A Sip of the Chalice

Azalea Raad
Imperial College London
azalea@doc.ic.ac.uk

Sophia Drossopoulou
Imperial College London
scd@doc.ic.ac.uk

ABSTRACT

Chalice is a verification tool for object-based concurrent programs. It supports verification of functional properties of the programs as well as providing a deadlock prevention mechanism. It is built on Implicit Dynamic Frames, fractional permissions and permission transfer.

Implicit Dynamic Frames have been formulated and proven sound using verification conditions and axiomatisation of the heap and stack. Verification in Chalice is specified in terms of weakest preconditions and havocing the heap.

In this paper we give a formalisation of the part of Chalice concerned with functional properties. We describe its operational semantics, Hoare logic and sketch the soundness proof. Our system is parametric with respect to the underlying assertion language.

1. INTRODUCTION

Chalice [5, 6] is a verification tool used for multi-threaded, object-based programs whose methodology centres around fractional permissions and permission transfer[2]. Chalice was built on *Implicit Dynamic Frames* (IDFs)[8] and was extended to concurrent programs.

Implicit dynamic frames have been defined in terms of weakest preconditions and have been proven sound in [8]. The meaning of the underlying IDF-based assertions has been explored and compared with separation logic based assertions in [7].

The implementation and verification conditions for Chalice have been informally described in [5]. To our knowledge, the exact syntax and semantics of Chalice assertions has not been formally stated, and differs from that in [7]; also no work has been undertaken to demonstrate soundness.

Our Contribution We focus on the part of Chalice that is concerned with functional correctness of programs and leave out those features pertaining to the deadlock prevention. For brevity we refer to this system as Chalice^f .

We give an operational semantics which distinguishes "real" operations, such as updating a field or locking an object, from "ghost" operations, such as permission transfer between threads. In our opinion, this treatment clarifies the implicit argument underpinning the soundness.

We find verification conditions/weakest preconditions somewhat indirect, and therefore we develop a Hoare logic for Chalice^f .

Because the meaning of assertions may vary across systems, we do not give a complete syntax for assertions. Instead, our model is parametric with respect to the assertions and their validity, provided these satisfy some necessary requirements that we have distilled. Thus we are agnostic as to the meaning of some of the logical connectives (eg $\rightarrow$), or validity of user-defined predicates, and we hope that our model can be applied to a family of assertion languages.

The rest of this paper is organised as follows. In section 2, we present a more detailed description of Chalice's methodology and its features. In section 3, we give the syntax of Chalice^f and its Hoare logic. We describe its operational semantics and state our assumptions about the assertions language. We sketch the soundness proof of Chalice^f and state the supporting lemmas. We discuss future work and conclude in section 4.

2. INTRODUCTION TO Chalice^f

In this section we introduce the part of Chalice that is concerned with the functional correctness of programs.

2.1 Permissions

A heap location may only be accessed by a thread if it has sufficient permissions. Chalice employs Boyland's fractional permissions[2] which allow the access permission to a memory location to be split up amongst several threads, thus enabling concurrent reading of that location. A thread can only write to a location if it has exclusive access to that location indicated by $\mathbf{1}$. On the other hand, any non-zero permission to a location is sufficient for a thread to read that location. A permission of n to field f of x is denoted by acc(x.f, n) where $0 \leq \mathsf{n} \leq 1$.

Methods and Permissions Transfer A method may only access a memory location if it has the permission to do so. The precondition of a method specifies all the permissions that it requires from its caller and in its postcondition the permissions that it returns to its caller. For instance, consider the code in Figure 1. Method `copy`, assigns the value of `y.f` to the field `f` of the caller. Since it *writes* to `this.f` and *reads* from `y.f` it requires write and read accesses to these locations respectively and hence the method contract includes `requires acc(this.f, 1) && acc(y.f,`

FTfJP'11, July 26, 2011, Lancaster, UK.

```
class C {
  var f:int;

  method copy(y)
    requires acc(this.f, 1) && acc(y.f, 0.1);
    ensures acc(this.f, 1) && acc(y.f, 0.1);
    {this.f := y.f;}

  method readOnly()
    requires acc(this.f, 1);
    ensures acc(this.f, 0.8);
    {this.f:= 7}
}
```

Figure 1: Illustration of permission transfer

0.1).

Once the execution of this method has ended, these permissions are handed back to the caller. In other words, the caller loses full access to `this.f` and partial access to `y.f` for the duration of the call. It does however regain these permissions upon exit from `copy`.

The permissions specified in the precondition of the `readOnly` method can be justified in the similar manner. This method requires full access to the `this.f` location. However, upon exit from this method only a partial access is given back to the caller and the remaining permission is retained by the system. That is, calling this method deems `this.f` immutable and no other thread can acquire full access to that location any more.

2.2 Threads

We now discuss how Chalice uses threads and how permissions are transferred when threads are forked and joined.

Creation of New Threads A `fork` statement creates a new thread and specifies the method to be executed upon creation of the new thread. For instance,

`fork tk:=x.m(`$\overline{y}$`);`

creates a new thread to execute the method `m` on `x` with $\overline{y}$ parameters. Furthermore, it creates a token that can be stored in the stack frame as a variable (`tk`) which can then be used to identify the thread it is associated with. Similar to method calls, upon forking a new thread the current thread is stripped from the permissions required by the precondition of the target method. These permissions are then granted to the newly forked thread.

One thread can wait for another to complete using the `join` statement. For instance, through

`join tk;`

the current thread waits for the thread associated with `tk` to complete and then terminates it. Just like method calls, upon termination of the forked thread the permissions in the postcondition of the target method are handed back to the joining thread. In fact, a method call can best be described as a `fork` immediately followed by a `join` [6].

Note that the way Chalice handles thread creation ensures freedom from data races. For instance, consider the `copy` method in Figure 1. While a thread is executing this method, no other thread can read or write to `this.f`, since the current thread has full permission to this location. This ensures that no other thread can read from a location we are currently modifying and hence there will be no data races.

```
class List{
  var length:int;
  invariant acc(this.length, 1) && length >= 0
}
```

Figure 2: Illustration of class invariants

On the other hand, other threads can read from `y.f` as the current thread only holds a read permission to this location. Note that having a read permission to a location indicates that no other thread can have full access to this location and therefore cannot modify it. Therefore, the value of a location that we are currently reading from cannot be altered by other threads.

2.3 Monitors

In Chalice, each class is associated with an *invariant* which can both hold access permissions (accessibility assertions) and describe properties about the objects of this kind (pure assertions). For instance, class `List` in Figure 2 declares a monitor invariant that states that the monitor of each `List` object holds full permission to its `length` field and that its length is non-negative.

When an object is shared amongst several threads, its monitor can be treated as a mutual exclusion lock to synchronise data access. A thread can compete for exclusive access to a shared memory location using the `acquire` statement. Once a thread has successfully obtained the monitor of an object, it is granted the permissions held by the object's monitor. When the thread no longer needs the exclusive access to the location, it releases the lock using the `release` statement. At this point, the target object becomes shared once again and the permissions originally held by the object's monitor are stripped from the current thread and are handed back to the object monitor.

States of an Object In Chalice, an object can be in one of three states: *available*, *held* and *not-a-monitor*. However, since the distinction between the *held* and *not-a-monitor* states lies in the deadlock prevention mechanism, we only consider the *available* and *held* states in Chalice^f .

When an object is newly allocated, it is in the *held* state and its monitor is held by the thread that created it. The `release` statement can be used to transfer an object from the *held* state to the *available* state. It checks that the monitor invariant holds and then transfers the permissions required by the monitor invariant to the object monitor. In the *available* state, the monitor invariant holds and the `acquire` statement can be used to transition the object to the *held* state.

3. FORMAL SYSTEM

In this sections we define the syntax, operational semantics and Hoare logic for Chalice^f , and prove its soundness.

3.1 Syntax

We give the syntax of Chalice^f in Figure 3.

Classes As with other object-oriented languages, classes keep track of fields and methods. Moreover, each class is associated with an assertion `(A)` corresponding to the *invariant* of the class. The invariant asserts certain properties about objects of this type that hold when the object is in the *available* state.

```
t::= CId                prog ::= class
class ::= CId → A × (FId → t) × (MId → meth)
meth ::= void m (t x) (requires A ensures A)  {e}
e::= e;e | new CId() | x.f:=y | x:=y.f | x.m(y)
    | if(B) {e} else {e} | acquire x | release x
    | TkId := fork x.m(y) | join TkId
B::= ...
A::= acc(x.f,n) | x.f=y | A*A | A∧A | ¬A | true |
...
```

Figure 3: Syntax of Chalice^f

Assertion Language and required Properties As discussed earlier, we do not define the syntax of assertions in Chalice^f . Thus our model is parametric with respect to assertions and their validity. However, we requite these to satisfy certain properties:

The assertion language is required to support at least the syntax required in Fig. 3, i.e accessibility assertions `acc(x.f, n)`, assertions inspecting the values of a memory locations `x.f = v`, and also to support the connectives $\wedge$, $\neg$, and $*$. We require these to have the expected meaning, as described in section 3.3. We are agnostic as to further connectives.

Moreover, we assume that our assertion language includes an inference system whereby an assertion can be derived from another, written as $\mathsf{A} \rightarrow_{\mathsf{a}} \mathsf{B}$. The formal definition of all our requirements is given in section 3.3.

Expressions We provide standard object-oriented expressions such as field access, method call and conditionals. We allow locking of objects through the `acquire` and `release` expressions. Creation and termination of new threads is achieved by using the `fork` and `join` commands. The syntax of the boolean expressions (b) is as expected.

3.2 Runtime Environment

We specify the runtime configurations in Figure 4.

Processes A process in our system is a triple $(\mathsf{e}, \tau, \sigma)$ where e denotes the expression to execute, τ is the thread identifier and σ is the stack frame. We characterise multiple threads by $(P|P)$.

Heap An object address in the heap is associated with its class, values of its fields and the state of its monitor. The monitor of an object contains a `ProcId` ranged over by τ indicating the thread that currently holds the monitor of the object. We use τ_g to represent a *ghost* thread that holds the monitor of an object whenever it is in the *available* state. That is, we assume the existence of a global thread τ_g that holds the monitor of all objects that are in the *available* state.

Heaps also keep track of the token information. Upon execution of a fork statement `fork tk` $:= \mathtt{x.m'}(\overline{\mathtt{y}})$, a new address in the heap is associated with (`tk`) and this address holds information about the forked thread. This information is a tuple of the form `(TK, {c:C, m:m', args:`$\iota.\bar{\iota}$`}, `τ`)` where τ is the identifier of the forked thread, `m` stores the identifier of the method to execute, `C` denotes the class identifier in which `m'` can be found, ι is the address of `x` and $\bar{\iota}$ are the addresses of $\overline{\mathtt{y}}$.

Permission Map and Permission Mask A *Permission map* (Π) keeps track of the permissions associated with each thread. Each thread is mapped onto a *permission mask* π. A permission mask associates a permission with each pair of object address and field identifier or token address and field identifier. When $\Pi(\tau)$ maps $(\iota, \mathtt{f})$ to n, then τ holds a permission of n to the ι.f location.

```
P::= (e, ProcId, σ) | P|P
H : ObjAddr → obj ∪ TkAddr → tk
obj :  CId × (FId → value) × ProcId
tk :  {TK} × (FId → value) × ProcId
Π:  ProcId → pMask
pMask : ObjAddr × FId → n  ∪  TkAddr × FId → n
        (n : Rational ∧ 0 ≤ n ≤ 1)
σ::= (Var → value) ∪ (TkId → TkAddr)
value::= null | ObjAddr | CId | MId | ObjAddr
```

Figure 4: Runtime Configuration

Stack Frame A stack frame σ maps variables and token identifiers to their values.

3.3 The assertion language

Validity Judgement and Permission Function We assume the existence of a judgement $\models$ of shape

$\mathsf{H}, \pi, \sigma \models \mathsf{A}$

which asserts the validity of assertions given a heap H, a permission mask π and a stack σ. We also assume the existence of a function $\mathcal{P}$,

$\mathcal{P} :: \mathsf{Heap} \times \mathsf{Stack\ Frame} \times \mathsf{A} \rightarrow \mathsf{pMask}$

which calculates the permissions required to satisfy an assertion. We also require a judgment

$\mathsf{A} \rightarrow_{\mathsf{a}} \mathsf{A}$

which determines when an assertion can be inferred from another.

We require that $\models$ has the following properties:

R1. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \ \wedge\ \overline{\mathsf{y}} = \mathsf{fv(A)} \ \wedge\ \sigma'(\overline{\mathsf{x}}) = \sigma(\overline{\mathsf{y}})$
$\implies \mathsf{H}, \pi, \sigma' \models \mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{y}}]$

R2. $\mathsf{H}, \pi, \sigma \models \mathsf{x.f} = \mathsf{y} \iff \mathsf{H}(\sigma(\mathsf{x}), \mathsf{f}) = \sigma(\mathsf{y})$

R3. $\mathsf{H}, \pi, \sigma \models \mathsf{acc(x.f, n)} \iff \pi(\sigma(\mathsf{x}), \mathsf{f}) \geq \mathsf{n}$

R4. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \wedge \mathsf{A}' \iff \mathsf{H}, \pi, \sigma \models \mathsf{A}$ and $\mathsf{H}, \pi, \sigma \models \mathsf{A}'$

R5. $\mathsf{H}, \pi, \sigma \models \mathsf{A} * \mathsf{A}' \implies$
$\forall(\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f}) + \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}')(\iota.\mathsf{f}) \leq 1]$
$\wedge\ \forall(\kappa)[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\kappa.\mathsf{g}) + \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}')(\kappa.\mathsf{g}) \leq 1]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

R6. $\mathsf{H}, \pi, \sigma \models \mathsf{true}$

We require that $\mathcal{P}$ has the following properties:

R7. $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}) = \mathcal{P}(\mathsf{H}, \sigma[\overline{\mathsf{y}} \mapsto \overline{\sigma(\mathsf{x})}], \mathsf{A}[\overline{\mathsf{y}}/\overline{\mathsf{x}}])$

R8. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \implies \mathsf{H}, \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}), \sigma \models \mathsf{A}$
$\wedge\ \forall(\iota.\mathsf{f})[\pi(\iota.\mathsf{f}) \geq \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f})]$

R9. $\forall \mathsf{H}, \sigma, \mathsf{A}, (\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f}) \neq \mathsf{Udf} \implies \iota.\mathsf{f} \in \mathsf{Dom(H)}]$
$\wedge\ \forall \mathsf{H}, \sigma, \mathsf{A}, (\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\kappa.\mathsf{g}) \neq \mathsf{Udf} \implies \kappa.\mathsf{g} \in \mathsf{Dom(H)}]$

We require that $\rightarrow_a$ has the property:

R10. $\mathsf{A} \rightarrow_{\mathsf{a}} \mathsf{A}' \quad \wedge \quad \mathsf{H}, \pi, \sigma \models \mathsf{A} \implies \mathsf{H}, \pi, \sigma \models \mathsf{A}'$

Self-framing Assertions We call an assertion self-framing if its validity is preserved in all heaps which agree in the locations mentioned in the permissions.

$\mathsf{SF(A)} \iff$

$\forall\ \mathsf{H}, \mathsf{H}', \pi, \sigma\ [\mathsf{H}, \pi, \sigma \models \mathsf{A}\ \wedge\ \mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{A})}{\equiv} \mathsf{H}'\ \Longrightarrow\ \mathsf{H}', \pi, \sigma \models \mathsf{A}]$
The $\overset{\pi}{\equiv}$ notation indicates agreement between two heaps and is defined in Appendix A[1].
We require for self-framing assertions that:
R11. If A_1 and A_2 are self-framing, then
$\mathsf{H}, \pi, \sigma \models \mathsf{A}_1 * \mathsf{A}_2\ \Longleftrightarrow$
$\exists \pi_1, \pi_2.[\mathsf{H}, \pi_1, \sigma \models \mathsf{A}_1\ \wedge\ \mathsf{H}, \pi_2, \sigma \models \mathsf{A}_2,\ \wedge\ \pi = \pi_1 \uplus \pi_2]$
where $\pi_1 \uplus \pi_2$ is as defined in Appendix A.
R12. A, A' is self framing, then $\mathsf{A} * \mathsf{A}'$ is also self-framing.

R13. We call an assertion B a *case-split assertion*, if
$\mathsf{SF}(\mathsf{B} \wedge \mathsf{A})\ \Longrightarrow\ \mathsf{SF}(\mathsf{A})$
$\wedge\ [\mathsf{H}, \pi, \sigma \models \mathsf{A}\ \Longrightarrow\ \mathsf{H}, \pi, \sigma \models \mathsf{B}\ \vee\ [\mathsf{H}, \pi, \sigma \models \neg \mathsf{B}].$

The above properties allow us to be agnostic as to the exact meaning of further logical connectives, such as $\rightarrow$, $-\!\!*$, $*\!\!-$ and user-defined predicates.

Validity Judgement from the viewpoint of a thread
We define the judgement $\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{A}$ such that:
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{A}\ \Longleftrightarrow\ \mathsf{H}, \Pi(\tau), \sigma \models \mathsf{A}$

3.4 Hoare Logic

We have formalised the verification conditions of Chalice through Hoare logic as illustrated in Figure 5. We now look at some of the more interesting rules.

New When a new object is created, the current thread is granted the full access to all of its fields and all fields are initialised with the `null` value .

Acquire and Release When the monitor of an object is acquired by a thread, its invariant can be assumed to hold. This is reflected in the postcondition (right hand side) of the **Acq.** rule. Note that this also implicitly gives access to all permissions required by that invariant.

On the other hand, an object may only be released if its invariant holds. In the **Rel.** rule, A represents the invariant of x. After releasing an object, the current thread may no longer assume its invariant to hold, this is reflected in the Hoare logic by the empty postcondition of **Rel**.

Fork and Join When forking a method to execute $\mathsf{x.m}(\overline{\mathsf{y}})$, the precondition of m must hold. This has been reflected in the precondition of the **Fork** rule. The postcondition of this rule guarantees that the variable `tk` correctly reflects the specification of the new thread.

The **Join** rule guarantees that the postcondition of the method holds, where its postcondition and actual parameters are reflected in the contents of `tk`.

We have used the shorthand
$\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}) \equiv\ \mathsf{acc}(\mathsf{tk.c}, 1)\ *\ \mathsf{tk.c} = \mathsf{C}$
$* \mathsf{acc}(\mathsf{tk.m}, 1)\ *\ \mathsf{tk.m} = \mathsf{m}$
$* \mathsf{acc}(\mathsf{tk.args}, 1)\ *\ \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$

Connection to Chalice Verification Conditions The Hoare logic judgements given in Figure 5, correspond to the verification conditions described in [5] using the *exhale* and *inhale* commands. `exhale A` has the effect of generating `assert A` statement and giving up any accessibility predicates in `A`. If the verification condition for a command `C` is `exhale A`, the corresponding Hoare triplet is `{A} C {}`. Mutatis mutandis for `inhale A`.

Note that the verification conditions given in [5] include additional checks missing from our Hoare logic. For instance, in order to verify a `release` statement, it is *asserted* that the object released is held by the current thread to ensure the *progress* of the system. However, in our model of Chalice^f , we are not concerned with the progress property and our Hoare logic does not reflect these constraints.

3.5 Rewriting Rules

We have divided the rewriting rules of Chalice^f into two parts, the *operational semantics* corresponding to the "real" execution of the program and the *permission passing semantics*, reflecting all ghost operations necessary for the soundness argument. That is, permission passing and keeping track of the contents of the tokens.

The operational semantics for one thread has the following format and thus keeps track of the changes applied to the contents of the thread and the heap.
$$\mathsf{P}, \mathsf{H} \rightsquigarrow \mathsf{P}', \mathsf{H}'$$
For n threads we have:
$\overline{\mathsf{P}}^{1\cdots\mathsf{n}}, \mathsf{H} \rightsquigarrow \overline{\mathsf{P}'}^{1\cdots\mathsf{m}}, \mathsf{H}'$ where $\mathsf{m} \in \{\mathsf{n}, \mathsf{n}+1, \mathsf{n}-1\}$.
The cases where $\mathsf{m} = \mathsf{n}{+}1$ and $\mathsf{m} = \mathsf{n}{\text{-}}1$ correspond to the execution of `fork` and `join` statements, respectively.

On the other hand, the permissions semantics for one thread has the following format and thus keeps track of the changes in permissions.
$$\mathsf{P}, \mathsf{H}, \Pi \rightsquigarrow \mathsf{H}', \Pi',$$
Similarly, for multiple threads we have:
$\overline{\mathsf{P}}^{1\cdots\mathsf{n}}, \mathsf{H}, \Pi \rightsquigarrow \mathsf{H}', \Pi'.$

3.6 Operational Semantics

We describe the operational semantics of Chalice^f in Figure 6. For brevity, we have omitted the judgements pertaining to the execution of boolean expressions and assume that they are as expected. We now explain some of the more interesting rules.

Acquire and Release The monitor of an object can only be acquired by a thread if the object is in the *available* state, that is its monitor contains the ghost thread τ_g. Upon successful acquisition of the lock, the monitor of the object is modified in the new heap to contain the current thread τ.

Conversely, an object can only be released by a thread if it is already held by that thread. That is, the monitor of the object contains the current thread τ. Once the object is released its monitor is modified to contain the ghost thread τ_g.

Fork and Join When a new thread is forked in association with token tk to execute $\mathsf{x.m}(\overline{\mathsf{y}})$, tk is mapped to a new address in the heap to hold the identifier of the newly forked thread. Moreover, a new stack frame is created for the new thread to reflect the values of the method receiver and arguments. The newly created thread then proceeds with the execution of the corresponding method.

On the other hand, when joining a thread through a token tk, we first check the identifier of the joined thread against that associated with tk. We then remove the information associated with that token from the heap.

Method Calls Upon a method call $\mathsf{x.m}(\overline{\mathsf{y}})$, a new thread is forked to execute $\mathsf{x.m}(\overline{\mathsf{y}})$ and is immediately joined after.

3.7 Permission Passing Semantics

[1]Note that our definition of self-framing is inspired from, but slightly different than that from [7].

FAss $\{\mathsf{acc}(\mathsf{x.f}, 1)\}\ \mathsf{x.f} := \mathsf{y}\ \{\mathsf{acc}(\mathsf{x.f}, 1) * \mathsf{x.f} = \mathsf{y}\}$

VAss $\dfrac{\mathsf{z} > 0}{\{\mathsf{acc}(\mathsf{x.f}, \mathsf{z})\}\ \mathsf{y} := \mathsf{x.f}\ \{\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}\}}$

Val. $\dfrac{\mathsf{SF(P)}}{\{\mathsf{P}\}\ \mathsf{v}\ \{\mathsf{P}\}}$

Seq. $\dfrac{\{\mathsf{P}\}\ \mathsf{C}_1\ \{\mathsf{Q}\} \qquad \{\mathsf{Q}\}\ \mathsf{C}_2\ \{\mathsf{R}\}}{\{\mathsf{P}\}\ \mathsf{C}_1; \mathsf{C}_2\ \{\mathsf{R}\}}$

Fork $\dfrac{\mathsf{Pre(m)} = \mathsf{P}(\overline{\mathsf{u}}) \qquad \mathsf{x.class} = \mathsf{C}}{\{\mathsf{P}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}\ \mathbf{fork\ tk} := \mathbf{x.m}(\overline{\mathsf{y}})\ \{\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})\}}$

Join $\dfrac{\mathsf{Post(m)} = \mathsf{A}(\overline{\mathsf{u}})}{\{\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})\}\ \mathsf{join\ tk}\ \{\mathsf{Post(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}}$

If B is a case-split assert.
$\dfrac{\{\mathsf{B} \wedge \mathsf{P}\}\ \mathsf{C1}\ \{\mathsf{Q}\} \quad \{\neg\mathsf{B} \wedge \mathsf{P}\}\ \mathsf{C2}\ \{\mathsf{Q}\}}{\{\mathsf{P}\}\ \mathsf{if(B)\ then\ C1\ else\ C2}\ \{\mathsf{Q}\}}$

Meth $\dfrac{\mathsf{Pre(m)} = \mathsf{P}(\overline{\mathsf{u}}) \qquad \mathsf{Post(m)} = \mathsf{Q}(\overline{\mathsf{w}})}{\{\mathsf{P}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}\ \mathbf{x.m}(\overline{\mathsf{y}})\ \{\mathsf{Q}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{w}}]\}}$

New $\dfrac{\mathsf{f_i} \in \mathsf{FS(C)}}{\{\}\ \mathsf{x} := \mathsf{new\ C}\ \{*\mathsf{acc}(\mathsf{x.f_i}, 1) * \mathsf{x.f_i} = \mathsf{null}\}}$

Acq. $\{\}\ \mathsf{acquire\ x}\ \{\mathsf{A}[\mathsf{x/this}]\}$

Rel. $\{\mathsf{A}[\mathsf{x/this}]\}\ \mathsf{release\ x}\ \{\}$

Con. $\dfrac{\mathsf{SF(P)} \qquad \mathsf{SF(Q)} \qquad \mathsf{P} \rightarrow_a \mathsf{P}' \qquad \{\mathsf{P}'\}\ \mathsf{C}\ \{\mathsf{Q}'\} \qquad \mathsf{Q}' \rightarrow_a \mathsf{Q}}{\{\mathsf{P}\}\ \mathsf{C}\ \{\mathsf{Q}\}}$

Frm $\dfrac{\{\mathsf{P}\}\ \mathsf{C}\ \{\mathsf{Q}\} \quad \mathsf{SF(R)}}{\{\mathsf{P} * \mathsf{R}\}\ \mathsf{C}\ \{\mathsf{Q} * \mathsf{R}\}}\ \mathsf{FV(R)} \cap \mathsf{Mods(C)} = \emptyset$

Figure 5: Hoare Logic for Chalice^f commands; A represents the invariant of object x in the acquire and release judgements and v represents a value.

FAssO $\dfrac{\sigma(\mathsf{x}) = \iota \qquad \mathsf{H}' = \mathsf{H}[\iota \mapsto \mathsf{H}(\iota)[\mathsf{f} \mapsto \sigma(\mathsf{y})]]}{(\mathsf{x.f} := \mathsf{y}\ ,\ \tau, \sigma), \mathsf{H} \rightsquigarrow (\sigma(\mathsf{y}), \tau, \sigma), \mathsf{H}'}$

VAssO $\dfrac{\sigma(\mathsf{x}) = \iota \qquad \mathsf{H}(\iota) \downarrow_2 (\mathsf{f}) = \mathsf{v} \qquad \sigma' = \sigma[\mathsf{y} \mapsto \mathsf{v}]}{(\mathsf{y} := \mathsf{x.f},\ \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{v}, \tau, \sigma'), \mathsf{H}}$

IfTO $\dfrac{}{(\mathsf{if(true)then\{e2\}else\{e3\}}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{e2}, \tau, \sigma), \mathsf{H}}$

IfFO $\dfrac{}{(\mathsf{if(false)then\{e2\}else\{e3\}}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{e3}, \tau, \sigma), \mathsf{H}}$

ValO $\dfrac{}{(\mathsf{v}; \mathsf{e}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{e}, \tau, \sigma), \mathsf{H}}$

MethO $\dfrac{\mathsf{tk} \notin \sigma}{(\mathsf{x.m}(\overline{\mathsf{y}}), \tau, \sigma), \mathsf{H} \rightsquigarrow ((\mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}); \mathsf{join\ tk}), \tau, \sigma), \mathsf{H}}$

SeqO $\dfrac{(\mathsf{e}_1, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{e}_1', \tau, \sigma'), \mathsf{H}'}{(\mathsf{e}_1; \mathsf{e}_2, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{e}_1'; \mathsf{e}_2, \tau, \sigma'), \mathsf{H}'}$

NewO $\dfrac{\mathsf{FS(C)} = \{\mathsf{t}_1\ \mathsf{f}_1 ..., \mathsf{t}_r\ \mathsf{f}_r\} \quad \iota \notin \mathsf{H} \quad \sigma' = \sigma[\mathsf{x} \mapsto \iota] \quad \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{C}, \{\mathsf{f}_1 : \mathsf{null}, ..., \mathsf{f}_r : \mathsf{null}\},\ \tau)]}{(\mathsf{x} := \mathsf{new\ C}\ , \tau, \sigma), \mathsf{H} \rightsquigarrow (\iota, \tau, \sigma'), \mathsf{H}'}$

AcqO $\dfrac{\sigma(\mathsf{x}) = \iota \qquad \mathsf{H}(\iota) \downarrow_3 = \tau_g \quad \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau)]}{(\mathsf{acquire\ x}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}'}$

RelO $\dfrac{\sigma(\mathsf{x}) = \iota \qquad \mathsf{H}(\iota) \downarrow_3 = \tau \quad \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau_g)]}{(\mathsf{release\ x}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}'}$

ForkO $\dfrac{\kappa \notin \mathsf{dom(H)} \quad \tau' \notin \mathsf{range(H)} \quad \mathsf{mBody(m)} = \mathsf{e}(\overline{\mathsf{u}}) \quad \sigma(\mathsf{x}) = \iota \qquad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \qquad \overline{\sigma(\mathsf{y})} = \overline{\iota} \quad \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau')] \quad \sigma' = \sigma[\mathsf{tk} \mapsto \kappa] \qquad \sigma'' = \mathsf{this} \mapsto \iota, \overline{\mathsf{u}} \mapsto \overline{\iota}}{(\mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}), \tau, \sigma), \mathsf{H} \rightsquigarrow ((\mathsf{null}, \tau, \sigma') | (\mathsf{e}, \tau', \sigma'')), \mathsf{H}'}$

JoinO $\dfrac{\mathsf{H}(\sigma(\mathsf{tk})) \downarrow_3 = \tau' \qquad \mathsf{H}' = \mathsf{H}[\sigma(\mathsf{tk}) \mapsto \epsilon]}{(\mathsf{v}, \tau', \sigma') | (\mathsf{join\ tk}, \tau, \sigma), \mathsf{H} \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}'}$

ThrdO $\dfrac{\overline{\mathsf{P}'}, \mathsf{H} \rightsquigarrow \overline{\mathsf{P}''}, \mathsf{H}'}{\overline{\mathsf{P}_1} \mid \overline{\mathsf{P}'} \mid \overline{\mathsf{P}_2}, \mathsf{H} \rightsquigarrow \overline{\mathsf{P}_1} \mid \overline{\mathsf{P}''} \mid \overline{\mathsf{P}_2}, \mathsf{H}',}$

Figure 6: Operational semantics of Chalice^f

RestP
$$\frac{e \in \{\, y := x.f,\ x.f := y,\ \texttt{if}\ldots,\ x.m(\overline{y}) \,\} \quad (e\ ,\ \tau,\sigma), H \rightsquigarrow (e',\tau,\sigma), H'}{(e\ ,\ \tau,\sigma), H \rightsquigarrow H', \Pi}$$

AcqP
$$\frac{\sigma(x) = \iota \quad H(\iota)\downarrow_1 = C \quad Invariant(C) = A \quad \mathcal{P}(H,\sigma,A[x/this]) = ps \quad \Pi' = \Pi[\tau + = ps,\ \ \tau_g - = ps]}{(\text{acquire } x,\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

RelP
$$\frac{\sigma(x) = \iota \quad H(\iota)\downarrow_1 = C \quad Invariant(C) = A \quad \mathcal{P}(H,\sigma,A[x/this]) = ps \quad \Pi' = \Pi[\tau - = ps,\ \ \tau_g + = ps]}{(\text{release } x,\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

ThrdP
$$\frac{P, H, \Pi \rightsquigarrow \overline{P'}, H', \Pi'}{\overline{P_1} \mid P \mid \overline{P_2}, H, \Pi \rightsquigarrow \overline{P_1} \mid \overline{P'} \mid \overline{P_2}, H', \Pi'}$$

NewP
$$\frac{FS(C) = \{t_1\ f_1 \ldots, t_r\ f_r\} \quad \iota = dom(H') \setminus dom(H) \quad \Pi' = \Pi[(\tau)(\iota, f_i) \mapsto 1]_{i\in 1\ldots r}}{(x := \text{new } C\ ,\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

ForkP
$$\frac{\sigma(x) = \iota \quad H(\iota)\downarrow_1 = C \quad pre(C,m) = A(\overline{u}) \quad \mathcal{P}(H,\sigma,A[x/this][\overline{x}/\overline{u}]) = ps \quad \kappa \in H' \setminus H \quad H(\kappa)\downarrow_3 = \tau' \quad \Pi'' = \Pi[\tau' \mapsto ps,\ \ \tau - = ps] \quad \Pi' = \Pi''[(\tau)(\kappa.g) \mapsto 1]\ \ \text{for } g \in \{c, m, args\}}{(\text{fork } tk := x.m(\overline{y}),\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

JoinP
$$\frac{\sigma(tk) = \kappa \quad H(\kappa)\downarrow_3 = \tau' \quad H(\kappa.args) = \iota.\overline{\iota} \quad H(\kappa.c) = C \quad H(\kappa.m) = m \quad Post(C,m) = A(\overline{u}) \quad \mathcal{P}(H,[this \mapsto \iota, \overline{u} \mapsto \overline{\iota}], A(\overline{u})) = ps \quad \Pi' = \Pi[\tau + = ps,\ \ \tau' - = ps]}{(v,\tau',\sigma)|(\text{join } tk,\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

SeqP
$$\frac{(e_1,\tau,\sigma), H, \Pi \rightsquigarrow (e_1',\tau,\sigma'), H', \Pi'}{(e_1; e_2,\tau,\sigma), H, \Pi \rightsquigarrow H', \Pi'}$$

Figure 7: Permission passing semantics of Chalice^f

We describe the permission semantics of Chalice^f in Figure 7. We now explain some of the more interesting rules.

New When a new object is created, the permission mask is modified such that the current thread is given full permissions (denoted by 1) on all its fields.

Acquire and Release When a thread acquires the monitor of an object, the permissions associated with its invariant A are stripped from the global thread τ_g and are granted to the acquiring thread τ. Note that these permissions are recalculated in the heap upon each acquisition. Therefore, it is possible that different sets of address and field pairs are involved when acquiring or releasing the same object at different times during execution.

Dually, when the monitor of an object is released, the permissions associated with its invariant A are stripped from the current thread τ and are granted to the global thread τ_g.

Fork and Join When a new thread is forked to execute $x.m(\overline{y})$, the permissions associated with the precondition of m are calculated and taken away from the forking thread τ and are granted to the new thread τ'.

On the other hand, when joining a thread, the permissions associated with the postcondition of m are calculated and taken away from the forked thread. These permissions are then granted back to the current thread.

3.8 Soundness Theorem

In order to express the argument for soundness more conveniently, we want to be able to talk succinctly about the effect of execution on the heap, as well as the permissions. Therefore, we will use the judgment

$$\overline{P}^{1\ldots n}, H, \Pi \rightsquigarrow \overline{P'}^{1\ldots m}, H', \Pi'$$

as a shorthand for

$$\overline{P}^{1\ldots n}, H \rightsquigarrow \overline{P'}^{1\ldots m}, H' \text{ and } \overline{P}^{1\ldots n}, H, \Pi \rightsquigarrow H', \Pi'$$

This judgment directly follows from Fig. 6 and 7, but for convenience, in the appendix, we given the rules for this judgment in Fig 8 in the appendix.

Well-Formed Configuration A system of n processes is well-formed with respect to a heap and a permission mask if and only if it satisfies the following properties.

$$\begin{aligned}
&WF(H, \Pi, (\overline{e, \tau, \sigma}^{1\ldots n})) \iff \\
&\textbf{a.}\ \forall\, \kappa.[H(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau) \implies \\
&\quad \exists j \in \{1\ldots n\}, \exists P.[\tau = \tau_j \ \wedge\ H, \Pi, \tau_j, \sigma_j \models P \\
&\quad \wedge\ \{P\}\ e_j\ \{Post(m)\} \wedge \sigma_j(this) = \iota \wedge \sigma_j(\overline{u}) = \overline{\iota}\ ] \\
&\textbf{b.}\ \forall \iota.f[\textstyle\sum_{\tau\in DOM(\Pi)} \Pi(\tau)(\iota.f)\ \leq 1] \\
&\quad \wedge\ \forall \kappa.g[\textstyle\sum_{\tau\in DOM(\Pi)} \Pi(\tau)(\kappa.g)\ \leq 1] \\
&\textbf{c.}\ H, \Pi, \overline{x \mapsto \iota}^{1\ldots m}, \tau_g \models \overline{*Invariant(\iota_i.class)[x_i/this]}^{1\ldots m} \\
&\quad \text{for } \overline{\iota}^{1\ldots m} = \{\iota | \iota \in Dom(H) \wedge H(\iota)\downarrow_3 = \tau_g\} \\
&\textbf{d.}\ \forall \kappa, \kappa'[H(\kappa)\downarrow_3 = \tau\ \wedge\ H(\kappa')\downarrow_3 = \tau\ \implies\ \kappa = \kappa']
\end{aligned}$$

In other words, in a well-configured configuration, **a.**, each thread token is associated with a thread present in the system, an assertion that it needs to satisfy upon termination and a set of locations representing the receiver and the arguments; the expression associated with that thread can be verified under a precondition P to lead to a configuration where the postcondition specified holds; the precondition P itself is satisfied in the current configuration.

Moreover, **b.**, the sum of all permissions to a memory location held by the threads present in the system is less than or equal to 1. Also, **c.**, the invariant of those objects in the *available* state holds in disjoint parts of the heap (assume that $x_i \neq x_i$ for $i \neq j$). Finally, **d.**, the mapping from thread tokens to threads is injective.

Program Verification A program is verified if and only if all methods of all its classes are verified.

$$\begin{aligned}
&Ver(Prog) \iff \forall\ C \in classes(Prog),\ \forall m \in methods(Prog, C) \\
&\qquad [SF(A)\ \wedge\ \{P\}\ e\ \{Q\} \wedge\ SF(P)\ \wedge\ SF(Q)] \\
&\text{where}\ \ P \equiv Pre(m) \quad Q \equiv Post(m) \quad e \equiv mBody(m) \\
&\qquad A \equiv Invariant(C)
\end{aligned}$$

Soundness Theorem The verification system of Chalice^f is sound in the sense that if an expression e_0 is verified by Chalice^f and its prescribed precondition is satisfied in the

current heap and permission mask, then the execution of e_0 will result in another expression e'_0, such that the final expression e'_0 itself can be verified using Chalice^f and its precondition is satisfied under the new heap and permission mask. Furthermore, execution of a single thread does not interfere with the state of other threads present in the system. That is, any of the previously verified threads can also be verified in the final configuration. Finally, the well-formedness of the system is preserved under execution. In other words, if the initial configuration of the system is well-formed, it remains well-formed after the execution of an expression.

The execution of an expression e_0 can result reduction to another expression e'_0 without affecting other threads, or in forking new thread, or in joining with another thread. We formalise soundness for the first scenario in the following theorem. Soundness for the latter two scenarios is formalised in Appendix B. The proof of these theorems is provided in Appendix D.

Soundness Theorem 1

$$\mathsf{Ver}(\mathsf{Prog}) \wedge (\{\mathsf{P_i}\}\ \mathsf{e_i}\ \{\mathsf{Q_i}\})_{i\in\{0\ldots n\}} \wedge (\mathsf{H},\Pi,\sigma_i,\tau_i \models \mathsf{P_i})_{i\in\{0\ldots n\}} \wedge (\mathsf{e}_0,\tau_0,\sigma_0)|(\overline{\mathsf{e},\tau,\sigma}^{1\ldots n}),\mathsf{H},\Pi \rightsquigarrow (\mathsf{e}'_0,\tau_0,\sigma'_0)|(\overline{\mathsf{e},\tau,\sigma}^{1\ldots n}),\mathsf{H}',\Pi' \wedge \mathsf{WF}(\mathsf{H},\Pi,(\overline{\mathsf{e},\tau,\sigma}^{0\ldots n}))$$

$$\Longrightarrow$$

1. $(\mathsf{H}',\Pi',\sigma'_i,\tau_i \models \mathsf{P_i})_{i\in\{1\ldots n\}}$
2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}',\Pi',\sigma'_0,\tau_0 \models \mathsf{P}'_0\ ]$
3. $\mathsf{WF}(\mathsf{H}',\Pi',(\mathsf{e}'_0,\tau_0,\sigma'_0)|(\overline{\mathsf{e},\tau,\sigma}^{1\ldots n})\ )$

4. CONCLUSIONS AND FUTURE WORK

A similar verification protocol to that of Chalice with has been proposed for separation logic as the underlying logic of assertions in [3] and has been proven sound in [4]. This system also supports re-entrant locks. It is developed on a partial heap semantics, and permissions are implicit.

Chalice has been developed with implicit dynamic frames as its underlying logic of assertions. Our formalization supports a whole heap semantics and makes permission passing explicit, and therefore in our opinion reflects more directly the idea behind the verification protocol. The treatment of the assertion language as, to some extent, external to our system will allow us to distill what is absolutely essential for the verification protocol. For instance, we realized that all assertions treated at the level of the Hoare logic rules had to be self-framing, even though they may internally consist of smaller, non-self-framing assertions.

We hope that our treatment of the assertion language as external will allow the model to be applicable to a wider range of assertion languages.

We believe that the separation of execution into the "real" operations, and "ghost" operations working on the permissions mask gives a natural account of the ideas behind the Chalice verification protocol. Note however, that we have been unable to represent the state of the tokens as ghost operqtions, because this would have made it impossible to combine with out other goal, which was to be agnostic as to the meaning of logical conectives,

As further work, we would like to consider in how far existing assertion languages satisfy the properties distill in section 3.3. We will consider the application of these ideas to other concurrency concepts, such as chords [1].

Acknowledgements

We are grateful to Alexander J. Summers for many fruitful conversations. We thank the anonymous reviewers for lots of interesting and constructive feedback.

5. REFERENCES

[1] Nick Benton, Luca Cardelli, and Cedric Fournet. Modern concurrency abstractions for c#. *TOPLAS*, 2004.

[2] John Boyland. Checking interference with fractional permissions. In *Static Analysis: 10th International Symposium*, pages 55–72. Springer, 2003.

[3] Christian Haack and Clément Hurlin. Separation logic contracts for a Java-like language with fork/join. In *International Conference on Algebraic Methodology and Software Technology (AMAST'08)*, July 2008.

[4] Clément Hurlin. *Specification and Verification of Multithreaded Object-Oriented Programs with Separation Logic*. PhD thesis, Université Nice - Sophia Antipolis, September 2009.

[5] K. Rustan Leino and Peter Müller. A basis for verifying multi-threaded programs. In *Proceedings of the 18th European Symposium on Programming Languages and Systems: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, ESOP '09, pages 378–393, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] K. Rustan Leino, Peter Müller, and Jan Smans. *Verification of Concurrent Programs with Chalice*, pages 195–222. Springer-Verlag, Berlin, Heidelberg, 2009.

[7] M. J. Parkinson and A. J. Summers. The relationship between separation logic and implicit dynamic frames. In Gilles Barthe, editor, *European Symposium on Programming (ESOP)*, volume 6602 of *Lecture Notes in Computer Science*. Springer-Verlag, 2011.

[8] Jan Smans, Bart Jacobs, and Frank Piessens. Implicit dynamic frames: Combining dynamic frames and separation logic. In *Proceedings of the 23rd European Conference on ECOOP 2009 — Object-Oriented Programming*, Genoa, pages 148–172, Berlin, Heidelberg, 2009. Springer-Verlag.

APPENDIX

A. ADDITIONAL DEFINITIONS

A.1 Definition ($\pi_1 \uplus \pi_2$)

The union of two permission masks is defined *if and only if*
$\forall(\iota.\mathsf{f}) \in \mathsf{Dom}(\pi_1) \cap \mathsf{Dom}(\pi_2).[\pi(\iota.\mathsf{f}) + \pi_2(\iota.\mathsf{f}) \leq 1]$
and is described as follows.

$$\pi_1 \uplus \pi_2(\iota.f) = \begin{cases} m & \text{if } \pi_1(\iota.f) = m \wedge \pi_2(\iota.f) = Udf \\ n & \text{if } \pi_2(\iota.f) = n \wedge \pi_1(\iota.f) = Udf \\ m+n & \text{if } \pi_1(\iota.f) = m \wedge \pi_2(\iota.f) = n \\ & \quad \wedge\ m+n \leq 1 \end{cases}$$

A.2 (H $\stackrel{\pi}{\equiv}$ H')

$$\mathsf{H} \stackrel{\pi}{\equiv} \mathsf{H}' \iff \forall(\iota.\mathsf{f})[\pi(\iota.\mathsf{f}) > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})] \wedge \forall(\kappa.\mathsf{g})[\pi(\kappa.\mathsf{g}) > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$$

where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

A.3 Definition ($\sigma_1 \uplus \sigma_2$)

The union of two stack frames σ_1 and σ_2 is defined *if and only if* $\mathsf{Dom}(\sigma_1) \cap \mathsf{Dom}(\sigma_2) = \emptyset$ and is described as follows.

$$\sigma_1 \uplus \sigma_2(x) = \begin{cases} \sigma_1(x) & \text{if } x \in Dom(\sigma_1) \wedge x \notin Dom(\sigma_2) \\ \sigma_2(x) & \text{if } x \in Dom(\sigma_2) \wedge x \notin Dom(\sigma_1) \end{cases}$$

A.4 Definition ($\pi_1 \subseteq \pi_2$)

Permission mask inclusion is defined as follows.

$$\pi_1 \subseteq \pi_2 \iff \forall(\iota.\mathsf{f}) \in \mathsf{Dom}(\pi_1)[\pi_1(\iota.\mathsf{f}) \leq \pi_2(\iota.\mathsf{f})]$$

B. COMPLEMENTARY SOUNDNESS THEOREMS FOR Chalice^f

As we pointed out earlier, the execution of an expression e_0 can result in three different situations:

- e_0 reduces to another expression e'_0 and all other threads in the system remain unchanged.
- e_0 reduces to another expression e'_0 and in doing so, a new thread is created. Meanwhile, all the other threads initially present in the system remain unchanged. This case corresponds to the execution of a `fork` statement.
- e_0 reduces to another expression e'_0 and results in the termination of another thread. All other threads remain unchanged in the system. This case corresponds to the execution of a `join` statement.

We have formalise the first scenario in the main part of the paper. The latter two scenarios are formalised here. The proofs of these theorems is provided in Appendix D.

Soundness Theorem 2
$\mathsf{Ver}(\mathsf{Prog})$
$\wedge(\{\mathsf{P}_i\}\ \mathsf{e}_i\ \{\mathsf{Q}_i\})_{i\in\{0\ldots n-1\}}$
$\wedge(\mathsf{H}, \Pi, \sigma_i, \tau_i \models \mathsf{P}_i)_{i\in\{0\ldots n-1\}}$
$\wedge\ (\mathsf{e}_0, \tau_0, \sigma_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n-1}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n-1})|(\mathsf{e}_n, \tau_n, \sigma_n), \mathsf{H}', \Pi'$
$\wedge\ \mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0\ldots n-1}))$
$\implies$
1. $(\mathsf{H}', \Pi', \sigma_i, \tau_i \models \mathsf{P}_i)_{i\in\{1\ldots n-1\}}$
2. $\exists \mathsf{P}_n, \mathsf{Q}_n.[\{\mathsf{P}_n\}\ \mathsf{e}_n\ \{\mathsf{Q}_n\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_n, \tau_n \models \mathsf{P}_n]$
3. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi' \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
4. $\mathsf{WF}(\mathsf{H}', \Pi', (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n}))$

Soundness Theorem 3
$\mathsf{Ver}(\mathsf{Prog})$
$\wedge\ (\{\mathsf{P}_i\}\ \mathsf{e}_i\ \{\mathsf{Q}_i\})_{i\in\{0\ldots n\}}$
$\wedge\ (\mathsf{H}, \Pi, \sigma_i, \tau_i \models \mathsf{P}_i)_{i\in\{0\ldots n\}}$
$(\mathsf{e}_0, \tau_0, \sigma_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n-1}), \mathsf{H}', \Pi'$
$\mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0\ldots n}))$
$\implies$
1. $(\mathsf{H}', \Pi', \sigma_i, \tau_i \models \mathsf{P}_i)_{i\in\{1\ldots n-1\}}$
2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi' \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
3. $\mathsf{WF}(\mathsf{H}', \Pi', (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\ldots n-1}))$

C. PROOF OF SUPPORTING LEMMAS

In our proofs we have used different names for the properties of asserions defined in section **??**, as follows

P1a. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \ \wedge\ \overline{\mathsf{y}} = \mathsf{fv}(\mathsf{A}) \ \wedge\ \sigma'(\overline{\mathsf{x}}) = \sigma(\overline{\mathsf{y}}) \implies \mathsf{H}, \pi, \sigma' \models \mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{y}}]$

P1b. $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}) = \mathcal{P}(\mathsf{H}, \sigma[\overline{\mathsf{y}} \mapsto \overline{\sigma(\mathsf{x})}], \mathsf{A}[\overline{\mathsf{y}}/\overline{\mathsf{x}}])$

P2. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \implies \mathsf{H}, \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}), \sigma \models \mathsf{A} \wedge \forall(\iota.\mathsf{f})[\pi(\iota.\mathsf{f}) \geq \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f})]$

P3. $\mathsf{A} \rightarrow_{\mathsf{a}} \mathsf{A}' \quad \wedge \quad \mathsf{H}, \pi, \sigma \models \mathsf{A} \implies \mathsf{H}, \pi, \sigma \models \mathsf{A}'$

P4a. $\mathsf{H}, \pi, \sigma \models \mathsf{x.f} = \mathsf{y} \iff \mathsf{H}(\sigma(\mathsf{x}), \mathsf{f}) = \sigma(\mathsf{y})$

P4b. $\mathsf{H}, \pi, \sigma \models \mathsf{acc}(\mathsf{x.f}, \mathsf{n}) \iff \pi(\sigma(\mathsf{x}), \mathsf{f}) \geq \mathsf{n}$

P5. If A_1 and A_2 are self-framing, then
$\mathsf{H}, \pi, \sigma \models \mathsf{A}_1 * \mathsf{A}_2 \iff$
$\exists \pi_1, \pi_2.[\mathsf{H}, \pi_1, \sigma \models \mathsf{A}_1 \ \wedge\ \mathsf{H}, \pi_2, \sigma \models \mathsf{A}_2, \ \wedge\ \pi = \pi_1 \uplus \pi_2]$
where $\pi_1 \uplus \pi_2$ is as defined in Appendix A.

P6. We call an assertion B a *case-split assertion*, if
$\mathsf{SF}(\mathsf{B} \wedge \mathsf{A}) \implies \mathsf{SF}(\mathsf{A}) \wedge [\mathsf{H}, \pi, \sigma \models \mathsf{A} \implies \mathsf{H}, \pi, \sigma \models \mathsf{B} \ \vee\ [\mathsf{H}, \pi, \sigma \models \neg \mathsf{B}].$

P7. $\mathsf{H}, \pi, \sigma \models \mathsf{A} \wedge \mathsf{A}' \iff \mathsf{H}, \pi, \sigma \models \mathsf{A}$ and $\mathsf{H}, \pi, \sigma \models \mathsf{A}'$

P8. A, A' is self framing, then $\mathsf{A} * \mathsf{A}'$ is also self-framing.

P9. $\mathsf{H}, \pi, \sigma \models \mathsf{A} * \mathsf{A}' \implies$
$\forall(\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f}) + \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}')(\iota.\mathsf{f}) \leq 1]$
$\wedge\ \forall(\kappa)[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\kappa.\mathsf{g}) + \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}')(\kappa.\mathsf{g}) \leq 1]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

P10. $\mathsf{H}, \pi, \sigma \models \mathsf{true}$

P11. $\forall \mathsf{H}, \sigma, \mathsf{A}, (\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota.\mathsf{f}) \neq \mathsf{Udf} \implies \iota.\mathsf{f} \in \mathsf{Dom}(\mathsf{H})]$
$\wedge\ \forall \mathsf{H}, \sigma, \mathsf{A}, (\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\kappa.\mathsf{g}) \neq \mathsf{Udf} \implies \kappa.\mathsf{g} \in \mathsf{Dom}(\mathsf{H})]$

P12. We use the shorthand
$\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}) \equiv \mathsf{acc}(\mathsf{tk.c}, 1) * \mathsf{tk.c} = \mathsf{C} * \mathsf{acc}(\mathsf{tk.m}, 1) * \mathsf{tk.m} = \mathsf{m} * \mathsf{acc}(\mathsf{tk.args}, 1) * \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$

C.1 Lemma 1

$\mathsf{H}, \Pi, (\mathsf{e}, \sigma, \tau) \rightsquigarrow (\mathsf{e}', \sigma', \tau), \mathsf{H}', \Pi' \ \wedge\ \Pi = \Pi_1 \uplus \Pi_2$
$\wedge\ \mathsf{H}, \Pi_1, (\mathsf{e}, \sigma, \tau) \rightsquigarrow (\mathsf{e}', \sigma', \tau), \mathsf{H}', \Pi'_1$
$\implies \Pi' = \Pi'_1 \uplus \Pi_2$

where $\uplus$ over Π has the obvious meaning.

Proof. By induction over the $\rightsquigarrow$ derivations.

C.2 Lemma 2

$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{P} * \mathsf{R} \ \wedge\ \{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{Q}\}$
$\wedge\ \mathsf{H}, \Pi, (\mathsf{e}, \sigma, \tau) \rightsquigarrow (\mathsf{e}', \sigma', \tau), \mathsf{H}', \Pi' \implies$
$\exists \Pi_1, \Pi_2.[\ \mathsf{H}, \Pi_1, \sigma, \tau \models \mathsf{P} \ \wedge\ \mathsf{H}, \Pi_2, \sigma, \tau \models \mathsf{R}$
$\wedge\ \Pi = \Pi_1 \uplus \Pi_2 \ \wedge\ \mathsf{H}, \Pi_1, (\mathsf{e}, \sigma, \tau) \rightsquigarrow (\mathsf{e}', \sigma', \tau), \mathsf{H}', \Pi'_1]$

where $\uplus$ over Π has the obvious meaning.

FAss $\dfrac{\sigma(\mathsf{x}) = \iota \quad \mathsf{H}' = \mathsf{H}[\iota \mapsto \mathsf{H}(\iota)[\mathsf{f} \mapsto \sigma(\mathsf{y})]]}{(\mathsf{x.f} := \mathsf{y}\ ,\ \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\sigma(\mathsf{y}), \tau, \sigma), \mathsf{H}', \Pi}$

VAss $\dfrac{\sigma(\mathsf{x}) = \iota \quad \mathsf{H}(\iota) \downarrow_2 (\mathsf{f}) = \mathsf{v} \quad \sigma' = \sigma[\mathsf{y} \mapsto \mathsf{v}]}{(\mathsf{y} := \mathsf{x.f},\ \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{v}, \tau, \sigma'), \mathsf{H}, \Pi}$

Val. $\dfrac{}{(\mathsf{v}; \mathsf{e}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}, \tau, \sigma), \mathsf{H}, \Pi}$

Seq. $\dfrac{(\mathsf{e}_1, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}_1', \tau, \sigma'), \mathsf{H}', \Pi'}{(\mathsf{e}_1; \mathsf{e}_2, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}_1'; \mathsf{e}_2, \tau, \sigma'), \mathsf{H}', \Pi'}$

Acq. $\dfrac{\begin{array}{l}\sigma(\mathsf{x}) = \iota \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{H}(\iota) \downarrow_3 = \tau_\mathsf{g} \\ \mathsf{Invariant}(\mathsf{C}) = \mathsf{A} \quad \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}[\mathsf{x}/\mathsf{this}]) = \mathsf{ps} \\ \Pi' = \Pi[\tau + = \mathsf{ps},\ \ \tau_\mathsf{g} - = \mathsf{ps}] \\ \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau)]\end{array}}{(\mathsf{acquire\ x}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}', \Pi'}$

Rel. $\dfrac{\begin{array}{l}\sigma(\mathsf{x}) = \iota \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{H}(\iota) \downarrow_3 = \tau \\ \mathsf{Invariant}(\mathsf{C}) = \mathsf{A} \quad \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}[\mathsf{x}/\mathsf{this}]) = \mathsf{ps} \\ \Pi' = \Pi[\tau - = \mathsf{ps},\ \ \tau_\mathsf{g} + = \mathsf{ps}] \\ \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau_\mathsf{g})]\end{array}}{(\mathsf{release\ x}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}', \Pi'}$

Thrd $\dfrac{\overline{\mathsf{P}}, \mathsf{H}, \Pi \rightsquigarrow \overline{\mathsf{P}'}, \mathsf{H}', \Pi'}{\overline{\mathsf{P}_1} \mid \overline{\mathsf{P}} \mid \overline{\mathsf{P}_2}, \mathsf{H}, \Pi \rightsquigarrow \overline{\mathsf{P}_1} \mid \overline{\mathsf{P}'} \mid \overline{\mathsf{P}_2}, \mathsf{H}', \Pi'}$

IfT $\dfrac{}{(\mathsf{if(true)then\{e2\}else\{e3\}}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e2}, \tau, \sigma), \mathsf{H}, \Pi}$

IfF $\dfrac{}{(\mathsf{if(false)then\{e2\}else\{e3\}}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e3}, \tau, \sigma), \mathsf{H}, \Pi}$

Meth $\dfrac{\mathsf{tk} \notin \sigma}{(\mathsf{x.m}(\overline{\mathsf{y}}), \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow ((\mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}); \mathsf{join\ tk}), \tau, \sigma), \mathsf{H}, \Pi}$

NewC $\dfrac{\begin{array}{l}\mathsf{FS}(\mathsf{C}) = \{\mathsf{t}_1\ \mathsf{f}_1 ..., \mathsf{t}_\mathsf{r}\ \mathsf{f}_\mathsf{r}\} \\ \iota \notin \mathsf{H} \quad \Pi' = \Pi[(\tau)(\iota, \mathsf{f}_\mathsf{i}) \mapsto 1]_{\mathsf{i} \in 1...\mathsf{r}} \\ \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{C}, \{\mathsf{f}_1 : \mathsf{null}, ..., \mathsf{f}_\mathsf{r} : \mathsf{null}\},\ \tau)]\end{array}}{(\mathsf{new\ C}\ , \tau, \sigma), \mathsf{H}, \Pi_0 \rightsquigarrow (\iota, \tau, \sigma), \mathsf{H}', \Pi'}$

Fork $\dfrac{\begin{array}{l}\sigma(\mathsf{x}) = \iota \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{pre}(\mathsf{C}, \mathsf{m}) = \mathsf{A} \\ \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A}[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]) = \mathsf{ps} \\ \tau' \notin \Pi \quad \Pi'' = \Pi[\tau' \mapsto \mathsf{ps},\ \ \tau - = \mathsf{ps}] \\ \Pi' = \Pi''[(\tau)(\kappa.\mathsf{g}) \mapsto 1] \text{ where } \mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\} \\ \kappa \notin \mathsf{H} \quad \sigma' = \sigma[\mathsf{tk} \mapsto \kappa] \\ \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\sigma(\mathsf{y})}\}, \tau')] \\ \mathsf{mBody}(\mathsf{C}, \mathsf{m}) = \mathsf{e}(\overline{\mathsf{u}}) \quad \sigma'' = [\mathsf{this} \mapsto \iota, \overline{\mathsf{u}} \mapsto \overline{\sigma(\mathsf{y})}]\end{array}}{\begin{array}{l}(\mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}), \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow \\ \qquad ((\mathsf{null}, \tau, \sigma')|(\mathsf{e}, \tau', \sigma'')), \mathsf{H}', \Pi'\end{array}}$

Join $\dfrac{\begin{array}{l}\sigma(\mathsf{tk}) = \kappa \quad \mathsf{H}(\kappa) \downarrow_3 = \tau' \quad \mathsf{H}(\kappa.\mathsf{args}) = \iota.\bar{\iota} \\ \mathsf{H}(\kappa.\mathsf{c}) = \mathsf{C} \quad \mathsf{H}(\kappa.\mathsf{m}) = \mathsf{m} \quad \mathsf{Post}(\mathsf{C}, \mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}}) \\ \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota, \overline{\mathsf{u}} \mapsto \bar{\iota}], \mathsf{A}(\overline{\mathsf{u}})) = \mathsf{ps} \\ \Pi' = \Pi[\tau + = \mathsf{ps},\ \ \tau' - = \mathsf{ps}] \\ \mathsf{H}' = \mathsf{H}[\kappa \mapsto \epsilon]\end{array}}{(\mathsf{v}, \tau', \sigma)|(\mathsf{join\ tk}, \tau, \sigma), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{null}, \tau, \sigma), \mathsf{H}', \Pi'}$

Figure 8: Combined Operational and permission passing semantics of Chalice^f

Proof. By induction over the Hoare Logic triplets.

C.3 Lemma 3

$\mathsf{H}, \Pi, (\mathsf{e}, \tau, \sigma) \rightsquigarrow (\mathsf{e}', \tau, \sigma'), \mathsf{H}', \Pi' \ \wedge \ \tau' \neq \tau \ \wedge \ \tau' \neq \tau_{\mathsf{g}}$
$\implies \quad \Pi'(\tau')(\iota.\mathsf{f}) = \Pi(\tau')(\iota.\mathsf{f})$

Proof. By straightforward induction over the $\rightsquigarrow$ derivations.

C.4 Lemma 4

$\{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{Q}\}$ (G1)
$\wedge\ \mathsf{H}, \Pi, \sigma, \tau \models \mathsf{P}$ (G2)
$\wedge\ \tau \neq \tau'$ (G3)
$\wedge\ \tau \neq \tau_{\mathsf{g}}$ (G4)
$\wedge\ [\ \mathsf{H}, \Pi, (\mathsf{e}'\tau', \sigma') \rightsquigarrow (\mathsf{e}'', \tau', \sigma''), \mathsf{H}', \Pi'$ (G5a)
$\quad \vee\ \{\mathsf{H}, \Pi, (\mathsf{e}', \tau', \sigma') \rightsquigarrow ((\mathsf{e}'', \tau', \sigma'')|(\mathsf{e}''', \tau'', \sigma_0)), \mathsf{H}', \Pi' \ \wedge \ \tau'' \neq \tau\}$ (G5b)
$\quad \vee\ \{\mathsf{H}, \Pi, ((\mathsf{v}, \tau'', \sigma_0)|(\mathsf{e}', \tau', \sigma')) \rightsquigarrow (\mathsf{e}'', \tau', \sigma''), \mathsf{H}', \Pi'$
$\qquad \wedge\ \tau'' \neq \tau\}$ (G5c) $]$
$\wedge\ \{\mathsf{P}'\}\ \mathsf{e}'\ \{\mathsf{Q}'\}$ (G6)
$\wedge\ \mathsf{H}, \Pi, \sigma', \tau' \models \mathsf{P}'$ (G7)
$\wedge\ \sum_{\tau_i \in \mathsf{DOM}(\Pi)} \Pi(\tau_i)(\iota.\mathsf{f}) \leq 1$ (G8)
$\quad \implies \quad \mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$

Proof. By induction over the Hoare Logic triplets ($\{\mathsf{P}'\}\ \mathsf{e}'\ \{\mathsf{Q}'\}$). Since we have $\{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{Q}\}$, in all the following cases from Lemma 7 we can deduce:
$\mathsf{SF}(\mathsf{P})$ (G9)

Case FldAss $\mathsf{P}' \equiv \mathsf{acc}(\mathsf{x.f}, 1)$ (I)
$\mathsf{e}' \equiv \mathsf{x.f} := \mathsf{y} \quad \Pi' = \Pi \quad \mathsf{H}' = \mathsf{H}[\sigma'(\mathsf{x})(\mathsf{f}) \mapsto \sigma'(\mathsf{y})]$

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$
In other words:
$\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall(\kappa.\mathsf{g}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

RTS. $\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι, f) such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ (II)
There are now two cases:
Case 1. $\iota.\mathsf{f} \neq \sigma'(\mathsf{x}).\mathsf{f}$
RTS. $\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$
Since H' is exactly the same as H except for the value of $\sigma'(\mathsf{x}).\mathsf{f}$, we can deduce that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (III)

Case 2.
$\iota.\mathsf{f} = \sigma'(\mathsf{x}).\mathsf{f}$ (IV)
RTS. $\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ From (G7) and (I) we know:
$\mathsf{H}, \Pi, \sigma', \tau' \models \mathsf{acc}(\mathsf{x.f}, 1)$ (V)
From the definition of $\models$ we know:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc}(\mathsf{x.f}, 1)$ (VI)
From (VI) and property P4b we have:
$\Pi(\tau')(\sigma'(\mathsf{x}).\mathsf{f}) \geq 1$ from (IV) that is:
$\Pi(\tau')(\iota.\mathsf{f}) \geq 1$ (VII)
On the other hand, From G2 we have:
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (VIII)
By definition, we know permission masks are a set of mappings from locations and field identifiers to (positive) numbers between 0 and 1 inclusively. That is, we know:
$\Pi(\tau)(\iota.\mathsf{f}) \geq 0$ (IX)
From (VII) and (IX) we have:
$\Pi(\tau)(\iota.\mathsf{f}) + \Pi(\tau')(\iota.\mathsf{f}) \geq 1$ (X)
On the other hand from (G8) we know:
$\Pi(\tau)(\iota.\mathsf{f}) + \Pi(\tau')(\iota.\mathsf{f}) \leq 1$ (XI)
From (X) and (XI) we know:
$\Pi(\tau)(\iota.\mathsf{f}) + \Pi(\tau')(\iota.\mathsf{f}) = 1$ (XII)
From (VII) and (XII) we can deduce:
$\Pi(\tau)(\iota.\mathsf{f}) = 0$ (XIII)
From (VIII), (XIII) and property P2 we can deduce:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota.\mathsf{f}) \leq 0$
However, from (II) we have:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota.\mathsf{f}) > 0$
Contradiction!!
Hence we can deduce:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (XIV)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$.
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of $\iota.f$, we can deuce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (XV)

From (III), (XIV) and (XV) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (XVI)
From (G9), (VIII), (XVI) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (XVII)
From (XVII) and the definition of $\models$ we know:
$\mathsf{H}', \Pi, \sigma, \tau \models \mathsf{P}$ (XVIII)
Finally, since $\Pi' = \Pi$, from (XVIII) we can deduce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Cases VarAss, IfT, IfF, Val, Meth
Since $\mathsf{H}' = \mathsf{H}$ and $\Pi' = \Pi$, from G2 we can deduce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Case NewC
$\mathsf{H}' = \mathsf{H}[\iota' \mapsto (\mathsf{C}, \{\mathsf{f}_1 : \mathsf{null} ... \mathsf{f}_n : \mathsf{null}\}, \tau')] \ \iota' \notin \mathsf{H} \ \mathsf{f}_1 ... \mathsf{f}_n \in \mathsf{FS}(\mathsf{C})$
$\Pi' = \Pi[(\tau')(\iota', \mathsf{f}_i) \mapsto 1] \ \mathsf{f}_i \in \mathsf{FS}(\mathsf{C})$

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$
In other words,
$\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall(\kappa.\mathsf{g}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

RTS. $\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι, f) such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ (I)
We know from P11 that: $\iota \in \mathsf{Dom}(\mathsf{H})$, that is $\iota \neq \iota'$.
On the other hand, since H' is exactly the same as H except for the value of ι', we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$. (II)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$.
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of ι', we can deuce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (III)

From (II) and (III) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (IV)
From (G2) and the definition of $\models$ we can deduce:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (V)
From (G9), (IV), (V) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (VI)
Since $\Pi'(\tau) = \Pi(\tau)$, we can deduce:
$\mathsf{H}', \Pi'(\tau), \sigma, \models \mathsf{P}$ (VII)
Finally, from (VII) and the definition of the $\models$ judgement, we can deuce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Case Acq. $\mathsf{e} \equiv \mathsf{acquire}\ \mathsf{x} \quad \sigma'(\mathsf{x}) = \iota' \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{Inv}(\mathsf{C}) = \mathsf{A}$
$\mathsf{H}' = \mathsf{H}[\iota' \mapsto (\mathsf{H}(\iota') \downarrow_1, \mathsf{H}(\iota') \downarrow_2, \tau')]$
$\Pi' = \Pi[\tau' + = \mathcal{P}(\mathsf{H}, \sigma', \mathsf{A}[\mathsf{x}/\mathsf{this}]), \ \tau_{\mathsf{g}} - = \mathcal{P}(\mathsf{H}, \sigma', \mathsf{A}[\mathsf{x}/\mathsf{this}])]$

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$
In other words,
$\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall(\kappa.\mathsf{g}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

RTS. $\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι, f) such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ (I)
There are now two cases:
Case 1. $\iota.\mathsf{f} \neq \iota'.\mathsf{f}$
RTS. $\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$

Since H' is exactly the same as H except for the value of ι', we can deduce that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (II)

Case 2. $\iota.\mathsf{f} = \iota'.\mathsf{f}$
RTS. $\mathsf{H}'(\iota'.\mathsf{f}) = \mathsf{H}(\iota'.\mathsf{f})$
From the definition of H', we know that $\mathsf{H}'(\iota') \downarrow_2 = \mathsf{H}(\iota') \downarrow_2$ and hence:
$\mathsf{H}'(\iota'.\mathsf{f}) = \mathsf{H}(\iota'.\mathsf{f})$ as required. (III)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$.
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of ι', we can deuce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (IV)

From (II), (III) and (IV) we have:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (V)
From (G2) and the definition of $\models$ we can deduce:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (VI)
From (G9), (V), (VI) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (VII)
Since $\Pi'(\tau) = \Pi(\tau)$, we can deduce:
$\mathsf{H}', \Pi'(\tau), \sigma, \models \mathsf{P}$ (VIII)
Finally, from (VIII) and the definition of the $\models$ judgement, we can deuce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Case Rel. $\mathsf{e} \equiv \mathsf{release\ x} \quad \sigma'(\mathsf{x}) = \iota' \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{Inv(C)} = \mathsf{A}$
$\mathsf{H}' = \mathsf{H}[\iota' \mapsto (\mathsf{H}(\iota') \downarrow_1, \mathsf{H}(\iota') \downarrow_2, \tau_\mathsf{g})]$
$\Pi' = \Pi[\tau' -= \mathcal{P}(\mathsf{H},\sigma',\mathsf{A}[\mathsf{x}/\mathsf{this}])], \ \tau_\mathsf{g} += \mathcal{P}(\mathsf{H},\sigma',\mathsf{A}[\mathsf{x}/\mathsf{this}])])]$
The proof of this case is similar to the previous case and is left out.

Case Fork
$\mathsf{e} \equiv \mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}) \quad \sigma'(\mathsf{x}) = \iota' \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{Pre(C,m)} = \mathsf{A}(\overline{\mathsf{u}})$
$\tau'' \notin \Pi \quad \Pi'' = \Pi[\tau'' \mapsto \mathsf{ps}, \ \tau' -= \mathsf{ps}] \quad \mathsf{ps} = \mathcal{P}(\mathsf{H},\sigma',\mathsf{A}[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])$
$\kappa \notin \mathsf{H} \quad \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\sigma'(\overline{\mathsf{y}})\}, \tau'')]$
$\Pi' = \Pi''[(\tau')(\kappa.\mathsf{g}) \mapsto 1] \quad$ for $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$
In other words:
$\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\wedge \ \forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

RTS. $\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι,f) such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
Since H' is exactly the same as H except for the value of κ, we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (I)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $(\kappa'.\mathsf{g})$ s.t. $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
RTS. $\mathsf{H}(\kappa'.\mathsf{g}) = \mathsf{H}'(\kappa'.\mathsf{g})$
Since we have: $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n}$, from property P11 we know that:
$\kappa' \in \mathsf{Dom(H)}$ (II)
Since we have $\kappa \notin \mathsf{Dom(H)}$, from (II) we can deduce:
$\kappa' \neq \kappa$ (III)
Since H' is exactly the same as H except for the value of κ, from (III) we deduce:
$\mathsf{H}'(\kappa') = \mathsf{H}(\kappa)$ as required. (IV)
From (I) and (IV) we can can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (V)
From (G2) and the definition of $\models$ we can deduce:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (VI)
From (V), (VI), (G9) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (VII)
As $\tau'' \notin \Pi$, we know $\tau \neq \tau''$ (VIII)
Since Π' is exactly the same as Π except for the values of τ'' and τ', from (VIII) we know:
$\Pi'(\tau) = \Pi(\tau)$. (IX)
From (VII) and (IX) we can deduce:
$\mathsf{H}', \Pi'(\tau), \sigma, \models \mathsf{P}$ (X)
Finally, from (X) and the definition of the $\models$ judgement, we can deuce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Case Join
$\mathsf{e} \equiv \mathsf{join\ tk} \quad \sigma(\mathsf{tk}) = \kappa \quad \mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau'') \quad \mathsf{Post(C,m)} = \mathsf{A}(\overline{\mathsf{u}})$
$\Pi' = \Pi[\tau'' -= \mathsf{ps}, \ \tau' += \mathsf{ps}] \quad \mathsf{ps} = \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota', \overline{\mathsf{u}} \mapsto \overline{\iota}], \mathsf{A}(\overline{\mathsf{u}}))$
$\mathsf{H}' = \mathsf{H}[\kappa \mapsto \mathsf{null}]$
$\mathsf{P}' \equiv \mathsf{Thread(tk, C, m, x.}\overline{\mathsf{y}})$ (I)

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$, that is:
$\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\wedge \ \forall(\kappa,\mathsf{g}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$

RTS. $\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι,f) such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
Since H' is exactly the same as H except for the value of κ, we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (II)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $(\kappa'.\mathsf{g})$ such that $\mathcal{P}(\mathsf{H},\sigma',\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$. (III)
There are now two cases:

Case 1.
$\kappa' \neq \sigma'(\mathsf{tk})$
RTS. $\mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of $\sigma'(\mathsf{tk})$, we can deduce that:
$\mathsf{H}'(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$ (IV)

Case 2. $\kappa' = \sigma'(\mathsf{tk}) = \kappa$
RTS. $\mathsf{H}'(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$
From (G7) and (I) we know:
$\mathsf{H}, \Pi, \sigma', \tau' \models \mathsf{Thread(tk, C, m, args)}$
From the definition of $\models$ we know:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{Thread(tk, C, m, args)}$ (V)
From P12 and (V) we derive:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc(tk.c, 1)} * \mathsf{acc(tk.m, 1)} * \mathsf{acc(tk.args, 1)}$
$* \ \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x.}\overline{\mathsf{y}}$ (VI)
From (VI) and applying property P5 three times we have:
$\mathsf{H}, \pi_1, \sigma' \models \mathsf{acc(tk.c, 1)}$ (VII)
$\mathsf{H}, \pi_2, \sigma' \models \mathsf{acc(tk.c, 1)}$ (VIII)
$\mathsf{H}, \pi_3, \sigma' \models \mathsf{acc(tk.c, 1)}$ (IX)
$\mathsf{H}, \pi_4, \sigma' \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x.}\overline{\mathsf{y}}$
where $\Pi(\tau') = \pi_1 \uplus \pi_2 \uplus \pi_3 \uplus \pi_4$ (X)
From (VII)-(X) and lemma 13 we have:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc(tk.c, 1)}$ (XI)
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc(tk.c, 1)}$ (XII)
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc(tk.c, 1)}$ (XIII)
From (X)-(XIII) and property P4b we have:
$\Pi(\tau')(\sigma'(\mathsf{tk}).\mathsf{g}) \geq 1 \quad$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (XIV)
By definition, we know permission masks are a set of mappings from locations and field identifiers to (positive) numbers between 0 and 1 inclusively. That is, we know:
$\Pi(\tau)(\kappa.\mathsf{g}) \geq 0$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (XV)
From (XIV) and (XV) we have:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) \geq 1$ (XVI)
On the other hand from (G8) we know:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) \leq 1$ (XVII)
From (XVI) and (XVII) we know:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) = 1$ (XVIII)
From (XIV) and (XVIII) we can deduce:
$\Pi(\tau)(\kappa.\mathsf{g}) = 0$ (XIX)
On the other hand, From G2 we have:
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (XX)
From (XX), (XIX) and property P2 we can deduce:
$\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa.\mathsf{g}) \leq 0$
However, from (III) we have:
$\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa.\mathsf{g}) > 0$
Contradiction!!
Hence we can deduce:
$\mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$ (XXI)
From (II), (IV) and (XXI) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (XXII)
From (G9), (XX), (XXII) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (XXIII)
From (XXIII) and the definition of $\models$ we know:
$\mathsf{H}', \Pi, \sigma, \tau \models \mathsf{P}$ (XXIV)
Finally, since $\Pi' = \Pi$, from (XXIV) we can deduce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$ as required.

Case Seq.
$\mathsf{e}' \equiv \mathsf{e}_1; \mathsf{e}_2$
$\{\mathsf{P}'\} \ \mathsf{e}_1 \ \{\mathsf{R}\}$ (I)

$\{R\}\ e_2\ \{Q'\}$

RTS. $H', \Pi', \sigma, \tau \models P$
Given the definition of e', from the operational semantics of the (Seq.) rule we know:
$H, \Pi, (e_1, \sigma', \tau') \rightsquigarrow (e_1', \sigma'', \tau'), H', \Pi'$ (II)
From (G1), (G2), (G3), (G4), (II), (I), (G7), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$ as required.

Case Frame
$P' \equiv A * C$ (I)
$\{A\}\ e'\ \{B\}$ (II)

RTS. $H', \Pi', \sigma, \tau \models P$
From (G7) and (I) we have:
$H, \Pi, \sigma', \tau' \models A * C$ (III)
From (III) and the definition of $\models$ we have:
$H, \Pi(\tau'), \sigma' \models A * C$ (IV)
From (IV) and property P5 we have:
$H, \pi_1, \sigma' \models A$ (V)
$H, \pi_2, \sigma' \models C$
$\Pi(\tau') = \pi_1 \uplus \pi_2$ (VI)
From (V), (VI) and lemma 13 we have:
$H, \Pi(\tau'), \sigma' \models A$ (VII)
From (VII) and the definition of $\models$ we have:
$H, \Pi, \sigma', \tau' \models A$ (VIII)
From (G1), (G2), (G3), (G4), (G5), (II), (VIII), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$ as required.

Case Conseq.
$P' \rightarrow_a A$ (I) $\{A\}\ e'\ \{B\}$ (II)

RTS. $H', \Pi', \sigma, \tau \models P$
From (G7), (I) and property P3 we have:
$H, \Pi, \sigma', \tau' \models A$ (III)
From (G1), (G2), (G3), (G4), (G5), (II), (III), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$ as required.

C.5 Lemma 5

$H, \Pi, \sigma, \tau \models A \ \wedge\ \Pi' = \Pi[\tau- = \mathcal{P}(H, \sigma, A),\ \tau'+ = \mathcal{P}(H, \sigma, A)]$
$\implies$ $H, \Pi', \sigma, \tau' \models A$

Proof. Since we have $H, \Pi, \sigma, \tau \models A$, by definition of the $\models$ judgement we have:
$H, \Pi(\tau), \sigma \models A$ (I)
From (I) and property P2 of the $\models$ judgement we can deduce:
$H, \mathcal{P}(H, \sigma, A), \sigma \models A$ (II)
Now from the definition of Π' we have:
$\Pi'(\tau) = \Pi(\tau) \setminus \mathcal{P}(H, \sigma, A)$ (III)
$\Pi'(\tau') = \Pi(\tau') \uplus \mathcal{P}(H, \sigma, A)$ (IV)
$\forall \tau_0[\tau_0 \neq \tau \wedge \tau_0 \neq \tau' \implies \Pi'(\tau_0) = \Pi(\tau_0)]$ (V)
From (II), (IV) and Lemma 13a we have:
$H, \Pi'(\tau'), \sigma \models A$ (VI)
From (VI) and the definition of $\models$ we can deduce: $H, \Pi', \sigma, \tau' \models A$ as required.

C.6 Lemma 6

$SF(P)$ (G1)
$H, \Pi, \sigma, \tau \models P$ (G2)
$\tau \neq \tau'$ (G3)
$e' \not\equiv \text{acquire x} \ \wedge\ e' \not\equiv \text{release x}$ (G4)
$[H, \Pi, (e', \tau', \sigma') \rightsquigarrow (e'', \tau', \sigma''), H', \Pi'$ (G5a)
$\quad \vee\ \{H, \Pi, (e', \tau', \sigma') \rightsquigarrow ((e'', \tau', \sigma'') | (e''', \tau''', \sigma''')), H', \Pi'\}$ (G5b)
$\quad \vee\ \{H, \Pi, ((v, \tau'', \sigma'') | (e', \tau', \sigma')) \rightsquigarrow (e'', \tau', \sigma''), H', \Pi'$ (G5c)
$\{P'\}\ e'\ \{Q'\}$ (G6)
$H, \Pi, \sigma', \tau' \models P'$ (G7)
$\sum \tau_i \in DOM(\Pi)\Pi(\tau_i)(\iota.f) \leq 1$
(G8)
$\implies$ $H', \Pi', \sigma, \tau \models P$

Proof. There are two cases to consider:

Case 1. $\tau \neq \tau_g$
This case follows immediately from Lemma 4.

Case 2. $\tau = \tau_g$
Proof. By induction over the Hoare Logic triplets ($\{P'\}\ e'\ \{Q'\}$).

Case FldAss
$P' \equiv \text{acc(x.f, 1)}$ (I)
$e' \equiv \text{x.f := y} \quad \Pi' = \Pi \quad H' = H[\sigma'(x)(f) \mapsto \sigma'(y)]$

First, we show that $H \overset{\mathcal{P}(H,\sigma,P)}{\equiv} H'$
In other words:
$\forall(\iota, f).[\mathcal{P}(H, \sigma, P)(\iota, f) = n \ \wedge\ n > 0 \implies H(\iota.f) = H'(\iota.f)]$
$\forall(\kappa.g).[\mathcal{P}(H, \sigma, P)(\kappa, g) = n \ \wedge\ n > 0 \implies H(\kappa.g) = H'(\kappa.g)]$
where $g \in \{c, m, args\}$

RTS. $\forall(\iota, f).[\mathcal{P}(H, \sigma, P)(\iota, f) = n \ \wedge\ n > 0 \implies H(\iota.f) = H'(\iota.f)]$
Take an arbitrary (ι, f) such that $\mathcal{P}(H, \sigma, P)(\iota, f) = n \ \wedge\ n > 0$ (II)
There are now two cases:
Case 1. $\iota.f \neq \sigma'(x).f$
RTS. $H'(\iota.f) = H(\iota.f)$
Since H' is exactly the same as H except for the value of $\sigma'(x).f$, we can deduce that:
$H'(\iota.f) = H(\iota.f)$ (III)

Case 2.
$\iota.f = \sigma'(x).f$ (IV)
RTS. $H'(\iota.f) = H(\iota.f)$ From (G7) and (I) we know:
$H, \Pi, \sigma', \tau' \models \text{acc(x.f, 1)}$ (V)
From the definition of $\models$ we know:
$H, \Pi(\tau'), \sigma' \models \text{acc(x.f, 1)}$ (VI)
From (VI) and property P4b we have:
$\Pi(\tau')(\sigma'(x).f) \geq 1$ from (IV) that is:
$\Pi(\tau')(\iota.f) \geq 1$ (VII)
On the other hand, From G2 we have:
$H, \Pi, \sigma, \tau \models P$, that is:
$H, \Pi(\tau), \sigma \models P$ (VIII)
By definition, we know permission masks are a set of mappings from locations and field identifiers to (positive) numbers between 0 and 1 inclusively. That is, we know:
$\Pi(\tau)(\iota.f) \geq 0$ (IX)
From (VII) and (IX) we have:
$\Pi(\tau)(\iota.f) + \Pi(\tau')(\iota.f) \geq 1$ (X)
On the other hand from (G8) we know:
$\Pi(\tau)(\iota.f) + \Pi(\tau')(\iota.f) \leq 1$ (XI)
From (X) and (XI) we know:
$\Pi(\tau)(\iota.f) + \Pi(\tau')(\iota.f) = 1$ (XII)
From (VII) and (XII) we can deduce:
$\Pi(\tau)(\iota.f) = 0$ (XIII)
From (VIII), (XIII) and property P2 we can deduce:
$\mathcal{P}(H, \sigma, P)(\iota.f) \leq 0$
However, from (II) we have:
$\mathcal{P}(H, \sigma, P)(\iota.f) > 0$
Contradiction!!
Hence we can deduce:
$H'(\iota.f) = H(\iota.f)$ (XIV)

RTS. $\forall(\kappa.g)[\mathcal{P}(H, \sigma, P)(\kappa, g) = n \ \wedge\ n > 0 \implies H(\kappa.g) = H'(\kappa.g)]$
Take an arbitrary $\kappa.g$ such that $\mathcal{P}(H, \sigma, P)(\kappa, g) = n \ \wedge\ n > 0$.
RTS. $H(\kappa.g) = H'(\kappa.g)$
Since H' is exactly the same as H except for the value of $\iota.f$, we can deuce:
$H(\kappa.g) = H'(\kappa.g)$ as required. (XV)

From (III), (XIV) and (XV) we can deduce:
$H \overset{\mathcal{P}(H,\sigma,P)}{\equiv} H'$ (XVI)
From (G1), (VIII), (XVI) and the definition of self-framing assertions we can deduce:
$H', \Pi(\tau), \sigma \models P$ (XVII)
From (XVII) and the definition of $\models$ we know:
$H', \Pi, \sigma, \tau \models P$ (XVIII)
Finally, since $\Pi' = \Pi$, from (XVIII) we can deduce:
$H', \Pi', \sigma, \tau \models P$, that is:
$H', \Pi', \sigma, \tau_g \models P$ as required.

Cases VarAss, IfT, IfF, Val, Meth
Since $H' = H$ and $\Pi' = \Pi$, from G2 we can deduce:
$H', \Pi', \sigma, \tau \models P$, that is:
$H', \Pi', \sigma, \tau_g \models P$ as required.

Case NewC
$H' = H[\iota' \mapsto (C, \{f_1 : \text{null} \ldots f_n : \text{null}\}, \tau')]\ \ \iota' \notin H\ f_1 \ldots f_n \in FS(C)$
$\Pi' = \Pi[(\tau')(\iota', f_i) \mapsto 1]\ \ f_i \in FS(C)$

First, we show that $H \overset{\mathcal{P}(H,\sigma,P)}{\equiv} H'$
In other words,

$\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall(\kappa.\mathsf{g}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$

RTS. $\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι,f) such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ (I)
We know from P11 that: $\iota \in \mathsf{Dom}(\mathsf{H})$, that is $\iota \neq \iota'$.
On the other hand, since H' is exactly the same as H except for the value of ι', we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$. (II)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$.
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of ι', we can deuce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (III)

From (II) and (III) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (IV)
From (G2) and the definition of $\models$ we can deduce:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (V)
From (G1), (IV), (V) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (VI)
Since $\Pi'(\tau) = \Pi(\tau)$, we can deduce:
$\mathsf{H}', \Pi'(\tau), \sigma, \models \mathsf{P}$ (VII)
Finally, from (VII) and the definition of the $\models$ judgement, we can deuce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}', \Pi', \sigma, \tau_{\mathsf{g}} \models \mathsf{P}$ as required.

Case Fork
$\mathsf{e}' \equiv \mathsf{fork\ tk} := \mathsf{x.m}(\overline{\mathsf{y}}) \quad \sigma'(\mathsf{x}) = \iota' \quad \mathsf{H}(\iota) \downarrow_1 = \mathsf{C} \quad \mathsf{Pre}(\mathsf{C},\mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}})$
$\tau'' \notin \Pi \quad \Pi'' = \Pi[\tau'' \mapsto \mathsf{ps},\ \tau' - = \mathsf{ps}] \quad \mathsf{ps} = \mathcal{P}(\mathsf{H},\sigma',\mathsf{A}[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])$
$\kappa \notin \mathsf{H} \quad \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\sigma'(\overline{\mathsf{y}})\}, \tau'')]$
$\Pi' = \Pi''[(\tau')(\kappa.\mathsf{g}) \mapsto 1] \quad$ for $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$
In other words:
$\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\wedge \ \forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
where $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$

RTS. $\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι,f) such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
Since H' is exactly the same as H except for the value of κ, we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (I)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $(\kappa'.\mathsf{g})$ s.t. $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
RTS. $\mathsf{H}(\kappa'.\mathsf{g}) = \mathsf{H}'(\kappa'.\mathsf{g})$
Since we have: $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n}$, from property P11 we know that:
$\kappa' \in \mathsf{Dom}(\mathsf{H})$ (II)
Since we have $\kappa \notin \mathsf{Dom}(\mathsf{H})$, from (II) we can deduce:
$\kappa' \neq \kappa$ (III)
Since H' is exactly the same as H except for the value of κ, from (III) we deduce:
$\mathsf{H}'(\kappa') = \mathsf{H}(\kappa)$ as required. (IV)
From (I) and (IV) we can can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (V)
From (G2) and the definition of $\models$ we can deduce:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (VI)
From (V), (VI), (G1) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (VII)
As $\tau'' \notin \Pi$, we know $\tau \neq \tau''$ (VIII)
Since Π' is exactly the same as Π except for the values of τ'' and τ', from (VIII) we know:
$\Pi'(\tau) = \Pi(\tau)$. (IX)
From (VII) and (IX) we can deduce:
$\mathsf{H}', \Pi'(\tau), \sigma, \models \mathsf{P}$ (X)
Finally, from (X) and the definition of the $\models$ judgement, we can deuce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}', \Pi', \sigma, \tau_{\mathsf{g}} \models \mathsf{P}$ as required.

Case Join
$\mathsf{e} \equiv \mathsf{join\ tk} \quad \sigma(\mathsf{tk}) = \kappa \quad \mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau'') \quad \mathsf{Post}(\mathsf{C},\mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}})$
$\Pi' = \Pi[\tau'' - = \mathsf{ps},\ \tau' + = \mathsf{ps}] \quad \mathsf{ps} = \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota', \overline{\mathsf{u}} \mapsto \overline{\iota}], \mathsf{A}(\overline{\mathsf{u}}))$
$\mathsf{H}' = \mathsf{H}[\kappa \mapsto \mathsf{null}]$
$\mathsf{P}' \equiv \mathsf{Thread}(\mathsf{tk},\mathsf{C},\mathsf{m},\mathsf{x}.\overline{\mathsf{y}})$ (I)

First, we show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$, that is:
$\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\wedge \ \forall(\kappa,\mathsf{g}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$

RTS. $\forall(\iota,\mathsf{f}).[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
Take an arbitrary (ι,f) such that $\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\iota,\mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
Since H' is exactly the same as H except for the value of κ, we know that:
$\mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (II)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa,\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $(\kappa'.\mathsf{g})$ such that $\mathcal{P}(\mathsf{H},\sigma',\mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
where $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$. (III)
There are now two cases:

Case 1.
$\kappa' \neq \sigma'(\mathsf{tk})$
RTS. $\mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the value of $\sigma'(\mathsf{tk})$, we can deduce that:
$\mathsf{H}'(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$ (IV)

Case 2. $\kappa' = \sigma'(\mathsf{tk}) = \kappa$
RTS. $\mathsf{H}'(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$
From (G7) and (I) we know:
$\mathsf{H}, \Pi, \sigma', \tau' \models \mathsf{Thread}(\mathsf{tk},\mathsf{C},\mathsf{m},\mathsf{args})$
From the definition of $\models$ we know:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{Thread}(\mathsf{tk},\mathsf{C},\mathsf{m},\mathsf{args})$ (V)
From P12 and (V) we derive:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc}(\mathsf{tk.c},1) * \mathsf{acc}(\mathsf{tk.m},1) * \mathsf{acc}(\mathsf{tk.args},1)$
$* \ \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$ (VI)
From (VI) and applying property P5 three times we have:
$\mathsf{H}, \pi_1, \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (VII)
$\mathsf{H}, \pi_2, \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (VIII)
$\mathsf{H}, \pi_3, \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (IX)
$\mathsf{H}, \pi_4, \sigma' \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$
where $\Pi(\tau') = \pi_1 \uplus \pi_2 \uplus \pi_3 \uplus \pi_4$ (X)
From (VII)-(X) and lemma 13 we have:
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (XI)
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (XII)
$\mathsf{H}, \Pi(\tau'), \sigma' \models \mathsf{acc}(\mathsf{tk.c},1)$ (XIII)
From (X)-(XIII) and property P4b we have:
$\Pi(\tau')(\sigma'(\mathsf{tk}).\mathsf{g}) \geq 1 \quad$ where $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$ (XIV)
By definition, we know permission masks are a set of mappings from locations and field identifiers to (positive) numbers between 0 and 1 inclusively. That is, we know:
$\Pi(\tau)(\kappa.\mathsf{g}) \geq 0$ where $\mathsf{g} \in \{\mathsf{c},\mathsf{m},\mathsf{args}\}$ (XV)
From (XIV) and (XV) we have:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) \geq 1$ (XVI)
On the other hand from (G8) we know:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) \leq 1$ (XVII)
From (XVI) and (XVII) we know:
$\Pi(\tau)(\kappa.\mathsf{g}) + \Pi(\tau')(\kappa.\mathsf{g}) = 1$ (XVIII)
From (XIV) and (XVIII) we can deduce:
$\Pi(\tau)(\kappa.\mathsf{g}) = 0$ (XIX)
On the other hand, From G2 we have:
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{P}$ (XX)
From (XX), (XIX) and property P2 we can deduce:
$\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa.\mathsf{g}) \leq 0$
However, from (III) we have:
$\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})(\kappa.\mathsf{g}) > 0$
Contradiction!!
Hence we can deduce:
$\mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$ (XXI)
From (II), (IV) and (XXI) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{P})}{\equiv} \mathsf{H}'$ (XXII)
From (G1), (XX), (XXII) and the definition of self-framing assertions we can deduce:
$\mathsf{H}', \Pi(\tau), \sigma \models \mathsf{P}$ (XXIII)
From (XXIII) and the definition of $\models$ we know:
$\mathsf{H}', \Pi, \sigma, \tau \models \mathsf{P}$ (XXIV)
Finally, since $\Pi' = \Pi$, from (XXIV) we can deduce:
$\mathsf{H}', \Pi', \sigma, \tau \models \mathsf{P}$, that is:
$\mathsf{H}', \Pi', \sigma, \tau_{\mathsf{g}} \models \mathsf{P}$ as required.

Case Seq.
$\mathsf{e} \equiv \mathsf{e}_1; \mathsf{e}_2$
$\{\mathsf{P}'\}\ \mathsf{e}_1\ \{\mathsf{R}\}$ (I)

$\{R\}\ e_2\ \{Q'\}$

RTS. $H', \Pi', \sigma, \tau \models P$
Given the definition of e', from the operational semantics of the (Seq.) rule we know:
$H, \Pi, (e_1, \sigma', \tau') \rightsquigarrow (e_1', \sigma'', \tau'), H', \Pi'$ (II)
From (G1), (G2), (G3), (G4), (II), (I), (G7), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$, that is:
$H', \Pi', \sigma, \tau_g \models P$ as required.

Case Frame
$P' \equiv A * C$ (I)
$\{A\}\ e'\ \{B\}$ (II)

RTS. $H', \Pi', \sigma, \tau \models P$
From (G7) and (I) we have:
$H, \Pi, \sigma', \tau' \models A * C$ (III)
From (III) and the definition of $\models$ we have:
$H, \Pi(\tau'), \sigma' \models A * C$ (IV)
From (IV) and property P5 we have:
$H, \pi_1, \sigma' \models A$ (V)
$H, \pi_2, \sigma' \models C$
$\Pi(\tau') = \pi_1 \uplus \pi_2$ (VI)
From (V), (VI) and lemma 13 we have:
$H, \Pi(\tau'), \sigma' \models A$ (VII)
From (VII) and the definition of $\models$ we have:
$H, \Pi, \sigma', \tau' \models A$ (VIII)
From (G1), (G2), (G3), (G4), (G5), (II), (VIII), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$, that is:
$H', \Pi', \sigma, \tau_g \models P$ as required.

Case Conseq.
$P' \rightarrow_a A$ (I) $\{A\}\ e'\ \{B\}$ (II)

RTS. $H', \Pi', \sigma, \tau \models P$
From (G7), (I) and property P3 we have:
$H, \Pi, \sigma', \tau' \models A$ (III)
From (G1), (G2), (G3), (G4), (G5), (II), (III), (G8) and the induction hypothesis we can deduce:
$H', \Pi', \sigma, \tau \models P$, that is:
$H', \Pi', \sigma, \tau_g \models P$ as required.

C.7 Lemma 7

$\forall\ e \in Prog[Ver(Prog) \ \wedge\ \{P\}\ e\ \{Q\} \implies SF(P) \ \wedge\ SF(Q)]$
Proof. By induction over the $\{P\}\ e\ \{Q\}$ derivations.

Case FAss. $P \equiv acc(x.f, 1)\ \ e \equiv x.f := y\ \ Q \equiv acc1) * x.f = y$

RTS. $SF(acc(x.f, 1)) \ \wedge\ SF(acc(x.f, 1) * x.f = y)$
From Lemma 10 we can deduce:
$SF(acc(x.f, 1))$ (I)
From Lemma 11 we can deduce:
$SF(acc(x.f, 1) * x.f = y)$ (II)
From (I) and (II) we can deduce:
$SF(acc(x.f, 1)) \ \wedge\ SF(acc(x.f, 1) * x.f = y)$ as required.

Case VAss. $P \equiv acc(x.f, z)\ \ e \equiv y := x.f\ \ Q \equiv acc(x.f, z) * y = x.f$ where $z > 0$

RTS. $SF(acc(x.f, z)) \ \wedge\ SF(acc(x.f, z) * y = x.f)$
From Lemma 10 we can deduce:
$SF(acc(x.f, z))$ (I)
From Lemma 11 and since $z > 0$ we can deduce:
$SF(acc(x.f, z) * y = x.f)$ (II)
From (I) and (II) we can deduce:
$SF(acc(x.f, z)) \ \wedge\ SF(acc(x.f, z) * y = x.f)$ as required.

Case If $e \equiv$ if(B) then C1 else C2
$\{B \wedge P\}$ C1 $\{Q\}$ (G1) $\qquad$ $\{\neg B \wedge P\}$ C2 $\{Q\}$ (G2)

RTS. $SF(P) \ \wedge\ SF(Q)$
By induction hypothesis from G1 we can deduce:
$SF(B \wedge P)$ (I)
$SF(Q)$ (II)
Since B represents a case split assertion, from (I) and property P6 we can deduce:
$SF(P)$ (III)
From (II) and (III) we can deduce:
$SF(P) \ \wedge\ SF(Q)$ as required.

Case Meth $P \equiv Pre(m)[x/this][\overline{y}/\overline{u}]\ \ e \equiv x.m(\overline{y})\ \ Q \equiv Post(m)[x/this][\overline{y}/\overline{u}]$

RTS. $SF(Pre(m)[x/this][\overline{y}/\overline{u}]) \ \wedge\ SF(Post(m)[x/this][\overline{y}/\overline{u}])$
From the premise of the lemma we know: Ver(Prog)
From the definition of Ver we know:
$SF(Pre(m))$ (I)
$SF(Post(m))$ (II)
From Lemma 8, (I) and (II) we deduce:
$SF(Pre(m)[x/this][\overline{y}/\overline{u}])$ (III)
$SF(Post(m)[x/this][\overline{y}/\overline{u}])$ (IV)
Hence we can deduce:
$Pre(m)[x/this][\overline{y}/\overline{u}] \ \wedge\ Post(m)[x/this][\overline{y}/\overline{u}]$ as required.

Case Val. $e \equiv v$
SF(P) (G1)

RTS. $SF(P) \ \wedge SF(P)$
From (G1) we trivially have $SF(P)$ as required.

Case New $P \equiv True\ \ e \equiv x := new\ C$
$Q \equiv *acc(x.f_i, 1) * x.f_i = null\ \ for\ f_i \in FS(C)$

RTS. $SF(True) \ \wedge\ SF(*[acc(x.f_i, 1) * x.f_i = null])$ for $f_i \in FS(C)$
We trivially have: $SF(True)$ (I)
From Lemma 11 we have:
$SF(*[acc(x.f_i, 1) * x.f_i = null])$ (II)
From (I) and (II) we have:
$SF(True) \ \wedge\ SF(*[acc(x.f_i, 1) * x.f_i = null])$ for $f_i \in FS(C)$ as required.

Case Seq $e \equiv C1; C2$
$\{P\}$ C1 $\{R\}$ (G1) $\qquad$ $\{R\}$ C2 $\{Q\}$ (G2)

RTS. $SF(P) \ \wedge\ SF(Q)$
By induction hypothesis from G1 we can deduce:
$SF(P)$ (I)
By induction hypothesis from G2 we can deduce:
$SF(Q)$ (II)
From (I) and (II) we can deduce:
$SF(P) \ \wedge\ SF(Q)$ as required.

Case Acq. $P \equiv True\ \ e \equiv acquire\ x\ \ Q \equiv A[x/this]$ where A is the invariant of x's class.

RTS. $SF(True) \ \wedge\ SF(A[x/this])$
We trivially know: $SF(True)$ (I)
From the premise of the lemma we know: $Ver(P)$ and by definition of Ver, we know:
$SF(A)$ (II)
From II and lemma 8 we can deduce:
$SF(A[x/this])$ (III)
from I and III we can deduce: $SF(True) \ \wedge\ SF(A[x/this])$ as required.

Case Rel $P \equiv A[x/this]\ \ e \equiv release\ x\ \ Q \equiv True$ where A is the invariant of x's class.

RTS. $SF(A[x/this]) \ \wedge\ SF(True)$
Proof of this case is analogous to the previos case and is left out.

Case Fork $P \equiv Pre(m)[x/this][\overline{y}/\overline{u}]\ \ e \equiv fork\ tk := x.m(\overline{y})$
$Q \equiv Thread(tk, C, m, x.\overline{y})\ \ class(m) = C$

RTS. $SF(Pre(m)[x/this][\overline{y}/\overline{u}]) \ \wedge\ SF(Thread(tk, C, m, x.\overline{y}))$
From the premise of the lemma we know: Ver(Prog)
From the definition of Ver we know:
$SF(Pre(m))$ (I)
From Lemma 8 and (I) we deduce:
$SF(Pre(m)[x/this][\overline{y}/\overline{u}])$ (II)
From Lemma 12, we can deduce:
$SF(Thread(tk, C, m, x.\overline{y}))$ (III)
From (II) and (III) we can deduce:
$SF(Pre(m)[x/this][\overline{y}/\overline{u}]) \ \wedge\ SF(Thread(tk, C, m, x.\overline{y}))$ as required.

Case Join $P \equiv Thread(tk, C, m, x.\overline{y})$
$e \equiv join\ tk\ \ Post(C, m) = A(\overline{u})\ \ Q \equiv A[x/this][\overline{y}/\overline{u}]$

RTS. $SF(Thread(tk, C, m, x.\overline{y})) \ \wedge\ SF(A[x/this][\overline{y}/\overline{u}])$
From the premise of the lemma we know: Ver(Prog)
From the definition of Ver we know:
$SF(Post(m))$ (I)
From Lemma 8 and (I) we deduce:
$SF(Post(m)[x/this][\overline{y}/\overline{u}])$ (II)
From Lemma 12, we can deduce:

$\mathsf{SF}(\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}))$ (III)
From (II) and (III) we can deduce:
$\mathsf{SF}(\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})) \ \wedge \ \mathsf{SF}(\mathsf{A}[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])$ as required.

Case Con.
$\mathsf{SF}(\mathsf{P}) \ \wedge \ \mathsf{SF}(\mathsf{Q})$ (G1)

RTS. $\mathsf{SF}(\mathsf{P}) \ \wedge \ \mathsf{SF}(\mathsf{Q})$
This is trivially obtained from (G1).

Case Frm.
$\{\mathsf{P}\}\ \mathsf{e}\ \{\mathsf{Q}\}$ (G1)
$\mathsf{SF}(\mathsf{R})$ (G2)

RTS. $\mathsf{SF}(\mathsf{P} * \mathsf{R}) \ \wedge \mathsf{SF}(\mathsf{Q} * \mathsf{R})$
From G1 and by the induction hypothesis, we can deduce:
$\mathsf{SF}(\mathsf{P})$ (I)
$\mathsf{SF}(\mathsf{Q})$ (II)
From G2, (I) and P8, we can deduce:
$\mathsf{SF}(\mathsf{P} * \mathsf{R})$ (III)
From G2, (II) and P8, we can deduce:
$\mathsf{SF}(\mathsf{Q} * \mathsf{R})$ (IV)
From (III) and (IV), we deduce:
$\mathsf{SF}(\mathsf{P} * \mathsf{R}) \ \wedge \ \mathsf{SF}(\mathsf{Q} * \mathsf{R})$ as required.

C.8 Lemma 8

$\forall\ \mathsf{A}.[\mathsf{SF}(\mathsf{A}) \implies \mathsf{SF}(\mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{y}}])]$

Proof. By Contradiction.
Take $\mathsf{A}' \equiv \mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{y}}]$ (I), where we have: $\neg\mathsf{SF}(\mathsf{A}')$
By definition of SF, this means:
$\exists \mathsf{H}, \mathsf{H}', \pi, \sigma', \iota.\mathsf{f}[\mathsf{H}, \pi, \sigma' \models \mathsf{A}' \ \wedge \ \mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma', \mathsf{A}')}{\equiv} \mathsf{H}' \ \wedge \ \mathsf{H}'(\iota.\mathsf{f}) \neq \mathsf{H}(\iota.\mathsf{f})]$ (II)
Take $\sigma \equiv \sigma'[\overline{\mathsf{y}} \mapsto \overline{\sigma'(\mathsf{x})}]$
In other words: $\sigma' = \sigma[\overline{\mathsf{x}} \mapsto \overline{\sigma(\mathsf{y})}]$ (III)
From (I), (II), (III) and property P1a we know:
$\mathsf{H}, \pi, \sigma \models \mathsf{A}$ (IV)
From (I), (II), (III) and property P1b we know:
$\mathcal{P}(\mathsf{H}, \sigma', \mathsf{A}') = \mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})$ (V)
From (IV) and (V), we can rewrite (II) as:
$\exists \mathsf{H}, \mathsf{H}', \pi, \sigma, \iota.\mathsf{f}[\mathsf{H}, \pi, \sigma \models \mathsf{A} \ \wedge \ \mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})}{\equiv} \mathsf{H}' \ \wedge \ \mathsf{H}'(\iota.\mathsf{f}) \neq \mathsf{H}(\iota.\mathsf{f})]$
In other words: $\neg\mathsf{SF}(\mathsf{A})$
However, from the premise of the lemma we know: $\mathsf{SF}(\mathsf{A})$
Contradiction!
We can hence deduce: $\mathsf{SF}(\mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{y}}])]$ as required.

C.9 Lemma 9

$\mathsf{H}, \pi, \sigma_1 \models \mathsf{A} \implies \mathsf{H}, \pi, \sigma_1 \uplus \sigma_2 \models \mathsf{A}$

Proof.
Assume: $\mathsf{H}, \pi, \sigma_1 \models \mathsf{A}$ (I)
RTS. $\mathsf{H}, \pi, \sigma_1 \uplus \sigma_2 \models \mathsf{A}$

Let $\mathsf{FV}(\mathsf{A}) = \overline{\mathsf{x}}$ (II)
and $\sigma' = \sigma_1 \uplus \sigma_2$. (III)
From the definition of $\sigma_1 \uplus \sigma_2$, we know:
$\sigma'(\overline{\mathsf{x}}) = \sigma(\overline{\mathsf{x}})$ (IV)
From (I), (II), (IV) and property P1a we can derive:
$\mathsf{H}, \pi, \sigma' \models \mathsf{A}[\overline{\mathsf{x}}/\overline{\mathsf{x}}]$, in other words:
$\mathsf{H}, \pi, \sigma' \models \mathsf{A}$.
Finally, from (III) we have:
$\mathsf{H}, \pi, \sigma_1 \uplus \sigma_2 \models \mathsf{A}$ as required.

C.10 Lemma 10

$\forall \mathsf{x}, \mathsf{f}, \mathsf{n}[\mathsf{SF}(\mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n}))]$

Proof.
Take an arbitrary x, f, n.

RTS. $\forall \mathsf{H}, \mathsf{H}', \sigma, \pi[\mathsf{H}, \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n}) \ \wedge \ \mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n}))}{\equiv} \mathsf{H}' \implies \mathsf{H}', \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n})]$

Take an arbitrary $\mathsf{H}, \mathsf{H}', \sigma, \pi$ such that:
$\mathsf{H}, \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n})$ (I)
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n}))}{\equiv} \mathsf{H}'$
RTS. $\mathsf{H}', \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n})$
From (I) and property P4b we can deduce:
$\pi(\sigma(\mathsf{x}), \mathsf{f}) \geq \mathsf{n}$ (II)
From (II) and property P4b we can deduce: $\mathsf{H}', \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n})$, that is:
$\mathsf{SF}(\mathsf{acc}(\mathsf{x}.\mathsf{f}, \mathsf{n}))$ as required.

C.11 Lemma 11

$\forall \mathsf{x}, \mathsf{f}_\mathsf{i}, \mathsf{y}_\mathsf{i}, \mathsf{n}_\mathsf{i}.[\bigwedge(\mathsf{n}_\mathsf{i} > 0) \implies \mathsf{SF}(*(\mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{i}, \mathsf{n}_\mathsf{i}) * \mathsf{x}.\mathsf{f}_\mathsf{i} = \mathsf{y}_\mathsf{i}))]$
for $\mathsf{i} \in \{1...\mathsf{m}\}$

Proof.
Take arbitrary $\mathsf{x}, \mathsf{f}_1, ..., \mathsf{f}_\mathsf{m}, \mathsf{y}_1, ..., \mathsf{y}_\mathsf{m}, \mathsf{n}_1, .., \mathsf{n}_\mathsf{m}$ such that:
$\mathsf{n}_1 > 0 \ \wedge \ ... \ \wedge \ \mathsf{n}_\mathsf{m} > 0$
RTS. $\mathsf{SF}(\mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m})$
In other words,

RTS. $\forall \mathsf{H}, \mathsf{H}', \pi, \sigma[\mathsf{H}, \pi, \sigma \models \mathsf{A} \ \wedge \ \mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})}{\equiv} \mathsf{H}' \implies \mathsf{H}', \pi, \sigma \models \mathsf{A}]$
where $\mathsf{A} \equiv \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m})$

Take an arbitrary $\mathsf{H}, \mathsf{H}', \pi, \sigma$ such that:
$\mathsf{H}, \pi, \sigma \models (\mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m})$ (I)
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, (\mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m}))}{\equiv} \mathsf{H}'$ (II)

RTS. $\mathsf{H}', \pi, \sigma \models (\mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m})$
From (I) and applying property P5 m times we can deduce:
$\mathsf{H}, \pi_1, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1$ (1)
...
$\mathsf{H}, \pi_\mathsf{m}, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m}$ (m)
where $\pi = \pi_1 \uplus ... \uplus \pi_\mathsf{m}$ (III)

From (1) and property P5 we know:
$\mathsf{H}, \pi_\mathsf{a}, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1)$ (IV)
$\mathsf{H}, \pi_\mathsf{b}, \sigma \models \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1$ (V)
where $\pi_1 = \pi_\mathsf{a} \uplus \pi_\mathsf{b}$ (VI)
From (IV) and property P4b we can deduce:
$\pi_\mathsf{a}(\sigma(\mathsf{x}), \mathsf{f}_1) \geq \mathsf{n}_1$ (VII)
From (VII) and property P4b we can deduce:
$\mathsf{H}', \pi_\mathsf{a}, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1)$ (VIII)
On the other hand, from (1) and property P9 we have:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1)(\sigma(\mathsf{x}).\mathsf{f}_1) \geq \mathsf{n}_1$ (IX):
We also know from our assumption that: $\mathsf{n}_1 > 0$ (X)
From (II), (IX), (X) and the definition of $\overset{\pi}{\equiv}$ we can deduce:
$\mathsf{H}'(\sigma(\mathsf{x}).\mathsf{f}_1) = \mathsf{H}(\sigma(\mathsf{x}).\mathsf{f}_1)$ (XI)
From (V) and property P4a we can deduce:
$\mathsf{H}(\sigma(\mathsf{x}).\mathsf{f}_1) = \sigma(\mathsf{y}_1)$ (XII)
From (XI) and (XII) we can deduce:
$\mathsf{H}'(\sigma(\mathsf{x}).\mathsf{f}_1) = \sigma(\mathsf{y}_1)$ (XIII)
From (XIII) and property P4a we can deduce:
$\mathsf{H}', \pi_\mathsf{b}, \sigma \models \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1$ (XIV)
From (VI), (VIII), (XIV) and property P5 we can deduce:
$\mathsf{H}', \pi_1, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1$ (XV)

In the similar manner to (IV)-(XV), we can show that:
$\mathsf{H}', \pi_2, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_2, \mathsf{n}_2) * \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1$ (2-a)
...
$\mathsf{H}', \pi_\mathsf{m}, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) * \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m}$ (m-a)
From (XV), (2-a),..., (m-a), (III) and property P5 we can deduce:
$\mathsf{H}', \pi, \sigma \models \mathsf{acc}(\mathsf{x}.\mathsf{f}_1, \mathsf{n}_1) \ * \ \mathsf{x}.\mathsf{f}_1 = \mathsf{y}_1 * ... * \mathsf{acc}(\mathsf{x}.\mathsf{f}_\mathsf{m}, \mathsf{n}_\mathsf{m}) \ * \ \mathsf{x}.\mathsf{f}_\mathsf{m} = \mathsf{y}_\mathsf{m}$
as required.

C.12 Lemma 12

$\forall \mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}, \overline{\mathsf{y}}.[\mathsf{SF}(\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}))]$

Proof.
Take an arbitrary $\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}, \overline{\mathsf{y}}$
RTS. $\mathsf{SF}(\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}))$
In other words, show:

$\forall \mathsf{H}, \mathsf{H}', \pi, \sigma.[\mathsf{H}, \pi, \sigma \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}) \wedge \mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}))}{\equiv} \mathsf{H}'$
$\implies \mathsf{H}', \pi, \sigma \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})]$
Take an arbitrary $\mathsf{H}, \mathsf{H}', \pi, \sigma$ such that:
$\mathsf{H}, \pi, \sigma \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$ (I)
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}))}{\equiv} \mathsf{H}'$ (II)
RTS. $\mathsf{H}', \pi, \sigma \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$
From (I) an property P12 we have:
$\mathsf{H}, \pi, \sigma \models \mathsf{acc}(\mathsf{tk.c}, 1) \ * \ \mathsf{tk.c} = \mathsf{C}$
$\quad * \ \mathsf{acc}(\mathsf{tk.m}, 1) \ * \ \mathsf{tk.m} = \mathsf{m}$
$\quad * \ \mathsf{acc}(\mathsf{tk.args}, 1) \ * \ \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$ (III)
From (III) and applying property P5 twice, we have:
$\mathsf{H}, \pi_1, \sigma \models \mathsf{acc}(\mathsf{tk.c}, 1) * \mathsf{tk.c} = \mathsf{C}$ (IV)

$\mathsf{H}, \pi_2, \sigma \models \mathsf{acc(tk.m, 1)} * \mathsf{tk.m} = \mathsf{m}$ (V)
$\mathsf{H}, \pi_3, \sigma \models \mathsf{acc(tk.args, 1)} * \mathsf{tk.args} = \mathsf{args}$ (VI)
where $\pi = \pi_1 \uplus \pi_2 \uplus \pi_3$ (VII)
From (IV) and property P5 we know:
$\mathsf{H}, \pi_4, \sigma \models \mathsf{acc(tk.c, 1)} \ \wedge \ \mathsf{H}, \pi_5, \sigma \models \mathsf{tk.c} = \mathsf{C}$ (VIII)
for some π_4, π_5 s.t. $\pi_1 = \pi_4 \uplus \pi_5$ (IX)
From (VIII) we have:
$\mathsf{H}, \pi_4, \sigma \models \mathsf{acc(tk.c, 1)}$ (X)
$\mathsf{H}, \pi_5, \sigma \models \mathsf{tk.c} = \mathsf{C}$ (XI)
$\pi_1 = \pi_4 \uplus \pi_5$ (XII)
From (X) and property P4b we can deduce:
$\pi_4(\sigma(\mathsf{tk}), \mathsf{c}) \geq 1$ (XIII)
From (XIII) and property P4b we can deduce:
$\mathsf{H}', \pi_4, \sigma \models \mathsf{acc(tk.c, 1)}$ (XIV)
On the other hand, from (IV) and property P9 we have:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{acc(tk.c, 1)} * \mathsf{tk.c} = \mathsf{C})(\sigma(\mathsf{tk}).\mathsf{c}) \geq 1$ (XV):
From (II), (XV) and the definition of $\overset{\pi}{\equiv}$ we can deduce:
$\mathsf{H}'(\sigma(\mathsf{tk}).\mathsf{c}) = \mathsf{H}(\sigma(\mathsf{tk}).\mathsf{c})$ (XVII)
From (XI) and property P4a we can deduce:
$\mathsf{H}(\sigma(\mathsf{tk}).\mathsf{c}) = \mathsf{C}$ (XVIII)
From (XVII) and (XVIII) we can deduce:
$\mathsf{H}'(\sigma(\mathsf{tk}).\mathsf{c}) = \mathsf{C})$ (XIX)
From (XIX) and property P4a we can deduce:
$\mathsf{H}', \pi_5, \sigma \models \mathsf{tk.c} = \mathsf{C}$ (XX)
From (XII), (XIV), (XX) and property P5 we can deduce:
$\mathsf{H}', \pi_1, \sigma \models \mathsf{acc(tk.c, 1)} * \mathsf{tk.c} = \mathsf{C}$ (XXI)

In the similar manner to (VIII)-(XXI), we can show that:
$\mathsf{H}', \pi_2, \sigma \models \mathsf{acc(tk.m, 1)} * \mathsf{tk.m} = \mathsf{m}$ (XXII)
$\mathsf{H}', \pi_3, \sigma \models \mathsf{acc(tk.args, 1)} * \mathsf{tk.args} = \mathsf{x.\bar{y}}$ (XXIII)
From (XXI), (XXII), (XXIII), (VII) and property P5 we can deduce:
$\mathsf{H}', \pi, \sigma \models \mathsf{acc(tk.c, 1)} \ * \ \mathsf{tk.c} = \mathsf{C}$
$* \ \mathsf{acc(tk.m, 1)} \ * \ \mathsf{tk.m} = \mathsf{m}$
$* \ \mathsf{acc(tk.args, 1)} \ * \ \mathsf{tk.args} = \mathsf{x.\bar{y}}$ that is:
$\mathsf{H}', \pi, \sigma \models \mathsf{Thread(tk, C, m, x.\bar{y})}$ (XXIV)
Hence from (I), (II), (XXIV) and the definition of SF we can deduce:
$\mathsf{SF(Thread(tk, C, m, x.\bar{y}))}$ as required

C.13 Lemma 13

a. $\mathsf{H}, \pi_1, \sigma \models \mathsf{A} \implies \mathsf{H}, \pi_1 \uplus \pi_2, \sigma \models \mathsf{A}$

Proof.
Assume: $\mathsf{H}, \pi_1, \sigma \models \mathsf{A}$ (I)
RTS. $\mathsf{H}, \pi_1 \uplus \pi_2, \sigma \models \mathsf{A}$
We know:$\forall \mathsf{H}', \pi', \sigma'.[\mathsf{H}', \pi', \sigma' \models \mathsf{True}]$
Hence, we can derive $\mathsf{H}, \pi_2, \sigma \models \mathsf{True}$ (II)
From (I), (II) and property P5 we can deduce:
$\mathsf{H}, \pi_1 \uplus \pi_2, \sigma \models \mathsf{A} * \mathsf{True}$ (III)
Since $\mathsf{A} * \mathsf{True} \equiv \mathsf{A}$, we can deduce:
$\mathsf{H}, \pi_1 \uplus \pi_2, \sigma \models \mathsf{A}$ as required.

b. $\mathsf{H}, \pi_1, \sigma \models \mathsf{A} \ \wedge \ \pi_1 \subseteq \pi_2 \implies \mathsf{H}, \pi_2, \sigma \models \mathsf{A}$

Proof.
Assume: $\mathsf{H}, \pi_1, \sigma \models \mathsf{A}$ (I)
$\pi_1 \subseteq \pi_2$ (II)
RTS. $\mathsf{H}, \pi_2, \sigma \models \mathsf{A}$

Let us define permission mask π_3 as follows.
$\mathsf{Dom}(\pi_3) = \mathsf{Dom}(\pi)$

$$\pi_3(\iota.f) = \begin{cases} \pi_2(\iota.f) & \text{if } (\iota.f) \notin Dom(\pi_1) \\ \pi_2(\iota.f) - \pi_1(\iota.f) & \text{otherwise} \end{cases}$$

Then from (II) and the definition of π_3 we have: $\pi_2 = \pi_1 \uplus \pi_2$. (III)
From (I), (III) and lemma 13a we can deduce:
$\mathsf{H}, \pi_2, \sigma \models \mathsf{A}$ as required.

C.14 Lemma 14

$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{Thread(tk, C_1, m_1, x.\bar{y})}$ (G1)
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{Thread(tk, C_2, m_2, w.\bar{z})}$ (G2)
$\implies$
$\mathsf{C_1} = \mathsf{C_2} \quad \wedge \quad \mathsf{m_1} = \mathsf{m_2} \quad \wedge \quad \sigma(\mathsf{x}) = \sigma(\mathsf{w}) \quad \wedge \quad \overline{\sigma(\mathsf{y})} = \overline{\sigma(\mathsf{z})}$

Proof.
From (G1) and the definition of the $\models$ judgement we have:
$\mathsf{H}, \Pi(\tau), \sigma \models \mathsf{Thread(tk, C_1, m_1, x.\bar{y})}$ (1)
From (1) and property P12 we have:
$\mathsf{H}, \Pi, \sigma, \tau \models \mathsf{acc(tk.c, 1)} * \mathsf{tk.c} = \mathsf{C_1}$
$* \ \mathsf{acc(tk.m, 1)} * \mathsf{tk.m} = \mathsf{m_1}$
$* \ \mathsf{acc(tk.args, 1)} * \mathsf{tk.args} = \sigma(\mathsf{x}).\overline{\sigma(\mathsf{y})}$ (2)
From (2) and property P5 we have:
$\mathsf{H}, \pi_1, \sigma \models \mathsf{tk.c} = \mathsf{C_1}$ (3)
$\mathsf{H}, \pi_2, \sigma, \tau \models \mathsf{acc(tk.c, 1)}$
$* \ \mathsf{acc(tk.m, 1)} * \mathsf{tk.m} = \mathsf{m_1}$
$* \ \mathsf{acc(tk.args, 1)} * \mathsf{tk.args} = \sigma(\mathsf{x}).\overline{\sigma(\mathsf{y})}$ (4)
$\Pi(\tau) = \pi_1 \uplus \pi_2$ (5)
From (3) we have:
$\mathsf{H}(\sigma(\mathsf{tk})) \downarrow_2 = \mathsf{C_1}$ (6)
In the similar fashion to (1)-(6), we can deduce:
$\mathsf{H}(\sigma(\mathsf{tk})) \downarrow_2 = \mathsf{C_2}$ (7)
From (6) and (7) we can deduce:
$\mathsf{C_1} = \mathsf{C_2}$ (8)
In the similar fashion to (1)-(8) we can deduce:
$\mathsf{m_1} = \mathsf{m_2}$ (9)
$\sigma(\mathsf{x}) = \sigma(\mathsf{w}) \quad \wedge \quad \overline{\sigma(\mathsf{y})} = \overline{\sigma(\mathsf{z})}$ (10)
From (8), (9) and (10) we have:
$\mathsf{C_1} = \mathsf{C_2} \quad \wedge \quad \mathsf{m_1} = \mathsf{m_2} \quad \wedge \quad \sigma(\mathsf{x}) = \sigma(\mathsf{w}) \quad \wedge \quad \overline{\sigma(\mathsf{y})} = \overline{\sigma(\mathsf{z})}$
as required.

D. SOUNDNESS THEOREM PROOF

In order to prove Theorem 1, we first start by proving the following auxiliary theorem that would help us in proving the first soundness theorem.

D.1 Theorem 1a

$\mathsf{Ver(Prog)}$ (G1)
$\{\mathsf{P_i}\} \ \mathsf{e_i} \ \{\mathsf{Q_i}\} \quad \mathsf{for \ i \in 0...n}$ (G2)
$\mathsf{H}, \Pi, \sigma_i, \tau_i \models \mathsf{P_i} \quad \mathsf{for \ i \in 0...n}$ (G3)
$(\mathsf{e_0}, \tau_0, \sigma_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1\cdots n}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e_0'}, \tau_0, \sigma_0')|(\overline{\mathsf{e}, \tau, \sigma}^{1\cdots n}), \mathsf{H}', \Pi'$ (G4)
$\mathsf{WF(H, \Pi, (\overline{e, \tau, \sigma}^{0\cdots n}))}$ (G5)
$\implies$
1. $\mathsf{H}', \Pi', \sigma_i', \tau_i \models \mathsf{P_i} \quad \mathsf{for \ i \in 1...n}$
2. $\exists \mathsf{P_0'}.[\ \{\mathsf{P_0'}\} \ \mathsf{e_0'} \ \{\mathsf{Q_0}\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{P_0'} \ ]$
3a. $\forall \ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c : C, m : m, args : } \iota.\bar{\iota}\}, \tau) \ \wedge \ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \ \{\mathsf{P}\} \ \mathsf{e_j} \ \{\mathsf{Post(m)}\}] \ \wedge \ \sigma_\mathsf{j}(\mathsf{this}) = \iota \ \wedge \ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota} \ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\bar{\mathsf{u}})$

3b. $\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$

3c. $\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models * \mathsf{A_i'}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom(H')} \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i/this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

3d. $\forall \kappa, \kappa'.[\mathsf{H}(\kappa) \downarrow_3 = \tau \ \wedge \ \mathsf{H}(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$

Proof. By induction over the Hoare Logic Triplets.
In all the following cases (1) is obtained from Lemma 4.

Furthermore, in all the following cases we know that:
$\tau_0 \neq \tau_\mathsf{g}$ (G6)
since τ_g is the ghost thread that does not take part in execution. In other words, we cannot have:
$\mathsf{H}, \Pi, (\mathsf{e_0}, \tau_\mathsf{g}, \sigma_0) \rightsquigarrow (\mathsf{e_0'}, \tau_\mathsf{g}, \sigma_0'), \mathsf{H}', \Pi'$.

Moreover, since from G2 we have:
$\{\mathsf{P_i}\} \ \mathsf{e_i} \ \{\mathsf{Q_i}\} \quad \mathsf{for \ i \in 0...n}$
From Lemma 7 we an deduce:
$\mathsf{SF(P_i)} \quad \mathsf{for \ i \in 0...n}$ (G7)

Case FldAss.
$\mathsf{P_0} \equiv \mathsf{acc(x.f, 1)} \quad \mathsf{e_0} \equiv \mathsf{x.f := y} \quad \mathsf{Q_0} \equiv \mathsf{acc(x.f, 1)} * \mathsf{x.f = y}$
$\mathsf{H}' = \mathsf{H}[\sigma(\mathsf{x}) \downarrow_2 (\mathsf{f}) \mapsto \mathsf{y}] \quad \Pi' = \Pi \quad \sigma_0' = \sigma_0 \quad \mathsf{e'} = \mathsf{null}$

RTS. 2. $\exists \mathsf{P}_0'.[\ \{\mathsf{P}_0'\}\ \mathsf{null}\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{P}_0'\ ]$

Take $\mathsf{P}_0' = \mathsf{Q}_0 = \mathsf{acc}(\mathsf{x.f}, 1) * \mathsf{x.f} = \mathsf{y}$ then we clearly have:
$\{\mathsf{P}_0'\}\ \mathsf{null}\ \{\mathsf{Q}_0\}$.
We know that $\Pi'(\tau)(\mathsf{x.f}) = \Pi(\tau)(\mathsf{x.f})$ and since
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{acc}(\mathsf{x.f}, 1)$, we can deduce $\mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{acc}(\mathsf{x.f}, 1)$.
Furthermore since $\mathsf{H}' = \mathsf{H}[(\mathsf{x}) \downarrow_2 (\mathsf{f}) \mapsto \sigma_0(\mathsf{y})]$, we know
$\mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{x.f} = \mathsf{y}$ holds.
Therefore, $\mathsf{H}', \Pi', \sigma_0', \tau_0 \models (\mathsf{acc}(\mathsf{x.f}, 1) * \mathsf{x.f} = \mathsf{y})$ holds.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\bar{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$. (I)
$\tau \neq \tau_0$ (II)
RTS. $\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\bar{\mathsf{u}})$

Since H' is exactly the same as H except for the value of $\sigma(\mathsf{x}).\mathsf{f}$, from (I) we have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$.
Now, from G5, we can deduce:
$\exists \mathsf{P} \exists \mathsf{j} \in \{1...\mathsf{n}\}[\tau = \tau_\mathsf{j}\ \wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}$ (III)
$\mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}\ ]$ (IV)
$\sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}$ (V)
From (III), (IV), (II), (G2), (G3), (G4), (G5) and applying Lemma 4 we get:
$\mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$ (VI)
Finally, by putting together (III), (V) and (VI) we can deduce:
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$
as required.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
Since $\Pi' = \Pi$ from G5 we can deduce: $\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$ (1)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (2) where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_\mathsf{g}$.
Since H' is exactly the same as H, except for the value of ι.f, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$ (5)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (6) where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (7)

Finally, from (5), (G2)-(G7) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since H' is exactly the same as H except for the value of $\sigma(\mathsf{x}).\mathsf{f}$, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case VarAss.
$\mathsf{P}_0 \equiv \mathsf{acc}(\mathsf{x.f}, \mathsf{z})$ $\quad$ $\mathsf{e}_0 \equiv \mathsf{y} := \mathsf{x.f}$ $\quad$ $\mathsf{Q}_0 \equiv \mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}$ where $\mathsf{z} > 0$
$\mathsf{H}' = \mathsf{H}$ $\quad$ $\Pi' = \Pi$ $\quad$ $\mathsf{v} = \mathsf{H}(\sigma_0(\mathsf{x})) \downarrow_2 (\mathsf{f})$ $\quad$ $\sigma_0' = \sigma_0[\mathsf{y} \mapsto \mathsf{v}]$ $\quad$ $\mathsf{e}' = \mathsf{v}$

RTS. 2. $\exists \mathsf{P}_0'.[\ \{\mathsf{P}_0'\}\ \mathsf{null}\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{P}_0'\ ]$

Take $\mathsf{P}_0' = \mathsf{Q}_0 = \mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}$ where $\mathsf{z} > 0$
RTS. $\{\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}\}\ \mathsf{null}\ \{\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}\}$
From Lemma 11 we know that:
$\mathsf{SF}(\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f})$ (I)
From (I) and the Hoare judgement(Val.) we know:
$\{\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}\}\ \mathsf{null}\ \{\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}\}$ as required.

RTS. $\mathsf{H}', \Pi', \sigma_0', \tau_0 \models (\mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f})$
From (G2) we have:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z})$ (II)
From (II) and the definition of the $\models$ judgement we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z})$ (III)
From (III) and property P4b we know:
$\Pi(\tau_0)(\sigma_0(\mathsf{x}).\mathsf{f}) \geq \mathsf{z}$ (IV)
Since $\Pi' = \Pi$, from (IV) we have:
$\Pi'(\tau_0)(\sigma_0(\mathsf{x}).\mathsf{f}) \geq \mathsf{z}$ (V)
Since σ_0' i exactly the same as σ_0 except for the value of y, we can deduce:
$\sigma_0'(\mathsf{x}) = \sigma_0(\mathsf{x})$ (VI)
From (V) and (VI) we have:
$\Pi'(\tau_0)(\sigma_0'(\mathsf{x}).\mathsf{f}) \geq \mathsf{z}$ (VII)
From (VII) and property P4b we have:
$\mathsf{H}, \Pi'(\tau_0), \sigma_0' \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z})$ (VIII)
From the definition of σ_0' we know:
$\sigma_0'(\mathsf{y}) = \mathsf{H}(\sigma_0(\mathsf{x}).\mathsf{f})$ (IX)
From (VI) and (IX) we have:
$\sigma_0'(\mathsf{y}) = \mathsf{H}(\sigma_0'(\mathsf{x}).\mathsf{f})$ (X)
Since $\mathsf{H}' = \mathsf{H}$, from (X) we have:
$\sigma_0'(\mathsf{y}) = \mathsf{H}'(\sigma_0'(\mathsf{x}).\mathsf{f})$ (XI)
From (XI) and property P4a we have:
$\mathsf{H}', \epsilon, \sigma_0' \models \mathsf{y} = \mathsf{x.f}$ (XII)
From (VIII), (XII) and property P5 we have:
$\mathsf{H}, \Pi'(\tau_0) \uplus \epsilon, \sigma_0' \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}$ (XIII)
From the definition of $\uplus$ on permission masks we know:
$\Pi'(\tau_0) \uplus \epsilon = \Pi'(\tau_0)$ (XIV)
From (XIII) and (XIV) we have:
$\mathsf{H}, \Pi'(\tau_0), \sigma_0' \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}$ (XV)
From (XV) and the definition of $\models$ we have:
$\mathsf{H}, \Pi', \sigma_0', \tau_0 \models \mathsf{acc}(\mathsf{x.f}, \mathsf{z}) * \mathsf{y} = \mathsf{x.f}$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\bar{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\bar{\iota}\}, \tau)$
$\tau \neq \tau_0$
RTS. $\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\bar{\mathsf{u}})$

Since $\mathsf{H}' = \mathsf{H}$, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\bar{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
Since $\Pi' = \Pi$, and from G5 we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A_i'}$ (1)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (2) where
$\mathsf{A_i'} = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad$ $\mathsf{A_i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_g$.
Since $\mathsf{H}' = \mathsf{H}$, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *A_i'$ (5)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (6) where
$A_i' = A_i[x_i/\mathsf{this}]$ $A_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (7)

Finally, from (5), (G2)-(G7) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [x_i \mapsto \iota_i], \tau_g \models *A_i'$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [x_i \mapsto \iota_i], \tau_g \models *A_i'$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$A_i' = A_i[x_i/\mathsf{this}]$ $A_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since $\mathsf{H}' = \mathsf{H}$, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case NewC.

$P_0 \equiv \mathsf{True}$ $e_0 \equiv x := \mathsf{new}\ C$ $Q_0 \equiv *\mathsf{acc}(x.f_i, 1) \ * x.f_i = \mathsf{null}$
$\iota \notin \mathsf{H}$ $\mathsf{H}' = \mathsf{H}[\ \iota \mapsto (C, \{f_1 : \mathsf{null}, ..., f_r : \mathsf{null}\},\ \tau_0)$ $\sigma' = \sigma[x \mapsto \iota]$ $e' = \iota$
$\Pi' = \Pi[(\tau_0)(\iota, f_i) \mapsto 1]$

RTS. 2. $\exists P_0'.[\ \{P_0'\}\ \iota\ \{Q_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_0', \tau_0 \models P_0'\ ]$

Take $P_0' = Q_0 = *\mathsf{acc}(x.f_i, 1) \ * x.f_i = \mathsf{null}$
RTS. $\{P_0'\}\ \iota\ \{Q_0\}$.
From Lemma 11 we know that P_0' is self-framing.
Hence, from the hoare logic judgement (Val.) we know:
$\{P_0'\}\ \iota\ \{Q_0\}$. as required.

RTS. $\mathsf{H}', \Pi', \sigma_0', \tau_0 \models P_0'$
Given the definition of Π', we know:
$\Pi'(\tau_0)(\iota, f_1) = 1$. (1-a)
...
$\Pi'(\tau_0)(\iota, f_r) = 1$. (r-a)

Let us divide $\Pi'(\tau_0)$ into r+1 distinct permission masks $(\pi_1, ..., \pi_{r+1})$, such that:

for $i \in \{1...r\}$ we have:
$\mathsf{Dom}(\pi_i) = (\iota.f_i)$ and

$$\pi_i(\iota'.f) = \begin{cases} \Pi'(\tau_0)(\iota.f_i) & \text{if } \iota'.f = \iota.f_i \\ \bot & \text{otherwise} \end{cases}$$

and for π_{r+1} we have:
$\mathsf{Dom}(\pi_{r+1}) = \mathsf{Dom}(\Pi'(\tau_0)) \setminus \{\mathsf{Dom}(\pi_1) \cup ... \cup \mathsf{Dom}(\pi_r)\}$ and

$$\pi_{r+1}(\iota'.f) = \begin{cases} \Pi'(\tau_0)(\iota'.f) & \text{if } \iota'.f \in Dom(\Pi'(\tau_0)) \\ & \wedge\ \iota'.f \neq \iota.f_1 \wedge .... \wedge \iota'.f \neq \iota.f_r \\ \bot & \text{otherwise} \end{cases}$$

Then from the definition of $\uplus$ on permission masks we have:
$\Pi'(\tau_0) = \pi_1 \uplus ... \uplus \pi_{r+1}$ (II)
From the definition of σ_0', we know:
$\sigma_0'(x) = \iota$ (III)
From (1-a)...(r-a) and the definitions of $\pi_1 ... \pi_r$ we can deduce:
$\pi_1(\iota, f_1) = 1$ (1-b)
...
$\pi_r(\iota, f_r) = 1$ (r-b)
From (III), (1-b),...,(r-b) and property P4b we can deduce:
$\mathsf{H}', \pi_1, \sigma_0' \models \mathsf{acc}(x.f_1, 1)$ (1-c)
...
$\mathsf{H}', \pi_r, \sigma_0' \models \mathsf{acc}(x.f_r, 1)$ (r-c)
From (1-c),...,(r-c) and property P5 we can deduce:
$\mathsf{H}', \pi_1 \uplus ... \uplus \pi_r, \sigma_0' \models \mathsf{acc}(x.f_1, 1) * ... * \mathsf{acc}(x.f_r, 1)$, that is:
$\mathsf{H}', \pi_1 \uplus ... \uplus \pi_r, \sigma_0' \models *(\mathsf{acc}(x.f_i, 1))$ (IV)
From definition of H', we know:
$\mathsf{H}'(\iota) = (C, \{f_1 : \mathsf{null}, ..., f_r : \mathsf{null}\},\ \tau_0)$. (V)
Therefore, from (III), (V) and property P4a we can deduce:
$\mathsf{H}', \epsilon, \sigma_0' \models x.f_1 = \mathsf{null}$ (1-d)
...
$\mathsf{H}', \epsilon, \sigma_0' \models x.f_r = \mathsf{null}$ (r-d)
From the definition of $\uplus$ for permission masks we can deduce:
$\epsilon \uplus \epsilon = \epsilon$ (VI)
From (1-d), ..., (r-d), (VI) and property P5 we can deduce:
$\mathsf{H}', \epsilon, \sigma_0' \models x.f_1 = \mathsf{null} * ... * x.f_r = \mathsf{null}$, that is:
$\mathsf{H}', \epsilon, \sigma_0' \models *(x.f_i = \mathsf{null})$ (VII)
From (VII) and Lemma 13 we can deduce:
$\mathsf{H}', \epsilon \uplus \pi_{r+1}, \sigma_0' \models *(x.f_i = \mathsf{null})$ (VIII)
From the definition of $\uplus$ for permission masks we can deduce:
$\pi_{r+1} = \epsilon \uplus \pi_{r+1}$ (IX)
From (VIII) and (IX) we can deduce:
$\mathsf{H}', \pi_{r+1}, \sigma_0' \models *(x.f_i = \mathsf{null})$ (X)
From (IV), (X) and property P5 we can deduce:
$\mathsf{H}', \pi_1 \uplus ... \uplus \pi_r \uplus \pi_{r+1}, \sigma_0' \models *(\mathsf{acc}(x.f_i, 1) * (x.f_i = \mathsf{null})$ (XI)
From (II) and (XI) we have:
$\mathsf{H}', \Pi'(\tau_0), \sigma_0' \models *(\mathsf{acc}(x.f_i, 1) * (x.f_i = \mathsf{null})$ (XII)
From (XII) and the definition of the $\models$ judgement we have:
$\mathsf{H}', \Pi', \sigma_0', \tau_0 \models *(\mathsf{acc}(x.f_i, 1) * (x.f_i = \mathsf{null})$, that is:
$\mathsf{H}', \Pi', \sigma_0', \tau_0 \models P_0'$ as required.

RTS. 3a.
RTS.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{c : C, m : m, \mathsf{args} : \iota.\bar{\iota}\}, \tau) \ \wedge \ \tau \neq \tau_0 \implies$
$\exists P, \exists j \in \{1...n\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models P$
$\wedge\ \{P\}\ e_j\ \{\mathsf{Post}(m)\}] \ \wedge \ \sigma_j(\mathsf{this}) = \iota \ \wedge \ \sigma_j(\bar{u}) = \bar{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{c : C, m : m, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$.
$\tau \neq \tau_0$
RTS. $\exists P, \exists j \in \{1...n\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models P$
$\wedge\ \{P\}\ e_j\ \{\mathsf{Post}(m)\}] \ \wedge \ \sigma_j(\mathsf{this}) = \iota' \ \wedge \ \sigma_j(\bar{u}) = \bar{\iota}\ ]$
where $\mathsf{Post}(m) = A(\bar{u})$

Since H' is exactly the same as H except for the new address ι, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{c : C, m : m, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.f[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.f) \ \leq 1]$
$\wedge\ \forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.g) \ \leq 1]$

Π' is exactly the same as Π except for *addition* of new permissions to the fields of new location (ι). In other words, for those locations already in H, Π' is the same as Π. Therefore, we can deduce:
$\forall \iota'[\iota' \in \mathsf{H} \implies \sum_{\tau_i \in \mathsf{DOM}(\Pi')} \Pi'(\tau_i)(\iota'.f) \leq 1$ (I)
$\forall \kappa'[\sum_{\tau_i \in \mathsf{DOM}(\Pi')} \Pi'(\tau_i)(\kappa'.g) \leq 1$ (II)
On the other hand, for the new location ι, we add a new entry for each field f of that object under the current thread (τ_0). Since ι did not exist in the heap before this point, no other thread could have had any permissions to any of its fields. Furthermore, since we gave τ_0 full permission (denoted by 1) to all fields of ι, we can deduce:
$\sum_{\tau_i \in \mathsf{DOM}(\Pi')} \Pi'(\tau_i)(\iota.f) \leq 1$ (III)
Finally, from (I), (II) and (III) we can deduce:
$\forall \iota.f[\sum_{\tau_i \in \mathsf{DOM}(\Pi')} \Pi'(\tau_i)(\iota.f) \leq 1]$
$\wedge\ \forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.g) \ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [x_i \mapsto \iota_i], \tau_g \models *A_i'$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$A_i' = A_i[x_i/\mathsf{this}]$ $A_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *A_i'$ (1)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (2) where
$A_i' = A_i[x_i/\mathsf{this}]$ $A_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_g$.
From the operational semantic of New we know that $\mathsf{H}'(\iota) \downarrow_3 \neq \tau_g$ and hence we can deduce:
$\iota' \neq \iota$
Since H' is exactly the same as H, except for the value of ι, and since $\iota' \neq \iota$, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *A_i'$ (5)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (6) where
$A_i' = A_i[x_i/\mathsf{this}]$ $A_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (7)

Finally, from (5), (G2)-(G7) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since H' is exactly the same as H except for the value of ι, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case Meth.

$\mathsf{P_0} \equiv \mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$ $\quad \mathsf{e_0} \equiv \mathsf{x.m}(\overline{\mathsf{y}})$ $\quad \mathsf{Q_0} \equiv \mathsf{Post(m)}[\overline{\mathsf{y}}/\overline{\mathsf{u}}]$
$\mathsf{H}' = \mathsf{H}$ $\quad \Pi' = \Pi$ $\quad \sigma'_0 = \sigma_0$ $\quad \mathsf{e}' = \mathsf{fork\ tk := x.m}(\overline{\mathsf{y}})\mathsf{;join\ tk}$ $\quad \mathsf{tk} \notin \sigma_0$

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'\ \{\mathsf{Q_0}\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$

Take $\mathsf{P}'_0 = \mathsf{P_0} = \mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$
RTS. $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$ From (G3) we have:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$ (I)
Since $\mathsf{H}' = \mathsf{H}, \sigma'_0 = \sigma_0$ and $\Pi' = \Pi$, from (I) we have:
$\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$ as required.

RTS. $\{\mathsf{P}'_0\}\ \mathsf{e}'\ \{\mathsf{Q_0}\}$, that is to show:
$\{\mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}$ $\mathsf{fork\ tk := x.m}(\overline{\mathsf{y}})\mathsf{;join\ tk}$
$\{\mathsf{Post(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}$. (II)

According to the Hoare logic Seq. judgement, (II) holds if and only if:
$\exists \mathsf{D}.[\{\mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}\ \mathsf{fork\ tk := x.m}(\overline{\mathsf{y}})\ \{\mathsf{D}\}$ (III)
$\wedge\ \{\mathsf{D}\}\ \mathsf{join\ tk}\ \{\mathsf{Post(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}$ (IV)

Take $\mathsf{D} \equiv \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$ where $\mathsf{Class(x)} = \mathsf{C}$.
RTS.
$\{\mathsf{Pre(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}\ \mathsf{fork\ tk := x.m}(\overline{\mathsf{y}})\ \{\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})\}$ (V)
$\wedge\ \{\mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})\}\ \mathsf{join\ tk}\ \{\mathsf{Post(m)}[\mathsf{x/this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]\}$ (VI)

By Hoare logic fork judgement (V) holds.
By Hoare logic join judgement (VI) holds.
Hence, (I) holds as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \ \wedge\ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
$\tau_\mathsf{j} \neq \tau_0$
RTS. $\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\overline{\mathsf{u}})$

Since $\mathsf{H}' = \mathsf{H}$, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$

Since $\Pi' = \Pi$, from G5 we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (1)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (2) where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_g$.
Since $\mathsf{H}' = \mathsf{H}$ is exactly the same as H, except for the value of $\iota.\mathsf{f}$, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (5)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (6) where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (7)

Finally, from (5), (G2)-(G7) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i}/\mathsf{this}]$ $\quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since $\mathsf{H}' = \mathsf{H}$, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case If.

$\mathsf{e_0} \equiv \mathsf{if(B)\ then\ e_1\ else\ e_2}$
$\mathsf{H}' = \mathsf{H}$ $\quad \sigma'_0 = \sigma_0$ $\quad \Pi' = \Pi$
$\{\mathsf{P_0} \wedge \mathsf{B}\}\ \mathsf{C_1}\ \{\mathsf{Q_0}\}$ (a)
$\{\mathsf{P_0} \wedge \neg \mathsf{B}\}\ \mathsf{C_2}\ \{\mathsf{Q_0}\}$ (b)

Case a. $\mathsf{B} \equiv \mathsf{True}$ (c)
$\mathsf{e}' = \mathsf{e_1}$ (d)

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e_1}\ \{\mathsf{Q_0}\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
Take $\mathsf{P}'_0 = \mathsf{P_0}$

RTS. $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P_0}$.
From (G3) we have:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{P_0}$. (I)
Since$\mathsf{H}' = \mathsf{H}, \sigma' = \sigma$ and $\Pi' = \Pi$, from (I) we have:
$\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P_0}$ as required.

RTS. $\{\mathsf{P_0}\}\ \mathsf{e_1}\ \{\mathsf{Q_0}\}$
We know (a) and (c), we have:
$\{\mathsf{P_0} \wedge \mathsf{True}\}\ \mathsf{e_1}\ \{\mathsf{Q_0}\}$, that is:
$\{\mathsf{P_0}\}\ \mathsf{e_1}\ \{\mathsf{Q_0}\}$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \ \wedge\ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
$\tau \neq \tau_0$
RTS. $\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\overline{\mathsf{u}})$

Since $\mathsf{H}' = \mathsf{H}$, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c : C}, \mathsf{m : m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
Since $\Pi' = \Pi$, from G5 we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$

for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A}_i[x_i/\mathsf{this}]$ $\quad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (1)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (2) where
$\mathsf{A}'_i = \mathsf{A}_i[x_i/\mathsf{this}]$ $\quad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_g$.
Since H' is exactly the same as H, except for the value of ι.f, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (5)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ (6) where
$\mathsf{A}'_i = \mathsf{A}_i[x_i/\mathsf{this}]$ $\quad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (7)

Finally, from (5), (G2)-(G7) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [x_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [x_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A}_i[x_i/\mathsf{this}]$ $\quad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since $\mathsf{H}' = \mathsf{H}$, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case Acquire

$\mathsf{P}_0 \equiv \mathsf{True}$ $\quad e_0 \equiv \mathsf{acquire}\ x$ $\quad \mathsf{Q}_0 \equiv \mathsf{A}[x/\mathsf{this}]$ where $\mathsf{A} = \mathsf{Invariant}(\sigma_0(x).\mathsf{class})$
$\sigma_0(x) = \iota$ $\quad \mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau)\ ]$ $\quad \sigma'_0 = \sigma_0$ $\quad e' = \mathsf{null}$
$\Pi' = \Pi[\tau+ = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A}), \quad \tau_g- = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A})\ ]$

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ e'\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
Take $\mathsf{P}'_0 = \mathsf{Q}_0 = \mathsf{A}[x/\mathsf{this}]$

RTS. $\{\mathsf{A}[x/\mathsf{this}]\}\ \mathsf{null}\ \{\mathsf{A}[x/\mathsf{this}]\}$
From (G1) we know and the definition of $\mathsf{Ver}(\mathsf{Prog})$, we have:
$\mathsf{SF}(\mathsf{A})$ (I)
From Lemma 8 and (I) we have:
$\mathsf{SF}(\mathsf{A}[x/\mathsf{this}])$ (II)
From (II) and the Hoare logic judgement (Val.) we have:
$\{\mathsf{A}[x/\mathsf{this}]\}\ \mathsf{null}\ \{\mathsf{A}[x/\mathsf{this}]\}$ as required.

RTS. $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{A}[x/\mathsf{this}]$
First we show that: $\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A})}{\equiv} \mathsf{H}'$
RTS. $\forall(\iota'.f).[\mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A})(\iota'.f) = n \ \wedge \ n > 0 \implies \mathsf{H}(\iota'.f) = \mathsf{H}'(\iota'.f)]$
$\wedge\ \forall(\kappa.g)[\pi(\mathsf{H}, \sigma, \mathsf{A})(\kappa, g) = n \ \wedge \ n > 0 \implies \mathsf{H}(\kappa.g) = \mathsf{H}'(\kappa.g)]$

Take an arbitrary $(\iota'.f)$ such that $\mathcal{P}(\mathsf{H}, \sigma, \mathsf{A})(\iota'.f) = n \ \wedge \ n > 0$ (III)
RTS. $\mathsf{H}(\iota'.f) = \mathsf{H}'(\iota'.f)$
There are two cases:

Case 1. $\iota' \neq \iota$
Since H' is exactly the same as H except for the value of ι, from the case assumption we know:
$\mathsf{H}'(\iota') = \mathsf{H}(\iota')$ and hence:
$\mathsf{H}'(\iota'.f) = \mathsf{H}(\iota'.f)$ (IV).

Case 2. $\iota' = \iota$
From the definition of H', we know:
$\mathsf{H}'(\iota) \downarrow_2 = \mathsf{H}(\iota) \downarrow_2$ and hence:
$\mathsf{H}'(\iota'.f) = \mathsf{H}(\iota'.f)$ (V).

RTS. $\forall(\kappa.g)[\pi(\mathsf{H}, \sigma, \mathsf{A})(\kappa, g) = n \ \wedge \ n > 0 \implies \mathsf{H}(\kappa.g) = \mathsf{H}'(\kappa.g)]$
Take an arbitrary $\kappa.g$ such that $\pi(\mathsf{H}, \sigma, \mathsf{A})(\kappa, g) = n \ \wedge \ n > 0$. (VI)
RTS. $\mathsf{H}(\kappa.g) = \mathsf{H}'(\kappa.g)$
Since H' is exactly the same as H except for the value of ι', we can deuce:
$\mathsf{H}(\kappa.g) = \mathsf{H}'(\kappa.g)$ (VII)

From (III)-(VII) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A})}{\equiv} \mathsf{H}'$ (VIII)

On the other hand from (G1) and the definition of $\mathsf{Ver}(\mathsf{Prog})$, we can deduce:
$\mathsf{SF}(\mathsf{A})$ (IX)
From G5 we know that:
$\mathsf{H}, \Pi, \sigma_g, \tau_g \models \mathsf{A}$ (X)
where $\sigma_g = \mathsf{this} \mapsto \iota$ (XI)
From (VIII), (IX), (X) and the definition of self-framing assertions we have:
$\mathsf{H}', \Pi, \sigma_g, \tau_g \models \mathsf{A}$ (XII)
From (XII) and the definition of the $\models$ judgement we have:
$\mathsf{H}', \Pi(\tau_g), \sigma_g \models \mathsf{A}$ (XIII)
From (XI) we know that:
$\mathsf{fv}(\sigma_g) = \{\mathsf{this}\}$ (XIV)
We also have: $\sigma_0(x) = \sigma_g(\mathsf{this}) = \iota$ (XV)
From (XIII), (XIV), (XV) and property P1 of the $\models$ judgement we can deduce:
$\mathsf{H}', \Pi(\tau_g), \sigma_0 \models \mathsf{A}[x/\mathsf{this}]$ (XVI)
From (XVI) and definition of the $\models$ judgement we have:
$\mathsf{H}', \Pi, \sigma_0, \tau_g \models \mathsf{A}[x/\mathsf{this}]$ (XVII)
Given (XVII), definition of Π' and Lemma 5 we have:
$\mathsf{H}', \Pi', \sigma_0, \tau_0 \models \mathsf{A}[x/\mathsf{this}]$ (XVIII)
Finally, since $\sigma'_0 = \sigma_0$, from (XVIII) we have:
$\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{A}[x/\mathsf{this}]$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{c : \mathsf{C}, m : m, \mathsf{args} : \iota.\bar{\iota}\}, \tau) \ \wedge \ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists j \in \{1...n\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ e_j\ \{\mathsf{Post}(m)\}] \ \wedge \ \sigma_j(\mathsf{this}) = \iota \ \wedge \ \sigma_j(\bar{u}) = \bar{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{c : \mathsf{C}, m : m, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$.
$\tau \neq \tau_0$
RTS. $\exists \mathsf{P}, \exists j \in \{1...n\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ e_j\ \{\mathsf{Post}(m)\}] \ \wedge \ \sigma_j(\mathsf{this}) = \iota' \ \wedge \ \sigma_j(\bar{u}) = \bar{\iota}\ ]$
where $\mathsf{Post}(m) = \mathsf{A}(\bar{u})$

Since H' is exactly the same as H except for the new address ι, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{c : \mathsf{C}, m : m, \mathsf{args} : \iota'.\bar{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.f[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.f) \ \leq 1]$
$\wedge\ \forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.g) \ \leq 1]$

Π' is exactly the same as Π except for the permissions of current thread τ_0 and the ghost thread τ_g. In Π' we have stripped τ_g from the permissions of object x's monitor and granted these permissions to τ_0. In other words, the sum of permissions have not changed from Π and we have just changed the owner of these permissions. Hence we can deduce:
$\forall \iota.f[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.f) = \sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.f)]$ (1)
$\forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.g) = \sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.g)]$ (2)
From G5 we have:
$\forall \iota.f[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.f) \ \leq 1]$ (3)
$\wedge\ \forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.g) \ \leq 1]$ (4)
From (1), (2), (3) and (4) we can deduce:
$\forall \iota.f[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.f) \ \leq 1]$
$\wedge\ \forall \kappa.g[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.g) \ \leq 1]$
as required.

RTS. 3c.
$\forall\ \iota.[\ \mathsf{H}'(\iota) \downarrow_3 = \tau_g \implies \mathsf{H}', \Pi', [\mathsf{this} \mapsto \iota], \tau_g \models \mathsf{A}\ ]$
where $\mathsf{A} = \mathsf{invariant}(\iota.\mathsf{class})$
Take an arbitrary ι' such that: $\mathsf{H}'(\iota') \downarrow_3 = \tau_g$. (1)
RTS. $\mathsf{H}', \Pi', [\mathsf{this} \mapsto \iota'], \tau_g \models \mathsf{A}$

From definition of H', we know that $\mathsf{H}'(\sigma_0(x)) \downarrow_3 = \tau_0$. Hence we can deduce:
$\iota' \neq \sigma_0(x)$ (2)
Since H' is exactly the same as H, except for the monitor of $\sigma_0(x)$, from (2) we know that:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$. (3)
From (G5), (3) and the definition of WF we know:
$\mathsf{H}, \Pi, [\mathsf{this} \mapsto \iota'], \tau_g \models \mathsf{A}$ (4)
Also from (G5) and the definition of WF we know:
$\mathsf{H}, \Pi, [x_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (5)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (6) where
$\mathsf{A}'_i = \mathsf{A}_i[x_i/\mathsf{this}]$ $\quad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (7)

Since from operational semantics of acquire we know $\mathsf{H}(\sigma_0(\mathsf{x})) \downarrow_3 = \tau_g$ and $\sigma_0(\mathsf{x}) = \iota$ from (6) we can deduce:
$\iota \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (8)
From (8) we can rewrite (5) as: $\mathsf{H}, \Pi, [\mathsf{x_j} \mapsto \iota_j, \mathsf{x}_\iota \mapsto \iota], \tau_g \models (*\mathsf{A}'_j) * \mathsf{A}'_\iota$ (9)
for $\iota_j \in \{\iota'' | \iota'' \in \mathsf{Dom}(\mathsf{H}) \wedge \iota'' \neq \iota \wedge \mathsf{H}(\iota'') \downarrow_3 = \tau_g\}$ (10) where
$\mathsf{A}'_j = \mathsf{A_j}[\mathsf{x_j/this}] \quad \mathsf{A_j} = \mathsf{Invariant}(\iota_j.\mathsf{class})$
$\mathsf{A}'_\iota = \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}] \quad \mathsf{A}_\iota = \mathsf{invariant}(\iota.\mathsf{class})$ (11)
From (9) and the definition of $\models$ we have:
$\mathsf{H}, \Pi(\tau_g), [\mathsf{x_j} \mapsto \iota_j, \mathsf{x}_\iota \mapsto \iota] \models (*\mathsf{A}'_j) * \mathsf{A}'_\iota$ (12)
From (12) and property P5 we have:
$\mathsf{H}, \pi_1, [\mathsf{x_j} \mapsto \iota_j, \mathsf{x}_\iota \mapsto \iota] \models (*\mathsf{A}'_j)$ (13)
$\mathsf{H}, \pi_2, [\mathsf{x_j} \mapsto \iota_j, \mathsf{x}_\iota \mapsto \iota] \models \mathsf{A}'_\iota$ (14)
$\Pi(\tau_g) = \pi_1 \uplus \pi_2$ (15)
From (15) and the definition of $\uplus$ on permission masks we know:
$\forall (\iota'.\mathsf{f}) \in \mathsf{Dom}(\pi_1)[\pi_1(\iota'.\mathsf{f}) = \Pi(\tau_g)(\iota'.\mathsf{f}) - \pi_2(\iota'.\mathsf{f})]$ (16)
On the other hand, from the definition of Π', we know that:
$\forall (\iota'.\mathsf{f})[\Pi'(\tau_g)(\iota'.\mathsf{f}) = \Pi(\tau_g)(\iota'.\mathsf{f}) - \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota)]$ (17)
From property P1b we know that:
$\mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota) = \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota, \mathsf{x_j} \mapsto \iota_j], \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}])$ (18)
From (11), (17) and (18) we have:
$\forall (\iota'.\mathsf{f})[\Pi'(\tau_g)(\iota'.\mathsf{f}) = \Pi(\tau_g)(\iota'.\mathsf{f}) - \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota, \mathsf{x_j} \mapsto \iota_j], \mathsf{A}'_\iota)(\iota'.\mathsf{f})]$ (19)
From (14) and property P2a we have:
$\forall (\iota'\mathsf{f})[\pi_2(\iota'.\mathsf{f}) \geq \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota, \mathsf{x_j} \mapsto \iota_j], \mathsf{A}'_\iota)]$ (20)
From (20) we have:
$\forall (\iota'.\mathsf{f})$
$[\Pi(\tau_g)(\iota'.\mathsf{f}) - \pi_2(\iota'.\mathsf{f}) \leq \Pi(\tau_g)(\iota'.\mathsf{f}) - \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota, \mathsf{x_j} \mapsto \iota_j], \mathsf{A}'_\iota)(\iota'.\mathsf{f})]$ (21)
From (19) and (21) we have:
$\forall (\iota'.\mathsf{f})[\Pi'(\tau_g)(\iota'.\mathsf{f}) \geq \Pi(\tau_g)(\iota'.\mathsf{f}) - \pi_2(\iota'.\mathsf{f})]$ (22)
From (16) and (22) we have:
$\forall (\iota'.\mathsf{f}) \in \mathsf{Dom}(\pi_1)[\pi_1(\iota'.\mathsf{f}) \leq \Pi'(\tau_g)(\iota'.\mathsf{f})]$ (23)
From (23) and the definition of $\subseteq$ on permission masks we have:
$\pi_1 \subseteq \Pi'(\tau_g)$ (24)
From (13), (24) and Lemma 13b we have:
$\mathsf{H}, \Pi'(\tau_g), [\mathsf{x_j} \mapsto \iota_j, \mathsf{x}_\iota \mapsto \iota] \models (*\mathsf{A}'_j)$ (24)
for $\iota_j \in \{\iota'' | \iota'' \in \mathsf{Dom}(\mathsf{H}) \wedge \iota'' \neq \iota \wedge \mathsf{H}(\iota'') \downarrow_3 = \tau_g\}$ (25) where
$\mathsf{A}'_j = \mathsf{A_j}[\mathsf{x_j/this}] \quad \mathsf{A_j} = \mathsf{Invariant}(\iota_j.\mathsf{class})$ (26)
Since H' is exactly the same as H except for the value of (ι) and since $\mathsf{H}'(\iota) \downarrow_3 \neq \tau_g$, we can rewrite (25)-(26) as:
$\mathsf{H}, \Pi'(\tau_g), [\mathsf{x_j} \mapsto \iota_j] \models (*\mathsf{A}'_j)$ (27)
for $\iota_j \in \{\iota'' | \iota'' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota'') \downarrow_3 = \tau_g\}$ (28) where
$\mathsf{A}'_j = \mathsf{A_j}[\mathsf{x_j/this}] \quad \mathsf{A_j} = \mathsf{Invariant}(\iota_j.\mathsf{class})$ (29)

We now show : $\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))}{\equiv} \mathsf{H}'$, that is:
$\forall (\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\iota.\mathsf{f}) > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall (\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\kappa.\mathsf{g}) > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$

RTS. $\forall (\iota'.\mathsf{f})[\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\iota'.\mathsf{f}) > 0 \implies \mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})]$
Take an arbitrary $\iota'.\mathsf{f}$ such that $\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\iota'.\mathsf{f}) > 0$
RTS. $\mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})$
Since H' is exactly the same as H except for the monitor of ι, we can deduce:
$\mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})$ as required. (30)

RTS. $\forall (\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\kappa.\mathsf{g}) > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))(\kappa.\mathsf{g}) > 0$
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the monitor of ι, we can deduce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (31)

From (30) and (31) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, [\mathsf{x_j} \mapsto \iota_j], (*\mathsf{A}'_j))}{\equiv} \mathsf{H}'$ (32)
On the other hand, from (G1) and the definition of Ver(Prog) for each of the invariants we can deduce:
$\mathsf{SF}(\mathsf{A_j})$ (33)
From (33) and lemma 8 we have:
$\mathsf{SF}(\mathsf{A_j}[\mathsf{x_j/this}])$, that is:
$\mathsf{SF}(\mathsf{A}'_j)$ (34)
From (34) and property P8 we have:
$\mathsf{SF}(*\mathsf{A}'_j))$ (35)
From (35), (32), (27), (28), (29) and the definition of self-framing assertions we have:
$\mathsf{H}', \Pi'(\tau_g), [\mathsf{x_j} \mapsto \iota_j] \models (*\mathsf{A}'_j)$
for $\iota_j \in \{\iota'' | \iota'' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota'') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_j = \mathsf{A_j}[\mathsf{x_j/this}] \quad \mathsf{A_j} = \mathsf{Invariant}(\iota_j.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since H' is exactly the same as H except for the value of ι', from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case Release

$\mathsf{P_0} \equiv \mathsf{A}[\mathsf{x/this}] \quad \mathsf{e_0} \equiv \mathsf{release\ x} \quad \mathsf{Q_0} \equiv \mathsf{True} \quad \sigma_0(\mathsf{x}) = \iota$
$\mathsf{H}' = \mathsf{H}[\iota \mapsto (\mathsf{H}(\iota) \downarrow_1, \mathsf{H}(\iota) \downarrow_2, \tau_g)] \quad \sigma' = \sigma \quad \mathsf{e}' = \mathsf{null}$
$\Pi' = \Pi[\tau - = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A}),\ \tau_g + = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A})\ ]$

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'\ \{\mathsf{Q_0}\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
Take $\mathsf{P}'_0 = \mathsf{Q_0} = \mathsf{True}$, it is clear that $\{\mathsf{P}'_0\}\ \mathsf{null}\ \{\mathsf{Q_0}\}$ holds.
Furthermore, $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{True}$ holds trivially.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1 \ldots \mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ such that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
$\tau \neq \tau_0$
RTS. $\exists \mathsf{P}, \exists \mathsf{j} \in \{1 \ldots \mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota'\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\overline{\mathsf{u}})$

Since H' is exactly the same as H except for the new address ι, we also have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota'.\overline{\iota}\}, \tau)$.
The rest of the proof is identical to the FldAss case.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$

Π' is exactly the same as Π except for the permissions of current thread τ_0 and the ghost thread τ_g. In Π' we have stripped τ_0 from the permissions of object o's monitor and granted these permissions to τ_g. In other words, the sum of permissions have not changed from Π and we have just changed the owner of these permissions. Hence we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f}) = \sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})]$ (1)
$\forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{gf}) = \sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})]$ (2)
From G5 we have:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f}) \ \leq 1]$ (3)
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ (4)
From (1), (2), (3) and (4) we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$
as required.

RTS. 3c.
$\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_j], \tau_g \models *\mathsf{A}'_j$
for $\iota_j \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_j = \mathsf{A_j}[\mathsf{x_j/this}] \quad \mathsf{A_j} = \mathsf{Invariant}(\iota_j.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x_i} \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$ (1)
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\}$ (2) where
$\mathsf{A}'_i = \mathsf{A_i}[\mathsf{x_i/this}] \quad \mathsf{A_i} = \mathsf{Invariant}(\iota_i.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_g$.
There are now two cases:
Case 1. $\iota' \neq \iota$
Since H' is exactly the same as H, except for the monitor of ι, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_g$ (4)
Case 2. $\iota' = \iota$
From the operational semantics of Release we know:
$\mathsf{H}(\iota) \downarrow_3 \neq \tau_g \ \wedge\ \mathsf{H}'(\iota) \downarrow_3 = \tau_g$ (5)
From (4) and (5) we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_g\} \cup \{\iota\}$ (6)

Since $\mathsf{P_0} \equiv \mathsf{A}_\iota[\mathsf{x/this}]$, from G3 we know that:

$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}]$ (7)
where $\mathsf{A}_\iota = \mathsf{invariant}(\iota.\mathsf{class})$
Given the definition of Π', (7) and lemma 5 we have:
$\mathsf{H}, \Pi', \sigma_0, \tau_\mathsf{g} \models \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}]$ (8)
Since A_ι is the invariant of a class, we know that its only free variable is this, that is:
$\mathsf{FV}(\mathsf{A}_\iota) = \{\mathsf{this}\}$, that is:
$\mathsf{FV}(\mathsf{A}_\iota[\mathsf{x}/\mathsf{this}] = \{\mathsf{x}\})$ (9)
From (8) and (9) and since $\sigma_0(\mathsf{x}) = \iota$ we have:
$\mathsf{H}, \Pi', [\mathsf{this} \mapsto \iota], \tau_\mathsf{g} \models \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}][\mathsf{this}/\mathsf{x}]$, that is:
$\mathsf{H}, \Pi', [\mathsf{this} \mapsto \iota], \tau_\mathsf{g} \models \mathsf{A}_\iota$ from the definition of $\models$, that is:
$\mathsf{H}, \Pi'(\tau_\mathsf{g}), [\mathsf{this} \mapsto \iota] \models \mathsf{A}_\iota$ (10)
From (10) and property P2 we have:
$\mathsf{H}, \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota), [\mathsf{this} \mapsto \iota] \models \mathsf{A}_\iota$ (11)
From the definition of Π', we know:
$\Pi'(\tau_\mathsf{g}) = \Pi(\tau_\mathsf{g}) \uplus \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}])$ (12)
From (9) and (12) and since $\sigma_0(\mathsf{x}) = \iota$, we know:
$\Pi'(\tau_\mathsf{g}) = \Pi(\tau_\mathsf{g}) \uplus \mathcal{P}(\mathsf{H}, [\mathsf{x} \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}])$ (13)
From property P1b we have:
$\mathcal{P}(\mathsf{H}, [\mathsf{x} \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}]) = \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}][\mathsf{this}/\mathsf{x}])$, that is:
$\mathcal{P}(\mathsf{H}, [\mathsf{x} \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}/\mathsf{this}]) = \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota)$ (14)
From (13) and (14) we have:
$\Pi'(\tau_\mathsf{g}) = \Pi(\tau_\mathsf{g}) \uplus \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota)$ (15)
Take a free variable x_ι such that:
$\mathsf{x}_\iota \notin \{\overline{\mathsf{x}_\mathsf{i}}\}$
From (9), (11) and property P1a we have:
$\mathsf{H}, \mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota), [\mathsf{x}_\iota \mapsto \iota] \models \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ (16)
From property P2 we have:
$\mathcal{P}(\mathsf{H}, [\mathsf{this} \mapsto \iota], \mathsf{A}_\iota) = \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}])$ (17)
From (16) and (17) we have:
$\mathsf{H}, \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]), [\mathsf{x}_\iota \mapsto \iota] \models \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ (18)
From (17) and (15) we have:
$\Pi'(\tau_\mathsf{g}) = \Pi(\tau_\mathsf{g}) \uplus \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}])$ (19)
Take $\sigma \equiv [\mathsf{x}_\iota \mapsto \iota] \uplus [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}]$ (20)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$
From (18), (20) and lemma 9 we have:
$\mathsf{H}, \mathcal{P}(\mathsf{H}, [\mathsf{x}_\iota \mapsto \iota], \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]), \sigma \models \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ (21)
From (1)-(3) and the definition of $\models$ we have:
$\mathsf{H}, \Pi(\tau_\mathsf{g}), [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}] \models *\mathsf{A}'_\mathsf{i}$ (22)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (23) where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}]$ $\mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (24)
From (22)-(24), (20) and lemma 9 we have:
$\mathsf{H}, \Pi(\tau_\mathsf{g}), \sigma, \models *\mathsf{A}'_\mathsf{i}$ (25)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (26) where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}]$ $\mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (27)
From (19), (21), (25) and property P5 we have:
$\mathsf{H}, \Pi'(\tau_\mathsf{g}), \sigma, \models (*\mathsf{A}'_\mathsf{i}) * \mathsf{A}'_\iota$ (28)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (29) where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}]$ $\mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (30)
$\mathsf{A}'_\iota = \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ $\mathsf{A}_\iota = \mathsf{Invariant}(\iota.\mathsf{class})$ (31)
We can rewrite (28)-(31) as:
$\mathsf{H}, \Pi'(\tau_\mathsf{g}), \sigma, \models (*\mathsf{A}'_\mathsf{j})$ (32)
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\} \cup \{\iota\}$ (33) where
$\mathsf{A}'_\mathsf{j} = \mathsf{A}_\mathsf{j}[\mathsf{x}_\mathsf{j}/\mathsf{this}]$ $\mathsf{A}_\mathsf{j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$ (34)
$\mathsf{A}'_\iota = \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ $\mathsf{A}_\iota = \mathsf{Invariant}(\iota.\mathsf{class})$ (35)

We now show: $\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, *\mathsf{A}'_\mathsf{j})}{\equiv} \mathsf{H}'$, that is:
$\forall(\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\iota.\mathsf{f}) > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})]$
$\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\kappa.\mathsf{g}) > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$

RTS. $\forall(\iota'.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\iota'.\mathsf{f}) > 0 \implies \mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})]$
Take an arbitrary $\iota'.\mathsf{f}$ such that $\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\iota'.\mathsf{f}) > 0$
RTS. $\mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})$
Since H' is exactly the same as H except for the monitor of ι, we can deduce:
$\mathsf{H}(\iota'.\mathsf{f}) = \mathsf{H}'(\iota'.\mathsf{f})$ as required. (36)

RTS. $\forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\kappa.\mathsf{g}) > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})]$
Take an arbitrary $\kappa.\mathsf{g}$ such that $\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))(\kappa.\mathsf{g}) > 0$
RTS. $\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$
Since H' is exactly the same as H except for the monitor of ι, we can deduce:
$\mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H}'(\kappa.\mathsf{g})$ as required. (37)

From (36) and (37) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma, (*\mathsf{A}'_\mathsf{j}))}{\equiv} \mathsf{H}'$ (38)
On the other hand, from (G1) and the definition of $\mathsf{Ver}(\mathsf{Prog})$ for each of the invariants we can deduce:
$\mathsf{SF}(\mathsf{A}_\mathsf{j})$ (39)
From (39) and lemma 8 we have:
$\mathsf{SF}(\mathsf{A}_\mathsf{j}[\mathsf{x}_\mathsf{j}/\mathsf{this}])$, that is:
$\mathsf{SF}(\mathsf{A}'_\mathsf{j})$ (40)
From (40) and property P8 we have:
$\mathsf{SF}(*(\mathsf{A}'_\mathsf{j}))$ (41)
From (41), (38), (32) and the definition of self-framing assertions we have:
$\mathsf{H}', \Pi'(\tau_\mathsf{g}), \sigma \models (*\mathsf{A}'_\mathsf{j})$ (42)
From (42) and the definition of $\models$ we have:
$\mathsf{H}', \Pi', \sigma, \tau_\mathsf{g} \models (*\mathsf{A}'_\mathsf{j})$ (43)
From (43) and (33)-(35) we have:
$\mathsf{H}', \Pi', \sigma, \tau_\mathsf{g} \models (*\mathsf{A}'_\mathsf{j})$ (44)
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\} \cup \{\iota\}$ (45) where
$\mathsf{A}'_\mathsf{j} = \mathsf{A}_\mathsf{j}[\mathsf{x}_\mathsf{j}/\mathsf{this}]$ $\mathsf{A}_\mathsf{j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$ (46)
$\mathsf{A}'_\iota = \mathsf{A}_\iota[\mathsf{x}_\iota/\mathsf{this}]$ $\mathsf{A}_\iota = \mathsf{Invariant}(\iota.\mathsf{class})$ (47)
From (6) and (20), we can rewrite (44)-(47) as:
$\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models (*\mathsf{A}'_\mathsf{j})$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A}_\mathsf{j}[\mathsf{x}_\mathsf{j}/\mathsf{this}]$ $\mathsf{A}_\mathsf{j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$
Take arbitrary κ, κ' such that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau$ (I)
$\mathsf{H}'(\kappa') \downarrow_3 = \tau$ (II)
Since H' is exactly the same as H except for the value of ι, from (I) and (II) we have:
$\mathsf{H}(\kappa) \downarrow_3 = \tau$ (III)
$\mathsf{H}(\kappa') \downarrow_3 = \tau$ (IV)
From (G5), (III) and (IV) we have:
$\kappa = \kappa'$ as required.

Case Frame

$\mathsf{P}_0 \equiv \mathsf{A} * \mathsf{C}$ $\mathsf{Q}_0 \equiv \mathsf{B} * \mathsf{C}$
$\{\mathsf{A}\}\ \mathsf{e}_0\ \{\mathsf{B}\}$ (I)
$\mathsf{SF}(\mathsf{C})$ (II)

From (G2) and the definition of P_0 we know:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{A} * \mathsf{C}$ (1)
From (1) and the definition of $\models$ we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{A} * \mathsf{C}$ (2)
From (2) and property P5 we have:
$\mathsf{H}, \pi_1, \sigma_0 \models \mathsf{A}$ (3)
$\mathsf{H}, \pi_2, \sigma_0 \models \mathsf{C}$ (4)
$\Pi(\tau_0) = \pi_1 \uplus \pi_2$ (5)
From (3), (5) and lemma 13a we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{A}$ (6)

RTS. 2.$\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\exists \mathsf{A}'.[\{\mathsf{A}'\}\ \mathsf{e}'_0\ \{\mathsf{B}\}$ (III)
$\wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{A}']$ (IV)
Take $\mathsf{P}'_0 = \mathsf{A}' * \mathsf{C}$.

1.**RTS.** $\{\mathsf{A}' * \mathsf{C}\}\ \mathsf{e}'_0\ \{\mathsf{B} * \mathsf{C}\}$
From (II), (III) and the Hoare logic judgement (Frame) we can deduce:
$\{\mathsf{A}' * \mathsf{C}\}\ \mathsf{e}'_0\ \{\mathsf{B} * \mathsf{C}\}$

2. **RTS.** $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{A}' * \mathsf{C}$
From (G2) we have:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{A} * \mathsf{C}$ (V)
From (I), (V), G4 and Lemma 2 we deduce:
$\exists \Pi_1, \Pi_2.[\ \Pi = \Pi_1 \uplus \Pi_2$ (VI)
$\wedge \quad \mathsf{H}, \Pi_1, \sigma_0, \tau_0 \models \mathsf{A}$ (VII)
$\wedge \quad \mathsf{H}, \Pi_2, \sigma_0, \tau_0 \models \mathsf{C}$ (VIII)
$\wedge \quad \mathsf{H}, \Pi_1, (\mathsf{e}_0, \sigma_0, \tau_0) \rightsquigarrow (\mathsf{e}'_0, \sigma'_0, \tau_0), \mathsf{H}', \Pi'_1\ ]$ (IX)
Furthermore, from G4, (VI), (IX) and Lemma 1 we know:
$\Pi' = \Pi'_1 \uplus \Pi_2$ (X)

On the other hand, given (G1), (I), (VII) , (IX), (G5) and the induction hypothesis, we can deduce
$\mathsf{H}', \Pi'_1, \sigma'_0, \tau_0 \models \mathsf{A}'$ (XI)
Finally, Given (VIII), (X), (XI) and property P5, we can deduce:
$\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{A}' * \mathsf{C}$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \wedge \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1 \ldots \mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m})\}] \wedge \sigma_\mathsf{j}(\mathsf{this}) = \iota \wedge \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j}\quad \wedge\quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
as requitred.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$

From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$

From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$

From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$
as required.

Case Seq.
$\mathsf{P}_0 \equiv \mathsf{A}$ $\mathsf{Q}_0 \equiv \mathsf{B}$ $\mathsf{e}_0 \equiv \mathsf{C}_1; \mathsf{C}_2$
$\mathsf{e}'_0 = \mathsf{C}'_1; \mathsf{C}_2$ where $\mathsf{H}, \Pi, (\mathsf{C}_1, \tau_0, \sigma_0) \rightsquigarrow (\mathsf{C}'_1, \tau_0, \sigma'_0), \mathsf{H}', \Pi'$ (I)
$\{\mathsf{A}\}\ \mathsf{C}_1\ \{\mathsf{D}\}$ (II) $\{\mathsf{D}\}\ \mathsf{C}_2\ \{\mathsf{B}\}$ (III)

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{C}'_1; \mathsf{C}_2\ \{\mathsf{B}\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
From (G1)-(G5), (I), (II) and the induction hypothesis we deduce:
$\exists \mathsf{R}.[\ \{\mathsf{R}\}\ \mathsf{C}'_1\ \{\mathsf{D}\}$ (IV)
$\wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}\ ]$ (V)
Take $\mathsf{P}'_0 \equiv \mathsf{R}$
RTS. $\{\mathsf{R}\}\ \mathsf{C}'_1; \mathsf{C}_2\ \{\mathsf{B}\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}$
From (III), (IV) and the hoare logic judgement (Seq.) we have:
$\{\mathsf{R}\}\ \mathsf{C}'_1; \mathsf{C}_2\ \{\mathsf{B}\}$ (VI)
From (V) and (VI) we have:
$\{\mathsf{R}\}\ \mathsf{C}'_1; \mathsf{C}_2\ \{\mathsf{B}\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j}\quad \wedge\quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

From (G1)-(G5), (I), (II) and the induction hypothesis we can deduce:
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j}\quad \wedge\quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
as requitred.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$

From (G1)-(G5), (I), (II) and the induction hypothesis we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$

From (G1)-(G5), (I), (II) and the induction hypothesis we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$

From (G1)-(G5), (I), (II) and the induction hypothesis we can deduce:
$\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$
as required.

Case Conseq.
$\mathsf{P}_0 \rightarrow_\mathsf{a} \mathsf{A}$ (I) $\mathsf{B} \rightarrow_\mathsf{a} \mathsf{Q}_0$ (II)
$\{\mathsf{A}\}\ \mathsf{e}_0\ \{\mathsf{B}\}$ (III)

From G3, we know that: $\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{P}_0$ (IV)
From (I), (IV) and Property P3 of the $\models$ judgement we know that:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{A}$ (V)

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
By (G1),...,(G5), (III), (V) and induction hypothesis we can deduce:
$\exists \mathsf{R}.[\{\mathsf{R}\}\ \mathsf{e}'_0\ \{\mathsf{B}\}$ (VI)
$\wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}]$ (VII)
Take $\mathsf{P}'_0 \equiv \mathsf{R}$
RTS. $\{\mathsf{R}\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}$
From (G2) we have:
$\{\mathsf{P}_0\}\ \mathsf{e}_0\ \{\mathsf{Q}_0\}$ (VIII)
From (VIII) and lemma 7 we can deduce:
$\mathsf{SF}(\mathsf{Q}_0)$ (IX)
From (II), (VI), (IX) and the Hoare logic rule of consequence, we can deduce that:
$\{\mathsf{R}\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\}$ (X)
Combining (VII) and (X) we get:
$\{\mathsf{R}\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\}\quad \wedge\quad \mathsf{H}', \Pi'\sigma'_0, \tau_0 \models \mathsf{R}$ as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j}\quad \wedge\quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

From (III), (V), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j}\quad \wedge\quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
as requitred.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$

From (III), (V), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f})\ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g})\ \leq 1]$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$

From (III), (V), (G1)-(G5) and the induction hypothesis we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x_j} \mapsto \iota_\mathsf{j}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{j}$
for $\iota_\mathsf{j} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{j} = \mathsf{A_j}[\mathsf{x_j}/\mathsf{this}]$ $\mathsf{A_j} = \mathsf{Invariant}(\iota_\mathsf{j}.\mathsf{class})$
as required.

RTS. 3d. $\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$

From (III), (V), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau\ \wedge \mathsf{H}'(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$
as required.

D.2 Theorem 1
$\mathsf{Ver(Prog)}$ (G1)
$\{\mathsf{P_i}\}\ \mathsf{e_i}\ \{\mathsf{Q_i}\}\quad \mathsf{for\ i} \in 0...\mathsf{n}$ (G2)
$\mathsf{H}, \Pi, \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P_i}\quad \mathsf{for\ i} \in 0...\mathsf{n}$ (G3)
$(\mathsf{e}_0, \tau_0, \sigma_0) | (\overline{\mathsf{e}, \tau, \sigma}^{1...\mathsf{n}}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0) | (\overline{\mathsf{e}, \tau, \sigma}^{1...\mathsf{n}}), \mathsf{H}', \Pi'$ (G4)
$\mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0...\mathsf{n}}))$ (G5)
$\Longrightarrow$

1. $H', \Pi', \sigma'_i, \tau_i \models P_i$ for $i \in 1...n$
2. $\exists P'_0.[\ \{P'_0\}\ e'_0\ \{Q_0\} \ \wedge \ H', \Pi', \sigma'_0, \tau_0 \models P'_0\ ]$
3. $WF(H', \Pi', (e'_0, \tau_0, \sigma'_0)|(\overline{e, \tau, \sigma}^{1...n}))$)

Proof.
RTS. 1. $H', \Pi', \sigma'_i, \tau_i \models P_i$ for $i \in 1...n$

This can be obtained from (G1)-(G5) and theorem 1a.

RTS. 2. $\exists P'_0.[\ \{P'_0\}\ e'_0\ \{Q_0\} \ \wedge \ H', \Pi', \sigma'_0, \tau_0 \models P'_0\ ]$
This can be obtained from (G1)-(G5) and theorem 1a.

RTS. 3a. $\forall\ \kappa.[H'(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau)$
$\exists P, \exists j \in \{1...n\}.[\tau = \tau_j \ \wedge \ H', \Pi', \tau_j, \sigma_j \models P$
$\wedge\ \{P\}\ e_j\ \{Post(m)\}]\ \wedge\ \sigma_j(this) = \iota\ \wedge\ \sigma_j(\overline{u}) = \overline{\iota}\ ]$
where $Post(m) = A(\overline{u})$

There are now two cases:
Case 1. $\tau \neq \tau_0$
This follows immediately from (G1)-(G5) and theorem 1a.

Case 2. $\tau = \tau_0$
RTS. $\forall\ \kappa.[H'(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau_0) \implies$
$\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}]\ \wedge\ \sigma'_0(this) = \iota\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$

Take an arbitrary κ such that:
$H'(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau_0)$ (I)
RTS. $\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}\ \wedge\ \sigma'_0(this) = \iota'\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$

Proof. By induction over the Hoare logic Triplets ($\{P_0\}\ e_0\ \{Q_0\}$).

Case FldAss.
Since H' is exactly the same as H except for the value of $\sigma_0(x).f$, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
From (II) and (G5) we have:
$\exists P.[H, \Pi, \tau_0, \sigma_0 \models P$ (III)
$\{P\}\ e_0\ \{Post(m)\}$ (IV)
$\sigma_0(this) = \iota'$ (V)
$\sigma_0(\overline{u}) = \overline{\iota}$ (VI)
Since we have $\sigma'_0 = \sigma_0$, from (V) and (VI) we can deduce:
$\sigma'_0(this) = \iota'$ (VII)
$\sigma'_0(\overline{u}) = \overline{\iota}$ (VIII)

On the other hand, from (III), (IV), (G1)-(G5) and theorem 1a we can deduce:
$\exists P'.[\{P'\}\ e'_0\ \{Post(m)\}\ \wedge\ H', \Pi', \sigma'_0, \tau_0 \models P'$ (IX)
By putting together (VII), (VIII) and (IX) we have:
$\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}\ \wedge\ \sigma'_0(this) = \iota'\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$
as required.

Case VarAss.
Since $H' = H$, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case NewC
Since H' is exactly the same as H except for the value of the new address ι, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case Meth
Since $H' = H$, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case IfT, IfF
Since $H' = H$, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case Acquire
Since H' is exactly the same as H except for the monitor of address ι, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case Release
Since H' is exactly the same as H except for the monitor of address ι, from (I) we have:
$H(\kappa) = (TK, \{c : C, m : m, args : \iota'.\overline{\iota}\}, \tau)$. (II)
The rest of the proof is identical to the FldAss. case.

Case Frame
From Hoare logic judgement Frame we have:
$\{A\}\ e_0\ \{B\}$ (I)

From (G2) and the definition of P_0 we know:
$H, \Pi, \sigma_0, \tau_0 \models A * C$ (1)
From (1) and the definition of $\models$ we have:
$H, \Pi(\tau_0), \sigma_0 \models A * C$ (2)
From (2) and property P5 we have:
$H, \pi_1, \sigma_0 \models A$ (3)
$H, \pi_2, \sigma_0 \models C$ (4)
$\Pi(\tau_0) = \pi_1 \uplus \pi_2$ (5)
From (3), (5) and lemma 13a we have:
$H, \Pi(\tau_0), \sigma_0 \models A$ (6)

From (I), (6), (G1)-(G5) and the induction hypothesis we can deduce:
$\forall\ \kappa.[H'(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau_0) \implies$
$\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}]\ \wedge\ \sigma'_0(this) = \iota\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$
as required.

Case Seq.
From Hoare logic judgement Seq. we have:
$e'_0 = C'_1; C_2$ where $H, \Pi, (C_1, \tau_0, \sigma_0) \rightsquigarrow (C'_1, \tau_0, \sigma'_0), H', \Pi'$ (I)
$\{P_0\}\ C_1\ \{D\}$ (II) $\{D\}\ C_2\ \{Q_0\}$ (III)

From (G1)-(G5), (I), (II) and the induction hypothesis we can deduce:
$\forall\ \kappa.[H'(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau_0) \implies$
$\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}]\ \wedge\ \sigma'_0(this) = \iota\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$
as required.

Case Conseq.
From Hoare logic judgement Conseq. we have:
$P_0 \rightarrow_a A$ (I) $B \rightarrow_a Q_0$ (II)
$\{A\}\ e_0\ \{B\}$ (III)

From G3, we know that: $H, \Pi, \sigma_0, \tau_0 \models P_0$ (IV)
From (I), (IV) and Property P3 of the $\models$ judgement we know that:
$H, \Pi, \sigma_0, \tau_0 \models A$ (V)

From (G1)-(G5), (III), (V) and induction hypothesis we can deduce:
$\forall\ \kappa.[H'(\kappa) = (TK, \{c : C, m : m, args : \iota.\overline{\iota}\}, \tau_0) \implies$
$\exists P.[H', \Pi', \tau_0, \sigma'_0 \models P$
$\wedge\ \{P\}\ e'_0\ \{Post(m)\}]\ \wedge\ \sigma'_0(this) = \iota\ \wedge\ \sigma'_0(\overline{u}) = \overline{\iota}\ ]$
as required.

RTS. 3b. $\forall \iota.f[\sum_{\tau \in DOM(\Pi')} \Pi'(\tau)(\iota.f) \ \leq 1]$

This can be obtained from (G1)-(G5) and theorem 1a.

RTS. 3c. $H', \Pi', [x_i \mapsto \iota_i], \tau_g \models *A'_i$
for $\iota_i \in \{\iota' | \iota' \in Dom(H') \wedge H'(\iota') \downarrow_3 = \tau_g\}$ where
$A'_i = A_i[x_i/this]$ $A_i = Invariant(\iota_i.class)$

This can be obtained from (G1)-(G5) and theorem 1a.

RTS. 3d. $\forall \kappa, \kappa'.[H(\kappa) \downarrow_3 = \tau \ \wedge \ H(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$

This can be obtained from (G1)-(G5) and theorem 1a.

D.3 Theorem 2a

In order to prove Theorem 2, we first start by proving the following auxiliary theorem that would help us in proving the second soundness theorem.

$Ver(Prog)$ (G1)
$\{P_i\}\ e_i\ \{Q_i\}$ for $i \in 0...n-1$ (G2)
$H, \Pi, \sigma_i, \tau_i \models P_i$ for $i \in 0...n-1$ (G3)
$(e_0, \tau_0, \sigma_0)|(\overline{e, \tau, \sigma}^{1...n-1}), H, \Pi \rightsquigarrow$
$(e'_0, \tau_0, \sigma'_0)|(\overline{e, \tau, \sigma}^{1...n-1})|(e_n, \tau_n, \sigma_n), H', \Pi'$ (G4)
$WF(H, \Pi, (\overline{e, \tau, \sigma}^{0...n-1}))$ (G5)
$\Longrightarrow$
1. $H', \Pi', \sigma_i, \tau_i \models P_i$ for $i \in 1...n-1$
2. $\exists P_n, Q_n.[\{P_n\}\ e_n\ \{Q_n\} \ \wedge \ H', \Pi', \sigma_n, \tau_n \models P_n]$

3. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
4a. $\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{Tk}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\bar{\iota}\}, \tau)\ \wedge\ \tau \neq \tau_0\ \Longrightarrow$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\}]\ \wedge\ \sigma_\mathsf{j}(\mathsf{this}) = \iota\ \wedge\ \sigma_\mathsf{j}(\bar{\mathsf{u}}) = \bar{\iota}\ ]$
where $\mathsf{Post(m)} = \mathsf{A}(\bar{\mathsf{u}})$

4b. $\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$

4c. $\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

4d. $\forall \kappa, \kappa'.[\mathsf{H}(\kappa) \downarrow_3 = \tau\ \wedge\ \mathsf{H}(\kappa') \downarrow_3 = \tau\ \Longrightarrow\ \kappa = \kappa']$

Proof. By induction over the Hoare Logic triplets.
The only applicable case is when:
$\mathsf{P}_0 \equiv \mathsf{Pre(m)}[\mathsf{x}/\mathsf{this}][\bar{\mathsf{y}}/\bar{\mathsf{u}}] \quad \mathsf{e}_0 \equiv \mathsf{fork\ tk} := \mathsf{x.m}(\bar{\mathsf{y}})$
$\mathsf{Q}_0 \equiv \mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)} \qquad \mathsf{H}(\sigma_0(\mathsf{x})) \downarrow_1 = \mathsf{C} \qquad \mathsf{Pre(m)} = \mathsf{A}(\bar{\mathsf{u}})$
$\tau_\mathsf{n} \notin \Pi \quad \kappa \notin \mathsf{H} \quad \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}\}, \tau_\mathsf{n})\ ]$
$\sigma'_0 = \sigma_0[\mathsf{tk} \mapsto \kappa] \quad \mathsf{e}'_0 = \mathsf{null}$
$\sigma_\mathsf{n} = [\mathsf{this} \mapsto \sigma_0(\mathsf{x}),\ \bar{\mathsf{u}} \mapsto \sigma_0(\bar{\mathsf{y}})] \quad \mathsf{e}_\mathsf{n} = \mathsf{mBody(m)}$
$\Pi'' = \Pi[\tau_0 - = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Pre(m)}[\mathsf{x}/\mathsf{this}][\bar{\mathsf{y}}/\bar{\mathsf{u}}])$
$\tau_\mathsf{n} + = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Pre(m)}[\mathsf{x}/\mathsf{this}][\bar{\mathsf{y}}/\bar{\mathsf{u}}])\ ]$
$\Pi' = \Pi''[(\tau_0)(\kappa.\mathsf{c}) \mapsto 1$
$(\tau_0)(\kappa.\mathsf{m}) \mapsto 1$
$(\tau_0)(\kappa.\mathsf{args}) \mapsto 1\ ]$

RTS. 1. $\mathsf{H}', \Pi', \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P}_\mathsf{i} \qquad$ for $\mathsf{i} \in 1...\mathsf{n}-1$
This can be derived from Lemma 4.

RTS. 2. $\exists \mathsf{P}_\mathsf{n}, \mathsf{Q}_\mathsf{n}.[\{\mathsf{P}_\mathsf{n}\}\ \mathsf{e}_\mathsf{n}\ \{\mathsf{Q}_\mathsf{n}\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{P}_\mathsf{n}]$
Take $\mathsf{P}_\mathsf{n} = \mathsf{Pre(m)},\ \mathsf{Q}_\mathsf{n} = \mathsf{Post(m)}$
RTS. $\{\mathsf{Pre(m)}\}\ \mathsf{mBody(m)}\ \{\mathsf{Post(m)}\}$
From (G1) and the definition of $\mathsf{Ver(Prog)}$ we have:
$\{\mathsf{Pre(m)}\}\ \mathsf{mBody(m)}\ \{\mathsf{Post(m)}\}$ as required.

RTS. $\mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Pre(m)}$
From G3 we have:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{Pre(m)}[\mathsf{x}/\mathsf{this}][\bar{\mathsf{y}}/\bar{\mathsf{u}}]$ (I)
Since $\mathsf{Pre(m)} = \mathsf{A}(\bar{\mathsf{u}})$, we can deduce:
$\mathsf{FV(Pre(m))} = \{\mathsf{this}, \bar{\mathsf{u}}\}$ (II)
Now given the definition of σ_n, from (I), (II) and property P1, we can deduce:
$\mathsf{H}, \Pi, \sigma_\mathsf{n}, \tau_0 \models \mathsf{Pre(m)}[\mathsf{x}/\mathsf{this}][\bar{\mathsf{y}}/\bar{\mathsf{u}}][\mathsf{this}/\mathsf{x}][\bar{\mathsf{u}}/\bar{\mathsf{y}}]$, that is:
$\mathsf{H}, \Pi, \sigma_\mathsf{n}, \tau_0 \models \mathsf{Pre(m)}$ (III)
From (III), the definition of Π'' and lemma 5 we can deduce:
$\mathsf{H}, \Pi'', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Pre(m)}$ (IV)
From (IV) and the definition of $\models$ we have:
$\mathsf{H}, \Pi''(\tau_\mathsf{n}), \sigma_\mathsf{n} \models \mathsf{Pre(m)}$ (V)
From definition of Π', we know:
$\Pi'(\tau_\mathsf{n}) = \Pi''(\tau_\mathsf{n})$ (VI)
From (V) and (VI) we have:
$\mathsf{H}, \Pi'(\tau_\mathsf{n}), \sigma_\mathsf{n} \models \mathsf{Pre(m)}$ (VII)
From (VII) and the definition of $\models$ we have:
$\mathsf{H}, \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Pre(m)}$ (VIII)

We now show: $\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})}{\equiv} \mathsf{H}'$
In other words, show:
$\forall(\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\iota.\mathsf{f}) > 0\ \Longrightarrow\ \mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})]$
$\wedge \quad \forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\kappa.\mathsf{g}) > 0\ \Longrightarrow\ \mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})]$

RTS. $\forall(\iota.\mathsf{f})[\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\iota.\mathsf{f}) > 0\ \Longrightarrow\ \mathsf{H}'(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})]$
Take an arbitrary $(\iota.\mathsf{f})$ such that $\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\iota.\mathsf{f}) > 0$
RTS. $\mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})$
Since H is exactly the same as H' except for the value of κ, we can deduce:
$\mathsf{H}(\iota.\mathsf{f}) = \mathsf{H}'(\iota.\mathsf{f})$ (IX)

RTS. $\wedge \quad \forall(\kappa.\mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\kappa.\mathsf{g}) > 0\ \Longrightarrow\ \mathsf{H}'(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})]$
Take an arbitrary $(\kappa'.\mathsf{g})$ such that $\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\kappa'.\mathsf{g}) > 0$
RTS. $\mathsf{H}(\kappa'.\mathsf{g}) = \mathsf{H}'(\kappa'.\mathsf{g})$
Since $\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})(\kappa'.\mathsf{g}) > 0$, from property P11 we can deduce:
$\kappa' \in \mathsf{Dom}(\mathsf{H})$ (X)
On the other hand from our given we know that:
$\kappa \notin \mathsf{Dom}(\mathsf{H})$ (XI)
From (X) and (XI) we can deduce:
$\kappa' \neq \kappa$ (XII)
Since H is exactly the same as H' except for the value of κ, from (XII) we can deduce:
$\mathsf{H}(\kappa'.\mathsf{g}) = \mathsf{H}'(\kappa'.\mathsf{g})$ (XIII)
From (IX) and (XIII) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H}, \sigma_\mathsf{n}, \mathsf{Pre(m)})}{\equiv} \mathsf{H}'$ (XIV)
On the other hand from (G1) and the definition of $\mathsf{Ver(Prog)}$ we know:
$\mathsf{SF(Pre(m))}$ (XV)
From (XV), (VIII), (XIV) and the definition of self-framing assertions we have:
$\mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Pre(m)}$ as required.

RTS. 3. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
Take $\mathsf{P}'_0 \equiv \mathsf{Q}_0 \equiv \mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}$.

RTS. $\{\mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}\}\ \mathsf{null}\ \{\mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}\}$
From Lemma 12 we know:
$\mathsf{SF(Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{))}$ (I)
From (XI) and the Hoare judgement (Val.) we can deduce:
$\{\mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}\}\ \mathsf{null}\ \{\mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}\}$ as required.

RTS. $\mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{Thread(tk, C, m, x.}\bar{\mathsf{y}}\mathsf{)}$.
Let us divide $\Pi'(\tau_0)$ into 4 distinct permission masks $(\pi_1, ..., \pi_4)$, such that:

$\mathsf{Dom}(\pi_1) = (\kappa.\mathsf{c})$ and

$$\pi_1(\kappa'.g) = \begin{cases} \Pi'(\tau_0)(\kappa.c) & \text{if } \kappa'.g = \kappa.c \\ \bot & \text{otherwise} \end{cases}$$

$\mathsf{Dom}(\pi_2) = (\kappa.\mathsf{m})$ and

$$\pi_2(\kappa'.g) = \begin{cases} \Pi'(\tau_0)(\kappa.m) & \text{if } \kappa'.g = \kappa.m \\ \bot & \text{otherwise} \end{cases}$$

$\mathsf{Dom}(\pi_3) = (\kappa.\mathsf{args})$ and

$$\pi_3(\kappa'.g) = \begin{cases} \Pi'(\tau_0)(\kappa.args) & \text{if } \kappa'.g = \kappa.args \\ \bot & \text{otherwise} \end{cases}$$

$\mathsf{Dom}(\pi_4) = \mathsf{Dom}(\Pi'(\tau_0)) \setminus \{\mathsf{Dom}(\pi_1) \cup \mathsf{Dom}(\pi_2) \cup \mathsf{Dom}(\pi_3)\}$ and

$$\pi_4(\iota'.f) = \begin{cases} \Pi'(\tau_0)(\iota'.f) & \text{if } \iota'.f \in Dom(\Pi'(\tau_0)) \wedge\ \iota'.f \neq \kappa.c \\ & \quad \wedge\ \iota'.f \neq \kappa.m \wedge \iota'.f \neq \kappa.args \\ \bot & \text{otherwise} \end{cases}$$

Then from the definition of $\uplus$ on permission masks we have:
$\Pi'(\tau_0) = \pi_1 \uplus \pi_2 \uplus \pi_3 \uplus \pi_4$ (II)
From the definition of σ'_0, we know:
$\sigma'_0(\mathsf{tk}) = \kappa$ (III)
From the definition of H' we know that:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}\}, \tau_\mathsf{n})$ (IV)
Since σ'_0 is exactly the same as σ except for the value of tk, we can deduce:
$\sigma'_0(\mathsf{x}) = \sigma_0(\mathsf{x}) \qquad \overline{\sigma'_0(\mathsf{y})} = \overline{\sigma_0(\mathsf{y})}$ (V)
From (IV) and (V) we have:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma'_0(\mathsf{x}).\overline{\sigma'_0(\mathsf{y})}\}, \tau_\mathsf{n})$ (VI)
From (III), (VI) and property P4a we can deduce:
$\mathsf{H}', \epsilon, \sigma'_0 \models \mathsf{tk.c} = \mathsf{C}$ (VII)
$\mathsf{H}', \epsilon, \sigma'_0 \models \mathsf{tk.m} = \mathsf{m}$ (VIII)
$\mathsf{H}', \epsilon, \sigma'_0 \models \mathsf{tk.args} = \mathsf{x}.\bar{\mathsf{y}}$ (IX)
From (VII), (VIII), (IX) and applying property P5 two times, we know:
$\mathsf{H}', \epsilon \uplus \epsilon \uplus \epsilon, \sigma'_0 \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\bar{\mathsf{y}}$ (X)
From the definition of $\uplus$ on permission masks we have: $\epsilon \uplus \epsilon \uplus \epsilon = \epsilon$ and hence from (X) we have:
$\mathsf{H}', \epsilon, \sigma'_0 \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\bar{\mathsf{y}}$ (XI)
From (XI) and lemma 13 we have:
$\mathsf{H}', \epsilon \uplus \pi_4, \sigma'_0 \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\bar{\mathsf{y}}$ (XII)
From the definition of $\uplus$ on permission masks we have:
$\pi_4 = \epsilon \uplus \pi_4$ (XIII)
From (XII), (XIII) we have:
$\mathsf{H}', \pi_4, \sigma'_0 \models \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x}.\bar{\mathsf{y}}$ (XIV)
From the definition of Π' we know:
$\Pi'(\tau_0)(\kappa.\mathsf{c}) = 1$ (XV)
$\Pi'(\tau_0)(\kappa.\mathsf{m}) = 1$ (XVI)
$\Pi'(\tau_0)(\kappa.\mathsf{args}) = 1$ (XVII)
From (XV) and the definition of π_1 we have:
$\pi_1(\kappa.\mathsf{c}) = 1$ (XVIII)
From (XVI) and the definition of π_2 we have:
$\pi_2(\kappa.\mathsf{m}) = 1$ (XIX)
From (XVII) and the definition of π_3 we have:
$\pi_3(\kappa.\mathsf{args}) = 1$ (XX)
From (III), (XVIII) and property P4b we have:
$\mathsf{H}', \pi_1, \sigma'_0 \models \mathsf{acc(tk.c}, 1)$ (XXI)
From (XIII), (XIX) and property P4b we have:
$\mathsf{H}', \pi_2, \sigma'_0 \models \mathsf{acc(tk.m}, 1)$ (XXII)
From (XIII), (XX) and property P4b we have:
$\mathsf{H}', \pi_3, \sigma'_0 \models \mathsf{acc(tk.args}, 1)$ (XXIII)

From (XIV), (XXI), (XXII), (XXIII) and applying property P5 three times we have:
$\mathsf{H}', \pi_1 \uplus \pi_2 \uplus \pi_3 \uplus \pi_4, \sigma_0' \models \mathsf{acc}(\mathsf{tk.c}, 1) * \mathsf{tk.c} = \mathsf{C}$
$* \ \mathsf{acc}(\mathsf{tk.m}, 1) * \mathsf{tk.m} = \mathsf{m}$
$* \ \mathsf{acc}(\mathsf{tk.args}, 1) * \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$ (XXIV)
From (II) and (XXIV) we have:
$\mathsf{H}', \Pi'(\tau_0), \sigma_0' \models \mathsf{acc}(\mathsf{tk.c}, 1) * \mathsf{tk.c} = \mathsf{C}$
$* \ \mathsf{acc}(\mathsf{tk.m}, 1) * \mathsf{tk.m} = \mathsf{m}$
$* \ \mathsf{acc}(\mathsf{tk.args}, 1) * \mathsf{tk.args} = \mathsf{x}.\overline{\mathsf{y}}$ (XXV)
From (XXV) and property P12 we have:
$\mathsf{H}', \Pi'(\tau_0), \sigma_0' \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$ (XXVI)
From (XXVI) and the definition of the $\models$ judgement we have:
$\mathsf{H}', \Pi', \sigma_0', \tau_0 \models \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$ as required.

RTS. 4a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \ \wedge\ \tau \neq \tau_0 \implies$
$\exists \mathsf{j} \in \{1...\mathsf{n}\}, \exists \mathsf{P}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}]$

Take an arbitrary κ' such that:
$\mathsf{H}'(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)$. (1)
$\tau \neq \tau_0$ (2)
RTS.
$\exists \mathsf{j} \in \{1...\mathsf{n}\}, \exists \mathsf{P}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}]$

We have the following cases:
Case 1.
κ' refers to a token other than the newly created token κ. That is:
$\kappa' \neq \kappa$ (3)
Since H' is exactly the same as H except for the value of κ, from (1) and (2) we can deduce:
$\mathsf{H}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)$. (4)
From G5 and (4) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ])$ (5)
From (5) and (2) we have:
$\{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}\}$ (6)
$\mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$ (7)
$\sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}$ (8)
$\mathsf{j} \in \{1...\mathsf{n}-1\}$ (9)
From (6), (7), (G2)-(G5) and applying Lemma 4 we get:
$\mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$ (10)
By putting together (6) and (8) - (10) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$ (11)

Case 2.
κ' refers to the newly created token κ. That is:
$\kappa' = \kappa$ (12)
From the definition of H' we have:
$\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}\}, \tau_\mathsf{n})$ (13)
From (12) and (13) we have:
$\mathsf{H}'(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}\}, \tau_\mathsf{n})$ (14)
From (1) and (14) we have:
$\tau = \tau_\mathsf{n}$ (15)
$\sigma_0(\mathsf{x}) = \iota \qquad \overline{\sigma_0(\mathsf{y})}) = \overline{\iota}$ (16)
From the definition of σ_n we have:
$\sigma_\mathsf{n}(\mathsf{this}) = \sigma_0(\mathsf{x}) \qquad \overline{\sigma_\mathsf{n}(\mathsf{u})) = \sigma_0(\mathsf{y}))}$ (17)
From (16) and (17) we have:
$\sigma_\mathsf{n}(\mathsf{this}) = \iota \qquad \overline{\sigma_\mathsf{n}(\mathsf{u}))} = \overline{\iota}$ (18)
In part 2 of the proof for the Fork case we showed that:
$\{\mathsf{Pre(m)}\}\ \mathsf{e}_\mathsf{n}\ \{\mathsf{Post(m)}\}$ (19)
$\mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Pre(m)}$ (20)
From (19) and (20) we have:
$\exists\ \mathsf{P}.[\mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{P} \quad \wedge \quad \{\mathsf{P}\}\ \mathsf{e}_\mathsf{n}\ \{\mathsf{Post(m)}\}]$ (21)
By putting together (15), (18) and (21) we have:
$\exists \mathsf{P}.[\tau = \tau_\mathsf{n} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{n}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{n}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{n}(\overline{\mathsf{u}}) = \overline{\iota}]$ (22)

From (11) and (22) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
as required.

RTS. 4b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$
From G5 we have:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (I)
From definition of Π'', we know Π'' is exactly the same as Π except for the permissions of current thread τ_0 and the newly created thread τ_n. In Π'' we have stripped τ_0 from the permissions of method m's precondition and granted these permissions to τ_n. In other words, the sum of permissions has not changed from Π and we have just changed the owner of these permissions. That is:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi'')} \Pi''(\tau)(\iota.\mathsf{f}) = \sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f})]$
$\forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi'')} \Pi''(\tau)(\kappa.\mathsf{g}) = \sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{g})]$ (II)
From (I) and (II) we have:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi'')} \Pi''(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi'')} \Pi''(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (III)

On the other hand we know Π' is exactly the same as Π'' except for *addition* of new permissions for κ. In other words, for those locations already in H, Π' is the same as Π''. Therefore, from (III) we can deduce:
$\forall \iota'.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota'.\mathsf{f}) \leq 1$
$\wedge\ \forall \kappa'.\mathsf{g}[\kappa' \in \mathsf{H} \implies \sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa'.\mathsf{g}) \ \leq 1]$ (IV)
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$
On the other hand, for the new location κ, we add a new entry for each of $\{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ under (τ_0). Since κ did not exist in the heap before this point ($\kappa \notin \mathsf{H}$), no other thread could have had any permissions to any of its fields, that is:
$\sum_{\tau \in \mathsf{DOM}(\Pi') \setminus \{\tau_0\}} \Pi'(\tau)(\kappa.\mathsf{g}) = 0$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (V)
Furthermore, we gave τ_0 full permission (denoted by 1) to $\kappa.\mathsf{c}, \kappa.\mathsf{m}, \kappa.\mathsf{args}$. That is:
$\Pi'(\tau_0)(\kappa.\mathsf{g}) = 1$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (VI)
From (V) and (VI) we have:
$\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \leq 1$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (VII)
From (IV) and (VII) we have:
$\forall \iota'.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota'.\mathsf{f}) \leq 1$
$\wedge\ \forall \kappa'.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa'.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$
as required.

RTS. 4c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}_\mathsf{i}'$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}_\mathsf{i}' = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}_\mathsf{i}'$ (1)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (2) where
$\mathsf{A}_\mathsf{i}' = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_\mathsf{g}$.
Since H' is exactly the same as H, except for the value of κ, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}_\mathsf{i}'$ (5)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (6) where
$\mathsf{A}_\mathsf{i}' = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (7)

On the other hand, from (G1) and the definition of $\mathsf{Ver(Prog)}$, for each one of the A_i we have:
$\mathsf{SF}(\mathsf{A}_\mathsf{i})$ (8)
From (8), definitions of A_i' and lemma 8 we have:
$\mathsf{SF}(\mathsf{A}_\mathsf{i}')$ (9)
From (9) and property P8 we have:
$\mathsf{SF}(*(\mathsf{A}_\mathsf{i}'))$ (10)

Furthermore, we know that:
$\tau_0 \neq \tau_\mathsf{g}$ (11)
Since τ_g is the ghost thread that does not take part in execution. In other words, we cannot have:
$\mathsf{H}, \Pi, (\mathsf{e}_0, \tau_\mathsf{g}, \sigma_0) \rightsquigarrow (\mathsf{e}_0', \tau_\mathsf{g}, \sigma_0') | (\mathsf{e}_\mathsf{n}, \tau_\mathsf{n}, \sigma_\mathsf{n}), \mathsf{H}', \Pi'$.

Finally, from (5), (10), (11), (G2)-(G5) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}_\mathsf{i}'$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}_\mathsf{i}'$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}_\mathsf{i}' = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$
as required.

RTS. 4d.
$\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge\ \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa'$

Take arbitrary κ_1, κ_2 such that:
$\mathsf{H}'(\kappa_1) \downarrow_3 = \tau$ (1)
$\mathsf{H}'(\kappa_2) \downarrow_3 = \tau$ (2)
There are now three cases to consider:

Case 1.
$\kappa_1 \neq \kappa \kappa_2 \neq \kappa$ (3)
Since H' is exactly the same as H except for the value of κ, from (3) we can deduce:
$\mathsf{H}(\kappa_1) \downarrow_3 = \tau$ (4)
$\mathsf{H}(\kappa_2) \downarrow_3 = \tau$ (5)
From (4), (5) and (G5) we have:
$\kappa_1 = \kappa_2$ as required.

Case 2. $\kappa_1 = \kappa \ \wedge \ \kappa_2 \neq \kappa$ (6)
From the operational semantics of Acquire we know that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau_n$ (7)
From (7) and (6) and (1) we have:
$\tau = \tau_n$ (8)
On the other hand, since H' is exactly the same as H except for the value of κ, from (6) we can deduce:
$\mathsf{H}'(\kappa_2) = \mathsf{H}(\kappa_2)$ (9)
From (2) and (8) we have:
$\mathsf{H}'(\kappa_2) \downarrow_3 = \tau_n$ (10)
From (9) and (1) we can deduce:
$\mathsf{H}(\kappa_2) \downarrow_3 = \tau_n$ (11)
From (11) we can deduce:
$\tau_n \in \mathsf{range}(\mathsf{H})$ (12)
From operational semantics of Acquire we have:
$\tau_n \notin \mathsf{range}(\mathsf{H})$ (13)
Contradiction!!
Hence we can deduce:
$\kappa_1 = \kappa_2$ as required.

Case 3. $\kappa_1 = \kappa \ \wedge \ \kappa_2 = \kappa$ (14)
From (14) we trivially have:
$\kappa_1 = \kappa_2$ as required.

D.4 Soundness Theorem 2

$\mathsf{Ver}(\mathsf{Prog})$ (G1)
$\{\mathsf{P}_i\}\ \mathsf{e}_i\ \{\mathsf{Q}_i\}$ for $i \in 0...n-1$ (G2)
$\mathsf{H}, \Pi, \sigma_i, \tau_i \models \mathsf{P}_i$ for $i \in 0...n-1$ (G3)
$(\mathsf{e}_0, \tau_0, \sigma_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1...n-1}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1...n-1})|(\mathsf{e}_n, \tau_n, \sigma_n), \mathsf{H}', \Pi'$ (G4)
$\mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0...n-1}))$ (G5)
$\Longrightarrow$
1. $\mathsf{H}', \Pi', \sigma_i, \tau_i \models \mathsf{P}_i$ for $i \in 1...n-1$
2. $\exists \mathsf{P}_n, \mathsf{Q}_n.[\{\mathsf{P}_n\}\ \mathsf{e}_n\ \{\mathsf{Q}_n\} \ \wedge \ \mathsf{H}', \Pi', \sigma_n, \tau_n \models \mathsf{P}_n]$
3. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \ \wedge \ \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
4. $\mathsf{WF}(\mathsf{H}', \Pi', (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1...n}))$

Proof. By induction over the Hoare triplets $(\{\mathsf{P}_0\}\ \mathsf{e}_0\ \{\mathsf{Q}_0\})$.

The only applicable case is when:
$\mathsf{P}_0 \equiv \mathsf{Pre}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}] \quad \mathsf{e}_0 \equiv \mathsf{fork}\ \mathsf{tk} := \mathsf{x}.\mathsf{m}(\overline{\mathsf{y}})$
$\mathsf{Q}_0 \equiv \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}}) \quad \mathsf{H}(\sigma_0(\mathsf{x})) \downarrow_1 = \mathsf{C} \quad \mathsf{Pre}(\mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}})$
$\tau_n \notin \Pi \quad \kappa \notin \mathsf{H} \quad \mathsf{H}' = \mathsf{H}[\kappa \mapsto (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}\}, \tau_n)\ ]$
$\sigma'_0 = \sigma_0[\mathsf{tk} \mapsto \kappa] \quad \mathsf{e}'_0 = \mathsf{null}$
$\sigma_n = [\mathsf{this} \mapsto \sigma_0(\mathsf{x}),\ \overline{\mathsf{u}} \mapsto \sigma_0(\overline{\mathsf{y}})] \quad \mathsf{e}_n = \mathsf{mBody}(\mathsf{m})$
$\Pi'' = \Pi[\tau_0 - = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Pre}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])$
$\tau_n + = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Pre}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])\ ]$
$\Pi' = \Pi''[(\tau_0)(\kappa.\mathsf{c}) \mapsto 1$
$(\tau_0)(\kappa.\mathsf{m}) \mapsto 1$
$(\tau_0)(\kappa.\mathsf{args}) \mapsto 1\ ]$

RTS. 1. $\mathsf{H}', \Pi', \sigma_i, \tau_i \models \mathsf{P}_i$ for $i \in 1...n-1$

This can be derived from (G1)-(G5) and Theorem 2a.

RTS. 2. $\exists \mathsf{P}_n, \mathsf{Q}_n.[\{\mathsf{P}_n\}\ \mathsf{e}_n\ \{\mathsf{Q}_n\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_n, \tau_n \models \mathsf{P}_n]$

This can be derived from (G1)-(G5) and Theorem 2a.

RTS. 3. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$

This can be derived from (G1)-(G5) and Theorem 2a.

RTS. 4a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \implies$
$\exists \mathsf{j} \in \{1...n\}, \exists \mathsf{P}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_j\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma_j(\mathsf{this}) = \iota \quad \wedge \quad \sigma_j(\overline{\mathsf{u}}) = \overline{\iota}]$
$\vee \exists \mathsf{P}.[\tau = \tau_0 \quad \wedge \quad \mathsf{H}', \Pi', \tau_0, \sigma'_0 \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ' such that:
$\mathsf{H}'(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)$. (1)
There a re now two cases two consider:

Case 1. $\tau \neq \tau_0$
This can be derived from (G1)-(G5) and Theorem 2a.

Case 2.
$\tau = \tau 0$ (2)
From the operational semantics of Acquire we know that:
$\mathsf{H}'(\kappa) \downarrow_3 = \tau_n$ (3)
From (1) and (2) we have:
$\mathsf{H}'(\kappa') \downarrow_3 = \tau_0$ (4)
Hence we can deduce:
$\kappa' \neq \kappa$ (5)

Since H' is exactly the same as H except for the value of κ, from (1) and (2) and (5) we can deduce:
$\mathsf{H}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau_0)$. (6)
From G5 and (6) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{0...n-1\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}, \Pi, \tau_j, \sigma_j \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_j\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma_j(\mathsf{this}) = \iota \quad \wedge \quad \sigma_j(\overline{\mathsf{u}}) = \overline{\iota}\ ])$ (7)
From (2) and (7) we have:
$\{\mathsf{P}\}\ \mathsf{e}_0\ \{\mathsf{Post}(\mathsf{m})\}$ (8)
$\mathsf{H}, \Pi, \tau_0, \sigma_0 \models \mathsf{P}$ (9)
$\sigma_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma_0(\overline{\mathsf{u}}) = \overline{\iota}$ (10)
On the other hand from the definition of σ'_0 we know:
$\sigma'_0(\mathsf{this}) = \sigma_0(\mathsf{this}) \ \wedge \ \sigma'_0(\overline{\mathsf{u}}) = \sigma_0(\overline{\mathsf{u}})$ (11)
From (10) and (11) we have:
$\sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}$ (12)
From (G1)-(G5), (8), (9) and Theorem 2a we have:
$\exists \mathsf{P}.[\{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post}(\mathsf{m})\} \ \wedge \ \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}]$ (13)
By putting together (2), (12) and (13) we have:
$\exists \mathsf{P}.[\tau = \tau_0 \quad \wedge \quad \mathsf{H}', \Pi', \tau_0, \sigma'_0 \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}\ ])$
as required.

RTS. 4b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

This can be derived from (G1)-(G5) and Theorem 2a.

RTS. 4c.

This can be derived from (G1)-(G5) and Theorem 2a.

RTS. 4d.

This can be derived from (G1)-(G5) and Theorem 2a.

D.5 Theorem 3a

In order to prove Theorem 3, we first start by proving the following auxiliary theorem that would help us in proving the third soundness theorem.

$\mathsf{Ver}(\mathsf{Prog})$ (G1)
$(\{\mathsf{P}_i\}\ \mathsf{e}_i\ \{\mathsf{Q}_i\})_{i \in \{0...n\}}$ (G2)
$(\mathsf{H}, \Pi, \sigma_i, \tau_i \models \mathsf{P}_i)_{i \in \{0...n\}}$ (G3)
$(\mathsf{e}_0, \tau_0, \sigma_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1...n}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0)|(\overline{\mathsf{e}, \tau, \sigma}^{1...n-1}), \mathsf{H}', \Pi'$ (G4)
$\mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0...n}))$ (G5)
$\Longrightarrow$
1. $(\mathsf{H}', \Pi', \sigma_i, \tau_i \models \mathsf{P}_i)_{i \in \{1...n-1\}}$
2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi' \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
3a. $\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{Tk}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \ \wedge \ \tau \neq \tau_0 \implies$
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...n-1\}.[\tau = \tau_j \quad \wedge \quad \mathsf{H}', \Pi', \tau_j, \sigma_j \models \mathsf{P}$
$\wedge\ \{\mathsf{P}\}\ \mathsf{e}_j\ \{\mathsf{Post}(\mathsf{m})\}] \ \wedge \ \sigma_j(\mathsf{this}) = \iota \ \wedge \ \sigma_j(\overline{\mathsf{u}}) = \overline{\iota}\ ]$
where $\mathsf{Post}(\mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}})$

3b. $\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$

3c. $\mathsf{H}', \Pi', [\mathsf{x}_i \mapsto \iota_i], \tau_g \models *\mathsf{A}'_i$
for $\iota_i \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_g\}$ where
$\mathsf{A}'_i = \mathsf{A}_i[\mathsf{x}_i/\mathsf{this}] \qquad \mathsf{A}_i = \mathsf{Invariant}(\iota_i.\mathsf{class})$

3d. $\forall \kappa, \kappa'.[\mathsf{H}(\kappa) \downarrow_3 = \tau \ \wedge \ \mathsf{H}(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa']$

Proof.
By induction over the Hoare logic triplets. The only applicable case is when:
$\mathsf{P_0} \equiv \mathsf{Thread(tk, C, m, x.\overline{y})}$ (a)
$\mathsf{e_0} \equiv \mathsf{join\ tk}$ (b)
$\mathsf{Q_0} \equiv \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}$ where $\mathsf{Post(m) = A(\overline{u})}$ (c)
$\sigma_0(\mathsf{tk}) = \kappa$ (d)
$\mathsf{H'} = \mathsf{H}[\kappa \mapsto \epsilon]$ (e)
$\sigma'_0 = \sigma_0$ (f)
$\mathsf{e'_0} = \mathsf{null}$ (g)
$\Pi' = \Pi[\tau_0 + = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]})$
$\tau_\mathsf{n} - = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]})]$ (h)

RTS. 1. $\mathsf{H'}, \Pi', \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P_i}$ for $\mathsf{i} \in 1...\mathsf{n}-1$
This can be derived from Lemma 4.

RTS. 2. $\exists \mathsf{P'_0}.[\ \{\mathsf{P'_0}\}\ \mathsf{e'_0}\ \{\mathsf{Q_0}\} \quad \wedge \quad \mathsf{H'}, \Pi', \sigma'_0, \tau_0 \models \mathsf{P'_0}\ ]$
Take $\mathsf{P'_0} = \mathsf{Q_0} = \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}$
RTS. $\{\mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}\}\ \mathsf{null}\ \{\mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}\}$
From (G1) and the definition of Ver(Prog), we have:
$\mathsf{SF(Post(m))}$ (1)
From (1) and lemma 8 we have:
$\mathsf{SF(Post(m)[x/this][\overline{y}/\overline{u}])}$ (2)
From (2) and the Hoare logic judgement (Val.) we have:
$\{\mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}\}\ \mathsf{null}\ \{\mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}\}$ as required.

RTS. $\mathsf{H'}, \Pi', \sigma'_0, \tau_0 \models \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}$
By looking at the operational semantics of the join rule, we can deduce:
$\mathsf{e_n} \equiv \mathsf{v}$ (1)
$\mathsf{H}(\sigma_0(\mathsf{tk})) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C'}, \mathsf{m} : \mathsf{m'}, \mathsf{args} : \alpha.\overline{\alpha}\}, \tau_\mathsf{n})$ (2)
From G3 and (a) we know:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{Thread(tk, C, m, x.\overline{y})}$ (3):
From (3) and definition of $\models$ we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{Thread(tk, C, m, x.\overline{y})}$ (4)
From (4) and property P12 we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{acc(tk.c, 1)} * \mathsf{tk.c} = \mathsf{C}$
$* \ \mathsf{acc(tk.m, 1)} * \mathsf{tk.m} = \mathsf{m}$
$* \ \mathsf{acc(tk.args, 1)} * \mathsf{tk.args} = \mathsf{x.\overline{y}}$ (5)
From (5) and applying property P5 three times we have:
$\mathsf{H}, \pi_1, \sigma_0 \models \mathsf{tk.c} = \mathsf{C}$ (6)
$\mathsf{H}, \pi_2, \sigma_0 \models \mathsf{tk.m} = \mathsf{m}$ (7)
$\mathsf{H}, \pi_3, \sigma_0 \models \mathsf{tk.args} = \mathsf{x.\overline{(y)}}$ (8)
$\mathsf{H}, \pi_4, \sigma_0 \models \mathsf{acc(tk, 1)} * \mathsf{acc(tk.m, 1)} * \mathsf{acc(tk.args, 1)}$
where $\Pi(\tau_0) = \pi_1 \uplus \pi_2 \uplus \pi_3 \uplus \pi_4$
From (6), (7), (8) and property P4a we have:
$\mathsf{H}(\sigma_0(\mathsf{tk.c})) = \mathsf{C}$ (9)
$\mathsf{H}(\sigma_0(\mathsf{tk.m})) = \mathsf{m}$ (10)
$\mathsf{H}(\sigma_0(\mathsf{tk.args})) = \sigma_0(\mathsf{x}).\overline{\sigma_0(\mathsf{y})}$ (11)
From (2) and (9)-(11) we have:
$\mathsf{H}(\sigma_0(\mathsf{tk})) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \alpha.\overline{\alpha}\}, \tau_\mathsf{n})$, from d we have:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \alpha.\overline{\alpha}\}, \tau_\mathsf{n})$, (12):
where $\sigma_0(\mathsf{x}) = \alpha \quad \wedge \quad \overline{\sigma_0(\mathsf{y})} = \overline{\alpha}$ (13)
From (12) and (G5) we have:
$\exists \mathsf{P}, \exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau_\mathsf{n} = \tau_\mathsf{j} \ \wedge \ \mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post(m)}\} \ \wedge \ \sigma_\mathsf{j}(\mathsf{this}) = \iota \wedge \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}]$
In other words, we have:
$\exists \mathsf{P}.[\mathsf{H}, \Pi, \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{P}$ (14)
$\wedge \quad \{\mathsf{P}\}\ \mathsf{e_n}\ \{\mathsf{Post(m)}\}]$ (15)
$\sigma_\mathsf{n}(\mathsf{this}) = \alpha \quad \overline{\sigma_\mathsf{n}(\mathsf{u})} = \overline{\alpha}$ (16)
From (1) and (15) we have:
$\exists \mathsf{P}.[\{\mathsf{P}\}\ \mathsf{v}\ \{\mathsf{Post(m)}\}$ (17)
From (17) and the Hoare judgement (Val.) we have:
$\mathsf{P} \equiv \mathsf{Post(m)}$ (18)
From (14) and (18) we have:
$\mathsf{H}, \Pi, \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Post(m)}$ (19)
From (17) and (18) we have:
$\{\mathsf{Post(m)}\}\ \mathsf{v}\ \{\mathsf{Post(m)}\}$ (20)

We now show that $\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{Post(m)})}{\equiv} \mathsf{H'}$
$\forall(\iota, \mathsf{f}).[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Post(m)})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\iota.\mathsf{f}) = \mathsf{H'}(\iota.\mathsf{f})]$
$\wedge \ \forall(\kappa, \mathsf{g})[\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Post(m)})(\kappa, \mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0 \implies \mathsf{H}(\kappa.\mathsf{g}) = \mathsf{H'}(\kappa.\mathsf{g})]$

Take an arbitrary (ι, f) such that:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{P})(\iota, \mathsf{f}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$
Since $\mathsf{H'}$ is exactly the same as H except for the value of κ, we know that:
$\mathsf{H'}(\iota.\mathsf{f}) = \mathsf{H}(\iota.\mathsf{f})$ (I)

Now take an arbitrary $(\kappa'.\mathsf{g})$ such that: $\mathcal{P}(\mathsf{H}, \sigma', \mathsf{P})(\kappa'.\mathsf{g}) = \mathsf{n} \ \wedge \ \mathsf{n} > 0$ (G6)
where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$.
There are now two cases:
Case 1. $\kappa' \neq \kappa$
RTS. $\mathsf{H'}(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$
Since $\mathsf{H'}$ is exactly the same as H except for the value of κ, we can deduce that:
$\mathsf{H'}(\kappa'.\mathsf{g}) = \mathsf{H}(\kappa'.\mathsf{g})$ (II)

Case 2. $\kappa' = \kappa = \sigma_0(\mathsf{tk})$
RTS. $\mathsf{H'}(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$ From (a) and (G3) we know:
$\mathsf{H}, \Pi, \sigma_0, \tau_0 \models \mathsf{Thread(tk, C, m, args)}$
From the definition of $\models$ we know:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{Thread(tk, C, m, args)}$ (III)
From P12 and (III) we derive:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{acc(tk.c, 1)} * \mathsf{acc(tk.m, 1)} * \mathsf{acc(tk.args, 1)}$
$* \ \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x.\overline{y}}$ (IV)
From property P5 and (IV) we can write:
$\mathsf{H}, \pi_1, \sigma_0 \models \mathsf{acc(tk.c, 1)}$ (V)
$\mathsf{H}, \pi_2, \sigma_0 \models \mathsf{acc(tk.m, 1)} * \mathsf{acc(tk.args, 1)}$
$* \ \mathsf{tk.c} = \mathsf{C} * \mathsf{tk.m} = \mathsf{m} * \mathsf{tk.args} = \mathsf{x.\overline{y}}$ (VI)
$\Pi(\tau_0) = \pi_1 \uplus \pi_2$ (VII)
From (V), (VII) and lemma 13 a we have:
$\mathsf{H}, \Pi(\tau_0), \sigma_0 \models \mathsf{acc(tk.c, 1)}$ (VIII)
From (VIII) and property P4b we have:
$\Pi(\tau_0)(\sigma_0(\mathsf{tk.c})) \geq 1$, that is:
$\Pi(\tau_0)(\kappa.\mathsf{c}) \geq 1$ (IX)
On the other hand, From (19) we have:
$\mathsf{H}, \Pi, \sigma_\mathsf{b}, \tau_\mathsf{n} \models \mathsf{Post(m)}$, that is:
$\mathsf{H}, \Pi(\tau_\mathsf{n}), \sigma_\mathsf{n} \models \mathsf{Post(m)}$ (X)
By definition, we know permission masks are a set of mappings from locations and field identifiers to *(* positive) numbers between 0 and 1 inclusively. That is, we know:
$\Pi(\tau_\mathsf{n})(\kappa.\mathsf{c}) \geq 0$ (XI)
From (IX) and (XI) we have:
$\Pi(\tau_0)(\kappa.\mathsf{c}) + \Pi(\tau_\mathsf{n})(\kappa.\mathsf{c}) \geq 1$ (XII)
On the other hand from (G8) we know:
$\Pi(\tau_0)(\kappa.\mathsf{c}) + \Pi(\tau_\mathsf{n})(\kappa.\mathsf{c}) \leq 1$ (XIII)
From (XII) and (XIII) we know:
$\Pi(\tau_0)(\kappa.\mathsf{c}) + \Pi(\tau_\mathsf{n})(\kappa.\mathsf{c}) = 1$ (XIV)
From (IX) and (XIV) we can deduce:
$\Pi(\tau_\mathsf{n})(\kappa.\mathsf{c}) = 0$ (XV)
From (X), (XV) and property P2 we can deduce:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Post(m)})(\kappa.\mathsf{c}) \leq 0$
However, from (G6) we have:
$\mathcal{P}(\mathsf{H}, \sigma, \mathsf{Post(m)})(\kappa.\mathsf{c}) > 0$
Contradiction!!
Hence we can deduce:
$\mathsf{H'}(\kappa.\mathsf{g}) = \mathsf{H}(\kappa.\mathsf{g})$ (XVI)

From (I), (II) and (XVI) we can deduce:
$\mathsf{H} \overset{\mathcal{P}(\mathsf{H},\sigma,\mathsf{Post(m)})}{\equiv} \mathsf{H'}$ (XVII)

From (1), (19), (XVII) and the definition of self-framing assertions we can deduce:
$\mathsf{H'}, \Pi, \sigma_\mathsf{n}, \tau_\mathsf{n} \models \mathsf{Post(m)}$ (21)
Given (h), (21) and Lemma 5 we have:
$\mathsf{H'}, \Pi', \sigma_\mathsf{n}, \tau_0 \models \mathsf{Post(m)}$ (22)
From (13) and (16) we have:
$\overline{\sigma_0(\mathsf{x})} = \overline{\sigma_\mathsf{n}(\mathsf{this})} = \alpha$ (23)
$\overline{\sigma_0(\mathsf{y})} = \overline{\sigma_\mathsf{n}(\mathsf{u})} = \overline{\alpha}$ (24)
From (c) we know that post $\mathsf{Post(m)}$ is defined in terms of $\overline{\mathsf{u}}$.
Hence we can deduce:
$\mathsf{FV(Post(m))} = \{\mathsf{this}, \overline{\mathsf{u}}\}$ (25)
From (22), (23), (24), (25) and property P1 of the $\models$ judgement we have:
$\mathsf{H'}, \Pi', \sigma_0, \tau_0 \models \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}$ (26)
Finally, from f and (26) we have:
$\mathsf{H'}, \Pi', \sigma'_0, \tau_0 \models \mathsf{Post(m)[x/this][\overline{y}/\overline{u}]}$
as required.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H'}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \ \wedge \ \tau \neq \tau_0 \implies$
$\exists \mathsf{j} \in \{1...\mathsf{n}-1\}, \exists \mathsf{P}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H'}, \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \ \{\mathsf{P}\}\ \mathsf{e_j}\ \{\mathsf{Post}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ' such that:
$\mathsf{H'}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C'}, \mathsf{m} : \mathsf{m'}, \mathsf{args} : \alpha.\overline{\alpha}\}, \tau)$ (1)
$\tau \neq \tau_0$ (2)
From (e) we know that:
$\mathsf{H'}(\kappa) = \epsilon$ (3)
From (1) and (3) we know that:
$\kappa' \neq \kappa$ (4)

Since $\mathsf{H'}$ is exactly the same as H except for the value of κ, from above we can deduce:
$\mathsf{H}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}', \mathsf{m} : \mathsf{m}', \mathsf{args} : \alpha.\overline{\alpha}\}, \tau)$. (5)
From G5, (5) and (2) we can deduce:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m}')\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \mathsf{alpha} \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\alpha}\ ]$ (6)
There are now two cases:

Case 1.
$\tau = \tau_\mathsf{j} \qquad \mathsf{j} \in \{1...\mathsf{n}-1\}$ (7)
From (6) we have:
$\{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m}')\}$ (8)
$\mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$ (9)
$\sigma_\mathsf{j}(\mathsf{this}) = \alpha \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\alpha}$ (10)
From (8), (9), (G2)-(G5) and applying Lemma 4 we get:
blah$\mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$ (11)
By putting together (7), (11), (8) and (10) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m}')\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \alpha \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\alpha}\ ]$ (12)

Case 2.
$\tau = \tau_\mathsf{n}$ (13)
From (5) and (13) we have:
$\mathsf{H}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}', \mathsf{m} : \mathsf{m}', \mathsf{args} : \alpha.\overline{\alpha}\}, \tau_\mathsf{n})$ (14)
By looking at the operational semantics of the join rule, we can deduce:
$\mathsf{H}(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}', \mathsf{m} : \mathsf{m}', \mathsf{args} : \alpha.\overline{\alpha}\}, \tau_\mathsf{n})$ (15)
From (14), (15) and (G5) we have:
$\kappa = \kappa'$ (16)
From (4) we have:
$\kappa \neq \kappa'$ (17)
Contradiction!!
Hence we can deduce:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m}')\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \alpha \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\alpha}\ ]$ (18)

From (11) and (18) we can deduce:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{1...\mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \sigma_\mathsf{j}, \tau_\mathsf{j} \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m}')\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \alpha \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\alpha}\ ]$
as required.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

From G5 we have:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{g}) \ \leq 1]$where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$ (I)
From definition of Π', we know Π'' is exactly the same as Π except for the permissions of current thread τ_0 and the joined thread τ_n. In Π' we have stripped τ_n from the permissions of method m's postcondition and granted these permissions to τ_0. In other words, the sum of permissions has not changed from Π and we have just changed the owner of these permissions. That is:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) = \sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\iota.\mathsf{f})]$
$\forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) = \sum_{\tau \in \mathsf{DOM}(\Pi)} \Pi(\tau)(\kappa.\mathsf{g})]$ (II)
From (I) and (II) we have:
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge\ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$
as required.

RTS. 3c.
RTS. $\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$

From (G5) we know:
$\mathsf{H}, \Pi, [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$ (1)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (2) where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (3)

Take an arbitrary ι' such that $\mathsf{H}'(\iota) \downarrow_3 = \tau_\mathsf{g}$.
Since H' is exactly the same as H, except for the value of κ, we can deduce:
$\mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}$.
In other words we have:
$\{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}) \wedge \mathsf{H}(\iota') \downarrow_3 = \tau_\mathsf{g}\} = \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (4)
From (4) we can rewrite (1-3) as:
$\mathsf{H}, \Pi, [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$ (5)
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ (6) where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$ (7)

On the other hand, from (G1) and the definition of $\mathsf{Ver(Prog)}$, for each one of the A_i we have:
$\mathsf{SF}(\mathsf{A}_\mathsf{i})$ (8)
From (8), definitions of A'_i and lemma 8 we have:
$\mathsf{SF}(\mathsf{A}'_\mathsf{i})$ (9)
From (9) and property P8 we have:
$\mathsf{SF}(*(\mathsf{A}'_\mathsf{i}))$ (10)

Furthermore, we know that:
$\tau_0 \neq \tau_\mathsf{g}$ (11)
Since τ_g is the ghost thread that does not take part in execution. In other words, we cannot have:
$\mathsf{H}, \Pi, (\mathsf{e}_0, \tau_\mathsf{g}, \sigma_0) | (\mathsf{e}_\mathsf{n}, \tau_\mathsf{n}, \sigma_\mathsf{n}) \rightsquigarrow (\mathsf{e}'_0, \tau_\mathsf{g}, \sigma'_0), \mathsf{H}', \Pi'$.

Finally, from (5), (10), (11), (G2)-(G5) and Lemma 6 we can deduce:
$\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$ (8)
From (6), (7) and (8) we have:
$\mathsf{H}', \Pi', [\mathsf{x}_\mathsf{i} \mapsto \iota_\mathsf{i}], \tau_\mathsf{g} \models *\mathsf{A}'_\mathsf{i}$
for $\iota_\mathsf{i} \in \{\iota' | \iota' \in \mathsf{Dom}(\mathsf{H}') \wedge \mathsf{H}'(\iota') \downarrow_3 = \tau_\mathsf{g}\}$ where
$\mathsf{A}'_\mathsf{i} = \mathsf{A}_\mathsf{i}[\mathsf{x}_\mathsf{i}/\mathsf{this}] \qquad \mathsf{A}_\mathsf{i} = \mathsf{Invariant}(\iota_\mathsf{i}.\mathsf{class})$
as required.

RTS. 3d.
$\forall \kappa, \kappa'.[\mathsf{H}'(\kappa) \downarrow_3 = \tau \ \wedge \ \mathsf{H}'(\kappa') \downarrow_3 = \tau \implies \kappa = \kappa'$

Take arbitrary κ_1, κ_2 such that:
$\mathsf{H}'(\kappa_1) \downarrow_3 = \tau$ (1)
$\mathsf{H}'(\kappa_2) \downarrow_3 = \tau$ (2)
From the operational semantics of the Join rule we have:
$\mathsf{H}'(\kappa) = \epsilon$ (3)
From (1) (2) and (3) we can deduce:
$\kappa_1 \neq \kappa$ (4)
$\kappa_2 \neq \kappa$ (5)
Since H' is exactly the same as H except for the value of κ, from (4) and (5) we can deduce:
$\mathsf{H}(\kappa_1) \downarrow_3 = \tau$ (6)
$\mathsf{H}(\kappa_2) \downarrow_3 = \tau$ (7)
From (6), (7) and (G5) we have:
$\kappa_1 = \kappa_2$ as required.

D.6 Soundness Theorem 3

$\mathsf{Ver(Prog)} \ \wedge \ (\{\mathsf{P_i}\}\ \mathsf{e_i}\ \{\mathsf{Q_i}\})_{\mathsf{i} \in \{0...\mathsf{n}\}} \ \wedge \ (\mathsf{H}, \Pi, \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P_i})_{\mathsf{i} \in \{0...\mathsf{n}\}}$
$(\mathsf{e}_0, \tau_0, \sigma_0) | (\overline{\mathsf{e}, \tau, \sigma}^{1...\mathsf{n}}), \mathsf{H}, \Pi \rightsquigarrow (\mathsf{e}'_0, \tau_0, \sigma'_0) | (\overline{\mathsf{e}, \tau, \sigma}^{1...\mathsf{n}-1}), \mathsf{H}', \Pi'$
$\mathsf{WF}(\mathsf{H}, \Pi, (\overline{\mathsf{e}, \tau, \sigma}^{0...\mathsf{n}}))$
$\implies$
1. $(\mathsf{H}', \Pi', \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P_i})_{\mathsf{i} \in \{1...\mathsf{n}-1\}}$
2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi' \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$
3. $\mathsf{WF}(\mathsf{H}', \Pi', (\mathsf{e}'_0, \tau_0, \sigma'_0) | (\overline{\mathsf{e}, \tau, \sigma}^{1...\mathsf{n}-1}))$

Proof.
By induction over the Hoare logic triplets. The only applicable case is when:
$\mathsf{P}_0 \equiv \mathsf{Thread}(\mathsf{tk}, \mathsf{C}, \mathsf{m}, \mathsf{x}.\overline{\mathsf{y}})$ (a)
$\mathsf{e}_0 \equiv \mathsf{join\ tk}$ (b)
$\mathsf{Q}_0 \equiv \mathsf{Post}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}]$ where $\mathsf{Post}(\mathsf{m}) = \mathsf{A}(\overline{\mathsf{u}})$ (c)
$\sigma_0(\mathsf{tk}) = \kappa$ (d)
$\mathsf{H}' = \mathsf{H}[\kappa \mapsto \epsilon]$ (e)
$\sigma'_0 = \sigma_0$ (f)
$\mathsf{e}'_0 = \mathsf{null}$ (g)
$\Pi' = \Pi[\tau_0 + = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Post}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])$
$\tau_\mathsf{n} - = \mathcal{P}(\mathsf{H}, \sigma_0, \mathsf{Post}(\mathsf{m})[\mathsf{x}/\mathsf{this}][\overline{\mathsf{y}}/\overline{\mathsf{u}}])]$ (h)

RTS. 1. $\mathsf{H}', \Pi', \sigma_\mathsf{i}, \tau_\mathsf{i} \models \mathsf{P_i} \qquad$ for $\mathsf{i} \in 1...\mathsf{n}-1$

This can be derived from (G1)-(G5) and Theorem 3a.

RTS. 2. $\exists \mathsf{P}'_0.[\ \{\mathsf{P}'_0\}\ \mathsf{e}'_0\ \{\mathsf{Q}_0\} \quad \wedge \quad \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}'_0\ ]$

This can be derived from (G1)-(G5) and Theorem 3a.

RTS. 3a.
$\forall\ \kappa.[\mathsf{H}'(\kappa) = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau) \implies$
$\exists \mathsf{j} \in \{1...\mathsf{n}-1\}, \exists \mathsf{P}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}', \Pi', \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}]$
$\vee \exists \mathsf{P}.[\tau = \tau_0 \quad \wedge \quad \mathsf{H}', \Pi', \tau_0, \sigma'_0 \models \mathsf{P}$
$\wedge \{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post}(\mathsf{m})\} \quad \wedge \quad \sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}\ ]$

Take an arbitrary κ' such that:
$\mathsf{H}'(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau)$. (1)
There a re now two cases two consider:

Case 1. $\tau \neq \tau_0$
This can be derived from (G1)-(G5) and Theorem 3a.

Case 2.
$\tau = \tau 0$ (2)
From the operational semantics of Join we know that:
$\mathsf{H}'(\kappa) = \epsilon$ (3)
From (1) and (3) we have:
$\kappa' \neq \kappa$ (4)

Since H' is exactly the same as H except for the value of κ, from (1) and (2) and (4) we can deduce:
$\mathsf{H}(\kappa') = (\mathsf{TK}, \{\mathsf{c} : \mathsf{C}, \mathsf{m} : \mathsf{m}, \mathsf{args} : \iota.\overline{\iota}\}, \tau_0)$. (5)
From G5 and (5) we have:
$\exists \mathsf{P}.\exists \mathsf{j} \in \{0 \ldots \mathsf{n}-1\}.[\tau = \tau_\mathsf{j} \quad \wedge \quad \mathsf{H}, \Pi, \tau_\mathsf{j}, \sigma_\mathsf{j} \models \mathsf{P}$
$\wedge \ \{\mathsf{P}\}\ \mathsf{e}_\mathsf{j}\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma_\mathsf{j}(\mathsf{this}) = \iota \quad \wedge \quad \sigma_\mathsf{j}(\overline{\mathsf{u}}) = \overline{\iota}\ ])$ (6)
From (2) and (6) we have:
$\{\mathsf{P}\}\ \mathsf{e}_0\ \{\mathsf{Post(m)}\}$ (7)
$\mathsf{H}, \Pi, \tau_0, \sigma_0 \models \mathsf{P}$ (8)
$\sigma_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma_0(\overline{\mathsf{u}}) = \overline{\iota}$ (9)
On the other hand from the definition of σ'_0 we know:
$\sigma'_0(\mathsf{this}) = \sigma_0(\mathsf{this}) \ \wedge \ \sigma'_0(\overline{\mathsf{u}}) = \sigma_0(\overline{\mathsf{u}})$ (10)
From (9) and (10) we have:
$\sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}$ (11)
From (G1)-(G5), (7), (8) and Theorem 3a we have:
$\exists \mathsf{P}.[\{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post(m)}\} \ \wedge \ \mathsf{H}', \Pi', \sigma'_0, \tau_0 \models \mathsf{P}]$ (12)
By putting together (2), (11) and (12) we have:
$\exists \mathsf{P}.[\tau = \tau_0 \quad \wedge \quad \mathsf{H}', \Pi', \tau_0, \sigma'_0 \models \mathsf{P}$
$\wedge \ \{\mathsf{P}\}\ \mathsf{e}'_0\ \{\mathsf{Post(m)}\} \quad \wedge \quad \sigma'_0(\mathsf{this}) = \iota \quad \wedge \quad \sigma'_0(\overline{\mathsf{u}}) = \overline{\iota}\ ])$
as required.

RTS. 3b.
$\forall \iota.\mathsf{f}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\iota.\mathsf{f}) \ \leq 1]$
$\wedge \ \forall \kappa.\mathsf{g}[\sum_{\tau \in \mathsf{DOM}(\Pi')} \Pi'(\tau)(\kappa.\mathsf{g}) \ \leq 1]$ where $\mathsf{g} \in \{\mathsf{c}, \mathsf{m}, \mathsf{args}\}$

This can be derived from (G1)-(G5) and Theorem 3a.

RTS. 3c.

This can be derived from (G1)-(G5) and Theorem 3a.

RTS. 3d.

This can be derived from (G1)-(G5) and Theorem 3a.