Detecting Local Channels in Distributed Poly/ML

Paul Steckler*

Keywords: static analysis, escape analysis, concurrency, communication, locality, ML

Abstract

Distributed Poly/ML is a variation on Standard ML that includes primitives for creating threads and for inter-thread communication. Threads may be spawned on remote machines. Values are sent from one thread to another over dynamically-created channels. A channel is considered local iff all its uses take place on the processor on which it was created. We present a constraint-based static analysis that detects local channels. Using a tree replacement technique, we show that constraint solutions may be maintained as invariants at each transition step in a concurrent operational semantics. Relying on such invariants, we prove the soundness of the analysis with respect to the operational semantics.

1 Introduction

Distributed Poly/ML (DP/ML) is an implementation of a variation on Standard ML [8] that provides primitives for creating threads and for communications between threads. An overview of the implementation is described in [5]. A child thread may run on the same processor as its parent thread, or on a different processor. Values are sent between threads over dynamically-created channels. Channels are themselves values and so may be passed from one thread to another. Channels may also be contained within data structures passed between threads. Therefore, a channel may be used in a thread running on a processor different than the one where it was created.

If it can be determined statically that a channel will be used only for communications on the processor on which it was created, the compiler run-time system may

*LFCS, Department of Computer Science, The King's Buildings, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, `steck@dcs.ed.ac.uk`.

be able to make useful assumptions about the channel's memory usage. DP/ML's run-time system includes a shared distributed memory manager that is responsible for maintaining coherence among the possibly several memory spaces used by a program [6]. Creating a channel allocates a memory block on a machine. If it can be determined that a channel will only be used for communications on the same processor on which the channel was created, the run-time system can treat the channel's memory as purely local. Another possibly beneficial consequence of locality is that all communications on a local channel are between thread pairs on a single processor. Such communications do not involve a network, so the run-time sytem may be able to use a superior protocol.

One goal of this work was to see if constraint-based static analysis methods could be adapted to the concurrent setting. In [19], [16], and [15], for instance, we used constraint-based analyses to support provably-correct code transformations for closure conversion and thunk creation. In those analyses, our proofs were of *partial* correctness: if untransformed programs terminated, then so did their transforms, with comparable results. Also, the reduction behavior of the languages used in thoses analyses was specified by big-step, "natural"-style operational semantics. By contrast, for DP/ML, we are interested in non-terminating computations: Imagine a thread that acts as a server for other, client threads. A big-step operational semantics is too inaccurate a specification of concurrency. Therefore, we specify the behavior of programs using an interleaving small-step operational semantics. As much as the analysis itself, we regard our proof method to show the soundness of the analysis as the contribution here. The proof method essentially consists of showing that once we have a solution to the specified constraints, we can easily produce solutions at each transition step.

2 Detecting locality

A channel is local iff it is used only on the processor on which it is created. Like many static analysis problems, it is undecidable whether a given channel is local. So our approach is to determine which channels may be non-local; the remaining channels are certain to be local. In this section, we consider the situations which require us to consider channels non-local. These considerations will be reflected in the constraints.

Channels are created using the `channel()` primitive. A thread sends a value to another thread using the `send()` primitive, which takes two arguments, a channel and a value. A receiving thread uses the `receive()` primitive, which takes a channel argument. Threads are created using either the `fork()` primitive, which runs the child thread on the same processor as its parent, or the `rfork()` primitive, which runs the child thread on a specified processor. Both `fork()` and `rfork()` take a function argument, whose body becomes the child thread. `rfork()` takes an additional processor argument that specifies where the child thread is to run.

Our analysis answers two questions about channels:

1. can channels escape from the processors where they are created?
2. for channels that may escape, can they reach a `send()` or `receive()` after escaping?

The analysis considers a channel as non-local if it may escape from the processor where it is created and subsequently reach a `send()` or `receive()`.

In the language we consider, there are three ways in which a channel may escape from one processor to another. One way is if the channel is a subterm of an `rfork()`'d function:

$$\begin{array}{l} \textbf{let } x = \texttt{channel()} \\ \textbf{in } \texttt{rfork}(\rho, \texttt{fn}\, y.\ ...\ x\ ...) \end{array}$$

where the **let** is sugared function application. Since the `rfork()`'d function may run on a different processor than its parent, we conclude that the channel x may escape.

A channel is a value that may be sent over channels. Therefore, another way in which a channel may escape is if the channel itself is sent between threads running on different processors. Suppose on processor ρ, we have just:

$$\texttt{send}(k', k)$$

and on processor ρ' we again have

$$\texttt{receive}(k')$$

where k and k' are channel constants. Since k may be sent over k', we conclude that k may escape from one processor to another.

A third way in which a channel may escape is if it is a proper subterm of a value sent between threads running on different processors. Suppose on processor ρ, we have the thread:

$$\texttt{send}(k', \texttt{fn}\, x.\ ...k\ ...)$$

and on processor ρ', we have:

$$\texttt{receive}(k')$$

where again k and k' are channel constants. Since k is contained in a function that may be transmitted over the channel k', k may escape.

These examples show how our analysis deals with question (1) above. Why do we require the additional information asked by question (2)? Simply determining whether a channel may escape from one processor to another would not capture the locality of a channel that escapes from the processor where it is created, but does not reach any communications primitive after escaping. Also, since our escape analysis is necessarily approximate, we may judge that a channel may escape, when in fact it does not. Adding the reaching information may give better results than an escape analysis alone. It may be possible to increase accuracy further by other techniques, such as folding of conditionals with known test results.

Since channels are created dynamically, our actual interest is not in which channels are local, but which occurrences of the `channel()` primitive produce only local channels. With that information, we can change such occurrences of `channel()` to occurrences of a different primitive `local-channel()` in DP/ML intermediate code.

3 The Language

Taking a similar tack to [4] for a partial evaluator, our analysis is performed on the untyped intermediate language of the DP/ML compiler. A simplified and idealized version of terms in that language is given by the grammar:

$$\begin{array}{lll} M & ::= & x \mid \mathbf{true} \mid \mathbf{false} \mid c \mid \rho \mid k \mid \mathtt{unit} \mid \mathtt{fn}^{\ell}\, x.M \mid MM \mid \\ & & \mathtt{if}\ M\ \mathtt{then}\ M\ \mathtt{else}\ M \mid \mathtt{channel}^{\ell}() \mid \mathtt{fork(}M\mathtt{)} \mid \mathtt{rfork(}M\mathtt{,}M\mathtt{)} \mid \\ & & \mathtt{send(}M\mathtt{,}M\mathtt{)} \mid \mathtt{receive(}M\mathtt{)} \end{array}$$

where

- c ranges over an infinite set of constants,
- ρ ranges over a finite set of processors,
- k ranges over an infinite set of channels, and
- `unit` is a constant corresponding to the ML type of the same name.

Let $\Lambda_{DP/ML}$ be the set of terms described by this grammar.

We will refer to terms of the form $\mathtt{fn}^{\ell}\, x.M$ as functions. Each function occurrence and each occurrence of `channel()` has an associated label drawn from some alphabet, which we need not specify. We use ℓ, ℓ', and so on as metavariables which range over the set of labels. Initially, each label in a $\Lambda_{DP/ML}$ term should be distinct. Transitions in the operational semantics produce new $\Lambda_{DP/ML}$ terms whose labels need not be distinct.

4 Operational Semantics

In describing the operational semantics for $\Lambda_{DP/ML}$, we use the following definitions:

Definitions 1

1. *A* thread *is a $\Lambda_{DP/ML}$ term.*
2. *Let TI be an infinite set of* thread identifiers*. We use p and q as typical elements of TI.*
3. *A* thread map Π *is a finite map from thread identifiers to pairs of processors and threads.*

For a thread map Π, we may write $\mathbf{proc}_\Pi(p)$ for the processor associated with the thread identifier p in Π. Likewise, we may write $\mathbf{thread}_\Pi(p)$ for the $\Lambda_{DP/ML}$ term associated with p in Π. If a thread map is indicated by the context, we may drop the subscripts and write just $\mathbf{proc}(p)$ and $\mathbf{thread}(p)$.

Our semantics is divided into a two-level execution hierarchy. We give transition relations for each level. The first level describes sequential evaluation; the corresponding one-step transition relation is $\underset{seq}{\longrightarrow}$. The second level describes transitions within a thread map; the one-step transition relation for this level is $\underset{con}{\longrightarrow}$.

Following [14] and [10], sequential evaluation consists of reduction within a context. Our evaluation contexts are given by the grammar:

$$\begin{array}{rcl} C & ::= & \_ \mid C\ M \mid M\ C \mid \\ & & \texttt{if}\ C\ \texttt{then}\ M\ \texttt{else}\ M \mid \texttt{fork(}C\texttt{)} \mid \\ & & \texttt{rfork(}C\texttt{,}M\texttt{)} \mid \texttt{rfork(}M\texttt{,}C\texttt{)} \mid \\ & & \texttt{send(}C\texttt{,}M\texttt{)} \mid \texttt{send(}M\texttt{,}C\texttt{)} \mid \texttt{receive(}C\texttt{)} \end{array}$$

where M ranges over $\Lambda_{DP/ML}$ terms. We write $C[M]$ to indicate that the hole in the context C is filled by M. Note that a context hole cannot occur in the body of a function. Our evaluation contexts do not require a definite evaluation order. Of course, DP/ML uses a specific evaluation order, but that is not significant for our analysis.

In the sequential semantics in Figure 1, the metavariable M ranges over arbitrary $\Lambda_{DP/ML}$ terms. The metavariable V is used to indicate values, which are functions and all constants.

In Figure 2, we give the rules for thread map evaluation. Within each rule, K is a set of channels.

Let $\underset{con}{\longrightarrow}^{+}$ be the transitive closure, and let $\underset{con}{\longrightarrow}^{*}$ be the reflexive and transitive closure of $\underset{con}{\longrightarrow}$. Also, $\Pi \underset{con}{\longrightarrow}^{n} \Pi''$ iff $\Pi \underset{con}{\longrightarrow}^{*} \Pi'$ in n steps.

β**:**

$$C[(\mathtt{fn}^{\ell}\,x.M)\ V] \underset{seq}{\longrightarrow} C[M[V/x]]$$

cond-true:

$$C[\,\mathtt{if}\ \mathbf{true}\ \mathtt{then}\ M\ \mathtt{else}\ N] \underset{seq}{\longrightarrow} C[M]$$

cond-false:

$$C[\,\mathtt{if}\ \mathbf{false}\ \mathtt{then}\ M\ \mathtt{else}\ N] \underset{seq}{\longrightarrow} C[N]$$

Figure 1: Sequential evaluation within contexts

seq:

$$\frac{C[M] \underset{seq}{\longrightarrow} C[M']}{K, \Pi[\,p:\rho, C[M]] \underset{con}{\longrightarrow} K, \Pi[\,p:\rho, C[M']]}$$

channel:

$$\frac{k \notin K}{K, \Pi[\,p:\rho, C[\mathtt{channel}^{\ell}()]] \underset{con}{\longrightarrow} K \cup \{k\}, \Pi[\,p:\rho, C[k]]}$$

fork:

$$\frac{q \notin Dom(\Pi) \cup \{p\}}{K, \Pi[\,p:\rho, C[\mathtt{fork(fn}^{\ell}\,x.M\mathtt{)}]] \underset{con}{\longrightarrow} K, \Pi[\,p:\rho, C[\mathtt{unit}]][\,q:\rho, M[\mathtt{unit}/x]]}$$

rfork:

$$\frac{q \notin Dom(\Pi) \cup \{p\}}{\begin{array}{r} K, \Pi[\,p:\rho, C[\mathtt{rfork(}\rho'\mathtt{,fn}^{\ell}\,x.M\mathtt{)}]] \underset{con}{\longrightarrow} \qquad\qquad \\ K, \Pi[\,p:\rho, C[\mathtt{unit}]][\,q:\rho', M[\mathtt{unit}/x]] \end{array}}$$

comm:

$$\frac{k \in K}{\begin{array}{r} K, \Pi[\,p:\rho, C[\mathtt{send(}k\mathtt{,}V\mathtt{)}]][\,q:\rho', C'[\mathtt{receive(}k\mathtt{)}]] \underset{con}{\longrightarrow} \\ K, \Pi[\,p:\rho, C[\mathtt{unit}]][\,q:\rho', C'[V]] \end{array}}$$

Figure 2: Thread map evaluation

5 Occurrences indices; labels

Each source program occurrence has an associated string called an *occurrence index*. Occurrence indices are finite strings over the alphabet:

$$\left\{ \begin{array}{l} bv, body, rator, rand, test, then, else, \\ ffun, rfpor, rffun, schan, sbody, rchan \end{array} \right\}$$

For an occurrence of a subterm within a term M, an occurrence index describes the path from the root of the parse tree for M, to the occurrence of the subterm. We may wish to refer to the occurrence indices of distinct terms in a thread map. To distinguish indices from distinct threads, we may qualify the indices by thread identifiers. For instance, we may write $p : i$ for the occurrence i from the term with thread identifier p.

We may think of an occurrence index as a pointer into a parse tree. A dereferencing operator $[\![-]\!]$ may be applied to recover the underlying term: if M is a term with occurrence index i, then $[\![i]\!] = M$.

For an occurrence of a function or `channel()` with index i, we write $\mathbf{lab}(i)$ to indicate the label associated with the occurrence.

The grammar of $\Lambda_{DP/ML}$ terms specifies several kinds of expressions. We may use predicates such as *Var*, *Fun*, *Fork*, and so on to test which production in the grammar produced an occurrence of a term. The predicate *Const* is true given an argument that is an ordinary constant, boolean constant, processor constant, channel constant, or the unit constant.

Note that $[\![-]\!]$, $\mathbf{lab}(-)$, and the predicates just mentioned implicitly depend on particular thread maps. It should be clear from context which thread map is meant when we use these constructs.

6 Propositions

Here we describe the annotations that are associated with program occurrences:

- A *flow* ϕ is a finite set of function labels. For a given occurrence, its flow is a conservative estimate of the labels of functions to which that occurrence might evaluate. Our flow annotations give us what is usually referred to as a *closure analysis*.
- A *channel set* κ is a finite set of `channel()` labels. For an occurrence i, its channel set is a conservative estimate of the labels of `channel()` occurrences that produce the channels to which i may evaluate.

- An *escape flag* θ is an element of the set $\{\mathbf{escape}^-, \mathbf{escape}^\pm\}$, with the ordering $\mathbf{escape}^- \leq \mathbf{escape}^\pm$. $\mathbf{escape}^\pm$ suggests that an occurrence may escape from one processor to another, while $\mathbf{escape}^-$ indicates that an occurrence certainly does not go from one processor to another.

- A *reachability flag*, σ, is an element of the set $\{\mathbf{reach}^-, \mathbf{reach}^\pm\}$, with the ordering $\mathbf{reach}^- \leq \mathbf{reach}^\pm$. $\mathbf{reach}^\pm$ suggests that a value may reach the channel-part of a `send()` or `receive()` primitive, while $\mathbf{reach}^-$ suggests that a value does not reach such a channel-part.

- A *linearity flag* ν is an element of the set $\{1, \infty\}$, ordered $1 \leq \infty$. A linearity flag of ∞ suggests that an occurrence may be duplicated by a function application; 1 indicates that an occurrence is not duplicated.

- A *locality flag* ω is an element of the set $\{\mathbf{local}^+, \mathbf{local}^\pm\}$, with the ordering $\mathbf{local}^+ \leq \mathbf{local}^\pm$. $\mathbf{local}^+$, when associated with an occurrence of `channel()`, suggests that it produces only local channels. $\mathbf{local}^\pm$ suggests that the primitive may produce non-local channels.

These descriptions are meant to guide the reader's intuition. An exact semantics of the annotations will be given later, in section 11.

In a source program, each occurrence i will be annotated with a pair $(\vec{\mathcal{P}}_i, \overleftarrow{\mathcal{P}}_i)$, which we may indicate as $\mathcal{P}_i$. We call such a pair a *proposition*.

The $\vec{\mathcal{P}}_i$ component of a proposition is a *forwards proposition*, and consists of a pair (ϕ_i, κ_i). The set of forwards propositions is partially-ordered. Say that $(\phi, \kappa) \leq (\phi', \kappa')$ iff $\phi \subseteq \phi'$ and $\kappa \subseteq \kappa'$. We define a join operation on forwards propositions: Let $(\phi, \kappa) \sqcup (\phi', \kappa') = (\phi \cup \phi', \kappa \cup \kappa')$.

The $\overleftarrow{\mathcal{P}}_i$ component of a proposition is a *backwards proposition*, and consists of a 4-tuple $(\theta_i, \sigma_i, \nu_i, \omega_i)$. The set of backwards propositions is partially-ordered. Say that $(\theta, \sigma, \nu, \omega) \leq (\theta', \sigma', \nu', \omega')$ iff $\theta \leq \theta'$, $\sigma \leq \sigma'$, $\nu \leq \nu'$, and $\omega \leq \omega'$. We can also define a join operation on backwards propositions. Let $(\theta, \sigma, \nu, \omega) \sqcup (\theta', \sigma', \nu', \omega') = (\mathbf{max}(\theta, \theta'), \mathbf{max}(\sigma, \sigma'), \mathbf{max}(\nu, \nu'), \mathbf{max}(\omega, \omega'))$.

By referring to the partial orders on forwards and backwards propositions, we can partially order the set of propositions. Let $\mathcal{P} = (\vec{\mathcal{P}}, \overleftarrow{\mathcal{P}})$ and $\mathcal{P}' = (\vec{\mathcal{P}}', \overleftarrow{\mathcal{P}}')$. Say that $\mathcal{P} \leq \mathcal{P}'$ iff $\vec{\mathcal{P}} \leq \vec{\mathcal{P}}'$ and $\overleftarrow{\mathcal{P}} \leq \overleftarrow{\mathcal{P}}'$. This partial order allows us to identify a least element in the set of propositions.

We also want another order relation on propositions. Again let $\mathcal{P} = (\vec{\mathcal{P}}, \overleftarrow{\mathcal{P}})$ and $\mathcal{P}' = (\vec{\mathcal{P}}', \overleftarrow{\mathcal{P}}')$. Say that $\mathcal{P} \stackrel{\rightarrow}{\leq} \mathcal{P}'$ iff $\vec{\mathcal{P}} \leq \vec{\mathcal{P}}'$ and $\overleftarrow{\mathcal{P}}' \leq \overleftarrow{\mathcal{P}}$. Intuitively, this relation reflects the notion that the forwards propositions, which consist of ϕ's and κ's, track the forward flow of function and channel labels, while the backwards propositions, consisting of θ, σ, ν, and ω annotations, track information which gets propagated from program points back to occurrences which may reach those points.

7 Annotation maps

An *annotation map* Γ associates occurrence indices with propositions. Such a map associates each node i in the parse tree of a thread with a proposition $\mathcal{P}_i$. The domain of an annotation map is prefix-closed, since each occurrence in a parse tree has an annotation. Hence, we may consider an annotation map itself as a tree.

When needed, we may write $\mathcal{P}_i^\Gamma$ for $\Gamma(i)$ to indicate that it is a particular annotation map Γ that associates the occurrence i with the proposition $\mathcal{P}_i$. We may also write $\mathcal{P}_{p:i}$ for the proposition for the occurrence i in the thread with thread identifier p. Similarly, we may write $\phi_{p:i}$ for the flow component of $\mathcal{P}_{p:i}$, and so on for the other components of propositions.

For an occurrence index i, the tree Γ/i is the subtree of Γ rooted at i with domain $\{j \mid i.j \in \mathit{Dom}(\Gamma)\}$, so that for all j in the domain of the subtree, $(\Gamma/i)(j) = \Gamma(i.j)$.

7.1 Local consistency

In Figure 3, we give constraints on the annotations in annotation maps. These constraints are presented as local conditions for each occurrence in a $\Lambda_{DP/ML}$ term. We refer to certain constraints on the annotations of functions as "escape constraints"; those are presented in Figure 4.

Let us define some terminology used in the constraints. We say that an annotation map for a term M is *remote-expectant* at an occurrence j in M iff

$$\begin{array}{l} \forall k, m \text{ such that } \mathit{Send}(k) \wedge \mathit{Receive}(m), \\ \quad \kappa_{k.schan} \cap \kappa_{m.rchan} \neq \emptyset, \text{ and} \\ \quad \text{exactly one of } k \text{ and } m \text{ is a subterm of } j, \\ \quad\quad \begin{cases} \theta_{k.sbody} = \mathbf{escape}^{\pm}, \\ \sigma_m = \mathbf{reach}^{\pm} \Rightarrow \omega_m = \mathbf{local}^{\pm} \end{cases} \end{array}$$

where the quantifier ranges over occurrences in M. We require remote-expectancy for functions that may be `rfork()`'d or may otherwise escape from one processor to another. The idea is that the value sent from a `send()` to a `receive()`, where one of the communications partners is a subterm of a possibly escaping function, may be sent from one processor to another. Therefore, we tag the body of the `send()` with a θ-annotation of $\mathbf{escape}^{\pm}$, which gets propagated back to any value that may be sent by that `send()`. Since a sent value may be a channel, in case the sent value may reach a `send()` or `receive()`, we tag the `receive()` partner as non-local, which gets propagated back to such a sent channel.

Similarly, an annotation map for a term M is *nonlinear-expectant* at an occurrence j in M iff

$$\nu_j = \infty \Rightarrow \begin{cases} \forall k, m \text{ such that } \mathit{Send}(k) \wedge \mathit{Receive}(m), \\ \quad \kappa_{k.schan} \cap \kappa_{m.rchan} \neq \emptyset, \text{ and} \\ \quad \text{both } k \text{ and } m \text{ are subterms of } j, \\ \qquad \begin{cases} \theta_{k.sbody} = \mathbf{escape}^{\pm}, \\ \sigma_m = \mathbf{reach}^{\pm} \Rightarrow \omega_m = \mathbf{local}^{\pm} \end{cases} \end{cases}$$

where the quantifier ranges over occurrences in M. We also require nonlinear-expectancy of functions that may be `rfork()`'d or may otherwise escape. If we have possibly-communicating `send()` and `receive()` subterms of a function that may escape, and that function may be duplicated, values may sent between duplicates running on different processors. Therefore, we tag the `send()` and `receive()` in anticipation of that possibility.

The constraints may be solved by iteration. We may start off by annotating each occurrence with a proposition that is $\leq$-minimal. Suppose we have a constraint $\mathcal{P} \vec{\leq} \mathcal{P}'$ that is not satisfied. Since our analysis mixes forwards and backwards components, we may need to adjust *both* $\mathcal{P}$ and $\mathcal{P}'$ by propagating forwards information from $\mathcal{P}$ to $\mathcal{P}'$ and backwards information from $\mathcal{P}'$ to $\mathcal{P}$. More specifically, suppose $\mathcal{P} = (\vec{\mathcal{P}}, \overleftarrow{\mathcal{P}})$, $\mathcal{P}' = (\vec{\mathcal{P}}', \overleftarrow{\mathcal{P}}')$. If $\mathcal{P} \vec{\not\leq} \mathcal{P}'$, then we may enforce the constraint by setting $\mathcal{P} = (\vec{\mathcal{P}}, \overleftarrow{\mathcal{P}} \sqcup \overleftarrow{\mathcal{P}}')$ and setting $\mathcal{P}' = (\vec{\mathcal{P}} \sqcup \vec{\mathcal{P}}', \overleftarrow{\mathcal{P}}')$. Propagating information in this way moves both $\mathcal{P}$ and $\mathcal{P}'$ up the $\leq$ order, not the $\vec{\leq}$ order. Termination is assured because the height of any chain is finite.

An annotation map gives the annotations of a particular term. An annotation map is *locally consistent for a term M* iff the constraints in Figure 3 are satisfied at all the occurrence indices of M. The universal quantified indices specified in the constraints range over the indices in M. Explicitly, an annotation map may have extra elements in its domain that are not indices in a given term; those extra elements are irrelevant when considering the local consistency of the annotation map for that term.

Since channels in DP/ML programs are created dynamically, they do not occur in source programs. Therefore, the DP/ML compiler does not have to solve the constraint on annotations of channel subterms of functions given in Figure 3.

A useful result about local consistency is:

Lemma 1 *Let Γ be a locally consistent annotation for a term M. Then for any i that is the index of a subterm of M, Γ/i is locally consistent for the subterm rooted at i.*

$Var(i) \Rightarrow$ if j is the binding occurrence for i, then $\mathcal{P}_j \overset{\rightarrow}{\leq} \mathcal{P}_i$

$Channel(i) \Rightarrow \mathbf{lab}(i) \in \kappa_i$

$Fun(i) \Rightarrow \mathbf{lab}(i) \in \phi_i$,
no. of free occurrences of $[\![i.bv]\!]$ in $[\![i.body]\!] > 1 \Rightarrow \nu_{i.bv} = \infty$,
$\theta_i = \mathbf{escape}^{\pm} \Rightarrow$
escape constraints hold at i
$\nu_i = \infty \Rightarrow$
for all free variable and function occurrences $i.q$, $\nu_{i.q} = \infty$

$App(i) \Rightarrow \forall j$ such that $\mathbf{lab}(j) \in \phi_{i.rator}$,
$$\begin{cases} \mathcal{P}_{i.rand} \overset{\rightarrow}{\leq} \mathcal{P}_{j.bv}, \\ \mathcal{P}_{j.body} \overset{\rightarrow}{\leq} \mathcal{P}_i \end{cases}$$

$Cond(i) \Rightarrow \mathcal{P}_{i.then}, \mathcal{P}_{i.else} \overset{\rightarrow}{\leq} \mathcal{P}_i$

$RFork(i) \Rightarrow \forall j$ such that $\mathbf{lab}(j) \in \phi_{i.rffun}$,
escape constraints hold at j

$Send(i) \Rightarrow \sigma_{i.schan} = \mathbf{reach}^{\pm}$

$Receive(i) \Rightarrow \sigma_{i.rchan} = \mathbf{reach}^{\pm}$,
$\forall j$ such that $Send(j) \wedge \kappa_{j.schan} \cap \kappa_{i.rchan} \neq \emptyset$,
$\mathcal{P}_{j.sbody} \overset{\rightarrow}{\leq} \mathcal{P}_i$

Figure 3: Local consistency constraints

$\forall j$ such that $i.j$ is a free variable occurrence in i,
$\theta_{i.j} = \mathbf{escape}^{\pm}$
$\sigma_{i.j} = \mathbf{reach}^{\pm} \Rightarrow \omega_{i.j} = \mathbf{local}^{\pm}$
$\forall m$ such that $ChanConst(i.m)$,
$\sigma_{i.m} = \mathbf{reach}^{\pm} \Rightarrow \omega_{i.m} = \mathbf{local}^{\pm}$
Γ is remote-expectant at i,
Γ is nonlinear-expectant at i

Figure 4: Escape constraints for a function i

Proof. By the local consistency of Γ/i at each node of the subtree. ∎

The preceding lemma is slightly less obvious than it sounds. The essential observation is that there are possibly fewer constraints on the annotations for the subterm of M than for M itself. For instance, suppose that in the subtree rooted at i contains an application with index j. By the local consistency conditions, we have constraints involving all k such that $\mathbf{lab}(k) \in \phi_{j.rator}$. The same application appears in M, with index, say, m. The comparable constraint involves all k' such that $\mathbf{lab}(k') \in \phi_{m.rator}$. Note that the sets of k's and k''s are not necessarily the same, because the term and its subterm may be different. But any function occurrence with a given label in the subterm is also in M. Therefore, if the constraints at m were satisfied in Γ for M, the constraints at j are also satisfied in Γ/i for the subterm. Similar considerations arise when considering the other constraints involving a universal quantifier, such as those for `rfork()`'s and `receive()`'s. Also, taking a subtree may take a variable out of the lexical scope of its binder. That has the effect of removing the constraint on the variable's annotation.

Similarly:

Lemma 2 *Let Γ be a locally consistent annotation map for a term M. Suppose we substitute a constant, other than a channel constant (an ordinary constant, boolean constant, processor constant, or the unit constant) for a subterm N of M. Then Γ is locally consistent for the substituted term.*

Proof. By considering the constraints for each occurrence in the substituted term.

Observe that there are no constraints on the kinds of constants we have indicated, so that a substitution does not introduce any new constraints. Any constraints that applied to occurrences in the replaced subterm no longer apply in the substituted term. ∎

Corollary Suppose Γ is a locally consistent annotation map for a term M with zero or more free occurrences of a variable x. Let c be any constant other than a channel constant. Then Γ is locally consistent for $M[c/x]$. ∎

We can handle channel constants, too, by making a slight restriction:

Lemma 3 *Let Γ be a locally consistent annotation map for a term M. Suppose we substitute a channel constant for a subterm N of M, where N is not a subterm of any function body. Then Γ is locally consistent for the substituted term.*

Proof. By considering the constraints for each occurrence in the substituted term.

Observe that the only possible constraints on constants are for channel constants in function bodies, so that a substitution as we have indicated does not introduce any new constraints. ∎

A thread map associates thread identifiers with threads, that is, with $\Lambda_{DP/ML}$ terms. Therefore, for a given thread map, we have a family of annotation maps indexed by the domain of the thread map. We write Γ_p to indicate the annotation map for the thread associated with thread identifier p. We say that such a family of annotation maps is *locally consistent for a thread map* Π iff for each thread identifier p in the domain of Π, the annotation map Γ_p is locally consistent for $\mathbf{thread}_\Pi(p)$.

7.2 Communicative consistency

Local consistency imposes constraints on the annotations of a particular term. Since data may be sent between threads, we use the additional notion of *communicative consistency* to account for those data flows. The communicative consistency constraints for annotations of occurrences in thread maps are given in Figure 5.

Most of the communicative consistency constraints are essentially the same as local consistency constraints, except that they hold between occurrences from different threads. As an example, we need the "communicative analogue" of remote-expectancy. Say that a family of annotation maps $\mathcal{G}$ is *distributed remote-expectant* at an occurrence $p:j$ iff

$$
\begin{array}{l}
\forall k, m \text{ such that } \mathit{Send}(k) \wedge \mathit{Receive}(m), \\
\quad \kappa_{k.schan} \cap \kappa_{m.rchan} \neq \emptyset \text{ and} \\
\quad \text{exactly one of } k \text{ and } m \text{ is a subterm of } p:j, \\
\qquad \begin{cases} \theta_{k.sbody} = \mathbf{escape}^{\pm}, \\ \sigma_m = \mathbf{reach}^{\pm} \Rightarrow \omega_m = \mathbf{local}^{\pm} \end{cases}
\end{array}
$$

This definition is essentially the same as that for remote-expectancy. The difference is that while remote-expectancy constrains the annotations on `send()`'s and `receive()`'s in a particular thread, distributed remote-expectancy constrains the annotations on `send()`'s and `receive()`'s in possibly different threads.

Observe:

Lemma 4 *Any family of annotation maps that is locally consistent for the empty thread map, or for a thread map with exactly one element in its domain, is communicative consistent for that thread map.*

Proof. True trivially. ∎

The significance of Lemma 4 is that the DP/ML compiler, which optimizes code for just an initial thread, only has to solve the local consistency constraints.

$$
\begin{array}{l}
\mathit{Fun}(p:i) \Rightarrow \\
\quad \theta_{p:i} = \mathbf{escape}^{\pm} \Rightarrow \\
\qquad \mathcal{G} \text{ is distributed remote-expectant at } p:i \\
\\
\mathit{App}(p:i) \Rightarrow \\
\quad \forall q \neq p, \\
\qquad \forall j \text{ such that } \mathbf{lab}(q:j) \in \phi_{p:i.rator}, \\
\qquad\quad \begin{cases} \mathcal{P}_{p:i.rand} \overset{\rightarrow}{\leq} \mathcal{P}_{q:j.bv}, \\ \mathcal{P}_{q:j.body} \overset{\rightarrow}{\leq} \mathcal{P}_{p:i} \end{cases} \\
\\
\mathit{RFork}(p:i) \Rightarrow \\
\quad \forall q, j \text{ such that } \mathbf{lab}(q:j) \in \phi_{p:i.rffun}, \\
\qquad \mathcal{G} \text{ is distributed remote-expectant at } q:j, \\
\quad \forall q \neq p, \\
\qquad \forall k \text{ such that } \mathbf{lab}(q:k) \in \phi_{p:i.rffun}, \\
\qquad\quad \text{escape constraints hold at } q:k \\
\\
\mathit{Receive}(p:i) \Rightarrow \\
\quad \forall q \neq p, \\
\qquad \forall j \text{ such that } \mathit{Send}(q:j) \wedge \kappa_{p:i.rchan} \cap \kappa_{q:j.schan} \neq \emptyset, \\
\qquad\quad \mathcal{P}_{q:j.sbody} \overset{\rightarrow}{\leq} \mathcal{P}_{p:i}, \text{ and} \\
\qquad\quad \mathbf{proc}(p) \neq \mathbf{proc}(q) \Rightarrow \\
\qquad\qquad \begin{cases} \theta_{q:j.sbody} = \mathbf{escape}^{\pm}, \\ \sigma_{p:i} = \mathbf{reach}^{\pm} \Rightarrow \omega_{p:i} = \mathbf{local}^{\pm} \end{cases}
\end{array}
$$

Figure 5: Communicative consistency constraints

8 A coherence condition

The κ component of a proposition is a set of labels of `channel()` occurrences, and intended to indicate which occurrences of `channel()` may have produced a channel constant. Since channels are fresh when created, we would expect that if we saw two occurrences of the same channel constant in a program, at least one `channel()` label would be common to the κ annotations for those occurrences.

To enforce this property, we impose an additional condition on families of annotation maps:

Definition 1 *A family of annotation maps for a thread map is* κ-coherent *iff for all pairs of distinct occurrences $p:i$, $q:j$ of channel constants, where p may be the same as q, such that $[\![p:i]\!] = [\![q:j]\!]$, $\kappa_{p:i} \cap \kappa_{q:j} \neq \emptyset$.*

In an actual DP/ML program, there are no channel constants when the program is started, so the family of annotation maps for the thread map containing just an initial thread is necessarily κ-coherent.

A `send()` and `receive()` can only communicate if they share a channel. Therefore, we expect that the channel-parts of a `send()` and `receive()` which may communicate will have κ-annotations whose intersection is non-empty. This related notion is captured by:

Definition 2 *An occurrence i of* `send()` *and an occurrence j of* `receive()` *are* possible communications partners *iff* $\kappa_{i.schan} \cap \kappa_{j.rchan} \neq \emptyset$.

9 Transitions and annotation map updates

Given a family of annotation maps for the thread map on the left-hand side of a $\xrightarrow[con]{}$-transition, we would like to produce another family of annotation maps for the thread map on the right-hand side that preserves the consistency and coherence conditions we have stated. Such preservation may be likened to familiar subject reduction properties for typed languages.

9.1 Tree replacement and merger

How may annotation maps be updated? We first present the usual notion of tree replacement.

For two annotation maps Γ, Γ', and an index i, $\Gamma[i \leftarrow \Gamma']$ indicates the tree obtained by replacing the subtree of Γ rooted at i with Γ'. The updated tree is a function defined by:

$$\Gamma[i \leftarrow \Gamma'](j) = \begin{cases} \Gamma'(k) & \text{if } j = i.k \\ \\ \Gamma(j) & \text{otherwise} \end{cases}$$

Note that Γ' may itself be a subtree of Γ.

For updating annotation maps, we will also want to perform a variation of tree replacement in which the updated tree retains its annotation at the replacement node. In this variation, the updated tree is a function defined by:

$$\Gamma[i \hookleftarrow \Gamma'](j) = \begin{cases} \Gamma'(k) & \text{if } j = i.k,\ k \neq \epsilon \\ \\ \Gamma(j) & \text{otherwise} \end{cases}$$

We call this operation *tree merger*. We can simultaneously merge several subtrees $\Gamma/i_1, \ldots, \Gamma/i_n$ of Γ with a single subtree Γ'. The updated tree has the expected definition:

$$\Gamma[i_1, \ldots, i_n \hookleftarrow \Gamma'](j) = \begin{cases} \Gamma'(k) & \text{if } j = i_s.k,\ 1 \le s \le n,\ k \ne \epsilon \\ \Gamma(j) & \text{otherwise} \end{cases}$$

10 Specifying the updates

Given a family of annotation maps for a thread map on the left-hand side of a $\underset{con}{\longrightarrow}$ transition, we can obtain a family of annotation maps for the thread map on the right-hand side. How to obtain the new family depends on the particular transition taken. In Figures 6 through 12, we present rules for annotation map updates for each kind of $\underset{con}{\longrightarrow}$ transition. In these figures, the unprimed Γ's are the left-hand side annotation maps, and the Γ′'s are the updated, right-hand side maps. Also in these figures, we define the notion of *occurrence predecessor* for right-hand side occurrences.

The occurrence predecessor of an occurrence on the right-hand side of a transition is some occurrence on the left-hand side. We may think of the occurrence predecessor of a right-hand side occurrence as the "same" occurrence on the left-hand side. Most, but not all all right-hand side occurrences have an occurrence predecessor. The exceptions are the fresh `unit` constants on the right-hands sides of **fork**, **rfork**, and **comm** transitions, and the fresh channel constant on the right-hand side of a **channel** transition. If a right-hand side occurrence has an occurrence predecessor, it is unique. We write $\mathbf{opred}(i)$ for the occurrence predecessor of an occurrence i.

If we have a reduction sequence, rather than a single transition, we can compose **opred** maps. Define $\mathbf{opred}^0$ to be the identity on occurrence indices, and for $n > 0$, $\mathbf{opred}^n = \mathbf{opred} \circ \mathbf{opred}^{n-1}$. When we write, say, $\mathbf{opred}(p : i)$, we contemplate an underlying transition step; so when we write, say, $\mathbf{opred}^n(p : i)$, we contemplate an underlying reduction sequence. In later discussion, we may also refer to $\mathbf{opred}^n(p : i)$, for $n > 1$, as an occurrence predecessor of $p : i$, as well as in the case for $n = 1$.

We have used tree merger in only two places in the update specifications. Notice which right-hand side annotations affected by that choice:

Definition 3 *Let i be the index of the left-hand side context hole in a $\underset{con}{\longrightarrow}$-transition. Say that an occurrence j from the left-hand side is* a value substitution *iff the transition is a*

- **seq**/β *transition, and $j = i.rand$, or a*
- **comm** *transition, and $j = i.sbody$*

Transition:
$\Pi[\,p : \rho, C[(\mathtt{fn}^{\ell}\, x.M)\ V]] \underset{con}{\longrightarrow} \Pi[\,p : \rho, C[M[V/x]]]$

Let i be the occurrence index of the context hole,
$i.rator.body.m_1, \ldots, i.rator.body.m_j$ be the free occurrences
of $[\![i.rator.bv]\!]$ in $i.rator.body$

For thread p :

$$\Gamma'_p = \Gamma_p[i \leftarrow \Gamma_p/i.rator.body[m_1, \ldots, m_j \hookleftarrow \Gamma_p/i.rand]]$$

for any RHS occurrence n,

$$\mathbf{opred}(n) = \begin{cases} i.rand.k & \text{if } n = i.m.k,\ m \in \{m_1, \ldots, m_j\} \\ i.rator.body.k & \text{if } n = i.k \\ n & \text{otherwise} \end{cases}$$

For any thread $q \in Dom(\Pi)$, other than p :
$\Gamma'_q = \Gamma_q$,
for any RHS occurrence n, $\mathbf{opred}(n) = n$

Figure 6: Updates for **seq**/β transition

Because we use tree merger at these points, a right-hand side occurrence whose occurrence predecessor is a value substitution retains the annotation of the the left-hand side occurrence where the value was substituted.

We are also interested in where substitutions occur:

Definition 4 *In a* $\underset{con}{\longrightarrow}$ *-transition, a left-hand side occurrence i is a* substitution site *iff any of the following is true:*

1. *i is the index of a context hole*
2. *in the case of a* **seq**/β *transition, i is the index of a free occurrence of the operator binder in the operator body*
3. *in the case of a* **fork** *or an* **rfork** *transition, i is the index of a free occurrence of the spawned function's binder in the function body*

Observe:

Transition:
$\Pi[\,p:\rho, C[\,\texttt{if}\ \texttt{true}\ \texttt{then}\ M\ \texttt{else}\ N]] \underset{con}{\longrightarrow} \Pi[\,p:\rho, C[M]]$

Let i be the occurrence index of the context hole

For thread p :

$\Gamma'_p = \Gamma_p[i \leftarrow \Gamma_p/i.then]$

for any RHS occurrence n,

$$\mathbf{opred}(n) = \begin{cases} i.then.k & \text{if } n = i.k \\ \\ n & \text{otherwise} \end{cases}$$

For any thread $q \in Dom(\Pi)$, other than p :
$\Gamma'_q = \Gamma_q$,
for any RHS occurrence $n, \mathbf{opred}(n) = n$

Figure 7: Updates for **seq**/cond-true transition

Lemma 5 *Let Π and Π' be thread maps, and suppose $\Pi \underset{con}{\longrightarrow} \Pi'$. Let s be a symbol in Σ and let i be an occurrence index such that $i.s$ is an occurrence in Π'. In Π,* $\mathbf{opred}(i).s$ *is a substitution site iff* $\mathbf{opred}(i.s)$ *is undefined or* $\mathbf{opred}(i.s) \neq \mathbf{opred}(i).s$.

Proof.

Note that at substitution sites, either (1) a left-hand side occurrence is substituted, so that on the right-hand side, the substituted occurrence has an occurrence predecessor, or (2) some fresh occurrence is substituted (a `unit` constant or a fresh channel constant), which has no occurrence predecessor.

$\Longrightarrow$

Suppose $\mathbf{opred}(i).s$ is a substitution site. Then $i.s$ may be a fresh unit constant or a fresh channel constant, so that $\mathbf{opred}(i.s)$ is undefined. But suppose $\mathbf{opred}(i.s)$ is defined. Then $\mathbf{opred}(k.s)$ must be the substituted term, so $\mathbf{opred}(k).s \neq \mathbf{opred}(k.s)$.

$\Longleftarrow$

Suppose $\mathbf{opred}(k.s)$ is undefined. By the definitions of **opred** for the various $\underset{con}{\longrightarrow}$ transition rules, the only occurrences $k.s$ without occurrence predecessors

Transition:
$\Pi[\,p:\rho, C[\,\texttt{if}\ \mathbf{true}\ \texttt{then}\ M\ \texttt{else}\ N]] \underset{con}{\longrightarrow} \Pi[\,p:\rho, C[N]]$

Let i be the occurrence index of the context hole

For thread p :

$\Gamma'_p = \Gamma_p[i \leftarrow \Gamma_p/i.else]$

for any RHS occurrence n,

$$\mathbf{opred}(n) = \begin{cases} i.else.k & \text{if } n = i.k \\ n & \text{otherwise} \end{cases}$$

For any thread $q \in Dom(\Pi)$, other than p :
$\Gamma'_q = \Gamma_q$,
for any RHS occurrence $n, \mathbf{opred}(n) = n$

Figure 8: Updates for **seq**/cond-false transition

Transition:
$\Pi[\,p:\rho, C[\texttt{channel}^{\ell}()]] \underset{con}{\longrightarrow} \Pi[\,p:\rho, C[k]]$
where k is a fresh channel constant

Let i be the occurrence index of the context hole

For thread p:

$\Gamma'_p = \Gamma_p$

for any RHS occurrence n, other than $n = i$,

$\mathbf{opred}(n) = n$

For any thread $q \in Dom(\Pi)$, other than p :
$\Gamma'_q = \Gamma_q$,
for any RHS occurrence $n, \mathbf{opred}(n) = n$

Figure 9: Updates for **channel** transition

Transition:
$\Pi[p : \rho, C[\texttt{fork(fn}^\ell\, x.M\texttt{)}]] \xrightarrow[con]{} \Pi[p : \rho, C[\texttt{unit}]][q : \rho, M[\texttt{unit}/x]]$
where q is a fresh process identifier

Let i be the occurrence index of the context hole in thread p

For thread p:

$\Gamma'_p = \Gamma_p$

for any RHS occurrence $p : n$, other than $n = i$,

$\textbf{opred}(p : n) = p : n$

For thread q:

$\Gamma'_q = \Gamma_p / i.\mathit{ffun}.\mathit{body}$

for any occurrence $q : n$,

$\textbf{opred}(q : n) = p : i.\mathit{ffun}.\mathit{body}.n,$ unless that is a free occurrence of $[\![p : i.\mathit{ffun}.\mathit{bv}]\!]$

For any thread $r \in \mathit{Dom}(\Pi)$, other than p or q:
$\Gamma'_r = \Gamma_r$,
for any RHS occurrence $r : n, \textbf{opred}(r : n) = r : n$

Figure 10: Updates for **fork** transition

are `unit` constants and fresh channel constants. By the $\xrightarrow[con]{}$ transition rules and the definitions of **opred**, $\textbf{opred}(k).s$ must be the substitution site for such occurrences.

On the other hand, suppose $\textbf{opred}(k).s \neq \textbf{opred}(k.s)$. Then some occurrence must have been substituted at $\textbf{opred}(k).s$. The definition of substitution site covers all the possibilities. ∎

By our specifications of annotation map updates, we have:

Lemma 6 *Let Π be a thread map with a locally consistent, communicative consistent, and κ-coherent family of annotation maps $\mathcal{G}$. Suppose that $\Pi \xrightarrow[con]{} \Pi'$, and let $\mathcal{G}'$ be the updated family of annotation maps for Π' for the particular transition*

Transition:
$\Pi[\,p:\rho, C[\texttt{rfork(}\rho'\texttt{,}\texttt{fn}^{\ell}\,x.M\texttt{)}]] \underset{con}{\longrightarrow} \Pi[\,p:\rho, C[\texttt{unit}]][\,q:\rho', M[\texttt{unit}/x]]$
where q is a fresh process identifier

Let i be the occurrence index of the context hole

For process p:

$\Gamma'_p = \Gamma_p$

for any RHS occurrence $p:n$, other than $n=i$,

$\mathbf{opred}(p:n) = p:n$

For process q:

$\Gamma'_q = \Gamma_p/i.\mathit{rffun.body}$

for any occurrence $q:n$,

$\mathbf{opred}(q:n) = p:i.\mathit{rffun.body.n}$, unless that is a free occurrence of $[\![p:i.\mathit{rffun.bv}]\!]$

For any process $r \in \mathit{Dom}(\Pi)$, other than p or q:
$\Gamma'_r = \Gamma_r$,
for any RHS occurrence $r:n, \mathbf{opred}(r:n) = r:n$

Figure 11: Updates for **rfork** transition

involved. Let $p:j$ be an occurrence in Π', so that Γ'_p is the annotation map for the thread in which $p:j$ occurs. If $\mathbf{opred}(p:j)$ *is defined, let Γ_q be the annotation map for the thread in Π in which* $\mathbf{opred}(p:j)$ *occurs.*

1. *if* $\mathbf{opred}(j)$ *is defined, then* $[\![\mathbf{opred}(j)]\!]$ *is generated by the same production in the DP/ML grammar as* $[\![j]\!]$.
2. *if $p:j$ is the index of a function occurrence or an occurrence of* `channel()`, *then* $\mathbf{opred}(p:j)$ *is defined and* $\mathbf{lab}(p:j) = \mathbf{lab}(\mathbf{opred}(p:j))$.
3. *if $p:j$ is the index of a constant other than the* `unit` *constant, then* $\mathbf{opred}(p:j)$ *is defined and* $[\![p:j]\!] = [\![\mathbf{opred}(p:j)]\!]$, *unless the* $\underset{con}{\longrightarrow}$ *-transition is a* **channel** *transition and $p:j$ is a fresh channel constant.*

Transition:

$$\Pi[\,p:\rho, C[\texttt{send(}k\texttt{,}V\texttt{)}]][\,q:\rho', C'[\texttt{receive(}k\texttt{)}]] \underset{con}{\longrightarrow} \Pi[\,p:\rho, C[\texttt{unit}]][\,q:\rho', C'[V]]$$

Let i, i' be the occurrence indices of the holes in the contexts $C[\,]$, $C'[\,]$

For process p, $\Gamma'_p = \Gamma_p$, and

for any RHS occurrence $p:n$, other than $n = i$,

$\textbf{opred}(p:n) = p:n$

For process q, $\Gamma'_q = \Gamma_q[i' \hookleftarrow \Gamma_p/p:i.\mathit{sbody}]$, and

for any RHS occurrence $q:n$,

$$\textbf{opred}(q:n) = \begin{cases} p:j.\mathit{sbody} & \text{if } n = i'.j \\ q:n & \text{otherwise} \end{cases}$$

For any process $r \in \mathit{Dom}(\Pi)$, other than p or q:
$\Gamma'_r = \Gamma_r$,
for any RHS occurrence $r:n, \textbf{opred}(r:n) = r:n$

Figure 12: Updates for **comm** transition

4. *if* $\textbf{opred}(p:j)$ *is defined, then* $\mathcal{P}^{\Gamma_q}_{\textbf{opred}(p\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_p}_{p:j}$.
5. *if* $\textbf{opred}(p:j)$ *is defined and* $\textbf{opred}(p:j)$ *is not a a value substitution in the transition, then* $\mathcal{P}^{\Gamma_q}_{\textbf{opred}(p\,:\,j)} = \mathcal{P}^{\Gamma'_p}_{p:j}$.

Proof. By the definition of **opred** for each $\underset{con}{\longrightarrow}$ -transition rule, the annotation map updates for each such rule, and the definitions of local consistency, communicative consistency, and κ-coherence. ∎

Lemma 6 is the key lemma in our proofs of invariance properties. The following two lemmas will also be useful.

Lemma 7 *Let* Π *be a thread map with a locally consistent, communicative consistent, and* κ*-coherent family of annotation maps* $\mathcal{G}$. *Suppose that* $\Pi \underset{con}{\longrightarrow} \Pi'$ *and there is an updated family of annotation maps for* Π'. *For any occurrence* i *from* Π', *and for any symbol* $s \in \Sigma$ *such that* $i.s$ *is also an occurrence in* Π', $\mathcal{P}_{i.s} \vec{\leq} \mathcal{P}_{\textbf{opred}(i).s}$.

Proof. By Lemma 6, local consistency, and the specifications of annotation map updates.

First assume that $\mathbf{opred}(i).s$ is not a substitution site. By Lemma 5, $\mathbf{opred}(i).s = \mathbf{opred}(i.s)$, so also $\mathcal{P}_{\mathbf{opred}(i).s} = \mathcal{P}_{\mathbf{opred}(i.s)}$. Suppose $\mathbf{opred}(i.s)$ is not a value substitution, then by Lemma 6(5), $\mathcal{P}_{\mathbf{opred}(i.s)} = \mathcal{P}_{i.s}$, so $\mathcal{P}_{i.s} = \mathcal{P}_{\mathbf{opred}(i).s}$, hence $\mathcal{P}_{i.s} \overset{\rightarrow}{\leq} \mathcal{P}_{\mathbf{opred}(i).s}$. Assume instead $\mathbf{opred}(i.s) = \mathbf{opred}(i).s$ is a value substitution. We shall show that this assumption leads to a contradiction. Then either (1) the transition is a $\mathbf{seq}/\beta$ transition, and $\mathbf{opred}(i)$ is the reduced application, or (2) the transition is a **comm** transition, and $\mathbf{opred}(i)$ is the communicating `send()`. But by the definitions of **opred** for $\mathbf{seq}/\beta$ and **comm** transitions, there is no right-hand side occurrence i such that $\mathbf{opred}(i)$ is the reduced application (in the case of (1)) or the communicating `send()` (in the case of (2)).

Now suppose $\mathbf{opred}(i).s$ is the site of a substitution. At the substitution site, either (1) the annotation map is unchanged, or tree merger is used, so that the annotation at the substitution site is unchanged, or (2) tree replacement is used, so that the annotation at the substitution site is the annotation of the substituted occurrence. In case (1) we have $\mathcal{P}_{i.s} = \mathcal{P}_{\mathbf{opred}(i).s}$; in case (2) we have $\mathcal{P}_{i.s} \overset{\rightarrow}{\leq} \mathcal{P}_{\mathbf{opred}(i).s}$, by the local consistency of $\mathcal{G}$. To verify the second of these, the reader may check that in all cases where the update is specified using tree replacement, there is a constraint that the proposition at the root of the substituted tree is $\overset{\rightarrow}{\leq}$ the proposition at the substitution site. ∎

Lemma 8 *Let* Π *be a thread map with a locally consistent, communicative consistent, and* κ*-coherent family of annotation maps* $\mathcal{G}$*. Suppose that* $\Pi \xrightarrow[con]{} \Pi'$ *and there is an updated family of annotation maps for* Π'*. For any occurrence* k *from* Π'*, and for any symbol* $s \in \Sigma$ *such that* $k.s$ *is also an occurrence in* Π'*, if either (1)* $\mathbf{opred}(k.s)$ *is a value substitution, or (2)* $\mathbf{opred}(k).s$ *is not the site of a substitution, then* $\mathcal{P}_{k.s} = \mathcal{P}_{\mathbf{opred}(k).s}$.

Proof. By the specifications of annotation map updates.

(1): In the specifications of annotation map updates for $\mathbf{seq}/\beta$ and **comm** transitions, the use of tree merger preserves the annotation at the site $\mathbf{opred}(k).s$ where $\mathbf{opred}(k.s)$ is substituted.

(2): By Lemma 5, $\mathbf{opred}(k).s = \mathbf{opred}(k.s)$, hence $\mathcal{P}_{\mathbf{opred}(k).s} = \mathcal{P}_{\mathbf{opred}(k.s)}$. Since $\mathbf{opred}(k).s$ is not the site of a substitution, $\mathbf{opred}(k.s)$ must not be a value substitution. By Lemma 6(5), $\mathcal{P}_{k.s} = \mathcal{P}_{\mathbf{opred}(k.s)}$. Hence, $\mathcal{P}_{k.s} = \mathcal{P}_{\mathbf{opred}(k).s}$. ∎

Another useful result is:

Lemma 9 *Let* $\mathcal{G}$ *be a locally consistent, communicative consistent, and* κ*-coherent family of annotation maps for a thread map* Π*, and suppose* $\Pi \xrightarrow[con]{} \Pi'$*. Let* $\mathcal{G}'$

be the updated family of annotation maps for Π'. Suppose also that $p : i$ is an occurrence of a `send()` *and $q : j$ is an occurrence of a* `receive()` *in Π' that are possible communications partners. Then in Π, $\mathbf{opred}(p : i)$ and $\mathbf{opred}(q : j)$ were occurrences of a* `send()` *and a* `receive()` *that were possible communications partners.*

Proof. By Definition 2 and Lemma 6.

Suppose $p : i$ and $q : j$ are possible communications partners. Then in $\mathcal{G}'$, by Definition 2, $\kappa_{p:i.schan} \cap \kappa_{q:j.rchan} \neq \emptyset$. By Lemma 6(1), $\mathbf{opred}(p : i)$ is also an occurrence of a `send()`, and $\mathbf{opred}(q : j)$ is also an occurrence of a `receive()`. By Lemma 7, $\mathcal{P}_{p:i.schan} \overset{\rightarrow}{\leq} \mathcal{P}_{\mathbf{opred}(p\,:\,i).schan}$ and $\mathcal{P}_{q:j.rchan} \overset{\rightarrow}{\leq} \mathcal{P}_{\mathbf{opred}(q\,:\,j).rchan}$. Therefore, $\kappa_{\mathbf{opred}(p\,:\,i).schan} \cap \kappa_{\mathbf{opred}(q\,:\,j).rchan} \neq \emptyset$. By Definition 2, $\mathbf{opred}(p : i)$ and $\mathbf{opred}(q : j)$ were possible communications partners. ∎

11 Semantics of propositions

Recall, the goal of our analysis is to determine which occcurrences of `channel()` produce channels that are used only locally, meaning that those channels participate in communications only on the processor on which they are created. Therefore, the meaning we give to propositions has an operational basis.

We need the following:

Definition 5 *Suppose for some $n \geq 0$, $\Pi \underset{con}{\longrightarrow}^n \Pi'$. An occurrence $q : j$ in Π' is* traceable to *an occurrence $p : i$ in Π, iff $\mathbf{opred}^n(q : j) = p : i$.*

Also:

Definition 6 *Suppose $\Pi \underset{con}{\longrightarrow}^+ \Pi'$. Let $p : i$ be an occurrence of* `channel()` *in Π. An occurrence $q : j$ of a channel constant k in Π' is* produced by *$p : i$ iff for some Π'' intermediate in the evaluation*

- *there exists a* `channel()` *occurrence $r : m$ in Π'' that is traceable to $p : i$ in Π,*
- *in one* **channel** *step involving $r : m$, $\Pi'' \underset{con}{\longrightarrow} \Pi'''$, so that $r : m$ is also the occurrence index of k in Π''', and*
- *$q : j$ in Π' is traceable to $r : m$ in Π'''.*

We say that $q : j$ in Π' is directly produced *by $r : m$ in Π''.*

The picture is:

$$\Pi \xrightarrow[con]{}^{*} \Pi'' \underbrace{\xrightarrow[con]{}}_{\textbf{channel}} \Pi''' \xrightarrow[con]{}^{*} \Pi'$$

Now we give our notion of satisfaction:

Definition 7 *Let Π be a thread map, and let $\mathcal{G}$ be a family of annotation maps, with one family member for each thread identifier in the domain of Π. Say that an evaluation $\Pi \xrightarrow[con]{}^{*} \Pi'$ satisfies $\mathcal{G}$, written*

$$\Pi \xrightarrow[con]{}^{*} \Pi' \models \mathcal{G}$$

iff for any `channel()` *occurrence $p : i$ in Π, if $\omega_{p:i} = \mathbf{local}^{+}$, then for any intermediate Π'' in the evaluation, including Π', for any occurrence $s : n$ of a channel constant in Π'' produced by $p : i$ in Π, such that $s : n$ is a channel-part subterm of either a* `send()` *or a* `receive()` *in Π'', and such that $s : n$ was directly produced by an occurrence $r : m$ in a thread map Π''' preceding Π'' in the evaluation, we have*

$$\mathbf{proc}_{\Pi''}(s) = \mathbf{proc}_{\Pi'''}(r)$$

We will use this notion of satisfaction as the criterion of the soundness of our analysis. Note that this semantics ignores all but the ω component of propositions. While we could have mentioned the other components in our semantics, since they do provide useful information about annotated programs, we do not need to do so for our local channel analysis.

12 Invariance results

If we use the annotation map updates specified, then local consistency, communicative consistency, κ-coherence, and ω-coherence are maintained across $\xrightarrow[con]{}$-transitions.

The invariance of κ-coherence is the easiest of these results:

Theorem 1 *Let Π be a thread map with a locally consistent, communicative consistent, and κ-coherent family of annotation maps $\mathcal{G}$. Suppose $\Pi \xrightarrow[con]{} \Pi'$, and let $\mathcal{G}'$ be the updated family of annotation maps for Π'. Then $\mathcal{G}'$ is κ-coherent.*

Proof. By Lemma 6.

Suppose i and j are distinct occurrences of the same channel constant on the right-hand side.

We note that $\mathbf{opred}(i)$ and $\mathbf{opred}(j)$ are both defined. Assume that, say, $\mathbf{opred}(i)$ is undefined, but $\mathbf{opred}(j)$ is defined. The reverse case is similar. Then i must be an occurrence of a fresh channel constant. By Lemma 6(1), $\mathbf{opred}(j)$ is also a channel constant. By Lemma 6(3), $[\![\mathbf{opred}(j)]\!] = [\![j]\!]$. Since we assumed $[\![i]\!] = [\![j]\!]$, i is not fresh, a contradiction. Assume now that both $\mathbf{opred}(i)$ and $\mathbf{opred}(j)$ are undefined. Then i and j are both occurrences of a fresh channel constant. But we assumed $[\![i]\!] = [\![j]\!]$, and the $\underset{con}{\longrightarrow}$ rules provide no way to create more than one occurrence of a fresh channel constant.

By Lemma 6(1), $\mathbf{opred}(i)$ and $\mathbf{opred}(j)$ are both channel constants. By Lemma 6(3), $[\![\mathbf{opred}(i)]\!] = [\![i]\!]$ and $[\![\mathbf{opred}(j)]\!] = [\![j]\!]$, so $[\![\mathbf{opred}(i)]\!] = [\![\mathbf{opred}(j)]\!]$. Therefore, by the κ-coherence of $\mathcal{G}$, $\kappa_{\mathbf{opred}(i)} \cap \kappa_{\mathbf{opred}(j)} \neq \emptyset$. But by Lemma 6(4), $\kappa_{\mathbf{opred}(i)} \subseteq \kappa_i$ and $\kappa_{\mathbf{opred}(j)} \subseteq \kappa_j$, so $\kappa_i \cap \kappa_j \neq \emptyset$. ∎

Now we give our more significant invariance results.

Theorem 2 *Let Π be a thread map with a family of annotation maps $\mathcal{G}$ that is locally consistent, communicative consistent, and κ-coherent. Suppose $\Pi \underset{con}{\longrightarrow} \Pi'$. Then the updated family of annotation maps $\mathcal{G}'$ is locally consistent for Π'.*

Proof. By considering each possible $\underset{con}{\longrightarrow}$-transition and $\mathcal{G}'$, the updated family of annotation maps.

For each possible $\underset{con}{\longrightarrow}$-transition, we check the local consistency of the annotation map or maps at each occurrence j on the right-hand side from those threads which change from the left-hand side. Assume each transition is as in Figures 6 to 12.

case seq/β

Let Γ be the left-hand side annotation map for the thread p; let Γ' be the updated right-hand side annotation map for the same thread.

Suppose $\mathit{Var}(j)$.

The only constraint is that if k is the binding occurrence for j, then $\mathcal{P}_k^{\Gamma'} \overset{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'}$. If j has no binding occurrence, we are done. Otherwise, assume k is the binding occurrence for j. By Lemma 6(1), $\mathit{Var}(\mathbf{opred}(j))$. Since substitution does not cause variable capture, $\mathbf{opred}(k)$ must have been the binding occurrence for $\mathbf{opred}(j)$. By the local consistency of Γ, $\mathcal{P}^{\Gamma}_{\mathbf{opred}(k)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j)}$. Now, neither $\mathbf{opred}(k)$ nor $\mathbf{opred}(j)$ is a value, so neither was a value substitution. By Lemma 6(5), $\mathcal{P}_k^{\Gamma'} = \mathcal{P}^{\Gamma}_{\mathbf{opred}(k)}$ and $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} = \mathcal{P}_j^{\Gamma'}$. Hence, $\mathcal{P}_k^{\Gamma'} \overset{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'}$.

Suppose $\mathit{Channel}(j)$.

By Lemma 6(1), also $\mathit{Channel}(\mathbf{opred}(j))$. Therefore, by the local consistency of Γ, $\mathbf{lab}(\mathbf{opred}(j)) \in \kappa^{\Gamma}_{\mathbf{opred}(j)}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} \overset{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'}$, so $\kappa^{\Gamma}_{\mathbf{opred}(j)} \subseteq$

$\kappa_j^{\Gamma'}$. Therefore, $\mathbf{lab}(\mathbf{opred}(j)) \in \kappa_j^{\Gamma'}$. By Lemma 6(2), $\mathbf{lab}(j) = \mathbf{lab}(\mathbf{opred}(j))$. Hence $\mathbf{lab}(j) \in \kappa_j^{\Gamma'}$.

Suppose $\mathit{Fun}(j)$.

The first constraint is that $\mathbf{lab}(j) \in \phi_j^{\Gamma'}$. The analysis follows that for `channel()` labels.

The next constraint is, if the number of free occurrences of the procedure binder in the procedure body is greater than 1, then $\nu_{j.bv}^{\Gamma'} = \infty$. Suppose the condition holds. By Lemma 6(1), also $\mathit{Fun}(\mathbf{opred}(j))$. Now, suppose β-reduction did not substitute the application operand in $[\![\mathbf{opred}(j)]\!]$. Then $[\![\mathbf{opred}(j)]\!] = [\![j]\!]$, so in $\mathbf{opred}(j)$, the number of free occurrences of the procedure binder in the procedure body is also greater than 1. So by the local consistency of Γ, $\nu_{\mathbf{opred}(j).bv}^{\Gamma} = \infty$. By Lemma 7, $\mathcal{P}_{j.bv}^{\Gamma'} \vec{\leq} \mathcal{P}_{\mathbf{opred}(j).bv}^{\Gamma}$. By the $\vec{\leq}$-order on propositions, also $\nu_{j.bv}^{\Gamma'} = \infty$. Suppose instead that β-reduction did substitute the application operand in $[\![\mathbf{opred}(j)]\!]$. Then $[\![j]\!]$ is a substitution instance of $[\![\mathbf{opred}(j)]\!]$. It is easy to show that in $\mathbf{opred}(j)$, the number of free occurrences of its binder in its body is the same as for j; so the result follows as before.

The escape constraints are triggered in case $\theta_j^{\Gamma'} = \mathbf{escape}^{\pm}$. Assume that condition holds, else we are done. By Lemma 6, $\mathcal{P}_{\mathbf{opred}(j)}^{\Gamma} \vec{\leq} \mathcal{P}_j^{\Gamma'}$, so by the $\vec{\leq}$-order on propositions, $\theta_j^{\Gamma'} \vec{\leq} \theta_{\mathbf{opred}(j)}^{\Gamma}$. Therefore, $\theta_{\mathbf{opred}(j)}^{\Gamma} = \mathbf{escape}^{\pm}$, so the escape constraints held in Γ.

We now examine each of the escape constraints.

Let $j.k$ be a free variable occurrence in j. We want to show two things, that $\theta_{j.k}^{\Gamma'} = \mathbf{escape}^{\pm}$, and that if $\sigma_{j.k}^{\Gamma'} = \mathbf{reach}^{\pm}$, then $\omega_{j.k}^{\Gamma'} = \mathbf{local}^{\pm}$. For the latter constraint, assume the condition holds.

By Lemma 6(1), also $\mathit{Var}(\mathbf{opred}(j.k))$. Since $j.k$ is a subterm of j, $\mathbf{opred}(j.k)$ may have been a subterm of $\mathbf{opred}(j)$, that is, $\mathbf{opred}(j.k) = \mathbf{opred}(j).k$, but it may instead be that $\mathbf{opred}(j.k)$ was a subterm of the substituted application operand.

First suppose that $\mathbf{opred}(j.k) = \mathbf{opred}(j).k$; then $\mathbf{opred}(j.k)$ was also a free variable occurrence in $\mathbf{opred}(j)$. By the local consistency of Γ, $\theta_{\mathbf{opred}(j.k)}^{\Gamma} = \mathbf{escape}^{\pm}$, and, if $\sigma_{\mathbf{opred}(j.k)}^{\Gamma} = \mathbf{reach}^{\pm}$, then $\omega_{\mathbf{opred}(j.k)}^{\Gamma} = \mathbf{local}^{\pm}$. Since it is not a value, $\mathbf{opred}(j.k)$ cannot have been a value substitution. So by Lemma 6(5), $\mathcal{P}_{\mathbf{opred}(j.k)}^{\Gamma} = \mathcal{P}_{j.k}^{\Gamma'}$, and $\theta_{j.k}^{\Gamma'} = \mathbf{escape}^{\pm}$, as desired. Under our assumption that $\sigma_{j.k}^{\Gamma'} = \mathbf{reach}^{\pm}$, also $\sigma_{\mathbf{opred}(j.k)}^{\Gamma} = \mathbf{reach}^{\pm}$. Therefore, by the local consistency of Γ, $\omega_{\mathbf{opred}(j.k)}^{\Gamma} = \mathbf{local}^{\pm}$. Again using Lemma 6(5), $\omega_{j.k}^{\Gamma'} = \mathbf{local}^{\pm}$.

Suppose instead that $\mathbf{opred}(j.k)$ was a subterm of the substituted operand. The operand was substituted for an occurrence of a free variable subterm of $\mathbf{opred}(j)$, say $\mathbf{opred}(j).m$. By the local consistency of Γ, $\theta_{\mathbf{opred}(j).m}^{\Gamma} = \mathbf{escape}^{\pm}$.

Now consider the substituted operand $i.rand$. Since the operand contained a variable subterm, it must have been a function. We can show that $\mathcal{P}^{\Gamma}_{i.rand} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).m}$ (we omit the straightforward details). Therefore, $\theta^{\Gamma}_{i.rand} = \mathbf{escape}^{\pm}$. Since $\mathbf{opred}(j.k)$ was a free variable occurrence in $i.rand$, from here, the analysis is the same as in the case that $\mathbf{opred}(j.k)$ was a subterm of $\mathbf{opred}(j)$.

Next, we wish to show that for all channel constant occurrences $j.m$, if $\sigma^{\Gamma'}_{j.m} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'}_{j.m} = \mathbf{local}^{\pm}$. Observe that a channel constant is a value, so in contrast to a variable, it may be a value substitution. If $\mathbf{opred}(j.m)$ is not a value substitution, the analysis is identical to the free variable case for the comparable constraint. If $\mathbf{opred}(j.m)$ is a value substitution, then it must have been the application operand $i.rand$, and it was substituted for some free variable in $\mathbf{opred}(j)$. Observe that $\mathbf{opred}(j).m$ must have been the index of the substituted free variable. By Lemma 8(1), $\mathcal{P}^{\Gamma'}_{j.m} = \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).m}$. So assume the condition in the constraint holds. Therefore, $\sigma^{\Gamma}_{\mathbf{opred}(j).m} = \mathbf{reach}^{\pm}$. Since $\mathbf{opred}(j).m$ was a free variable in $\mathbf{opred}(j)$, by the local consistency of Γ, $\omega^{\Gamma}_{\mathbf{opred}(j).m} = \mathbf{local}^{\pm}$. Therefore, $\omega^{\Gamma'}_{j.m} = \mathbf{local}^{\pm}$.

We now want to show that Γ' is remote-expectant at j. Consider any pair of possible communications partners in the p thread such that exactly one of the pair is a subterm of j. We want to show that in Γ', the body of the `send()` occurrence has a θ annotation of $\mathbf{escape}^{\pm}$, and that if the σ annotation of the `receive()` occurrence is $\mathbf{reach}^{\pm}$, then its ω annotation is $\mathbf{local}^{\pm}$. By Lemma 9, the occurrence predecessors of the pair were possible communications partners.

Suppose it is the `send()` that is a subterm of j, and the `receive()` is not. The reverse situation is similar, and we will not analyze it further. Let these be $j.k$ and m, respectively. Now, $\mathbf{opred}(j.k)$ was a subterm of either $\mathbf{opred}(j)$ or the substituted operand, $i.rand$. In the first case, $\mathbf{opred}(j)$ had a θ-annotation of $\mathbf{escape}^{\pm}$, so Γ was remote-expectant at $\mathbf{opred}(j)$. In the second case, we can show, as we did for free variable subterms, that $i.rand$ had a θ-annotation of $\mathbf{escape}^{\pm}$, so Γ was remote-expectant at $i.rand$. We will look just at the first case.

Consider $\mathbf{opred}(m)$. If $\mathbf{opred}(m)$ was not a subterm of $\mathbf{opred}(j)$, then by the remote-expectancy of Γ, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j.k).sbody}$, hence $\theta^{\Gamma'}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Now assume that $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(m)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{m}$. By the $\stackrel{\rightarrow}{\leq}$-order on propositions, $\sigma^{\Gamma'}_{m} \leq \sigma^{\Gamma}_{\mathbf{opred}(m)}$, so $\sigma^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$. So by the local consistency of Γ, $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. Since a `receive()` cannot be a value substitution, by Lemma 6(5), also $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$.

But it may have been that $\mathbf{opred}(m)$ was also a subterm of $\mathbf{opred}(j)$. In that case, it must have been that the β-reduction duplicated the operand $i.rand$, and $\mathbf{opred}(m)$ was a subterm of $i.rand$. In fact, in that case, $\mathbf{opred}(j)$ was either the same as $i.rand$, or a proper subterm of it. To see this, note that $\mathbf{opred}(m)$ was a subterm of both $i.rand$ and $\mathbf{opred}(j)$, hence one must have been a subterm of the other; but reduction cannot occur in a function body, hence $i.rand$ cannot have been a proper subterm of $\mathbf{opred}(j)$. The β-reduction duplicates $i.rand$, so the number of free occurrences of the binder of the application operator, $[\![i.bv]\!]$, in its body, $[\![i.body]\!]$, must have been greater than 1. Hence, by the local consistency of Γ, $\nu^{\Gamma}_{i.bv} = \infty$. Also by local consistency, we can show that $\nu^{\Gamma}_{i.rand} = \infty$; since $\mathbf{opred}(j)$ is either the same as $i.rand$, or a function subterm of $i.rand$, $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$. By the nonlinear-expectancy of Γ at $\mathbf{opred}(j)$, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. So by Lemma 7, also $\theta^{\Gamma'}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Again assume that $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$, so $\sigma^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$, and again by the nonlinear-expectancy of Γ at $\mathbf{opred}(j)$, $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. Again by Lemma 6(5), $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$.

Next we show the nonlinear-expectancy of Γ' at j. Assume $\nu^{\Gamma'}_{j} = \infty$, otherwise we are done. By Lemma 6(4), also $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$. We have already shown that Γ was nonlinear-expectant at $\mathbf{opred}(j)$. Consider a `send()` occurrence $j.k$ and a `receive()` occurrence m that are possible communications partners, and also both subterms of j. We wish to show that $\theta^{\Gamma'}_{j.k.sbody} = \mathbf{escape}^{\pm}$ and, if $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$. By Lemma 9, $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ were possible communications partners. On the left-hand side, $\mathbf{opred}(j.k)$ was either a subterm of $\mathbf{opred}(j)$ or a subterm of $i.rand$; similarly for $\mathbf{opred}(m)$. We consider each of the four possibilities. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j.k).sbody}$, so for each of the four cases, it suffices to prove that $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. We assume that $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(m)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{m}$, so by the $\stackrel{\rightarrow}{\leq}$-order on propositions, also $\sigma^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$. Since a `receive()` cannot be a value substitution, by Lemma 6(5), $\omega^{\Gamma'}_{m} = \omega^{\Gamma}_{\mathbf{opred}(m)}$. Therefore, for each of the four cases, it suffices to prove that $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$.

1. Suppose both $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ were subterms of $\mathbf{opred}(j)$. By the nonlinear-expectancy of Γ at $\mathbf{opred}(j)$, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$ and $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$.

2. If both $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ were subterms of $i.rand$, and neither was a subterm of $\mathbf{opred}(j)$, then the operand must have been substituted for a free variable in $\mathbf{opred}(j)$. Since $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$, also that free variable occurrence must have had a ν-annotation of ∞; since $\theta^{\Gamma}_{\mathbf{opred}(j)} = \mathbf{escape}^{\pm}$, the free variable occurrence also had a θ-annotation of $\mathbf{escape}^{\pm}$. It is straightforward to show that $\nu^{\Gamma}_{i.rand} = \infty$ and $\theta^{\Gamma}_{i.rand} = \mathbf{escape}^{\pm}$. Therefore, Γ was nonlinear-expectant at $i.rand$, so $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$ and $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$.

3. Suppose $\mathbf{opred}(j.k)$ was a subterm of $\mathbf{opred}(j)$, and not a subterm of $i.rand$, but $\mathbf{opred}(m)$ was a subterm of $i.rand$, and not a subterm of $\mathbf{opred}(j)$. Then by the remote-expectancy of Γ at j, we have $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$

and $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$.

4. Suppose $\mathbf{opred}(j.k)$ was a subterm of $i.rand$, and not a subterm of $\mathbf{opred}(j)$, but $\mathbf{opred}(m)$ was a subterm of $\mathbf{opred}(j)$, and not a subterm of $i.rand$. In this case, too, the operand must have been substituted for a free variable in $\mathbf{opred}(j)$, and by the local consistency of Γ, such a free variable occurrence had a θ-annotation of $\mathbf{escape}^{\pm}$. Therefore, $\theta^{\Gamma}_{i.rand} = \mathbf{escape}^{\pm}$, so Γ was remote-expectant at $i.rand$. By the definition of remote-expectancy, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$ and $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$.

Finally for the $Fun(j)$ subcase, we have the constraint that if $\nu^{\Gamma'}_{j} = \infty$, then all free variable and function occurrence subterms of j have a ν-annotation of ∞. Suppose the condition holds. By Lemma 6(4) and the $\vec{\leq}$-order on propositions, also $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$. We shall soon deal with free variable subterms of j, but consider first a function subterm of j, say $j.k$. $\mathbf{opred}(j.k)$ may have been a subterm of $\mathbf{opred}(j)$, or it may have been a subterm of $i.rand$. In the first case, by the local consistency of Γ, $\nu^{\Gamma}_{\mathbf{opred}(j.k)} = \infty$. Such a subterm was not a value substitution. Therefore, by Lemma 6(5), $\nu^{\Gamma'}_{j.k} = \infty$.

In the second case, it must be that the operand was substituted for a free variable occurrence in $\mathbf{opred}(j)$. By the local consistency of Γ, that free variable occurrence had a ν-annotation of ∞, hence $\nu^{\Gamma}_{i.rand} = \infty$ (we omit the details). $\mathbf{opred}(j.k)$ may have been the operand itself, or a proper subterm of the operand. If $\mathbf{opred}(j.k) = i.rand$, then $\mathbf{opred}(j.k)$ was a value substitution, so by Lemma 8(1), $\mathcal{P}^{\Gamma'}_{j.k} = \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).k}$. Now, $\mathbf{opred}(j).k$ was the substituted free variable, which, as we stated, had a ν-annotation of ∞. So also $\nu^{\Gamma'}_{j.k} = \infty$. Otherwise, $\mathbf{opred}(j.k)$ is a proper subterm of $i.rand$, and so not a value substitution. The only kind of value that may have proper subterms is a function. By the local consistency of Γ, such a function subterm of $i.rand$ has a ν-annotation of ∞. So by Lemma 6(5), $\nu^{\Gamma'}_{j.k} = \nu^{\Gamma}_{\mathbf{opred}(j.k)} = \infty$.

Now consider a free variable subterm of j, say $j.m$. We want to show that $j.m$ has a ν-annotation of ∞. $\mathbf{opred}(j.m)$ may have been a subterm of $\mathbf{opred}(j)$, or it may have been a subterm of $i.rand$. In the first case, $\mathbf{opred}(j.m)$ was also free in $\mathbf{opred}(j)$, so by the local consistency of Γ, $\nu^{\Gamma}_{\mathbf{opred}(j.m)} = \infty$. By Lemma 6(5), $\nu^{\Gamma'}_{j.m} = \infty$. In the second case, we can show that $\nu^{\Gamma}_{i.rand} = \infty$. $\mathbf{opred}(j.m)$ must have been a proper subterm of the operand and so not a value substitution. $\mathbf{opred}(j.m)$ was also a free variable subterm of $i.rand$. By the local consistency of Γ, $\mathbf{opred}(j.m)$ had a ν-annotation of ∞. By Lemma 6(5), $\nu^{\Gamma'}_{j.m} = \nu^{\Gamma}_{\mathbf{opred}(j.m)} = \infty$.

Suppose $App(j)$.

We have the constraints that for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma'}_{j.rator}$, $\mathcal{P}^{\Gamma'}_{j.rand} \vec{\leq} \mathcal{P}^{\Gamma'}_{k.bv}$ and $\mathcal{P}^{\Gamma'}_{k.body} \vec{\leq} \mathcal{P}^{\Gamma'}_{j}$. Since $App(j)$, also $App(\mathbf{opred}(j))$, by Lemma 6(1). Therefore, by the local consistency of Γ, for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma}_{\mathbf{opred}(j).rator}$,

$\mathcal{P}^{\Gamma}_{\mathbf{opred}(j).rand} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{k.bv}$, and $\mathcal{P}^{\Gamma}_{k.body} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j)}$.

By Lemma 7, $\phi^{\Gamma'}_{j.rator} \subseteq \phi^{\Gamma}_{\mathbf{opred}(j).rator}$. Consider any function occurrence k from the right-hand side term such that $\mathbf{lab}(k) \in \phi^{\Gamma'}_{j.rator}$. Since $\mathbf{lab}(\mathbf{opred}(k)) = \mathbf{lab}(k)$, $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma}_{\mathbf{opred}(j).rator}$. By the local consistency of Γ, $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma}_{\mathbf{opred}(j).rator}$ implies $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j).rand} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(k).bv}$ and $\mathcal{P}^{\Gamma}_{\mathbf{opred}(k).body} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j)}$. Since $\mathbf{opred}(k).bv$ cannot have been the site of a substitution, by Lemma 8(2), $\mathcal{P}^{\Gamma'}_{k.bv} = \mathcal{P}^{\Gamma}_{\mathbf{opred}(k).bv}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.rand} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).rand}$. Therefore, $\mathcal{P}^{\Gamma'}_{j.rand} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{k.bv}$. Also by Lemma 7, $\mathcal{P}^{\Gamma'}_{k.body} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(k).body}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$. Hence, $\mathcal{P}^{\Gamma'}_{k.body} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$.

Suppose *Cond*(j).

By Lemma 6(1), also *Cond*$(\mathbf{opred}(j))$. By the local consistency of Γ, $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j).then}$, $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j).else} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.then} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).then}$, and $\mathcal{P}^{\Gamma'}_{j.else} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).else}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$. Hence $\mathcal{P}^{\Gamma'}_{j.then}$, $\mathcal{P}^{\Gamma'}_{j.else} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$.

Suppose *RFork*(j).

In this case, for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma'}_{j.rffun}$, we require that the escape constraints hold at k. Consider such a k. By Lemma 6(2), $\mathbf{lab}(k) = \mathbf{lab}(\mathbf{opred}(k))$. By Lemma 6(1), also *RFork*$(\mathbf{opred}(j))$. By Lemma 7, $\phi^{\Gamma'}_{j.rffun} \stackrel{\rightarrow}{\leq} \phi^{\Gamma}_{\mathbf{opred}(j).rffun}$. Hence, $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma}_{\mathbf{opred}(j).rffun}$. Therefore, in Γ, the escape constraints held at $\mathbf{opred}(k)$.

Now consider the reasoning we used in the *Fun* subcase to show that the escape constraints held at function j' in Γ'. We began by assuming the condition $\theta^{\Gamma'}_{j'} = \mathbf{escape}^{\pm}$. Using Lemma 6(4), we had that $\theta^{\Gamma}_{\mathbf{opred}(j')} = \mathbf{escape}^{\pm}$. By local consistency, the escape constraints held in Γ at $\mathbf{opred}(j')$. That was sufficient to show that the escape constraints held in Γ' at j'. Here, for the *RFork* subcase, we have shown that the escape constraints hold in Γ at $\mathbf{opred}(k)$, using the local consistency of Γ at $\mathbf{opred}(j)$. Therefore, the same reasoning we used in the *Fun* subcase can be used to show that the escape constraints hold in Γ' at k.

Suppose *Send*(j).

The only constraint is that $\sigma^{\Gamma'}_{j.schan} = \mathbf{reach}^{\pm}$. By Lemma 6(1), also *Send*$(\mathbf{opred}(j))$. Since Γ is locally consistent, we have $\sigma^{\Gamma}_{\mathbf{opred}(j).schan} = \mathbf{reach}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.schan} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).schan}$. By the $\stackrel{\rightarrow}{\leq}$-order on propositions, $\sigma^{\Gamma'}_{j.schan} = \mathbf{reach}^{\pm}$.

Suppose *Receive*(j).

The first constraint is that $\sigma^{\Gamma'}_{j.rchan} = \mathbf{reach}^{\pm}$. We can use the similar reasoning as in the *Send* subcase for the constraint on σ annotations.

The other constraint is that for all k such that $Send(k)$ and $\kappa^{\Gamma'}_{k.schan} \cap \kappa^{\Gamma'}_{j.rchan} \neq \emptyset$, $\mathcal{P}^{\Gamma'}_{k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$. Choose any such k. By Lemma 9, $\mathbf{opred}(j)$ and $\mathbf{opred}(k)$ were possible communications partners on the left-hand side. Also, both those occurrences were in the p thread on the left-hand side. So by the local consistency of Γ, $\mathcal{P}^{\Gamma}_{\mathbf{opred}(k).sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(k).sbody}$; by Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$. So $\mathcal{P}^{\Gamma'}_{k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$.

case seq/cond-true

We shall omit the proof of local consistency for the **seq**/cond-false case, which is similar. As in the case for a **seq**/β transition, we omit the thread identifier subscripts on annotation maps; let Γ be the left-hand side annotation map for the thread p, and let Γ' the annotation map for the right-hand side thread p.

Suppose $Var(j)$.

The one constraint is that if k is the binding occurrence for j, then $\mathcal{P}^{\Gamma'}_{k} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$. Use the same reasoning as in the **seq**/β case.

Suppose $Channel(j)$.

The only constraint is that $\mathbf{lab}(j) \in \phi^{\Gamma'}_{j}$. Use the same reasoning as in the **seq**/β case.

Suppose $Fun(j)$.

The first constraint is that $\mathbf{lab}(j) \in \phi^{\Gamma'}_{j}$. This can be shown using the same reasoning as in the **seq**/β case for `channel()` labels.

Next, we have the constraint that if the number of free occurrences of the procedure binder in the procedure body is greater than 1, then $\nu^{\Gamma'}_{j} = \infty$. A conditional redex cannot occur in a function body, so $[\![j]\!] = [\![\mathbf{opred}(j)]\!]$. Therefore, in $\mathbf{opred}(j)$, the number of free occurrences of its binder in its body is the same as for j. Assume the condition holds. By the local consistency of Γ, $\nu^{\Gamma}_{\mathbf{opred}(j).bv} = \infty$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.bv} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).bv}$, so also $\nu^{\Gamma'}_{j.bv} = \infty$.

Next we consider the escape constraints that are triggered if $\theta^{\Gamma'}_{j} = \mathbf{escape}^{\pm}$. Assume the condition holds. By Lemma 6(1), also $Fun(\mathbf{opred}(j))$; by Lemma 6(5), also $\theta^{\Gamma}_{\mathbf{opred}(j)} = \mathbf{escape}^{\pm}$.

The first constraints triggered by the condition are that for all free variable occurrences $j.k$, $\theta^{\Gamma'}_{j.k} = \mathbf{escape}^{\pm}$, and, if $\sigma^{\Gamma'}_{j.k} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'}_{j.k} = \mathbf{local}^{\pm}$. Assume the condition holds for the latter constraint. Observe that $\mathbf{opred}(j.k)$ must have been a free variable occurrence in $\mathbf{opred}(j)$. So by the local consistency of Γ, $\theta^{\Gamma}_{\mathbf{opred}(j.k)} = \mathbf{escape}^{\pm}$. Since $\mathbf{opred}(j.k)$ is not a value, it cannot have been a value substitution. By Lemma 6(5), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j.k)} = \mathcal{P}^{\Gamma'}_{j.k}$, so $\theta^{\Gamma'}_{j.k} = \theta^{\Gamma}_{\mathbf{opred}(j.k)}$.

Therefore, $\theta^{\Gamma'}_{j.k} = \mathbf{escape}^{\pm}$. Similarly, $\sigma^{\Gamma}_{\mathbf{opred}(j.k)} = \mathbf{reach}^{\pm}$, so by local consistency, $\omega^{\Gamma}_{\mathbf{opred}(j.k)} = \mathbf{local}^{\pm}$. So also $\omega^{\Gamma'}_{j.k} = \mathbf{local}^{\pm}$.

Next we consider the constraint that for any channel constant occurrence $j.k$, if $\sigma^{\Gamma'}_{j.k} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'}_{j.k} = \mathbf{local}^{\pm}$. We can use almost the same reasoning as we just used for free variables. Although channel constants are values, unlike variables, $\mathbf{opred}(j.k)$ cannot have been a value substitution. For a **seq**/cond-true transition, no left-hand side occurrence is a value substitution.

Next we show that Γ' is remote-expectant at j. Consider any pair of possible communications partners such that exactly one of the pair is a subterm of j. We want to show that in Γ', the body of the `send()` occurrence has a θ annotation of $\mathbf{escape}^{\pm}$, and that if the σ annotation of the `receive()` occurrence is $\mathbf{reach}^{\pm}$, then its ω annotation is $\mathbf{local}^{\pm}$. By Lemma 9, the occurrence predecessors of the pair were possible communications partners. Both occurrence predecessors were in the p thread on the left-hand side.

Suppose it is the `send()` that is a subterm of j, and the `receive()` is not. As in the **seq**/β case, the reverse situation is similar, so we will not discuss it. Let these occurrences be $j.k$ and m, respectively. $\mathbf{opred}(j.k)$ must have been a subterm of $\mathbf{opred}(j)$. Now, $\mathbf{opred}(j)$ had a θ-annotation of $\mathbf{escape}^{\pm}$, so Γ was remote-expectant at $\mathbf{opred}(j)$.

Consider $\mathbf{opred}(m)$. $\mathbf{opred}(m)$ must not have been a subterm of $\mathbf{opred}(j)$. So by the remote-expectancy of Γ, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j.k).sbody}$, so also $\theta^{\Gamma'}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Assume that $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(m)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{m}$. By the $\stackrel{\rightarrow}{\leq}$-order on propositions, also $\sigma^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$. By the local consistency of Γ, $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. Since a `receive()` cannot be a value substitution, by Lemma 6(5), also $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$.

Next we show the nonlinear-expectancy of Γ' at j. Assume $\nu^{\Gamma'}_{j} = \infty$, otherwise we are done. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{j}$, so $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$. Since also $\theta^{\Gamma}_{\mathbf{opred}(j)} = \mathbf{escape}^{\pm}$, Γ was nonlinear-expectant at $\mathbf{opred}(j)$. Consider a `send()` occurrence $j.k$ with a possible communications partner m that is also a subterm of j. We wish to show that $\theta^{\Gamma'}_{j.k.sbody} = \mathbf{escape}^{\pm}$, and that if $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$. By Lemma 9, $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ were possible communications partners. On the left-hand side, $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ must both have been subterms of $\mathbf{opred}(j)$. By the nonlinear-expectancy of Γ at $\mathbf{opred}(j)$, $\theta^{\Gamma}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j.k).sbody}$. By the $\stackrel{\rightarrow}{\leq}$-order on propositions, also $\theta^{\Gamma}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Assume that $\sigma^{\Gamma'}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma}_{\mathbf{opred}(m)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'}_{m}$, so $\sigma^{\Gamma'}_{m} \leq \sigma^{\Gamma}_{\mathbf{opred}(m)}$, hence $\sigma^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$. By the nonlinear-expectancy of Γ at $\mathbf{opred}(j)$, $\omega^{\Gamma}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. A `receive()` cannot be a value substitution, so by Lemma 6(5), also $\omega^{\Gamma'}_{m} = \mathbf{local}^{\pm}$.

Last for the *Fun* subcase, we have the constraint that if $\nu_j^{\Gamma'} = \infty$, then all free variable and function subterms of j have a ν-annotation of ∞. Suppose the condition holds. By Lemma 6(4), also $\nu^{\Gamma}_{\mathbf{opred}(j)} = \infty$. Consider a free variable or function subterm $j.k$ of j. $\mathbf{opred}(j.k)$ must have been a subterm of $\mathbf{opred}(j)$. If $j.k$ is a free variable in j, then $\mathbf{opred}(j.k)$ was a free variable in $\mathbf{opred}(j)$. So by the local consistency of Γ, $\nu^{\Gamma}_{\mathbf{opred}(j.k)} = \infty$. $\mathbf{opred}(j.k)$ cannot have been a value substitution, so by Lemma 6(5), also $\nu^{\Gamma'}_{j.k} = \infty$.

Suppose *App*(j).

Use the same reasoning as in the *App* subcase for $\mathbf{seq}/\beta$ transitions.

Suppose *Cond*(j).

Use the same reasoning as in the *Cond* subcase for $\mathbf{seq}/\beta$ transitions.

Suppose *RFork*(j).

We want to show that for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma'}_{j.\mathit{rffun}}$, the escape constraints hold at k. Consider such a k. By Lemma 6(2), $\mathbf{lab}(k) = \mathbf{lab}(\mathbf{opred}(k))$. By Lemma 6(1), also *RFork*$(\mathbf{opred}(j))$. By Lemma 7, $\mathcal{P}^{\Gamma'}_{j.\mathit{rffun}} \vec{\leq} \mathcal{P}^{\Gamma}_{\mathbf{opred}(j).\mathit{rffun}}$. Hence, $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma}_{\mathbf{opred}(j).\mathit{rffun}}$. Therefore in Γ, the escape constraints held at $\mathbf{opred}(k)$. Using the same reasoning we used in the *Fun* subcase, above, we can show that the escape constraints hold in Γ' at k.

Suppose *Send*(j).

Use the same reasoning as in the *Send* subcase for $\mathbf{seq}/\beta$ transitions.

Suppose *Receive*(j).

Use the same reasoning as in the *Receive* subcase for $\mathbf{seq}/\beta$ transitions.

case channel

The context hole cannot be in a function body, therefore local consistency follows by Lemma 3.

case fork

Let Γ be the annotation map for the parent thread p on the left-hand side of the transition, let Γ' be the annotation map for the p thread on the right-hand side, and let Γ'' be the annotation map for the child thread q on the right-hand side.

The local consistency of Γ' follows by Lemma 2.

Since the function body M is a subterm of the left-hand side term containing the `fork()`, by Lemma 1, Γ'' is locally consistent for M. By the Corollary to Lemma 2, Γ'' is locally consistent for $M[\mathtt{unit}/x]$.

case rfork

This case is nearly identical to that for **fork**.

case comm

Let Γ_p be the annotation map for the p thread on the left-hand side of the transition. Γ_q is the annotation map for the q thread on the left-hand side of the transition. Γ'_p and Γ'_q are the updated annotation maps for the corresponding right-hand side threads.

The local consistency of Γ'_p follows by Lemma 2.

For Γ'_q, we need to show local consistency at each occurrence $q : j$. Since the thread q is understood, we write j instead of $q : j$ in the following discussion.

Suppose $\mathit{Var}(j)$.

The only constraint is that if k is the binding occurrence for j, then $\mathcal{P}_k^{\Gamma'_q} \stackrel{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'_q}$. If j has no binding occurrence, the constraint is satisfied. Otherwise, let k be the binding occurrence for j. Since the context hole in $C'[\,]$ is not in a function body, either j and k are both in the context $C'[\,]$, otherwise both are occurrences in the transmitted value V. In either case, by Lemma 6(1), $\mathit{Var}(\mathbf{opred}(j))$. Suppose both j and k are occurrences in the context. $\mathbf{opred}(k)$ must have been the binding occurrence for $\mathbf{opred}(j)$, and by the local consistency of Γ_q, $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)}$. Since neither $\mathbf{opred}(j)$ nor $\mathbf{opred}(k)$ is a value substitution, $\mathcal{P}_k^{\Gamma'_q} = \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k)}$, and $\mathcal{P}_j^{\Gamma'_q} = \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)}$. Hence, $\mathcal{P}_k^{\Gamma'_q} \stackrel{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'_q}$. A similar analysis holds if both j and k are occurrences in V, using the local consistency of Γ_p.

Suppose $\mathit{Channel}(j)$.

By Lemma 6(1), also $\mathit{Channel}(\mathbf{opred}(j))$. $\mathbf{opred}(j)$ may have been the transmitted value or a subterm of the transmitted value, so $\mathbf{opred}(j)$ may have been an occurrence in either p or q. Assume $\mathbf{opred}(j)$ was in p; the case for q is similar. By the local consistency of Γ_p, $\mathbf{lab}(\mathbf{opred}(j)) \in \kappa^{\Gamma_p}_{\mathbf{opred}(j)}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)} \stackrel{\rightarrow}{\leq} \mathcal{P}_j^{\Gamma'_q}$, so $\kappa^{\Gamma_p}_{\mathbf{opred}(j)} \subseteq \kappa_j^{\Gamma'_q}$. Therefore, $\mathbf{lab}(\mathbf{opred}(j)) \in \kappa_j^{\Gamma'_q}$. By Lemma 6(2), $\mathbf{lab}(j) = \mathbf{lab}(\mathbf{opred}(j))$. Hence $\mathbf{lab}(j) \in \kappa_j^{\Gamma'_q}$.

Suppose $\mathit{Fun}(j)$.

We have the constraint that $\mathbf{lab}(j) \in \phi_j^{\Gamma'_q}$. We can use similar reasoning as just presented for occurrences of `channel()`.

The next constraint is that if the number of free occurrences of the procedure binder in the procedure body is greater than 1, then $\nu^{\Gamma'_q}_{j.bv} = \infty$. Again we observe that $\mathbf{opred}(j)$ is an occurrence in one of the left-hand side terms q or p. We claim that in $\mathbf{opred}(j)$, the number of free occurrences of its binder in its body is the same as for j. To see this, note that the context hole in $C'[\,]$ cannot be in the body of a function; hence j and $\mathbf{opred}(j)$ must be occurrences of the same function. So suppose that $\mathbf{opred}(j)$ is an occurrence from the p term; in this case, j must be a subterm of V. By the local consistency of Γ_p, $\nu^{\Gamma_p}_{\mathbf{opred}(j).bv} = \infty$. By Lemma 6(4), $\mathcal{P}^{\Gamma'_q}_{j.bv} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j).bv}$. By the $\vec{\leq}$-order on propositions, $\nu^{\Gamma_p}_{\mathbf{opred}(j).bv} \leq \nu^{\Gamma'_q}_{j.bv}$, so $\nu^{\Gamma'_q}_{j.bv} = \infty$. A similar analysis works for the case that $\mathbf{opred}(j)$ is an occurrence from the q thread, using the local consistency of Γ_q.

Next consider the escape constraints that are triggered if $\theta^{\Gamma'_q}_j = \mathbf{escape}^{\pm}$. Assume the condition holds. Suppose that $\mathbf{opred}(j)$ is an occurrence from the p thread on the left-hand side. Since $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)} \vec{\leq} \mathcal{P}^{\Gamma'_q}_j$, we have $\theta^{\Gamma'_q}_j \leq \theta^{\Gamma_p}_{\mathbf{opred}(j)}$. Therefore, $\theta^{\Gamma_p}_{\mathbf{opred}(j)} = \mathbf{escape}^{\pm}$, and the escape constraints hold in Γ_p at $\mathbf{opred}(j)$. The analysis to show that $\theta^{\Gamma_q}_{\mathbf{opred}(j)} = \mathbf{escape}^{\pm}$ is similar in the case that $\mathbf{opred}(j)$ is an occurrence from the q thread on the left-hand side; the analysis to show the escape constraints hold in Γ'_q at $\mathbf{opred}(j)$ as a consequence is similar to the analysis in the following discussion.

We begin by showing that for all free variable occurrences $j.k$, $\theta^{\Gamma'_q}_{j.k} = \mathbf{escape}^{\pm}$, and, if $\sigma^{\Gamma'_q}_{j.k} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'_q}_{j.k} = \mathbf{local}^{\pm}$. Assume the condition holds for the latter constraint. Now, $\mathbf{opred}(j.k)$ was a free variable occurrence in $\mathbf{opred}(j)$, since, as we reasoned earlier, j and $\mathbf{opred}(j)$ are occurrences of the same function. By the local consistency of Γ_p, $\theta^{\Gamma_p}_{\mathbf{opred}(j.k)} = \mathbf{escape}^{\pm}$. Since it is a variable and not a value, $\mathbf{opred}(j.k)$ cannot have been a value substitution. By Lemma 6(5), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j.k)} = \mathcal{P}^{\Gamma'_q}_{j.k}$, so also $\theta^{\Gamma'_q}_{j.k} = \mathbf{escape}^{\pm}$. Also by Lemma 6(5), $\sigma^{\Gamma_p}_{\mathbf{opred}(j.k)} = \mathbf{reach}^{\pm}$. By the local consistency of Γ_p, $\omega^{\Gamma_p}_{\mathbf{opred}(j.k)} = \mathbf{local}^{\pm}$. Once more by Lemma 6(5), $\omega^{\Gamma'_q}_{j.k} = \mathbf{local}^{\pm}$.

Next is the constraint for all channel constant occurrences $j.m$ that if $\sigma^{\Gamma'_q}_{j.m} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'_q}_{j.m} = \mathbf{local}^{\pm}$. We can use the same reasoning as for the comparable constraint for free variables. However, since it is a value, a channel constant may be a value substitution. But $\mathbf{opred}(j.m)$ cannot have been a value substitution here, since j and $\mathbf{opred}(j)$ are the same function. So the analysis we used for free variables remains valid for channel constants.

Next we show that Γ'_q is remote-expectant at j. Consider any pair of possible communications partners such that exactly one of the pair is a subterm of j. We want to show that in Γ'_q, the body of the `send()` occurrence has a θ annotation of $\mathbf{escape}^{\pm}$, and that if the `receive()` occurrence has a σ annotation of $\mathbf{reach}^{\pm}$, then its ω annotation is $\mathbf{local}^{\pm}$. By Lemma 9, the occurrence predecessors of the pair were possible communications partners.

Suppose it is the `send()` that is a subterm of j, and the `receive()` is not. The reverse situation is similar. Let these be $j.k$ and m, respectively. $\mathbf{opred}(j.k)$ was a subterm of $\mathbf{opred}(j)$, so by our assumption that $\mathbf{opred}(j)$ was in the p thread, so was $\mathbf{opred}(j.k)$. Since $\mathbf{opred}(j)$ also had a θ-annotation of $\mathbf{escape}^{\pm}$, Γ_p was remote-expectant at $\mathbf{opred}(j)$.

Now consider $\mathbf{opred}(m)$, which cannot have been a subterm of $\mathbf{opred}(j)$. However, $\mathbf{opred}(m)$ may have been from either the p or q threads.

Suppose $\mathbf{opred}(m)$ is from p. Then by the remote-expectancy of Γ_p at $\mathbf{opred}(j)$, $\theta^{\Gamma_p}_{\mathbf{opred}(j.k).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{j.k.sbody} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j.k).sbody}$, so by the $\vec{\leq}$-ordering on propositions, $\theta^{\Gamma'_q}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Now assume that $\sigma^{\Gamma'_q}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(m)} \vec{\leq} \mathcal{P}^{\Gamma'_q}_{m}$. By the $\vec{\leq}$-ordering on propositions, also $\sigma^{\Gamma_p}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$. So by the remote-expectancy of Γ_p, $\omega^{\Gamma_p}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. Since a `receive()` cannot be a value substitution, by Lemma 6(5), also $\omega^{\Gamma'_q}_{m} = \mathbf{local}^{\pm}$.

Now suppose $\mathbf{opred}(m)$ is from q. We can use similar reasoning as just presented for the case that $\mathbf{opred}(m)$ is from p, except that we rely on the *distributed* remote-expectancy of $\mathcal{G}$ at $\mathbf{opred}(j)$.

Last for the escape constraints, we show the nonlinear-expectancy of Γ'_q at j. Assume $\nu^{\Gamma'_q}_{j} = \infty$, else we are done. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)} \vec{\leq} \mathcal{P}^{\Gamma'_q}_{j}$, so also $\theta^{\Gamma_p}_{\mathbf{opred}(j)} = \infty$. Therefore, Γ_p was nonlinear-expectant at $\mathbf{opred}(j)$. Consider now a `send()` occurrence $j.k$ with a possible communications partner m also a subterm of j. Both $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ must have been subterms of $\mathbf{opred}(j)$. By Lemma 9, $\mathbf{opred}(j.k)$ and $\mathbf{opred}(m)$ were possible communications partners. By the nonlinear-expectancy of Γ_p at $\mathbf{opred}(j)$, $\theta^{\Gamma_p}_{\mathbf{opred}(j.k).sbody}$ was $\mathbf{escape}^{\pm}$. By Lemma 7 and the $\vec{\leq}$-order on propositions, also $\theta^{\Gamma'_q}_{j.k.sbody} = \mathbf{escape}^{\pm}$. Now assume $\sigma^{\Gamma'_q}_{m} = \mathbf{reach}^{\pm}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(m)} \vec{\leq} \mathcal{P}^{\Gamma'_q}_{m}$, hence $\sigma^{\Gamma'_q}_{m} \leq \sigma^{\Gamma_p}_{\mathbf{opred}(m)}$. So also $\sigma^{\Gamma_p}_{\mathbf{opred}(m)} = \mathbf{reach}^{\pm}$, and by the nonlinear-expectancy of Γ_p at $\mathbf{opred}(j)$, $\omega^{\Gamma_p}_{\mathbf{opred}(m)} = \mathbf{local}^{\pm}$. Since a `receive()` cannot be a value substitution, by Lemma 6(5), also $\omega^{\Gamma'_q}_{m} = \mathbf{local}^{\pm}$.

Finally, we have the constraint that if $\nu^{\Gamma'_q}_{j} = \infty$, then all free variable and function subterms of j have a ν-annotation of ∞. Suppose the condition holds. Again, $\mathbf{opred}(j)$ may have been in the p or q threads on the left-hand side. Assume $\mathbf{opred}(j)$ was in the q thread; the analysis for the case that it was in the p thread is similar. By Lemma 6(4) and the $\vec{\leq}$-order on propositions, also $\nu^{\Gamma_q}_{\mathbf{opred}(j)} = \infty$. Let $j.k$ be a free variable or function occurrence in j. Now, $\mathbf{opred}(j.k) = \mathbf{opred}(j).k$, since j and $\mathbf{opred}(j)$ are the same function. If $j.k$ is a free variable in j, then $\mathbf{opred}(j.k)$ was a free variable in $\mathbf{opred}(j)$. By the local consistency of Γ_q, $\mathbf{opred}(j).k$ had a ν-annotation of ∞. $\mathbf{opred}(j.k)$ was not a value substitution, so by Lemma 6(5), $\mathcal{P}^{\Gamma'_q}_{j.k} = \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j.k)} = \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).k}$. Hence $\nu^{\Gamma'_q}_{j.k} = \infty$.

Suppose $App(j)$.

The first set of constraints requires that for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma'_q}_{j.rator}$, $\mathcal{P}^{\Gamma'_q}_{j.rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{k.bv}$ and $\mathcal{P}^{\Gamma'_q}_{k.body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. By Lemma 6(1), $App(\mathbf{opred}(j))$.

Consider any function occurrence k from the right-hand side thread q such that $\mathbf{lab}(k) \in \phi^{\Gamma'_q}_{j.rator}$. Now, $\mathbf{opred}(j).rator$ may be an occurrence in either the p or q threads. Assume it is in p; the case for q is similar. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{j.rator} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j).rator}$. By the $\overset{\rightarrow}{\leq}$-order on propositions, $\phi^{\Gamma'_q}_{j.rator} \subseteq \phi^{\Gamma_p}_{\mathbf{opred}(j).rator}$, so $\mathbf{lab}(k) \in \phi^{\Gamma_p}_{\mathbf{opred}(j).rator}$. By Lemma 6(2), $\mathbf{lab}(k) = \mathbf{lab}(\mathbf{opred}(k))$, so $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma_p}_{\mathbf{opred}(j).rator}$. $\mathbf{opred}(k)$ need not be an occurrence in p; it may be from q.

First suppose $\mathbf{opred}(k)$ is from p. Then by the local consistency of Γ_p, $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j).rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(k).bv}$ and $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(k).body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)}$. Since $\mathbf{opred}(k).bv$ is not the site of a substitution, by Lemma 8(2), $\mathcal{P}^{\Gamma'_q}_{k.bv} = \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(k).bv}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{j.rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j).rand}$. Hence, $\mathcal{P}^{\Gamma'_q}_{j.rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{k.bv}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{k.body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(k).body}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. Hence, $\mathcal{P}^{\Gamma'_q}_{k.body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$.

Now suppose $\mathbf{opred}(k)$ is from q. Then by the *communicative* consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j).rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).bv}$ and $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)}$. From here, we may reason as in the case that $\mathbf{opred}(k)$ is from p, *mutatis mutandis*.

Suppose $Cond(j)$.

By Lemma 6(1), also $Cond(\mathbf{opred}(j))$. Now, $\mathbf{opred}(j)$ is either from p or from q. Suppose $\mathbf{opred}(j)$ is from q; the other case is similar. So also $\mathbf{opred}(j).then$ and $\mathbf{opred}(j).else$ are from q. Because Γ_q is locally consistent, $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).then} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)}$ and $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).else} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{j.then} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).then}$ and $\mathcal{P}^{\Gamma'_q}_{j.else} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).else}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. Hence, $\mathcal{P}^{\Gamma'_q}_{j.then} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$ and $\mathcal{P}^{\Gamma'_q}_{j.else} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$.

Suppose $RFork(j)$.

In the *Fun* subcase above, under the assumption that $\theta^{\Gamma'_q}_{j} = \mathbf{escape}^{\pm}$, we showed that θ for $\mathbf{opred}(j)$ was also $\mathbf{escape}^{\pm}$. By the local consistency of Γ_p, or Γ_q, depending on the location of $\mathbf{opred}(j)$, we showed that the escape constraints held for $\mathbf{opred}(j)$ and its subterms. That was sufficient to show that the escape constraints held for j and its subterms.

We want to show that for all k such that $\mathbf{lab}(k) \in \phi^{\Gamma'_q}_{j.rffun}$, the escape constraints hold at k. Consider such a k. By Lemma 6(2), $\mathbf{lab}(k) = \mathbf{lab}(\mathbf{opred}(k))$. By Lemma 6(1), also $RFork(\mathbf{opred}(j))$. If $\mathbf{opred}(j).rffun$ is from p, we have by

Lemma 7, $\phi^{\Gamma'_q}_{j.\mathit{rffun}} \subseteq \phi^{\Gamma_p}_{\mathbf{opred}(j).\mathit{rffun}}$, so $\mathbf{lab}(\mathbf{opred}(k)) \in \phi^{\Gamma_p}_{\mathbf{opred}(j).\mathit{rffun}}$. Therefore, if $\mathbf{opred}(k)$ is also in p, we have satisfaction of the escape constraints in Γ_p at $\mathbf{opred}(k)$. By the same reasoning we used in the *Fun* subcase above, we also have satisfaction of the escape constraints in Γ'_q at k. The case that both $\mathbf{opred}(j).\mathit{rffun}$ and $\mathbf{opred}(k)$ are in q is similar, using the local consistency of Γ_q.

However, the quantification over all k in the constraint introduces a slight difficulty. It might be that $\mathbf{opred}(j).\mathit{rffun}$ is from p, but $\mathbf{opred}(k)$ is from q, or vice-versa. In those cases, the local consistency of Γ_p and Γ_q does not tell us that the escape constraints held at $\mathbf{opred}(k)$. Recall, universal quantifiers appearing in the local constraints range over occurrences in individual threads. Therefore, for these cases, we rely on the comparable communicative consistency constraint for $\mathit{RFork}(j)$ to show that the escape constraints hold at $\mathbf{opred}(k)$. So in these cases also, the escape constraints hold in Γ'_q at k.

Suppose $\mathit{Send}(j)$.

The constraint is that $\sigma^{\Gamma'_q}_{j.\mathit{schan}} = \mathbf{reach}^{\pm}$. By Lemma 6(1), also $\mathit{Send}(\mathbf{opred}(j))$. Suppose $\mathbf{opred}(j)$ is from q. Since Γ_q is locally consistent, $\sigma^{\Gamma_q}_{\mathbf{opred}(j).\mathit{schan}} = \mathbf{reach}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{j.\mathit{schan}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j).\mathit{schan}}$. By the $\overset{\rightarrow}{\leq}$-order on propositions, $\sigma^{\Gamma'}_{j.\mathit{schan}} = \mathbf{reach}^{\pm}$. The case that $\mathbf{opred}(j)$ is from p is similar.

Suppose $\mathit{Receive}(j)$.

The first constraint is that $\sigma^{\Gamma'_q}_{j.\mathit{rchan}} = \mathbf{reach}^{\pm}$. Use similar reasoning as in the *Send* subcase for the constraint on σ annotations.

Now consider a k from the right-hand side q term, such that $\mathit{Send}(k)$, $\kappa^{\Gamma'_q}_{k.\mathit{schan}} \cap \kappa^{\Gamma'_q}_{j.\mathit{rchan}} \neq \emptyset$. By Lemma 6(1), also $\mathit{Send}(\mathbf{opred}(k))$. By Lemma 9, $\mathbf{opred}(j)$ and $\mathbf{opred}(k)$ were communications partners.

Suppose first that $\mathbf{opred}(j)$ and and $\mathbf{opred}(k).\mathit{sbody}$ were both from q. By the local consistency of Γ_q, $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{k.\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).\mathit{sbody}}$, and by Lemma 6(4) $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(j)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. Hence, $\mathcal{P}^{\Gamma'_q}_{k.\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. The case that both $\mathbf{opred}(j)$ and and $\mathbf{opred}(k).\mathit{sbody}$ were from p is similar.

Suppose instead that $\mathbf{opred}(j)$ was from p, but $\mathbf{opred}(k).\mathit{sbody}$ was from q. By the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_q}_{k.\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_q}_{\mathbf{opred}(k).\mathit{sbody}}$ and by Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(j)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. Hence, $\mathcal{P}^{\Gamma'_q}_{k.\mathit{sbody}} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_q}_{j}$. The case that $\mathbf{opred}(j)$ was from q and $\mathbf{opred}(k).\mathit{sbody}$ was from p is similar. ∎

Our last invariance result is:

Theorem 3 *Let Π be a thread map with a family of annotation maps $\mathcal{G}$ that is locally consistent, communicative consistent, and κ-coherent. Suppose $\Pi \xrightarrow[con]{} \Pi'$.*

Then the updated family of annotation maps $\mathcal{G}'$ is communicative consistent for Π'.

Proof. By considering each possible $\xrightarrow[con]{}$-transition and $\mathcal{G}'$, the updated family of annotation maps.

For each possible $\xrightarrow[con]{}$-transition, we check the communicative consistency of the annotation map or maps at each occurrence $r : j$ on the right-hand side, where r is an arbitrary thread. Assume each transition is as in Figures 6 to 12. As before, an unprimed Γ_s is the annotation map for a left-hand side thread s; Γ'_s is the updated annotation map for s on the right-hand side.

case seq/β

As shown in Figure 6, the β-reduction occurs in thread p.

Suppose $\mathit{Fun}(r : j)$.

We want to show that if $\theta^{\Gamma'_r}_{r:j} = \mathbf{escape}^{\pm}$, then $\mathcal{G}'$ is distributed remote-expectant at $r : j$. Assume the condition holds. Since no communication between threads occurs, $\mathbf{opred}(r : j)$ is in the thread r on the left-hand side. By Lemma 6(4), $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, so by the $\stackrel{\rightarrow}{\leq}$-order on propositions, also $\theta^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)} = \mathbf{escape}^{\pm}$. Since also $\mathit{Fun}(\mathbf{opred}(r : j))$, by the communicative consistency of $\mathcal{G}$, $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(r : j)$.

Consider a `receive()` that is a subterm of $r : j$, say $r : j.k$, and a `send()`, say $s : m$, such that $r : j.k$ and $s : m$ are possible communications partners. By Lemma 9, $\mathbf{opred}(r : j.k)$ and $\mathbf{opred}(s : m)$ were possible communications partners. Note that $r : j.k$ and $s : m$ may be in the same or in different threads.

Suppose that $\mathbf{opred}(r : j.k)$ was a subterm of $\mathbf{opred}(r : j)$.

Suppose first that $\mathbf{opred}(s : m)$ was not a subterm of $\mathbf{opred}(r : j)$, including the possibility the two occurrences were in different threads. By the distributed remote-expectancy of $\mathcal{G}$ at $\mathbf{opred}(r : j)$, $\theta^{\Gamma_s}_{\mathbf{opred}(s\,:\,m).\mathit{sbody}} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:m.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,m).\mathit{sbody}}$, so $\theta^{\Gamma_s}_{\mathbf{opred}(s\,:\,m).\mathit{sbody}} \leq \theta^{\Gamma'_s}_{s:m.\mathit{sbody}}$. Therefore, $\theta^{\Gamma'_s}_{s:m.\mathit{sbody}} = \mathbf{escape}^{\pm}$. Assume that $\sigma^{\Gamma'_r}_{r:j.k} = \mathbf{reach}^{\pm}$. Since a `receive()` is not a value, by Lemma 6(5), $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathcal{P}^{\Gamma'_r}_{r:j.k}$. Hence, $\sigma^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathbf{reach}^{\pm}$. Then by the distributed remote-expectancy of $\mathcal{G}$ at $\mathbf{opred}(r : j)$, $\omega^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathbf{local}^{\pm}$. So also $\omega^{\Gamma'_r}_{r:j.k} = \mathbf{local}^{\pm}$.

Now suppose that $\mathbf{opred}(s : m)$ was a subterm of $\mathbf{opred}(r : j)$. Therefore, $s = r$. In this case, it must have been that $\mathbf{opred}(r : j)$ was duplicated by the β-reduction. The application operand $i.\mathit{rand}$ must have contained $\mathbf{opred}(r : j)$, and the operand must have been substituted for a pair of free occurrences of the operator binder $[\![i.\mathit{rator}.\mathit{bv}]\!]$ in the operator body. By the local consistency of Γ_r, $\nu^{\Gamma_r}_{i.\mathit{rator}.\mathit{bv}} = \infty$. By the nonlinear-expectancy of Γ_r at

$r : j$, $\theta^{\Gamma_r}_{\mathbf{opred}(r\,:\,m).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:m.sbody} \vec{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,m).sbody}$, so $\theta^{\Gamma_r}_{\mathbf{opred}(r\,:\,m).sbody} \leq \theta^{\Gamma'_r}_{r:m.sbody}$. So also $\theta^{\Gamma'_r}_{r:m.sbody} = \mathbf{escape}^{\pm}$. Now assume that $\sigma^{\Gamma'_r}_{r:j.k} = \mathbf{reach}^{\pm}$. Since a `receive()` is not a value, by Lemma 6(5), $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathcal{P}^{\Gamma'_r}_{r:j.k}$. Hence also $\sigma^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathbf{reach}^{\pm}$. Then by the nonlinear-expectancy of Γ_r at $\mathbf{opred}(r : j)$, $\omega^{\Gamma_r}_{\mathbf{opred}(r\,:\,j.k)} = \mathbf{local}^{\pm}$. So also $\omega^{\Gamma'_r}_{r:j.k} = \mathbf{local}^{\pm}$.

Suppose instead that $\mathbf{opred}(r : j.k)$ was a subterm of the substituted operand. We can show that in Γ_r, the operand had a θ-annotation of $\mathbf{escape}^{\pm}$. Therefore, $\mathcal{G}$ was distributed remote-expectant at the operand. From here, we can reason as in the case that $\mathbf{opred}(r : j.k)$ was a subterm of $\mathbf{opred}(r : j)$. We can distinguish the situations where $\mathbf{opred}(s : m)$ was and was not a subterm of the operand.

We may reverse the roles of $r : j.k$ and $s : m$, so that the former is a `send()` and the latter is a `receive()`. A similar analysis gives the desired result.

Suppose $App(r : j)$.

We wish to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rator}$, that $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$ and $\mathcal{P}^{\Gamma'_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Consider such a function occurrence $s : k$. By Lemma 7 and the $\vec{\leq}$-order on propositions, $\phi^{\Gamma'_r}_{r:j.rator} \subseteq \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rator}$. By Lemma 6(2), $\mathbf{lab}(s : k) = \mathbf{lab}(\mathbf{opred}(s : k))$, so $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rator}$. Note that $\mathbf{opred}(s : k)$ and $\mathbf{opred}(r : j)$ were in different threads on the left-hand side. By the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rand} \vec{\leq} \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).bv}$ and $\mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).body} \vec{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rand}$. Since $\mathbf{opred}(s : k).bv$ cannot have been the site of a substitution, by Lemma 8(2), $\mathcal{P}^{\Gamma'_s}_{s:k.bv} = \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).bv}$. Therefore, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$. By Lemma 7, $\mathcal{P}^{\Gamma_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).body}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Hence $\mathcal{P}^{\Gamma'_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Suppose $RFork(r : j)$.

First we want to show that for all s and k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rffun}$, that $\mathcal{G}'$ is distributed remote-expectant at $s : k$.

Choose any such occurrence $s : k$. $\mathbf{opred}(r : j)$ was an occurrence of an `rfork()` in the r thread on the left-hand side. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:j.rffun} \vec{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. Therefore, $\phi^{\Gamma'_r}_{r:j.rffun} \subseteq \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. Also, by Lemma 6(2), $\mathbf{lab}(s : k) = \mathbf{lab}(\mathbf{opred}(s : k))$. So $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. By the communicative consistency of $\mathcal{G}$, $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(s : k)$. From here, we may reason as we did in the *Fun* subcase for the distributed remote-expectancy constraint.

Next we want to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s : k) \in r :$

$j.rffun$, that the escape constraints hold in Γ'_s at $s : k$.

Consider such a function occurrence $s : k$. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:j.rffun} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. So $\phi^{\Gamma'_r}_{r:j.rffun} \subseteq \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. Also, by Lemma 6(2), $\mathbf{lab}(\mathbf{opred}(s : k)) = \mathbf{lab}(s : k)$, so $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).rffun}$. By the communicative consistency of $\mathcal{G}$, the escape constraints held at $\mathbf{opred}(s : k)$. Now, if $s \neq p$, then the s thread is the same on both sides of the transition, and $\Gamma_s = \Gamma'_s$, so clearly the escape constraints must also hold at $s : k$. But suppose $s = p$. In this case, as we showed for Theorem 2 in the *Fun* case for a $\mathbf{seq}/\beta$ transition, if the escape constraints held at the occurrence predecessor of a right-hand side function occurrence, then the escape constraints also hold at the function occurrence itself.

Suppose $Receive(r : j)$.

We want to show that for all $s \neq r$, for all k such that $Send(s : k)$, if $r : j$ and $s : k$ are possible communications partners, then (1) $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, and (2) $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$ implies $\theta^{\Gamma'_s}_{s:k.sbody} = \mathbf{escape}^{\pm}$, and if $\sigma^{\Gamma'_s}_{r:j} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'_s}_{r:j} = \mathbf{local}^{\pm}$.

By Lemma 9, $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were also communications partners. Since these occurrence predecessors were also in the r and s threads, respectively, by the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).sbody}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_r}_{\mathbf{opred}(r\,:\,j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Hence, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Now suppose $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$. Then also $\mathbf{proc}_{\Pi}(r) \neq \mathbf{proc}_{\Pi}(s)$. By the communicative consistency of $\mathcal{G}$, $\theta^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).sbody} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_s}_{\mathbf{opred}(s\,:\,k).sbody}$. By the $\stackrel{\rightarrow}{\leq}$-order on propositions, also $\theta^{\Gamma'_s}_{s:k.sbody} = \mathbf{escape}^{\pm}$. Since a `receive()` cannot be a value substitution, $\mathcal{P}^{\Gamma_s}_{\mathbf{opred}(r\,:\,j)} = \mathcal{P}^{\Gamma'_s}_{r:j}$. Assume $\sigma^{\Gamma'_s}_{r:j} = \mathbf{reach}^{\pm}$. Then also $\sigma^{\Gamma_s}_{\mathbf{opred}(r\,:\,j)} = \mathbf{reach}^{\pm}$. By the communicative consistency of $\mathcal{G}$, $\omega^{\Gamma_s}_{\mathbf{opred}(r\,:\,j)} = \mathbf{local}^{\pm}$. So also $\omega^{\Gamma'_s}_{r:j} = \mathbf{local}^{\pm}$.

case seq/cond-true

We omit the similar **seq**/cond-false case.

Suppose $Fun(r : j)$.

We want to show that if $\theta^{\Gamma'_r}_{r:j} = \mathbf{escape}^{\pm}$, then $\mathcal{G}'$ is distributed remote-expectant at $r : j$. We can use almost the same analysis as in the $\mathbf{seq}/\beta$ case. Since the conditional reduction does not involve substitution for a free variable, if we have an occurrence $r : j.k$ that is a subterm of $r : j$, then $\mathbf{opred}(r : j.k)$ is certainly a subterm of $\mathbf{opred}(r : j)$. Also, if an occurrence $s : m$ is not a subterm of $r : j$, then $\mathbf{opred}(s : m)$ is certainly not a subterm of $\mathbf{opred}(r : j)$. Therefore, we can exclude some of the possibilities we considered in the $\mathbf{seq}/\beta$ case. In particular, we need not consider the possibility of duplicated terms.

Suppose $App(r:j)$.

Use the same reasoning as in the *App* subcase for $\mathbf{seq}/\beta$ transitions.

Suppose $RFork(r:j)$.

First we wish to show that for all s and k such that $\mathbf{lab}(s:k) \in \phi^{\Gamma'_r}_{r:j.rffun}$, that $\mathcal{G}'$ is distributed remote-expectant at $s:k$. As in the *Fun* subcase above, we may follow the reasoning used in the $\mathbf{seq}/\beta$ case for the *Fun* and *RFork* subcases, excluding the irrelevant situations.

Next, we wish to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s:k) \in r:j.sbody$, that the escape constraints hold in Γ'_s at $s:k$.

Consider such a function occurrence $s:k$. We can show that $\mathbf{lab}(\mathbf{opred}(s:k)) \in \phi^{\Gamma_r}_{\mathbf{opred}(r\,:\,j).sbody}$. By the communicative consistency of $\mathcal{G}$, the escape constraints held at $\mathbf{opred}(s:k)$. If $s \neq p$, then the s term is the same on both sides of the transition, also $\Gamma_s = \Gamma'_s$, so the escape constraints hold at $s:k$. Suppose instead $s = p$. As we showed in Theorem 2 in the *Fun* subcase for a $\mathbf{seq}$/cond-true transition, if the escape constraints held at the occurrence predecessor of a right-hand function occurrence, such as $s:k$, then the escape constraints also hold at the function occurrence itself.

Suppose $Receive(r:j)$.

Use the same reasoning as in the *Receive* subcase for $\mathbf{seq}/\beta$ transitions.

case channel

We have $\mathcal{G} = \mathcal{G}'$. All that is different between the left-hand and right-hand sides is the fresh channel constant in the thread p, which replaces an occurrence of `channel()`. The only communicative consistency constraint that might apply to that channel constant is the escape constraint that applies to channel constants in function bodies. Since the context hole in the p thread cannot be in the body of the function, in fact that constraint does not apply. Since $\mathcal{G}$ was communicative consistent, so is $\mathcal{G}'$.

case fork

Suppose $Fun(r:j)$.

We want to show that if $\theta^{\Gamma'_r}_{r:j} = \mathbf{escape}^{\pm}$, then $\mathcal{G}'$ is distributed remote-expectant at $r:j$. Assume the condition holds. Now, $\mathbf{opred}(r:j)$ may be from the thread r, but if r is the child thread, then $\mathbf{opred}(r:j)$ is an occurrence in the parent thread. In Figure 10, p is the parent thread. Let r' be the left-hand side thread containing $\mathbf{opred}(r:j)$, so either $r' = r$ or $r' = p$. By Lemma 6(4), $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, so also $\theta^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} = \mathbf{escape}^{\pm}$. By the communicative consistency of $\mathcal{G}$, $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(r:j)$.

Consider a `receive()` that is a subterm of $r : j$, say $r : j.k$, and a `send()` occurrence, say $s : m$, that is not a subterm of $r : j$. As usual, the situation where the `send()` is a subterm of $r : j$ and the `receive()` is not a subterm is similar. Assume the occurrences are possible communications partners. By Lemma 9, $\mathbf{opred}(r : j.k)$ and $\mathbf{opred}(s : m)$ were possible communications partners.

Now, $\mathbf{opred}(r : j.k)$ and $\mathbf{opred}(s : m)$ may have been in in the same or in distinct threads on the left-hand side. In either case, however, $\mathbf{opred}(r : j.k)$ is certainly a subterm of $\mathbf{opred}(r : j)$, and $\mathbf{opred}(s : m)$ was certainly not a subterm of $\mathbf{opred}(r : j)$. This is the same situation we had in the **seq**/cond-true case, so the reasoning we used there for this constraint applies.

Suppose $App(r : j)$.

We want to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rator}$, that $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$ and $\mathcal{P}^{\Gamma'_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Consider a function occurrence $s : k$ such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rator}$. If $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ are in distinct threads on the left-hand side, then use the same reasoning as in the *App* subcases for **seq**/β and **seq**/cond-true transitions.

If $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ are not in distinct threads, then they must both have been occurrences in the parent thread p. We know $App(\mathbf{opred}(r : j))$. By Lemma 7 and the $\vec{\leq}$-order on propositions, $\phi^{\Gamma'_r}_{r:j.rator} \subseteq \phi^{\Gamma_p}_{\mathbf{opred}(r\,:\,j).rator}$. By Lemma 6(2), $\mathbf{lab}(s : k) = \mathbf{lab}(\mathbf{opred}(s : k))$, so $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_p}_{\mathbf{opred}(r\,:\,j).rator}$. Therefore, by the local consistency of Γ_p, $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j).rand} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).bv}$ and $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).body} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j).rand}$. $\mathbf{opred}(s : k).bv$ cannot have been the site of a substitution, so by Lemma 8(2), $\mathcal{P}^{\Gamma'_s}_{s:k.bv} = \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).bv}$. Therefore, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$. By Lemma 7, $\mathcal{P}^{\Gamma_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).body}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Hence, $\mathcal{P}^{\Gamma'_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Suppose $RFork(r : j)$.

First we wish to show that for all s and k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rffun}$, that $\mathcal{G}'$ is distributed remote-expectant at $s : k$.

Choose any such occurrence $s : k$. As in the **seq**/β and **seq**/cond-true cases, we can show that $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(s : k)$. From here, we may reason as in the **seq**/cond-true case, observing that there is no possibility of term duplication.

Next, we show that for all $s \neq r$ and all k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rffun}$, that the escape constraints hold in Γ'_s at $s : k$.

Consider such a function occurrence $s : k$. Let r' be the thread where $\mathbf{opred}(r : j)$ occurred on the left-hand side. We can show, as in the $\mathbf{seq}/\beta$ and $\mathbf{seq}$/cond-true cases, that $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).\mathit{rffun}}$.

Suppose $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were occurrences in distinct threads on the left-hand side. Then by the communicative consistency of $\mathcal{G}$, the escape constraints held at $\mathbf{opred}(s : k)$. As we showed in Theorem 2 in the *Fun* subcase for **fork** transitions, since the escape constraints held at $\mathbf{opred}(s : k)$, the escape constraints also hold at $s : k$.

On the other hand, suppose $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were both in the parent thread p on the left-hand side. Then by the local consistency of Γ_p, the escape constraints held at $\mathbf{opred}(s : k)$. So again, the escape constraints also hold at $s : k$.

Suppose *Receive*$(r : j)$.

We want to show that for all $s \neq r$, for all k such that *Send*$(s : k)$, if $r : j$ and $s : k$ are possible communications partners, then (1) $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, and (2) $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$ implies that $\theta^{\Gamma'_s}_{s:k.\mathit{sbody}} = \mathbf{escape}^{\pm}$ and that $\sigma^{\Gamma'_s}_{r:j} = \mathbf{reach}^{\pm}$ implies $\omega^{\Gamma'_s}_{r:j} = \mathbf{local}^{\pm}$.

By Lemma 9, $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were also possible communications partners. Suppose these occurrence predecessors were in distinct threads. Then by the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)}$, where s' and r' are the appropriate left-hand side threads. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).\mathit{sbody}}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Therefore, $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Suppose instead that these occurrence predecessors were both in the parent thread p. So by the local consistency of Γ_p, $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).\mathit{sbody}}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. So $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Now suppose $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$. Suppose that r and s were also threads on the left-hand side. In that case, $\mathbf{opred}(r : j)$ was in the r thread on the left-hand side, and $\mathbf{opred}(s : k)$ was in the s thread. Also, $\mathbf{proc}_{\Pi}(r) \neq \mathbf{proc}_{\Pi}(s)$, so we may reason as in the $\mathbf{seq}/\beta$ and $\mathbf{seq}$/cond-true cases for the same constraint.

But suppose that not both of r and s were threads on the left-hand side. One of r and s must be the child thread, and the other is a thread other than the parent thread. Suppose r is the child thread. Then on the left-hand side, $\mathbf{opred}(r : j)$ was an occurrence in the parent thread and $\mathbf{opred}(s : k)$ was an occurrence in the s thread. Since $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$ and the child thread runs on the same processor as its parent, we have $\mathbf{proc}_{\Pi}(p) \neq \mathbf{proc}_{\Pi}(s)$. Suppose instead that s is the child thread. Then on the left-hand side, $\mathbf{opred}(s : k)$ was an occurrence in the parent thread and $\mathbf{opred}(r : j)$ was an occurrence in the r thread. So again $\mathbf{proc}_\Pi(p) \neq \mathbf{proc}_\Pi(r)$. From this point, we may reason much as in the $\mathbf{seq}/\beta$ and $\mathbf{seq}$/cond-true cases for this same constraint. Since the occurrence predecessor of the child thread is in the parent thread, we just need to substitute Γ_p at relevant points in the proof in place of Γ_r or Γ_s.

case rfork

Suppose $\mathit{Fun}(r : j)$.

Use the same reasoning as in the *Fun* subcase for **fork** transitions.

Suppose $\mathit{App}(r : j)$.

Use the same reasoning as in the *App* subcase for **fork** transitions.

Suppose $\mathit{RFork}(r : j)$.

Use the same reasoning as in the *RFork* subcase for **fork** transitions.

Suppose $\mathit{Receive}(r : j)$.

We want to show that for all $s \neq r$, for all k such that $\mathit{Send}(s : k)$, if $r : j$ and $s : k$ are possible communications partners, then (1) $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, and (2) $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$ implies $\theta^{\Gamma'_s}_{s:k.\mathit{sbody}} = \mathbf{escape}^{\pm}$, and $\sigma^{\Gamma'_s}_{r:j} = \mathbf{reach}^{\pm}$ implies $\omega^{\Gamma'_s}_{r:j} = \mathbf{local}^{\pm}$.

For (1), we may reason exactly as in the *Receive* subcase for **fork** transitions.

Suppose that $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$. Suppose also that r and s were distinct threads on the left-hand side. Then $\mathbf{proc}_\Pi(r) \neq \mathbf{proc}_\Pi(s)$. From here, we may reason as in the $\mathbf{seq}/\beta$, $\mathbf{seq}$/cond-true, and **fork** cases for the same constraint.

But suppose that not both of r and s were threads on the left-hand side. One of r and s must be the child thread; the other may be the parent thread, or another thread. If $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ occurred in threads on different processors, then again we may reason as in the $\mathbf{seq}/\beta$, $\mathbf{seq}$/cond-true, and **fork** cases.

Suppose now that the two occurrence predecessors were in threads on the same processor.

Suppose that r is the child thread. Then $\mathbf{opred}(r : j)$ must have been a subterm of the `rfork()`'d function.

One possibility is that $\mathbf{opred}(s : k)$ is also an occurrence in the parent thread. $\mathbf{opred}(s : k)$ was not a subterm of the `rfork()`'d function. By the local consistency of Γ_p, Γ_p was remote-expectant at the `rfork()`'d function. By Lemma 9, $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were possible communications partners. By the remote-expectancy of Γ_p, $\theta^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).\mathit{sbody}} = \mathbf{escape}^{\pm}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.\mathit{sbody}} \stackrel{\rightarrow}{\leq}$

$\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).sbody}$. By the $\vec{\leq}$-order on propositions, also $\theta^{\Gamma'_s}_{s:k.sbody} = \mathbf{escape}^{\pm}$. Assume $\sigma^{\Gamma'_r}_{r:j} = \mathbf{reach}^{\pm}$. Since a `receive()` is not a value, by Lemma 6(5), $\mathcal{P}^{\Gamma_r}_{r:j} = \mathcal{P}^{\Gamma'_p}_{\mathbf{opred}(r\,:\,j)}$. So also $\sigma^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} = \mathbf{reach}^{\pm}$. By the remote-expectancy of Γ_p, $\omega^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} = \mathbf{local}^{\pm}$. So also $\omega^{\Gamma'_r}_{r:j} = \mathbf{local}^{\pm}$.

The other possibility is that $\mathbf{opred}(s : k)$ is an occurrence in a thread different from the parent thread, but running on the same processor as the parent thread. In this case, the distributed remote-expectancy of $\mathcal{G}$ at the `rfork()`'d function gives the desired results.

Suppose s is the child thread. Therefore, $\mathbf{opred}(s : k)$ must have been a subterm of the `rfork()`'d function.

One possibility is that $\mathbf{opred}(r : j)$ is also an occurrence in the parent thread. $\mathbf{opred}(r : j)$ was not a subterm of the `rfork()`'d function. From here, we may reason as in the case that r is the child thread, and $\mathbf{opred}(s : k)$ was also in the parent thread.

The other possibility is that $\mathbf{opred}(r : j)$ is an occurrence in a thread different than the parent thread, but running on the same processor as the parent thread. Once more, the distributed remote-expectancy of $\mathcal{G}$ at the `rfork()`'d function gives the desired results.

case comm

Suppose $\mathit{Fun}(r : j)$.

We want to show that if $\theta^{\Gamma'_r}_{r:j} = \mathbf{escape}^{\pm}$, then $\mathcal{G}'$ is distributed remote-expectant at $r : j$. Assume the condition holds. Since a value is transmitted, $\mathbf{opred}(r : j)$ may be from a left-hand side thread other than r. Let r' be the thread in which $\mathbf{opred}(r : j)$ occurs. By Lemma 6(4), $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, so by the $\vec{\leq}$-order on propositions, also $\theta^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} = \mathbf{escape}^{\pm}$. By the communicative consistency of $\mathcal{G}$, $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(r : j)$.

Consider a `receive()` that is a subterm of $r : j$, say $r : j.k$, and a `send()` occurrence, say $s : m$, that is not a subterm of $r : j$. As in the **seq**/cond-true, **fork**, and **rfork** cases, $\mathbf{opred}(r : j.k)$ must have been a subterm of $\mathbf{opred}(r : j)$, and $\mathbf{opred}(s : m)$ must not have been a subterm of $\mathbf{opred}(r : j)$. Therefore, we may use the same reasoning as in those cases for this constraint. Reversing the roles of $r : j.k$ and $s : m$ allows to obtain the desired results by a similar analysis.

Suppose $\mathit{App}(r : j)$.

We wish to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rator}$, that $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \vec{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$ and $\mathcal{P}^{\Gamma'_s}_{s:k.body} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Consider such a function occurrence $s : k$. Now, $\mathbf{opred}(r : j)$ could have been in a thread other than r on the left-hand side. Let r' be that thread. By Lemma 7 and the ordering on propositions, $\phi^{\Gamma'_r}_{r:j.rator} \overset{\rightarrow}{\leq} \phi^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).rator}$. By Lemma 6(2), $\mathbf{lab}(s : k) = \mathbf{lab}(\mathbf{opred}(s : k))$, so $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).rator}$. Note that $\mathbf{opred}(s : k)$ and $\mathbf{opred}(r : j)$ may have been in the same or in different threads on the left-hand side.

Suppose $\mathbf{opred}(s : k)$ and $\mathbf{opred}(r : j)$ were in different threads on the left-hand side. Let s' be the thread in which $\mathbf{opred}(s : k)$ occurs. By the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).bv}$ and $\mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).rand}$. Since $\mathbf{opred}(s : k).bv$ cannot have been the site of a substitution, by Lemma 8(2), $\mathcal{P}^{\Gamma'_s}_{s:k.bv} = \mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).bv}$. Therefore, $\mathcal{P}^{\Gamma'_r}_{r:j.rand} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_s}_{s:k.bv}$. By Lemma 7, $\mathcal{P}^{\Gamma_s}_{s:k.body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).body}$; by Lemma 6(4), $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Hence, $\mathcal{P}^{\Gamma'_s}_{s:k.body} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Now suppose $\mathbf{opred}(s : k)$ and $\mathbf{opred}(r : j)$ were in the same thread on the left-hand side. Let t be that thread. The analysis is similar to that in the last paragraph, except that we rely on the local consistency of Γ_t instead of the communicative consistency of $\mathcal{G}$.

Suppose *RFork*$(r : j)$.

We wish to show that for all s and k such that $\mathbf{lab}(s : k) \in \phi^{\Gamma'_r}_{r:j.rffun}$, that $\mathcal{G}'$ is distributed remote-expectant at $s : k$.

Choose any such occurrence $s : k$. Let r' be the thread in which $\mathbf{opred}(r : j)$ occurred. We can show that $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).rffun}$, so $\mathcal{G}$ was distributed remote-expectant at $\mathbf{opred}(s : k)$. From here, the analysis is as in the *Fun* subcase for this constraint.

Next, we wish to show that for all $s \neq r$, for all k such that $\mathbf{lab}(s : k) \in r : j.sbody$, that the escape constraints hold at $s : k$.

Consider such a function occurrence $s : k$. Let r' be the left-hand side thread where $\mathbf{opred}(r : j)$ occurs. We can show, as in previous cases for the same constraint, that $\mathbf{lab}(\mathbf{opred}(s : k)) \in \phi^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j).sbody}$. If $\mathbf{opred}(r : j)$ and $\mathbf{opred}(s : k)$ were in different threads, then by the communicative consistency of $\mathcal{G}$, the escape constraints held at $\mathbf{opred}(s : k)$. If they were in the same thread r', then the escape constraints held at $\mathbf{opred}(s : k)$ by the local consistency of $\Gamma_{r'}$. As we showed in Theorem 2 in the *Fun* subcase for a **comm** transition, if the escape constraints held at the occurrence predecessor of a right-hand function occurrence, such as $s : k$, then the escape constraints also hold at the function occurrence itself.

Suppose *Receive*$(r : j)$.

We want to show that for all $s \neq r$, for all k such that *Send*$(s : k)$, and $r : j$

and $s:k$ are possible communications partners, that (1) $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$, and (2) $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$ implies $\theta^{\Gamma'_s}_{s:k.sbody} = \mathbf{escape}^{\pm}$, and if $\sigma^{\Gamma'_s}_{r:j} = \mathbf{reach}^{\pm}$, then $\omega^{\Gamma'_s}_{r:j} = \mathbf{local}^{\pm}$.

Choose any such $s:k$. By Lemma 9, $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ were also communications partners. Suppose these occurrence predecessors were in different threads. Then by the communicative consistency of $\mathcal{G}$, $\mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).sbody} \vec{\leq} \mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)}$, where s' and r' are the appropriate left-hand side threads. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \vec{\leq} \mathcal{P}^{\Gamma_{s'}}_{\mathbf{opred}(s\,:\,k).sbody}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_{r'}}_{\mathbf{opred}(r\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Therefore, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

The only other possibility is that the occurrence predecessors were both in the `send()`'ing thread p. So by the local consistency of Γ_p, $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).sbody} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)}$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \vec{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).sbody}$. By Lemma 6(4), $\mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$. Hence $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \vec{\leq} \mathcal{P}^{\Gamma'_r}_{r:j}$.

Suppose that $\mathbf{proc}_{\Pi'}(r) \neq \mathbf{proc}_{\Pi'}(s)$. Suppose also that $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ were in distinct threads on different processors on the left-hand side. Then we may reason as in previous cases for the same constraint.

Suppose instead that $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ were in the same thread, or in different threads on the same processor, on the left-hand side. Then on the right-hand side, at least one of $r:j$ and $s:k$ must be running on a processor different than its occurrence predecessor. Therefore, at least one of $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ must have been a subterm of the transmitted value. But if both $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ were such subterms, then on the right-hand side, $r:j$ and $s:k$ would be occurrences in the same thread. Therefore, exactly one of $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ was a subterm of transmitted value. Also, the transmitted value must have been a function.

Consider the communicating `send()` and `receive()` on the left-hand side. They must have been in distinct threads on different processors. By the κ-coherence of $\mathcal{G}$, the intersection of the κ annotations of the channel-parts of the `send()` and `receive()` was non-empty. So by the communicative consistency of $\mathcal{G}$, the transmitted function, which was the body of the `send()`, had a θ annotation of $\mathbf{escape}^{\pm}$.

There are now two possibilities to consider: (1) that $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ were both occurrences in the `send()`'ing thread p, and (2) that one of $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ was an occurrence in p, and the other occurred in some other left-hand side thread.

In the case of (1), by local consistency, Γ_p was remote-expectant at the transmitted function. Regardless of which of $\mathbf{opred}(r:j)$ and $\mathbf{opred}(s:k)$ was the subterm of the transmitted function, by remote-expectancy, $\theta^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).sbody} =$

escape$^\pm$. By Lemma 7, $\mathcal{P}^{\Gamma'_s}_{s:k.sbody} \overset{\rightarrow}{\leq} \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(s\,:\,k).sbody}$, so $\theta^{\Gamma'_s}_{s:k.sbody} = \mathbf{escape}^\pm$. Assume now that $\sigma^{\Gamma'_r}_{r:j} = \mathbf{reach}^\pm$. Since a `receive()` is not a value, $\mathcal{P}^{\Gamma'_r}_{r:j} = \mathcal{P}^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)}$. Hence, by remote-expectancy, $\omega^{\Gamma_p}_{\mathbf{opred}(r\,:\,j)} = \mathbf{local}^\pm$, and also $\omega^{\Gamma'_r}_{r:j} = \mathbf{local}^\pm$.

For (2), by the communicative consistency of $\mathcal{G}$, $\mathcal{G}$ was distributed remote-expectant at the transmitted function. From here, we may reason as we did for (1), except that the annotation for one of the occurrence predecessors is not given by Γ_p, but by the annotation map for some other left-hand side thread. ∎

13 Soundness

Any solution to the local consistency and communicative consistency constraints that is also κ-coherent is sound. Our main result is:

Theorem 4 (Soundness) *Let* Π *be a thread map with a family of annotation maps* $\mathcal{G}$ *that is locally consistent, communicative consistent, and* κ*-coherent. If* $\Pi \underset{con}{\longrightarrow}^* \Pi'$*, then* $\Pi \underset{con}{\longrightarrow}^* \Pi' \models \mathcal{G}$.

Proof. By Theorems 2 and 3.

Let $p : i$ be an occurrence of `channel()` in Π that $\mathcal{G}$ gives a ω annotation of $\mathbf{local}^+$. Let Π'' be any intermediate thread map in the evaluation, possibly the same as Π'. Suppose there is an occurrence $s : n$ of a channel constant k in Π'', such that $s : n$ was produced by $p : i$ in Π, and $s : n$ is the channel-part subterm of a `send()` or the channel-part subterm of a `receive()`. Suppose further that $s : n$ was directly produced by some $r : m$ in a thread map Π''' preceding Π'' in the evaluation. Then $\mathbf{proc}_{\Pi''}(s) = \mathbf{proc}_{\Pi'''}(r)$.

Assume instead that $\mathbf{proc}_{\Pi''}(s) \neq \mathbf{proc}_{\Pi'''}(r)$. Then at least one of the following must occur in the evaluation $\Pi''' \underset{con}{\longrightarrow}^+ \Pi''$:

1. an occurrence of k was a subterm of an `rfork()`'d function
2. an occurrence of k was a subterm of a function transmitted between threads on different processors
3. an occurrence of k itself was transmitted between threads on different processors

Each of (1), (2), and (3) represents a $\underset{con}{\longrightarrow}$-transition with an occurrence of k in a thread on the left-hand side. Consider the family of annotation maps for the thread map on the left-hand side of the transition, given by updating the family of annotation maps at each preceding step, starting with $\mathcal{G}$. We claim that in each of these situations, the σ annotation for the occurrence of k must be **reach**$^{\pm}$. By iterating Theorem 2, we can produce a locally consistent family of annotation maps for Π''. In Π'', since k is a channel-part subterm of a `send()` or `receive()`, it has a σ annotation of **reach**$^{\pm}$. The occurrence of k in (1), (2), or (3) is an occurrence predecessor of the k in Π''. By iterating Lemma 6(4), and using the $\vec{\leq}$-order on propositions, the occurrence of k in (1), (2), or (3) also has a σ annotation of **reach**$^{\pm}$, as claimed.

Suppose (1) happens.

By Theorem 2, we can obtain a locally-consistent annotation map for the thread containing the `rfork()`. By local consistency, the label of the `rfork()`'d function must have been an element of ϕ for the function-part of the `rfork()`. As we have stated, the σ annotation for the occurrence of k in that function was **reach**$^{\pm}$. By local consistency, that occurrence of k also had a ω-annotation of **local**$^{\pm}$. By Lemma 6(4) and the transitivity of the $\vec{\leq}$ relation on propositions, any occurrence predecessor of this k also had a ω-annotation of **local**$^{\pm}$. In particular, the occurrence of k in the thread map where it was created had a ω-annotation of **local**$^{\pm}$. By the specification of annotation map updates, the occurrence $r : m$ of `channel()` also had a ω-annotation of **local**$^{\pm}$. By Lemma 6(4) and the transitivity of the $\vec{\leq}$ relation on propositions, any occurrence predecessor of $r : m$, including $p : i$, also must have had a ω-annotation of **local**$^{\pm}$. This contradicts our supposition that $p : i$ had a ω annotation of **local**$^{+}$. So (1) cannot happen.

Suppose (2) happens.

Consider the transmitted function that is the body of the `send()`. By the κ-coherence and communicative consistency of the family of annotation maps in which the `send()`'ing thread occurs, the θ annotation for the transmitted function is **escape**$^{\pm}$. So by the local consistency of the annotation map for that thread, the ω annotation for the occurence of k in the function body must be **local**$^{\pm}$. From here, we can reason as in the previous case to derive a contradiction. Therefore, (2) cannot happen.

Suppose (3) happens.

Consider the occurrence of the `receive()` that participates in the communication. By the κ-coherence and communicative consistency of the family of annotation maps in which the `send()`'ing thread occurs, if the σ annotation for the `receive()` is **reach**$^{\pm}$, then its ω annotation is **local**$^{\pm}$. By the specification of annotation map updates, the proposition for a transmitted value on the right-hand side of a **comm** transition is the same as that for the participating `receive()` occurrence. We have shown that the σ annotation for the channel constant on the right-hand side of the transition is **reach**$^{\pm}$. Therefore, the σ annotation for the `receive()` on the left-hand side of the transition is **reach**$^{\pm}$. By the κ-coherence and communicative consistency of the family of annotation maps in which the `receive()` thread occurs, the `receive()`'s ω annotation is $\mathbf{local}^{\pm}$. Therefore, the ω annotation for the channel constant k on the right-hand side of the transition is also $\mathbf{local}^{\pm}$. Again we can derive a contradiction, so (3) cannot happen. ∎

14 Related work

Many authors have described constraint-based static analyses. Besides our own work already cited, see, for example, Heintze's Ph.D. work on set-based analysis [2] and Palsberg and Schwartzbach's safety analysis [11].

Most work on static analysis for concurrent programs is relatively new. Peng and Purushothaman presented a dataflow analysis to detect deadlock for communicating automata [12]. Reif and Smolka describe analyses for a language of communicating processes with both fixed and dynamic communication topologies [13]. Their methods are extensions of conventional dataflow analysis techniques for sequential languages. Mercouroff used abstract interpretation to determine which processes may communicate in a CSP-like language [7]. More recently, Jagannathan and Weeks have used abstract interpretation to analyze a higher-order language with communication using shared locations [3]. Nielson and Nielson have done a variety of recent work using enhanced type systems [10][9]. Colby uses abstract interpretation to determine the communications topology of concurrent programs [1]. Our κ annotations give us similar information, although Colby's is a finer, polyvariant analysis.

The linearity component of our propositions is similar in spirit to the usage analysis of Turner, et al., which tracks the number of uses of a term using an extended Hindley-Milner type system [18], although our purposes are quite different.

To aid in source-level debugging of programs, Tip associates terms with earlier terms in a rewrite system [17]. This technique appears similar to our use of occurrence predecessors.

15 Conclusion

We have given a constraint-based static analysis for programs in a concurrent language that is provably sound. As far as we know, this is the first constraint-based static analysis for a concurrent language. Our proof method relies on showing that given a solution to the constraints for the original program, we may produce solutions at every transition step. As usual in such analyses, there are constraints between the annotations of occurrences in individual terms. Because data flows between threads, there are also constraints between the annotations of occurrences in different threads.

We think this style of proof can support many different analyses for languages with small-step operational behavior, whether sequential or concurrent. An open challenge is whether such an analysis can justify more complex program transformations.

Our analysis detects occurrences of the `channel()` primitive that produce dynamic channels certain to be used only on the same processor where they are created. Another useful analysis would be to detect local references – those references which are read or mutated only on the processor where they are created. Such an analysis would be very similar to the one here.

16 Acknowledgments

David N. Turner pointed us to the right notion of locality and convinced us to formulate the analysis using a value-passing operational semantics.

References

[1] Christopher Colby. Analyzing the communication topology of concurrent programs. In *ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manupulation*, pages 202–14, June 1995.

[2] Nevin Heintze. *Set Based Program Analysis.* PhD thesis, Carnegie-Mellon University, October 1992.

[3] Suresh Jagannathan and Stephen Weeks. Analyzing stores and references in a parallel symbolic language. In *Proc. ACM Conf. on Lisp and Functional Programming*, pages 294–306, 1994.

[4] Karoline Malmkjaer, Nevin Heintze, and Olivier Danvy. ML partial evaluation using set-based analysis. In *1994 ACM SIGPLAN Workshop on ML and its Applications*, pages 112–19, Orlando, Florida, June 1994.

[5] David C.J. Matthews. A distributed concurrent implementation of standard ML. LFCS Report Series ECS-LFCS-91-174, University of Edinburgh, August 1991.

[6] David C.J. Matthews and Thierry Le Sergent. LEMMA: A distributed shared memory with global and local garbage collection. In Henry Baker, editor, *Proc. Intl. Workshop in Memory Management*, Lecture Notes in Computer Science, pages 297–311. Springer-Verlag, September 1995.

[7] N. Mercouroff. An algorithm for analyzing communicating processes. In *Mathematical Foundations of Programming Semantics*, volume 598 of *Lecture*

Notes in Computer Science, pages 312–25, Berlin, Heidelberg, and New York, 1991. Springer-Verlag.

[8] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, Cambridge, MA, 1989.

[9] Flemming Nielson and Hanne Riis Nielson. Constraints for polymorphic behaviours of concurrent ML. In *Proc. Constraints in Computational Logic '94*, number 845 in Lecture Notes in Computer Science, pages 73–88, Berlin, Heidelberg, and New York, September 1994. Springer-Verlag.

[10] Hanne Riis Nielson and Flemming Nielson. Higher-order concurrent programs with finite communication topology. In *Conference Record of 21st ACM Symposium on Principles of Programming Languages*, pages 84–97, January 1994.

[11] Jens Palsberg and Michael I. Schwartzbach. Safety analysis versus type inference. *Information and Computation*, 118(1):128–41, April 1995.

[12] Wuxu Peng and S. Purushothaman. Towards dataflow analysis of communicating finite state machines. In *Proc. 8th Annual ACM Symposium on Principles of Distributed Computing*, pages 45–58, August 1989.

[13] John H. Reif and Scott A. Smolka. Data flow analysis of distributed communicating processes. *Intl. J. of Parallel Programming*, 19(1):1–30, 1990.

[14] J.H. Reppy. *Higher-Order Concurrency*. PhD thesis, Cornell University, 1992. Report 92-1285.

[15] Paul Steckler and Mitchell Wand. Selective thunkification. In Baudouin Le Charlier, editor, *Proceedings of the 1st International Static Analysis Symposium*, volume 864 of *Lecture Notes in Computer Science*, pages 162–78, Berlin, Heidelberg, and New York, 1994. Springer-Verlag.

[16] Paul Steckler and Mitchell Wand. Tracking available values for lightweight closures (summary). In Neil Jones and Carolyn Talcott, editors, *Proc. Atlantique Workshop on Semantics Based Program Manipulation*, pages 63–70, 1994. Available as DIKU Report No. 94/12, University of Copenhagen.

[17] Frank Tip. Generic techniques for source-level debugging and dynamic program slicing. In Peter D. Mosses, Mogens Nielsen, and Michael I. Schwartzbach, editors, *Proc. TAPSOFT '95*, number 915 in Lecture Notes in Computer Science, pages 516–30, Berlin, Heidelberg, and New York, May 1995. Springer-Verlag.

[18] David N. Turner, Philip Wadler, and Christian Mossin. Once upon a type. In *Proc. 7th Intl. Conf. on Functional Programming and Computer Architecture*, pages 1–11, June 1995.

[19] Mitchell Wand and Paul Steckler. Selective and lightweight closure conversion. In *Conference Record of 21st ACM Symposium on Principles of Programming Languages*, pages 435–45, 1994.