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[()] \sim A_\gamma \,@\, \omega$ by Lemma C.6.

By Lemma C.7,

$C, \kappa[()] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$.

From

$C, \kappa[()] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$,

$\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} M : A$ where we let $\omega' = \mathcal{P}(\gamma')$,

we have $C, \kappa[()] \;\mathbf{at}\; \gamma, M \;\mathbf{at}\; \gamma' :: \Lambda, \gamma \sim A_\gamma \,@\, \omega, \gamma' \sim A \,@\, \omega'$ by Proposition C.9.

The case for $C, \kappa[\mathsf{eval\ box}\ M] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \star$ is similar, except that we use Lemma C.8 instead of Lemma C.7.

Case $\dfrac{\mathit{new}\ \gamma' \,@\, \omega'}{C, \kappa[\mathsf{eval\ box}_{\omega'}\ M] \;\mathbf{at}\; \gamma \Longrightarrow C, \kappa[()] \;\mathbf{at}\; \gamma, M \;\mathbf{at}\; \gamma'}\ \mathsf{Reval@}$:

The proof is similar to Case Reval, except that we use ω' without creating a fresh node.

Case $\dfrac{\mathit{new}\ \gamma'}{C, \kappa[\mathsf{future\ box}\ M] \;\mathbf{at}\; \gamma \Longrightarrow C, \kappa[\mathsf{syncvar}\ \gamma'] \;\mathbf{at}\; \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v \;\mathbf{at}\; \gamma'}\ \mathsf{Rfuture}$:

If $C, \kappa[\mathsf{future\ box}\ M] \;\mathbf{at}\; \gamma :: \Lambda, \gamma \sim A_\gamma \,@\, \omega$, then $\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{future\ box}\ M] \sim A_\gamma \,@\, \omega$.

By Lemma 4.4, future box M typechecks:

$$\dfrac{\dfrac{\mathit{fresh}\ \omega' \quad \Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} M : \bigcirc A}{\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{box}\ M : \Box\bigcirc A}\ \Box\mathsf{I}_W}{\Lambda, \gamma \sim A_\gamma \,@\, \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{future\ box}\ M \sim A\ \mathsf{sync} \,@\, \omega^*}\ \mathsf{Tfuture}$$

or

$$\dfrac{\dfrac{\textit{fresh}\ \omega' \quad \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} M : \bigcirc_{\omega''} A}{\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{box}\ M : \Box \bigcirc_{\omega''} A}\ \Box\mathsf{I}_W}{\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{future\ box}\ M \sim A\ \mathsf{sync}_{\omega''} \mathbin{@} \omega^*}\ \mathsf{Tfuture}$$

In the first case,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{future\ box}\ M] \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{future\ box}\ M \sim A\ \mathsf{sync} \mathbin{@} \omega^*$ for an arbitrary node ω^*,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} M : \bigcirc A$ for a fresh node ω',
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} \mathsf{letcir}\ v = M\ \mathsf{in}\ v \sim A \mathbin{@} \omega^*$ for an arbitrary node ω^*,
and we let $\omega' = \mathcal{P}(\gamma')$.
By weakening on $\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{future\ box}\ M] \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{future\ box}\ M \sim A\ \mathsf{sync} \mathbin{@} \omega^*$ for an arbitrary node ω^*,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} \mathsf{letcir}\ v = M\ \mathsf{in}\ v \sim A \mathbin{@} \omega^*$ for an arbitrary node ω^*.
By the rules Tsvar and Val_W,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{syncvar}\ \gamma' \sim A\ \mathsf{sync} \mathbin{@} \omega^*$ for an arbitrary node ω^*,
By Lemma C.6,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{syncvar}\ \gamma'] \sim A_\gamma \mathbin{@} \omega$,
By applying Lemma C.10 to
$C, \kappa[\mathsf{future\ box}\ M]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{syncvar}\ \gamma'] \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} \mathsf{letcir}\ v = M\ \mathsf{in}\ v \sim A \mathbin{@} \omega^*$ for an arbitrary node ω^*,
we have $C, \kappa[\mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \star$.

In the second case, we prove $C, \kappa[\mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A \mathbin{@} \omega''$; the proof is similar to the first case, except that we use Lemma C.11.

The case for $C, \kappa[\mathsf{future\ box}\ M]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \star$ is similar, except that we use Lemmas C.12 and C.13.

Case $\dfrac{\textit{new}\ \gamma' \mathbin{@} \omega'}{C, \kappa[\mathsf{future\ box}_{\omega'}\ M]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[\mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma, \mathsf{letcir}\ v = M\ \mathsf{in}\ v\ \mathbf{at}\ \gamma'}\ \mathsf{Rfuture@}$:

The proof is similar to Case Rfuture, except that we use ω' without creating a fresh node.

Case $\dfrac{}{C, \kappa[\mathsf{syncwith\ syncvar}\ \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' \Longrightarrow C, \kappa[V]\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma'}\ \mathsf{Rswith}$:

If $C, \kappa[\mathsf{syncwith\ syncvar}\ \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'$, then
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{syncwith\ syncvar}\ \gamma'] \sim A_\gamma \mathbin{@} \omega$,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V \sim A_{\gamma'} \mathbin{@} \omega'$.
By Lemma 4.4 and the rules Tsvar' and Tswith',
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{syncwith\ syncvar}\ \gamma' \sim A_{\gamma'} \mathbin{@} \omega'$.
If $\mathcal{P}(\gamma') = \omega'$ (whether $\mathcal{P}(\gamma) = \mathcal{P}(\gamma')$ or not),
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} V \sim A_{\gamma'} \mathbin{@} \omega'$ by the rule Val_W.
If $\mathcal{P}(\gamma') \neq \omega'$
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} V : A_{\gamma'}$ by the rule Val_W, and
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} V \sim A_{\gamma'} \mathbin{@} \omega'$ by the rule Val_W.
By Lemma C.6,
$\Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[V] \sim A_\gamma \mathbin{@} \omega$,
By Lemma C.7,
$C, \kappa[V]\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \omega'$.
The case for $C, \kappa[\mathsf{syncwith\ syncvar}\ \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \mathbin{@} \omega, \gamma' \sim A_{\gamma'} \mathbin{@} \star$ is similar.

The cases for

$C, \kappa[\textsf{syncwith syncvar } \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \star, \gamma' \sim A_{\gamma'} \ @\ \omega'$ and

$C, \kappa[\textsf{syncwith syncvar } \gamma']\ \mathbf{at}\ \gamma, V\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \star, \gamma' \sim A_{\gamma'} \ @\ \star$

are also similar, except that we use Lemma C.8 instead of Lemma C.7.

Case $\dfrac{\textit{new } \gamma'}{C, \kappa[\textsf{newchan}_A]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[\textsf{chanvar } \gamma']\ \mathbf{at}\ \gamma, \textsf{nil}\ \mathbf{at}\ \gamma'}\ \textsf{Rnewc}$:

If $C, \kappa[\textsf{newchan}_A]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma \ @\ \omega$, then

$\Lambda, \gamma \sim A_\gamma \ @\ \omega; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\textsf{newchan}_A] \sim A_\gamma \ @\ \omega$.

By weakening on $\Lambda, \gamma \sim A_\gamma \ @\ \omega$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\textsf{newchan}_A] \sim A_\gamma \ @\ \omega$.

By Lemma 4.4, $\textsf{newchan}_A$ typechecks:

$$\dfrac{}{\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \textsf{newchan}_A \sim A \textsf{ chan} \ @\ \omega^*}\ \textsf{Tnewc}$$

By the rules $\textsf{Tchanv}$ and $\textsf{Val}_W$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \textsf{chanvar } \gamma' \sim A \textsf{ chan} \ @\ \omega^*$

By Lemma C.6,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\textsf{chanvar } \gamma'] \sim A_\gamma \ @\ \omega$.

By the rule $\textsf{Tvnil}$ and $\textsf{Val}_W$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma')} \textsf{nil} \sim A \textsf{ vlist} \ @\ \omega^*$ for an arbitrary node ω^*.

By applying Lemma C.10 to

$C, \kappa[\textsf{newchan}_A]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma \ @\ \omega$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\textsf{chanvar } \gamma'] \sim A_\gamma \ @\ \omega$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma')} \textsf{nil} \sim A \textsf{ vlist} \ @\ \omega^*$ for an arbitrary node ω^*,

we have

$C, \kappa[\textsf{chanvar } \gamma']\ \mathbf{at}\ \gamma, \textsf{nil}\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A \textsf{ vlist} \ @\ \star$.

The case for $C, \kappa[\textsf{newchan}_A]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma \ @\ \star$ is similar, except that we use Lemma C.12.

Case $\dfrac{}{C, \kappa[\textsf{readchan chanvar } \gamma']\ \mathbf{at}\ \gamma, V_h :: V_t\ \mathbf{at}\ \gamma' \Longrightarrow C, \kappa[V_h]\ \mathbf{at}\ \gamma, V_t\ \mathbf{at}\ \gamma'}\ \textsf{Readc}$:

If $C, \kappa[\textsf{readchan chanvar } \gamma']\ \mathbf{at}\ \gamma, V_h :: V_t\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star$, then

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\textsf{readchan chanvar } \gamma'] \sim A_\gamma \ @\ \omega$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_h :: V_t \sim A_{\gamma'} \ @\ \omega^*$ for an arbitrary node ω^*.

By the rules $\textsf{Val}_W$ and $\textsf{Tvcon}$,

$A_{\gamma'} = A \textsf{ vlist}$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} V_h \sim A \ @\ \omega^*$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_t \sim A_{\gamma'} \ @\ \omega^*$.

By Lemma 4.4 and the rules $\textsf{Tchanv}$ and $\textsf{Treadc}$,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \textsf{readchan chanvar } \gamma' \sim A \ @\ \omega^*$,

By Lemma C.6,

$\Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[V_h] \sim A_\gamma \ @\ \omega$.

By Lemma C.7,

$C, \kappa[V_h]\ \mathbf{at}\ \gamma, V_h :: V_t\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star$.

By Lemma C.8,

$C, \kappa[V_h]\ \mathbf{at}\ \gamma, V_t\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \omega, \gamma' \sim A_{\gamma'} \ @\ \star$.

The case for $C, \kappa[\textsf{readchan chanvar } \gamma']\ \mathbf{at}\ \gamma, V_h :: V_t\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma \ @\ \star, \gamma' \sim A_{\gamma'} \ @\ \star$ is similar, except that we use Lemma C.8 instead of Lemma C.7.

The two cases with $\gamma' \sim A_{\gamma'} @ \omega'$ for some node ω' are impossible because of the rule Tchanv.

Case $\dfrac{}{\begin{array}{l} C, \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' \Longrightarrow \\ \qquad C, \kappa[V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil}\ \mathbf{at}\ \gamma' \end{array}}\ \mathsf{Rwritec}$:

If $C, \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star$, then

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V] \sim A_\gamma @ \omega$,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_1 :: \cdots :: V_n :: \mathsf{nil} \sim A_{\gamma'} @ \omega^*$ for an arbitrary node ω^*.

By the rules Val_W and Tvcon,

$A_{\gamma'} = A\ \mathsf{vlist}$,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_1 \sim A @ \omega^*$,

$\cdots$,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_n \sim A @ \omega^*$.

By Lemma 4.4 and the rules Tchanv, Twritec, and Val_W,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V \sim A @ \omega^*$,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} V \sim A @ \omega^*$,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V \sim A @ \omega^*$.

By Lemma C.6,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[V] \sim A_\gamma @ \omega$.

By Lemma C.7,

$C, \kappa[V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star$.

By the rules Val_W, Tvcon, Tvnil,

$\Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} V_1 :: \cdots :: V_n :: V :: \mathsf{nil} \sim A_{\gamma'} @ \omega^*$.

By Lemma C.8,

$C, \kappa[V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil}\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma @ \omega, \gamma' \sim A_{\gamma'} @ \star$.

The case for $C, \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil}\ \mathbf{at}\ \gamma' :: \Lambda, \gamma \sim A_\gamma @ \star, \gamma' \sim A_{\gamma'} @ \star$ is similar, except that we use Lemma C.8 instead of Lemma C.7.

The two cases with $\gamma' \sim A_{\gamma'} @ \omega'$ for some node ω' are impossible because of the rule Tchanv.

Case $\dfrac{v \sim A @ \omega \in \Gamma^{\mathsf{perm}} \quad v \rightarrow_{\mathsf{perm}} V \quad \mathcal{P}(\gamma) = \omega}{C, \kappa[v]\ \mathbf{at}\ \gamma \Longrightarrow C, \kappa[V]\ \mathbf{at}\ \gamma}\ \mathsf{Rvalvar}$:

If $C, \kappa[v]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma @ \omega'$, then

$\Lambda, \gamma \sim A_\gamma @ \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[v] \sim A_\gamma @ \omega'$.

Since $v \sim A @ \omega \in \Gamma^{\mathsf{perm}}$ and $\mathcal{P}(\gamma) = \omega$,

$\Lambda, \gamma \sim A_\gamma @ \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} v : A$.

By the assumption on V and weakening,

$\Lambda, \gamma \sim A_\gamma @ \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} V : A$.

Since $\mathcal{P}(\gamma) = \omega$,

$\Lambda, \gamma \sim A_\gamma @ \omega'; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma)} \kappa[V] \sim A_\gamma @ \omega'$ by Lemma C.6.

By Lemma C.7,

$C, \kappa[V]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma @ \omega'$.

The case for $C, \kappa[V]\ \mathbf{at}\ \gamma :: \Lambda, \gamma \sim A_\gamma @ \star$ is similar, except that we use Lemma C.8 instead of Lemma C.7.

□

Proof of Lemma 4.3:

Proof. By induction on the structure of $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M \sim A @ \omega'$. (Below we reuse all metavariables.)

Case Cvar_W:

impossible.

Case Vvar_W:

$M = v$, $\omega = \omega'$, and $v \sim A \mathbin{@} \omega' \in \Gamma^{\mathsf{perm}}$.

Cases $\supset\!\mathsf{I}_W$, $\Box\mathsf{I}_W$, $\Box\mathsf{I}'_W$, $\bigcirc\mathsf{I}_W$, $\bigcirc\mathsf{I}'_W$, $\mathsf{T}()$, Tsvar, Tsvar', Tvnil, Tvcon, Tchanv:

$M = V \neq v$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega'} V : A}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} V \sim A \mathbin{@} \omega'}\ \mathsf{Val}_W\ (\omega \neq \omega')$:

If $V = v$, then $v \sim A \mathbin{@} \omega' \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M : A \supset B \quad \Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} N : A}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M\ N : B}\ \supset\!\mathsf{E}_W$:

If $M = V \neq v$,

$M = \lambda x\!:\!A.\, M'$ by Lemma 4.2.

$M\ N = ([])[(\lambda x\!:\!A.\, M')\ N]$ and $(\lambda x\!:\!A.\, M')\ N \longrightarrow [N/x]M'$.

If $M = v$,

$v \sim A \supset B \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

$M\ N = ([]\ N)[v]$ and $v \sim A \supset B \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $M\ N = (\kappa\ N)[M']$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M : \Box A \quad \Lambda; \cdot, x :: A; \Gamma^{\mathsf{perm}} \vdash_{\omega} N \sim B \mathbin{@} \omega'}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} \mathsf{letbox}\ x = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \Box\mathsf{E}_W$:

If $M = V \neq v$,

$M = \mathsf{box}\ M'$ by Lemma 4.2.

$\mathsf{letbox}\ x = M\ \mathsf{in}\ N = ([])[\mathsf{letbox}\ x = \mathsf{box}\ M'\ \mathsf{in}\ N]$ and $\mathsf{letbox}\ x = \mathsf{box}\ M'\ \mathsf{in}\ N \longrightarrow [M'/x]N$.

If $M = v$,

$v \sim \Box A \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

$\mathsf{letbox}\ x = M\ \mathsf{in}\ N = (\mathsf{letbox}\ x = []\ \mathsf{in}\ N)[v]$ and $v \sim \Box A \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' \mathbin{@} \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $\mathsf{letbox}\ x = M\ \mathsf{in}\ N = (\mathsf{letbox}\ x = \kappa\ \mathsf{in}\ N)[M']$.

Case $\Box\mathsf{E}'_W$ is similar to Case $\Box\mathsf{E}_W$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M : \bigcirc A \quad \Lambda; \cdot, v \sim A; \Gamma^{\mathsf{perm}} \vdash_{\omega} N \sim B \mathbin{@} \omega'}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} \mathsf{letcir}\ v = M\ \mathsf{in}\ N \sim B \mathbin{@} \omega'}\ \bigcirc\mathsf{E}_W$:

If $M = V \neq v'$,

$M = \mathsf{cir}\ M'$ by Lemma 4.2 and

$$\dfrac{\textit{fresh}\ \omega^* \quad \Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} M' \sim A \mathbin{@} \omega^*}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\omega} \mathsf{cir}\ M' : \bigcirc A}\ \bigcirc\mathsf{I}_W.$$

1) If $M' = V' \neq v''$,

$\mathsf{letcir}\ v = M\ \mathsf{in}\ N = ([])[\mathsf{letcir}\ v = \mathsf{cir}\ V'\ \mathsf{in}\ N]$ and $\mathsf{letcir}\ v = \mathsf{cir}\ V'\ \mathsf{in}\ N \longrightarrow [V'/v]N$.

2) $M' = v''$ is impossible.

3) If $M' \neq V'$,

$M' = \kappa[M'']$ by induction hypothesis where

$M'' = v''$ and $v'' \sim A' @ \omega \in \Gamma^{\mathsf{perm}}$,

$M'' \longrightarrow N'$, or

M'' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V''$.

Then we let $\mathsf{letcir}\ v = M\ \mathsf{in}\ N = (\mathsf{letcir}\ v = \mathsf{cir}\ \kappa\ \mathsf{in}\ N)[M'']$.

If $M = v'$,

$v' \sim \bigcirc A @ \omega \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

$\mathsf{letcir}\ v = M\ \mathsf{in}\ N = (\mathsf{letcir}\ v = []\ \mathsf{in}\ N)[v']$ and $v' \sim \bigcirc A @ \omega \in \Gamma^{\mathsf{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v'$ and $v' \sim A' @ \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $\mathsf{letcir}\ v = M\ \mathsf{in}\ N = (\mathsf{letcir}\ v = \kappa\ \mathsf{in}\ N)[M']$.

Case $\bigcirc\mathsf{E}'_W$ is similar to Case $\bigcirc\mathsf{E}_W$, except that Subcases 1) and 2) are now combined as follows:

If $M' = V'$,

$\mathsf{letcir}\ v = M\ \mathsf{in}\ N = ([])[\mathsf{letcir}\ v = \mathsf{cir}_{\omega^*}\ V'\ \mathsf{in}\ N]$ and $\mathsf{letcir}\ v = \mathsf{cir}_{\omega^*}\ V'\ \mathsf{in}\ N \longrightarrow [V'/v]N$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M : A_{prim}}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A_{prim} @ \omega'}$ $\mathsf{Prim}{\sim}_W\ (\omega \neq \omega')$:

If $M = V \neq v$ by induction hypothesis, we are done.

$M = v$ and $v \sim A_{prim} @ \omega \in \Gamma^{\mathsf{perm}}$ cannot happen by the assumption on Γ^{perm}.

If $M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' @ \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$,

then we are done.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M : \Box A}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{eval}\ M : \mathsf{unit}}$ Teval :

If $M = V \neq v$,

$M = \mathsf{box}\ M'$ by Lemma 4.2.

$\mathsf{eval}\ M = ([])[\mathsf{eval\ box}\ M']$.

If $M = v$,

$v \sim \Box A @ \omega \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

$\mathsf{eval}\ M = (\mathsf{eval}\ [])[v]$ and $v \sim \Box A @ \omega \in \Gamma^{\mathsf{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' @ \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $\text{eval } M = (\text{eval } \kappa)[M']$.

Case Teval@ is similar to Case Teval.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega M : \Box\bigcirc A}{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega \text{future } M \sim A \text{ sync} @ \omega^*}$ Tfuture :

If $M = V \neq v$,

$M = \text{box } M'$ by Lemma 4.2.

$\text{future } M = ([])[\text{future box } M']$.

If $M = v$,

$v \sim \Box\bigcirc A @ \omega \in \Gamma^{\text{perm}}$ by the rule Vvar_W.

$\text{future } M = (\text{future } [])[v]$ and $v \sim \Box\bigcirc A @ \omega \in \Gamma^{\text{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' @ \omega \in \Gamma^{\text{perm}}$,

$M' \longrightarrow N'$, or

M' is $\text{eval box } N'$, $\text{eval box}_{\omega''} N'$, $\text{future box } N'$, $\text{future box}_{\omega''} N'$, $\text{syncwith syncvar } \gamma$, $\text{newchan}_{B'}$, $\text{readchan chanvar } \gamma$, or $\text{writechan } (\text{chanvar } \gamma)\ V'$.

Then we let $\text{future } M = (\text{future } \kappa)[M']$.

Cases Tfuture@, Tfuture′, and Tfuture@′ are similar to Case Tfuture.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega M : A \text{ sync}}{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega \text{syncwith } M \sim A @ \omega^*}$ Tswith :

If $M = V \neq v$,

$M = \text{syncvar } \gamma$ by Lemma 4.2.

$\text{syncwith } M = ([])[\text{syncwith syncvar } \gamma]$.

If $M = v$,

$v \sim A \text{ sync} @ \omega \in \Gamma^{\text{perm}}$ by the rule Vvar_W.

$\text{syncwith } M = (\text{syncwith } [])[v]$ and $v \sim A \text{ sync} @ \omega \in \Gamma^{\text{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' @ \omega \in \Gamma^{\text{perm}}$,

$M' \longrightarrow N'$, or

M' is $\text{eval box } N'$, $\text{eval box}_{\omega''} N'$, $\text{future box } N'$, $\text{future box}_{\omega''} N'$, $\text{syncwith syncvar } \gamma$, $\text{newchan}_{B'}$, $\text{readchan chanvar } \gamma$, or $\text{writechan } (\text{chanvar } \gamma)\ V'$.

Then we let $\text{syncwith } M = (\text{syncwith } \kappa)[M']$.

Case Tswith′ is similar to Case Tswith.

Case $\dfrac{}{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega \text{newchan}_A \sim A \text{ chan} @ \omega^*}$ Tnewc :

$\text{newchan}_A = ([])[\text{newchan}_A]$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega M : A \text{ chan}}{\Lambda; \cdot; \Gamma^{\text{perm}} \vdash_\omega \text{readchan } M \sim A @ \omega^*}$ Treadc :

If $M = V \neq v$,

$M = \text{chanvar } \gamma$ by Lemma 4.2.

$\text{readchan } M = ([])[\text{readchan chanvar } \gamma]$.

If $M = v$,

$v \sim A \text{ chan} @ \omega \in \Gamma^{\text{perm}}$ by the rule Vvar_W.

$\text{readchan } M = (\text{chanvar } [])[v]$ and $v \sim A \text{ chan} @ \omega \in \Gamma^{\text{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A' @ \omega \in \Gamma^{\text{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $\mathsf{readchan}\ M = (\mathsf{readchan}\ \kappa)[M']$.

Case $\dfrac{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M : A\ \mathsf{chan} \qquad \mathit{fresh}\ \omega' \quad \Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega N \sim A\ @\ \omega'}{\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{writechan}\ M\ N \sim A\ @\ \omega^*}\ \mathsf{Twritec}$:

If $M = V \neq v$,

$M = \mathsf{chanvar}\ \gamma$ by Lemma 4.2.

1) If $N = V' \neq v'$,

$\mathsf{writechan}\ M\ N = ([])[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V']$.

2) If $N = v'$ is impossible.

3) If $N \neq V'$,

$N = \kappa[N']$ by induction hypothesis where

$N' = v'$ and $v' \sim A'\ @\ \omega \in \Gamma^{\mathsf{perm}}$,

$N' \longrightarrow N''$, or

N' is $\mathsf{eval\ box}\ N''$, $\mathsf{eval\ box}_{\omega''}\ N''$, $\mathsf{future\ box}\ N''$, $\mathsf{future\ box}_{\omega''}\ N''$, $\mathsf{syncwith\ syncvar}\ \gamma'$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma'$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V''$.

Then we let $\mathsf{writechan}\ M\ N = (\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ \kappa)[N']$.

If $M = v$,

$v \sim A\ \mathsf{chan}\ @\ \omega \in \Gamma^{\mathsf{perm}}$ by the rule Vvar_W.

$\mathsf{writechan}\ M\ N = (\mathsf{writechan}\ []\ N)[v]$ and $v \sim A\ \mathsf{chan}\ @\ \omega \in \Gamma^{\mathsf{perm}}$.

If $M \neq V$,

$M = \kappa[M']$ by induction hypothesis where

$M' = v$ and $v \sim A'\ @\ \omega \in \Gamma^{\mathsf{perm}}$,

$M' \longrightarrow N'$, or

M' is $\mathsf{eval\ box}\ N'$, $\mathsf{eval\ box}_{\omega''}\ N'$, $\mathsf{future\ box}\ N'$, $\mathsf{future\ box}_{\omega''}\ N'$, $\mathsf{syncwith\ syncvar}\ \gamma$, $\mathsf{newchan}_{B'}$, $\mathsf{readchan\ chanvar}\ \gamma$, or $\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma)\ V'$.

Then we let $\mathsf{writechan}\ M\ N = (\mathsf{writechan}\ \kappa\ N)[M']$. □

Proof of Theorem 4.5:

Proof.
Suppose $C = C_0, M\ \mathbf{at}\ \gamma$. By the rule Tcfg, we have $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega M \sim A\ @\ \omega'$ for $\mathcal{P}(\gamma) = \omega$ and a certain node ω'. By Lemma 4.3, we consider the following cases:

- $M = V \neq v$.
- $M = v$ (where $v \sim A\ @\ \omega' \in \Gamma^{\mathsf{perm}}$)
- $M = \kappa[v]$, $v \sim B\ @\ \omega \in \Gamma^{\mathsf{perm}}$, and

$$\frac{v \sim B\ @\ \omega \in \Gamma^{\mathsf{perm}} \quad v \rightarrow_{\mathsf{perm}} V \quad \mathcal{P}(\gamma) = \omega}{C_0, \kappa[v]\ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[V]\ \mathbf{at}\ \gamma}\ \mathsf{Rvalvar}.$$

- $M = \kappa[N]$ where $N \longrightarrow N'$, and

$$\frac{N \longrightarrow N'}{C_0, \kappa[N]\ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[N']\ \mathbf{at}\ \gamma}\ \mathsf{Rcfg}.$$

- $M = \kappa[\textsf{eval box } N]$ and

$$\frac{\mathit{new}\ \gamma'}{C_0, \kappa[\textsf{eval box } N] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[()] \ \mathbf{at}\ \gamma, N \ \mathbf{at}\ \gamma'} \ \textsf{Reval}.$$

- $M = \kappa[\textsf{eval box}_{\omega''}\ N]$ and

$$\frac{\mathit{new}\ \gamma' \ @\ \omega''}{C_0, \kappa[\textsf{eval box}_{\omega''}\ N] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[()] \ \mathbf{at}\ \gamma, N \ \mathbf{at}\ \gamma'} \ \textsf{Reval@}.$$

- $M = \kappa[\textsf{future box } N]$ and

$$\frac{\mathit{new}\ \gamma'}{C_0, \kappa[\textsf{future box } N] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[\textsf{syncvar } \gamma'] \ \mathbf{at}\ \gamma, \textsf{letcir } v = N \textsf{ in } v \ \mathbf{at}\ \gamma'} \ \textsf{Rfuture}.$$

- $M = \kappa[\textsf{future box}_{\omega''}\ N]$ and

$$\frac{\mathit{new}\ \gamma' \ @\ \omega''}{C_0, \kappa[\textsf{future box}_{\omega''}\ N] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[\textsf{syncvar } \gamma'] \ \mathbf{at}\ \gamma, \textsf{letcir } v = N \textsf{ in } v \ \mathbf{at}\ \gamma'} \ \textsf{Rfuture@}.$$

- $M = \kappa[\textsf{syncwith syncvar } \gamma']$ and $V \ \mathbf{at}\ \gamma' \notin C_0$. (*e.g.*, $M \ \mathbf{at}\ \gamma' \in C_0$ and M is not a value.)
- $M = \kappa[\textsf{syncwith syncvar } \gamma']$, $V \ \mathbf{at}\ \gamma' \in C_0$, and

$$\frac{}{C_0, \kappa[\textsf{syncwith syncvar } \gamma'] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[V] \ \mathbf{at}\ \gamma} \ \textsf{Rswith}.$$

- $M = \kappa[\textsf{newchan}_B]$ and

$$\frac{\mathit{new}\ \gamma'}{C_0, \kappa[\textsf{newchan}_B] \ \mathbf{at}\ \gamma \Longrightarrow C_0, \kappa[\textsf{chanvar } \gamma'] \ \mathbf{at}\ \gamma, \textsf{nil} \ \mathbf{at}\ \gamma'} \ \textsf{Rnewc}.$$

- $M = \kappa[\textsf{readchan chanvar } \gamma']$.

  By Lemma 4.4,

  $\Lambda; \cdot; \Gamma^{\textsf{perm}} \vdash_\omega \textsf{readchan chanvar } \gamma' \sim B \ @\ \omega''$.

  By the rule Treadc (optionally preceded by the rule $\textsf{Prim}{\sim}_W$ if B is a primitive type),

  $\Lambda; \cdot; \Gamma^{\textsf{perm}} \vdash_\omega \textsf{chanvar } \gamma' : B \textsf{ chan}$.

  By the rule Tchanv,

  $\gamma' \sim B \textsf{ vlist} \ @ \star \in \Lambda$.

  Since $C :: \Lambda$,

  $C = C_0', M \ \mathbf{at}\ \gamma, N \ \mathbf{at}\ \gamma'$ and $\Lambda; \cdot; \Gamma^{\textsf{perm}} \vdash_{\mathcal{P}(\gamma')} N \sim B \textsf{ vlist} \ @\ \omega^*$ for a fresh node ω^*.

  - $N = V_h :: V_t$ and

  $$\frac{}{C_0', \kappa[\textsf{readchan chanvar } \gamma'] \ \mathbf{at}\ \gamma, V_h :: V_t \ \mathbf{at}\ \gamma' \Longrightarrow C_0', \kappa[V_h] \ \mathbf{at}\ \gamma, V_t \ \mathbf{at}\ \gamma'} \ \textsf{Rreadc}.$$

  - $N \neq V_h :: V_t$.

- $M = \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]$.

  By Lemma 4.4,

  $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V \sim B \ @\ \omega''$.

  By the rule Twritec (optionally preceded by the rule Prim$\sim_W$ if B is a primitive type),

  $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_\omega \mathsf{chanvar}\ \gamma' : B\ \mathsf{chan}$.

  By the rule Tchanv,

  $\gamma' \sim B\ \mathsf{vlist}\ @ \star \in \Lambda$.

  Since $C :: \Lambda$,

  $C = C_0', M\ \mathbf{at}\ \gamma, N\ \mathbf{at}\ \gamma'$ and $\Lambda; \cdot; \Gamma^{\mathsf{perm}} \vdash_{\mathcal{P}(\gamma')} N \sim B\ \mathsf{vlist}\ @\ \omega^*$ for a fresh node ω^*.

  - $N = V_1 :: \cdots :: V_n :: \mathsf{nil}$ and

$$\frac{}{\begin{array}{r} C_0', \kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' \Longrightarrow \\ C_0', \kappa[V]\ \mathbf{at}\ \gamma, V_1 :: \cdots :: V_n :: V :: \mathsf{nil}\ \mathbf{at}\ \gamma' \end{array}}\ \mathsf{Rwritec}$$

  - $N \neq V_1 :: \cdots :: V_n :: \mathsf{nil}$.

Therefore, if there exists no C' such that $C \Longrightarrow C'$, C consists only of the following:
$V\ \mathbf{at}\ \gamma$,
$\kappa[\mathsf{syncwith}\ \mathsf{syncvar}\ \gamma']\ \mathbf{at}\ \gamma$ (where $V\ \mathbf{at}\ \gamma' \notin C$),
$\kappa[\mathsf{readchan}\ \mathsf{chanvar}\ \gamma']\ \mathbf{at}\ \gamma$ (where $V_h :: V_t\ \mathbf{at}\ \gamma' \notin C$),
$\kappa[\mathsf{writechan}\ (\mathsf{chanvar}\ \gamma')\ V]\ \mathbf{at}\ \gamma$ (where $V_1 :: \cdots :: V_n :: \mathsf{nil}\ \mathbf{at}\ \gamma' \notin C$). □