# A TYPE SYSTEM EQUIVALENT TO MODEL CHECKING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mayur Hiru Naik

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2003

# ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Jens Palsberg. It has been a privilege working with Jens. Over the years, I have known him not only as an excellent researcher, but also as a wonderful person.

I would like to thank the researchers in the Software Productivity Tools group at Microsoft Research, particularly, Tom Ball and Sriram Rajamani, for teaching me much about model checking during an internship in the summer of 2002, and Jakob Rehof, for serving on my thesis committee and providing useful comments. I would also like to thank the other members of my committee, Prof. Ananth Grama and Prof. Zhiyuan Li.

Special thanks go to my colleagues in the Secure Software Systems (S3) group. My work has directly benefited from that of Dennis Brylow and Di Ma. I would also like to thank Bogdan Carbunar, Gergana Markova, and Tom VanDrunen for creating a cheerful work environment.

Finally, I am grateful to my parents for their love and support.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

Page

# LIST OF FIGURES

# ABSTRACT

Naik, Mayur Hiru, M.S., Purdue University, August 2003. A Type System Equivalent to Model Checking. Major Professor: Jens Palsberg.

Type systems and model checking are two approaches to finding bugs in software. Are they equally powerful? On the surface, they are quite different: type checking works on the program itself, often in a modular fashion, while model checking works on a model of the program. Type soundness ensures that all reachable program states satisfy a certain property provided the program is well typed, while model-checking soundness guarantees the same without any assumptions about types. Thus, model checking seems to be more powerful than type checking: Type soundness essentially says that type checking implies model checking, but there are not many cases where the converse holds. In this thesis, we present a type system that is equivalent to model checking. We work with a simple model checking problem, namely, deadline analysis: For an interrupt-driven program, will every interrupt be handled within the deadline? For the interrupt calculus of Palsberg and Ma, we construct finite-state models of programs that enable straightforward model checking for deadline analysis. We then present a type system which type checks exactly those programs that are accepted by the model checker. We use singleton types and, more significantly, intersection and union types containing information about time. Our result sheds light on the relationship between type systems and model checking, and it provides a means of explaining to the programmer *why* a run of a model checker succeeded and found no bugs. The reason is that the proof of equivalence is constructive and therefore enables type inference by model checking. Thus, a tool can output a version of the program that is annotated with timing information.

# 1. INTRODUCTION

## 1.1 Background

Several approaches to static program analysis are converging. The gulfs between flow analysis, type inference, and model checking are narrowing, as depicted in Figure 1.1. There is a growing sense that what can be accomplished with one technique can also be accomplished with the others. Previously published equivalences include:

Recursive Types + Subtyping = Flow-insensitive & Context-insensitive Flow Analysis
(Palsberg & O'Keefe [38])

Data Flow Analysis = Model Checking + Abstract Interpretation
(Schmidt & Steffen [44])

Can we combine such results and get a bridge between type systems and model checking? Not easily! The problem is that the results are for quite different flow

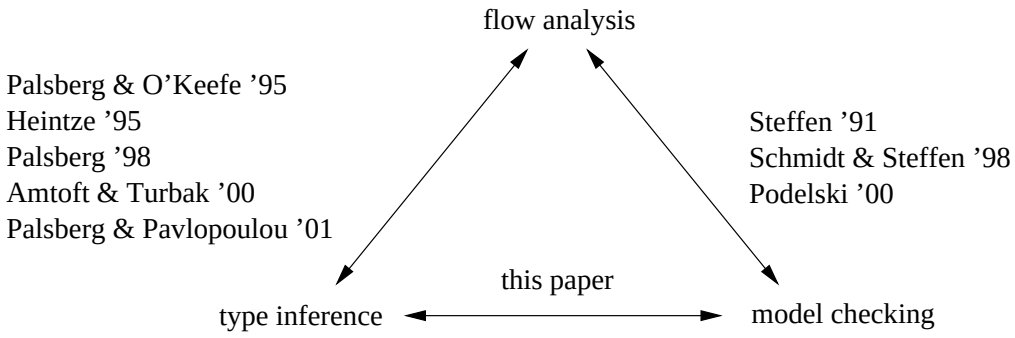


Figure 1.1 Relationships between three approaches to static analysis

| | Type Checking | Model Checking |
|---|---|---|
| Differences | works on the program | works on a model |
| | modular | whole program |
| | syntax driven | semantics driven |
| | checks all code | checks only reachable code |
| | precise check followed by abstraction | abstraction followed by precise check |
| Similarities | defined inductively | defined inductively |
| | finitary | finitary |

Figure 1.2 Comparison of type checking and model checking

analysis problems. Moreover, all of the published equivalences between type systems and flow analysis [38, 25, 36, 3, 39] are for higher-order languages, while those between flow analysis and model checking [45, 44, 42] are for first-order languages.

Our goal is to establish a connection between type systems and model checking. Both can be used to find bugs in software, so it seems natural to ask whether they are equally powerful. On the surface, they are quite different, see Figure 1.2. Type checking works on the program itself, often in a modular fashion, while model checking works on a model of the whole program. Type systems are usually defined in a mostly syntax-driven fashion with one type rule for each syntactic construct, while model checking is performed in a more semantics-driven style, based on the reachable state-space of an abstract semantics. Type systems tend to check all code, while model checking usually concentrates on reachable code, that is, code that is not "dead." A relatively minor similarity between type checking and model checking is that the set of type rules and the reachable state-space are both defined inductively. A more significant similarity is that a given type derivation uses only a finite number of

types, just as finite-state model checking works with a finite reachable state-space. This similarity is essential to this paper.

A fundamental difference between type systems and model checking comes to light when considering standard correctness results. Type soundness ensures that all reachable program states satisfy a certain property provided the program is well typed, while model-checking soundness guarantees the same without any assumptions about types. For example, a common type soundness theorem may say "well-typed programs cannot go wrong" [32]. Usually, "going wrong" is formalized as getting stuck in the operational semantics. With more mathematical notation, for a program $P$, a type $\tau$, typing judgments of the form $\tau \vdash P$, and a small-step semantics relation $\rightarrow$ with initial state $P_0$, we can write:

$$\text{If } \tau \vdash P_0 \text{ then } (\forall P' : P_0 \rightarrow^* P' \;\Rightarrow\; P' \text{ is not stuck}). \tag{1.1}$$

Notice that type checking does a precise type check using a predefined set of abstractions, namely, the types, after which the existence of a derivable type judgment implies that the program has the desired property. Model checking, however, is not concerned with types. Rather, the goal of model checking is to answer "direct" questions about programs, such as:

$$\forall P' : P_0 \rightarrow^* P' \;\Rightarrow\; P' \text{ is not stuck.}$$

Such questions are usually undecidable, so model checking works with a model, that is, an abstract semantics $\hookrightarrow$ with abstract states $Q$ and initial state $Q_0$, and can answer questions such as:

$$\forall Q' : Q_0 \hookrightarrow^* Q' \;\Rightarrow\; Q' \text{ is not stuck.}$$

Model-checking soundness then states that:

$$\begin{aligned} &\text{If } (\forall Q' : Q_0 \hookrightarrow^* Q' \;\Rightarrow\; Q' \text{ is not stuck}), \text{ then} \\ &\quad (\forall P' : P_0 \rightarrow^* P' \;\Rightarrow\; P' \text{ is not stuck}). \end{aligned} \tag{1.2}$$

```
ei                  handler 1 {          handler 2 {
loop {                 skip                 skip
   skip                iret                 ei
}                   }                       skip
                                            iret
                                        }
```

Figure 1.3 An example program in the interrupt calculus

Thus, in contrast to type checking, model checking first makes an abstraction, and then answers a precise question.

Overall, model checking seems to be more powerful than type checking: Type soundness essentially says that type checking implies model checking, but there are not many cases where the converse holds. To the best of our knowledge, there are no published equivalences between type checking and model checking.

## 1.2 The Model Checking Problem

We will focus on a simple, previously-studied model checking problem, namely, deadline analysis of interrupt-driven software:

*Deadline Analysis:* Will every interrupt be handled within the deadline?

Brylow and Palsberg [8] have presented a tool for practical deadline analysis of hand-written assembly code. Their tool does a static analysis that handles a myriad of low-level issues and relies on a moderate number of testing oracles.

We will study deadline analysis in a variant of the interrupt calculus of Palsberg and Ma [37]. Until now, the interrupt calculus has mainly been used to study algorithms for bounding the stack size of interrupt-driven software [37, 10]. A program in the calculus consists of a main procedure and one interrupt handler for each kind

of interrupt. Such a program models a microcontroller that receives and handles externally generated interrupts.

An example program with two interrupt handlers is shown in Figure 1.3. When a handler is called, interrupt handling is automatically disabled, and when a handler returns, at the point of iret, interrupt handling is automatically enabled. Moreover, a handler can enable interrupt handling, and thereby allow interruptions of itself, by executing the statement ei. For example, `handler 1` will execute without interruption and return, while `handler 2` will first do some processing, then execute ei and thereby allow interruptions, and finally do more processing. This implements that `handler 1` has higher priority: `handler 2` allows `handler 1` to interrupt it as soon as it has completed the most urgent of its tasks.

In our version of the interrupt calculus, timing plays a central role and is an integral part of the semantics. Each kind of interrupt has a period, that is, a lower bound on the interval between two interrupts of the same kind. Additionally, there is a deadline for handling each interrupt. Note that an interrupt can occur at a time when its handler is disabled and, furthermore, if its handler is later enabled, it may itself be interrupted by other kinds of interrupts. Thus, the deadline analysis problem is: Given a program together with periods and deadlines for all of the kinds of interrupts, decide whether every interrupt will be handled within its deadline. It is straightforward to design a model checker that does sound deadline analysis. In chapter 3, we will present one such model checker and prove a soundness theorem of the form (1.2). Henceforth, whenever we refer to model checking, it is this model checker that we mean.

## 1.3 Our Results

We present a type system equivalent to model checking. The type system type checks exactly those programs that are accepted by the model checker. We use singleton types [51, 52, 50] and intersection and union types [12, 26, 41, 46, 13, 7, 28, 47, 39]. The latter contain information about time. For example, the type of a

statement is of the form:

$$\bigwedge_{i \in A}((imr_i, \widehat{\overline{I}}_i) \longrightarrow \bigvee_{j \in B_i}(imr_j, \widehat{\overline{I}}_j))$$

Here, an $imr$ is a singleton type, namely, the type of the single value $imr$ of a variable imr in the program state that keeps track of which handlers are enabled, while an $\widehat{\overline{I}}$ is the type of a variable $\overline{I}$ in the program state that records how long each kind of interrupt has been waiting to be handled. In the degenerate case, the type of a statement assumes the form $(imr, \widehat{\overline{I}}) \longrightarrow (imr', \widehat{\overline{J}})$, which means that if the statement begins executing in context $(imr, \widehat{\overline{I}})$, then it will finish executing in context $(imr', \widehat{\overline{J}})$. However, not all statements will be executed right away; in particular, they may be interrupted. Thus, in general, if a statement begins executing in a particular context, then it will finish executing in one of several possible contexts. This is captured precisely using union types. Similarly, a statement might begin executing in one of a variety of contexts and in each such context finish executing in a different set of contexts. This is captured precisely using intersection types.

Our type system has one type rule for each syntactic construct. The type system does not use subtyping or recursive types. The unusual aspects are the use of singleton types and the use of intersection and union types with information about time. The type system satisfies a soundness theorem of the form (1.1).

Our result sheds light on the relationship between type systems and model checking, and it provides a means of explaining to the programmer *why* a run of a model checker succeeded and found no bugs. The reason is that the proof of equivalence is constructive and therefore enables type inference by model checking. Thus, a tool can output a version of the program that is annotated with timing information. We will show an annotated version of the example program in Figure 1.3 later in the paper.

Our equivalence theorem is of the form:

$$\tau \vdash P_0 \text{ if and only if } (\forall Q' : Q_0 \rightarrow^* Q' \ \Rightarrow \ Q' \text{ is not stuck}).$$

The forward direction is a variant of type soundness in which the step relation is the abstract semantics rather than the concrete semantics. It can be proved using

a well-known technique [35, 49] (see also [37] for a related proof for a much simpler type system for the interrupt calculus). The backward direction is proved by building a type derivation from the reachable state-space.

## 1.4 Related Work

Our work is inspired by recent progress in several fields. We view our work as a step towards relating the areas of flow-sensitive type systems and resource-usage analysis. As these fields continue to mature, there is great potential for further synergy and interplay between them. Here, we will briefly summarize the most closely related work in these areas, and also mention previous efforts to bridge different approaches to static analysis.

**Flow-Sensitive Type Systems**

Our type system is value sensitive, flow sensitive, and context sensitive. Value sensitivity is captured by singleton types [51, 52, 50] which are a special case of dependent types [31, 24]. Flow and context sensitivity is captured by intersection and union types [12, 26, 41, 46, 13, 7, 28, 47, 39]. Singleton, intersection, and union types have been used in the framework of refinement types [23, 18, 30]. The primary goal of refinement types is to find more bugs at compile time in programs that are already well typed in a conventional type system. It would be interesting to explore the relationship between refinement types and model checking.

**Resource-Usage Analysis**

Many programs must follow protocols, obey policies, or meet resource bounds. Conventionally, such properties have been checked using model checking [11, 5]. Recently, other approaches to static analysis have been applied in this domain, including type systems [16, 27, 29, 19, 22], data flow analysis [17], and theorem proving [20, 21]. Moreover, many approaches to resource-aware compilation for embedded systems use static analyses based on integer linear programming [48, 43, 34, 4]. The techniques presented in this paper may help establish connections between type-based approaches and model checking.

**Bridging Approaches to Static Analysis**

In addition to the equivalence results illustrated in Figure 1.1, there is other recent work on bridging different approaches to static analysis. Palsberg and Smith [40] show that a version of constrained types type check the same programs as the type system of Amadio and Cardelli with subtyping and recursive types [2]. Cousot [14] shows how several type systems can be derived as abstract interpretations of an untyped lambda-calculus. Cousot and Cousot [15] present a language- and semantics-independent, compositional, and inductive method for specifying formal semantics as well as proofs and analyses of programs in equivalent fixpoint, equational, constraint, closure-condition, rule-based, and game-theoretic form. Mossin [33] presents a sound and complete type-based flow analysis: it predicts a redex if and only if there exists a reduction sequence such that the redex will be reduced. Mossin's approach uses intersection types annotated with flow information; a related approach to flow analysis has been presented by Banerjee [6]. Abadi and Blanchet [1] prove the equivalence of a type-based and a logic-programming based approach to analyzing security protocols. Chaki, Rajamani and Rehof [9] present a type-based approach to model checking behavioral properties of distributed and asynchronous message-passing systems expressed in the $\pi$-calculus. They extract CCS models as types from the source code and perform compositional model checking on the types.

# 2. THE INTERRUPT CALCULUS

## 2.1 Syntax

Figure 2.1 presents the abstract syntax of the interrupt calculus. An interrupt-driven embedded system $\kappa$ consists of an environment $\rho$ communicating with a program $p$ by means of interrupts. The environment consists of a sequence of devices. We assume a fixed number of devices $N$. For arbitrary metavariable $x$, we use $\overline{x}$ as shorthand for the sequence $x_1, ..., x_N$ and we use the notation $\overline{x}(i) = x_i$. We use $\overline{x} = \overline{0}$ as shorthand for $\forall i \in 1..N : \overline{x}(i) = 0$ and $\overline{x} = \overline{y}$ as shorthand for $\forall i \in 1..N : \overline{x}(i) = \overline{y}(i)$. We use metavariables $u$ and $v$ to range over devices. The environment is specified as $(\overline{r}, \overline{d})$ where $\overline{r}$ and $\overline{d}$ denote the *periods* and *deadlines* of devices: $\overline{r}(u)$ is the minimum time between the arrivals of successive interrupts from device $u$ and $\overline{d}(u)$ is the maximum time within which an interrupt from device $u$ must be handled since its arrival. A program consists of a main procedure $m$ and interrupt handlers $\overline{h}$ where $\overline{h}(u)$ handles interrupts from device $u$. Metavariable $a$ ranges over $m$ and $h$. A statement $s$ is either a typical statement in an imperative language or a specialized statement of the form ei or imr := imr $\wedge$ $imr$ that is used for interrupt control. Statements skip, ei, imr := imr $\wedge$ $imr$, and $x := e$ are called primitive statements. We identify programs that are equivalent under the smallest congruence generated by the rules:

$$\begin{aligned}
(s_1 \ ; \ s_2) \ ; \ m &= s_1 \ ; \ (s_2 \ ; \ m) \\
(s_1 \ ; \ s_2) \ ; \ h &= s_1 \ ; \ (s_2 \ ; \ h) \\
(s_1 \ ; \ s_2) \ ; \ s &= s_1 \ ; \ (s_2 \ ; \ s)
\end{aligned}$$

With these rules, we can rearrange any $m$, $h$, or $s$ into one of the forms:

$$\mathsf{loop}\ s \qquad \mathsf{iret} \qquad \mathsf{skip}; a \qquad \mathsf{ei}; a \qquad \mathsf{imr} := \mathsf{imr} \wedge imr; a \qquad x := e; a$$
$$(\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a$$

$$
\begin{array}{llll}
\text{(system)} & \kappa & ::= & (\,\rho\,,\,p\,) \\
\text{(environment)} & \rho & ::= & (\,\overline{r}\,,\,\overline{d}\,) \qquad \overline{r} \geq \overline{d} > \overline{0} \\
\text{(program)} & p & ::= & (m,\,\overline{h}) \\
\text{(main)} & m & ::= & \mathsf{loop}\ s \mid s\ ;\ m \\
\text{(handler)} & h & ::= & \mathsf{iret} \mid s\ ;\ h \\
\text{(statements)} & s & ::= & \mathsf{skip} \mid \mathsf{ei} \mid \mathsf{imr} := \mathsf{imr} \wedge imr \mid x := e \mid \\
 & & & \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \mid s_1\ ;\ s_2 \\
\text{(expression)} & e & ::= & c \mid x \mid x + c \mid x_1 + x_2
\end{array}
$$

Figure 2.1 Interrupt calculus syntax

Henceforth, we assume that we are looking at a fixed system with environment $(\overline{r}, \overline{d})$ and program $(m_0, \overline{h})$.

## 2.2 Semantics

A program state $P$ is a 5-tuple $\langle a, imr, \overline{I}, \omega, \sigma \rangle$ where $a$ is the program counter, $imr$ is the value of the interrupt mask register $\mathsf{imr}$, $\overline{I}$ is the latency vector, $\omega$ is the stack generated by the grammar $\omega ::= \langle a, imr \rangle :: \omega \mid \mathsf{nil}$, and $\sigma$ is the store. We use the notation $P.a = a$, $P.imr = imr$, $P.\overline{I} = \overline{I}$, and $P.\omega = \omega$.

The $\mathsf{imr}$ provides a mechanism for masking interrupts from some or all devices and thereby enforcing mutual exclusion in critical sections of the program. It is a binary string $b_0\, b_1 \ldots b_N$ where $b_0$ is called the *master bit*. We use the notation $\mathsf{imr}(i) = b_i$. Interrupts from device $u$ are *enabled* if $\mathsf{imr}(0) = \mathsf{imr}(u) = 1$ and *disabled* otherwise. We use $\mathsf{t}_i$ to denote the value of $\mathsf{imr}$ where $b_i$ is 1 and the remaining bits are 0's. We use $\wedge$ to denote bitwise logical conjunction, $\vee$ to denote bitwise logical disjunction, and $\neg$ to denote bitwise logical negation.

The latency vector $\overline{I}$ is a vector of $N$ integers such that:

- If $\overline{I}(u) \geq 0$ then an interrupt from device $u$ has been *pending* for $\overline{I}(u)$ units, in particular, if $\overline{I}(u) = 0$ then an interrupt from device $u$ has just arrived.
- If $\overline{I}(u) < 0$ then no interrupt from device $u$ has been pending and, additionally, the next interrupt from device $u$ will arrive in precisely $|\overline{I}(u)|$ units.

A device $u$ is *latent* if interrupts from it are enabled and an interrupt from it has been pending. The set of latent devices is defined as follows:

Definition 2.1 $\mathcal{L}(imr, \overline{I}) = \{\ u \in 1..N \mid imr(0) = imr(u) = 1 \text{ and } \overline{I}(u) \geq 0\ \}$

The initial program state $P_0$ is of the form $\langle m_0, \neg \mathsf{t}_0, \overline{I}, \mathsf{nil}, \lambda x.0 \rangle$ where $\overline{I} \leq \overline{0}$. Notice that there are denumerably many initial program states. A small-step operational semantics for the interrupt calculus is given by the reflexive, transitive closure of the relation $\rightarrow$ on program states defined in Figure (2.2). The semantics uses the following auxiliary definitions:

$$\begin{aligned} \psi_u(imr) &= imr \wedge \neg(\mathsf{t}_0 \vee \mathsf{t}_u) \\ \\ \xi_\sigma(c) &= c \\ \xi_\sigma(x) &= \sigma(x) \\ \xi_\sigma(x + c) &= \sigma(x) + c \\ \xi_\sigma(x_1 + x_2) &= \sigma(x_1) + \sigma(x_2) \end{aligned}$$

For notational simplicity, we assume that each of the primitive statements and $\mathsf{iret}$ executes in $t$ units of time, where $t > 0$, and that the invocation of an interrupt handler, the jump in a loop, and the test in a branching statement are instantaneous.

If device $u$ is latent, then rule (2.1) is applied and handler $\overline{h}(u)$ is invoked. If multiple devices are latent, then one of them is handled non-deterministically. If no device is latent, then one of rules (2.2)–(2.9) is applied, depending upon the program counter, in which case the effect of primitive statements and $\mathsf{iret}$ on the latency vector is described by the corresponding latency vector transfer function in defn. (2.2).

Definition 2.2 **(Latency Vector Transfer Functions)**

(primitive statements)

$$\delta(\overline{I}) \;=\; \overline{J} \text{ s.t. } \forall u \in 1..N :\; \overline{J}(u) = \overline{I}(u) + t$$

($\mathsf{iret}_v$)

$$\delta_v(\overline{I}) \;=\; \overline{J} \text{ s.t. } \forall u \in 1..N :\; \overline{J}(u) = \begin{cases} \overline{I}(u) + t - r,\ r \geq \overline{r}(u) & \text{if } u = v \\ \overline{I}(u) + t & \text{otherwise} \end{cases}$$

We use the notation $\mathsf{iret}_v$ instead of iret because the execution of iret marks the completion of the handling of an interrupt from a device (device $v$ in this case) and the latency vector transfer function for iret needs $v$ to obtain the minimum time $\overline{r}(v)$ between the arrivals of successive interrupts from that device in order to determine, without violating the constraint imposed by $\overline{r}(v)$, when the next interrupt from that device will arrive. This is elucidated in the case $(u = v)$ in the definition of $\delta_v$ in defn. (2.2). For convenience, we summarize the effect of primitive statements on the imr in defn. (2.3).

Definition 2.3 **(imr Transfer Function, primitive statements)**

$$\begin{array}{lcll} \chi_s(imr) & = & imr & \text{if } s = \mathsf{skip} \text{ or } x := e \\ \chi_s(imr) & = & imr \vee \mathsf{t}_0 & \text{if } s = \mathsf{ei} \\ \chi_s(imr) & = & imr \wedge imr' & \text{if } s = \mathsf{imr} := \mathsf{imr} \wedge imr' \end{array}$$

Finally, the deadline analysis problem for the interrupt calculus is to decide whether $(P_0 \rightarrow^* P \Rightarrow P.\overline{I} < \overline{d})$.

$$\langle a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle \overline{h}(u), \psi_u(imr), \overline{I}, \langle a, imr\rangle :: \omega, \sigma\rangle$$
$$\text{if } u \in \mathcal{L}(imr, \overline{I}) \tag{2.1}$$

$$\langle \mathsf{iret}_v, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle a, imr', \delta_v(\overline{I}), \omega', \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \text{ and } \omega = \langle a, imr'\rangle :: \omega' \tag{2.2}$$

$$\langle \mathsf{loop}\ s, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle s; \mathsf{loop}\ s, imr, \overline{I}, \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \tag{2.3}$$

$$\langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle s_1; a, imr, \overline{I}, \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \text{ and } \sigma(x) = 0 \tag{2.4}$$

$$\langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle s_2; a, imr, \overline{I}, \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \text{ and } \sigma(x) \neq 0 \tag{2.5}$$

$$\langle \mathsf{skip}; a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle a, imr, \delta(\overline{I}), \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \tag{2.6}$$

$$\langle \mathsf{ei}; a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle a, imr \vee \mathsf{t}_0, \delta(\overline{I}), \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \tag{2.7}$$

$$\langle \mathsf{imr} := \mathsf{imr} \wedge imr'; a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle a, imr \wedge imr', \delta(\overline{I}), \omega, \sigma\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \tag{2.8}$$

$$\langle x := e; a, imr, \overline{I}, \omega, \sigma\rangle \;\rightarrow\; \langle a, imr, \delta(\overline{I}), \omega, \sigma\{x \mapsto \xi_\sigma(e)\}\rangle$$
$$\text{if } \mathcal{L}(imr, \overline{I}) = \emptyset \tag{2.9}$$

Figure 2.2 Interrupt calculus semantics

# 3. A MODEL CHECKER FOR DEADLINE ANALYSIS

## 3.1 Abstraction

Programs in the interrupt calculus are infinite-state (due to the store and latency vector components in the program state) and deadline analysis is undecidable (due to the store). We thereby define an abstraction (model) for interrupt calculus programs that eliminates the store and approximates the latency vector. Together, store elimination and latency vector approximation yields a finite reachable state-space, while store elimination alone renders the deadline analysis problem decidable.

An abstract program state $Q$ is a 4-tuple $\langle a, imr, \widehat{\overline{I}}, \omega \rangle$ where $\widehat{\overline{I}}$ is the abstract latency vector. We use the notation $Q.a = a$, $Q.imr = imr$, $Q.\widehat{\overline{I}} = \widehat{\overline{I}}$, and $Q.\omega = \omega$. The abstract latency vector $\widehat{\overline{I}}$ is a vector of $N$ integers such that:

- If $\widehat{\overline{I}}(u) \geq 0$ then an interrupt from device $u$ might be pending and, additionally, if it has been pending, then it will be so for at most $\widehat{\overline{I}}(u)$ units.
- If $\widehat{\overline{I}}(u) < 0$ then no interrupt from device $u$ has been pending and, additionally, the next interrupt from device $u$ will not arrive in less than $|\widehat{\overline{I}}(u)|$ units.

Notice that $\widehat{\overline{I}}$ approximates $\overline{I}$ if $\overline{I} \leq \widehat{\overline{I}}$.

A device $u$ is *apparently latent* if interrupts from it are enabled and an interrupt from it might be pending. The set of apparently latent devices is defined as follows:

Definition 3.1 $\widehat{\mathcal{L}}(imr, \widehat{\overline{I}}) = \{\ u \in 1..N \mid imr(0) = imr(u) = 1 \text{ and } \widehat{\overline{I}}(u) \geq 0\ \}$

The initial abstract program state $Q_0$ is $\langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil} \rangle$. The abstract small-step operational semantics is given by the reflexive, transitive closure of the binary relation $\hookrightarrow$ on abstract program states defined in Figure (3.1).

$$\langle a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle \overline{h}(u), \psi_u(imr), \widehat{\overline{I}}, \langle a, imr\rangle :: \omega\rangle \quad \text{if } u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{I}}) \tag{3.1}$$

$$\langle \mathsf{iret}_v, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle a, imr', \gamma_v(imr, \widehat{\overline{I}}), \omega'\rangle \quad \text{if } \omega = \langle a, imr'\rangle :: \omega' \tag{3.2}$$

$$\langle \mathsf{loop}\ s, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle s; \mathsf{loop}\ s, imr, \widehat{\overline{I}}, \omega\rangle \tag{3.3}$$

$$\langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle s_1; a, imr, \widehat{\overline{I}}, \omega\rangle \tag{3.4}$$

$$\langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle s_2; a, imr, \widehat{\overline{I}}, \omega\rangle \tag{3.5}$$

$$\langle \mathsf{skip}; a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle a, imr, \gamma(imr, \widehat{\overline{I}}), \omega\rangle \tag{3.6}$$

$$\langle \mathsf{ei}; a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle a, imr \vee \mathsf{t}_0, \gamma(imr, \widehat{\overline{I}}), \omega\rangle \tag{3.7}$$

$$\langle \mathsf{imr} := \mathsf{imr} \wedge imr'; a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle a, imr \wedge imr', \gamma(imr, \widehat{\overline{I}}), \omega\rangle \tag{3.8}$$

$$\langle x := e; a, imr, \widehat{\overline{I}}, \omega\rangle \hookrightarrow \langle a, imr, \gamma(imr, \widehat{\overline{I}}), \omega\rangle \tag{3.9}$$

Figure 3.1 Abstract semantics

If device $u$ is apparently latent, then rule (3.1) might be applied and handler $\overline{h}(u)$ invoked. If multiple devices are apparently latent, then one of them might be handled non-deterministically. If no device is apparently latent or if none of the apparently latent devices is veritably latent, then one of rules (3.2)–(3.9) will be applied, in which case the effect of primitive statements and iret on the abstract latency vector is described by the corresponding abstract latency vector transfer function in defn. (3.2).

Suppose that one of rules (3.2)–(3.9) is applied, that is, suppose that no device is apparently latent or none of the apparently latent devices is veritably latent. We expand on the case of the transfer function for primitive statements in which device $u$ is apparently latent, namely, the case $u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{I}})$. (The case $u \neq v \wedge u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{I}})$ of the transfer function for iret is similar.) Let $\overline{I}$ and $\overline{J}$ denote the latency vector before and after executing the statement. From the hypothesis that device $u$ is apparently

latent and not veritably latent, we have $\overline{I}(u) < 0$ (see defns. (2.1) and (3.1)). From $\overline{I}(u) < 0$ and defn. (2.2), we have $\overline{J}(u) < t$. Thus, $\overline{J}(u)$ can be approximated by $t-1$. Notice that $\overline{J}(u)$ can also be approximated by any integer $\geq t-1$, but $t-1$ is the best possible approximation.

Definition 3.2 **(Abstract Latency Vector Transfer Functions)**

(primitive statements)

$$\gamma(imr, \widehat{\overline{I}}) \;=\; \widehat{\overline{J}} \text{ s.t. } \forall u \in 1..N :\; \widehat{\overline{J}}(u) = \begin{cases} t-1 & \text{if } u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{I}}) \\ \widehat{\overline{I}}(u) + t & \text{otherwise} \end{cases}$$

($\mathsf{iret}_v$)

$$\gamma_v(imr, \widehat{\overline{I}}) \;=\; \widehat{\overline{J}} \text{ s.t. } \forall u \in 1..N :\; \widehat{\overline{J}}(u) = \begin{cases} \widehat{\overline{I}}(u) + t - \overline{r}(u) & \text{if } u = v \\ t-1 & \text{if } u \neq v \;\wedge u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{I}}) \\ \widehat{\overline{I}}(u) + t & \text{otherwise} \end{cases}$$

The reachable state-space of the model is as follows:

Definition 3.3 $\mathcal{R} = \{\; Q \mid Q_0 \hookrightarrow^* Q \;\}$.

The model checker decides whether the model satisfies the property $(Q_0 \hookrightarrow^* Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$. The reachable state-space $\mathcal{R}$ is finite if this property is satisfied. This is because the program counter, $\mathsf{imr}$, and stack components in an abstract (or even concrete) state of a given program have finitely many values irrespective of whether this property is satisfied while the abstract latency vector component has finitely many values if this property is satisfied. (The stack component has finitely many values irrespective of whether the property is satisfied because whenever handler $\overline{h}(u)$ is invoked, bit $b_u$ of the $\mathsf{imr}$ is reset, and it is next set only after $\overline{h}(u)$ returns. Thus, the execution of $\overline{h}(u)$ can never be interrupted by another invocation of $\overline{h}(u)$ directly or transitively, since invoking $\overline{h}(u)$ requires $b_u$ to be set, see rules (2.1) and (3.1). In other words, the stack size can never exceed $N-1$.) Thus, we have the following result:

Proposition 3.1 If $\forall Q \in \mathcal{R} :\; Q.\widehat{\overline{I}} < \overline{d}$ then $\mathcal{R}$ is finite.

## 3.2 Soundness

This section is devoted to proving the soundness of the model checker. For this purpose, we first define a binary relation $\mathcal{A}$ between concrete and abstract program states and then prove that it is a simulation.

Definition 3.4 $\mathcal{A}$ is a binary relation between concrete and abstract program states such that $(P, Q) \in \mathcal{A}$ if $P.a = Q.a$, $P.imr = Q.imr$, $P.\overline{I} \leq Q.\widehat{\overline{I}}$, and $P.\omega = Q.\omega$.

Lemma 3.1 **(Single-step Simulation)**

If $(P, Q) \in \mathcal{A}$ and $P \rightarrow P'$, then $\exists Q'$ such that $Q \hookrightarrow Q'$ and $(P', Q') \in \mathcal{A}$.

*Proof.* By case analysis of the rule used in $P \rightarrow P'$:

1. Rule (2.1).

   We have $P = \langle a, imr, \overline{I}, \omega, \sigma \rangle$ and $P' = \langle \overline{h}(u), \psi_u(imr), \overline{I}, \langle a, imr \rangle :: \omega, \sigma \rangle$ with $u \in \mathcal{L}(imr, \overline{I})$, and $Q = \langle a, imr, \widehat{\overline{I}}, \omega \rangle$ with $\overline{I} \leq \widehat{\overline{I}}$. Choose $Q' = \langle \overline{h}(u), \psi_u(imr), \widehat{\overline{I}}, \langle a, imr \rangle :: \omega \rangle$. We have $Q \hookrightarrow Q'$ from rule (3.1) and we have $(P', Q') \in \mathcal{A}$ from $\overline{I} \leq \widehat{\overline{I}}$.

2. Rule (2.2).

   We have $P = \langle \mathsf{iret}_v, imr, \overline{I}, \langle a, imr' \rangle :: \omega', \sigma \rangle$ and $P' = \langle a, imr', \overline{J}, \omega', \sigma \rangle$ with $\mathcal{L}(imr, \overline{I}) = \emptyset$ and $\overline{J} = \delta_v(\overline{I})$, and $Q = \langle \mathsf{iret}_v, imr, \widehat{\overline{K}}, \omega \rangle$ with $\overline{I} \leq \widehat{\overline{K}}$. Choose $Q' = \langle a, imr', \widehat{\overline{L}}, \omega' \rangle$ with $\widehat{\overline{L}} = \gamma_v(imr, \widehat{\overline{K}})$. Then, we have $Q \hookrightarrow Q'$ from rule (3.2). Next, we prove that $(P', Q') \in \mathcal{A}$, that is, $\forall u \in 1..N : \overline{J}(u) \leq \widehat{\overline{L}}(u)$. There are two subcases:

   (a) $(u = v)$.

   From $\overline{J} = \delta_v(\overline{I})$ and defn. (2.2), we have $\overline{J}(v) = \overline{I}(v) + t - r$ and $r \geq \overline{r}(v)$. From $\widehat{\overline{L}} = \gamma_v(\widehat{\overline{K}}, imr)$ and defn. (3.2), we have $\widehat{\overline{L}}(v) = \widehat{\overline{K}}(v) + t - \overline{r}(v)$. From $\overline{J}(v) = \overline{I}(v) + t - r$ and $\widehat{\overline{L}}(v) = \widehat{\overline{K}}(v) + t - \overline{r}(v)$ and $\overline{I}(v) \leq \widehat{\overline{K}}(v)$ and $r \geq \overline{r}(v)$, we have $\overline{J}(v) \leq \widehat{\overline{L}}(v)$.

(b) ($u \neq v$). There are two subcases:

i. $u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{K}})$.

From $u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{K}})$ and defn. (3.1), we have $imr(0) \wedge imr(u) = 1$. From $imr(0) \wedge imr(u) = 1$ and $\mathcal{L}(imr, \overline{I}) = \emptyset$ and defn. (2.1), we have $\overline{I}(u) < 0$. From $\overline{J} = \delta_v(\overline{I})$ and $u \neq v$ and defn. (2.2), we have $\overline{J}(u) = \overline{I}(u) + t$. From $\overline{J}(u) = \overline{I}(u) + t$ and $\overline{I}(u) < 0$, we have $\overline{J}(u) \leq t - 1$. From $\widehat{\overline{L}} = \gamma_v(imr, \widehat{\overline{K}})$ and $u \neq v$ and $u \in \widehat{\mathcal{L}}(imr, \widehat{\overline{K}})$ and defn. (3.2), we have $\widehat{\overline{L}}(u) = t - 1$. Finally, from $\overline{J}(u) \leq t - 1$ and $\widehat{\overline{L}}(u) = t - 1$, it follows that $\overline{J}(u) \leq \widehat{\overline{L}}(u)$.

ii. $u \notin \widehat{\mathcal{L}}(imr, \widehat{\overline{K}})$.

From $\widehat{\overline{L}} = \gamma_v(imr, \widehat{\overline{K}})$ and $u \neq v$ and $u \notin \widehat{\mathcal{L}}(imr, \widehat{\overline{K}})$ and defn. (3.2), we have $\widehat{\overline{L}}(u) = \widehat{\overline{K}}(u) + t$. From $\widehat{\overline{L}}(u) = \widehat{\overline{K}}(u) + t$ and $\overline{I}(u) \leq \widehat{\overline{K}}(u)$, we have $\widehat{\overline{L}}(u) \geq \overline{I}(u) + t$. From $\overline{J} = \delta_v(\overline{I})$ and $u \neq v$ and defn. (2.2), we have $\overline{J}(u) = \overline{I}(u) + t$. Finally, from $\overline{J}(u) = \overline{I}(u) + t$ and $\widehat{\overline{L}}(u) \geq \overline{I}(u) + t$, it follows that $\overline{J}(u) \leq \widehat{\overline{L}}(u)$.

3. Rules (2.3)–(2.5).

The proofs are similar to that in item (1) above.

4. Rule (2.6).

We have $P = \langle \mathsf{skip}; a, imr, \overline{I}, \omega, \sigma \rangle$ and $P' = \langle a, imr, \overline{J}, \omega, \sigma \rangle$ with $\mathcal{L}(imr, \overline{I}) = \emptyset$ and $\overline{J} = \delta(\overline{I})$, and $Q = \langle \mathsf{skip}; a, imr, \widehat{\overline{K}}, \omega \rangle$ with $\overline{I} \leq \widehat{\overline{K}}$. Choose $Q' = \langle a, imr, \widehat{\overline{L}}, \omega \rangle$ with $\widehat{\overline{L}} = \gamma(imr, \widehat{\overline{K}})$. We have $Q \hookrightarrow Q'$ from rule (3.6). The proof of $(P', Q') \in \mathcal{A}$ is similar to that in item 2(b) above.

5. Rules (2.7)–(2.9).

The proofs are similar to that in item (4) above. □

Lemma 3.2 **(Multi-step Simulation)**

If $(P, Q) \in \mathcal{A}$ and $P \rightarrow^n P'$, then $\exists Q'$ such that $Q \hookrightarrow^n Q'$ and $(P', Q') \in \mathcal{A}$.

*Proof.* By induction on $n$. The base case ($n = 0$) is trivial. To prove the induction step, suppose $(P, Q) \in \mathcal{A}$ and $P \rightarrow^n P' \rightarrow P''$. From $(P\,, Q\,) \in \mathcal{A}$ and $P \rightarrow^n P'$ and the induction hypothesis, we have $\exists Q'$ such that $Q \hookrightarrow^n Q'$ and $(P', Q') \in \mathcal{A}$. From $(P', Q') \in \mathcal{A}$ and $P' \rightarrow P''$ and lemma (3.1), we have $\exists Q''$ such that $Q' \hookrightarrow Q''$ and $(P'', Q'') \in \mathcal{A}$. □

Finally, the following corollary proves the soundness of model checking.

Corollary 3.1 **(Soundness of Model Checking)**

If $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ then $(P_0 \rightarrow^n P \Rightarrow P.\overline{I} < \overline{d})$.

*Proof.* From $P_0 \in \{\ \langle m_0, \neg \mathsf{t}_0, \overline{I}, \mathsf{nil}, \lambda x.0\rangle \mid \overline{I} \leq \overline{0}\ \}$ and $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil}\rangle$ and defn. (3.4), we have $(P_0, Q_0) \in \mathcal{A}$. From $(P_0, Q_0) \in \mathcal{A}$ and $P_0 \rightarrow^n P$ and lemma (3.2), we have $\exists Q$ such that $Q_0 \hookrightarrow^n Q$ and $(P, Q) \in \mathcal{A}$. From $(P, Q) \in \mathcal{A}$ and defn. (3.4), we have $P.\overline{I} \leq Q.\widehat{\overline{I}}$. From $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ and $Q_0 \hookrightarrow^n Q$, we have $Q.\widehat{\overline{I}} < \overline{d}$. From $P.\overline{I} \leq Q.\widehat{\overline{I}}$ and $Q.\widehat{\overline{I}} < \overline{d}$, we have $P.\overline{I} < \overline{d}$. □

# 4. THE TYPE SYSTEM

## 4.1 Syntax

The syntax of types is shown in Figure (4.1). We use imr values $imr$ and abstract latency vector values $\widehat{\overline{I}}$ as types. An imr value $imr$ is a singleton type, that is, it is the type of the single value $imr$ of imr, while an abstract latency vector value $\widehat{\overline{I}}$ is the type of any latency vector value $\overline{I}$ such that $\overline{I} \leq \widehat{\overline{I}}$. We use $\alpha$ to range over pairs of the form $(imr, \widehat{\overline{I}})$.

$$
\begin{array}{lrcl}
\text{(type of } s\text{)} & S & ::= & \bigwedge_{i \in A}(\alpha_i \longrightarrow \bigvee_{j \in B_i} \alpha_j) \\
\text{(type of } m\text{)} & M & ::= & \bigvee_{i \in A} \alpha_i \\
\text{(type of } h\text{)} & H & ::= & \bigwedge_{i \in A}(\alpha_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \\
\text{(handler type)} & \tau & ::= & \bigwedge_{i \in A}(\widehat{\overline{I}}_i \stackrel{imr_i}{\longrightarrow} \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \\
 & \alpha & ::= & (imr, \widehat{\overline{I}})
\end{array}
$$

Figure 4.1 Syntax of types

The effect of a statement $s$ on the imr and the abstract latency vector is described by the intersection type $\bigwedge_{i \in A}(\alpha_i \longrightarrow \bigvee_{j \in B_i} \alpha_j)$ which states that it is safe to begin executing $s$ in one of contexts $\{\alpha_i \,|\, i \in A\}$ and that if $s$ begins executing in context $\alpha_i$, then it will finish executing in one of contexts $\{\, \alpha_j \,|\, j \in B_i \,\}$. A main part $m$ has the union type $\bigvee_{i \in A} \alpha_i$ which states that it is safe to begin executing $m$ in one of contexts $\{\alpha_i \,|\, i \in A\}$. The type does not describe the contexts in which $m$ will finish executing since the main procedure never returns; it loops indefinitely. A handler part $h$ has the

intersection type $\bigwedge_{i\in A}(\alpha_i \longrightarrow \bigvee_{j\in B_i}\widehat{\overline{I}}_j)$ which is similar to the type of a statement except that it does not describe the effect of $h$ on the imr. This is because handlers do not have any externally observable effect on the imr since the imr is pushed onto the stack when a handler is invoked and popped when it returns, thereby canceling any effect the handler might have had on the imr. Handler $\overline{h}(u)$ has the intersection type $\bigwedge_{i\in A}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j\in B_i}\widehat{\overline{I}}_j)$ which states that it is safe to begin executing $\overline{h}(u)$ in one of contexts $\{\,(imr_i, \widehat{\overline{I}}_i)\,|\,i \in A\,\}$ and that if $\overline{h}(u)$ begins executing in context $(imr_i, \widehat{\overline{I}}_i)$, then it will finish executing in one of contexts $\{\ (imr_i, \widehat{\overline{I}}_j)\ |\ j\ \in\ B_i\ \}$. We use $\overline{\tau}$ to denote handler types such that $\overline{\tau}(u)$ is the type of $\overline{h}(u)$. We require all intersection and union types to be non-empty:

Definition 4.1 **(Well-formedness of Types)**

1. If $S = \bigwedge_{i\in A}(\alpha_i \longrightarrow \bigvee_{j\in B_i}\alpha_j)$ then $|A| \geq 1$ and $\forall i \in A:\ |B_i| \geq 1$.
2. If $M = \bigvee_{i\in A}\alpha_i$ then $|A| \geq 1$.
3. If $H = \bigwedge_{i\in A}(\alpha_i \longrightarrow \bigvee_{j\in B_i}\widehat{\overline{I}}_j)$ then $|A| \geq 1$ and $\forall i \in A:\ |B_i| \geq 1$.
4. If $\tau = \bigwedge_{i\in A}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j\in B_i}\widehat{\overline{I}}_j)$ then $|A| \geq 1$ and $\forall i \in A:\ |B_i| \geq 1$.

## 4.2 Type Rules

We use the following forms of type judgments:

| Type Judgment | Meaning |
|---|---|
| $\overline{\tau} \vdash \overline{h}(u)\ :\ \overline{\tau}(u)$ | handler $\overline{h}(u)$ has type $\overline{\tau}(u)$ |
| $\overline{\tau}\ \vdash\ s\ :\ S$ | $s$ type checks |
| $\overline{\tau}, M \vdash\ m$ | $m$ type checks |
| $\overline{\tau}\ \vdash\ h\ :\ H$ | $h$ type checks |
| $\overline{\tau}, \widehat{\overline{I}} \vdash\ \omega$ | stack $\omega$ type checks |
| $\overline{\tau}, \widehat{\overline{I}} \vdash\ P$ | program state $P$ type checks |

All judgments have a typing environment $\overline{\tau}$ that specifies the types of all handlers. The judgment $\overline{\tau} \vdash \overline{h}(u) : \overline{\tau}(u)$ states that handler $\overline{h}(u)$ has type $\overline{\tau}(u)$ in environment

$\overline{\tau}$. We use $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ as shorthand for the family of judgments $\overline{\tau} \vdash \overline{h}(u) : \overline{\tau}(u)$. Judgments $\overline{\tau} \vdash s : S$ and $\overline{\tau}, M \vdash m$ and $\overline{\tau} \vdash h : H$ state that statement $s$, main part $m$, and handler part $h$ have types $S$, $M$, and $H$, respectively, in environment $\overline{\tau}$. The judgment $\overline{\tau}, \widehat{\overline{I}} \vdash \omega$ states that the stack $\omega$ type checks in environment $\overline{\tau}$ in a context in which the abstract latency vector is $\widehat{\overline{I}}$; judgment $\overline{\tau}, \widehat{\overline{I}} \vdash P$ states the same for program state $P$. The type rules are as follows (The side-conditions of rules (4.1) and (4.10) are elaborated in defn. (4.2)):

$$\frac{\begin{array}{c} s \text{ is of the form } \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr,\ \text{ or } x := e \\ \overline{\tau} \vdash s\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \\ \left[\begin{array}{l} \forall i \in A :\ \forall j \in B_i :\ imr_j = \chi_s(imr_i) \text{ and} \\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \text{ is safe w.r.t. } \overline{\tau} \end{array}\right] \end{array}}{} \tag{4.1}$$

$$\frac{\begin{array}{c} \overline{\tau} \vdash s_1\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^t} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash s_2\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^f} imr_j, \widehat{\overline{I}}_j) \end{array}}{\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j)} \tag{4.2}$$

$$\frac{\begin{array}{c} \overline{\tau} \vdash s_1\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash s_2\ :\ \bigwedge_{i \in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i'} imr_j, \widehat{\overline{I}}_j) \end{array}}{\overline{\tau} \vdash s_1; s_2\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k \in \bigcup\{B_j' \mid j \in B_i\}} imr_k, \widehat{\overline{I}}_k)} \quad [\forall i \in A : B_i \subseteq A'] \tag{4.3}$$

$$\frac{\overline{\tau} \vdash \overline{h}\ :\ \overline{\tau} \qquad \overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle m, imr_{i'}, \overline{I}, \mathsf{nil}, \sigma \rangle} \quad \left[\overline{I} \leq \widehat{\overline{I}}_{i'} \text{ and } i' \in A\right] \tag{4.4}$$

$$\frac{\begin{array}{c} \overline{\tau} \vdash \overline{h}\ :\ \overline{\tau} \qquad \overline{\tau} \vdash h\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \\ \forall j \in B_{i'}\ :\ \overline{\tau}, \widehat{\overline{I}}_j \vdash \omega \end{array}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle h, imr_{i'}, \overline{I}, \omega, \sigma \rangle} \quad \left[\overline{I} \leq \widehat{\overline{I}}_{i'} \text{ and } i' \in A\right] \tag{4.5}$$

$$\frac{\overline{\tau}, \bigvee_{i\in A} imr_i, \widehat{\overline{I}}_i \ \vdash \ m}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle m, imr_{i'}\rangle :: \mathsf{nil}} \qquad [i' \in A] \tag{4.6}$$

$$\frac{\overline{\tau} \vdash h \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \qquad \forall j \in B_{i'} : \ \overline{\tau}, \widehat{\overline{I}}_j \vdash \omega}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle h, imr_{i'}\rangle :: \omega} \qquad [i' \in A] \tag{4.7}$$

$$\frac{\overline{\tau} \vdash s \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j)}{\overline{\tau}, \bigvee_{i\in A} imr_i, \widehat{\overline{I}}_i \vdash \mathsf{loop}\ s} \qquad [\forall i \in A : \ B_i \subseteq A] \tag{4.8}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash s \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau}, \bigvee_{i\in A'} imr_i, \widehat{\overline{I}}_i \vdash m\end{array}}{\overline{\tau}, \bigvee_{i\in A} imr_i, \widehat{\overline{I}}_i \vdash s; m} \qquad [\forall i \in A : \ B_i \subseteq A'] \tag{4.9}$$

$$\begin{array}{c}\overline{\tau} \vdash \mathsf{iret}_v \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \\ \left[\bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \text{ is safe w.r.t. } \overline{\tau} \text{ and } v\right]\end{array} \tag{4.10}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash s \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash h \ : \ \bigwedge_{i\in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B'_i} \widehat{\overline{I}}_j)\end{array}}{\overline{\tau} \vdash s; h \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in \bigcup\{B'_j \,|\, j\in B_i\}} \widehat{\overline{I}}_k)} \qquad [\forall i \in A : \ B_i \subseteq A'] \tag{4.11}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash \overline{h}(u) \ : \ \bigwedge_{i\in A}(\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \\ \forall i \in A : \ imr_i(0) = imr_i(u) = 1\end{array}}{\overline{\tau} \vdash \overline{h}(u) \ : \ \bigwedge_{i\in A}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j\in B_i} \widehat{\overline{I}}_j)} \tag{4.12}$$

Definition 4.2 **(Safety Condition)**

(primitive statements)
Let $\forall u \in 1..N : \overline{\tau}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$. Then $S = \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}$ if $\forall i \in A$:

1. $\forall u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) : \begin{cases} (a)\ \ i \in A^u \\ (b)\ \ \forall k \in B_i^u : \begin{cases} i.\ \ k \in A \text{ and } imr_k = imr_i \\ ii.\ \ B_k \subseteq B_i \end{cases} \end{cases}$
2. $\exists j \in B_i : \widehat{\overline{I}}_j = \gamma(imr_i, \widehat{\overline{I}}_i) \text{ and } \widehat{\overline{I}}_j < \overline{d}$

($\mathsf{iret}_v$)
Let $\forall u \in 1..N : \overline{\tau}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$. Then $H = \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}$ and $v$ if $\forall i \in A$:

3. $\forall u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) : \begin{cases} (a)\ \ i \in A^u \\ (b)\ \ \forall k \in B_i^u : \begin{cases} i.\ \ k \in A \text{ and } imr_k = imr_i \\ ii.\ \ B_k \subseteq B_i \end{cases} \end{cases}$
4. $\exists j \in B_i : \widehat{\overline{I}}_j = \gamma_v(imr_i, \widehat{\overline{I}}_i) \text{ and } \widehat{\overline{I}}_j < \overline{d}$

Rule (4.1) type checks primitive statements. The rule has two side-conditions on the type of a primitive statement $s$. The first side-condition requires the type to describe the effect of executing $s$ on the $\mathsf{imr}$ in the manner prescribed by the $\mathsf{imr}$ transfer function in defn. (2.3). The second side-condition requires the type to describe the following:

1. The effect, of executing (possibly several) handlers before executing $s$, on the abstract latency vector in accordance with condition (1) in defn. (4.2). Recall that it is not necessary to check that the type correctly describes the effect of executing handlers on the $\mathsf{imr}$ since the handlers have no externally observable effect on the $\mathsf{imr}$.

2. The effect of executing $s$ on the abstract latency vector in accordance with condition (2) in defn. (4.2).

We expand upon conditions (1) and (2) in defn. (4.2). Suppose $s$ begins executing in context $(imr_i, \widehat{\overline{I}}_i)$ indexed by $i \in A$. There are two cases depending upon the set of devices that are apparently latent in this context:
Case 1: If device $u$ is apparently latent, then handler $\overline{h}(u)$ might be invoked, and condition (1) ensures the following:

(a) It is safe to begin executing $\overline{h}(u)$ in the context $(imr_i, \widehat{\overline{I}}_i)$, that is, $i \in A^u$.

(b) If $\overline{h}(u)$ finishes executing in context $(imr_i, \widehat{\overline{I}}_k)$, that is, if $k \in B_i^u$, then:

   i. It is safe to resume executing $s$ in context $(imr_i, \widehat{\overline{I}}_k)$, that is, $k \in A$ and $imr_k = imr_i$.

   ii. Each possible context in which $s$ finishes executing after resuming execution in the context $(imr_k, \widehat{\overline{I}}_k)$ is also a possible context in which $s$ finishes executing after beginning execution in the context $(imr_i, \widehat{\overline{I}}_i)$, that is, $B_k \subseteq B_i$.

Case 2: If no device is apparently latent or if none of the apparently latent devices is veritably latent, then $s$ will be executed, and condition (2) ensures that the effect of $s$ on the abstract latency vector is described in the manner prescribed by the abstract latency vector transfer function in defn. (3.2). Additionally, condition (2) checks the property that all deadlines are met.

Rule (4.2) type checks branching statements. Rule (4.3) type checks sequentially composed statements. The side-condition of rule (4.3) states that if $s_1$ finishes executing in a certain context, then it should be safe to begin executing $s_2$ in that context. Notice that it does not require that if $s_2$ begins executing in a certain context, then $s_1$ should finish executing in that context. This is because an interrupt handler might be invoked at the program point between $s_1$ and $s_2$ in a context $(imr, \widehat{\overline{I}})$ in which $s_1$ finished executing, and return to that program point in a different context $(imr, \widehat{\overline{J}})$. As

a result, $s_2$ will begin executing in context $(imr, \widehat{\overline{J}})$ even though $s_1$ finished executing in context $(imr, \widehat{\overline{I}})$.

Rules (4.4) and (4.5) type check program states while rules (4.6) and (4.7) type check the stack. Rules (4.8) and (4.9) type check main parts $\mathsf{loop}\ s$ and $s; m$. Rules (4.10) and (4.11) type check handler parts $\mathsf{iret}$ and $s; h$. Finally, rule (4.12) lifts the type of the handler part $\overline{h}(u)$ to the handler type $\overline{\tau}(u)$.

## 4.3 From Type Checking to Model Checking

In the rest of this chapter, we prove lemma (4.6) which states that if a program type checks, then the model checker accepts it. The following three lemmas, namely, lemmas (4.1), (4.2), and (4.3), set the stage for proving lemma (4.4) which shows type preservation with respect to the abstract semantics and is the key lemma in proving lemma (4.6).

Lemma 4.1 **(Interrupt, Statement)**

Let $\forall u \in 1..N : \overline{\tau}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$. If $\overline{\tau} \vdash s : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$, then:

$(a)$ $i' \in A^u$

$(b)$ $\forall k \in B_{i'}^u :$ $\begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k \subseteq B_{i'} \end{cases}$

*Proof.* By induction on the structure of the derivation of $\overline{\tau} \vdash s : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$. There are 3 cases depending upon which one of rules (4.1), (4.2), and (4.3) was used last in the derivation:

- Rule (4.1). We have:

$$s \text{ is of the form } \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr,\ \text{or } x := e$$
$$\overline{\tau} \vdash s\ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$$
$$\left[\begin{array}{l} \forall i \in A :\ \forall j \in B_i :\ imr_j = \chi_s(imr_i) \text{ and} \\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \text{ is safe w.r.t. } \overline{\tau} \end{array}\right]$$

Then, the lemma follows from $\bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and item (1) of defn. (4.2).

- Rule (4.2). We have:

$$\frac{\begin{array}{c}\overline{\tau} \vdash s_1 \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash s_2 \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^f} imr_j, \widehat{\overline{I}}_j)\end{array}}{\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j)}$$

To prove:

$(a)$ $i' \in A^u$

$(b)$ $\forall k \in B_{i'}^u : \begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k^t \cup B_k^f \subseteq B_{i'}^t \cup B_{i'}^f \end{cases}$

From $\overline{\tau} \vdash s_1 : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$(c)$ $i' \in A^u$

$(d)$ $\forall k \in B_{i'}^u : \begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k^t \subseteq B_{i'}^t \end{cases}$

From $\overline{\tau} \vdash s_2 : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^f} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$(e)$ $i' \in A^u$

$(f)$ $\forall k \in B_{i'}^u : \begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k^f \subseteq B_{i'}^f \end{cases}$

Then, $(a)$ follows from $(c)$ or $(e)$, $(b)$ $(i)$ follows from $(d)$ $(i)$ or $(f)$ $(i)$, and $(b)$ $(ii)$ follows from $(d)$ $(ii)$ and $(f)$ $(ii)$.

- Rule (4.3). We have:

$$\frac{\begin{array}{c}\overline{\tau} \vdash s_1 \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash s_2 \ : \ \bigwedge_{i\in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i'} imr_j, \widehat{\overline{I}}_j)\end{array}}{\overline{\tau} \vdash s_1; s_2 \ : \ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in \bigcup\{B_j' | j\in B_i\}} imr_k, \widehat{\overline{I}}_k)} \qquad [\forall i \in A : B_i \subseteq A']$$

To prove:

$(a)$ $i' \in A^u$

$(b)$ $\forall k \in B^u_{i'}$ : $\begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & \bigcup \{ B'_j \mid j \in B_k \} \subseteq \bigcup \{ B'_j \mid j \in B_{i'} \} \end{cases}$

From $\overline{\tau} \vdash s_1 : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$(c)$ $i' \in A^u$

$(d)$ $\forall k \in B^u_{i'}$ : $\begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k \subseteq B_{i'} \end{cases}$

Then, $(a)$ follows from $(c)$, $(b)$ $(i)$ follows from $(d)$ $(i)$, and $(b)$ $(ii)$ follows from $(d)$ $(ii)$. □

Lemma 4.2 **(Interrupt, Main part)**

Let $\forall u \in 1..N : \overline{\tau}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B^u_i} \widehat{\overline{I}}_j)$. If $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$, then:

$(a)$ $i' \in A^u$

$(b)$ $\forall k \in B^u_{i'} :$ $k \in A \text{ and } imr_k = imr_{i'}$

*Proof.* By case analysis of the rule that was used last in the derivation of $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m$:

- Rule (4.8). We have:

$$\frac{\overline{\tau} \vdash s \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)}{\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash \mathsf{loop}\ s}$$

  The lemma follows from $\overline{\tau} \vdash s : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and lemma (4.1).

- Rule (4.9). We have:

$$\frac{\overline{\tau} \vdash s \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \qquad \overline{\tau}, \bigvee_{i \in A'} imr_i, \widehat{\overline{I}}_i \vdash m}{\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash s; m}$$

The lemma follows from $\overline{\tau} \vdash s : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and lemma (4.1). □

Lemma 4.3 **(Interrupt, Handler part)**

Let $\forall u \in 1..N : \overline{\tau}(u) = \bigwedge_{i\in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j\in B_i^u} \widehat{\overline{I}}_j)$. If $\overline{\tau} \vdash h : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$, then:

(a) $i' \in A^u$

(b) $\forall k \in B_{i'}^u :$ $\begin{cases} i. \;\; k \in A \text{ and } imr_k = imr_{i'} \\ ii. \;\; B_k \subseteq B_{i'} \end{cases}$

*Proof.* By case analysis of the rule that is used last in the derivation of $\overline{\tau} \vdash h : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j)$:

- Rule (4.10). We have:

$$\frac{\overline{\tau} \vdash \mathsf{iret}_v \;:\; \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \quad \left[\bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j) \text{ is safe w.r.t. } v, \overline{\tau}\right]}{}$$

  The lemma follows from $\bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} \widehat{\overline{I}}_j)$ is safe w.r.t. $v, \overline{\tau}$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and item (3) of defn. (4.2).

- Rule (4.11). We have:

$$\frac{\overline{\tau} \vdash s \;:\; \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j) \quad \overline{\tau} \vdash h \;:\; \bigwedge_{i\in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B'_i} \widehat{\overline{I}}_j)}{\overline{\tau} \vdash s; h \;:\; \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in \bigcup\{B'_j \mid j\in B_i\}} \widehat{\overline{I}}_k)}$$

  To prove:

  (a) $i' \in A^u$

  (b) $\forall k \in B_{i'}^u :$ $\begin{cases} i. \;\; k \in A \text{ and } imr_k = imr_{i'} \\ ii. \;\; \bigcup \{ B'_j \mid j \in B_k \} \subseteq \bigcup \{ B'_j \mid j \in B_{i'} \} \end{cases}$

  From $\overline{\tau} \vdash s : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and lemma (4.1), we have:

$(c)$ $i' \in A^u$

$(d)$ $\forall k \in B^u_{i'}$ : $\begin{cases} i. & k \in A \text{ and } imr_k = imr_{i'} \\ ii. & B_k \subseteq B_{i'} \end{cases}$

Then, $(a)$ follows from $(c)$, $(b)$ $(i)$ follows from $(d)$ $(i)$, and $(b)$ $(ii)$ follows from $(d)$ $(ii)$. □

Every abstract program state in the reachable state-space is *consistent* in the sense of following definition.

Definition 4.3 **(Consistent Abstract Program State)**

An abstract program state $\langle a, imr, \widehat{\overline{I}}, \omega \rangle$ is consistent if either (1) $\omega = \mathsf{nil}$ and $a = m$ or (2) $\omega = \langle h^k, imr^k \rangle :: \ldots :: \langle h^1, imr^1 \rangle :: \langle m, imr \rangle :: \mathsf{nil}$ and $a = h$, for $0 \leq k < N$ where $k = 0$ means $\omega = \langle m, imr \rangle :: \mathsf{nil}$.

It is straightforward to show that consistency is preserved by the abstract semantics.

Proposition 4.1 If $Q$ is consistent and $Q \hookrightarrow^n Q'$ then $Q'$ is consistent.

We are now ready to prove type preservation with respect to the abstract semantics.

Lemma 4.4 **(Single-step Type Preservation)**

If $Q$ is consistent and $Q.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q.\widehat{\overline{I}} \vdash \langle Q.a, Q.imr, Q.\widehat{\overline{I}}, Q.\omega, \sigma \rangle$ and $Q \hookrightarrow Q'$, then $Q'.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q'.\widehat{\overline{I}} \vdash \langle Q'.a, Q'.imr, Q'.\widehat{\overline{I}}, Q'.\omega, \sigma \rangle$.

*Proof.* By case analysis of the rule used in $Q \hookrightarrow Q'$. There are 9 cases depending upon which one of rules (3.1)–(3.9) was used:

- Rule (3.1).

  We have $Q = \langle a, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega \rangle$ and $Q' = \langle \overline{h}(u), \psi_u(imr_{i'}), \widehat{\overline{I}}_{i'}, \langle a, imr_{i'} \rangle :: \omega \rangle$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and $Q.\widehat{\overline{I}} < \overline{d}$. To prove:

  1. $Q'.\widehat{\overline{I}} < \overline{d}$
  2. $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \overline{h}(u), \psi_u(imr_{i'}), \widehat{\overline{I}}_{i'}, \langle a, imr_{i'} \rangle :: \omega, \sigma \rangle$.

From $Q.\widehat{\overline{I}} < \overline{d}$ and $Q.\widehat{\overline{I}} = Q'.\widehat{\overline{I}}$, we have $Q'.\widehat{\overline{I}} < \overline{d}$, which proves (1). We next prove (2). Since $Q$ is consistent, there are two subcases:

Subcase 1: $a = m$ and $\omega = \mathsf{nil}$.

We have:

$(a)\ \forall u \in 1..N :\ \overline{\tau}(u) = \bigwedge_{i \in A^u} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$

From rule (4.4), we have:

$$\frac{\overline{\tau} \vdash \overline{h} \ :\ \overline{\tau} \qquad \overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle m, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma \rangle} \qquad [i' \in A]$$

From $\overline{\tau} \vdash \overline{h} \ :\ \overline{\tau}$ and $(a)$ and rule (4.12), we have:

$(b)\ \overline{\tau} \vdash \overline{h}(u) \ :\ \bigwedge_{i \in A^u} (\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$

From $(a)$ and $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and lemma (4.2), we have:

$(c)\ i' \in A^u$

$(d)\ \forall k \in B_{i'}^u :\ k \in A \ and \ imr_k = imr_{i'}$

From $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash m$ and $(d)$ and rule (4.6), we have:

$(e)\ \forall k \in B_{i'}^u :\ \overline{\tau}, \widehat{\overline{I}}_k \vdash \langle m, imr_{i'} \rangle :: \mathsf{nil}$

From $\overline{\tau} \vdash \overline{h} \ :\ \overline{\tau}$ and $(b)$ and $(c)$ and $(e)$ and rule (4.5), we have $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \overline{h}(u), \psi_u(imr_{i'}), \widehat{\overline{I}}_{i'}, \langle m, imr_{i'} \rangle :: \mathsf{nil}, \sigma \rangle$, which proves (2).

Subcase 2: $a = h$.

We have:

$(a)\ \forall u \in 1..N :\ \overline{\tau}(u) = \bigwedge_{i \in A^u} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$

From rule (4.5), we have:

$$\frac{\begin{array}{c}\overline{\tau} \vdash \overline{h} \ :\ \overline{\tau} \qquad \overline{\tau} \vdash h \ :\ \bigwedge_{i \in A} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \\ \forall j \in B_{i'} :\ \overline{\tau}, \widehat{\overline{I}}_j \vdash \omega \end{array}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle h, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega, \sigma \rangle} \qquad [i' \in A]$$

From $\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau}$ and $(a)$ and rule (4.12), we have:

$(b)\ \overline{\tau} \vdash \overline{h}(u) \ : \ \bigwedge_{i \in A^u}(\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$

From $(b)$ and $\overline{\tau} \vdash h : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and $i' \in A$ and $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$ and lemma (4.3), we have:

$(c)\ i' \in A^u$

$(d)\ \forall k \in B_{i'}^u : \begin{cases} i.\ k \in A \ and \ imr_k = imr_{i'} \\ ii.\ B_k \subseteq B_{i'} \end{cases}$

From $\forall j \in B_{i'} : \ \overline{\tau}, \widehat{\overline{I}}_j \vdash \omega$ and $(d)\ (ii)$, we have:

$(e)\ \forall k \in B_{i'}^u : \ \forall j \in B_k : \ \overline{\tau}, \widehat{\overline{I}}_j \vdash \omega$

From $\overline{\tau} \vdash h \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and $(d)\ (i)$ and $(e)$ and rule (4.7), we have:

$(f)\ \forall k \in B_{i'}^u : \ \overline{\tau}, \widehat{\overline{I}}_k \vdash \langle h, imr_{i'} \rangle :: \omega$

From $\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau}$ and $(b)$ and $(c)$ and $(f)$ and rule (4.5), we have $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \overline{h}(u), \psi_u(imr_{i'}), \widehat{\overline{I}}_{i'}, \langle h, imr_{i'} \rangle :: \omega, \sigma \rangle$, which proves (2).

- Rule (3.2).

  We have $Q = \langle \mathsf{iret}_v, imr_{i'}, \widehat{\overline{I}}_{i'}, \langle a, imr_r \rangle :: \omega' \rangle$ and $Q' = \langle a, imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega' \rangle$ and $Q.\widehat{\overline{I}} < \overline{d}$. To prove:

  1. $Q'.\widehat{\overline{I}} < \overline{d}$
  2. $\overline{\tau}, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle a, imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega', \sigma \rangle$

  Since $Q$ is consistent, there are two subcases:

  Subcase 1: $a = m$ and $\omega' = \mathsf{nil}$.

  From rules (4.5) and (4.6), we have:

$$\frac{\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau} \qquad \overline{\tau} \vdash \mathsf{iret}_v \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \qquad \forall j \in B_{i'} : \ \dfrac{\overline{\tau}, \bigvee_{k \in C} imr_k, \widehat{\overline{I}}_k \ \vdash \ m}{\overline{\tau}, \widehat{\overline{I}}_j \vdash \langle m, imr_r \rangle :: \mathsf{nil}}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \mathsf{iret}_v, imr_{i'}, \widehat{\overline{I}}_{i'}, \langle m, imr_r \rangle :: \mathsf{nil}, \sigma \rangle}$$

where:

(a) $i' \in A$

(b) $\bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}, v$

(c) $\forall j \in B_{i'} : \exists k \in C : (imr_r, \widehat{\overline{I}}_j) = (imr_k, \widehat{\overline{I}}_k)$

From (a) and (b), $\exists j'$ such that (see item (4) of defn. (4.2)):

(d) $j' \in B_{i'}$

(e) $\widehat{\overline{I}}_{j'} = \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'})$

(f) $\widehat{\overline{I}}_{j'} < \overline{d}$

From $(e)$ and $(f)$, we have $\gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) < \overline{d}$, which proves (1). From $(c)$–$(e)$, $\exists k'$ such that:

(g) $k' \in C$ and $(imr_{k'}, \widehat{\overline{I}}_{k'}) = (imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}))$

Finally, from $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and $\overline{\tau}, \bigvee_{k \in C} imr_k, \widehat{\overline{I}}_k \vdash m$ and $(g)$ and rule (4.4), we have $\overline{\tau}, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle m, imr_r, \gamma_v(imr_r, \widehat{\overline{I}}_{i'}), \mathsf{nil}, \sigma \rangle$, which proves (2).

Subcase 2: $a = h$.

From rules (4.5) and (4.7), we have:

$$\forall k' \in C' : \frac{\overline{\tau} \vdash h \ : \ \bigwedge_{k \in C}(imr_k, \widehat{\overline{I}}_k \longrightarrow \bigvee_{l \in D_k} \widehat{\overline{I}}_l) \qquad \forall l \in D_{k'} : \ \overline{\tau}, \widehat{\overline{I}}_l \vdash \omega'}{\overline{\tau}, \widehat{\overline{I}}_{k'} \vdash \langle h, imr_{k'} \rangle :: \omega'}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau} \qquad \overline{\tau} \vdash \mathsf{iret}_v \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \\ \forall j \in B_{i'} : \ \overline{\tau}, \widehat{\overline{I}}_j \vdash \langle h, imr_r \rangle :: \omega'\end{array}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \mathsf{iret}_v, imr_{i'}, \widehat{\overline{I}}_{i'}, \langle h, imr_r \rangle :: \omega', \sigma \rangle}$$

where:

(a) $i' \in A$

(b) $\bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}, v$

(c) $\bigcup_{j \in B_{i'}} \{ (imr_r, \widehat{\overline{I}}_j) \} = \bigcup_{k' \in C'} \{ (imr_{k'}, \widehat{\overline{I}}_{k'}) \}$

$(d)$ $C' \subseteq C$

From $(a)$ and $(b)$, $\exists j'$ such that (see item (4) of defn. (4.2)):

$(e)$ $j' \in B_{i'}$ and

$(f)$ $\widehat{\overline{I}}_{j'} = \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'})$ and

$(g)$ $\widehat{\overline{I}}_{j'} < \overline{d}$

From $(f)$ and $(g)$, we have $\gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) < \overline{d}$, which proves (1). From $(c)$–$(f)$, $\exists k''$ such that:

$(h)$ $k'' \in C'$ and

$(i)$ $(imr_{k''}, \widehat{\overline{I}}_{k''}) = (imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}))$

From $(d)$ and $(h)$ and $(i)$, we have:

$(j)$ $k'' \in C$ and $(imr_{k''}, \widehat{\overline{I}}_{k''}) = (imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}))$

From $\forall k' \in C' : \forall l \in D_{k'} : \overline{\tau}, \widehat{\overline{I}}_l \vdash \omega'$ and $(h)$, we have:

$(k)$ $\forall l \in D_{k''} : \overline{\tau}, \widehat{\overline{I}}_l \vdash \omega'$.

Finally, from $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and $\overline{\tau} \vdash h : \bigwedge_{k \in C}(imr_k, \widehat{\overline{I}}_k \longrightarrow \bigvee_{l \in D_k} \widehat{\overline{I}}_l)$ and $(j)$ and $(k)$ and rule (4.5), we have $\overline{\tau}, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle h, imr_r, \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega', \sigma \rangle$, which proves (2).

- Rule (3.3).

  We have $Q = \langle \mathsf{loop}\ s, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil} \rangle$ and $Q' = \langle s; \mathsf{loop}\ s, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil} \rangle$ and $Q.\widehat{\overline{I}} < \overline{d}$. To prove:

  1. $Q'.\widehat{\overline{I}} < \overline{d}$
  2. $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle s; \mathsf{loop}\ s, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma \rangle$

  From $Q.\widehat{\overline{I}} < \overline{d}$ and $Q.\widehat{\overline{I}} = Q'.\widehat{\overline{I}}$, we have $Q'.\widehat{\overline{I}} < \overline{d}$, which proves (1). From rules (4.4) and (4.8), we have:

$$\frac{\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau} \quad \dfrac{\overline{\tau} \vdash s \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)}{\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash \mathsf{loop}\ s}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \mathsf{loop}\ s, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma \rangle}$$

where:

($a$) $i' \in A$

($b$) $\forall i \in A : \ B_i \subseteq A$

From $\overline{\tau} \vdash s : \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ and $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash \mathsf{loop}\ s$ and ($b$) and rule (4.9), we have:

($c$) $\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash s; \mathsf{loop}\ s$

From $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and ($c$) and ($a$) and rule (4.4), we have $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle s; \mathsf{loop}\ s, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma \rangle$, which proves (2).

- Rule (3.4).

  We have $Q = \langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); a, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega \rangle$ and $Q' = \langle s_1; a, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega \rangle$ and $Q.\widehat{\overline{I}} < \overline{d}$. To prove:

  1. $Q'.\widehat{\overline{I}} < \overline{d}$
  2. $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle s_1; a, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega, \sigma \rangle$

  From $Q.\widehat{\overline{I}} < \overline{d}$ and $Q.\widehat{\overline{I}} = Q'.\widehat{\overline{I}}$, we have $Q'.\widehat{\overline{I}} < \overline{d}$, which proves (1). We next prove (2). Since $Q$ is consistent, there are two subcases:

  Subcase 1: $a = m$ and $\omega = \mathsf{nil}$.

  From rules (4.4), (4.9), and (4.2), we have:

$$\frac{\begin{array}{c}\overline{\tau} \vdash s_1 \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^t} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash s_2 \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^f} imr_j, \widehat{\overline{I}}_j)\end{array}}{\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j)}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \ : \ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau}, \bigvee_{i \in A'} imr_i, \widehat{\overline{I}}_i \vdash m\end{array}}{\overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); m}$$

$$\frac{\overline{\tau} \vdash \overline{h} \ : \ \overline{\tau} \qquad \overline{\tau}, \bigvee_{i \in A} imr_i, \widehat{\overline{I}}_i \vdash (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); m}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); m, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma \rangle}$$

where:

$(a)$ $i' \in A$

$(b)$ $\forall i \in A:\ B_i^t \subseteq A'$

$(c)$ $\forall i \in A:\ B_i^f \subseteq A'$

From $\overline{\tau} \vdash s_1 : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t} imr_j, \widehat{\overline{I}}_j)$ and $\overline{\tau}, \bigvee_{i\in A'} imr_i, \widehat{\overline{I}}_i \vdash m$ and $(b)$ and rule (4.9), we have:

$(d)$ $\overline{\tau}, \bigvee_{i\in A} imr_i, \widehat{\overline{I}}_i \vdash s_1; m$

From $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and $(d)$ and $(a)$ and rule (4.4), we have $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle s_1; m, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma\rangle$, which proves (2).

Subcase 2: $a = h$.

From rules (4.5), (4.11), and (4.2), we have:

$$\frac{\overline{\tau} \vdash s_1 \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t} imr_j, \widehat{\overline{I}}_j) \qquad \overline{\tau} \vdash s_2 \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^f} imr_j, \widehat{\overline{I}}_j)}{\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j)}$$

$$\frac{\overline{\tau} \vdash \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t \cup B_i^f} imr_j, \widehat{\overline{I}}_j) \qquad \overline{\tau} \vdash h \ :\ \bigwedge_{i\in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i'} \widehat{\overline{I}}_j)}{\overline{\tau} \vdash (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); h \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in \bigcup\{B_j' \mid j\in B_i^t \cup B_i^f\}} \widehat{\overline{I}}_k)}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash \overline{h} : \overline{\tau} \\ \overline{\tau} \vdash (\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); h \ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in \bigcup\{B_j' \mid j\in B_i^t \cup B_i^f\}} \widehat{\overline{I}}_k) \\ \forall k \in \bigcup \{\ B_j' \mid j \in B_{i'}^t\ \} :\ \overline{\tau}, \widehat{\overline{I}}_k \vdash \omega \\ \forall k \in \bigcup \{\ B_j' \mid j \in B_{i'}^f\ \} :\ \overline{\tau}, \widehat{\overline{I}}_k \vdash \omega\end{array}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle(\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2); h, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega, \sigma\rangle}$$

where:

($a$) $i' \in A$

($b$) $\forall i \in A:\ B_i^t \subseteq A'$

($c$) $\forall i \in A:\ B_i^f \subseteq A'$

From $\overline{\tau} \vdash s_1 : \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i^t} imr_j, \widehat{\overline{I}}_j)$ and $\overline{\tau} \vdash h : \bigwedge_{i\in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i'} \widehat{\overline{I}}_j)$ and ($b$) and rule (4.11), we have:

($d$) $\overline{\tau} \vdash s_1; h\ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k\in\bigcup\{B_j' \mid j\in B_i^t\}} \widehat{\overline{I}}_k)$

From $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and ($d$) and ($a$) and $\forall k \in \bigcup \{\ B_j' \mid j \in B_{i'}^t\ \} :\ \overline{\tau}, \widehat{\overline{I}}_k \vdash \omega$ and rule (4.5), we have $\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle s_1; h, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega, \sigma\rangle$, which proves (2).

- Rule (3.5).

  The proof is similar to that for rule (3.4).

- Rule (3.6).

  We have $Q = \langle \mathsf{skip}; a, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega\rangle$ and $Q' = \langle a, imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega\rangle$ and $Q.\widehat{\overline{I}} < \overline{d}$. To prove:

  1. $Q'.\widehat{\overline{I}} < \overline{d}$
  2. $\overline{\tau}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle a, imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega, \sigma\rangle$

  Since $Q$ is consistent, there are two subcases:

  Subcase 1: $a = m$ and $\omega = \mathsf{nil}$.

  From rules (4.4), (4.9), and (4.1), we have:

$$\dfrac{\overline{\tau} \vdash h\ :\ \overline{\tau} \qquad \dfrac{\overline{\tau} \vdash \mathsf{skip}\ :\ \bigwedge_{i\in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j\in B_i} imr_j, \widehat{\overline{I}}_j) \qquad \overline{\tau}, \bigvee_{i\in A'} imr_i, \widehat{\overline{I}}_i \vdash m}{\overline{\tau}, \bigvee_{i\in A} imr_i, \widehat{\overline{I}}_i \vdash \mathsf{skip}; m}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \mathsf{skip}; m, imr_{i'}, \widehat{\overline{I}}_{i'}, \mathsf{nil}, \sigma\rangle}$$

  where:

  ($a$) $i' \in A$

  ($b$) $\forall i \in A:\ B_i \subseteq A'$

(c) $\forall i \in A :\ \forall j \in B_i :\ imr_j = imr_i$

(d) $\bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}$

From $(a)$ and $(d)$, $\exists j'$ such that (see item (2) of defn. (4.2)):

(e) $j' \in B_{i'}$

(f) $\widehat{\overline{I}}_{j'} = \gamma(imr_{i'}, \widehat{\overline{I}}_{i'})$

(g) $\widehat{\overline{I}}_{j'} < \overline{d}$

From $(f)$ and (g), we have $\gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) < \overline{d}$, which proves (1). From $(a)$–$(c)$ and $(e)$ and $(f)$, we have:

(h) $j' \in A'$ and $(imr_{j'}, \widehat{\overline{I}}_{j'}) = (imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}))$

Finally, from $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and $\overline{\tau}, \bigvee_{i \in A'} imr_i, \widehat{\overline{I}}_i \vdash m$ and $(h)$ and rule (4.4), we have $\overline{\tau}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle m, imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}), \mathsf{nil}, \sigma \rangle$, which proves (ii).

Subcase 2: $a = h$.

From rules (4.5), (4.11), and (4.1), we have:

$$\frac{\begin{array}{c}\overline{\tau} \vdash \mathsf{skip} \ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j) \\ \overline{\tau} \vdash h \ :\ \bigwedge_{i \in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B'_i} \widehat{\overline{I}}_j)\end{array}}{\overline{\tau} \vdash \mathsf{skip}; h \ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k \in \bigcup\{B'_j \mid j \in B_i\}} \widehat{\overline{I}}_k)}$$

$$\frac{\begin{array}{c}\overline{\tau} \vdash \overline{h} \ :\ \overline{\tau} \\ \overline{\tau} \vdash \mathsf{skip}; h \ :\ \bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{k \in \bigcup\{B'_j \mid j \in B_i\}} \widehat{\overline{I}}_k) \\ \forall k \in \bigcup \{\ B'_j \mid j \in B_{i'}\ \} :\ \overline{\tau}, \widehat{\overline{I}}_k \vdash \omega\end{array}}{\overline{\tau}, \widehat{\overline{I}}_{i'} \vdash \langle \mathsf{skip}; h, imr_{i'}, \widehat{\overline{I}}_{i'}, \omega, \sigma \rangle}$$

where:

(a) $i' \in A$

(b) $\forall i \in A :\ B_i \subseteq A'$

(c) $\forall i \in A :\ \forall j \in B_i :\ imr_j = imr_i$

(d) $\bigwedge_{i \in A}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} imr_j, \widehat{\overline{I}}_j)$ is safe w.r.t. $\overline{\tau}$

From $(a)$ and $(d)$, $\exists j'$ such that (see item (2) of defn. (4.2)):

$(e)$ $j' \in B_{i'}$

$(f)$ $\widehat{\overline{I}}_{j'} = \gamma(imr_{i'}, \widehat{\overline{I}}_{i'})$

$(g)$ $\widehat{\overline{I}}_{j'} < \overline{d}$

From $(f)$ and $(g)$, we have $\gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) < \overline{d}$, which proves (1). From $(a)$–$(c)$ and $(e)$ and $(f)$, we have:

$(h)$ $j' \in A'$ and $(imr_{j'}, \widehat{\overline{I}}_{j'}) = (imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}))$

From $\forall k \in \bigcup \{ B'_j \mid j \in B_{i'} \} \vdash \omega$ and $(e)$, we have:

$(i)$ $\forall k \in B_{j'} : \overline{\tau}, \widehat{\overline{I}}_k \vdash \omega$

Finally, from $\overline{\tau} \vdash \overline{h} : \overline{\tau}$ and $\overline{\tau} \vdash h : \bigwedge_{i \in A'}(imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B'_i} \widehat{\overline{I}}_j)$ and $(i)$ and $(h)$ and rule (4.5), we have $\overline{\tau}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) \vdash \langle h, imr_{i'}, \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}), \omega, \sigma \rangle$, which proves (2).

- Rules (3.7)–(3.9).

  The proofs are similar to that for rule (3.6). □

Lemma 4.5 **(Multi-step Type Preservation)**

If $Q$ is consistent and $Q.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q.\widehat{\overline{I}} \vdash \langle Q.a, Q.imr, Q.\widehat{\overline{I}}, Q.\omega, \sigma \rangle$ and $Q \hookrightarrow^n Q'$, then $Q'.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q'.\widehat{\overline{I}} \vdash \langle Q'.a, Q'.imr, Q'.\widehat{\overline{I}}, Q'.\omega, \sigma \rangle$.

*Proof.* By induction on $n$. The base case ($n = 0$) is trivial. To prove the induction step, suppose $Q$ is consistent and $Q.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q.\widehat{\overline{I}} \vdash \langle Q.a, Q.imr, Q.\widehat{\overline{I}}, Q.\omega, \sigma \rangle$ and $Q \hookrightarrow^n Q' \hookrightarrow Q''$. From $Q$ is consistent $Q \hookrightarrow^n Q'$ and prop. (4.1), we have $Q'$ is consistent. From $Q$ is consistent and $Q.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q.\widehat{\overline{I}} \vdash \langle Q.a, Q.imr, Q.\widehat{\overline{I}}, Q.\omega, \sigma \rangle$ and $Q \hookrightarrow^n Q'$ and the induction hypothesis, we have $Q'.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q'.\widehat{\overline{I}} \vdash \langle Q'.a, Q'.imr, Q'.\widehat{\overline{I}}, Q'.\omega, \sigma \rangle$. From $Q'$ is consistent and $Q'.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q'.\widehat{\overline{I}} \vdash \langle Q'.a, Q'.imr, Q'.\widehat{\overline{I}}, Q'.\omega, \sigma \rangle$ and $Q' \hookrightarrow Q''$ and lemma (4.4), we have $Q''.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q''.\widehat{\overline{I}} \vdash \langle Q''.a, Q''.imr, Q''.\widehat{\overline{I}}, Q''.\omega, \sigma \rangle$. □

Finally, the following lemma states that if a program type checks, then the model checker accepts it.

Lemma 4.6 **(From Type Checking to Model Checking)**
If $\exists \overline{\tau} : \ \overline{\tau}, \overline{0} \vdash P_0$ then $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$.

*Proof.* From $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil} \rangle$ and defn. (4.3), we have $Q_0$ is consistent. From $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil} \rangle$ and $\overline{d} > \overline{0}$, we have $Q_0.\widehat{\overline{I}} < \overline{d}$. From $\overline{\tau}, \overline{0} \vdash P_0$ and $P_0 \in \{ \langle m_0, \neg \mathsf{t}_0, \overline{I}, \mathsf{nil}, \lambda x.0 \rangle \mid \ \overline{I} \leq \overline{0} \}$, we have $\overline{\tau}, \overline{0} \vdash \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil}, \lambda x.0 \rangle$. From $\overline{\tau}, \overline{0} \vdash \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil}, \lambda x.0 \rangle$ and $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil} \rangle$, we have $\overline{\tau}, Q_0.\widehat{\overline{I}} \vdash \langle Q_0.a, Q_0.imr, Q_0.\widehat{\overline{I}}, Q_0.\omega, \lambda x.0 \rangle$. From $Q_0$ is consistent and $Q_0.\widehat{\overline{I}} < \overline{d}$ and $\overline{\tau}, Q_0.\widehat{\overline{I}} \vdash \langle Q_0.a, Q_0.imr, Q_0.\widehat{\overline{I}}, Q_0.\omega, \lambda x.0 \rangle$ and $Q_0 \hookrightarrow^n Q$ and lemma (4.5), we have $Q.\widehat{\overline{I}} < \overline{d}$. □

We obtain type soundness with respect to the concrete semantics as the following corollary.

Corollary 4.1 **(Type Soundness)** If $\exists \overline{\tau} : \ \overline{\tau}, \overline{0} \vdash P_0$ then $(P_0 \rightarrow^n P \Rightarrow P.\overline{I} < \overline{d})$.

*Proof.* Combine lemma (4.6) and corollary (3.1). □

# 5. EXAMPLE

In this chapter, we illustrate how we construct types from the reachable state-space. Consider the interrupt-driven embedded system in Figure (5.1). The program in this system is the same as that in Figure (1.3) except that each primitive statement and iret has a unique label. An excerpt of the reachable state-space $\mathcal{R}$ for this system is shown in Figure (5.2) in the form of rules. A rule $Q \overset{n}{\hookrightarrow} Q'$ is an instantiation of rule $(n)$ in Figure (3.1). The notation $Q!$ indicates that the state-space reachable from $Q$ is not shown for brevity. Finally, instantiations of rules (3.2)–(3.9) use the approximation $t$ instead of $t-1$ in the abstract latency vector transfer function. (Recall from chapter 3 that any approximation $\geq t-1$ is correct).

$$\overline{r}(1) = \overline{r}(2) = 40,\ \overline{d}(1) = \overline{d}(2) = 40,\ t = 5$$

```
ei0                handler 1 {          handler 2 {
loop {                skip1                skip21
    skip0             iret1                ei2
}                  }                       skip22
                                           iret2
                                       }
```

Figure 5.1 An example interrupt-driven embedded system

To construct the type of any $m$ in the program, we compute the set $\mathbb{A}^m_{\mathcal{R}} = \{\, Q \in \mathcal{R} \mid Q.a = m \,\}$. Then, the type of $m$ is $\bigvee_{Q \in \mathbb{A}^m_{\mathcal{R}}} Q.imr, Q.\widehat{\overline{I}}$.

To construct the type of any $h$ in the program, we compute the set $\mathbb{A}^h_{\mathcal{R}} = \{\, Q \in \mathcal{R} \mid Q.a = h \,\}$ and for each $Q \in \mathbb{A}^h_{\mathcal{R}}$, the set $\mathbb{B}^Q_{\mathcal{R}} = \{\, Q' \in \mathcal{R} \mid Q'.a =$

$a$ and $Q'$ is reachable from $Q$ in $\mathcal{R}$ }, where $Q.\omega = \langle a, imr_r \rangle :: \omega_r$. Then, the type of $h$ is $\bigwedge_{Q \in \mathbb{A}^h_{\mathcal{R}}} (Q.imr, Q.\widehat{\overline{I}} \longrightarrow \bigvee_{Q' \in \mathbb{B}^Q_{\mathcal{R}}} Q'.\widehat{\overline{I}})$. Also, the handler type $\overline{\tau}(u)$ is constructed from the type of $\overline{h}(u)$, namely, we have $\overline{\tau}(u) = \bigwedge_{i \in A} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ if the type of $\overline{h}(u)$ is $\bigwedge_{i \in A} (\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ where $\forall i \in A : imr_i(0) = imr_i(u) = 1$, see rule (4.12).

To construct the type of any $s$ in the program, we compute the set $\mathbb{A}^{s;a}_{\mathcal{R}} = \{ Q \in \mathcal{R} \mid Q.a = s; a \}$ and for each $Q \in \mathbb{A}^{s;a}_{\mathcal{R}}$, the set $\mathbb{B}^Q_{\mathcal{R}} = \{ Q' \in \mathcal{R} \mid Q'.a = a$ and $Q'$ is reachable from $Q$ in $\mathcal{R} \}$. Then, the type of $s$ is $\bigwedge_{Q \in \mathbb{A}^{s;a}_{\mathcal{R}}} (Q.imr, Q.\widehat{\overline{I}} \longrightarrow \bigvee_{Q' \in \mathbb{B}^Q_{\mathcal{R}}} Q'.imr, Q'.\widehat{\overline{I}})$.

We now illustrate the construction of $\overline{\tau}$ for the example system from the reachable state-space in Figure (5.2). Consider `handler 1`. The types of $\mathsf{skip}_1$ and $\mathsf{iret}_1$ are as follows:

$$\begin{array}{lcll} \mathsf{skip}_1 & : & \ldots \wedge (000, 15\ 15 \longrightarrow 000, 20\ 20) \wedge & \text{From (5.9)} \\ & & \quad (000,\ 5\ 20 \longrightarrow 000, 10\ 25) \wedge \ \ldots & \text{From (5.15)} \\ \mathsf{iret}_1 & : & \ldots \wedge (000, 20\ 20 \longrightarrow -15\ 25) \wedge & \text{From (5.10)} \\ & & \quad (000, 10\ 25 \longrightarrow -25\ 30) \wedge \ \ldots & \text{From (5.16)} \end{array}$$

We must show that the above types satisfy the side-conditions of rules (4.1) and (4.10) respectively, in particular, that they are safe w.r.t. $\overline{\tau}$ and $\overline{\tau}, 1$ respectively. We will do so once we construct $\overline{\tau}$. Next, the type of $\mathsf{skip}_1; \mathsf{iret}_1$ is as follows:

$$\begin{array}{lcll} \mathsf{skip}_1; \mathsf{iret}_1 & : & \ldots \wedge (000, 15\ 15 \longrightarrow -15\ 25) \wedge & \text{From (5.9) and (5.10)} \\ & & \quad (000,\ 5\ 20 \longrightarrow -25\ 30) \wedge \ \ldots & \text{From (5.15) and (5.16)} \end{array}$$

It is easy to see from the types of $\mathsf{skip}_1$ and $\mathsf{iret}_1$ and $\mathsf{skip}_1; \mathsf{iret}_1$ that $\mathsf{skip}_1; \mathsf{iret}_1$ type checks by rule (4.11), in particular, the side-condition of rule (4.11) is trivially satisfied. From the type of $\mathsf{skip}_1; \mathsf{iret}_1$ and rule (4.12), it follows that:

$$\overline{\tau}(1) \ = \ \ldots \ \wedge \ (15\ 15 \xrightarrow{110} -15\ 25) \ \wedge \ (5\ 20 \xrightarrow{110} -25\ 30) \ \wedge \ \ldots$$

We next consider `handler 2`. The types of $\mathsf{skip}_{21}$, $\mathsf{ei}_2$, $\mathsf{skip}_{22}$, and $\mathsf{iret}_2$ are as follows:

$$
\begin{aligned}
\mathsf{skip}_{21} \;&:\; \ldots \wedge (010,\ \ 5\ \ 5 \longrightarrow 010,\ \ 10\ 10) \wedge \ldots \\
\mathsf{ei}_2 \;&:\; \ldots \wedge (010,\ 10\ 10 \longrightarrow 110,\ 15\ 15) \wedge \ldots \\
\mathsf{skip}_{22} \;&:\; \ldots \wedge (110,\ 15\ 15 \longrightarrow (110,\ \ 5\ \ 20 \ \vee\ 110,\ \text{–}10\ 30)) \wedge \\
&\qquad\qquad (110, \text{–}15\ 25 \longrightarrow 110, \text{–}10\ 30) \wedge \ldots \\
\mathsf{iret}_2 \;&:\; \ldots \wedge (110,\ \ 5\ \ 20 \longrightarrow (\ 5\ \text{–}15 \ \vee\ \text{–}20\ \text{–}5)) \wedge \\
&\qquad\qquad (110, \text{–}10\ 30 \longrightarrow \ \text{–}5\ \ \text{–}5) \wedge \\
&\qquad\qquad (110, \text{–}25\ 30 \longrightarrow \text{–}20\ \text{–}5) \wedge \ \ldots
\end{aligned}
$$

The types of $\mathsf{skip}_{22};\mathsf{iret}_2$ and $\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2$ and $\mathsf{skip}_{21};\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2$ are as follows:

$$
\begin{aligned}
\mathsf{skip}_{22};\mathsf{iret}_2 \;&:\; \ldots \wedge (110,\ \ 15\ 15 \longrightarrow (\text{–}5\ \text{–}5 \vee 5\ \text{–}15 \vee \text{–}20\ \text{–}5)) \wedge \\
&\qquad\qquad (110, \text{–}15\ 25 \longrightarrow \ \text{–}5\ \text{–}5) \wedge \ \ldots \\
\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2 \;&:\; \ldots \wedge (010,\ 10\ 10 \longrightarrow (\text{–}5\ \text{–}5 \vee 5\ \text{–}15 \vee \text{–}20\ \text{–}5)) \wedge \\
\mathsf{skip}_{21};\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2 \;&:\; \ldots \wedge (010,\ \ 5\ \ 5 \longrightarrow (\text{–}5\ \text{–}5 \vee 5\ \text{–}15 \vee \text{–}20\ \text{–}5)) \wedge \ \ldots
\end{aligned}
$$

Notice that the property of the side-condition of rule (4.11) that each context in which $h$ begins executing need not be a context in which $s$ finishes executing is needed to type check $\mathsf{skip}_{22};\mathsf{iret}_2$ because the domain of the type of $\mathsf{iret}_2$ does not exactly match the range of the type of $\mathsf{skip}_{22}$, namely, (110, –25 30) is present in the domain of the type of $\mathsf{iret}_2$ but not in the range of the type of $\mathsf{skip}_{22}$. The situation is similar in type checking $\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2$, namely, (110, –15 25) is present in the domain of the type of $\mathsf{skip}_{22};\mathsf{iret}_2$ but not in the range of the type of $\mathsf{ei}_2$. From the type of $\mathsf{skip}_{21};\mathsf{ei}_2;\mathsf{skip}_{22};\mathsf{iret}_2$ and rule (4.12), it follows that:

$$
\overline{\tau}(2) \;=\; \ldots \wedge (5\ 5 \xrightarrow{111} (\text{–}5\ \text{–}5 \ \vee\ 5\ \text{–}15 \ \vee\ \text{–}20\ \text{–}5)) \wedge \ \ldots
$$

Finally, we need to show that the types of the primitive statements in the two handlers and the types of $\mathsf{iret}_1$ and $\mathsf{iret}_2$ satisfy the side-conditions of rules (4.1) and (4.10) respectively. We will do so only for the type of $\mathsf{skip}_{22}$. It is trivial to see that it satisfies the first side-condition. To show that it satisfies the second side-condition,

we need to show that each of the components in the intersection type of $\mathsf{skip}_{22}$ satisfy conditions (1) and (2) in defn. (4.2). We will do so only for the first component:
Condition 1. We have $\widehat{\mathcal{L}}(110, 15\ 15) = \{1\}$. Recall that:

$$\overline{\tau}(1) \;=\; \ldots \;\wedge\; (15\ 15 \xrightarrow{110} {-}15\ 25) \;\wedge\; (5\ 20 \xrightarrow{110} {-}25\ 30) \;\wedge\; \ldots$$

(a) It is evident from $\overline{\tau}(1)$ that it is safe to begin executing `handler 1` in context $(110, 15\ 15)$.

(b) If `handler 1` begins executing in context $(110, 15\ 15)$, then it finishes executing in context $(110, {-}15\ 25)$. It is evident from the type of $\mathsf{skip}_{22}$ that:

i. It is safe to resume executing $\mathsf{skip}_{22}$ in context $(110, {-}15\ 25)$.

ii. Each possible context in which $\mathsf{skip}_{22}$ finishes executing after resuming execution in context $(110, {-}15\ 25)$ is also a possible context in which $\mathsf{skip}_{22}$ finishes executing after beginning execution in context $(110, 15\ 15)$, namely, $\{\ (110, {-}10\ 30)\ \} \subseteq \{\ (110, 5\ 20), (110, {-}10\ 30)\ \}$.

Condition 2. There is a context $(110, 5\ 20)$ in which $\mathsf{skip}_{22}$ finishes executing after beginning execution in context $(110, 15\ 15)$ such that $(5\ 20) = \gamma(110, 15\ 15)$ and $(5\ 20) < (40\ 40)$.

Thus far, we have shown that both handlers type check, that is, $\overline{\tau} \vdash \overline{h} : \overline{\tau}$. It is possible to construct the type $M$ of the main procedure in a similar fashion and show that $\overline{\tau}, M \vdash \mathsf{ei}_0; \mathsf{loop}\ \mathsf{skip}_0$ and that $(011,\ 0\,0)$ is a component of $M$. Then, from rule (4.4), it will follow that the program is well typed, that is, $\overline{\tau}, \overline{0} \vdash P_0$.

$[\omega$ denotes $\langle \mathsf{loop\ skip}_0, 111\rangle :: \mathsf{nil}]$

$$\langle \mathsf{ei}_0; \mathsf{loop\ skip}_0, 011, 0\ 0, \mathsf{nil}\rangle \overset{3.7}{\hookrightarrow} \langle \mathsf{loop\ skip}_0, 111, 5\ 5, \mathsf{nil}\rangle \tag{5.1}$$

$$\langle \mathsf{loop\ skip}_0, 111, 5\ 5, \mathsf{nil}\rangle \overset{3.3}{\hookrightarrow} \langle \mathsf{skip}_1; \mathsf{iret}_1, 001, 5\ 5, \omega\rangle \tag{5.2}$$

$$\langle \mathsf{loop\ skip}_0, 111, 5\ 5, \mathsf{nil}\rangle \overset{3.3}{\hookrightarrow} \langle \mathsf{skip}_{21}; \mathsf{ei}_2; \mathsf{skip}_{22}; \mathsf{iret}_2, 010, -25\ 15, \omega\rangle \tag{5.3}$$

$$\langle \mathsf{loop\ skip}_0, 111, 5\ 5, \mathsf{nil}\rangle \overset{3.3}{\hookrightarrow} \langle \mathsf{skip}_0; \mathsf{loop\ skip}_0, 111, 5\ 5, \mathsf{nil}\rangle! \tag{5.4}$$

$$\langle \mathsf{skip}_{21}; \mathsf{ei}_2; \mathsf{skip}_{22}; \mathsf{iret}_2, 010, 5\ 5, \omega\rangle \overset{3.6}{\hookrightarrow} \langle \mathsf{ei}_2; \mathsf{skip}_{22}; \mathsf{iret}_2, 010, 10\ 10, \omega\rangle \tag{5.5}$$

$$\langle \mathsf{ei}_2; \mathsf{skip}_{22}; \mathsf{iret}_2, 010, 10\ 10, \omega\rangle \overset{3.7}{\hookrightarrow} \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110, 15\ 15, \omega\rangle \tag{5.6}$$

$$\langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110, 15\ 15, \omega\rangle \overset{3.1}{\hookrightarrow} \langle \mathsf{skip}_1; \mathsf{iret}_1, 000, 15\ 15, \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110\rangle :: \omega\rangle \tag{5.7}$$

$$\langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110, 15\ 15, \omega\rangle \overset{3.6}{\hookrightarrow} \langle \mathsf{iret}_2, 110, 5\ 20, \omega\rangle \tag{5.8}$$

$$\langle \mathsf{skip}_1; \mathsf{iret}_1, 000, 15\ 15, \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110\rangle :: \omega\rangle \overset{3.6}{\hookrightarrow} \langle \mathsf{iret}_1, 000, 20\ 20, \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110\rangle :: \omega\rangle \tag{5.9}$$

$$\langle \mathsf{iret}_1, 000, 20\ 20, \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110\rangle :: \omega\rangle \overset{3.2}{\hookrightarrow} \langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110, -15\ 25, \omega\rangle \tag{5.10}$$

$$\langle \mathsf{skip}_{22}; \mathsf{iret}_2, 110, -15\ 25, \omega\rangle \overset{3.6}{\hookrightarrow} \langle \mathsf{iret}_2, 110, -10\ 30, \omega\rangle \tag{5.11}$$

$$\langle \mathsf{iret}_2, 110, -10\ 30, \omega\rangle \overset{3.2}{\hookrightarrow} \langle \mathsf{loop\ skip}_0, 111, -5\ -5, \mathsf{nil}\rangle! \tag{5.12}$$

$$\langle \mathsf{iret}_2, 110, 5\ 20, \omega\rangle \overset{3.1}{\hookrightarrow} \langle \mathsf{skip}_1; \mathsf{iret}_1, 000, 5\ 20, \langle \mathsf{iret}_2, 110\rangle :: \omega\rangle \tag{5.13}$$

$$\langle \mathsf{iret}_2, 110, 5\ 20, \omega\rangle \overset{3.2}{\hookrightarrow} \langle \mathsf{loop\ skip}_0, 111, 5\ -15, \mathsf{nil}\rangle! \tag{5.14}$$

$$\langle \mathsf{skip}_1; \mathsf{iret}_1, 000, 5\ 20, \langle \mathsf{iret}_2, 110\rangle :: \omega\rangle \overset{3.6}{\hookrightarrow} \langle \mathsf{iret}_1, 000, 10\ 25, \langle \mathsf{iret}_2, 110\rangle :: \omega\rangle \tag{5.15}$$

$$\langle \mathsf{iret}_1, 000, 10\ 25, \langle \mathsf{iret}_2, 110\rangle :: \omega\rangle \overset{3.2}{\hookrightarrow} \langle \mathsf{iret}_2, 110, -25\ 30, \omega\rangle \tag{5.16}$$

$$\langle \mathsf{iret}_2, 110, -25\ 30, \omega\rangle \overset{3.2}{\hookrightarrow} \langle \mathsf{loop\ skip}_0, 111, -20\ -5, \mathsf{nil}\rangle! \tag{5.17}$$

Figure 5.2 Excerpt of the reachable state-space of the interrupt-driven system in Figure (5.1).

# 6. FROM MODEL CHECKING TO TYPE CHECKING

In this chapter, we prove lemma (6.20) which states that if a program is accepted by the model checker, then it type checks. Our proof architecture consists of three stages:

- In section (6.1), we describe how we construct types from the reachable state-space.
- In section (6.2), we prove that each constructed type is well formed in the sense of defn. (4.1).
- In section (6.3), we prove that each statement, each main part, and each handler part in the program has a valid type derivation.

In section (6.4), we put the three stages together to prove lemma (6.20). A peculiarity of our proof architecture is that the first and second stages do not explicitly use the assumption that the model checker accepts the program, namely, the assumption that $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ holds; they merely require that the reachable state-space is finite. However, notice that if the reachable state-space is infinite, then $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ does not hold (see the argument preceding proposition (3.1)), and the model checker rejects the program. Thus, the first and second stages implicitly use the assumption that the model checker accepts the program. The third stage uses the assumption explicitly, namely, each statement, each main part, and each handler part in the program has either a primitive statement or an $\mathsf{iret}$, whose type rules have a side-condition that checks that all deadlines are met in its type (see items (2) and (4) of defn. (4.2)).

## 6.1 Type Construction

The primary difficulty in constructing types from the reachable state-space is that in order to construct the type of a statement $s$ or a handler part $h$, we need to compute, for each context in which $s$ or $h$ might begin executing, the set of possible contexts in which it will finish executing, namely, for each $\langle s; a, imr, \widehat{\overline{I}}, \omega\rangle \in \mathcal{R}$, we need to compute the set $\{\ \langle a, imr', \widehat{\overline{J}}, \omega\rangle \in \mathcal{R} \mid \langle a, imr', \widehat{\overline{J}}, \omega\rangle$ is reachable from $\langle s; a, imr, \widehat{\overline{I}}, \omega\rangle$ in $\mathcal{R}\ \}$, and for each $\langle h, imr, \widehat{\overline{I}}, \omega\rangle \in \mathcal{R}$ where $\omega = \langle a, imr'\rangle :: \omega'$, we need to compute the set $\{\ \langle a, imr', \widehat{\overline{J}}, \omega'\rangle \in \mathcal{R} \mid \langle a, imr', \widehat{\overline{J}}, \omega'\rangle$ is reachable from $\langle h, imr, \widehat{\overline{I}}, \omega\rangle$ in $\mathcal{R}\ \}$.

We choose a unique index for each element in the set $\{\ (imr, \widehat{\overline{I}}, \omega) \mid \langle a, imr, \widehat{\overline{I}}, \omega\rangle \in \mathcal{R}\ \}$ and denote the set of all these indices $\mathcal{U}$. Let $\mathcal{P}(A)$ denote the power-set of set $A$ and let $A^n$ denote the $n$-fold Cartesian product over set $A$. Then, we define a function $\mathcal{F}_{\mathcal{R}} : (\mathcal{P}(\mathcal{U}))^n \longrightarrow (\mathcal{P}(\mathcal{U}))^n$ such that for each $\langle s; a, imr, \widehat{\overline{I}}, \omega\rangle \in \mathcal{R}$ (resp. $\langle h, imr, \widehat{\overline{I}}, \omega\rangle \in \mathcal{R}$) and for each $i \in \mathcal{U}$, $\mathcal{F}_{\mathcal{R}}$ has an argument $arg_{\mathcal{R}}^{s;a,i}$ (resp. $arg_{\mathcal{R}}^{h,i}$) and a result $res_{\mathcal{R}}^{s;a,i}$ (resp. $res_{\mathcal{R}}^{h,i}$). We require that $\mathcal{R}$ is finite so that $n$ is finite.

Definition 6.1 $\mathcal{F}_{\mathcal{R}} : (\mathcal{P}(\mathcal{U}))^n \longrightarrow (\mathcal{P}(\mathcal{U}))^n = \lambda\,(arg_{\mathcal{R}}^{a,i})^n\ .\ (res_{\mathcal{R}}^{a,i})^n$ where:

1. If $a = s; a'$ where $s = \mathsf{skip}$, $\mathsf{ei}$, $\mathsf{imr} := \mathsf{imr} \wedge imr'$, or $x := e$, then:
$$res_{\mathcal{R}}^{s;a',i} = \bigcup \{\ arg_{\mathcal{R}}^{s;a',j} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } j \in arg_{\mathcal{R}}^{\overline{h}(u),j^\bullet}\ \} \cup \{\ j^\star\ \} \text{ where}$$
$$j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s; a', imr_i\rangle :: \omega_i$$
$$j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = \chi_s(imr_i) \wedge \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_i$$

2. If $a = s; a'$ where $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$ then:
$$res_{\mathcal{R}}^{s;a',i} = arg_{\mathcal{R}}^{s_1;a',i} \cup arg_{\mathcal{R}}^{s_2;a',i}$$

3. If $a = s; a'$ where $s = s_1; s_2$ then:
$$res_{\mathcal{R}}^{s;a',i} = \bigcup \{\ arg_{\mathcal{R}}^{s_2;a',j} \mid j \in arg_{\mathcal{R}}^{s_1;(s_2;a'),i}\ \}$$

4. If $a = h$ where $h = \mathsf{iret}_v$ and $\omega_i = \langle a', imr_r \rangle :: \omega_r$ then:

$$res_{\mathcal{R}}^{h,i} = \bigcup \{\, arg_{\mathcal{R}}^{h,j} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } j \in arg_{\mathcal{R}}^{\overline{h}(u),j^\bullet} \,\} \cup \{\, j^\star \,\} \text{ where}$$
$$j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i$$
$$j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = imr_r \wedge \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_r$$

5. If $a = h$ where $h = s; h'$ then:

$$res_{\mathcal{R}}^{h,i} = \bigcup \{\, arg_{\mathcal{R}}^{h',j} \mid j \in arg_{\mathcal{R}}^{s;h',i} \,\}$$

Note that in the above definition, we do not refer to any argument as $arg_{\mathcal{R}}^{a,i}$ or to any result as $res_{\mathcal{R}}^{a,i}$ because this notation can be ambiguous, namely, $a$ might be of the form $s; a'$ and $h$ simultaneously, in which case it is not clear whether $arg_{\mathcal{R}}^{a,i}$ refers to $arg_{\mathcal{R}}^{s;a',i}$ or $arg_{\mathcal{R}}^{h,i}$, and similarly for $res_{\mathcal{R}}^{a,i}$. It is straightforward to check that $\mathcal{F}_{\mathcal{R}}$ is monotone in which case it follows from Tarski's fixed-point theorem that it has a least fixed point $\mu\mathcal{F}_{\mathcal{R}}$. We next define the following sets:

Definition 6.2 We have:

1. $\mathbb{A}_{\mathcal{R}}^a = \{\, i \mid \langle a, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R} \,\}$
2. $\mathbb{B}_{\mathcal{R}}^{a,i} = \pi^{a,i}\ \mu\mathcal{F}_{\mathcal{R}}$

We are now ready to construct from $\mathcal{R}$ the type of each statement $s$, each main part $m$, and each handler part $h$ in the program, denoted $type_{\mathcal{R}}(s)$, $type_{\mathcal{R}}(m)$, and $type_{\mathcal{R}}(h)$ respectively, as well as the handler types, denoted $\overline{\tau}_{\mathcal{R}}$.

Definition 6.3 **(Type Construction)**

$$type_{\mathcal{R}}(\,s\,) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}} imr_j, \widehat{\overline{I}}_j)$$
$$type_{\mathcal{R}}(m) = \bigvee_{i \in \mathbb{A}_{\mathcal{R}}^m} imr_i, \widehat{\overline{I}}_i$$
$$type_{\mathcal{R}}(\,h\,) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^h} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{h,i}} \widehat{\overline{I}}_j)$$

$\forall u \in 1..N$ :

$$\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \text{ if}$$
$$type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A} (\phi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j) \text{ and}$$
$$\forall i \in A :\ imr_i(0) = imr_i(u) = 1$$

The precondition in the construction of $\overline{\tau}_{\mathcal{R}}(u)$ states that each context $(imr_i, \widehat{\overline{I}}_i)$ in which the handler part $\overline{h}(u)$ begins executing must satisfy the condition $imr_i(0) = imr_i(u) = 0$. The following proposition states that this precondition indeed holds:

Proposition 6.1 $\forall u \in 1..N : \ (i \in \mathbb{A}_{\mathcal{R}}^{\overline{h}(u)} \ \Rightarrow \ imr_i = \phi_u(imr_i))$

We will sometimes need to appeal to the fact that, for each $s;a$ (resp. $h$) and for each $i \in \mathcal{U}$ such that $\langle s;a, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$ (resp. $\langle h, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$), the indices $j^\bullet$ and $j^\star$ specified in the definition of $res_{\mathcal{R}}^{s;a,i}$ (resp. $res_{\mathcal{R}}^{h,i}$) in item (1) (resp. item (4)) of defn. (6.1) do exist in $\mathcal{U}$. This is proved in the following lemma.

Lemma 6.1 We have:

1. If $s = \mathsf{skip}$, $\mathsf{ei}$, $\mathsf{imr} := \mathsf{imr} \wedge imr'$, or $x := e$ and $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$, then $\langle \overline{h}(u), imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}, \omega_{j^\bullet}\rangle \in \mathcal{R}$ where $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s;a, imr_i\rangle :: \omega_i$.
2. If $s = \mathsf{skip}$, $\mathsf{ei}$, $\mathsf{imr} := \mathsf{imr} \wedge imr'$, or $x := e$ and $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$, then $\langle a, imr_{j^\star}, \widehat{\overline{I}}_{j^\star}, \omega_{j^\star}\rangle \in \mathcal{R}$
   where $imr_{j^\star} = \chi_s(imr_i) \wedge \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_i$.
3. If $h = \mathsf{iret}_v$ and $i \in \mathbb{A}_{\mathcal{R}}^{h}$ and $u \in \widehat{\mathcal{L}}(imr_i, \omega_i)$, then $\langle \overline{h}(u), imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}, \omega_{j^\bullet}\rangle \in \mathcal{R}$ where $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i\rangle :: \omega_i$.
4. If $h = \mathsf{iret}_v$ and $i \in \mathbb{A}_{\mathcal{R}}^{h}$ and $\omega_i = \langle a, imr_r\rangle :: \omega_r$, then $\langle a, imr_{j^\star}, \widehat{\overline{I}}_{j^\star}, \omega_{j^\star}\rangle \in \mathcal{R}$ where $imr_{j^\star} = imr_r \wedge \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_r$.

*Proof.*

1. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and defn. (6.2), we have $\langle s;a, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$. From $\langle s;a, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and rule (3.1), we have $\langle \overline{h}(u), imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}, \omega_{j^\bullet}\rangle \in \mathcal{R}$ where $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s;a, imr_i\rangle :: \omega_i$.
2. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and defn. (6.2), we have $\langle s;a, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$. From $\langle s;a, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}$ and rules (3.6)–(3.9), we have $\langle a, imr_{j^\star}, \widehat{\overline{I}}_{j^\star}, \omega_{j^\star}\rangle \in \mathcal{R}$ where $imr_{j^\star} = \chi_s(imr_i) \wedge \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_i$.

3. From $i \in \mathbb{A}^h_{\mathcal{R}}$ and defn. (6.2), we have $\langle h, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R}$. From $\langle h, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and rule (3.1), we have $\langle \overline{h}(u), imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}, \omega_{j^\bullet} \rangle \in \mathcal{R}$ where $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i$.
4. From $i \in \mathbb{A}^h_{\mathcal{R}}$ and defn. (6.2), we have $\langle h, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R}$. From $\langle h, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R}$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$ and rule (3.2), we have $\langle a, imr_{j^\star}, \widehat{\overline{I}}_{j^\star}, \omega_{j^\star} \rangle \in \mathcal{R}$ where $imr_{j^\star} = imr_r \wedge \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_r$. □

There are several useful relationships among the sets defined in defn. (6.2). We state them in proposition (6.2) and lemma (6.2).

Proposition 6.2 We have:

1. $\mathbb{A}^{\mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 ; a}_{\mathcal{R}} = \mathbb{A}^{s_1;a}_{\mathcal{R}} = \mathbb{A}^{s_2;a}_{\mathcal{R}}$
2. $\mathbb{A}^{\mathsf{loop}\ s}_{\mathcal{R}} = \mathbb{A}^{s;\mathsf{loop}\ s}_{\mathcal{R}}$

Lemma 6.2 We have:

1. If $a = s; a'$ where $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$ then
   $\forall i \in \mathbb{A}^{s;a'}_{\mathcal{R}} : \forall u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) : \forall j \in \mathbb{B}^{\overline{h}(u),j^\bullet}_{\mathcal{R}} : \mathbb{B}^{s;a',j}_{\mathcal{R}} \subseteq \mathbb{B}^{s;a',i}_{\mathcal{R}}$
   where $j^\bullet \in \mathcal{U}$ such that $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s; a, imr_i \rangle :: \omega_i$.
2. If $a = s; a'$ where $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$ then
   $\forall i \in \mathbb{A}^{s;a'}_{\mathcal{R}} : \exists j^\star \in \mathbb{B}^{s;a',i}_{\mathcal{R}} : \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i)$
3. If $a = s; a'$ where $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$ then
   $\forall i \in \mathbb{A}^{s;a'}_{\mathcal{R}} : \mathbb{B}^{s;a',i}_{\mathcal{R}} = \mathbb{B}^{s_1;a',i}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a',i}_{\mathcal{R}}$
4. If $a = s; a'$ where $s = s_1; s_2$ then
   $\forall i \in \mathbb{A}^{s;a'}_{\mathcal{R}} : \mathbb{B}^{s;a',i}_{\mathcal{R}} = \bigcup \{ \mathbb{B}^{s_2;a',j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s_1;(s_2;a'),i}_{\mathcal{R}} \}$
5. If $a = h$ where $h = \mathsf{iret}_v$ then
   $\forall i \in \mathbb{A}^h_{\mathcal{R}} : \forall u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) : \forall j \in \mathbb{B}^{\overline{h}(u),j^\bullet}_{\mathcal{R}} : \mathbb{B}^{h,j}_{\mathcal{R}} \subseteq \mathbb{B}^{h,i}_{\mathcal{R}}$
   where $j^\bullet \in \mathcal{U}$ such that $imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i$.

6. If $a = h$ where $h = \mathsf{iret}_v$ then
   $\forall i \in \mathbb{A}^h_{\mathcal{R}} : \exists j^\star \in \mathbb{B}^{h,i}_{\mathcal{R}} : \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i)$
7. If $a = h$ where $h = s; h'$ then
   $\forall i \in \mathbb{A}^h_{\mathcal{R}} : \mathbb{B}^{h,i}_{\mathcal{R}} = \bigcup \{ \mathbb{B}^{h',j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} \}$

*Proof.* Immediate from defns. (6.1) and (6.2), except items (2) and (6) which additionally appeal to items (2) and (4), respectively, of lemma (6.1) to show that $j^\star$ exists. □

If $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ (resp. $\mathbb{A}^h_{\mathcal{R}}$), then each index $j \in \mathbb{B}^{s;a,i}_{\mathcal{R}}$ (resp. $\mathbb{B}^{h,i}_{\mathcal{R}}$) has certain properties that we state in lemmas (6.3) and (6.4). We first generalize the $\mathsf{imr}$ transfer function of defn. (2.3) to compound statements:

Definition 6.4 ($\mathsf{imr}$ **Transfer Function, compound statements)**

| | | | | |
|---|---|---|---|---|
| *1.* | $\phi_s(imr)$ | $=$ | $\{ \chi_s(imr) \}$ | if $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$ |
| *2.* | $\phi_s(imr)$ | $=$ | $\phi_{s_1}(imr) \cup \phi_{s_2}(imr)$ | if $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$ |
| *3.* | $\phi_s(imr)$ | $=$ | $\phi_{s_2} \circ \phi_{s_1}(imr)$ | if $s = s_1; s_2$ |

Lemma 6.3 We have:

1. If $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $j \in \mathbb{B}^{s;a,i}_{\mathcal{R}}$, then $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_s(imr_i)$ and $w_j = w_i$.
2. If $i \in \mathbb{A}^h_{\mathcal{R}}$ and $j \in \mathbb{B}^{h,i}_{\mathcal{R}}$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$, then $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = imr_r$ and $\omega_j = \omega_r$.

*Proof.* Let $\mathcal{F}^n_{\mathcal{R}}$ denote $n$ applications of $\mathcal{F}_{\mathcal{R}}$. We have $\mu\mathcal{F}_{\mathcal{R}} = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$ where $n'$ is such that $\mathcal{F}^{n'+1}_{\mathcal{R}}(\overline{\emptyset}) = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$. Define $\mathbb{B}^{a,i,n}_{\mathcal{R}} = \pi^{a,i}\ \mathcal{F}^n_{\mathcal{R}}(\overline{\emptyset})$. From $\mathbb{B}^{a,i}_{\mathcal{R}} = \pi^{a,i}\ \mu\mathcal{F}_{\mathcal{R}}$ (see defn. (6.2)) and $\mu\mathcal{F}_{\mathcal{R}} = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$ and $\mathbb{B}^{a,i,n}_{\mathcal{R}} = \pi^{a,i}\ \mathcal{F}^n_{\mathcal{R}}(\overline{\emptyset})$, we have $\mathbb{B}^{a,i}_{\mathcal{R}} = \mathbb{B}^{a,i,n'}_{\mathcal{R}}$. We will prove that $\forall n$ :

1. If $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $j \in \mathbb{B}^{s;a,i,n}_{\mathcal{R}}$, then $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_s(imr_i)$ and $w_j = w_i$.
2. If $i \in \mathbb{A}^{h}_{\mathcal{R}}$ and $j \in \mathbb{B}^{h,i,n}_{\mathcal{R}}$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$, then $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = imr_r$ and $\omega_j = \omega_r$.

The proof is by induction on $n$.

Base case ($n = 0$): Trivial since $\forall i \in \mathbb{A}^{s;a}_{\mathcal{R}} : \mathbb{B}^{s;a,i,0}_{\mathcal{R}} = \emptyset$ and $\forall i \in \mathbb{A}^{h}_{\mathcal{R}} : \mathbb{B}^{h,i,0}_{\mathcal{R}} = \emptyset$.

Induction step:

Let $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $j \in \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}}$. To prove: $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_s(imr_i)$ and $w_j = w_i$. There are 3 cases:

- $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$

  From item (1) of defn. (6.1), we have:

$$\begin{aligned} \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}} \ = \ & \bigcup \{\, \mathbb{B}^{s;a,k,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } k \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}} \,\} \cup \{\, j^\star \,\} \text{ where} \\ & j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s; a, imr_i \rangle :: \omega_i \\ & j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = \chi_s(imr_i) \wedge \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_i \quad (6.1) \end{aligned}$$

  From (6.1) and $j \in \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}}$, we have either $j = j^\star$, in which case the result follows from item (2) of lemma (6.1), or $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $\exists j' \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}$ such that $j \in \mathbb{B}^{s;a,j',n}_{\mathcal{R}}$. Consider the latter case. From $j' \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}$ and $\omega_{j^\bullet} = \langle s; a, imr_i \rangle :: \omega_i$ and the induction hypothesis, we have $\langle s; a, imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and $imr_{j'} = imr_i$ and $\omega_{j'} = \omega_i$. From $\langle s; a, imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and defn. (6.2), we have $j' \in \mathbb{A}^{s;a}_{\mathcal{R}}$. From $j' \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $j \in \mathbb{B}^{s;a,j',n}_{\mathcal{R}}$ and the induction hypothesis, we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_s(imr_{j'})$ and $\omega_j = \omega_{j'}$. From $imr_j \in \phi_s(imr_{j'})$ and $imr_{j'} = imr_i$, we have $imr_j \in \phi_s(imr_i)$. From $\omega_j = \omega_{j'}$ and $\omega_{j'} = \omega_i$, we have $\omega_j = \omega_i$.

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From item (2) of defn. (6.1), we have $\mathbb{B}^{s;a,i,n+1}_{\mathcal{R}} = \mathbb{B}^{s_1;a,i,n}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a,i,n}_{\mathcal{R}}$. From $\mathbb{B}^{s;a,i,n+1}_{\mathcal{R}} = \mathbb{B}^{s_1;a,i,n}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a,i,n}_{\mathcal{R}}$ and $j \in \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}}$, we have $j \in \mathbb{B}^{s_1;a,i,n}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a,i,n}_{\mathcal{R}}$.

We will consider the case $j \in \mathbb{B}_{\mathcal{R}}^{s_1;a,i,n}$; the case $j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,i,n}$ is similar. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and item (1) of lemma (6.2), we have $i \in \mathbb{A}_{\mathcal{R}}^{s_1;a}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s_1;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s_1;a,i,n}$ and the induction hypothesis, we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_{s_1}(imr_i)$ and $\omega_j = \omega_i$. From $imr_j \in \phi_{s_1}(imr_i)$ and item (2) of defn. (6.4), we have $imr_j \in \phi_s(imr_i)$.

- $s = s_1; s_2$

  From item (3) of defn. (6.1), we have $\mathbb{B}_{\mathcal{R}}^{s;a,i,n+1} = \bigcup \{ \mathbb{B}_{\mathcal{R}}^{s_2;a,k,n} \mid k \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n} \}$. From $\mathbb{B}_{\mathcal{R}}^{s;a,i,n+1} = \bigcup \{ \mathbb{B}_{\mathcal{R}}^{s_2;a,k,n} \mid k \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n} \}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i,n+1}$, we have $\exists j' \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n}$ such that $j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,j',n}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $s = s_1; s_2$ and $(s_1; s_2)a = s_1; (s_2; a)$, we have $i \in \mathbb{A}_{\mathcal{R}}^{s_1;(s_2;a)}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s_1;(s_2;a)}$ and $j' \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n}$ and the induction hypothesis, we have $\langle s_2; a, imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and $imr_{j'} \in \phi_{s_1}(imr_i)$ and $\omega_{j'} = \omega_i$. From $\langle s_2; a, imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and defn. (6.2), we have $j' \in \mathbb{A}_{\mathcal{R}}^{s_2;a}$. From $j' \in \mathbb{A}_{\mathcal{R}}^{s_2;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,j',n}$ and the induction hypothesis, we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j \in \phi_{s_2}(imr_{j'})$ and $\omega_j = \omega_{j'}$. From $imr_{j'} \in \phi_{s_1}(imr_i)$ and $imr_j \in \phi_{s_2}(imr_{j'})$ and item (3) of defn. (6.4), we have $imr_j \in \phi_s(imr_i)$. From $\omega_j = \omega_{j'}$ and $\omega_{j'} = \omega_i$, we have $\omega_j = \omega_i$.

Let $i \in \mathbb{A}_{\mathcal{R}}^{h}$ and $j \in \mathbb{B}_{\mathcal{R}}^{h,i,n+1}$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$. To prove: $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = imr_r$ and $w_j = w_r$. There are 2 cases:

- $h = \mathsf{iret}_v$

  From item (4) of defn. (6.1), we have:

$$\begin{aligned} \mathbb{B}_{\mathcal{R}}^{h,i,n+1} \;=\; & \bigcup \{ \mathbb{B}_{\mathcal{R}}^{h,k,n} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n} \} \cup \{ j^\star \} \text{ where} \\ & j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i \\ & j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = imr_r \wedge \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_r \end{aligned} \tag{6.2}$$

  From (6.2) and $j \in \mathbb{B}_{\mathcal{R}}^{h,i,n+1}$, we have either $j = j^\star$, in which case the result follows from item (4) of lemma (6.1), or $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $\exists j' \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}$ such that $j \in \mathbb{B}_{\mathcal{R}}^{h,j',n}$. Consider the latter case. From $j' \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}$ and $\omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i$ and the induction hypothesis, we have $\langle h, imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$

and $imr_{j'} = imr_i$ and $\omega_{j'} = \omega_i$. From $\omega_{j'} = \omega_i$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$, we have $\omega_{j'} = \langle a, imr_r \rangle :: \omega_r$. From $j \in \mathbb{B}_{\mathcal{R}}^{h,j',n}$ and $\omega_{j'} = \langle a, imr_r \rangle :: \omega_r$ and the induction hypothesis, we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = imr_r$ and $\omega_j = \omega_r$.

- $h = s; h'$

  From item (5) of defn. (6.1), we have $\mathbb{B}_{\mathcal{R}}^{h,i,n+1} = \bigcup \{ \mathbb{B}_{\mathcal{R}}^{h',k,n} \mid k \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n} \}$. From $\mathbb{B}_{\mathcal{R}}^{h,i,n+1} = \bigcup \{ \mathbb{B}_{\mathcal{R}}^{h',k,n} \mid k \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n} \}$ and $j \in \mathbb{B}_{\mathcal{R}}^{h,i,n+1}$, we have $\exists j' \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}$ such that $j \in \mathbb{B}_{\mathcal{R}}^{h',j',n}$. From $i \in \mathbb{A}_{\mathcal{R}}^{h}$ and $h = s; h'$, we have $i \in \mathbb{A}_{\mathcal{R}}^{s;h'}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;h'}$ and $j' \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}$ and the induction hypothesis, we have $\langle h', imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and $imr_{j'} \in \phi_s(imr_i)$ and $\omega_{j'} = \omega_i$. From $\langle h', imr_{j'}, \widehat{\overline{I}}_{j'}, \omega_{j'} \rangle \in \mathcal{R}$ and defn. (6.2), we have $j' \in \mathbb{A}_{\mathcal{R}}^{h'}$. From $\omega_{j'} = \omega_i$ and $\omega_i = \langle a, imr_r \rangle :: \omega_r$, we have $\omega_{j'} = \langle a, imr_r \rangle :: \omega_r$. From $j' \in \mathbb{A}_{\mathcal{R}}^{h'}$ and $j \in \mathbb{B}_{\mathcal{R}}^{h',j',n}$ and $\omega_{j'} = \langle a, imr_r \rangle :: \omega_r$ and the induction hypothesis, we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = imr_r$ and $\omega_j = \omega_r$. □

Lemma 6.4 $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;a} : \mathbb{B}_{\mathcal{R}}^{s;a,i} \subseteq \mathbb{A}_{\mathcal{R}}^{a}$

*Proof.* Let $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}$ and item (1) of lemma (6.3), we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$. From $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and defn. (6.2), we have $j \in \mathbb{A}_{\mathcal{R}}^{a}$. □

## 6.2 Well-formedness of Constructed Types

In this section, we prove that the types constructed from the reachable state-space are well formed in the sense of defn. (4.1), namely, lemmas (6.7), (6.9), and (6.10) prove that there exists some context in which each main part $m$, each handler part $h$, and each statement $s$ will begin executing, while lemmas (6.11) and (6.12) prove that, in each context in which each statement $s$ and each handler part $h$ can begin executing, there exists some context in which they will finish executing.

Definition 6.5 Define:

$$
\begin{array}{lrll}
1. & \eta_s(imr, \widehat{\overline{I}}) & = & (\chi_s(imr), \gamma(imr, \widehat{\overline{I}})) \quad \text{if } s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',\ \text{or } x := e \\
2. & \eta_s(imr, \widehat{\overline{I}}) & = & \eta_{s_1}(imr, \widehat{\overline{I}}) \quad \text{if } s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \\
3. & \eta_s(imr, \widehat{\overline{I}}) & = & \eta_{s_2} \circ \eta_{s_1}(imr, \widehat{\overline{I}}) \quad \text{if } s = s_1; s_2
\end{array}
$$

Lemma 6.5 If $Q \in \mathcal{R}$ and $Q.a = s; a$, then $\exists Q' \in \mathcal{R} : Q'.a = a$.

*Proof.* Suppose $Q = \langle s; a, imr, \widehat{\overline{I}}, \omega \rangle$. Choose $Q' = \langle a, imr', \widehat{\overline{J}}, \omega \rangle$ where $(imr', \widehat{\overline{J}}) = \eta_s(imr, \widehat{\overline{I}})$. The proof is by induction on the structure of $s$. There are 3 cases:

- $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr,\ \text{or } x := e$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and rules (3.6)–(3.9), we have $\langle a, imr', \widehat{\overline{J}}, \omega \rangle \in \mathcal{R}$ where $(imr', \widehat{\overline{J}}) = (\chi_s(imr), \gamma(imr, \widehat{\overline{I}}))$. From $(imr', \widehat{\overline{J}}) = (\chi_s(imr), \gamma(imr, \widehat{\overline{I}}))$ and item (1) of defn. (6.5), we have $(imr', \widehat{\overline{J}}) = \eta_s(imr, \widehat{\overline{I}})$.

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and rule (3.4), we have $\langle s_1; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$. From $\langle s_1; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and the induction hypothesis, we have $\langle a, imr', \widehat{\overline{J}}, \omega \rangle \in \mathcal{R}$ where $(imr', \widehat{\overline{J}}) = \eta_{s_1}(imr, \widehat{\overline{I}})$. From $(imr', \widehat{\overline{J}}) = \eta_{s_1}(imr, \widehat{\overline{I}})$ and item (2) of defn. (6.5), we have $(imr', \widehat{\overline{J}}) = \eta_s(imr, \widehat{\overline{I}})$.

- $s = s_1; s_2$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and $(s_1; s_2); a = s_1; (s_2; a)$, we have $\langle s_1; (s_2; a), imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$. From $\langle s_1; (s_2; a), imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and the induction hypothesis, we have $\langle s_2; a, imr', \widehat{\overline{J}}, \omega \rangle \in \mathcal{R}$ where $(imr', \widehat{\overline{J}}) = \eta_{s_1}(imr, \widehat{\overline{I}})$. From $\langle s_2; a, imr', \widehat{\overline{J}}, \omega \rangle \in \mathcal{R}$ and the induction hypothesis, we have $\langle a, imr'', \widehat{\overline{K}}, \omega \rangle \in \mathcal{R}$ where $(imr'', \widehat{\overline{K}}) = \eta_{s_2}(imr', \widehat{\overline{J}})$. From $(imr', \widehat{\overline{J}}) = \eta_{s_1}(imr, \widehat{\overline{I}})$ and $(imr'', \widehat{\overline{K}}) = \eta_{s_2}(imr', \widehat{\overline{J}})$ and item (3) of defn. (6.5), we have $(imr'', \widehat{\overline{K}}) = \eta_s(imr, \widehat{\overline{I}})$. □

Definition 6.6 Define:

$$
\begin{array}{lrll}
1. & \epsilon(s) & = & t \quad \text{if } s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',\ \text{or } x := e \\
2. & \epsilon(s) & = & \epsilon(s_1) \quad \text{if } s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \\
3. & \epsilon(s) & = & \epsilon(s_1) + \epsilon(s_2) \quad \text{if } s = s_1; s_2
\end{array}
$$

Lemma 6.6 If $Q \in \mathcal{R}$ and $Q.a = s;a$ and $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$, then $\exists Q' \in \mathcal{R} : Q'.a = a \wedge Q'.\widehat{\overline{I}}(u) = Q.\widehat{\overline{I}}(u) + \epsilon(s)$.

*Proof.* Suppose $Q = \langle s; a, imr, \widehat{\overline{I}}, \omega \rangle$. The proof is by induction on the structure of $s$. There are 3 cases:

- $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr,$ or $x := e$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and rules (3.6)–(3.9), we have $\langle a, \chi_s(imr), \widehat{\overline{J}}, \omega \rangle \in \mathcal{R}$ where $\widehat{\overline{J}} = \gamma(imr, \widehat{\overline{I}})$. From $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$, we have $u \notin \widehat{\mathcal{L}}(imr, \widehat{\overline{I}})$. From $\widehat{\overline{J}} = \gamma(imr, \widehat{\overline{I}})$ and $u \notin \widehat{\mathcal{L}}(imr, \widehat{\overline{I}})$ and defn. (3.2), we have $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + t$. From $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + t$ and item (1) of defn. (6.6), it follows that $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s)$.

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and rule (3.4), we have $\langle s_1; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$. From $\langle s_1; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and the induction hypothesis, we have $\langle a, imr', \widehat{\overline{J}}, \omega' \rangle \in \mathcal{R}$ where $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1)$. From $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1)$ and item (2) of defn. (6.6), it follows that $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s)$.

- $s = s_1; s_2$

  From $\langle s; a, imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and $(s_1; s_2); a = s_1; (s_2; a)$, we have $\langle s_1; (s_2; a), imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$. From $\langle s_1; (s_2; a), imr, \widehat{\overline{I}}, \omega \rangle \in \mathcal{R}$ and $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and the induction hypothesis, we have $\langle s_2; a, imr', \widehat{\overline{J}}, \omega' \rangle \in \mathcal{R}$ where $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1)$. From $\langle s_2; a, imr', \widehat{\overline{J}}, \omega' \rangle \in \mathcal{R}$ and $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and the induction hypothesis, we have $\langle a, imr'', \widehat{\overline{K}}, \omega' \rangle \in \mathcal{R}$ where $\widehat{\overline{K}}(u) = \widehat{\overline{J}}(u) + \epsilon(s_2)$. From $\widehat{\overline{K}}(u) = \widehat{\overline{J}}(u) + \epsilon(s_2)$ and $\widehat{\overline{J}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1)$, we have $\widehat{\overline{K}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1) + \epsilon(s_2)$. From $\widehat{\overline{K}}(u) = \widehat{\overline{I}}(u) + \epsilon(s_1) + \epsilon(s_2)$ and item (3) of defn. (6.6), it follows that $\widehat{\overline{K}}(u) = \widehat{\overline{I}}(u) + \epsilon(s)$. □

Lemma 6.7 **(Reachability, $\mathbb{A}^m_{\mathcal{R}}$)** $|\mathbb{A}^m_{\mathcal{R}}| > 1$.

*Proof.* Recall from defn. (6.2) that $\mathbb{A}^m_{\mathcal{R}} = \{\ i \mid \langle m, imr_i, \widehat{\overline{I}}_i, \omega_i\rangle \in \mathcal{R}\ \}$. We will prove that $\exists Q \in \mathcal{R} : Q.a = m$ from which it will follow that $|\mathbb{A}^m_{\mathcal{R}}| > 1$. The proof is by induction on the nesting level of $m$. Define the nesting level inductively as follows:

- The nesting level of $m_0$ is 0.
- If $m$ is of the form $s; m'$ and the nesting level of $m$ is $n$, then the nesting level of $m'$ is $n+1$.

Base case ($n = 0$): To prove that $\exists Q \in \mathcal{R} :\ Q.a = m_0$. Choose $Q = Q_0$. From defn. (3.3), we have $Q_0 \in \mathcal{R}$. From $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil}\rangle$, we have $Q_0.a = m_0$.
Induction step: Suppose $m$ is of the form $s; m'$ and the nesting level of $m$ is $n$. To prove: $\exists Q' \in \mathcal{R} : Q'.a = m'$. From the induction hypothesis, we have $\exists Q \in \mathcal{R} : Q.a = m$. From $Q \in \mathcal{R}$ and $Q.a = m$ and the hypothesis that $m$ is of the form $s; m'$ and lemma (6.5), it follows that $\exists Q' \in \mathcal{R} : Q'.a = m'$. □

Lemma 6.8 **(Reachability, $\overline{\mathbf{h}}$)**

If $\mathcal{R}$ is finite then $\forall u \in 1..N : \exists Q \in \mathcal{R} : Q.a = \overline{h}(u)$.

*Proof.* Let $u \in 1..N$. Suppose for the sake of contradiction that $\nexists Q \in \mathcal{R} : Q.a = \overline{h}(u)$. Then, we have $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$. [Suppose for the sake of contradiction that $\exists Q'' \in \mathcal{R} : u \in \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$. From $Q'' \in \mathcal{R}$ and $u \in \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and rule (3.1), $\exists Q \in \mathcal{R} : Q.a = \overline{h}(u)$, a contradiction.] We will construct an infinite sequence of abstract program states $[Q^i]_{i \in \mathbf{N}}$ such that $\forall i \in \mathbf{N} : Q^i \in \mathcal{R} \wedge Q^i.a = \mathsf{loop}\ s$ and prove that $\forall i \in \mathbf{N} : Q^{i+1}.\widehat{\overline{I}}(u) > Q^i.\widehat{\overline{I}}(u)$ whence we will have $\forall i, i' \in \mathbf{N} : Q^i \neq Q^{i'}$. From $\forall i \in \mathbf{N} : Q^i \in \mathcal{R}$ and $\forall i, i' \in \mathbf{N} :\ Q^i \neq Q^{i'}$, we will have that $\mathcal{R}$ is infinite, a contradiction. Construct the sequence inductively as follows:
Base case: From lemma (6.7), $|\mathbb{A}^{\mathsf{loop}\ s}_{\mathcal{R}}| > 1$ whence $\exists Q \in \mathcal{R} : Q.a = \mathsf{loop}\ s$. Choose $Q^0 = Q$. Clearly, $Q^0 \in \mathcal{R}$ and $Q^0.a = \mathsf{loop}\ s$.

Induction step: Suppose that $Q^n \in \mathcal{R}$ and $Q^n.a = \mathsf{loop}\ s$. From rule (3.3), $\exists Q \in \mathcal{R} : Q.a = s; \mathsf{loop}\ s$. From $Q \in \mathcal{R}$ and $Q.a = s; \mathsf{loop}\ s$ and $\forall Q'' \in \mathcal{R} : u \notin \widehat{\mathcal{L}}(Q''.imr, Q''.\widehat{\overline{I}})$ and lemma (6.6), $\exists Q' \in \mathcal{R} : Q'.a = \mathsf{loop}\ s \wedge Q'.\widehat{\overline{I}}(u) = Q.\widehat{\overline{I}}(u) + \epsilon(s)$. Choose $Q^{n+1} = Q'$. Clearly, $Q^{n+1} \in \mathcal{R}$ and $Q^{n+1}.a = \mathsf{loop}\ s$. From defn. (6.6) and the assumption that $t > 0$, it is easy to see that $\epsilon(s) > 0$. From $Q^{n+1}.\widehat{\overline{I}}(u) = Q^n.\widehat{\overline{I}}(u) + \epsilon(s)$ and $\epsilon(s) > 0$, it follows that $Q^{n+1}.\widehat{\overline{I}}(u) > Q^n.\widehat{\overline{I}}(u)$. □

Lemma 6.9 **(Reachability, $\mathbb{A}^h_{\mathcal{R}}$)** If $\mathcal{R}$ is finite then $|\mathbb{A}^h_{\mathcal{R}}| > 1$.

*Proof.* Recall from defn. (6.2) that $\mathbb{A}^h_{\mathcal{R}} = \{\ i \mid \langle h, imr_i, \widehat{\overline{I}}_i, \omega_i \rangle \in \mathcal{R}\ \}$. We will prove that $\exists Q \in \mathcal{R} : Q.a = h$ from which it will follow that $|\mathbb{A}^h_{\mathcal{R}}| > 1$. The proof is by induction on the nesting level of $h$. Define the nesting level inductively as follows:

- The nesting level of $\overline{h}(u)$ for all $u \in 1..N$ is 0.
- If $h$ is of the form $s; h'$ and the nesting level of $h$ is $n$, then the nesting level of $h'$ is $n + 1$.

Base case ($n = 0$): From the hypothesis that $\mathcal{R}$ is finite and lemma (6.8), we have $\forall u \in 1..N :\ \exists Q \in \mathcal{R} :\ Q.a = \overline{h}(u)$.
Induction step: Suppose $h$ is of the form $s; h'$ and the nesting level of $h$ is $n$. To prove: $\exists Q' \in \mathcal{R} : Q'.a = h'$. From the induction hypothesis, we have $\exists Q \in \mathcal{R} : Q.a = h$. From $Q \in \mathcal{R}$ and $Q.a = h$ and the hypothesis that $h$ is of the form $s; h'$ and lemma (6.5), it follows that $\exists Q' \in \mathcal{R} : Q'.a = h'$. □

Lemma 6.10 **(Reachability, $\mathbb{A}^{s;a}_{\mathcal{R}}$)** If $\mathcal{R}$ is finite then $|\mathbb{A}^{s;a}_{\mathcal{R}}| > 1$.

*Proof.* Note that $s; a$ must be either $m$ or $h$. If $s; a$ is $m$ then the lemma follows from lemma (6.7). If $s; a$ is $h$ then the lemma follows from lemma (6.9). □

Lemma 6.11 **(Reachability, $\mathbb{B}^{s;a,i}_{\mathcal{R}}$)** If $\mathcal{R}$ is finite then $\forall i \in \mathbb{A}^{s;a}_{\mathcal{R}} : |\mathbb{B}^{s;a,i}_{\mathcal{R}}| > 1$.

*Proof.* By induction on the structure of $s$. Let $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$.

Base case ($s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$):

From $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and item (2) of lemma (6.2), we have $\exists j \in \mathbb{B}^{s;a,i}_{\mathcal{R}} : \widehat{\overline{I}}_j = \gamma(imr_i, \widehat{\overline{I}}_i)$ whence $|\mathbb{B}^{s;a,i}_{\mathcal{R}}| > 1$.

Induction step: There are 2 cases:

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and item (3) of lemma (6.2), we have $\mathbb{B}^{s;a,i}_{\mathcal{R}} = \mathbb{B}^{s_1;a,i}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a,i}_{\mathcal{R}}$. From $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and item (1) of lemma (6.2), we have $i \in \mathbb{A}^{s_1;a}_{\mathcal{R}}$. From $i \in \mathbb{A}^{s_1;a}_{\mathcal{R}}$ and the induction hypothesis, we have $|\mathbb{B}^{s_1;a,i}_{\mathcal{R}}| > 1$. From $\mathbb{B}^{s;a,i}_{\mathcal{R}} = \mathbb{B}^{s_1;a,i}_{\mathcal{R}} \cup \mathbb{B}^{s_2;a,i}_{\mathcal{R}}$ and $|\mathbb{B}^{s_1;a,i}_{\mathcal{R}}| > 1$, we have $|\mathbb{B}^{s;a,i}_{\mathcal{R}}| > 1$.

- $s = s_1; s_2$

  From $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and item (4) of lemma (6.2), we have $\mathbb{B}^{s;a,i}_{\mathcal{R}} = \bigcup \{\ \mathbb{B}^{s_2;a,j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}\ \}$. From $i \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $s = s_1; s_2$ and $(s_1; s_2); a = s_1; (s_2; a)$, we have $i \in \mathbb{A}^{s_1;(s_2;a)}_{\mathcal{R}}$. From $i \in \mathbb{A}^{s_1;(s_2;a)}_{\mathcal{R}}$ and the induction hypothesis, we have $|\mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}| > 1$. We will next prove that $\forall j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}} : |\mathbb{B}^{s_2;a,j}_{\mathcal{R}}| > 1$. From $\mathbb{B}^{s;a,i}_{\mathcal{R}} = \bigcup \{\ \mathbb{B}^{s_2;a,j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}\ \}$ and $|\mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}| > 1$ and $\forall j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}} : |\mathbb{B}^{s_2;a,j}_{\mathcal{R}}| > 1$, we will have $|\mathbb{B}^{s;a,i}_{\mathcal{R}}| > 1$.

  To prove: $\forall j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}} : |\mathbb{B}^{s_2,j}_{\mathcal{R}}| > 1$. Let $j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}$. From $i \in \mathbb{A}^{s_1;(s_2;a)}_{\mathcal{R}}$ and lemma (6.4), we have $\mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}} \subseteq \mathbb{A}^{s_2;a}_{\mathcal{R}}$. From $j \in \mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}}$ and $\mathbb{B}^{s_1;(s_2;a),i}_{\mathcal{R}} \subseteq \mathbb{A}^{s_2;a}_{\mathcal{R}}$, we have $j \in \mathbb{A}^{s_2;a}_{\mathcal{R}}$. From $j \in \mathbb{A}^{s_2;a}_{\mathcal{R}}$ and the induction hypothesis, we have $|\mathbb{B}^{s_2;a,j}_{\mathcal{R}}| > 1$. □

Lemma 6.12 **(Reachability, $\mathbb{B}^{h,i}_{\mathcal{R}}$)** If $\mathcal{R}$ is finite then $\forall i \in \mathbb{A}^{h}_{\mathcal{R}} : |\mathbb{B}^{h,i}_{\mathcal{R}}| > 1$.

*Proof.* By induction on the structure of $h$. Let $i \in \mathbb{A}^{h}_{\mathcal{R}}$.

Base case ($h = \mathsf{iret}_v$):

From $i \in \mathbb{A}^{h}_{\mathcal{R}}$ and item (6) of lemma (6.2), we have $\exists j \in \mathbb{B}^{h,i}_{\mathcal{R}} : \widehat{\overline{I}}_j = \gamma_v(imr_i, \widehat{\overline{I}}_i)$ whence $|\mathbb{B}^{h,i}_{\mathcal{R}}| > 1$.

Induction step ($h = s;h'$):

From $i \in \mathbb{A}^h_{\mathcal{R}}$ and item (7) of lemma (6.2), we have $\mathbb{B}^{h,i}_{\mathcal{R}} = \bigcup \{ \mathbb{B}^{h',j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} \}$. From $i \in \mathbb{A}^h_{\mathcal{R}}$ and $h = s;h'$, we have $i \in \mathbb{A}^{s;h'}_{\mathcal{R}}$. From the hypothesis that $\mathcal{R}$ is finite and $i \in \mathbb{A}^{s;h'}_{\mathcal{R}}$ and lemma (6.11), we have $|\mathbb{B}^{s;h',i}_{\mathcal{R}}| > 1$. We will next prove that $\forall j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} : |\mathbb{B}^{h',j}_{\mathcal{R}}| > 1$. From $\mathbb{B}^{h,i}_{\mathcal{R}} = \bigcup \{ \mathbb{B}^{h',j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} \}$ and $|\mathbb{B}^{s;h',i}_{\mathcal{R}}| > 1$ and $\forall j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} : |\mathbb{B}^{h',j}_{\mathcal{R}}| > 1$, we will have $|\mathbb{B}^{h,i}_{\mathcal{R}}| > 1$.

To prove: $\forall j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} : |\mathbb{B}^{h',j}_{\mathcal{R}}| > 1$. Let $j \in \mathbb{B}^{s;h',i}_{\mathcal{R}}$. From $i \in \mathbb{A}^{s;h'}_{\mathcal{R}}$ and lemma (6.4), we have $\mathbb{B}^{s;h',i}_{\mathcal{R}} \subseteq \mathbb{A}^{h'}_{\mathcal{R}}$. From $j \in \mathbb{B}^{s;h',i}_{\mathcal{R}}$ and $\mathbb{B}^{s;h',i}_{\mathcal{R}} \subseteq \mathbb{A}^{h'}_{\mathcal{R}}$, we have $j \in \mathbb{A}^{h'}_{\mathcal{R}}$. From $j \in \mathbb{A}^{h'}_{\mathcal{R}}$ and the induction hypothesis, we have $|\mathbb{B}^{h',j}_{\mathcal{R}}| > 1$. □

We are now ready to show that each type constructed from the reachable state-space is well formed.

Lemma 6.13 **(Well-formedness of Types)**

If $\mathcal{R}$ is finite then all of $type_{\mathcal{R}}(s)$, $type_{\mathcal{R}}(m)$, $type_{\mathcal{R}}(h)$, and $\overline{\tau}_{\mathcal{R}}$ are well formed.

*Proof.* We have to prove conditions (1)–(4) in defn. (4.1).

Proof of condition (1):

Let $type_{\mathcal{R}}(s) = \bigwedge_{i \in \mathbb{A}^{s;a}_{\mathcal{R}}} (\alpha_i \longrightarrow \bigvee_{j \in \mathbb{B}^{s;a,i}_{\mathcal{R}}} \alpha_j)$. From the hypothesis that $\mathcal{R}$ is finite and lemma (6.10) and lemma (6.11), we have $|\mathbb{A}^{s;a}_{\mathcal{R}}| > 1$ and $\forall i \in \mathbb{A}^{s;a}_{\mathcal{R}} : |\mathbb{B}^{s;a,i}_{\mathcal{R}}| > 1$.

Proof of condition (2):

Let $type_{\mathcal{R}}(m) = \bigvee_{i \in \mathbb{A}^m_{\mathcal{R}}} \alpha_i$. From lemma (6.7), we have $|\mathbb{A}^m_{\mathcal{R}}| > 1$.

Proof of condition (3):

Let $type_{\mathcal{R}}(h) = \bigwedge_{i \in \mathbb{A}^h_{\mathcal{R}}} (\alpha_i \longrightarrow \bigvee_{j \in \mathbb{B}^{h,i}_{\mathcal{R}}} \widehat{\overline{I}}_j)$. From the hypothesis that $\mathcal{R}$ is finite and lemma (6.9) and lemma (6.12), we have $|\mathbb{A}^h_{\mathcal{R}}| > 1$ and $\forall i \in \mathbb{A}^h_{\mathcal{R}} : |\mathbb{B}^{h,i}_{\mathcal{R}}| > 1$.

Proof of condition (4):

Let $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$. From $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A} (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and defn. (6.3), we have $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A} (\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$. From $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A} (\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and the proof of condition (3) above, we have $type_{\mathcal{R}}(\overline{h}(u))$ is well formed whence $|A| > 1$ and $\forall i \in A : |B_i| > 1$. □

## 6.3 Type Derivations

In this section, we prove that each statement, each main part, and each handler part in the program has a valid type derivation, namely, lemmas (6.16) and (6.17) and (6.18) prove that $\overline{\tau}_{\mathcal{R}} \vdash s : type_{\mathcal{R}}(s)$ and $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m) \vdash m$ and $\overline{\tau}_{\mathcal{R}} \vdash h : type_{\mathcal{R}}(h)$, respectively.
Lemma (6.14) is used in the proof of lemma (6.16) to show that, for primitive statement $s$, $type_{\mathcal{R}}(s)$ satisfies the second side-condition of rule (4.1).

Lemma 6.14 If $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$ and $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$, then $\exists j \in \mathcal{U}$ such that:

$(a)\ \overline{\tau}_{\mathcal{R}}(u) = \ ... \bigwedge (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}} \widehat{\overline{I}}_k) \bigwedge \ ...$

$(b)\ \forall k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j} : \begin{cases} i.\ \ k \in \mathbb{A}_{\mathcal{R}}^{s;a} \text{ and } imr_k = imr_i \\ ii.\ \ \mathbb{B}_{\mathcal{R}}^{s;a,k} \subseteq \mathbb{B}_{\mathcal{R}}^{s;a,i} \end{cases}$

*Proof.* From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and item (2) of lemma (6.1), we have $\langle \overline{h}(u), imr_j, \widehat{\overline{I}}_j,\ \omega_j \rangle \in \mathcal{R}$ and $imr_j = \psi_u(imr_i) \wedge \widehat{\overline{I}}_j = \widehat{\overline{I}}_i \wedge \omega_j = \langle s; a, imr_i \rangle :: \omega_i$. From $\langle \overline{h}(u), imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and defn. (6.2), we have $j \in \mathbb{A}_{\mathcal{R}}^{\overline{h}(u)}$.
Proof of $(a)$: From $j \in \mathbb{A}_{\mathcal{R}}^{\overline{h}(u)}$ and defn. (6.3), we have $type_{\mathcal{R}}(\overline{h}(u)) = ... \bigwedge (imr_j, \widehat{\overline{I}}_j \longrightarrow \bigvee_{k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}} \widehat{\overline{I}}_k) \bigwedge ....$ From $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and defn. (3.1), we have $imr_i(0) = imr_i(u) = 1$. From $type_{\mathcal{R}}(\overline{h}(u)) = ... \bigwedge (imr_j, \widehat{\overline{I}}_j \longrightarrow \bigvee_{k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}} \widehat{\overline{I}}_k) \bigwedge ...$ and $imr_j = \psi_u(imr_i)$ and $imr_i(0) = imr_i(u) = 1$ and $\widehat{\overline{I}}_j = \widehat{\overline{I}}_i$ and defn. (6.3), we have $\overline{\tau}_{\mathcal{R}}(u) = \ ... \bigwedge (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}} \widehat{\overline{I}}_k) \bigwedge \ ....$
Proof of $(b)$: Let $k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}$.

- Proof of $(b)$ $(i)$: From $j \in \mathbb{A}_{\mathcal{R}}^{\overline{h}(u)}$ and $k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}$ and $\omega_j = \langle s; a, imr_i \rangle :: \omega_i$ and lemma (6.3), we have $\langle s; a, imr_k, \widehat{\overline{I}}_k, \omega_k \rangle \in \mathcal{R}$ and $imr_k = imr_i$. From $\langle s; a, imr_k, \widehat{\overline{I}}_k, \omega_k \rangle \in \mathcal{R}$ and defn. (6.2), we have $k \in \mathbb{A}_{\mathcal{R}}^{s;a}$.
- Proof of $(b)$ $(ii)$: From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $k \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j}$ and item (1) of lemma (6.2), we have $\mathbb{B}_{\mathcal{R}}^{s;a,k} \subseteq \mathbb{B}_{\mathcal{R}}^{s;a,i}$. □

Lemma (6.15) is used in the proof of lemma (6.18) to show that, when $h = \mathsf{iret}$, $type_{\mathcal{R}}(h)$ satisfies the second side-condition of rule (4.10).

Lemma 6.15 If $h = \mathsf{iret}$ and $i \in \mathbb{A}^h_{\mathcal{R}}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$, then $\exists j \in \mathcal{U}$ such that:

$(a)\ \overline{\tau}_{\mathcal{R}}(u) = \ldots \bigwedge (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}} \widehat{\overline{I}}_k) \bigwedge \ldots$

$(b)\ \forall k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}} : \begin{cases} i.\ \ k \in \mathbb{A}^h_{\mathcal{R}} \text{ and } imr_k = imr_i \\ ii.\ \ \mathbb{B}^{h,k}_{\mathcal{R}} \subseteq \mathbb{B}^{h,i}_{\mathcal{R}} \end{cases}$

*Proof.* From $i \in \mathbb{A}^h_{\mathcal{R}}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and item (4) of lemma (6.1), we have $\langle \overline{h}(u), imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and $imr_j = \psi_u(imr_i) \wedge \widehat{\overline{I}}_j = \widehat{\overline{I}}_i \wedge \omega_j = \langle h, imr_i \rangle :: \omega_i$. From $\langle \overline{h}(u), imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ and defn. (6.2), we have $j \in \mathbb{A}^{\overline{h}(u)}_{\mathcal{R}}$.
Proof of $(a)$: From $j \in \mathbb{A}^{\overline{h}(u)}_{\mathcal{R}}$ and defn. (6.3), we have $type_{\mathcal{R}}(\overline{h}(u)) = \ldots \bigwedge (imr_j, \widehat{\overline{I}}_j \longrightarrow \bigvee_{k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}} \widehat{\overline{I}}_k) \bigwedge \ldots$. From $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and defn. (3.1), we have $imr_i(0) = imr_i(u) = 1$. From $type_{\mathcal{R}}(\overline{h}(u)) = \ldots \bigwedge (imr_j, \widehat{\overline{I}}_j \longrightarrow \bigvee_{k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}} \widehat{\overline{I}}_k) \bigwedge \ldots$ and $imr_j = \psi_u(imr_i)$ and $imr_i(0) = imr_i(u) = 1$ and $\widehat{\overline{I}}_j = \widehat{\overline{I}}_i$ and defn. (6.3), we have $\overline{\tau}_{\mathcal{R}}(u) = \ldots \bigwedge (\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}} \widehat{\overline{I}}_k) \bigwedge \ldots$.
Proof of $(b)$: Let $k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}$.

- Proof of $(b)$ $(i)$: From $j \in \mathbb{A}^{\overline{h}(u)}_{\mathcal{R}}$ and $k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}$ and $\omega_j = \langle h, imr_i \rangle :: \omega_i$ and lemma (6.3), we have $\langle h, imr_k, \widehat{\overline{I}}_k, \omega_k \rangle \in \mathcal{R}$ and $imr_k = imr_i$. From $\langle h, imr_k, \widehat{\overline{I}}_k, \omega_k \rangle \in \mathcal{R}$ and defn. (6.2), we have $k \in \mathbb{A}^h_{\mathcal{R}}$.
- Proof of $(b)$ $(ii)$: From $i \in \mathbb{A}^h_{\mathcal{R}}$ and $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $k \in \mathbb{B}^{\overline{h}(u),j}_{\mathcal{R}}$ and item (5) of lemma (6.2), we have $\mathbb{B}^{h,k}_{\mathcal{R}} \subseteq \mathbb{B}^{h,i}_{\mathcal{R}}$. □

We are now ready to prove that each statement, each main part, and each handler part in the program has a valid type derivation.

Lemma 6.16 If $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ then $\overline{\tau}_{\mathcal{R}} \vdash s : type_{\mathcal{R}}(s)$.

*Proof.* By induction on the structure of $s$. There are 3 cases:

- $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$

  From defn. (6.3), we have $type_{\mathcal{R}}(s) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}} imr_j, \widehat{\overline{I}}_j)$. To prove:

  1. $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;a} \ :\ \forall j \in \mathbb{B}_{\mathcal{R}}^{s;a,i} \ :\ imr_j = \chi_s(imr_i)$
  2. $type_{\mathcal{R}}(s)$ is safe w.r.t. $\overline{\tau}$

  From (1) and (2) and rule (4.1), it will follow that $\overline{\tau}_{\mathcal{R}} \vdash s \ :\ type_{\mathcal{R}}(s)$, namely, (1) and (2) will ensure that the side-conditions of rule (4.1) are satisfied.

  To prove (1), let $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}$ and lemma (6.3), we have $imr_j \in \phi_s(imr_i)$. From $imr_j \in \phi_s(imr_i)$ and item (1) of defn. (6.4), we have $imr_j = \chi_s(imr_i)$.

  To prove (2), we prove conditions (1) and (2) in defn. (4.2).

  Proof of condition (1): Condition (1) follows from lemma (6.14).

  Proof of condition (2): To prove $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;a} : \exists j \in \mathbb{B}_{\mathcal{R}}^{s;a,i} : \widehat{\overline{I}}_j = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \widehat{\overline{I}}_j < \overline{d}$. Let $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and item (2) of lemma (6.2), we have $\exists j \in \mathbb{B}_{\mathcal{R}}^{s;a,i} : \widehat{\overline{I}}_j = \gamma(imr_i, \widehat{\overline{I}}_i)$. We next prove that $\widehat{\overline{I}}_j < \overline{d}$. From $i \in \mathbb{A}_{\mathcal{R}}^{s;a}$ and $j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}$ and lemma (6.3), we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$, we have $\widehat{\overline{I}}_j < \overline{d}$.

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From defn. (6.3), we have $type_{\mathcal{R}}(s) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}} imr_j, \widehat{\overline{I}}_j)$. From the induction hypothesis, we have:

  1. $\overline{\tau}_{\mathcal{R}} \vdash s_1 \ :\ \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s_1;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s_1;a,i}} imr_j, \widehat{\overline{I}}_j)$
  2. $\overline{\tau}_{\mathcal{R}} \vdash s_2 \ :\ \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s_2;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,i}} imr_j, \widehat{\overline{I}}_j)$

To prove:

3. $\mathbb{A}_{\mathcal{R}}^{s;a} = \mathbb{A}_{\mathcal{R}}^{s_1;a} = \mathbb{A}_{\mathcal{R}}^{s_2;a}$
4. $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;a} : \mathbb{B}_{\mathcal{R}}^{s;a,i} = \mathbb{B}_{\mathcal{R}}^{s_1;a,i} \cup \mathbb{B}_{\mathcal{R}}^{s_2;a,i}$

From (1)–(4) and rule (4.2), we will have $\overline{\tau} \vdash s : type_{\mathcal{R}}(s)$, namely, (3) will ensure that the domains of the types of $s$ and $s_1$ and $s_2$ are related in the manner specified in rule (4.2), and (4) will ensure the same for their ranges.
We have (3) from item (1) of lemma (6.2). We have (4) from item (3) of lemma (6.2).

- $s = s_1; s_2$

From defn. (6.3), we have $type_{\mathcal{R}}(s) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{(s_1;s_2);a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{(s_1;s_2);a,i}} imr_j, \widehat{\overline{I}}_j)$.
From the induction hypothesis, we have:

1. $\overline{\tau}_{\mathcal{R}} \vdash s_1 : \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s_1;(s_2;a)}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i}} imr_j, \widehat{\overline{I}}_j)$
2. $\overline{\tau}_{\mathcal{R}} \vdash s_2 : \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s_2;a}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,i}} imr_j, \widehat{\overline{I}}_j)$

To prove:

3. $\mathbb{A}_{\mathcal{R}}^{(s_1;s_2);a} = \mathbb{A}_{\mathcal{R}}^{s_1;(s_2;a)}$
4. $\forall i \in \mathbb{A}_{\mathcal{R}}^{(s_1;s_2);a} : \mathbb{B}_{\mathcal{R}}^{(s_1;s_2);a,i} = \bigcup \{ \mathbb{B}_{\mathcal{R}}^{(s_2;a),j} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i} \}$
5. $\forall i \in \mathbb{A}_{\mathcal{R}}^{s_1;(s_2;a)} : \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i} \subseteq \mathbb{A}_{\mathcal{R}}^{s_2;a}$

From (1)–(5) and rule (4.3), we will have $\overline{\tau}_{\mathcal{R}} \vdash s : type_{\mathcal{R}}(s)$, namely, (3) will ensure that the domains of the types of $s_1; s_2$ and $s_1$ are related in the manner specified in rule (4.3), (4) will ensure the same for the ranges of the types of $s_1; s_2$ and $s_2$, and (5) will ensure that the side-condition of rule (4.3) is satisfied.
We have (3) from $(s_1; s_2); a = s_1; (s_2; a)$, (4) from item (4) of lemma (6.2), and (5) from lemma (6.4). □

Lemma 6.17 If $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ then $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m) \vdash m$.

*Proof.* By induction on the structure of $m$. There are 2 cases:

- $m = \mathsf{loop}\ s$

  From defn. (6.3), we have $type_{\mathcal{R}}(m) = \bigvee_{i \in \mathbb{A}_{\mathcal{R}}^{\mathsf{loop}\ s}} imr_i, \widehat{\overline{I}}_i$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (6.16), we have:

  1. $\overline{\tau}_{\mathcal{R}} \vdash s\ :\ \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;\mathsf{loop}\ s}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;\mathsf{loop}\ s,i}} imr_j, \widehat{\overline{I}}_j)$

  To prove:

  2. $\mathbb{A}_{\mathcal{R}}^{\mathsf{loop}\ s} = \mathbb{A}_{\mathcal{R}}^{s;\mathsf{loop}\ s}$
  3. $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;\mathsf{loop}\ s}\ :\ \mathbb{B}_{\mathcal{R}}^{s;\mathsf{loop}\ s,i} \subseteq \mathbb{A}_{\mathcal{R}}^{\mathsf{loop}\ s}$

  From (1)–(3) and rule (4.8), we will have $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m) \vdash m$, namely, (2) will ensure that the domains of the types of $m$ and $s$ are related in the manner specified in rule (4.8), and (3) will ensure that the side-condition of rule (4.8) is satisfied.

  We have (2) from item (2) of lemma (6.2). We have (3) from lemma (6.4).

- $m = s; m'$

  From defn. (6.3), we have $type_{\mathcal{R}}(m) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{m}} imr_i, \widehat{\overline{I}}_i$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (6.16), we have:

  1. $\overline{\tau}_{\mathcal{R}} \vdash s\ :\ \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;m'}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;m',i}} imr_j, \widehat{\overline{I}}_j)$

  From the induction hypothesis, we have:

  2. $\overline{\tau}_{\mathcal{R}}, \bigvee_{i \in \mathbb{A}_{\mathcal{R}}^{m'}} imr_i, \widehat{\overline{I}}_i \vdash m'$

  To prove:

  3. $\forall i \in \mathbb{A}_{\mathcal{R}}^{s;m'}\ :\ \mathbb{B}_{\mathcal{R}}^{s;m',i} \subseteq \mathbb{A}_{\mathcal{R}}^{m'}$

  From (1)–(3) and rule (4.9), we will have $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m) \vdash m$, namely, (3) will ensure that the side-condition of rule (4.9) is satisfied.

  We have (3) from lemma (6.4). □

Lemma 6.18 If $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ then $\overline{\tau}_{\mathcal{R}} \vdash h : type_{\mathcal{R}}(h)$.

*Proof.* By induction on the structure of $h$. There are 2 cases:

- $h = \mathsf{iret}_v$

  From defn. (6.3), we have $type_{\mathcal{R}}(h) = \bigwedge_{i \in \mathbb{A}^h_{\mathcal{R}}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}^{h,i}_{\mathcal{R}}} \widehat{\overline{I}}_j)$. To prove:

  1. $type_{\mathcal{R}}(h)$ is safe w.r.t. $\overline{\tau}$ and $v$

  From (1) and rule (4.10), it will follow that $\overline{\tau}_{\mathcal{R}} \vdash h \ : \ type_{\mathcal{R}}(h)$, namely, (1) will ensure that the side-condition of rule (4.10) is satisfied.

  To prove (1), we prove conditions (3) and (4) in defn. (4.2).

  Proof of condition (3): Condition (3) follows from lemma (6.15).

  Proof of condition (4): To prove $\forall i \in \mathbb{A}^h_{\mathcal{R}} : \exists j \in \mathbb{B}^{h,i}_{\mathcal{R}} : \widehat{\overline{I}}_j = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \widehat{\overline{I}}_j < \overline{d}$. Let $i \in \mathbb{A}^h_{\mathcal{R}}$. From $i \in \mathbb{A}^h_{\mathcal{R}}$ and item (6) of lemma (6.2), we have $\exists j \in \mathbb{B}^{h,i}_{\mathcal{R}} : \widehat{\overline{I}}_j = \gamma_v(imr_i, \widehat{\overline{I}}_i)$. We next prove that $\widehat{\overline{I}}_j < \overline{d}$. From $i \in \mathbb{A}^h_{\mathcal{R}}$ and $j \in \mathbb{B}^{h,i}_{\mathcal{R}}$ and lemma (6.3), we have $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$ where $\omega_i = \langle a, imr_j \rangle :: \omega_j$. From $\forall Q \in \mathcal{R} : \ Q.\widehat{\overline{I}} < \overline{d}$ and $\langle a, imr_j, \widehat{\overline{I}}_j, \omega_j \rangle \in \mathcal{R}$, we have $\widehat{\overline{I}}_j < \overline{d}$.

- $h = s; h'$

  From defn. (6.3), we have $type_{\mathcal{R}}(h) = \bigwedge_{i \in \mathbb{A}^h_{\mathcal{R}}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}^{h,i}_{\mathcal{R}}} \widehat{\overline{I}}_j)$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (6.16), we have:

  1. $\overline{\tau}_{\mathcal{R}} \vdash s \ : \ \bigwedge_{i \in \mathbb{A}^{s;h'}_{\mathcal{R}}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}^{s;h',i}_{\mathcal{R}}} imr_j, \widehat{\overline{I}}_j)$

  From the induction hypothesis, we have:

  2. $\overline{\tau}_{\mathcal{R}} \vdash h' \ : \ \bigwedge_{i \in \mathbb{A}^{h'}_{\mathcal{R}}} (imr_i, \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in \mathbb{B}^{h',i}_{\mathcal{R}}} \widehat{\overline{I}}_j)$

  To prove:

  3. $\forall i \in \mathbb{A}^h_{\mathcal{R}} \ : \ \mathbb{B}^{h,i}_{\mathcal{R}} = \bigcup \{ \ \mathbb{B}^{h',j}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i}_{\mathcal{R}} \ \}$

  4. $\forall i \in \mathbb{A}^{s;h'}_{\mathcal{R}} \ : \ \mathbb{B}^{s;h',i}_{\mathcal{R}} \subseteq \mathbb{A}^{h'}_{\mathcal{R}}$

  From (1)–(4) and rule (4.11), we will have $\overline{\tau}_{\mathcal{R}} \vdash h \ : \ type_{\mathcal{R}}(h)$, namely, (3) will ensure that the ranges of the types of $h$ and $h'$ are related in the manner

specified in rule (4.11), and (4) will ensure that the side-condition of rule (4.11) is satisfied.

We have (3) from item (7) of lemma (6.2). We have (4) from lemma (6.4). □

## 6.4 Putting It All Together

We are now ready to prove that if a program is accepted by the model checker, then it type checks.

Lemma 6.19 If $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ then $\overline{\tau}_{\mathcal{R}}, \overline{0} \vdash P_0$.

*Proof.*

From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and prop. (3.1), we have $\mathcal{R}$ is finite. From $\mathcal{R}$ is finite and lemma (6.13), we have that all of $type_{\mathcal{R}}(s)$, $type_{\mathcal{R}}(m)$, $type_{\mathcal{R}}(h)$, and $\overline{\tau}_{\mathcal{R}}$ are well formed. Suppose $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$. From $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$ and defn. (6.3), we have $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A^u} (\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$ and $\forall i \in A^u : imr_i(0) = imr_i(u) = 1$. We first prove that $\overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : \overline{\tau}_{\mathcal{R}}(u)$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (6.18), we have $\overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : type_{\mathcal{R}}(\overline{h}(u))$. From $\overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : type_{\mathcal{R}}(\overline{h}(u))$ and $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A^u}(\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$ and $\forall i \in A^u : imr_i(0) = imr_i(u) = 1$ and rule (4.12), it follows that $\overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : \bigwedge_{i \in A^u}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i^u} \widehat{\overline{I}}_j)$, *i.e.*, $\overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : \overline{\tau}_{\mathcal{R}}(u)$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (6.17), we have $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m_0) \vdash m_0$. From $Q_0 \in \mathcal{R}$ and $Q_0 = \langle m_0, \neg \mathsf{t}_0, \overline{0}, \mathsf{nil} \rangle$ and defn. (6.2), we have $i_0 \in \mathbb{A}_{\mathcal{R}}^{m_0}$ with $(imr_{i_0}, \widehat{\overline{I}}_{i_0}) = (\neg \mathsf{t}_0, \overline{0})$. From $i_0 \in \mathbb{A}_{\mathcal{R}}^{m_0}$ and $(imr_{i_0}, \widehat{\overline{I}}_{i_0}) = (\neg \mathsf{t}_0, \overline{0})$ and defn. (6.3), we have $type_{\mathcal{R}}(m_0) = ... \bigvee \neg \mathsf{t}_0, \overline{0} \bigvee ....$ Finally, from $P_0 \in \{ \langle m_0, \neg \mathsf{t}_0, \overline{I}, \mathsf{nil}, \lambda x.0 \rangle \mid \overline{I} \leq \overline{0} \}$ and $\forall u \in 1..N : \overline{\tau}_{\mathcal{R}} \vdash \overline{h}(u) : \overline{\tau}_{\mathcal{R}}(u)$ and $\overline{\tau}_{\mathcal{R}}, type_{\mathcal{R}}(m_0) \vdash m_0$ and $type_{\mathcal{R}}(m_0) = ... \bigvee \neg \mathsf{t}_0, \overline{0} \bigvee ...$ and rule (4.4), we have $\overline{\tau}_{\mathcal{R}}, \overline{0} \vdash P_0$. □

Lemma 6.20 **(From Model Checking to Type Checking)**
If $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ then $\exists \overline{\tau} : \overline{\tau}, \overline{0} \vdash P_0$.

*Proof.* Combine defn. (3.3) and lemma (6.19). □

Finally, we prove our main result that a program type checks if and only if the model checker accepts it.

Theorem 6.1 **(Equivalence)** $\exists \overline{\tau} : \overline{\tau}, \overline{0} \vdash P_0$ if and only if $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$.

*Proof.* Combine lemmas (4.6) and (6.20). □

# 7. COHERENCE

It is desirable that our type system possess the property that if the domains of multiple components in an intersection type match, then the ranges of those components also match. We call this property *coherence* and state it formally as follows:

Definition 7.1 **(Type Coherence)**

1. If $S = \bigwedge_{i \in A}(\alpha_i \longrightarrow \bigvee_{j \in B_i} \alpha_j)$ then
   $\forall i, i' \in A : \ \alpha_i = \alpha_{i'} \Rightarrow \{ \alpha_j \mid j \in B_i \} = \{ \alpha_j \mid j \in B_{i'} \}$.
2. If $H = \bigwedge_{i \in A}(\alpha_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ then
   $\forall i, i' \in A : \ \alpha_i = \alpha_{i'} \Rightarrow \{ \widehat{\overline{I}}_j \mid j \in B_i \} = \{ \widehat{\overline{I}}_j \mid j \in B_{i'} \}$.
3. If $\tau = \bigwedge_{i \in A}(\widehat{\overline{I}}_i \xrightarrow{imr_i} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ then
   $\forall i, i' \in A : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{ \widehat{\overline{I}}_j \mid j \in B_i \} = \{ \widehat{\overline{I}}_j \mid j \in B_{i'} \}$.

Our proof from type checking to model checking does not require the intersection types in our type system to possess the coherence property. However, the intersection types constructed in our proof from model checking to type checking always possess this property:

Lemma 7.1 **(Model Coherence)** If $\mathcal{R}$ is finite then:

1. $\forall i, i' \in \mathbb{A}^{s;a}_{\mathcal{R}} : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i}_{\mathcal{R}} \} = \{ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i'}_{\mathcal{R}} \}$
2. $\forall i, i' \in \mathbb{A}^{h}_{\mathcal{R}} : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{ \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i}_{\mathcal{R}} \} = \{ \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i'}_{\mathcal{R}} \}$

*Proof.* Let $\mathcal{F}^n_{\mathcal{R}}$ denote $n$ applications of $\mathcal{F}_{\mathcal{R}}$. Then, $\mu\mathcal{F}_{\mathcal{R}} = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$ where $n'$ is such that $\mathcal{F}^{n'+1}_{\mathcal{R}}(\overline{\emptyset}) = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$. Define $\mathbb{B}^{a,i,n}_{\mathcal{R}} = \pi^{a,i} \ \mathcal{F}^n_{\mathcal{R}}(\overline{\emptyset})$. From $\mathbb{B}^{a,i}_{\mathcal{R}} = \pi^{a,i} \ \mu\mathcal{F}_{\mathcal{R}}$ (recall defn. (6.2)) and $\mu\mathcal{F}_{\mathcal{R}} = \mathcal{F}^{n'}_{\mathcal{R}}(\overline{\emptyset})$ and $\mathbb{B}^{a,i,n}_{\mathcal{R}} = \pi^{a,i} \ \mathcal{F}^n_{\mathcal{R}}(\overline{\emptyset})$, we have $\mathbb{B}^{a,i}_{\mathcal{R}} = \mathbb{B}^{a,i,n'}_{\mathcal{R}}$. We will prove that $\forall n$ :

1. $\forall i, i' \in \mathbb{A}^{s;a}_{\mathcal{R}} \ : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i,n}_{\mathcal{R}}\ \} = \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i',n}_{\mathcal{R}}\ \}$
2. $\forall i, i' \in \mathbb{A}^{h}_{\mathcal{R}} \ : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{\ \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i,n}_{\mathcal{R}}\ \} = \{\ \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i',n}_{\mathcal{R}}\ \}$

The proof is by induction on $n$.

Base case ($n = 0$): Trivial since $\forall i \in \mathbb{A}^{s;a}_{\mathcal{R}} \ : \ \mathbb{B}^{s;a,i,0}_{\mathcal{R}} = \emptyset$ and $\forall i \in \mathbb{A}^{h}_{\mathcal{R}} \ : \ \mathbb{B}^{h,i,0}_{\mathcal{R}} = \emptyset$.

Induction step:

Suppose $i, i' \in \mathbb{A}^{s;a}_{\mathcal{R}}$ and $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$. To prove $\{(imr_j, \widehat{\overline{I}}_j) | j \in \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}}\} = \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i',n+1}_{\mathcal{R}}\ \}$. There are 3 cases:

- $s = \mathsf{skip},\ \mathsf{ei},\ \mathsf{imr} := \mathsf{imr} \wedge imr',$ or $x := e$

  From item (1) of defn. (6.1), we have:

$$\begin{aligned}
\mathbb{B}^{s;a,i,n+1}_{\mathcal{R}} \ = \ & \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i\,, \widehat{\overline{I}}_i\,) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}\ \} \ \cup \ \{\ j^\star\ \} \text{ where} \\
& j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle s; a, imr_i \rangle :: \omega_i \\
& j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = \chi_s(imr_i) \wedge \widehat{\overline{I}}_{j^\star} = \gamma(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_i \qquad (7.1) \\
\mathbb{B}^{s;a,i',n+1}_{\mathcal{R}} \ = \ & \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}}\ \} \ \cup \ \{\ j^*\ \} \text{ where} \\
& j^\circ \in \mathcal{U} \text{ s.t. } imr_{j^\circ} = \psi_u(imr_{i'}) \wedge \widehat{\overline{I}}_{j^\circ} = \widehat{\overline{I}}_{i'} \wedge \omega_{j^\circ} = \langle s; a, imr_{i'} \rangle :: \omega_{i'} \\
& j^* \in \mathcal{U} \text{ s.t. } imr_{j^*} = \chi_s(imr_{i'}) \wedge \widehat{\overline{I}}_{j^*} = \gamma(imr_{i'}, \widehat{\overline{I}}_{i'}) \wedge \omega_{j^*} = \omega_{i'} \qquad (7.2)
\end{aligned}$$

  From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$ and (7.1)–(7.2), we have $(imr_{j^\star}, \widehat{\overline{I}}_{j^\star}) = (imr_{j^*}, \widehat{\overline{I}}_{j^*})$.

  We will prove that:

$$\begin{aligned}
& \{\ (imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i\,, \widehat{\overline{I}}_i\,) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}\ \}\ \} \subseteq \\
& \{\ (imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}}\ \}\ \} \qquad (7.3) \\
& \{\ (imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i\,, \widehat{\overline{I}}_i\,) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}\ \}\ \} \supseteq \\
& \{\ (imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\ \mathbb{B}^{s;a,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}}\ \}\ \} \qquad (7.4)
\end{aligned}$$

  From $(imr_{j^\star}, \widehat{\overline{I}}_{j^\star}) = (imr_{j^*}, \widehat{\overline{I}}_{j^*})$ and (7.1)–(7.4), we will have $\{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i,n+1}_{\mathcal{R}}\ \} = \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}^{s;a,i',n+1}_{\mathcal{R}}\ \}$.

From $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (\psi_u(imr_i), \widehat{\overline{I}}_i)$ and $(imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ}) = (\psi_u(imr_{i'}), \widehat{\overline{I}}_{i'})$ and $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$, we have $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ})$. From $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ})$ and the induction hypothesis, we have:

$$\{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n} \,\} \;=\; \{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\circ,n} \,\} \tag{7.5}$$

We now prove (7.3). Suppose $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $x \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}$. From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$, we have $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$. From $x \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}$ and (7.5), $\exists y \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\circ,n}$ such that $\widehat{\overline{I}}_x = \widehat{\overline{I}}_y$. From $x \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}$ and $\omega_{j^\bullet} = \langle s; a, imr_i \rangle :: \omega_i$ and lemma (6.3), we have $imr_x = imr_i$, and, similarly, from $y \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\circ,n}$ and $\omega_{j^\circ} = \langle s; a, imr_{i'} \rangle :: \omega_{i'}$ and lemma (6.3), we have $imr_y = imr_{i'}$. From $imr_x = imr_i$ and $imr_y = imr_{i'}$ and $imr_i = imr_{i'}$, we have $imr_x = imr_y$. Finally, from $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$ and the induction hypothesis, it follows that $\{\, (imr_k, \widehat{\overline{I}}_k) \mid\ k \in \mathbb{B}_{\mathcal{R}}^{s;a,x,n} \,\} = \{\, (imr_k, \widehat{\overline{I}}_k) \mid k \in \mathbb{B}_{\mathcal{R}}^{s;a,y,n} \,\}$, thereby proving (7.3). The proof of (7.4) is similar.

- $s = \mathsf{if0}\ x\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$

  From item (2) of defn. (6.1), we have:

$$\mathbb{B}_{\mathcal{R}}^{s;a,i,n+1} \;=\; \mathbb{B}_{\mathcal{R}}^{s_1;a,i,n} \,\cup\, \mathbb{B}_{\mathcal{R}}^{s_2;a,i,n} \tag{7.6}$$

$$\mathbb{B}_{\mathcal{R}}^{s;a,i',n+1} \;=\; \mathbb{B}_{\mathcal{R}}^{s_1;a,i',n} \,\cup\, \mathbb{B}_{\mathcal{R}}^{s_2;a,i',n} \tag{7.7}$$

  From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$$\{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;a,i,n} \,\} \;=\; \{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;a,i',n} \,\} \tag{7.8}$$

$$\{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,i,n} \,\} \;=\; \{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_2;a,i',n} \,\} \tag{7.9}$$

  From (7.6)–(7.9), it follows that $\{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;a,i,n+1} \,\} = \{\, (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;a,i',n+1} \,\}$.

- $s = s_1; s_2$

  From item (3) of defn. (6.1), we have:

$$\mathbb{B}_{\mathcal{R}}^{s;a,i,n+1} \;=\; \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n} \,\} \tag{7.10}$$

$$\mathbb{B}_{\mathcal{R}}^{s;a,i',n+1} \;=\; \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i',n} \,\} \tag{7.11}$$

We will prove that:

$$\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n} \,\}\,\} \subseteq$$
$$\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i',n} \,\}\,\} \quad (7.12)$$
$$\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n} \,\}\,\} \supseteq$$
$$\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{s_2;a,j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i',n} \,\}\,\} \quad (7.13)$$

From (7.10)–(7.13), we will have $\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \mathbb{B}_{\mathcal{R}}^{s;a,i,n+1}\,\} = \{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \mathbb{B}_{\mathcal{R}}^{s;a,i',n+1}\,\}$.

From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$$\{\,(imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n}\,\} \;=\; \{\,(imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i',n}\,\} \quad (7.14)$$

We now prove (7.12). Suppose $x \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n}$. From $x \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i,n}$ and (7.14), $\exists y \in \mathbb{B}_{\mathcal{R}}^{s_1;(s_2;a),i',n}$ such that $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$. From $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$ and the induction hypothesis, we have $\{\,(imr_k, \widehat{\overline{I}}_k) \mid k \in \mathbb{B}_{\mathcal{R}}^{s_2;a,x,n}\,\} = \{(imr_k, \widehat{\overline{I}}_k) \mid k \in \mathbb{B}_{\mathcal{R}}^{s_2;a,y,n}\}$, thereby proving (7.12). The proof of (7.13) is similar.

Suppose $i, i' \in \mathbb{A}_{\mathcal{R}}^h$ and $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$. To prove $\{\widehat{\overline{I}}_j \mid j \in \mathbb{B}_{\mathcal{R}}^{h,i,n+1}\,\} = \{\widehat{\overline{I}}_j \mid j \in \mathbb{B}_{\mathcal{R}}^{h,i',n+1}\,\}$. There are 2 cases:

- $h = \mathsf{iret}_v$

  From item (4) of defn. (6.1), we have:

$$\begin{aligned}
\mathbb{B}_{\mathcal{R}}^{h,i,n+1} \;=\; & \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{h,j,n} \mid u \in \widehat{\mathcal{L}}(imr_i\,, \widehat{\overline{I}}_i\,) \text{ and } j \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\bullet,n}\,\} \cup \{\, j^\star \,\} \text{ where} \\
& j^\bullet \in \mathcal{U} \text{ s.t. } imr_{j^\bullet} = \psi_u(imr_i) \wedge \widehat{\overline{I}}_{j^\bullet} = \widehat{\overline{I}}_i \wedge \omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i \\
& j^\star \in \mathcal{U} \text{ s.t. } imr_{j^\star} = imr_r \wedge \widehat{\overline{I}}_{j^\star} = \gamma_v(imr_i, \widehat{\overline{I}}_i) \wedge \omega_{j^\star} = \omega_r \quad (7.15) \\
\mathbb{B}_{\mathcal{R}}^{h,i',n+1} \;=\; & \bigcup \{\, \mathbb{B}_{\mathcal{R}}^{h,j,n} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}_{\mathcal{R}}^{\overline{h}(u),j^\circ,n}\,\} \cup \{\, j^* \,\} \text{ where} \\
& j^\circ \in \mathcal{U} \text{ s.t. } imr_{j^\circ} = \psi_u(imr_{i'}) \wedge \widehat{\overline{I}}_{j^\circ} = \widehat{\overline{I}}_{i'} \wedge \omega_{j^\circ} = \langle h, imr_{i'} \rangle :: \omega_{i'} \\
& j^* \in \mathcal{U} \text{ s.t. } imr_{j^*} = imr_r \wedge \widehat{\overline{I}}_{j^*} = \gamma_v(imr_{i'}, \widehat{\overline{I}}_{i'}) \wedge \omega_{j^*} = \omega_r \quad (7.16)
\end{aligned}$$

where $\omega_i = \langle a, imr_r \rangle :: \omega_r$ and $\omega_{i'} = \langle a', imr_{r'} \rangle :: \omega_{r'}$. From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$ and (7.15)–(7.16), we have $\widehat{\overline{I}}_{j^\star} = \widehat{\overline{I}}_{j^*}$. We will prove that:

$$\begin{aligned}
&\{\, \widehat{\overline{I}}_k \mid k \in \bigcup \{\, \mathbb{B}^{h,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}} \,\} \,\} \subseteq \\
&\{\, \widehat{\overline{I}}_k \mid k \in \bigcup \{\, \mathbb{B}^{h,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}} \,\} \,\} \qquad (7.17) \\
&\{\, \widehat{\overline{I}}_k \mid k \in \bigcup \{\, \mathbb{B}^{h,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}} \,\} \,\} \supseteq \\
&\{\, \widehat{\overline{I}}_k \mid k \in \bigcup \{\, \mathbb{B}^{h,j,n}_{\mathcal{R}} \mid u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'}) \text{ and } j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}} \,\} \,\} \qquad (7.18)
\end{aligned}$$

From $\widehat{\overline{I}}_{j^\star} = \widehat{\overline{I}}_{j^*}$ and (7.15)–(7.18), it will follow that $\{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i,n+1}_{\mathcal{R}} \} = \{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{h,i',n+1}_{\mathcal{R}} \,\}$.

From $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (\psi_u(imr_i), \widehat{\overline{I}}_i)$ and $(imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ}) = (\psi_u(imr_{i'}), \widehat{\overline{I}}_{i'})$ and $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$, we have $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ})$. From $(imr_{j^\bullet}, \widehat{\overline{I}}_{j^\bullet}) = (imr_{j^\circ}, \widehat{\overline{I}}_{j^\circ})$ and the induction hypothesis, we have:

$$\{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}} \,\} \;=\; \{\, \widehat{\overline{I}}_j \mid j \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}} \,\} \qquad (7.19)$$

We now prove (7.17). Suppose $u \in \widehat{\mathcal{L}}(imr_i, \widehat{\overline{I}}_i)$ and $x \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}$. From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$, we have $u \in \widehat{\mathcal{L}}(imr_{i'}, \widehat{\overline{I}}_{i'})$. From $x \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}$ and (7.19), $\exists y \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}}$ such that $\widehat{\overline{I}}_x = \widehat{\overline{I}}_y$. From $x \in \mathbb{B}^{\overline{h}(u),j^\bullet,n}_{\mathcal{R}}$ and $\omega_{j^\bullet} = \langle h, imr_i \rangle :: \omega_i$ and lemma (6.3), we have $imr_x = imr_i$, and, similarly, from $y \in \mathbb{B}^{\overline{h}(u),j^\circ,n}_{\mathcal{R}}$ and $\omega_{j^\circ} = \langle h, imr_{i'} \rangle :: \omega_{i'}$ and lemma (6.3), we have $imr_y = imr_{i'}$. From $imr_x = imr_i$ and $imr_y = imr_{i'}$ and $imr_i = imr_{i'}$, we have $imr_x = imr_y$. Finally, from $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$ and the induction hypothesis, it follows that $\{\, \widehat{\overline{I}}_k \mid k \in \mathbb{B}^{h,x,n}_{\mathcal{R}} \} = \{\, \widehat{\overline{I}}_k \mid k \in \mathbb{B}^{h,y,n}_{\mathcal{R}} \,\}$, thereby proving (7.17). The proof of (7.18) is similar.

- $h = s; h'$

  From item (5) of defn. (6.1), we have:

$$\begin{aligned}
\mathbb{B}^{h,i,n+1}_{\mathcal{R}} &= \bigcup \{\, \mathbb{B}^{h',j,n}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i,n}_{\mathcal{R}} \,\} \qquad (7.20) \\
\mathbb{B}^{h,i',n+1}_{\mathcal{R}} &= \bigcup \{\, \mathbb{B}^{h',j,n}_{\mathcal{R}} \mid j \in \mathbb{B}^{s;h',i',n}_{\mathcal{R}} \,\} \qquad (7.21)
\end{aligned}$$

We will prove that:

$$\{\ \widehat{\overline{I}}_k \mid k \in \bigcup \{\ \mathbb{B}_{\mathcal{R}}^{h',j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}\ \}\ \} \subseteq$$
$$\{\ \widehat{\overline{I}}_k \mid k \in \bigcup \{\ \mathbb{B}_{\mathcal{R}}^{h',j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i',n}\ \}\ \} \tag{7.22}$$
$$\{\ \widehat{\overline{I}}_k \mid k \in \bigcup \{\ \mathbb{B}_{\mathcal{R}}^{h',j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}\ \}\ \} \supseteq$$
$$\{\ \widehat{\overline{I}}_k \mid k \in \bigcup \{\ \mathbb{B}_{\mathcal{R}}^{h',j,n} \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i',n}\ \}\ \} \tag{7.23}$$

From (7.20)–(7.23), we will have $\{\ \widehat{\overline{I}}_k \mid k \in \mathbb{B}_{\mathcal{R}}^{h,i,n+1}\ \} = \{\ \widehat{\overline{I}}_k \mid k \in \mathbb{B}_{\mathcal{R}}^{h,i',n+1}\ \}$. From $(imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'})$ and the induction hypothesis, we have:

$$\{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}\ \} \quad = \quad \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;h',i',n}\ \} \tag{7.24}$$

We now prove (7.22). Suppose $x \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}$. From $x \in \mathbb{B}_{\mathcal{R}}^{s;h',i,n}$ and (7.24), $\exists y \in \mathbb{B}_{\mathcal{R}}^{s;h',i',n}$ such that $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$. From $(imr_x, \widehat{\overline{I}}_x) = (imr_y, \widehat{\overline{I}}_y)$ and the induction hypothesis, we have $\{\ \widehat{\overline{I}}_k \mid k \in \mathbb{B}_{\mathcal{R}}^{h',x,n}\ \} = \{\ \widehat{\overline{I}}_k \mid k \in \mathbb{B}_{\mathcal{R}}^{h',y,n}\ \}$, thereby proving (7.22). The proof of (7.23) is similar. □

It follows that if a program is typable using the type system of chapter 4, then it is typable using the same type system with the additional constraint of type coherence on all intersection types as stated in defn. (7.1).

Lemma 7.2 If $\exists \overline{\tau} :\ \overline{\tau}, \overline{0} \vdash P_0$ then $P_0$ is typable with the additional constraint of type coherence on all intersection types.

*Proof.* From $\overline{\tau}, \overline{0} \vdash P_0$ and lemma (4.6), we have $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$. From $(Q_0 \hookrightarrow^n Q \Rightarrow Q.\widehat{\overline{I}} < \overline{d})$ and defn. (3.3), we have $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$. From $\forall Q \in \mathcal{R} : Q.\widehat{\overline{I}} < \overline{d}$ and lemma (3.1), we have $\mathcal{R}$ is finite. We now prove conditions (1)–(3) in defn. (7.1).

Proof of condition (1):

Let $type_{\mathcal{R}}(s) = \bigwedge_{i \in \mathbb{A}_{\mathcal{R}}^{s;a}} (\alpha_i \longrightarrow \bigvee_{j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}} \alpha_j)$. From $\mathcal{R}$ is finite and item (1) of lemma (7.1), we have $\forall i, i' \in \mathbb{A}_{\mathcal{R}}^{s;a} :\ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;a,i}\ \} = \{\ (imr_j, \widehat{\overline{I}}_j) \mid j \in \mathbb{B}_{\mathcal{R}}^{s;a,i'}\ \}$.

Proof of condition (2):

Let $type_{\mathcal{R}}(h) = \bigwedge_{i \in \mathbb{A}^h_{\mathcal{R}}} (\alpha_i \longrightarrow \bigvee_{j \in \mathbb{B}^{h,i}_{\mathcal{R}}} \widehat{\overline{I}}_j)$. From $\mathcal{R}$ is finite and item (2) of lemma (7.1), we have $\forall i, i' \in \mathbb{A}^h_{\mathcal{R}} \ : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{\widehat{\overline{I}}_j \,|\, j \in \mathbb{B}^{h,i}_{\mathcal{R}}\} = \{\widehat{\overline{I}}_j \,|\, j \in \mathbb{B}^{h,i'}_{\mathcal{R}}\}$.

Proof of condition (3):

Let $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A}(\widehat{\overline{I}}_i \stackrel{imr_i}{\longrightarrow} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$. From $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in A}(\widehat{\overline{I}}_i \stackrel{imr_i}{\longrightarrow} \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and defn. (6.3), we have $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A}(\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$. From $type_{\mathcal{R}}(\overline{h}(u)) = \bigwedge_{i \in A}(\psi_u(imr_i), \widehat{\overline{I}}_i \longrightarrow \bigvee_{j \in B_i} \widehat{\overline{I}}_j)$ and the proof of condition (2) above, we have $\forall i, i' \in A \ : \ (imr_i, \widehat{\overline{I}}_i) = (imr_{i'}, \widehat{\overline{I}}_{i'}) \Rightarrow \{\, \widehat{\overline{I}}_j \mid j \in B_i \,\} = \{\, \widehat{\overline{I}}_j \mid j \in B_{i'} \,\}$. $\square$

# 8. CONCLUSION

We have established the first equivalence between a type system and a model checker. Our proof from type checking to model checking is essentially type preservation with respect to the model checker's semantics while the proof from model checking to type checking constructs types and shows that they are well formed and that the program has a valid type derivation. Our technique has several elements that are general in nature and should be useful in proving other equivalence results between type checking and model checking as well as in devising synergistic program analyses that embody an interplay between a type system and a model checker. We also intend to explore a general type-based framework, in the spirit of refinement types, for resource-usage analysis.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

[1] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proceedings of POPL'02, SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 33–44, 2002.

[2] Roberto M. Amadio and Luca Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993. Also in Proceedings of POPL'91.

[3] Torben Amtoft and Franklyn Turbak. Faithful translations between polyvariant flows and polymorphic types. In *Proceedings of ESOP'00, European Symposium on Programming*, pages 26–40. Springer-Verlag (*LNCS* 1782), 2000.

[4] Andrew Appel and Lal George. Optimal spilling for CISC machines with few registers. In Cindy Norris and Jr. James B. Fenwick, editors, *Proceedings of PLDI'01, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 243–253, 2001.

[5] Thomas Ball and Sriram Rajamani. The SLAM project: Debugging system software via static analysis. In *Proceedings of POPL'02, SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 1–3, 2002.

[6] Anindya Banerjee. A modular, polyvariant and type-based closure analysis. In *Proceedings of ICFP'97, ACM International Conference on Functional Programming*, pages 1–10, 1997.

[7] Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Ugo De'Liguoro. Intersection and union types: Syntax and semantics. *Information and Computation*, 119(2):202–230, 1995.

[8] Dennis Brylow and Jens Palsberg. Deadline analysis of interrupt-driven software. In *Proceedings of FSE'03, ACM SIGSOFT International Symposium on the Foundations of Software Engineering joint with ESEC'03, European Software Engineering Conference*, Helsinki, Finland, September 2003. To appear.

[9] Sagar Chaki, Sriram K. Rajamani, and Jakob Rehof. Types as models: Model checking message-passing programs. In *Proceedings of POPL'02, SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 45–57, 2002.

[10] Krishnendu Chatterjee, Di Ma, Rupak Majumdar, Tian Zhao, Thomas A. Henzinger, and Jens Palsberg. Stack size analysis of interrupt driven software. In *Proceedings of SAS'03, International Static Analysis Symposium*, pages 109–126. Springer-Verlag (*LNCS* 2694), San Diego, June 2003.

[11] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 2000.

[12] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Principal type schemes and lambda-calculus semantics. In J. Seldin and J. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 535–560. Academic Press, 1980.

[13] Mario Coppo and Paola Giannini. A complete type inference algorithm for simple intersection types. In *Proceedings of CAAP'92*, pages 102–123. Springer-Verlag (*LNCS* 581), 1992.

[14] Patrick Cousot. Types as abstract interpretations. In *Proceedings of POPL'97, 24th Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 316–331, 1997.

[15] Patrick Cousot and Radhia Cousot. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form. In Pierre Wolper, editor, *Proceedings of CAV'95, Seventh International Conference on Computer Aided Verification*, pages 293–308. Springer-Verlag (*LNCS* 939), 1995.

[16] Karl Crary and Stephanie Weirich. Resource bound certification. In *Proceedings of POPL'00, 27nd Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 184–198, 2000.

[17] Manuvir Das, Sorin Lerner, and Mark Seigle. ESP: Path-sensitive program verification in polynomial time. In *Proceedings of PLDI'02, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 57–68, 2002.

[18] Rowan Davies and Frank Pfenning. Intersection types and computational effects. In *Proceedings of ICFP'00, ACM SIGPLAN International Conference on Functional Programming*, pages 198–208, 2000.

[19] Robert DeLine and Manuel Fahndrich. Enforcing high-level protocols in low-level software. In *Proceedings of PLDI'01, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 59–69, 2001.

[20] David Detlefs, K. Rustan Leino, Greg Nelson, and James Saxe. Extended static checking. Technical Report 159, Compaq Systems Research Center, 1998.

[21] Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. Extended static checking for Java. In *Proceedings of PLDI'02, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 234–245, 2002.

[22] Jeffrey S. Foster, Tachio Terauchi, and Alex Aiken. Flow-sensitive type qualifiers. In *Proceedings of PLDI'02, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1–12, 2002.

[23] Tim Freeman and Frank Pfenning. Refinement types for ML. In *Proceedings of PLDI'91, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 268–277, 1991.

[24] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. In *Proceedings of LICS'87, IEEE Symposium on Logic in Computer Science*, pages 194–204, 1987.

[25] Nevin Heintze. Control-flow analysis and type systems. In *Proceedings of SAS'95, International Static Analysis Symposium*, pages 189–206. Springer-Verlag (*LNCS* 983), Glasgow, Scotland, September 1995.

[26] J. Roger Hindley. Types with intersection: An introduction. *Formal Aspects of Computing*, 4:470–486, 1991.

[27] Atsushi Igarashi and Naoki Kobayashi. Resource usage analysis. In *Proceedings of POPL'02, SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 331–342, 2002.

[28] Trevor Jim. What are principal typings and what are they good for? In *Proceedings of POPL'96, 23nd Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 42–53, 1996.

[29] Naoki Kobayashi. Type systems for concurrent processes: From deadlock-freedom to livelock-freedom, time-boundedness. In *IFIP TCS*, pages 365–389, 2000.

[30] Yitzhak Mandelbaum, David Walker, and Robert Harper. An effective theory of type refinements. In *Proceedings of ICFP'03, ACM International Conference on Functional Programming*, 2003.

[31] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

[32] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[33] Christian Mossin. Exact flow analysis. In *Proceedings of SAS'97, International Static Analysis Symposium*, pages 250–264. Springer-Verlag (*LNCS*), 1997.

[34] Mayur Naik and Jens Palsberg. Compiling with code-size constraints. *ACM Transactions on Embedded Computing Systems*. To appear. Preliminary version in Proceedings of LCTES'02, Languages, Compilers, and Tools for Embedded Systems joint with SCOPES'02, Software and Compilers for Embedded Systems, pages 120–129, Berlin, Germany, June 2002.

[35] Flemming Nielson. The typed lambda-calculus with first-class processes. In *Proceedings of PARLE'89*, pages 357–373, 1989.

[36] Jens Palsberg. Equality-based flow analysis versus recursive types. *ACM Transactions on Programming Languages and Systems*, 20(6):1251–1264, 1998.

[37] Jens Palsberg and Di Ma. A typed interrupt calculus. In *FTRTFT'02, 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems*, pages 291–310. Springer-Verlag (*LNCS* 2469), Oldenburg, Germany, September 2002.

[38] Jens Palsberg and Patrick M. O'Keefe. A type system equivalent to flow analysis. *ACM Transactions on Programming Languages and Systems*, 17(4):576–599, July 1995. Preliminary version in Proceedings of POPL'95, 22nd Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages, pages 367–378, San Francisco, California, January 1995.

[39] Jens Palsberg and Christina Pavlopoulou. From polyvariant flow information to intersection and union types. *Journal of Functional Programming*, 11(3):263–317, May 2001. Preliminary version in Proceedings of POPL'98, 25th Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages, pages 197–208, San Diego, California, January 1998.

[40] Jens Palsberg and Scott Smith. Constrained types and their expressiveness. *ACM Transactions on Programming Languages and Systems*, 18(5):519–527, September 1996.

[41] Benjamin Pierce. Programming with intersection types, union types, and polymorphism. Technical Report CMU–CS–91–106, Carnegie Mellon University, 1991.

[42] Andreas Podelski. Model checking as constraint solving. In *Proceedings of SAS'00, International Static Analysis Symposium*, pages 22–37. Springer-Verlag (*LNCS* 1824), 2000.

[43] H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. Hu, C-H. Hsu, and U. Kremer. Energy-conscious compilation based on voltage scaling. In *Proceedings of the 2002 Joint Conference on Languages, Compilers and Tools for Embedded Systems & Software and Compilers for Embedded Systems (LCTES/SCOPES '02)*, pages 2–11, June 2002.

[44] David Schmidt and Bernhard Steffen. Program analysis as model checking of abstract interpretations. In *Proceedings of SAS'98, Static Analysis Symposium*, pages 351–380. Springer-Verlag (*LNCS* 1503), 1998.

[45] Bernhard Steffen. Data flow analysis as model checking. In T. Ito and A.R. Meyer, editors, *Proceedings of TACS'91, Theoretical Aspects of Computer Science*, pages 346–364. Springer-Verlag (*LNCS* 526), 1991.

[46] Stefan van Bakel. *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*. PhD thesis, Catholic University of Nijmegen, 1991.

[47] J. B. Wells, Allyn Dimock, Robert Muller, and Franklyn Turbak. A typed intermediate language for flow-directed compilation. In *Proceedings of TAPSOFT'97, Theory and Practice of Software Development*, pages 757–771. Springer-Verlag (*LNCS* 1214), 1997.

[48] Kent Wilken, Jack Liu, and Mark Heffernan. Optimal instruction scheduling using integer programming. pages 121–133, 2000.

[49] Andrew Wright and Matthias Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94, 1994.

[50] Hongwei Xi. Imperative programming with dependent types. In *Proceedings of LICS'00, Fifteenth IEEE Symposium on Logic in Computer Science*, pages 375–387. IEEE Computer Society Press, 2000.

[51] Hongwei Xi and Frank Pfenning. Eliminating array bound checking through dependent types. In *Proceedings of PLDI'98, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 249–257, 1998.

[52] Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *Proceedings of POPL'99, 26th Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 214–227, 1999.