

CERIAS Tech Report 2004-81

**BOUNDING THE STACK SIZE OF INTERRUPT-DRIVEN
PROGRAMS**

by Di Ma

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

BOUNDING THE STACK SIZE OF INTERRUPT-DRIVEN PROGRAMS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Di Ma

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2004

To my grandparents

for bringing me up, for their love and support

ACKNOWLEDGMENTS

This thesis would not have existed without the help of many people. Among them, I own a great debt to my adviser, Jens Palsberg. It is his enthusiasm about the field of programming languages and about academic research in general that influenced me to embark on my journey to a Ph.D.. His deep insights have always helped me overcome my difficulties. His interest and facility with mathematical proofs prompted me to develop a similar taste. He is intimately familiar with the literature in the field of programming languages, from which I have constantly benefited in the course of my research. I would also like to thank him for his patience in guiding me into the realm of scientific research, especially at the beginning, when I felt so ignorant and thought that perhaps no one else would be patient enough to bother.

I thank Tian Zhao for his collaboration on the work on type inference for interrupt calculus. Section 2.3 of this thesis is largely based on our paper manuscript [64], which grew out of our numerous discussions in the summer of 2002. I thank Mayur Naik for illuminating me the interesting topic of model checking and type systems, which ultimately became the third chapter of this thesis. I was also influenced by his elegant style in writing mathematical proofs. I thank Dennis Brylow for providing me with interesting interrupt-driven example programs, without which there would be no interrupt calculi.

I am also grateful to Scott J. Baxter of the English department for carefully reading all the English text in this thesis and giving many suggestions for improvement.

Thanks to Professor Mitchell Wand of Northeastern University for commenting on my thesis draft and suggesting many improvements. Thanks to Thomas VanDrunen, Bogdan Carbunar, Deepak Rao Bobbarjung, and Lukasz Ziarek of the S^3 lab who each read part of an early draft of this thesis and gave helpful comments.

Thanks to Professor Jan Vitek for graciously providing me with financial support in the fall semester of 2003.

Thanks to my thesis committee: Jens Palsberg, Jan Vitek, Antony Hosking, Suresh Jagannathan and Mitchell Wand for giving me many advices.

Thanks to all the lab members of the S^3 lab for their kindness over the last five years.

I would also like to thank my grandparents and my parents for their patience and support over the years. My grandfather's health had been my greatest concern in the last two years. Although he did not make it to see my graduation, his love and support will be a strong and firm pillar in my heart for the rest of my life.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	viii
ABSTRACT	ix
1 Introduction	1
1.1 Interrupt-driven systems	1
1.2 Formal software verification	3
1.2.1 Contributions of the thesis	4
1.3 The interrupt computation model	5
1.4 The stack size problem and goals of the thesis	7
1.5 Interrupt calculi	9
1.6 Typed interrupt calculus	10
1.7 Outline of the remainder of the thesis	16
2 Typed interrupt calculus	18
2.1 Interrupt calculus	18
2.1.1 Syntax	18
2.1.2 Semantics	19
2.1.3 Nontermination	21
2.1.4 Monotonicity	22
2.2 A type system for interrupt calculus	23
2.2.1 Types	23
2.2.2 Type rules	24
2.2.3 Stack-size boundedness	27
2.3 Type inference	36
2.3.1 Summary of the results on complexity	36

	Page
2.3.2	Constraints 38
2.3.3	From types to constraints 39
2.3.4	Solving constraints 44
2.3.5	Stack-size checking is in PSPACE 47
2.3.6	Type inference is in PSPACE 48
2.3.7	Maximum stack size problem is PSPACE-hard 49
2.3.8	Stack-size checking is PSPACE-hard 60
2.4	Related work 62
2.4.1	Stack checking for interrupt-driven programs 62
2.4.2	Sized types and dependent types 63
2.4.3	Event-driven FRP 64
2.4.4	The Giotto language and the embedded machine 65
2.4.5	Esterel 66
2.5	Future directions 67
3	Periodic interrupt calculus 68
3.1	Background 68
3.2	Periodic interrupt calculus 70
3.2.1	Syntax 70
3.2.2	Semantics 71
3.2.3	Nontermination 74
3.2.4	Latency-space safety analysis 76
3.3	Abstract semantics 76
3.4	Type system 83
3.4.1	Types 83
3.4.2	Type rules 86
3.4.3	Type soundness 90
3.5	An example 107
3.6	Type construction 114

	Page
3.6.1 ψ function	115
3.6.2 Soundness, stack-irrelevancy and completeness of ψ	121
3.6.3 Constructing types and type judgments	153
3.6.4 Well-formedness of types and type judgments	155
3.6.5 Constructing type derivations	161
3.7 Equivalence relation	181
3.7.1 Model checking vs. type checking	182
3.8 Related work	183
3.8.1 Model checking	184
3.8.2 Type systems	186
4 Conclusion	188
LIST OF REFERENCES	190
VITA	195

LIST OF FIGURES

Figure		Page
1.1	An example of indefinitely interrupting	7
1.2	A program for copying data from one device to another device.	12
1.3	Two selfish handlers	13
1.4	Two prioritized handlers	14
1.5	Two cooperative handlers	15
1.6	Two fancy handlers	16
1.7	A timer	17
2.1	Syntax of interrupt calculus	19
2.2	Semantics of interrupt calculus	20
2.3	Summary of results	37
2.4	Enable relation of interrupt handlers	55
3.1	Syntax of periodic interrupt calculus	70
3.2	Semantics of the periodic interrupt calculus	73
3.3	Abstract semantics of the periodic interrupt calculus	79
3.4	The example program	108
3.5	Excerpt of the reachable states of the example program	109
3.6	Stack size growth over time	110

ABSTRACT

Ma, Di. Ph.D., Purdue University, August, 2004. Bounding the Stack Size of Interrupt-driven Programs. Major Professor: Jens Palsberg.

A widely-used class of real-time, reactive, embedded systems is called *interrupt-driven systems* [8]. Programming of interrupt-driven systems is notoriously difficult and error-prone. This is because such systems are usually equipped with a small amount of memory while being asked to handle as many external interrupts as possible. Furthermore, such systems demand responsive handling of interrupts. Due to the fact that an interrupt may happen at any time, a handler can be interrupted by another interrupt, making the stack grow in order to store the context information for the current handler. The problem with such a scenario is that it may lead to stack overflow. Traditionally, this problem has been avoided by forbidding other interrupts during the execution of the handler. However, doing this puts tremendous limit on the number of interrupts which can be handled. Moreover, it greatly increases the response time for interrupts, resulting in an inefficient system and causing a potential predictability problem: the handling of an interrupt can be so long that the next interrupt occurrence is missed.

In this thesis, we lay a formal framework, which, to the best of our knowledge, is the first in the field, to ensure stack boundedness, to give the tightest possible upper bound of the stack usage for interrupt-driven programs, and to guarantee predictability. Specifically, we develop two formal languages, *interrupt calculus* and *periodic interrupt calculus*, to capture the characteristics of interrupt-driven systems. We advocate intersection types and union types from the field of programming languages as a convenient vehicle to solve these problems. We base our analysis on two type systems which are designed for the two calculi. Our results show that the calculi

demonstrate the desired capability for characterizing interrupt-driven programs. We show that once an interrupt calculus program type checks, there can be no stack overflow; we prove that the type inference problem for interrupt calculus is in PSPACE. For type-checked periodic interrupt calculus programs, we show that not only can the stack not overflow, but that it is also guaranteed that no single interrupt can be missed. In addition, our building of the types and type derivations of the periodic interrupt calculus programs unveils an equivalence relation between model checking and type systems, which may be of interest in its own right.

1 INTRODUCTION

1.1 Interrupt-driven systems

Real-time, reactive and embedded systems are becoming increasingly widely used nowadays. At the same time, there is also an increasing demand that those systems not only meet real-time requirements but also be free of run-time errors, such as stack overflow. For example, Klaus Brunnstein reported that a software glitch in the real-time system that controlled the railway switch at Hamburg-Altona station caused the entire Altona switch tower to shutdown on the first day of its on-site testing (“Stack overflow shuts down new Altona switch tower on first day” [41].) It was later found out that the glitch caused a stack overflow under certain conditions. It is fortunate that the software bug showed up in the testing in Altona case. These kinds of program errors are usually difficult to find out under most conditions, and, therefore, are often difficult to reproduce.

In this thesis, we will focus the scope of our study on a common class of real-time, reactive embedded systems which are called *interrupt-driven* systems as described by Brylow, Damgaard and Palsberg [8]. In particular, we study the stack boundedness problem and the predictable interrupt latency problem of such a class of systems.

Programming in interrupt-driven systems is notoriously difficult and error-prone because (1) such systems are usually equipped with only a very small amount of memory while being asked to handle as many external interrupts as possible; (2) interrupts may happen at any time, which makes it difficult for programs to control and manage the use of the system hardware. Furthermore, such systems demand fast and responsive handling of interrupts in order to promote high efficiency. Due to the fact that an interrupt may be triggered at any time, an interrupt handler can be written in such a way so that it is allowed to be interrupted by another interrupt,

leading to a larger stack size because the context information of the current handler has to be stored on the stack. The problem with this is that, in such a scenario, there is no guarantee that the stack will not overflow. Traditionally, the problem has been avoided by writing a handler in such a way that it forbids other interrupts during the execution of the handler. However, doing this tremendously limits the number of interrupts which can be handled. Moreover, it greatly increases the response time for interrupts, which not only results in a very inefficient system, but also introduces a potential predictability problem: the handling of an interrupt is too long to allow the next interrupt occurrence to be handled.

We thus identify two important issues for designing and developing software for such systems:

1. *Constrained resources* Most embedded software runs on resource-constrained processors which are not equipped with computation hardware matching those on today's personal computers because of economic concerns, low power consumption, size, or other constraints.

For example, consider the system that uses the Zilog Z80 micro-controller described in [8]. The memory space of the micro-controller consists of only 256 8-bit on-chip registers. Despite this limitation, the programmers need to write programs to fit in the controller's available space in order to allow the device to perform network communications, fan control, temperature control and a digit panel control interface, as well as other operations.

Once the processor, RAM, and other specifications have been chosen for an embedded system, the programmers will face a particularly vexing problem: the software has to be written so that program never runs out of stack space during execution.

2. *Predictability* Real-time, reactive embedded systems require that most external events be handled in a timely and predictable fashion. That is, real-time embedded systems should guarantee a pre-specified, finite response time to each

external stimulus. For example, the auto-piloting control software running on jet airplanes should handle the changes in external air turbulence responsively (a pre-defined time interval) in order to guarantee the safe and comfortable operation of the plane. It is well known that writing predictable software that meets real-time constraints is notoriously difficult

This thesis concentrates on these two important issues. Specifically, we lay a formal framework, which, to the best of our knowledge, is the first in the field to ensure the stack boundedness, to give the tightest upper bound of the stack usage for interrupt-driven programs, and to guarantee the predictability. We develop two formal languages, *interrupt calculus* and *periodic interrupt calculus*, to capture the characteristics of the interrupt-driven systems. We advocate types and type systems from the field of programming languages as a convenient vehicle for solving these problems.

1.2 Formal software verification

The issues described in the previous section are well-known to the software community. These problems have traditionally been dealt with, in reality, by exhaustive software testing or worst-case execution time [15,19,42,50]. More recently, formal software verification has been receiving increasingly greater attention. Many techniques have been explored to validate certain properties regarding the software implementations. Among them, formal language verification techniques such as type based program analysis [36,37,43] and software model checking [3,4,8,9,11,17,18,20,53] seem to be gaining momentum recently as powerful program verification techniques.

There is a large body of work which tries to apply formal language verification techniques to the field of real-time, embedded systems to deal with the constrained-resource problem and predictability. Hughes, Pareto and Sabry [30,47,48] use *sized types* to reason the boundedness of data structure in the context of employing functional languages (ML) in real-time, embedded systems. A number of languages have

been designed to facilitate the correct real-time system designs and implementations: Wan, Taha and Hudak describe Event-driven FRP (Functional Reactive Programming) [28]; Henzinger, Horowitz and Kirsch devise the Giotto Language and Embedded machine [22–25]. Both languages are used to ensure the predictable behaviors of real-time embedded systems. Basu, Kumar, Polorny and Ramakrishan [6] develop a resource-constrained model checking technique which is capable of predicting whether program execution requires unbounded stack size.

Although a large body of research on model checking has been dedicated to the problem, there is little work that looks into the field of interrupt-driven software, except for the following two, which study the problem in the context of interrupt-driven software. Brylow, Damgaard and Palsberg [8] abstract the interrupt program into a control flow graph; they then model check the size of bounded stack of interrupt driven programs by running a context free reachability algorithm on the graph. Brylow and Palsberg [9] use almost the same model checking strategy to analyze the problem of whether each interrupt’s deadline can be met. Their method involves identifying different loops in the program, as well as worst case execution time analysis.

Our approach to the problem is to employ types and type systems of programming languages in order to solve the problems. Moreover, we try to bridge the gap between resource constraint, predictability problem of interrupt-driven, real-time embedded systems and the application of formal language analysis in this field. Our work can be viewed as a synergistic combination between formal programming language, real-time embedded systems and resource-aware compilation [39].

We will give a more detailed summary of related work and compare their results with ours at the end of Chapter 2 and Chapter 3.

1.2.1 Contributions of the thesis

This thesis offers the following four technical contributions [12, 44, 64].

- We introduce two formal languages, namely, *interrupt calculus* and *periodic interrupt calculus*, as the basic tools to formally analyze the stack size for interrupt-driven programs.
- We design two type systems for the interrupt calculi to ensure stack boundedness, to give the tightest upper bound of the stack usage, which shows our type system is *sound* with respect to stack size. Our method of incorporating the stack size values into type systems is novel.
- We build timing information into the type system for the *periodic interrupt calculus*, which allows the stack size to be analyzed under the constraints of interrupt latencies.
- Our study of the *periodic interrupt calculus* reveals an interesting equivalence relation between model checking and type systems with respect to bounding the stack size of interrupt-driven programs. The relation itself may be of interest in its own right.

1.3 The interrupt computation model

Interrupt-driven embedded systems generally have a fixed number of interrupt sources (interrupt devices) with a software handler defined for each source. When an interrupt occurs, control is transferred automatically to the handler for that interrupt source, unless interrupt processing is disabled. If disabled, the processing of the interrupt will wait for interrupt processing to be enabled.

Interrupt Mask Register While some modern, general-purpose CPUs have sophisticated ways of handling internal and external interrupts, the notion of an interrupt mask register (imr) is widely used. This is especially true for the processors that are used in embedded systems with small memory size, a need for low power consumption, as well as other constraints. We list some characteristics of four processors that are often used in embedded systems as follows:

product	Processor	# of interrupt sources	master bit
Microcontroller	Zilog Z86	6	yes
iPAQ Pocket PC	Intel strongARM, XScale	21	no
Palm	Motorola Dragonball (68000 Family)	22	yes
Microcontroller	Intel MCS-51 Family (8051 etc)	6	yes

Each of these processors have similar-looking imr's. For example, consider the imr for the MCS-51 (The imr is called interrupt enable (IE) register):

EA	-	ET2	ES	ET1	EX1	ET0	EX0
----	---	-----	----	-----	-----	-----	-----

These bits have the following meanings respectively:

- EA: enable/disable all interrupt handling,
- -: reserved (not used), and
- each of the remaining six bits corresponds to a specific interrupt source.

We will refer to the EA bit (and similar bits on other processors) as the *master bit*. The idea is that for a particular interrupt handler to be enabled, *both* the master bit and the bit for that interrupt handler have to be enabled. This particular semantics is supported by the Z86, Dragonball, MCS-51, and many other processors.

Interrupt Handling The processors used in embedded interrupt-driven systems usually save the current processor status, including the values in general registers, the current program counter, interrupt mask register, when starting to handle an interrupt. We model this kind of handling in the following manner: when an interrupt handler is called, a return address is saved on the stack, the processor automatically turns *off* the master bit, and then starts to execute the code of the handler. At the time of return, the processor turns the master bit back *on*, restores the program counter to the previously saved return address and starts to execute code from the return address.

1.4 The stack size problem and goals of the thesis

In order to obtain responsive handling, it is necessary to keep interrupt processing enabled most of the time, including in *the body* of lower priority interrupt handlers. However, this allows the possibility that interrupt handlers could be themselves interrupted, thus making it difficult to understand whether real-time constraints can be met. Conversely, to write reliable code with a given real-time property, it is often simplest to disable interrupts in the body of interrupt handlers. This may delay the handling of other interrupts, therefore making it difficult for the system to have other desired real-time properties. The resultant tension between fast response times, easy-to-understand and reliable code encourages developers to write code which is often difficult to test and debug.

A particularly notorious programming error in the interrupt-driven software occurs when the interrupt handlers are allowed to interrupt each other indefinitely. Such a situation leads to both an unbounded stack and a potential unbounded handling time which leads to two serious violations: a violation of the resource constraints and a violation of the predictability properties of the system.

```

handler 1 {                                handler 2 {
    // do something                          // do something
    enable-handling-of-interrupt-2          enable-handling-of-interrupt-1
    // do something else                    // do something else
    iret                                    iret
}                                           }

```

Figure 1.1. An example of indefinitely interrupting

Consider the two interrupt handlers in Figure 1.1. Suppose an interrupt from source 1 arrives first, so handler 1 is called. Before returning, handler 1 enables handling of interrupts from source 2, and, unfortunately, an interrupt from source

2 arrives before handler 1 has returned. Thus, handler 2 is called, and it, in turn, enables handling of interrupts from source 1 before returning, thus, allowing a highly undesirable cyclic handling which results in an unbounded stack and an indefinite handling time.

Clearly, the error is not about misusing data; rather, it is about the need for unbounded resources. Previously, static checking for such errors could only be done using model checking. However, what is needed for model checking is the whole-program analysis which cannot check program fragments [8, 54]; if the program is altered, model checking has to be run again on the entire program.

Another closely related problem is predicting the tightest upper bound of the stack size caused by the interrupts. This problem is of particular interest because in the process of designing the real-world embedded systems, it is often the case that the amount of memory is small and fixed, due to the economic reasons, and designers are constantly asked the question: does the stack fit into the amount of available RAM? This need can be met by providing the tightest upper bound to the designers. In this context, modular program checking of the stack size property would be also beneficial because it saves both design and development time.

A traditional type system does not (1) check for stack unboundedness error; or (2) give the tightest upper bound of the stack size usage. The goal of this thesis is to present type systems that guarantee stack boundedness by enabling modular type checking and to give the tightest upper bound for the stack size.

Each type contains information about the stack size and also serves as documentation for the program. When an interrupt handler is altered, it is sufficient to re-type check only that particular interrupt handler. Thus, we see that type checking is more modular than model checking in this setting.

We study this problem in two different yet closely related settings: *interrupt calculus* and *periodic interrupt calculus*.

1.5 Interrupt calculi

In order to design type systems for the stack size problem, we need a minimal setting in which to study interrupt-driven systems. For many programming paradigms, there is a small calculus which allows the study of properties in a language-independent way and which makes it tractable to prove key properties. For example, for functional programming there is the λ -calculus [5], for concurrent programming there is Milner’s calculus of communicating systems [34], for object-oriented programming there is the Abadi-Cardelli object calculus [1], and for mobile computation there is the π -calculus [35] and the ambient calculus [10]. However, these calculi do not offer any notion of interrupts and interrupt handling. While such concepts might be introduced on top of one of those calculi, we believe that it is better to design a new calculus with interrupts at the core. This new calculus should focus on the essential concepts and ignore everything else.

In this thesis, we introduce two versions of interrupt calculus: *interrupt calculus* and *periodic interrupt calculus* which contain essential constructs for programming interrupt-driven systems. A program in the calculi consists of two parts: a main part and a number of interrupt handlers. The program execution has access to:

- an interrupt mask register that can be updated during computation,
- a stack for storing return addresses, and
- a memory of integer variables; output is done via memory-mapped I/O.

The calculi are intended for modeling embedded systems that should run “forever,” and for which termination would be considered a disastrous error. To model such a situation, the calculi are designed such that no program can terminate; nontermination is guaranteed.

Each element on the stack is a return address. In order to measure the size of the stack, we simply count the number of elements on the stack.

Interrupt calculus Interrupt calculus is a viable and convenient vehicle for us to study the stack size problem for interrupt-driven programs. It captures the interrupt mechanism at its core by explicitly designating an *imr* register of N bits. Specifically, we represent the imr as a bit sequence $b_0b_1 \dots b_n$, where $b_i \in \{0, 1\}$, b_0 is the master bit, and, for $i > 0$, b_i is the bit for interrupts from source i which is handled by handler i . It is important to notice that the master bit is the most significant bit, and that the bit for handler 1 is the second-most significant bit, and so on. This layout is different from some processors, and it simplifies the notation used later. We assume that if the master bit and the bit that corresponds to interrupt source i are both turned on, then there is a call of the interrupt handler i .

Periodic interrupt calculus Periodic interrupt calculus is an extension of the interrupt calculus in the sense that it incorporates a temporal (periods) dimension into the interrupt mechanism: each interrupt source periodically generates an interrupt. We focus on the predictability property that *no interrupt can be missed*, which means that an interrupt should start to be handled before the next interrupt comes, and its impact on the stack size. Therefore, in the context of periodic interrupt calculus, the stack size is further bound by the latency constraints. Specifically, all interrupt sources have their own periods and the periods are part of the program specification. The system semantics also takes a timer (latency) for each interrupt source that: (1) ticks down whenever an instruction is executed, and (2) is incremented whenever the handler is called. We assume that if (1) the master bit and the bit that corresponds to interrupt source i are both turned on, and (2) the timer holds a negative value, then there is a call of the interrupt handler i .

1.6 Typed interrupt calculus

Our approach is to use types to analyze the stack size problem. To this end, we present type systems that guarantee stack boundedness and enable modular type checking.

In this section, we will focus our attention on illustrating the type system for interrupt calculus. We will introduce the type system for periodic interrupt calculus in Chapter 3.

A type for a handler contains information about the imr on entry and the imr at the point of return. Given that a handler can be called at different points in the program where the imr may have different values, the type of a handler is an intersection type [13, 26] of the form:

$$\bigwedge_{j=1}^n ((\widehat{imr})^j \xrightarrow{\delta^j} (\widehat{imr}')^j).$$

where the j^{th} component of the intersection means:

if the handler is called in a situation where the imr can be conservatively approximated by $(\widehat{imr})^j$, then at the point of return, the imr can conservatively be approximated by $(\widehat{imr}')^j$, and during that call, the stack will grow by at most δ^j elements, excluding the return address for the call itself.

The annotations δ^j help in checking that the stack is bounded. Our type system with annotated types is an example of a type-based analysis [43].

We will illustrate our use of types with six examples of increasing sophistication. Among these example programs, the first five type check, while the sixth program illustrates the limitations of our type system. We will use the concrete syntax that is supported by our type checker; later, in Chapter 2, we will give an abstract syntax that is similar to the concrete syntax used here.

Note that an imr value, say, 11, will be written as **11b** in the concrete syntax, to remind the reader that it is a binary value. In addition, the following type of a handler

$$\bigwedge_{j=1}^n ((\widehat{imr})^j \xrightarrow{\delta^j} (\widehat{imr}')^j).$$

will be written $((\widehat{imr})^1 \rightarrow (\widehat{imr}')^1 : \delta^1) \dots ((\widehat{imr})^n \rightarrow (\widehat{imr}')^n : \delta^n)$.

```

Maximum stack size: 1

imr = imr or 11b
loop {
    if ( gotchar == 0 ) {
        outdata = achar
        gotchar = 1
    } else {
        skip
    }
}

handler 1 [ ( 11b -> 11b : 0 ) ] {
    achar = indata
    gotchar = 0
    iret
}

```

Figure 1.2. A program for copying data from one device to another device.

The program in Figure 1.2 is an interrupt calculus version of Example 3-5 from Wolf's textbook [60, p.113]. The program uses memory-mapped I/O; two variables map to the device registers; and copies data from one device to another device.

- **indata**: the input device writes data in this register and
- **outdata**: the output device reads data from this register.

The line `maximum stack size: 1` is a part of the program text. It tells the type checker to check that the stack can never be of a size greater than one. The number 1 is a count of return addresses on the stack; nothing other than return addresses can be put on the stack in our calculus. The header of the handler contains the annotation `11b -> 11b : 0`. This header is a type which says that if the handler is called in a situation where the `imr` can be conservatively approximated by 11, then it will return in a situation where the `imr` can be conservatively approximated by 11, and the stack will not grow during the call. The value 11 in this example should be read as follows:

```

maximum stack size: 1
                                handler 1 [ ( 111b -> 111b : 0 ) ] {
                                skip
                                iret
imr = imr or 111b
                                }
loop {
                                handler 2 [ ( 111b -> 111b : 0 ) ] {
                                skip
                                iret
                                }
}
                                }

```

Figure 1.3. Two selfish handlers

the leftmost bit is the master bit, and the next bit is the bit for handler 1. The value 11 means that handler 1 is enabled.

The program in Figure 1.3 has two handlers which do not enable the master bit of `imr` in their code. with the difference being that there are now two handlers. The handlers cannot be interrupted so the maximum stack size is 1. Notice that since there are two handlers, the `imr` has three bits. Those bits are organized as follows. The leftmost bit is, as always, the master bit. The next bit is the bit for handler 1, and the rightmost bit is the bit for handler 2.

The program in Figure 1.4 illustrates how to program a notion of prioritized handlers where handler 1 has a higher priority than handler 2. While handler 1 cannot be interrupted by handler 2, it is possible for handler 2 to be interrupted by handler 1. Handler 2 achieves such a situation by disabling its own bit in the `imr` with the statement `imr = imr and 110b`, and then enabling the master bit with the statement `imr = imr or 100b`. Thus, handler 2 can be interrupted before it returns. Accordingly, the maximum stack size is 2. The type for handler 1 is an intersection type which reflects the fact that handler 1 can be called both from the main part of the program and from handler 2. If it is called from the main part, then the `imr` is 111, and if it is called from handler 2, then the `imr` is 110. The type for handler 2


```

                                handler 1 [ ( 111b -> 111b : 0 )
                                                ( 110b -> 110b : 0 ) ] {
maximum stack size: 2                skip
                                        iret
imr = imr or 111b                    }
loop {                                handler 2 [ ( 111b -> 111b : 1 ) ] {
    skip                               skip
    imr = imr or 111b                 imr = imr and 110b
}                                       imr = imr or 100b
                                        iret
                                        }

```

Figure 1.4. Two prioritized handlers

has been given the annotation 1 because handler 2 can be interrupted by handler 1, which, in turn, cannot be interrupted.

The program in Figure 1.5 illustrates how both handlers can allow the other handler to interrupt. Each handler uses the discipline of disabling its own bit in the `imr` before setting the master bit to 1. Doing this ensures that the maximum stack size is two.

Finally, the program in Figure 1.6 illustrates that n handlers can lead to a bounded stack where the bound is greater than n . In this case, we have two handlers and a maximum stack size of three. A stack size of three is achieved by first calling handler 1, then calling handler 2, and finally calling handler 1 again.

While our type system can type check many common programming idioms, as illustrated above, there are useful programs that it cannot type check. For example, the program in Figure 1.7, written by Dennis Brylow, is a 60 second timer. The `OUT` variable will be 0 for 60 seconds after a request for interrupt 2. There are two interrupt handlers:

```

maximum stack size: 2

imr = imr or 111b
loop {
    imr = imr or 111b
}

handler 1 [ ( 111b -> 101b : 1 )
           ( 110b -> 100b : 0 ) ] {
    imr = imr and 101b
    imr = imr or 100b
    iret
}

handler 2 [ ( 111b -> 110b : 1 )
           ( 101b -> 100b : 0 ) ] {
    imr = imr and 110b
    imr = imr or 100b
    iret
}

```

Figure 1.5. Two cooperative handlers

- The first handler is for an external timer that is expected to request an interrupt once each second.
- The second handler is a trigger. When it arrives, the `OUT` variable will become 0 for 60 seconds. Then `OUT` will become 1, and will remain so until the next trigger event.

Our type system cannot handle this pattern where handler 2 disables itself and then enables handler 1, and where the main program disables handler 1 and enables handler 2. Thus, while the program in Figure 1.7 has a maximum stack size of 2, it does not type check in our type system.

```

                                handler 1 [ ( 111b -> 111b : 2 )
                                        ( 110b -> 100b : 0 ) ] {
                                        imr = imr and 101b
                                        imr = imr or 100b
maximum stack size: 3          iret
                                }
imr = imr or 111b              handler 2 [ ( 111b -> 100b : 1 )
loop {                          ( 101b -> 100b : 1 ) ] {
    imr = imr or 111b          imr = imr and 110b
                                imr = imr or 010b
                                imr = imr or 100b
                                imr = imr and 101b
                                iret
                                }
}

```

Figure 1.6. Two fancy handlers

1.7 Outline of the remainder of the thesis

In Chapter 2, we introduce our interrupt calculus, its syntax and semantics; and we prove that no program can terminate. We also present a type system which ensures the stack boundedness, and we present algorithms of how to automatically infer the types that we need.

In Chapter 3, we present the syntax and semantics of the periodic interrupt calculus. We give an abstract semantics of the concrete semantics, and, based on the abstract semantics, we introduce our type system. We show how to construct the types out of the abstract model. This construction illustrates an interesting equivalence relation between model checking and type checking.

In Chapter 4, we make concluding remarks about our work by highlighting potential applications of this work.

```

maximum stack size: 1

SEC = SEC + 60
imr = imr or 110b
loop {
    if( SEC == 0 ) {
        OUT = 1
        imr = imr and 101b
        imr = imr or 001b
    } else {
        OUT = 0
    }
}

handler 1 [ ( 111b -> 111b : 0 )
           ( 110b -> 110b : 0 ) ] {
    SEC = SEC + (-1)
    iret
}

handler 2 [ ( 111b -> 110b : 0 )
           ( 101b -> 110b : 0 ) ] {
    SEC = 60
    imr = imr and 110b
    imr = imr or 010b
    iret
}

```

Figure 1.7. A timer

2 TYPED INTERRUPT CALCULUS

In this chapter, we formally introduce the interrupt calculus, its syntax and semantics, and we show that no program can terminate. We also present a type system for the calculus and prove stack boundedness; that is, once the program type checks, it guarantees a bounded stack size. Furthermore, we provide algorithms on how to infer the types for the interrupt calculus programs and study the complexity of type inference. We conclude the chapter by summarizing related work and by providing suggestions for future research.

2.1 Interrupt calculus

2.1.1 Syntax

Figure 2.1.1 gives the syntax of the interrupt calculus. We use x to range over a set of program variables, we use imr to range over bit strings, and we use c to range over integer constants. The over bar notation \bar{h} denotes a sequence $h_1 \dots h_n$; we will use the notation $\bar{h}(i) = h_i$. We use a to range over m and h .

We identify programs which are equivalent under the smallest congruence generated by the following 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 or h into one of the following seven forms:

$$\text{loop } s \quad \text{iret} \quad x = e; a \quad \text{imr} = \text{imr} \wedge \text{imr}; a \quad \text{imr} = \text{imr} \vee \text{imr}; a$$

(program) $p ::= (m, \bar{h})$
 (main) $m ::= \text{loop } s \mid s ; m$
 (handler) $h ::= \text{iret} \mid s ; h$
 (statements) $s ::= x = e \mid \text{imr} = \text{imr} \wedge \text{imr} \mid \text{imr} = \text{imr} \vee \text{imr} \mid$
 $\text{if0 } (x) s_1 \text{ else } s_2 \mid s_1 ; s_2 \mid \text{skip}$
 (expression) $e ::= c \mid x \mid x + c \mid x_1 + x_2$

Figure 2.1. Syntax of interrupt calculus

$(\text{if0 } (x) s_1 \text{ else } s_2); a \quad \text{skip}; a.$

2.1.2 Semantics

We use R to denote a *store*, that is, a partial function which maps program variables to integers.

We use σ to denote a *stack* generated by the grammar: $\sigma ::= \text{nil} \mid a :: \sigma$. We define the size of a stack as follows: $|\text{nil}| = 0$ and $|a :: \sigma| = 1 + |\sigma|$.

If $\text{imr} = b_0 b_1 \dots b_n$, where $b_i \in \{0, 1\}$, then we will use the notation $\text{imr}(i) = b_i$. The predicate *enabled* is defined as follows:

$$\text{enabled}(\text{imr}, i) = (\text{imr}(0) = 1) \wedge (\text{imr}(i) = 1) \quad i \in 1..n.$$

We use 0 to denote the imr value where all bits are 0. We use \mathbf{t}_i to denote the imr value where all bits are 0's except that the i th bit is set to 1. We will use \wedge to denote bitwise logical conjunction, \vee to denote bitwise logical disjunction, \leq to denote bitwise logical implication, and $\neg(\cdot)$ to denote bitwise logical negation. Notice that $\text{enabled}(\mathbf{t}_0 \vee \mathbf{t}_i, j)$ is true for $i = j$ and false otherwise. The imr values, ordered by \leq , form a lattice with the bottom element 0.

A *program state* is a tuple $\langle \bar{h}, R, \text{imr}, \sigma, a \rangle$. We will refer to a as *the current statement*; it models the instruction pointer of a CPU. We use P to range over program

states. If $P = \langle \bar{h}, R, imr, \sigma, a \rangle$, then we use the notation $P.stk = \sigma$. For $p = (m, \bar{h})$, the initial program state for executing p is $P_p = \langle \bar{h}, \lambda x.0, 0, nil, m \rangle$, where the function $\lambda x.0$ is defined on the variables which are used in the program p .

A small-step operational semantics for the language is given by the reflexive, transitive closure of the relation \rightarrow on program states in Figure 2.1.2.

$$\langle \bar{h}, R, imr, \sigma, a \rangle \rightarrow \langle \bar{h}, R, imr \wedge \neg \mathbf{t}_0, a :: \sigma, \bar{h}(i) \rangle \quad (2.1)$$

if $enabled(imr, i)$

$$\langle \bar{h}, R, imr, \sigma, \mathbf{iret} \rangle \rightarrow \langle \bar{h}, R, imr \vee \mathbf{t}_0, \sigma', a \rangle \text{ if } \sigma = a :: \sigma' \quad (2.2)$$

$$\langle \bar{h}, R, imr, \sigma, \mathbf{loop } s \rangle \rightarrow \langle \bar{h}, R, imr, \sigma, s; \mathbf{loop } s \rangle \quad (2.3)$$

$$\langle \bar{h}, R, imr, \sigma, x = e; a \rangle \rightarrow \langle \bar{h}, R\{x \mapsto eval_R(e)\}, imr, \sigma, a \rangle \quad (2.4)$$

$$\langle \bar{h}, R, imr, \sigma, \mathbf{imr} = \mathbf{imr} \wedge imr'; a \rangle \rightarrow \langle \bar{h}, R, imr \wedge imr', \sigma, a \rangle \quad (2.5)$$

$$\langle \bar{h}, R, imr, \sigma, \mathbf{imr} = \mathbf{imr} \vee imr'; a \rangle \rightarrow \langle \bar{h}, R, imr \vee imr', \sigma, a \rangle \quad (2.6)$$

$$\langle \bar{h}, R, imr, \sigma, (\mathbf{if}0(x) s_1 \mathbf{else } s_2); a \rangle \rightarrow \langle \bar{h}, R, imr, \sigma, s_1; a \rangle \text{ if } R(x) = 0 \quad (2.7)$$

$$\langle \bar{h}, R, imr, \sigma, (\mathbf{if}0(x) s_1 \mathbf{else } s_2); a \rangle \rightarrow \langle \bar{h}, R, imr, \sigma, s_2; a \rangle \text{ if } R(x) \neq 0 \quad (2.8)$$

$$\langle \bar{h}, R, imr, \sigma, \mathbf{skip}; a \rangle \rightarrow \langle \bar{h}, R, imr, \sigma, a \rangle \quad (2.9)$$

Figure 2.2. Semantics of interrupt calculus

We define the function $eval_R(e)$ as follows:

$$eval_R(c) = c$$

$$eval_R(x) = R(x)$$

$$eval_R(x + c) = R(x) + c$$

$$eval_R(x_1 + x_2) = R(x_1) + R(x_2).$$

Rule (2.1) states that if an interrupt is enabled, then it may occur. The rule says that if $enabled(imr, i)$, then it is a possible transition to push the current statement on the stack, make $\bar{h}(i)$ the current statement, and turn off the master bit in the imr . Notice that we make no assumptions about the arrivals of interrupts; any enabled interrupt can occur at any time, and, conversely, no interrupt must occur.

Rule (2.2) models interrupt return. The rule says that to return from an interrupt, remove the top element of the stack, make the removed top element the current statement, and turn on the master bit.

Rule (2.3) is an unfolding rule for loops, and Rules (2.4)–(2.9) are standard rules for statements.

2.1.3 Nontermination

We say that a program p *can terminate* if $P_p \rightarrow^* P'$ and there is no P'' such that $P' \rightarrow P''$.

We say that a program state $\langle \bar{h}, R, imr, \sigma, a \rangle$ is *consistent* if and only if (1) $\sigma = \text{nil}$ and $a = m$; or (2) $\sigma = h^k :: \dots :: h^1 :: m :: \text{nil}$ and $a = h$, for $k \geq 0$, where $k = 0$ means $\sigma = m :: \text{nil}$.

Lemma 2.1.1 (Consistency Preservation) *If P is consistent and $P \rightarrow P'$, then P' is consistent.*

Proof A straightforward case analysis of $P \rightarrow P'$. ■

Lemma 2.1.2 (Progress) *If P is consistent, then there exists P' such that $P \rightarrow P'$.*

Proof There are two cases of P :

- $P = \langle \bar{h}, R, imr, \text{nil}, m \rangle$. There are two cases of m :
 - if $m = \text{loop } s$, then Rule (2.3) gives $P' = \langle \bar{h}, R, imr, \text{nil}, s; \text{loop } s \rangle$, and

- if $m = s; m'$, then Rules (2.4)–(2.9) ensure that there exists a state P' such that $P \rightarrow P'$.
- $P = \langle \bar{h}, R, imr, h^k :: \dots :: h^1 :: m :: \text{nil}, h \rangle$, $k \geq 0$. There are two cases of h :
 - if $h = \text{iret}$, then either $k = 0$ and $s = m :: \text{nil}$, and Rule (2.2) gives $P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, \text{nil}, m \rangle$, or $k > 0$ and hence $P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, h^{k-1} :: \dots :: h^1 :: m :: \text{nil}, h^k \rangle$, and
 - if $h = s; h'$, then Rules (2.4)–(2.9) ensure that there exists a state P' such that $P \rightarrow P'$.

■

We obtain the following result regarding the termination of interrupt calculus programs.

Theorem 2.1.1 (Nontermination) *No program can terminate.*

Proof Suppose a program p can terminate; that is, suppose $P_p \rightarrow^* P'$ and there is no P'' such that $P' \rightarrow P''$. Notice first that P_p is consistent by consistency criterion (1). From Lemma 2.1.1 and induction on the number of execution steps in $P_p \rightarrow^* P'$, we have that P' is consistent. From Lemma 2.1.2 we have that there exists P'' such that $P' \rightarrow P''$, a contradiction.

■

2.1.4 Monotonicity

We will differentiate two versions of the interrupt calculus: monotonic interrupt calculus programs and non-monotonic interrupt calculus programs.

A monotonic interrupt calculus programs is a program such that all handler callings observe the following behavioral property: when the handler returns (immediately after executing the `iret` statement), the `imr` value is less than or equal to the

imr value at the time when the handler is called. A non-monotonic interrupt calculus program does not follow this property.

We will focus on the monotonic interrupt calculus programs in the remaining sections of this chapter. Chatterjee et al. [12] present an enriched version of the interrupt calculus. They give an algorithm which does not require the monotonicity restriction. In addition, the algorithm they use is based on the context-free graph.

2.2 A type system for interrupt calculus

We now present a type system for the interrupt calculus. This type system type checks the monotonic version of the interrupt calculus.

2.2.1 Types

We will use imr values as types. When we intend an imr value to be used as a type, we will use the mnemonic device of writing it with a hat, for example, \widehat{imr} .

We will use the bitwise logical implication \leq as the subtype relation. For example, $101 \leq 111$. We will also use \leq to specify the relationship between an imr value and its type. When we want to express that an imr value imr has type \widehat{imr} , we will write $imr \leq \widehat{imr}$. The meaning of this expression is that \widehat{imr} is a conservative approximation of imr , that is, if a bit in imr is 1, then the corresponding bit in \widehat{imr} is also 1.

We use K to range over the integers, and we use δ to range over the nonnegative integers.

We use τ to range over intersection types of the form:

$$\bigwedge_{j=1}^q ((\widehat{imr})^j \xrightarrow{\delta^j} (\widehat{imr})^j).$$

We use $\bar{\tau}$ to range over a sequence $\tau_1 \dots \tau_n$; we will use the notation $\bar{\tau}(i) = \tau_i$.

2.2.2 Type rules

We will use the following forms of type judgments:

Type Judgment	Meaning
$\bar{\tau} \vdash_K p$	Program p type checks
$\bar{\tau} \vdash_K P$	Program state P type checks
$\bar{\tau}, \widehat{imr} \vdash_K \sigma$	Stack σ type checks
$\bar{\tau} \vdash h : \tau$	Interrupt handler h has type τ
$\bar{\tau}, \widehat{imr} \vdash_K m$	Main part m type checks
$\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}'$	Handler h type checks
$\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}'$	Statement s type checks

A judgment $\bar{\tau} \vdash_K p$ for a program is related to the concrete syntax used in Section 1.6 as follows. We can dissect the concrete syntax into four parts: (1) a maximum stack size K , (2) the types $\bar{\tau}$ for the handlers, (3) a main part m , and (4) a collection \bar{h} of handlers. When we talk about a program (m, \bar{h}) in the abstract syntax, the two other parts K and $\bar{\tau}$ seem left out. However, they reappear in the judgment: $\bar{\tau} \vdash_K (m, \bar{h})$. Thus, that judgment can be read simply as: “the program type checks.”

The judgment $\bar{\tau} \vdash_K P$ for a program state extends the typing of programs to program states.

The judgment $\bar{\tau}, \widehat{imr} \vdash_K m$ means that if the handlers are of type $\bar{\tau}$, and the imr has type \widehat{imr} , then m type checks. The integer K bounds the stack size so that it is at most K . We can view K as a “stack budget” in the sense that any time an element is placed on the stack, the budget goes down by one, and when an element is removed from the stack, the budget goes up by one. This type system ensures that the budget does not go below zero.

The judgment $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}'$ means that if the handlers are of type $\bar{\tau}$, and the imr has type \widehat{imr} , then h type checks, and at the point of returning from the handler, the imr has type \widehat{imr}' . The integer K means that during the call, the stack

will grow by at most K elements. Notice that “during the call” may include calls to other interrupt handlers.

The judgment $\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}'$ has a meaning similar to that of $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}'$.

For two sequences $\bar{h}, \bar{\tau}$ of the same length, we will use the abbreviation:

$$\bar{\tau} \vdash \bar{h} : \bar{\tau}$$

to denote the family of judgments

$$\bar{\tau} \vdash \bar{h}(i) : \bar{\tau}(i)$$

for all i in the common domain of \bar{h} and $\bar{\tau}$.

We will use the abbreviation:

$$safe(\bar{\tau}, \widehat{imr}, K) = \left[\begin{array}{l} \forall i \in 1 \dots n \\ \text{if } enabled(\widehat{imr}, i) \\ \text{then, whenever } \bar{\tau}(i) = \dots \wedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \wedge \dots, \\ \text{we have } \delta + 1 \leq K \end{array} \right].$$

The side condition of $safe(\bar{\tau}, \widehat{imr}, K)$ is used to guarantee that it is safe for an interrupt handler to be called. Intuitively, the stack should grow at most δ elements during the call, plus a return address for the call itself.

We will sometimes use imr_b , as a mnemonic, for the return imr value of an interrupt handler.

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau}, 0 \vdash_K m}{\bar{\tau} \vdash_K (m, \bar{h})} \quad (2.10)$$

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K m}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, nil, m \rangle} \quad (2.11)$$

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b \quad \bar{\tau}, \widehat{imr}_b \vdash_K \sigma}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, \sigma, h \rangle} \quad (2.12)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_{K+1} m}{\overline{\tau}, \widehat{imr} \vdash_K m :: \text{nil}} \quad (2.13)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_{K+1} h : \widehat{imr}_b \quad \overline{\tau}, \widehat{imr}_b \vdash_{K+1} \sigma}{\overline{\tau}, \widehat{imr} \vdash_K h :: \sigma} \quad (2.14)$$

$$\frac{\overline{\tau}, (\widehat{imr})^j \wedge \neg \mathbf{t}_0 \vdash_{\delta^j} h : (\widehat{imr})^j \quad j \in 1..n}{\overline{\tau} \vdash h : \bigwedge_{j=1}^n ((\widehat{imr})^j \xrightarrow{\delta^j} (\widehat{imr})^j)} \quad (2.15)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}}{\overline{\tau}, \widehat{imr} \vdash_K \text{loop } s} \quad \left[\text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.16)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}' \quad \overline{\tau}, \widehat{imr}' \vdash_K m}{\overline{\tau}, \widehat{imr} \vdash_K s; m} \quad (2.17)$$

$$\overline{\tau}, \widehat{imr} \vdash_K \text{iret} : \widehat{imr}' \quad \left[\widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}' \text{ and } \text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.18)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}' \quad \overline{\tau}, \widehat{imr}' \vdash_K h : \widehat{imr}''}{\overline{\tau}, \widehat{imr} \vdash_K s; h : \widehat{imr}''} \quad (2.19)$$

$$\overline{\tau}, \widehat{imr} \vdash_K x = e : \widehat{imr} \quad \left[\text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.20)$$

$$\overline{\tau}, \widehat{imr} \vdash_K \text{imr} = \text{imr} \wedge \text{imr}' : \widehat{imr} \wedge \text{imr}' \quad \left[\text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.21)$$

$$\overline{\tau}, \widehat{imr} \vdash_K \text{imr} = \text{imr} \vee \text{imr}' : \widehat{imr} \vee \text{imr}' \quad \left[\text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.22)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_K s_1 : \widehat{imr}' \quad \overline{\tau}, \widehat{imr} \vdash_K s_2 : \widehat{imr}'}{\overline{\tau}, \widehat{imr} \vdash_K \text{if0 } (x) s_1 \text{ else } s_2 : \widehat{imr}'} \quad \left[\text{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.23)$$

$$\frac{\overline{\tau}, \widehat{imr} \vdash_K s_1 : \widehat{imr}_1 \quad \overline{\tau}, \widehat{imr}_1 \vdash_K s_2 : \widehat{imr}_2}{\overline{\tau}, \widehat{imr} \vdash_K s_1; s_2 : \widehat{imr}_2} \quad (2.24)$$

$$\overline{\tau}, \widehat{imr} \vdash_K \mathbf{skip} : \widehat{imr} \quad \left[\mathit{safe}(\overline{\tau}, \widehat{imr}, K) \right] \quad (2.25)$$

Rule (2.10) is for type checking whole programs.

Rules (2.11)–(2.12) are for type checking program states. The actual imr value is abstracted to a type \widehat{imr} which is used to type check the current statement. In Rule (2.12), the last two hypotheses ensure that interrupts can return to their callers in a type-safe way. In particular, the last hypothesis in Rule (2.12) type checks the stack, which is done by Rules (2.13)–(2.14).

Rule (2.15) says that the type of handler is an intersection type, so the handler must have all of the component types of the intersection. For each component type, the annotation δ^j is used as the bound on how much the stack can grow during a call to the handler. Notice that an intersection of different components cannot be reduced to a single component. The rule type checks the handler with the master bit initially turned off.

Rules (2.16)–(2.25) are type rules for statements. They are flow-sensitive to the imr , and most of them have the side condition $\mathit{safe}(\overline{\tau}, \widehat{imr}, K)$. The side condition ensures that if an enabled interrupt occurs, then the handler can both be called and return in a type-safe way.

2.2.3 Stack-size boundedness

This subsection is devoted to type checking of type-annotated interrupt calculus programs and to proving the following soundness property about the type system of interrupt calculus: given a natural number K , once an interrupt calculus program is type checked, then it is guaranteed that the maximum stack size of executing the program is bounded by K .

Formally, for a program state P , define $maxStackSize(P)$ to be either

- the least $K \geq 0$ such that for all P' , if $P \rightarrow^* P'$, then $|P'.stk| \leq K$, or
- “infinite” if no such K exists.

We will show that if $\bar{\tau} \vdash_K p$, then $maxStackSize(P_p) \leq K$.

We now prove the following lemmas which will be used later in this section.

Lemma 2.2.1 (Safe-Guarantee, Statements) *If $\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}'$, then $safe(\bar{\tau}, \widehat{imr}, K)$.*

Proof By induction on the derivation of $\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}'$; we omit the details. ■

Lemma 2.2.2 (Safe-Guarantee, Handlers) *If $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}'$, then $safe(\bar{\tau}, \widehat{imr}, K)$.*

Proof By induction on the derivation of $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}'$, using Lemma 2.2.1; we omit the details. ■

Lemma 2.2.3 (Safe-Guarantee, Main) *If $\bar{\tau}, \widehat{imr} \vdash_K m$, then $safe(\bar{\tau}, \widehat{imr}, K)$.*

Proof By induction on the derivation of $\bar{\tau}, \widehat{imr} \vdash_K m$, using Lemma 2.2.1; we omit the details. ■

Lemma 2.2.4 (Safe-Weakening) *If $K_1 \leq K_2$ and $safe(\bar{\tau}, \widehat{imr}, K_1)$, then $safe(\bar{\tau}, \widehat{imr}, K_2)$.*

Proof From $K_1 \leq K_2$ and

$$safe(\bar{\tau}, \widehat{imr}, K_1) = \left[\begin{array}{l} \forall i \in 1 \dots n \\ \text{if } enabled(\widehat{imr}, i) \\ \text{then, whenever } \bar{\tau}(i) = \dots \wedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \wedge \dots, \\ \text{we have } \delta + 1 \leq K_1 \end{array} \right]$$

we have

$$\left[\begin{array}{l} \forall i \in 1 \dots n \\ \text{if } \text{enabled}(\widehat{imr}, i) \\ \text{then, whenever } \bar{\tau}(i) = \dots \wedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \wedge \dots, \\ \text{we have } \delta + 1 \leq K_2 \end{array} \right]$$

that is, $\text{safe}(\bar{\tau}, \widehat{imr}, K_2)$. ■

Lemma 2.2.5 (K-Weakening, Statements) *If $K_1 \leq K_2$ and $\bar{\tau}, \widehat{imr} \vdash_{K_1} s : \widehat{imr}'$, then $\bar{\tau}, \widehat{imr} \vdash_{K_2} s : \widehat{imr}'$.*

Proof We proceed by induction on the derivation of $\bar{\tau}, \widehat{imr} \vdash_{K_1} s : \widehat{imr}'$. There are six subcases, depending on which one of Rules (2.20)–(2.25) was last used in the derivation of $\bar{\tau}, \widehat{imr} \vdash_{K_1} s : \widehat{imr}'$.

- Rule (2.20). We have

$$\bar{\tau}, \widehat{imr} \vdash_{K_1} x = e : \widehat{imr} \quad \left[\text{safe}(\bar{\tau}, \widehat{imr}, K_1) \right].$$

From $K_1 \leq K_2$, $\text{safe}(\bar{\tau}, \widehat{imr}, K_1)$, and Lemma 2.2.4, we have $\text{safe}(\bar{\tau}, \widehat{imr}, K_2)$.

Hence, $\bar{\tau}, \widehat{imr} \vdash_{K_2} x = e : \widehat{imr}$.

- Rule (2.21). The proof is similar to that for Rule (2.20).
- Rule (2.22). The proof is similar to that for Rule (2.20).
- Rule (2.23). We have

$$\frac{\bar{\tau}, \widehat{imr} \vdash_{K_1} s_1 : \widehat{imr}' \quad \bar{\tau}, \widehat{imr} \vdash_{K_1} s_2 : \widehat{imr}'}{\bar{\tau}, \widehat{imr} \vdash_{K_1} \text{if0}(x) s_1 \text{ else } s_2 : \widehat{imr}'} \quad \left[\text{safe}(\bar{\tau}, \widehat{imr}, K_1) \right].$$

From the induction hypothesis, we have $\bar{\tau}, \widehat{imr} \vdash_{K_2} s_1 : \widehat{imr}'$ and $\bar{\tau}, \widehat{imr} \vdash_{K_2} s_2 : \widehat{imr}'$. From $K_1 \leq K_2$, $\text{safe}(\bar{\tau}, \widehat{imr}, K_1)$, and Lemma 2.2.4, we have $\text{safe}(\bar{\tau}, \widehat{imr}, K_2)$. Hence, $\bar{\tau}, \widehat{imr} \vdash_{K_2} \text{if0}(x) s_1 \text{ else } s_2 : \widehat{imr}'$.

- Rule (2.24). We have

$$\frac{\overline{\tau}, \widehat{imr} \vdash_{K_1} s_1 : \widehat{imr}_1 \quad \overline{\tau}, \widehat{imr}_1 \vdash_{K_1} s_2 : \widehat{imr}_2}{\overline{\tau}, \widehat{imr} \vdash_{K_1} s_1; s_2 : \widehat{imr}_2}.$$

From the induction hypothesis, we have $\overline{\tau}, \widehat{imr} \vdash_{K_2} s_1 : \widehat{imr}_1$ and $\overline{\tau}, \widehat{imr}_1 \vdash_{K_2} s_2 : \widehat{imr}_2$. Hence, $\overline{\tau}, \widehat{imr} \vdash_{K_2} s_1; s_2 : \widehat{imr}_2$.

- Rule (2.25). The proof is similar to that for Rule (2.20). ■

Lemma 2.2.6 (K-Weakening, Handlers) *If $K_1 \leq K_2$ and $\overline{\tau}, \widehat{imr} \vdash_{K_1} h : \widehat{imr}'$, then $\overline{\tau}, \widehat{imr} \vdash_{K_2} h : \widehat{imr}'$.*

Proof We proceed by induction on the derivation of $\overline{\tau}, \widehat{imr} \vdash_{K_1} h : \widehat{imr}'$. There are two subcases depending on which one of Rules (2.18)–(2.19) was the last one used in the derivation of $\overline{\tau}, \widehat{imr} \vdash_{K_1} h : \widehat{imr}'$.

- Rule (2.18). We have

$$\overline{\tau}, \widehat{imr} \vdash_{K_1} \text{iret} : \widehat{imr}' \quad \left[\widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}' \text{ and } \text{safe}(\overline{\tau}, \widehat{imr}, K_1) \right]$$

From $K_1 \leq K_2$, $\text{safe}(\overline{\tau}, \widehat{imr}, K_1)$, and Lemma 2.2.4, we have $\text{safe}(\overline{\tau}, \widehat{imr}, K_2)$.

From $\text{safe}(\overline{\tau}, \widehat{imr}, K_2)$ and $\widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}'$, we have

$$\overline{\tau}, \widehat{imr} \vdash_{K_2} \text{iret} : \widehat{imr}'$$

- Rule (2.19). We have

$$\frac{\overline{\tau}, \widehat{imr} \vdash_{K_1} s : \widehat{imr}' \quad \overline{\tau}, \widehat{imr}' \vdash_{K_1} h : \widehat{imr}''}{\overline{\tau}, \widehat{imr} \vdash_{K_1} s; h : \widehat{imr}''}.$$

From Lemma 2.2.5, we have

$$\overline{\tau}, \widehat{imr} \vdash_{K_2} s : \widehat{imr}'.$$

From the induction hypothesis, we have

$$\overline{\tau}, \widehat{imr}' \vdash_{K_2} h : \widehat{imr}''.$$

From $\bar{\tau}, \widehat{imr} \vdash_{K_2} s : \widehat{imr}'$ and $\bar{\tau}, \widehat{imr}' \vdash_{K_2} h : \widehat{imr}''$, we can use Rule (2.19) to derive $\bar{\tau}, \widehat{imr} \vdash_{K_2} s; h : \widehat{imr}''$.

■

Type preservation

We now prove the type preservation theorem which is stated in Theorem (2.2.2).

Theorem 2.2.1 (Single-step type preservation) *Suppose P is a consistent program state. If $\bar{\tau} \vdash_K P$, $K \geq 0$, and $P \rightarrow P'$, then $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$, where $K' = K + |P.stk| - |P'.stk|$.*

Proof There are nine cases depending on which one of Rules (2.1)–(2.9) was used to derive $P \rightarrow P'$.

- Rule (2.1). We have $\langle \bar{h}, R, imr, \sigma, a \rangle \rightarrow \langle \bar{h}, R, imr \wedge \neg t_0, a :: \sigma, \bar{h}(i) \rangle$ and $enabled(imr, i)$. Since P is consistent, there are two subcases.

Subcase 1: We have $P = \langle \bar{h}, R, imr, nil, m \rangle$ and

$P' = \langle \bar{h}, R, imr \wedge \neg t_0, m :: nil, \bar{h}(i) \rangle$. From $\bar{\tau} \vdash_K P$ and Rule (2.11), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K m}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, nil, m \rangle}$$

From $\bar{\tau}, \widehat{imr} \vdash_K m$, and Lemma 2.2.3, we have that:

$$safe(\bar{\tau}, \widehat{imr}, K) = \left[\begin{array}{l} \forall i \in 1 \dots n \\ \text{if } enabled(\widehat{imr}, i) \\ \text{then, whenever } \bar{\tau}(i) = \dots \wedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \wedge \dots, \\ \text{we have } \delta + 1 \leq K \end{array} \right]$$

is true. From $safe(\bar{\tau}, \widehat{imr}, K)$ and $enabled(\widehat{imr}, i)$, it follows that:

$$\bar{\tau}(i) = \dots \wedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \wedge \dots \quad \delta + 1 \leq K.$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$ and Rule (2.15), we have $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta} h_i : \widehat{imr}$. From $\delta \leq K - 1$, $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta} h_i : \widehat{imr}$, and Lemma 2.2.6, we have

$$\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{K-1} h_i : \widehat{imr}$$

From $\bar{\tau}, \widehat{imr} \vdash_K m$ and Rule (2.13), we have $\bar{\tau}, \widehat{imr} \vdash_{K-1} m :: \text{nil}$. From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $imr \wedge \neg \mathbf{t}_0 \leq \widehat{imr} \wedge \neg \mathbf{t}_0$, $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{K-1} h_i : \widehat{imr}$, $\bar{\tau}, \widehat{imr} \vdash_{K-1} m :: \text{nil}$, and $K' = K + |P.stk| - |P'.stk| = K - 1 \geq \delta \geq 0$, we can use Rule (2.12) to derive $\bar{\tau} \vdash_{K'} P'$.

Subcase 2: We have $P = \langle \bar{h}, R, imr, \sigma, h \rangle$, $P' = \langle \bar{h}, R, imr \wedge \neg \mathbf{t}_0, h :: \sigma, \bar{h}(i) \rangle$. From $\bar{\tau} \vdash_K P$ and Rule (2.12), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b \quad \bar{\tau}, \widehat{imr}_b \vdash_K \sigma}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, \sigma, h \rangle}$$

From $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b$, and Lemma 2.2.2, we have that

$$safe(\bar{\tau}, \widehat{imr}, K) = \left[\begin{array}{l} \forall i \in 1 \dots n \\ \text{if } enabled(\widehat{imr}, i) \\ \text{then, whenever } \bar{\tau}(i) = \dots \bigwedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \bigwedge \dots, \\ \text{we have } \delta + 1 \leq K \end{array} \right]$$

is true. From $safe(\bar{\tau}, \widehat{imr}, K)$ and $enabled(\widehat{imr}, i)$, it follows that

$$\bar{\tau}(i) = \dots \bigwedge (\widehat{imr} \xrightarrow{\delta} \widehat{imr}) \bigwedge \dots \quad \delta + 1 \leq K.$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$ and Rule (2.15), we have $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta} h_i : \widehat{imr}$. From $\delta \leq K - 1$, $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta} h_i : \widehat{imr}$, and Lemma 2.2.6, we have

$$\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{K-1} h_i : \widehat{imr}.$$

From $\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b$, $\bar{\tau}, \widehat{imr}_b \vdash_K \sigma$, and Rule (2.14), we have

$$\bar{\tau}, \widehat{imr} \vdash_{K-1} h :: \sigma.$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $imr \wedge \neg \mathbf{t}_0 \leq \widehat{imr} \wedge \neg \mathbf{t}_0$, $\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{K-1} h_i : \widehat{imr}$, $\bar{\tau}, \widehat{imr} \vdash_{K-1} h :: \sigma$, and $K' = K + |P.stk| - |P'.stk| = K - 1 \geq \delta \geq 0$, we can use Rule (2.12) to derive $\bar{\tau} \vdash_{K'} P'$.

- Rule (2.2). We have $\langle \bar{h}, R, imr, \sigma, iret \rangle \rightarrow \langle \bar{h}, R, imr \vee \mathbf{t}_0, \sigma', a \rangle$, and $\sigma = a :: \sigma'$. Since P is consistent, there are two subcases.

Subcase 1: We have $P = \langle \bar{h}, R, imr, m :: \text{nil}, iret \rangle$ and

$P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, \text{nil}, m \rangle$. From $\bar{\tau} \vdash_K P$, Rule (2.12), and Rule (2.13), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K iret : \widehat{imr}_b \quad \frac{\bar{\tau}, \widehat{imr}_b \vdash_{K+1} m}{\bar{\tau}, \widehat{imr}_b \vdash_K m :: \text{nil}}}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, m :: \text{nil}, iret \rangle}$$

From $\bar{\tau}, \widehat{imr} \vdash_K iret : \widehat{imr}_b$, we have $\widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}_b$. From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $imr \vee \mathbf{t}_0 \leq \widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}_b$, $\bar{\tau}, \widehat{imr}_b \vdash_{K+1} m$, and $K' = K + |P.stk| - |P'.stk| = K + 1$, we can use Rule (2.11) to derive $\bar{\tau} \vdash_{K'} P'$.

Subcase 2: We have $P = \langle \bar{h}, R, imr, h^k :: \sigma', iret \rangle$ and

$P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, \sigma', h^k \rangle$. From $\bar{\tau} \vdash_K P$, Rule (2.12), and Rule (2.14), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K iret : \widehat{imr}_b \quad \bar{\tau}, \widehat{imr}_b \vdash_K h^k :: \sigma'}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, h^k :: \sigma', iret \rangle}$$

where $\bar{\tau}, \widehat{imr}_b \vdash_K h^k :: \sigma'$ is derived as follows:

$$\frac{\bar{\tau}, \widehat{imr}_b \vdash_{K+1} h^k : \widehat{imr}_b^k \quad \bar{\tau}, \widehat{imr}_b^k \vdash_{K+1} \sigma'}{\bar{\tau}, \widehat{imr}_b \vdash_K h^k :: \sigma'}$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $imr \vee \mathbf{t}_0 \leq \widehat{imr} \vee \mathbf{t}_0 \leq \widehat{imr}_b$, $\bar{\tau}, \widehat{imr}_b \vdash_{K+1} h^k : \widehat{imr}_b^k$, $\bar{\tau}, \widehat{imr}_b^k \vdash_{K+1} \sigma'$, and $K' = K + |P.stk| - |P'.stk| = K + 1$ we can use Rule (2.12) to derive $\bar{\tau} \vdash_{K'} P'$.

- Rule (2.3). We have $\langle \bar{h}, R, imr, \text{nil}, \text{loop } s \rangle \rightarrow \langle \bar{h}, R, imr, \text{nil}, s; \text{loop } s \rangle$. From $\bar{\tau} \vdash_K P$, Rule (2.11), and Rule (2.16), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \frac{\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}}{\bar{\tau}, \widehat{imr} \vdash_K \text{loop } s}}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, \text{nil}, \text{loop } s \rangle}$$

From $\bar{\tau}, \widehat{imr} \vdash_K s : \widehat{imr}$, $\bar{\tau}, \widehat{imr} \vdash_K \text{loop } s$, and Rule (2.17) we have $\bar{\tau}, \widehat{imr} \vdash_K s; \text{loop } s$. From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $imr \leq \widehat{imr}$, $\bar{\tau}, \widehat{imr} \vdash_K s; \text{loop } s$, and $K' = K + |P.stk| - |P'.stk| = K$, we can use Rule (2.11) to derive $\bar{\tau} \vdash_{K'} P'$.

- Rule (2.4). We have $\langle \bar{h}, R, imr, \sigma, x = e; a \rangle \rightarrow \langle \bar{h}, R\{x \mapsto eval_R(e)\}, imr, \sigma, a \rangle$. Since P is consistent, there are two subcases.

Subcase 1: $P = \langle \bar{h}, R, imr, nil, x = e; m \rangle$ and

$P' = \langle \bar{h}, R\{x \mapsto eval_R(e)\}, imr, nil, m \rangle$.

From $\bar{\tau} \vdash_K P$, Rule (2.11), and Rule (2.17), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \frac{\bar{\tau}, \widehat{imr} \vdash_K x = e : \widehat{imr} \quad \bar{\tau}, \widehat{imr} \vdash_K m}{\bar{\tau}, \widehat{imr} \vdash_K x = e; m}}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, nil, x = e; m \rangle}.$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}, imr \leq \widehat{imr}, \bar{\tau}, \widehat{imr} \vdash_K m$, and $K' = K + |P.stk| - |P'.stk| = K$, we can use Rule (2.11) to derive $\bar{\tau} \vdash_{K'} P'$.

Subcase 2: $P = \langle \bar{h}, R, imr, \sigma, x = e; h \rangle$ and

$P' = \langle \bar{h}, R\{x \mapsto eval_R(e)\}, imr, \sigma, h \rangle$. From $\bar{\tau} \vdash_K P$, Rule (2.12), and

Rule (2.19), we have the derivation:

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad imr \leq \widehat{imr} \quad \frac{\bar{\tau}, \widehat{imr} \vdash_K x = e : \widehat{imr} \quad \frac{\bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b}{\bar{\tau}, \widehat{imr} \vdash_K x = e; h : \widehat{imr}_b} \quad \bar{\tau}, \widehat{imr}_b \vdash_K \sigma}{\bar{\tau}, \widehat{imr} \vdash_K x = e; h : \widehat{imr}_b}}{\bar{\tau} \vdash_K \langle \bar{h}, R, imr, \sigma, x = e; h \rangle}.$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}, imr \leq \widehat{imr}, \bar{\tau}, \widehat{imr} \vdash_K h : \widehat{imr}_b, \bar{\tau}, \widehat{imr}_b \vdash_K \sigma$, and $K' = K + |P.stk| - |P'.stk| = K$, we can use Rule (2.12) to derive $\bar{\tau} \vdash_{K'} P'$.

- Rules (2.5)–(2.9). The proofs are similar to that for Rule (2.4); we omit the details. ■

Theorem 2.2.2 (Multi-Step Type Preservation) *Suppose P is a consistent program state. If $\bar{\tau} \vdash_K P$, $K \geq 0$, and $P \rightarrow^* P'$, then $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$, where $K' = K + |P.stk| - |P'.stk|$.*

Proof We need to prove that

$\forall n \geq 0$, if $\bar{\tau} \vdash_K P$, $K \geq 0$, and $P \rightarrow^n P'$, then $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$,
where $K' = K + |P.stk| - |P'.stk|$.

We proceed by induction on n . In the base case of $n = 0$, we have $P = P'$, so $K' = K + |P.stk| - |P.stk| = K$. From $P' = P$ and $K' = K$, we have $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$.

In the induction step, assume that the property is true for n . Suppose $\bar{\tau} \vdash_K P$, $K \geq 0$, and $P \rightarrow^n P' \rightarrow P''$. From the induction hypothesis, we have $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$, where

$$K' = K + |P.stk| - |P'.stk| \quad (2.26)$$

From Lemma 2.1.1 we have that P' is consistent. From Theorem 2.2.1, we have $\bar{\tau} \vdash_{K''} P''$ and $K'' \geq 0$, where

$$K'' = K' + |P'.stk| - |P''.stk| \quad (2.27)$$

From Equations (2.26) and (2.27), we have

$$\begin{aligned} K'' &= K' + |P'.stk| - |P''.stk| \\ &= K + |P.stk| - |P'.stk| + |P'.stk| - |P''.stk| \\ &= K + |P.stk| - |P''.stk| \end{aligned}$$

as desired. ■

Theorem 2.2.3 (Stack Boundedness) *If $\bar{\tau} \vdash_K p$, then $\maxStackSize(P_p) \leq K$.*

Proof Suppose $\bar{\tau} \vdash_K p$. For $K \geq 0$ and any program state P' , we need to prove that if $P_p \rightarrow^* P'$, then $|P'.stk| \leq K$. Notice first that P_p is consistent, and that $\bar{\tau} \vdash_K P_p$ is derivable. From $\bar{\tau} \vdash_K P_p$, $K \geq 0$, $P_p \rightarrow^* P'$, and Theorem 2.2.2, we have $\bar{\tau} \vdash_{K'} P'$ and $K' \geq 0$, where $K' = K + |P_p.stk| - |P'.stk|$. From $K' = K + |P_p.stk| - |P'.stk|$ and $|P_p.stk| = 0$, we have $K' = K - |P'.stk|$, so, since $K' \geq 0$, we have $|P'.stk| \leq K$, as desired. ■

2.3 Type inference

In Section 2.2, we introduced a type system for interrupt calculus and described how the type checking process guarantees stack boundedness and certifies that the stack size is within a given bound. In an ideal world, the engineering of interrupt-driven software would proceed by first specifying the types of all interrupt handlers, then checking that those types guarantee a stack with a size within a desired bound, and finally, writing and type checking the actual code in a modular fashion.

In this section, we will answer the following fundamental question: How do we take interrupt-driven software and automatically annotate it with types that guarantee stack boundedness and a stack size within a given bound? Translated into the setting of the typed interrupt calculus, this question becomes a type inference problem. In Section 2.3.1, we summarize our results on complexity. After that, we analyze the complexity of the type inference problem.

2.3.1 Summary of the results on complexity

We present algorithms and lower bounds for two variants of the type inference problem for the interrupt calculus. Our main result is that, given a program and a bound on the stack size, a variant of type inference called stack-size checking is in PSPACE. In addition to the PSPACE result of Stack-Size Checking, there are other naturally-occurring type inference problems that are PSPACE-complete. For example, type inference with finite types and atomic subtyping over a general partial order of base types is PSPACE-complete [21, 57].

We study four problems, concerning an untyped program p , a natural number K , and types $\bar{\tau}$:

- **Maximum Stack Size.** Given p, K , is $\maxStackSize(P_p) \leq K$?
- **Type Checking.** Given $p, K, \bar{\tau}$, is $\bar{\tau} \vdash_K p$ derivable?

- **Stack-Size Checking.** Given p, K , does there exist $\bar{\tau}$ such that $\bar{\tau} \vdash_K p$ is derivable?
- **Type Inference.** Given p , does there exist $\bar{\tau}$ and K such that $\bar{\tau} \vdash_K p$ is derivable?

P_p is the initial program state for a program p , $\text{maxStackSize}(P_p)$, defined in subsection 2.2.3, is the least upper bound on the sizes of all stacks which can occur during an execution starting at P_p , and $\bar{\tau} \vdash_K p$ is a type judgment which says that, with the types $\bar{\tau}$ for the interrupt handlers, the program p type checks and has a maximum stack size of no more than K . The table in Figure 2.3.1 below summarizes our results:

Problem	Lower Bound	Upper Bound
Maximum Stack Size	PSPACE-hard (Theorem 2.3.5)	<i>Open</i>
Type Checking	<i>Open</i>	polynomial time (Sec. 2.2)
Stack-Size Checking	PSPACE-hard (Theorem 2.3.7)	PSPACE (Theorem 2.3.3)
Type Inference	<i>Open</i>	PSPACE (Theorem 2.3.4)

Figure 2.3. Summary of results

A basic observation which can be made is that the types can be exponentially large in the size of the program. The Type Checking problem is in polynomial time because the types are part of the input. Thus, Type Checking is polynomial in a possibly large input. A related phenomenon can happen with ML programs [32, 33].

Our approach to showing the PSPACE upper bounds is to first reduce the Stack-Size Checking problem to a constraint problem and then solve the constraint problem in PSPACE. We also show that the Type Inference problem is polynomial-time reducible to the Stack-Size Checking problem. Our approach to showing the PSPACE lower bounds is to reduce the Maximal Stack Size problem to the Stack-Size Checking problem, and then show that the Maximal Stack Size problem is PSPACE-hard.

2.3.2 Constraints

We build a constraint system out of the un-typed interrupt calculus program whose solutions are all viable types for the program.

Let IMR be the set of bit vectors of length $n + 1$: $IMR = \{\underbrace{b \dots b}_n \mid b = 0, 1\}$. Let $\mathcal{V} = \cup_{i=0}^n V_i$ be a set of variables ranging over IMR , where the sets V_1, \dots, V_n are pairwise disjoint. We assume that there are distinct variables $v_i^{init} \in V_i$, for $i \in 0..n$. We use v, w to range over \mathcal{V} . The constraints are of the following forms

$$\left. \begin{array}{l} (v = v') \\ (v \leq v') \\ (v = v' \wedge imr') \\ (v = v' \vee imr') \end{array} \right\} v, v' \in V_i \quad i \in 0..N$$

$$(v) \quad v \in V_i \quad i \in 0..N$$

We will use $X \rightarrow_p Y$ to denote a space of partial function from X to Y . We will use $\mathcal{D}(F)$ to denote the domain of a partial function F .

A solution to the constraint set C is a pair (\bar{F}, \bar{G}) :

$$\left. \begin{array}{l} F_i : IMR \rightarrow_p (V_i \rightarrow IMR) \\ G_i : IMR \rightarrow_p Nat \\ \mathcal{D}(F_i) = \mathcal{D}(G_i) \end{array} \right\} \forall i \in 0..N$$

$$\mathcal{D}(F_0) = \mathcal{D}(G_0) = \{0 \underbrace{0 \dots 0}_n\}$$

$$imr \in \mathcal{D}(F_i) \Rightarrow F_i(imr)v_i^{init} = imr \wedge \neg \mathbf{t}_0 \quad \forall i \in 1..N$$

such that for $imr \in \mathcal{D}(F_i)$ and $v, v' \in V_i$

Constraints	Solutions
$(v = v')$	$F_i(imr)v = F_i(imr)v'$
$(v \leq v')$	$F_i(imr)v \leq F_i(imr)v'$
$(v = v' \wedge imr')$	$F_i(imr)v = F_i(imr)v' \wedge imr'$
$(v = v' \vee imr')$	$F_i(imr)v = F_i(imr)v' \vee imr'$
(v)	$\forall j \in 1..N : enabled(F_i(imr)v, j) \Rightarrow$ $F_i(imr)v \in \mathcal{D}(F_j) \wedge$ $G_j(F_i(imr)v) + 1 \leq G_i(imr).$

Intuitively, a solution (\bar{F}, \bar{G}) consists of: (1) a sequence of functions \bar{F} that each, for a given initial imr value, maps variables to imr values and (2) a sequence of functions \bar{G} that each, for a given initial imr value, gives a integer which is intended to be the δ for a given component of an intersection type.

For each $i \in 0..n$, we use C_i^{imr} to denote the union of (1) the constraints of the first four forms where the variables are in V_i and (2) the constraint $v_i^{init} = imr \wedge \neg \mathbf{t}_0$. the last constraint is satisfied by (\bar{F}, \bar{G}) if and only if $F_i(imr)v_i^{init} = imr \wedge \neg \mathbf{t}_0$.

2.3.3 From types to constraints

For a program p , we will define a constraint system that uses the following variables: (1) for each occurrence of a , the variables v_a and w_a , (2) for each occurrence of `iret`, the variable v'_{iret} , in addition to the variables $v_i^{init}, i \in 0..n$.

From a program p , we generate a set of constraints $\mathcal{C}(p)$ in the following manner.

for each occurrence of s :	constraint
$s_1; m$	$(v_{s_1} = v_s), (v_m = w_{s_1})$
loop s_1	$(v_{s_1} = v_s), (v_{s_1} = w_{s_1}), (v_s)$
$s_1; h$	$(v_{s_1} = v_s), (w_{s_1} = v_h)$
iret	$(w_s = v_s \vee \mathbf{t}_0), (v_s), (v'_s = v^{init} \vee \mathbf{t}_0), (w_s \leq v'_s)$
$s_1; s_2$	$(v_{s_1} = v_s), (w_{s_1} = v_{s_2}), (w_s = w_{s_2})$
$x = e$	$(w_s = v_s), (v_s)$
$\text{imr} = \text{imr} \wedge \text{imr}$	$(w_s = v_s \wedge \text{imr}), (v_s)$
$\text{imr} = \text{imr} \vee \text{imr}$	$(w_s = v_s \vee \text{imr}), (v_s)$
if0 (x) then s_1 else s_2	$(v_s), (w_s = w_{s_1}), (w_s = w_{s_2}), (v_{s_1} = v_s), (v_{s_2} = v_s)$
skip	$(w_s = v_s), (v_s)$

where $v_s \in V_i \quad i \in 0..N$. (If s occurs more than once, then the variables v_s and w_s will be ambiguous. However, it will always be clear from the context which occurrence is meant.)

Notice that the size of $\mathcal{C}(p)$ is linear in the size of p . The following connection between Stack-Size Checking and constraint solving can be proved using standard techniques.

Theorem 2.3.1 *Let p be a program and let K be a natural number. There exists $\bar{\tau}$ such that $\bar{\tau} \vdash_K p$ if and only if the constraint system $\mathcal{C}(p)$ has a solution (\bar{F}, \bar{G}) such that $G_0(0) \leq K$.*

Proof Suppose that there exists $\bar{\tau}$ such that $\bar{\tau} \vdash_K p$. Let

$$\mathcal{D}(\bar{\tau}(i)) = \{\widehat{\text{imr}} \mid \bar{\tau}(i) = \dots \wedge \widehat{\text{imr}} \xrightarrow{\delta} \widehat{\text{imr}} \wedge \dots\}$$

We construct the domain of the solution (\bar{F}, \bar{G}) :

$$\mathcal{D}(F_i) = \mathcal{D}(G_i) = \mathcal{D}(\bar{\tau}(i)) \quad \forall i \in 1..N$$

$$\mathcal{D}(F_0) = \mathcal{D}(G_0) = 0$$

We construct \bar{G} as the following:

$$G_i(\widehat{imr}) = \delta \text{ if } \bar{\tau}(i) = \dots \wedge \widehat{imr} \xrightarrow{\delta} \widehat{imr} \wedge \dots$$

where $i \in 1..N$ and $G_0(0) = K$.

We construct \bar{F} by the following method. (We only give the construction of the F function for the statement `iret` and `loop`; the other cases are similar.)

For the statement `iret`, we have the following fragment from its occurrence in the type derivation tree:

$$\frac{\bar{\tau}, \widehat{imr}_1 \vdash_{\delta} \text{iret} : \widehat{imr} \quad \left[\widehat{imr}_1 \vee \mathbf{t}_0 \leq \widehat{imr} \text{ and } \text{safe}(\bar{\tau}, \widehat{imr}_1, \delta) \right]}{\dots} \\ \frac{\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta} h_i : \widehat{imr}}{\bar{\tau} \vdash h_i : \dots \wedge \widehat{imr} \xrightarrow{\delta} \widehat{imr} \wedge \dots}$$

We construct $F_i(\widehat{imr})$ as follows: $F_i(\widehat{imr})v_{\text{iret}} = \widehat{imr}_1$, $F_i(\widehat{imr})w_{\text{iret}} = \widehat{imr}_1 \vee \mathbf{t}_0$, $F_i(\widehat{imr})v'_{\text{iret}} = F_i(\widehat{imr})v^{\text{init}} \vee \mathbf{t}_0 = \widehat{imr}$, therefore, we have $F_i(\widehat{imr})w_{\text{iret}} = \widehat{imr}_1 \vee \mathbf{t}_0 \leq \widehat{imr} = F_i(\widehat{imr})v'_{\text{iret}}$.

For the statement `loop`, we have the following fragment from its occurrence in the type derivation tree:

$$\frac{\bar{\tau}, \widehat{imr}_1 \vdash_K s_1 : \widehat{imr}_1 \quad \left[\text{safe}(\bar{\tau}, \widehat{imr}_1, K) \right]}{\bar{\tau}, \widehat{imr}_1 \vdash_K \text{loop } s_1} \\ \frac{\dots}{\bar{\tau}, 0 \vdash_K m}$$

We construct $F_i(\widehat{imr})$ as follows: $F_0(\widehat{imr})v_{\text{loop } s_1} = F_0(\widehat{imr})v_{s_1} = \widehat{imr}_1$, $F_0(\widehat{imr})w_{s_1} = F_0(\widehat{imr})v_{s_1} = \widehat{imr}_1$.

The \bar{F} constructed by the above method trivially satisfy the constraints of the forms: $(v = v')$, $(v = v' \wedge imr_c)$, $(v = v' \vee imr_c)$, and $(v \leq v')$.

It remains to be shown that the constraints of the form (v) are satisfied by (\bar{F}, \bar{G}) . Here we only prove that the constraint (v_{iret}) generated by statement `iret` is satisfied. Other constraints are proved in a similar manner.

Since

$$\frac{\bar{\tau}, \widehat{imr}_1 \vdash_{\delta'} \text{iret} : \widehat{imr} \quad \left[\widehat{imr}_1 \vee \mathbf{t}_0 \leq \widehat{imr} \text{ and } \text{safe}(\bar{\tau}, \widehat{imr}_1, \delta') \right]}{\dots}$$

$$\frac{\bar{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta'} h_i : \widehat{imr}}{\bar{\tau} \vdash h_i : \dots \wedge \widehat{imr} \xrightarrow{\delta'} \widehat{imr} \wedge \dots}$$

we have $\widehat{imr} \in \mathcal{D}(F_i) = \mathcal{D}(G_i)$, $F_i(\widehat{imr})v_{\text{iret}} = \widehat{imr}_1$ and $G_i(\widehat{imr}) = \delta'$. Since $\text{safe}(\bar{\tau}, \widehat{imr}_1, \delta')$, we have

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ \text{if } \text{enabled}(\widehat{imr}_1, j) \\ \text{then, whenever } \bar{\tau}(j) = \dots \wedge (\widehat{imr}_1 \xrightarrow{\delta} \widehat{imr}_1) \wedge \dots, \\ \text{we have } \delta + 1 \leq \delta' \end{array} \right].$$

Therefore, if $\text{enabled}(\widehat{imr}_1, j)$ is true, then $\widehat{imr}_1 \in \mathcal{D}(G_j) = \mathcal{D}(F_j)$ and $G_j(\widehat{imr}_1) = \delta$.

Hence, we have

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ \text{enabled}(F_i(\widehat{imr})v_{\text{iret}}, j) \Rightarrow \\ F_i(\widehat{imr})v_{\text{iret}} \in \mathcal{D}(F_j) \wedge \\ G_j(F_i(\widehat{imr})v_{\text{iret}}) + 1 \leq G_i(\widehat{imr}) \end{array} \right]$$

which is constraint (v_{iret}) .

Hence, (\bar{F}, \bar{G}) is a solution to constraints $\mathcal{C}(p)$.

Conversely, suppose the constraint system $\mathcal{C}(p)$ has a solution (\bar{F}, \bar{G}) and $G_0(0) \leq K$. We build types $\bar{\tau}$ of the handlers \bar{h} as follows:

$$\bar{\tau}(i) = \bigwedge_{\widehat{imr} \in \mathcal{D}(F_i)} \widehat{imr} \xrightarrow{G_i(\widehat{imr})} \widehat{imr}$$

and $F_i(\widehat{imr})v^{\text{init}} = \widehat{imr} \wedge \mathbf{t}_0$. where $i \in 1..N$.

We inductively build type derivations of $\bar{\tau}, 0 \vdash_K m$ and $\bar{\tau} \vdash \bar{h}$ from the (\bar{F}, \bar{G}) .

We only show the building of derivation of iret and loop . Others are similarly built.

For `iret`, the constraint variables are v_{iret} , w_{iret} and v'_{iret} . Suppose we are building the type derivation of `iret` in the following context

$$\frac{\overline{\tau}, \widehat{imr}_1 \vdash_{\delta'} \text{iret} : \widehat{imr} \quad \left[\widehat{imr}_1 \vee \mathbf{t}_0 \leq \widehat{imr} \text{ and } \text{safe}(\overline{\tau}, imr_1, \delta') \right]}{\dots}$$

$$\frac{\overline{\tau}, \widehat{imr} \wedge \neg \mathbf{t}_0 \vdash_{\delta'} h_i : \widehat{imr}}{\overline{\tau} \vdash h_i : \dots \wedge \widehat{imr} \xrightarrow{\delta'} \widehat{imr} \wedge \dots}$$

where $\widehat{imr} \in \mathcal{D}(F_i) = \mathcal{D}(G_i)$, $\delta' = G_i(\widehat{imr})$, $F_i(\widehat{imr})v_{\text{iret}} = \widehat{imr}_1$, $F_i(\widehat{imr})w_{\text{iret}} = F_i(\widehat{imr})v_{\text{iret}} \vee \mathbf{t}_0 = \widehat{imr}_1 \vee \mathbf{t}_0$, $F_i(\widehat{imr})v'_{\text{iret}} = F_i(\widehat{imr})v^{\text{init}} \vee \mathbf{t}_0 = \widehat{imr}$. It is necessary to

1. $\widehat{imr}_1 \vee \mathbf{t}_0 \leq \widehat{imr}$
2. the $\text{safe}(\overline{\tau}, imr_1, K)$ condition is satisfied

Since `iret` generates the following constraints, which the solutions satisfy, we have:

$$(w_{\text{iret}} \leq v'_{\text{iret}})$$

$$F_i(\widehat{imr})w_{\text{iret}} = \widehat{imr}_1 \vee \mathbf{t}_0 \leq F_i(\widehat{imr})v'_{\text{iret}} = \widehat{imr}$$

So (1) is satisfied.

Since the solutions also satisfy the constraint (v_{iret}) generated at `iret`, we have

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ \text{enabled}(F_i(\widehat{imr})v_{\text{iret}}, j) \Rightarrow \\ F_i(\widehat{imr})v_{\text{iret}} \in \mathcal{D}(F_j) \wedge \\ G_j(F_i(\widehat{imr})v_{\text{iret}}) + 1 \leq G_i(\widehat{imr}) \end{array} \right]$$

which is

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ \text{if } \text{enabled}(\widehat{imr}_1, j) \\ \text{then, whenever } \overline{\tau}(j) = \dots \wedge (\widehat{imr}_1 \xrightarrow{\delta} \widehat{imr}_1) \wedge \dots, \\ \text{we have } \delta + 1 \leq \delta' \end{array} \right]$$

where $\delta = G_j(F_i(\widehat{imr})v_{\text{iret}}) = G_j(\widehat{imr}_1)$. Therefore, 2 is satisfied.

For `loop`, suppose we are building its type derivation in the following context

$$\frac{\frac{\overline{\tau}, \widehat{imr}_1 \vdash_K s_1 : \widehat{imr}_1 \quad \left[safe(\overline{\tau}, \widehat{imr}_1, K) \right]}{\overline{\tau}, \widehat{imr}_1 \vdash_K \text{loop } s_1}}{\dots}{\overline{\tau}, 0 \vdash_K m}$$

where $0 \in \mathcal{D}(F_0) = \mathcal{D}(G_0)$ and $F_0(\widehat{imr})v_{\text{loop } s_1} = F_0(\widehat{imr})v_{s_1} = F_0(\widehat{imr})w_{s_1} = \widehat{imr}_1$.

We now need to show that the $safe(\overline{\tau}, imr_1, K)$ condition is satisfied.

Since the solutions satisfy the constraint $(v_{\text{loop } s_1})$, we have

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ enabled(F_0(\widehat{imr})v_{\text{loop } s_1}, j) \Rightarrow \\ F_0(\widehat{imr})v_{\text{loop } s_1} \in \mathcal{D}(F_j) \wedge \\ G_j(F_i(\widehat{imr})v_{\text{loop } s_1}) + 1 \leq G_0(0) \leq K \end{array} \right]$$

which is

$$\left[\begin{array}{l} \forall j \in 1 \dots n \\ \text{if } enabled(\widehat{imr}_1, j) \\ \text{then, whenever } \overline{\tau}(j) = \dots \wedge (\widehat{imr}_1 \xrightarrow{\delta} \widehat{imr}_1) \wedge \dots, \\ \text{we have } \delta + 1 \leq K \end{array} \right]$$

where $\delta = G_j(F_0(\widehat{imr})v_{\text{loop } s_1}) = G_j(\widehat{imr}_1)$. ■

2.3.4 Solving constraints

A *simple constraint system* is a constraint system with the property that for each $i \in 0..n$, C_i^{imr} either has a unique solution or is unsatisfiable. It is straightforward to show that every constraint system $\mathcal{C}(p)$ is simple. So, in the remainder of this section, we focus on solving simple constraint systems.

For a simple constraint system C , define:

$$D = \{ error \} \cup \{ \overline{F} \mid \exists \overline{G} :$$

$$\left. \begin{array}{l} F_i : IMR \rightarrow_p (V_i \rightarrow IMR) \\ G_i : IMR \rightarrow_p Nat \\ \mathcal{D}(F_i) = \mathcal{D}(G_i) \end{array} \right\} \forall i \in 0..N$$

$$\mathcal{D}(F_0) = \mathcal{D}(G_0) = \{0 \underbrace{0 \dots 0}_n\}$$

$\forall i \in 1..n$, if $imr \in \mathcal{D}(F_i)$, then

$F_i(imr)$ is the unique solution of C_i^{imr}

and, for each constraint (v) , where $v \in V_i$,

$$\forall j \in 1..N : \text{enabled}(F_i(imr)v, j) \wedge F_i(imr)v \in \mathcal{D}(F_j) \Rightarrow \\ G_j(F_i(imr)v) + 1 \leq G_i(imr) \}.$$

We can consider D to be the space of candidates for the \bar{F} part of a solution to a simple constraint system, or *error* in case no solution exists. We equip D with an ordering \leq which is defined as follows:

$$\forall d \in D : d \leq \text{error}$$

$$\bar{F} \leq \bar{F}' \text{ if and only if } \forall i, imr : F_i(imr) = \perp \vee F_i(imr) = F_i'(imr).$$

Notice that (D, \leq) is a finite partial order in which $\lambda imr. \perp$ is the bottom element and *error* is the top element.

Define

$$\begin{array}{l} \psi : D \rightarrow D \\ \psi(\text{error}) = \text{error} \\ \psi(\bar{F}) = \left\{ \begin{array}{l} \bar{F}' \quad \bar{F}' = \bar{F} \cup \\ \quad \cup \{F_j : F_i(imr)v \mapsto \text{the unique solution of } C_j^{F_i(imr)v} \mid \\ \quad i \in 0..n, imr \in \mathcal{D}(F_i), (v) \in C \text{ where } v \in V_i, j \in 1..n : \\ \quad \text{enabled}(F_i(imr)v, j)\} \wedge \\ \exists \bar{G} : G_0(0) \leq K \wedge \\ \forall i \in 0..n, \forall imr \in \mathcal{D}(F_i), \forall (v) \in C \text{ where } v \in V_i, \forall j \in 1..n : \\ \quad \text{enabled}(F_i(imr)v, j) \Rightarrow G_j(F_i(imr)v) + 1 \leq G_i(imr) \\ \text{error} \quad \text{otherwise.} \end{array} \right. \end{array}$$

The case of “otherwise” covers two subcases. First, there may be a situation where:

$$\begin{aligned} & \exists i \in 0..n, \exists imr \in \mathcal{D}(F_i), \exists (v) \in C \text{ where } v \in V_i, \exists j \in 1..n : \\ & \text{enabled}(F_i(imr)v, j) \wedge C_j^{F_i(imr)v} \text{ is unsatisfiable,} \end{aligned}$$

that is, we would like to define F_j on $F_i(imr)v$, but no suitable definition is possible.

Second, there may be a situation where:

$$\begin{aligned} & \forall \bar{G}, G_0(0) > K \vee \exists i \in 0..n, \exists imr \in \mathcal{D}(F_i), \exists (v) \in C \text{ where } v \in V_i, \exists j \in 1..n : \\ & \text{enabled}(F_i(imr)v, j) \wedge F_i(imr)v \in \mathcal{D}(F_j) \wedge G_j(F_i(imr)v) + 1 \not\leq G_i(imr), \end{aligned}$$

that is, we cannot find a suitable \bar{G} to pair up with \bar{F} .

Notice that ψ is monotone. From the least fixed-point theorem, we have that ψ has a least fixed point, which we denote $\text{lfp } \psi$, and that $\text{lfp } \psi = \sqcup_{n \geq 0} \psi^n(\lambda imr. \perp)$. It is straightforward to prove the following relationship between solutions of C and fixed points of ψ .

Lemma 2.3.1 *Given a simple constraint system C and K, \bar{F} , we have that there exists \bar{G} such that (\bar{F}, \bar{G}) satisfies C and $G_0(0) \leq K$ if and only if $\bar{F} \neq \text{error}$ and \bar{F} is a fixed point of ψ .*

Lemma 2.3.2 *Given a simple constraint system C and a natural number K , we have that C has a solution (\bar{F}, \bar{G}) such that $G_0(0) \leq K$ if and only if $\text{lfp } \psi \neq \text{error}$.*

Proof Suppose $\text{lfp } \psi \neq \text{error}$. From Lemma 2.3.1 we have that there exists \bar{G} such that (\bar{F}, \bar{G}) satisfies C and $G_0(0) \leq K$.

Conversely, suppose C has solution (\bar{F}, \bar{G}) such that $G_0(0) \leq K$. From Lemma 2.3.1 we have that $\bar{F} \neq \text{error}$ and \bar{F} is a fixed point of ψ . Since \bar{F} is a fixed point of ψ we have $\text{lfp } \psi \leq \bar{F}$, and since $\bar{F} \neq \text{error}$, we have $\text{lfp } \psi \neq \text{error}$. ■

From Lemma 2.3.2 we know that we can check whether C has a solution (\bar{F}, \bar{G}) such that $G_0(0) \leq K$ simply by checking whether $\text{lfp } \psi \neq \text{error}$. One approach to checking the condition is to first compute $\text{lfp } \psi$ and then check whether it is equal

to *error*. It is straightforward to use the characterization $lfp \psi = \sqcup_{n \geq 0} \psi^n(\lambda imr. \perp)$ to compute $lfp \psi$ in exponential time. If the goal is to annotate a program with inferred types, then this is the right approach. However, if all we are concerned with is whether inferred types exist, then we can use the nondeterministic algorithm which will be presented in the proof of Theorem 2.3.2 in the following section.

2.3.5 Stack-size checking is in PSPACE

Theorem 2.3.2 *Given a simple constraint system C and natural number K , we can decide in PSPACE whether C has a solution (\bar{F}, \bar{G}) such that $G_0(0) \leq K$.*

Proof Since $PSPACE = co-NPSPACE$, the following co-NPSPACE algorithm is sufficient.

Step 0: Let $imr = 0$, $i = 0$, $F = \lambda imr. \perp$, and $d = 0$.

Step 1: If C_i^{imr} is unsatisfiable, then terminate with error. Otherwise, let F be the unique solution of C_i^{imr} .

Step 2: Nondeterministically choose a constraint (v) where $v \in V_i$. If there does not exist $j \in \{1..n\}$ such that $enabled(F(imr)v, j)$ is true, then terminate with success.

Otherwise, nondeterministically choose $j \in \{1..n\}$ such that $enabled(F(imr)v, j)$ is true; we let $d = d + 1$, $imr = F(imr)v$, $i = j$.

If $d > K$, then we terminate the algorithm with error, otherwise, repeat **Step 1**.

This algorithm nondeterministically checks whether $lfp \psi = error$, using polynomial space for storing each F -function plus some extra constant space. ■

Theorem 2.3.3 *Stack-Size Checking is in PSPACE.*

Proof Combine Theorem 2.3.1, and Theorem 2.3.2. ■

2.3.6 Type inference is in PSPACE

Lemma 2.3.3 *Type Inference is polynomial-time reducible to Stack-Size Checking.*

Proof Given an instance p of the Type Inference problem, where p has n interrupt handlers, we define an instance $p, 2^n$ of the Stack Size Checking problem.

We need to prove that there exists $\bar{\tau}, K$ such that $\bar{\tau} \vdash_K p$ is derivable if and only if there exists $\bar{\tau}$ such that $\bar{\tau} \vdash_{2^n} p$ is derivable.

The “if” direction is immediate. Conversely, suppose $\exists \bar{\tau}, K$ such that $\bar{\tau} \vdash_K p$ is derivable and that $\bar{\tau}(i) = \bigwedge_{\forall j} (\widehat{imr}_i^j \xrightarrow{\delta_i^j} \widehat{imr}_i^j)$.

In the derivation of $\bar{\tau} \vdash_K p$, every safe condition is of the form $\text{safe}(\bar{\tau}, \widehat{imr}, U)$, where U is either K or δ_i^j . If we collect all the inequalities in each safe condition, we then have a set of inequalities I of the form $\delta_{i'}^{j'} + 1 \leq \delta_i^j$ and $\delta_i^j + 1 \leq K$.

We can represent I by a directed graph G consisting of nodes δ_i^j and K , in addition, there is an edge from a to b iff the inequality $b + 1 \leq a$ is in I . It is clear that the least value we can assign to node a in G is equal to the longest path starting from a . The graph G should be acyclic because, otherwise, at least one node in G would be assigned an infinite value.

Suppose that $\delta_{i'}^{j'}$ is reachable from δ_i^j and $imr_{i'}^{j'} = imr_i^j$. Then there exists a node U reachable from δ_i^j and an edge from U to $\delta_{i'}^{j'}$, which means that $\delta_{i'}^{j'} + 1 \leq U$ is in I . By definition of the safe condition, if $imr_{i'}^{j'} = imr_i^j$, then $\delta_{i'}^{j'} + 1 \leq U$ is in I iff $\delta_i^j + 1 \leq U$ is in I . Thus, there exists an edge from U to δ_i^j , and, consequently, there is a cycle in G from δ_i^j to U and back to δ_i^j .

Therefore, if $\delta_{i'}^{j'}$ is reachable from δ_i^j , then $imr_{i'}^{j'} \neq imr_i^j$. For any two nodes δ_i^j and $\delta_{i'}^{j'}$ on any path of G , we have $imr_i^j \neq imr_{i'}^{j'}$. Since there are at most 2^n number of distinct imr values, the length of the longest path starting from K is at most 2^n .

If we construct $\bar{\tau}'$ by replacing every δ_i^j in $\bar{\tau}$ with the length of the longest path from δ_i^j in G and let K' equal to the length of longest path from K in G , then $\bar{\tau}' \vdash_{K'} p$ is still derivable.

Since $K' \leq 2^n$, $\bar{\tau}' \vdash_{2^n} p$ is derivable as well. ■

Theorem 2.3.4 *Type Inference is in PSPACE.*

Proof Combine Lemma 2.3.3 and Theorem 2.3.3. ■

2.3.7 Maximum stack size problem is PSPACE-hard

We now prove that the maximum stack size problem is PSPACE-hard. We first show that the problem is co-NP-hard and, then, further prove that it is PSPACE-hard.

We define a subclass of monotonic interrupt calculus which we call simple interrupt calculus and show the maximum stack size problem is co-NP-hard and PSPACE-hard for this class. It follows that maximum stack size is PSPACE-hard for monotonic interrupt driven programs.

For imr', imr'' where $imr'(0) = 0$ and $imr''(0) = 0$, define $\mathcal{H}(imr'; imr'')$ to be the interrupt handler

$$\begin{aligned} imr &= imr \wedge \neg imr'; \\ imr &= imr \vee (t_0 \vee imr''); \\ imr &= imr \wedge \neg(t_0 \vee imr''); \\ &iret. \end{aligned}$$

A *simple interrupt calculus program* is an interrupt calculus program whose main part is of the form

$$\begin{aligned} imr &= imr \vee (imr_S \vee t_0); \\ &loop\ skip \end{aligned}$$

where $imr_S(0) = 0$ and every interrupt handler is of the form $\mathcal{H}(imr'; imr'')$. Intuitively, a handler of a simple interrupt calculus program first disables some handlers, it then enables other handlers and enables interrupt handling. Such a situation would allow a handler to be interrupted by other handlers. After that, it disables interrupt

handling, and makes certain that the handlers that are enabled on exit are a subset of those that were enabled on entry to the handler.

For a handler $h(i)$ of the form $\mathcal{H}(imr'; imr'')$, we define function $f_i(imr) = imr \wedge (\neg imr') \vee imr''$. Given a simple interrupt calculus program p , we define a directed graph $G(p) = (V, E)$ such that

- $V = \{ imr \mid imr(0) = 0 \}$,
- $E = \{ (imr, f_i(imr), i) \mid t_i \leq imr \}$ is a set of labeled edges from imr to $f_i(imr)$ with label $i \in \{1..n\}$.

The edge $(imr, f_i(imr), i)$ in $G(p)$ represents the call to the interrupt handler $h(i)$ when imr value is imr . We define imr_S as the start node of $G(p)$ and we define $\mathcal{M}(imr)$ as the longest path in $G(p)$ from node imr . The notation $\mathcal{M}(imr)$ is ambiguous because it leaves the graph unspecified; however, in all cases below, the graph in question can be inferred from the context.

Lemma 2.3.4 *For a simple interrupt calculus program p , we have that $maxStackSize(P_p) = |\mathcal{M}(imr_S)|$.*

Proof By definition, the state of a simple interrupt program $p = (m, \bar{h})$ is of the form $\langle \bar{h}, 0, imr, \sigma, a \rangle$, in addition, the stack size of p increases whenever an interrupt is handled, and we have state transition of the form

$$\langle \bar{h}, 0, imr, \sigma, a \rangle \rightarrow \langle \bar{h}, 0, imr \wedge \neg t_0, a :: \sigma, \bar{h}(i) \rangle \text{ if } imr \geq t_i \vee t_0.$$

Let a_i , $i \in \{1..4\}$ represent the four statements in the body of an interrupt handler such that any handler is of the form $a_1; a_2; a_3; a_4$. By definition of the simple interrupt program, the master bit is enabled only between a_2 and a_3 , where calls to other handlers may occur. In addition, after a call to a interrupt handler returns, the imr value is always less than or equal to the imr value before the call. So, we know that, during a call to handler h_i with initial imr value equal to imr , the only possible states where interrupts maybe be handled are of the form

$$\langle \bar{h}, 0, imr', \sigma, a_3; a_4 \rangle, \text{ where } imr' \leq f_i(imr).$$

Let \rightarrow^* be the reflexive, transitive closure of \rightarrow . Equipped with this notation, we only need to examine state transitions of the following form to compute $\text{maxStackSize}(P_p)$:

$$\langle \bar{h}, 0, imr, \sigma, a \rangle \rightarrow^* \langle \bar{h}, 0, imr', a :: \sigma, a_3; a_4 \rangle,$$

where $imr' \leq f_i(imr) \forall i$, such that $t_i \vee t_0 \leq imr$.

Let $P = \langle \bar{h}, 0, imr, \sigma, a \rangle$ and $P' = \langle \bar{h}, 0, imr', \sigma, a \rangle$. We can show by induction that $\text{maxStackSize}(P) \leq \text{maxStackSize}(P')$ if $imr \leq imr'$.

Therefore, it is sufficient to consider state transitions of the form

$$\langle \bar{h}, 0, imr, \sigma, a \rangle \rightarrow^* \langle \bar{h}, 0, f_i(imr), a :: \sigma, a_3; a_4 \rangle, \text{ where } imr \geq t_i \vee t_0.$$

In the main loop, the possible states where interrupts may be handled are of the form

$$\langle \bar{h}, 0, imr_S \vee t_0, \text{nil}, \text{loop skip} \rangle$$

$$\langle \bar{h}, 0, imr_S \vee t_0, \text{nil}, \text{skip}; \text{loop skip} \rangle.$$

Let a_0 be the statements of the form **loop skip** or **skip;loop skip**.

To compute $\text{maxStackSize}(P_p)$, we only need to consider transitions of the form

$$\langle \bar{h}, 0, imr_S \vee t_0, \sigma, a_0 \rangle \rightarrow^* \langle \bar{h}, 0, f_i(imr_S) \vee t_0, a_0 :: \sigma, a_3; a_4 \rangle,$$

where $imr_S \geq t_i$, and

$$\langle \bar{h}, 0, imr, \sigma, a_3; a_4 \rangle \rightarrow^* \langle \bar{h}, 0, f_j(imr), a_3; a_4 :: \sigma, a'_3; a'_4 \rangle,$$

where $imr \geq t_j \vee t_0$.

It should now be clear that it is sufficient to use $imr \wedge \neg t_0$ to represent states that we are interested in with starting states represented by imr_S . These two kinds of transitions described above can be uniquely represented by edges of the form $(imr_S, f_i(imr_S), i), (imr \wedge \neg t_0, f_j(imr \wedge \neg t_0), j)$ in graph $G(p)$.

Therefore, $\text{maxStackSize}(P_p)$ is equal to the length of the longest path in $G(p)$ from the start node imr_S . ■

Lemma 2.3.5 *For a simple interrupt calculus program p , and a subgraph of $G(p)$, we have that if $imr \leq imr_1 \vee imr_2$, then $|\mathcal{M}(imr)| \leq |\mathcal{M}(imr_1)| + |\mathcal{M}(imr_2)|$.*

Proof We first prove the following claim.

Claim 1: If $imr \leq imr_1 \vee imr_2$, and P is a path from node imr to imr' , then we can find a path P_1 from imr_1 to imr'_1 and a path P_2 from node imr_2 to imr'_2 such that $|P| = |P_1| + |P_2|$ and $imr' \leq imr'_1 \vee imr'_2$.

Proof of Claim 1. We proceed by induction on the length of P . The base case of $|P| = 0$ is trivially true. Suppose Claim 1 is true for $|P| = k$ and that P' is P appended with an edge to imr'' . We need to prove the case of P' . Since P ends at imr' , $\exists t_i \leq imr'$ such that $imr'' = f_i(imr')$. By the induction hypothesis, $t_i \leq imr' \leq imr'_1 \vee imr'_2$. Thus, $\exists a \in \{1, 2\}$ such that $t_i \leq imr'_a$. Suppose that $t_i \leq imr'_1$ (the case of $t_i \leq imr'_2$ is similar and is omitted). We can let P'_1 be P_1 appended with an edge to imr''_1 where $imr''_1 = f_i(imr'_1)$. By definition of f_i , we have $f_i(imr') \leq f_i(imr'_1) \vee imr'_2$. Thus, we have $|P'| = |P| + 1 = |P_1| + 1 + |P_2| = |P'_1| + |P_2|$ and $imr'' \leq imr''_1 \vee imr'_2$.

We can apply Claim 1 to the situation with $\mathcal{M}(imr)$ as the path P from imr to 0. Since $|\mathcal{M}(imr_1)| \geq |P_1|$ and $|\mathcal{M}(imr_2)| \geq |P_2|$, we have $|\mathcal{M}(imr)| \leq |\mathcal{M}(imr_1)| + |\mathcal{M}(imr_2)|$. ■

co-NP-hardness for simple interrupt calculus

We begin by first showing that the Maximum Stack Size problem is co-NP-hard. Later, we show for the simple interrupt calculus problems, then problem is PSPACE-hard. Lastly, we show that the Maximum Stack Size problem for monotonic interrupt calculus programs is PSPACE-hard.

Lemma 2.3.6 *The Maximum Stack Size problem is co-NP-hard.*

Proof We will do a reduction from the SAT problem. Suppose we are given a SAT instance ϕ with L boolean clauses and n boolean variables x_1, x_2, \dots, x_n , where ϕ_{ij} is the j th literal of the i th clause of ϕ . We construct a simple interrupt program

$p = (m, \bar{h})$ as follows. Let $\bar{h} = \{ h_{ij} \mid \phi_{ij} \text{ is a clause of } \phi \}$, $h_{ij} = \mathcal{H}((\bigvee_j t_{ij}) \vee (\bigvee_{\phi_{i'j'} = \neg \phi_{ij}} t_{i'j'}); 0)$, and $imr_0 = \bigvee_{i,j} t_{ij}$.

Recall that if $h_{ij} = \mathcal{H}(imr'; imr'')$, then $f_{ij}(x) = x \wedge imr' \vee imr''$. We use e_{ij} to represent edges of the form $(imr, f_{ij}(imr))$ in $G(p)$.

Claim 1: $|\mathcal{M}(imr_0)| \leq L$.

Proof of Claim 1. By definition, every edge in $G(p)$ is of the form $(imr, f_{ij}(imr))$, where $t_{ij} \leq imr$. By definition, $f_{ij}(imr) = imr \wedge (\neg \bigvee_j t_{ij}) \wedge (\neg \bigvee_{\phi_{i'j'} = \neg \phi_{ij}} t_{i'j'})$. Thus, $\mathcal{M}(imr_0)$ contains at most one edge per clause in ϕ and hence $|\mathcal{M}(imr_0)| \leq L$.

Claim 2: ϕ is satisfiable if and only if $|\mathcal{M}(imr_0)| = L$.

Proof of Claim 2. Suppose that T is a satisfiable truth assignment to ϕ . For each i , we let $\mathcal{M}(imr_0)$ contain an edge e_{ij} for the first j such that $T(\phi_{ij}) = true$. This composition of $\mathcal{M}(imr_0)$ is possible because if e_{ij} is in $\mathcal{M}(imr_0)$, then any edge $e_{i'j'}$ with $\phi_{i'j'} = \neg \phi_{ij}$ is not in $\mathcal{M}(imr_0)$, since $T(\phi_{i'j'}) = false$. Since, for each i , there is one edge e_{ij} in $\mathcal{M}(imr_0)$, we have $|\mathcal{M}(imr_0)| = L$.

Conversely, if $|\mathcal{M}(imr_0)| = L$, then for each i , there exists an edge e_{ij} in $\mathcal{M}(imr_0)$. Let $T(\phi_{ij}) = true$ iff $\mathcal{M}(imr_0)$ contains edge e_{ij} . T is a truth assignment to ϕ because $T(\phi_{ij}) = true$ iff $T(\phi_{i'j'}) = false$ for all $\phi_{i'j'} = \neg \phi_{ij}$. Since for each i , at least one $T(\phi_{ij}) = true$, truth assignment T satisfies ϕ .

We conclude

$$\begin{aligned} & \phi \text{ is not satisfiable} \\ & \text{iff } |\mathcal{M}(imr_0)| \leq L - 1 \quad (\text{Claims 1+2}) \\ & \text{iff } \mathit{maxStackSize}(P_p) \leq L - 1 \quad (\text{Theorem 2.3.4}), \end{aligned}$$

so the Maximum Stack Size problem is co-NP-hard. ■

PSPACE-hardness for simple interrupt calculus

We now show PSPACE-hardness for simple interrupt calculus. Our proof is based on a polynomial-time reduction from the *quantified boolean satisfiability* (QSAT) problem [46].

We first illustrate our reduction by a small example. Suppose we are given a QSAT instance $S = \exists x_2 \forall x_1 \phi$ with $\phi = (l_{11} \vee l_{12}) \wedge (l_{21} \vee l_{22}) = (x_2 \vee \neg x_1) \wedge (x_2 \vee x_1)$,

We construct a simple interrupt program $p = (m, \bar{h})$ with an imr register, where $\bar{h} = \{h(x_i), h(\bar{x}_i), h(w_i), h(\bar{w}_i), h(l_{ij}) \mid i, j = 1, 2\}$ are 12 handlers. The imr contains 13 bits: a master bit, and each remaining bit maps one-to-one to each handler in \bar{h} . Let $\mathcal{D} = \{x_i, \bar{x}_i, w_i, \bar{w}_i, l_{ij} \mid i, j = 1, 2\}$. We use t_x , where $x \in \mathcal{D}$, to denote the imr value where all bits are 0's except for the bit corresponding to handler $h(x)$, which is set to 1. The initial imr value imr_S is set to $imr_S = t_{x_2} \vee t_{\bar{x}_2}$.

We now construct \bar{h} . Let $E(h(x))$, $x \in \mathcal{D}$, be the set of handlers that $h(x)$ enables. This *enable* relation between the handlers of our example is illustrated in Figure 2.4, where there is an edge from $h(x_i)$ to $h(x_j)$ iff $h(x_i)$ enables $h(x_j)$. Let $D(h(x))$, $x \in \mathcal{D}$, be the set of handlers that $h(x)$ disables. Let $L = \{h(l_{ij}) \mid i, j = 1, 2\}$. The $D(h(x)), x \in \mathcal{D}$, are defined as follows:

$$D(h(x_2)) = D(h(\bar{x}_2)) = \{h(x_2), h(\bar{x}_2)\} \quad (2.28)$$

$$D(h(x_1)) = \{h(x_1)\} \quad D(h(\bar{x}_1)) = \{h(\bar{x}_1)\} \quad (2.29)$$

$$D(h(w_2)) = D(h(\bar{w}_2)) = \{h(x_1), h(\bar{x}_1)\} \cup \{h(w_i), h(\bar{w}_i) \mid i = 1, 2\} \cup L \quad (2.30)$$

$$D(h(w_1)) = D(h(\bar{w}_1)) = \{h(w_1), h(\bar{w}_1)\} \cup L \quad (2.31)$$

$$D(h(l_{ij})) = \{h(l_{i1}), h(l_{i2})\} \cup \{h(w_k) \mid l_{ij} = \neg x_k\} \cup \{h(\bar{w}_k) \mid l_{ij} = x_k\}. \quad (2.32)$$

If $h(x) = \mathcal{H}(imr'; imr'')$, then $imr' = \bigvee_{h(y) \in E(h(x))} t_y$ and $imr'' = \bigvee_{h(z) \in D(h(x))} t_z$, where $x, y, z \in \mathcal{D}$.

We claim that the QSAT instance S is satisfiable iff $|\mathcal{M}(imr_S)| = 10$, where $imr_S = t_{x_2} \vee t_{\bar{x}_2}$. We sketch the proof as follows.

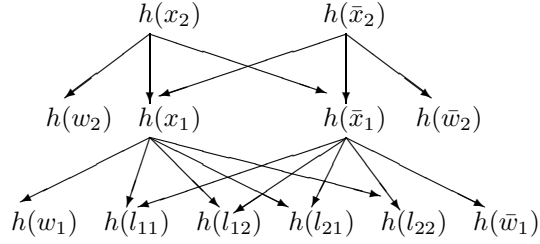


Figure 2.4. Enable relation of interrupt handlers

Let $imr_L = \bigvee_{h(l) \in L} t_l$, where $l \in \mathcal{D}$. From (2.32) and Figure 2.4, it can be shown that $|\mathcal{M}(imr_L)| = 2$. From Figure 2.4, we have $E(h(x_1)) = \{h(w_1)\} \cup L$; and together with (2.31), and (2.32), it can be shown that

$$|\mathcal{M}(t_{x_1})| = 1 + |\mathcal{M}(t_{w_1} \vee imr_L)| \leq 2 + |\mathcal{M}(imr_L)| = 4$$

and this equality holds iff $\exists j_1, j_2 \in 1, 2$, such that $l_{1j_1}, l_{2j_2} \neq \neg x_1$, because otherwise handler $h(w_1)$ would certainly be disabled. Similarly, it can be shown that $|\mathcal{M}(t_{\bar{x}_1})| \leq 4$, and that

$$|\mathcal{M}(t_{x_1} \vee t_{\bar{x}_1})| \leq |\mathcal{M}(t_{x_1})| + |\mathcal{M}(t_{\bar{x}_1})| \leq 8,$$

where the equality holds iff $\exists j_1, j_2$, such that $l_{1j_1}, l_{2j_2} \neq \neg x_1$ and $\exists j'_1, j'_2$, such that $l_{1j'_1}, l_{2j'_2} \neq x_1$. From Figure 2.4, we have $E(h(x_2)) = \{h(w_2), h(x_1), h(\bar{x}_1)\}$. Thus,

$$|\mathcal{M}(t_{x_2})| = 1 + |\mathcal{M}(t_{w_2} \vee t_{x_1} \vee t_{\bar{x}_1})| \leq 2 + |\mathcal{M}(t_{x_1} \vee t_{\bar{x}_1})| = 10,$$

and it can be shown from (2.30) and (2.32) that the equality holds iff $\exists j_1, j_2$ such that $l_{ij_1}, l_{ij_2} \neq \neg x_2, \neg x_1$ and $\exists j'_1, j'_2$ such that $l_{ij'_1}, l_{ij'_2} \neq \neg x_2, x_1$, which implies that both $x_2 = \text{true}, x_1 = \text{true}$ and $x_2 = \text{true}, x_1 = \text{false}$ are satisfiable truth assignments to ϕ . Similarly, it can be shown that $|\mathcal{M}(t_{\bar{x}_2})| = 10$ iff both $x_2 = \text{false}, x_1 = \text{true}$ and $x_2 = \text{false}, x_1 = \text{false}$ are satisfiable truth assignments to ϕ .

From (2.28), we have $|\mathcal{M}(t_{x_2} \vee t_{\bar{x}_2})| = \max(|\mathcal{M}(t_{x_2})|, |\mathcal{M}(t_{\bar{x}_2})|)$. Therefore, $|\mathcal{M}(imr_S)| = 10$ iff there exists x_2 such that for all x_1 , ϕ is satisfiable, or equivalently iff S is satisfiable. For our example, S is satisfiable since $\exists x_2 = \text{true}$ such that $\forall x_1$, ϕ is satisfiable. Correspondingly, $|\mathcal{M}(imr_S)| = |\mathcal{M}(x_2)| = 10$.

Theorem 2.3.5 *The exact maximum stack size problem for monotonic interrupt programs is PSPACE-hard.*

Proof We will do a reduction from the QSAT problem. Suppose we are given an instance of QSAT problem

$$S = \exists x_n \forall x_{n-1} \dots \exists x_2 \forall x_1 \phi,$$

where ϕ is a 3SAT instance in conjunctive normal form of n variables x_n, \dots, x_1 and L boolean clauses. Let ϕ_{ij} be the j th literal of the i th clause in ϕ and $\phi = \bigwedge_{i=1}^L \bigvee_{j=1}^3 \phi_{ij}$.

We construct a program $p = (m, \bar{h})$ and $\bar{h} = \{h(i) \mid i \in \{1 \dots 3L + 4n\}\}$.

As we did earlier, we define a graph $G(p) = (V, E)$ such that $V = \{imr \mid imr(0) = 0\}$ and $E = \{(imr, f_i(imr), i) \mid t_i \leq imr\}$, where $f_i(imr) = imr \wedge \neg imr' \vee imr''$ iff $h(i) = \mathcal{H}(imr'; imr'')$.

For the sake of clarity, we define three kinds of indices: $d_{ij} = 3(i-1) + j$, where $i \in \{1..L\}, j \in \{1..3\}$; $q_i^a = 3L + 4i - 3 + a$, and $w_i^a = 3L + 4i - 1 + a$, where $i \in \{1..n\}, a \in \{0, 1\}$.

Let

$$\begin{aligned} D &= \{d_{i1}, d_{i2}, d_{i3} \mid \forall i \in \{1..L\}\} \\ D_{ij} &= \{d_{i1}, d_{i2}, d_{i3}\} \cup \{w_k^a \mid (a = 1 \wedge \phi_{ij} = x_k) \vee (a = 0 \wedge \phi_{ij} = \neg x_k)\} \\ W_i &= \{w_j^a \mid \forall j \in \{1..i\}, \forall a \in \{0, 1\}\} \\ Q_i &= \{q_j^a \mid \forall j \in \{1..i\}, \forall a \in \{0, 1\}\}. \end{aligned}$$

We will use the abbreviation

$$imr_0 = \bigvee_{i \in D} t_i, \quad imr_k = t_{q_k^0} \vee t_{q_k^1} \quad \forall k \in \{1..n\}.$$

Assume that n is even. For all $a \in \{0, 1\}$, let

$$\begin{aligned} f_{q_{2k-1}^a}(x) &= x \wedge \neg t_{q_{2k-1}^a} \vee (imr_{2k-2} \vee t_{w_{2k-1}^a}), \quad \forall k \in \{1..n/2\} \\ f_{q_{2k}^a}(x) &= x \wedge \neg imr_{2k} \vee (imr_{2k-1} \vee t_{w_{2k}^a}), \quad \forall k \in \{1..n/2\} \end{aligned}$$

$$\begin{aligned}
f_{w_k^a}(x) &= x \wedge \neg \bigvee_{i \in DUQ_{k-1} \cup W_k} t_i, & \forall k \in \{1..n\} \\
f_{d_{ij}}(x) &= x \wedge \neg \bigvee_{k \in D_{ij}} t_k, & \forall i \in \{1..L\}, j \in \{1, 2, 3\}.
\end{aligned}$$

Given an imr value r , we define the graph $G_r(p)$ to be the subgraph of $G(p)$ such that any edge labeled d_{ij} is removed for all i, j such that $\phi_{ij} = \neg x_k$ and $t_{w_k^0} \leq r$, or $\phi_{ij} = x_k$ and $t_{w_k^1} \leq r$. We use $\mathcal{M}_r(imr)$ to denote the longest path in $G_r(p)$ from imr .

Claim 1: $\forall k \in \{1..\frac{n}{2}\}, |\mathcal{M}_r(imr_{2k})| = \max_{a \in \{0,1\}} |\mathcal{M}_r(t_{q_{2k}^a})|$, and $|\mathcal{M}_r(imr_0)| \leq L$.

Proof of Claim 1. By definition, we have that $\forall a \in \{0, 1\}, f_{q_{2k}^a}(x) = x \wedge \neg imr_{2k} \vee (imr_{2k-1} \vee t_{w_{2k}^a})$, from which the claim follows.

By definition of $f_{d_{ij}}$, for each $i \in \{1..L\}$, $\mathcal{M}(imr_0)$ can contain at most one edge with label d_{ij} , where $j \in \{1, 2, 3\}$. Thus, $|\mathcal{M}(imr_0)| \leq L$.

Claim 2: $|\mathcal{M}_r(imr_{2k-1})| = \sum_{b \in \{0,1\}} |\mathcal{M}_r(t_{q_{2k-1}^b})|$.

Proof of Claim 2. From Lemma 2.3.5, we have $|\mathcal{M}_r(imr_{2k-1})| \leq \sum_{b \in \{0,1\}} |\mathcal{M}_r(t_{q_{2k-1}^b})|$.

Let P be the path from imr_{2k-1} to $t_{q_{2k-1}^1}$ constructed from $\mathcal{M}_r(t_{q_{2k-1}^0})$ by replacing any node imr on $\mathcal{M}_r(t_{q_{2k-1}^0})$ with $imr \vee t_{q_{2k-1}^1}$. It is straightforward to show that if edge (imr, imr', i) is on $\mathcal{M}_r(t_{q_{2k-1}^0})$, then $(imr \vee t_{q_{2k-1}^1}, imr' \vee t_{q_{2k-1}^1}, i)$ is on P . If we concatenate P with $\mathcal{M}_r(t_{q_{2k-1}^1})$, then we have a path from imr_{2k-1} of length $|\mathcal{M}_r(t_{q_{2k-1}^0})| + |\mathcal{M}_r(t_{q_{2k-1}^1})|$.

Claim 3: $|\mathcal{M}(imr_n)| \leq 2^{n/2}(6 + L) - 6$.

Proof of Claim 3. It is sufficient to prove that $|\mathcal{M}(imr_{2k})| \leq 2^k(6 + L) - 6$. For all $a \in \{0, 1\}$ we have:

$$\begin{aligned}
|\mathcal{M}(t_{q_{2k}^a})| &= 1 + |\mathcal{M}(imr_{2k-1} \vee t_{w_{2k}^a})| \\
&\leq 2 + |\mathcal{M}(imr_{2k-1})| \\
&= 2 + \sum_{b \in \{0,1\}} |\mathcal{M}(t_{q_{2k-1}^b})|
\end{aligned}$$

$$\begin{aligned}
&= 4 + \sum_{b \in \{0,1\}} |\mathcal{M}(imr_{2k-2} \vee t_{w_{2k-1}^b})| \\
&\leq 6 + 2|\mathcal{M}(imr_{2k-2})|
\end{aligned}$$

$$\begin{aligned}
|\mathcal{M}(imr_{2k})| &= \max_{a \in \{0,1\}} (|\mathcal{M}(t_{q_{2k}^0})|, |\mathcal{M}(t_{q_{2k}^1})|) \\
&\leq 6 + 2|\mathcal{M}(imr_{2k-2})|
\end{aligned}$$

From the last inequality and Claim 1, it is straightforward to show the claim by induction on k .

Claim 4: For any r and $a \in \{0,1\}$, $|\mathcal{M}_r(t_{q_{2k}^a})| = 2^k(6+L) - 6$ iff $\forall b \in \{0,1\}$, $|\mathcal{M}_{r'}(t_{q_{2k-1}^b})| = 2^{k-1}(6+L) - 4$, where $r' = r \vee t_{w_{2k}^a}$.

Proof of Claim 4. Suppose that $|\mathcal{M}_r(t_{q_{2k}^a})| = 2^k(6+L) - 6$. The path $\mathcal{M}_r(t_{q_{2k}^a})$ must contain the edge with label w_{2k}^a because otherwise

$$\begin{aligned}
|\mathcal{M}_r(t_{q_{2k}^a})| &= 1 + |\mathcal{M}_r(imr_{2k-1} \vee t_{w_{2k}^a})| \\
&= 1 + |\mathcal{M}_r(imr_{2k-1})| \\
&\leq 1 + |\mathcal{M}(imr_{2k-1})| \leq 2^k(6+L) - 7.
\end{aligned}$$

By definition of $f_{q_{2k}^a}$, for any node imr on the path $\mathcal{M}_r(imr_{2k-1} \vee t_{q_{2k}^a})$, we have $f_{w_{2k}^a}(imr) = 0$. Thus, the edge labeled w_{2k}^a can only be the last edge on $\mathcal{M}_r(t_{q_{2k}^a})$. By definition of $f_{d_{ij}}$, the longest path from $imr_{2k-1} \vee t_{w_{2k}^a}$ containing edge labeled w_{2k}^a does not contain any edge labeled d_{ij} for all i, j such that $\phi_{ij} = x_{2k}$ if $a = 1$, and $\phi_{ij} = \neg x_{2k}$ if $a = 0$. This path is the same path in $G_{r'}(p)$, where $r' = r \vee t_{w_{2k}^a}$. Therefore,

$$\begin{aligned}
|\mathcal{M}_r(t_{q_{2k}^a})| = 2^k(6+L) - 6 &= |\mathcal{M}_{r'}(t_{q_{2k}^a})| \\
&= 1 + |\mathcal{M}_{r'}(imr_{2k-1} \vee t_{w_{2k}^a})| \\
&\leq 1 + |\mathcal{M}_{r'}(imr_{2k-1})| + |\mathcal{M}_{r'}(t_{w_{2k}^a})| \\
&= 2 + \sum_{b \in \{0,1\}} |\mathcal{M}_{r'}(t_{q_{2k-1}^b})|, \text{ and} \\
\sum_{b \in \{0,1\}} |\mathcal{M}_{r'}(t_{q_{2k-1}^b})| &\geq 2^k(6+L) - 8.
\end{aligned}$$

Since

$$\begin{aligned}
|\mathcal{M}_{r'}(t_{q_{2^{k-1}}}^b)| &= 1 + |\mathcal{M}_{r'}(imr_{2^{k-2}} \vee t_{w_{2^{k-1}}}^b)| \\
&\leq 2 + |\mathcal{M}_{r'}(imr_{2^{k-2}})| \\
&\leq 2 + |\mathcal{M}(imr_{2^{k-2}})| = 2^{k-1}(6 + L) - 4,
\end{aligned}$$

we have $\forall b \in \{0, 1\}$, $|\mathcal{M}_{r'}(t_{q_{2^{k-1}}}^b)| = 2^{k-1}(6 + L) - 4$.

Conversely, assume that for all $b \in \{0, 1\}$, $|\mathcal{M}_{r'}(t_{q_{2^{k-1}}}^b)| = 2^{k-1}(6 + L) - 4$ where $r' = r \vee t_{w_{2^k}}^a$. From Claim 2, we know that $|\mathcal{M}_{r'}(imr_{2^{k-1}})| = \sum_{b \in \{0, 1\}} |\mathcal{M}_{r'}(t_{q_{2^{k-1}}}^b)| = 2^k(6 + L) - 8$.

Let P be a path from $imr_{2^{k-1}} \vee t_{w_{2^k}}^a$ to $t_{w_{2^k}}^a$ constructed from $\mathcal{M}_{r'}(imr_{2^{k-1}})$ by replacing any node imr on $\mathcal{M}_{r'}(imr_{2^{k-1}})$ with $imr \vee t_{w_{2^k}}^a$. It is straightforward to show that if edge (imr, imr', i) is on $\mathcal{M}_{r'}(imr_{2^{k-1}})$, then $(imr \vee t_{w_{2^k}}^a, imr' \vee t_{w_{2^k}}^a, i)$ is on P as well.

If we concatenate P with $\mathcal{M}_{r'}(t_{w_{2^k}}^a)$, then we have a path from $imr_{2^{k-1}} \vee t_{w_{2^k}}^a$ in graph $G_{r'}(p)$ of length $2^k(6 + L) - 7$. Thus, $|\mathcal{M}_r(t_{q_{2^k}}^a)| = |\mathcal{M}_{r'}(t_{q_{2^k}}^a)| = 2^k(6 + L) - 6$.

Claim 5: for any r' and $b \in \{0, 1\}$, $|\mathcal{M}_{r'}(t_{q_{2^{k-1}}}^b)| = 2^{k-1}(6 + L) - 4$ iff $|\mathcal{M}_{r''}(imr_{2^{k-2}})| = 2^{k-1}(6 + L) - 6$, where $r'' = r' \vee t_{w_{2^{k-1}}}^b$,

Proof of Claim 5. The proof is similar to the proof of Claim 4, we omit the details.

Claim 6: $|\mathcal{M}(imr_n)| = 2^{n/2}(6 + L) - 6$ iff $\exists a_n \forall a_{n-1} \dots \exists a_2 \forall a_1 \in \{0, 1\}$, such that for $r = \bigvee_{k \in \{1..n\}} t_{w_k}^{a_k}$, $|\mathcal{M}_r(imr_0)| = L$.

Proof of Claim 6. From Claims 1+3, we know that $|\mathcal{M}_r(imr_{2^k})| = 2^k(6 + L) - 6$ iff $\exists a \in \{0, 1\}$ such that $|\mathcal{M}_r(t_{q_{2^k}}^a)| = 2^k(6 + L) - 6$. Together with Claim 4 and Claim 5, we have that for $k \in \{1..n/2\}$,

$$\begin{aligned}
|\mathcal{M}_r(imr_{2^k})| &= 2^k(6 + L) - 6, \text{ iff } \exists a \in \{0, 1\}, \forall b \in \{0, 1\}, \text{ such that} \\
|\mathcal{M}_{r''}(imr_{2^{k-2}})| &= 2^{k-1}(6 + L) - 6, \text{ where } r'' = r \vee t_{w_{2^k}}^a \vee t_{w_{2^{k-1}}}^b.
\end{aligned}$$

It is straightforward to prove by induction from $k = n/2$ to 1, that

$$\begin{aligned}
|\mathcal{M}(imr_n)| &= 2^{n/2}(6 + L) - 6 \text{ iff } \exists a_n \forall a_{n-1} \dots \exists a_{2^k} \forall a_{2^{k-1}}, \\
\text{such that } |\mathcal{M}_r(imr_{2^{k-2}})| &= 2^{k-1}(6 + L) - 6, \text{ where } r = \bigvee_{i=2^{k-1}}^n t_{w_i}^{a_i}.
\end{aligned}$$

This claim follows when $k = 1$.

Claim 7: S is satisfiable iff $\exists a_n \forall a_{n-1} \dots \exists a_2 \forall a_1 \in \{0, 1\}$, we have $|\mathcal{M}_r(\text{imr}_0)| = L$, where $r = \bigvee_{k=1}^n t_{w_k^{a_k}}$.

Proof of Claim 7. It is sufficient to prove that ϕ is satisfiable iff $\exists a_n, \dots, a_1 \in \{0, 1\}$, such that for $r = \bigvee_{k=1}^n t_{w_k^{a_k}}$, we have $|\mathcal{M}_r(\text{imr}_0)| = L$.

Suppose we have a_n, \dots, a_1 such that $r = \bigvee_{k=1}^n t_{w_k^{a_k}}$, $|\mathcal{M}_r(\text{imr}_0)| = L$. We can construct a truth assignment T by defining $T(x_k) = \text{true}$ if $a_k = 0$ and $T(x_k) = \text{false}$ if $a_k = 1$. By definition of $f_{d_{ij}}$, for each $i \in \{1..L\}$, there exists a j such that the edge labeled d_{ij} is on $\mathcal{M}_r(\text{imr}_0)$. By definition of \mathcal{M}_r , if an edge labeled d_{ij} is on $\mathcal{M}_r(\text{imr}_0)$ and $\phi_{ij} = x_k$, then $a_k = 0$, and $T(x_k) = \text{true}$; and if $\phi_{ij} = \neg x_k$, then $a_k = 1$ and $T(x_k) = \text{false}$. $T(\phi_{ij}) = \text{true}$ in both cases. Therefore, T satisfies ϕ .

Conversely, suppose T satisfies ϕ . We can construct $r = \bigvee_{k=1}^n t_{w_k^{a_k}}$ from T by defining $a_k = 0$ if $T(x_k) = \text{true}$ and $a_k = 1$ if $T(x_k) = \text{false}$. For each $i \in \{1..L\}$, there exists j such that $T(\phi_{ij}) = \text{true}$, which means that the edge labeled d_{ij} can be on the path $\mathcal{M}_r(\text{imr}_0)$. Therefore, $|\mathcal{M}_r(\text{imr}_0)| = L$.

We conclude that

S is not satisfiable

iff we do not have $\exists a_n \forall a_{n-1} \dots \exists a_2 \forall a_1$,

such that for $r = \bigvee_{k=1}^n t_{w_k^{a_k}}$, $|\mathcal{M}_r(\text{imr}_0)| = L$ (Claim 7)

iff $|\mathcal{M}(\text{imr}_S)| \neq 2^{n/2}(6 + L) - 7$, where $\text{imr}_S = \text{imr}_n$ (Claims 3+6)

iff $\text{maxStackSize}(P_p) \neq 2^{n/2}(6 + L) - 7$ (Lemma 2.3.4),

so the exact maximum stack size problem is co-PSPACE-hard, that is, PSPACE-hard. ■

2.3.8 Stack-size checking is PSPACE-hard

Theorem 2.3.6 *For simple interrupt calculus programs, Maximum Stack-Size is polynomial-time reducible to Stack-Size Checking.*

Proof It is sufficient to prove that, given a simple interrupt calculus program $p = (m, \bar{h})$ and a natural number K , we have $\text{maxStackSize}(P_p) \leq K$ if and only if there exists $\bar{\tau}$ such that $\bar{\tau} \vdash_K p$ is derivable.

Suppose first that $\text{maxStackSize}(P_p) \leq K$. From Theorem 2.3.4 and $\text{maxStackSize}(P_p) \leq K$, we have $|\mathcal{M}(\text{imr}_0)| \leq K$.

For each node $\widehat{\text{imr}}$ in $G(p)$, if $t_i \vee t_0 \leq \widehat{\text{imr}}$, then let $\bar{\tau}(i) = \dots \wedge \widehat{\text{imr}} \xrightarrow{\delta} \widehat{\text{imr}} \wedge \dots$, where $\delta = |\mathcal{M}(f_i(\widehat{\text{imr}}))|$.

We need to show that $\bar{\tau} \vdash_K (m, \bar{h})$ is derivable; that is, we need to show that $\bar{\tau} \vdash \bar{h} : \bar{\tau}$ and $\bar{\tau}, 0 \vdash_K m$ are derivable.

1. $\bar{\tau} \vdash \bar{h} : \bar{\tau}$.

We need to show that $\bar{\tau} \vdash h_i : \bar{\tau}(i)$ for all i . Suppose that $\bar{\tau}(i) = \bigwedge_{j=1}^q (\widehat{\text{imr}}^j \xrightarrow{\delta^j} \widehat{\text{imr}}^j)$. It is sufficient to show that $\forall j \in \{1..q\}$, we have $\bar{\tau}, \widehat{\text{imr}}^j \wedge \neg t_0 \vdash_{\delta^j} h_i : \widehat{\text{imr}}^j$.

Suppose that the body of h_i is

$$\begin{aligned} \text{imr} &= \text{imr} \wedge \text{imr}' ; \\ \text{imr} &= \text{imr} \vee (t_0 \vee \text{imr}'') \\ \text{imr} &= \text{imr} \wedge \neg(t_0 \vee \text{imr}'') \\ \text{iret}. \end{aligned}$$

Since the master bit is disabled everywhere except between the second and third lines, we only need to show that

$$\begin{aligned} \bar{\tau}, \widehat{\text{imr}}^j \wedge \text{imr}' \vee \text{imr}'' \vdash_{\delta^j} \text{imr} = \text{imr} \wedge \neg(t_0 \vee \text{imr}'') : \widehat{\text{imr}}^j \wedge \neg t_0 \wedge \text{imr}', \\ \bar{\tau}, \widehat{\text{imr}}^j \wedge \text{imr}' \wedge \neg t_0 \vdash_{\delta^j} \text{iret} : \widehat{\text{imr}}^j, \text{ and } (\widehat{\text{imr}}^j \wedge \neg t_0 \wedge \text{imr}') \vee t_0 \leq \widehat{\text{imr}}^j, \end{aligned}$$

which is trivially true, and $[\text{safe}(\bar{\tau}, \widehat{\text{imr}}, \delta^j)]$, where $\widehat{\text{imr}} = \widehat{\text{imr}}^j \wedge \text{imr}' \vee \text{imr}''$. By definition $[\text{safe}(\bar{\tau}, \widehat{\text{imr}}, \delta^j)]$ is true iff $\forall k \in \{1..q\}$, if $t_k \vee t_0 \leq \widehat{\text{imr}}$, then whenever $\bar{\tau}(k) = \dots \wedge (\widehat{\text{imr}} \xrightarrow{\delta} \widehat{\text{imr}}) \wedge \dots$, we have $\delta + 1 \leq \delta^j$.

From construction of $\bar{\tau}$, we know that $\delta = |\mathcal{M}(f_k(\widehat{\text{imr}}))|$. Since $\widehat{\text{imr}} = \widehat{\text{imr}}^j \wedge \text{imr}' \vee \text{imr}'' = f_i(\widehat{\text{imr}}^j)$, we have that $\delta^j = |\mathcal{M}(\widehat{\text{imr}}^j)|$. Also since $(\widehat{\text{imr}}, f_k(\widehat{\text{imr}})) \in E$, we know that $\delta^j \geq \delta + 1$.

2. $\bar{\tau}, 0 \vdash_K m$.

The body of the main loop is

$$\text{imr} = \text{imr} \vee \text{imr}_0;$$

$$\text{loop skip};$$

Thus, it is sufficient to prove that $\bar{\tau}, \text{imr}_0 \vdash_K \text{skip} : \text{imr}_0$ and $[\text{safe}(\bar{\tau}, \text{imr}_0, K)]$. By definition, imr_0 is the start node of $G(p)$; thus, $K = |\mathcal{M}(\text{imr}_0)|$. To prove $[\text{safe}(\bar{\tau}, \text{imr}_0, K)]$, notice that if $\bar{\tau}(k) = \dots \wedge (\text{imr}_0 \xrightarrow{\delta} \text{imr}_0) \wedge \dots$, then $\delta = |\mathcal{M}(f_k(\text{imr}_0))|$. Since $(\text{imr}_0, f_k(\text{imr}_0)) \in E$ and $|\mathcal{M}(\text{imr}_0)| = K$ we have that $K \geq |\mathcal{M}(f(\text{imr}_0))| + 1 = \delta + 1$.

Conversely, suppose that there exists $\bar{\tau}$ such that $\bar{\tau} \vdash_K p$ is derivable. From Theorem 2.2.3 and $\bar{\tau} \vdash_K p$ being derivable, we have that $\text{maxStackSize}(P_p) \leq K$. ■

Theorem 2.3.7 *Stack-Size Checking is PSPACE-hard.*

Proof Combine Theorem 2.3.5 and Theorem 2.3.6. ■

2.4 Related work

There is a large body of research dedicated to the problems of event-driven real-time systems in general. Virtually all of it assumes interrupt handling can be considered instantaneous and fails to directly address interrupt handling and its related stack size boundedness problem (except [8] which is described below), even though their implementations utilize the interrupt as a underlying mechanism. Among the papers written on this topic, the following works are particularly relevant.

2.4.1 Stack checking for interrupt-driven programs

Brylow, Damgaard and Palsberg [8] abstract the interrupt program into a control flow graph; they then model check the size of bounded stack of interrupt driven programs by running a context free reachability algorithm on the graph. Their work

is the most relevant one to ours because both works consider the stack boundedness problem for interrupt-driven programs. In contrast to their work, we take another approach. We abstract the interrupt systems and formally define a formal language (interrupt calculus) that captures the interrupt mechanism. We use type systems (intersection types) to study the stack boundedness problem in this formal environment. We check the boundedness property by running type checking algorithm, rather than model checking.

2.4.2 Sized types and dependent types

Hughes, Pareto, and Sabry [30, 47, 48] use *sized types* to reason about liveness, termination, and space boundedness of reactive systems programmed in functional languages. There are similarities between their work and ours: both use types annotated with explicit space information to reason the space boundedness, both use constraint solving to infer the type annotations (space), and both enable modular, type-based, compile-time type checking on the program to verify that the program runs in bounded space. The sized type can be viewed as a specialized form of dependent types [61–63]. (This extends the ML type systems with a parameterized size.) By contrast, our type system of handlers is a hybrid of dependent types (stack growth δ) and context-sensitive (intersection types), value-sensitive (imr values), flow-sensitive analysis. In addition, as outlined below, there are four differences which separate their work from ours:

1. Even though both their work and ours address space boundedness, their concept of space boundedness is used in the sense that, for example, recursive data structures should grow within a certain bound and functions should process inputs within a constant amount of memory space, while our work is concerned with stack boundedness caused by calling handlers when handling interrupts. Thus, their work does not address interrupt handling, while ours is built upon an interrupt computation model;

2. Their work does not study complexity bounds;
3. Their work shows how to do type checking which relies on a moderate number of type annotations. Their type checking process involves solving a constraint system in order to be able to infer the *size* of space used by programs. By contrast, we not only show how to type check the annotated interrupt programs, we also show how to automatically infer the types and type annotations.
4. We use constraint systems to infer the intersection types and type annotations (stack growth δ), that is, type inference, rather than to type check programs.

2.4.3 Event-driven FRP

Functional reactive programming (FRP) [28] is a high-level declarative language for programming reactive systems. It has been implemented as an embedded language in Haskell at Yale. Wan, Taha, Hudak [58, 59] present two variations of FRP called real-time FRP and event-driven FRP. Both real-time FRP and event-driven FRP are designed such that the time and space behavior of a program are necessarily bounded. Of the two languages, Event-driven FRP is especially germane to the interrupt calculus.

The event-driven FRP is a simplified variant of FRP and is intended for programming interrupt-driven micro-controllers. The event-driven FRP features *events* and *behaviors*. The behaviors can change their states (values) only when an event occurs. No events in Event-driven FRP can occur simultaneously, implying that no handling of the events can overlap. Upon occurrences of events, Event-driven FRP programs first compute the values that depend on the previous state of the computation, and then, after that, update the state of the computation.

However, there is a fundamental difference between the semantics of Event-driven FRP and our interrupt calculus. In Event-driven FRP, during the handling of an event, there is no other event which can occur. This serialization of interrupt events flattens the handling of events. By contrast, our interrupt calculus allows interrupts

to occur which are handled during the execution of a handler. Because of this assumption regarding events, Event-driven FRP programs are not concerned with the problem of stack boundedness caused by interrupting a running handler. However, we believe that allowing handlers to be interrupted can result in more responsive and more fine-tuned (in terms of time) real-time software. For example, in [59], a simple RoboCup controller program is shown in which a timer counter interrupt occurs much more frequently than any other sources of interrupts. If the running duration of other handlers is so long that the processor has to delay, or even drop, the interrupts of the timer counter, then the output of the controller may experience instability, thus causing unpredictable system behavior. This problem is caused by the mismatch between the slow lower-level of interrupt-handling responsiveness and fast interrupt arrivals. Our interrupt calculus achieves more responsive interrupt-handling by allowing long-handled, but infrequent, interrupts to be interrupted by short-handled, but frequent, interrupts. Doing this also enables more fine-tuned real-time programs because short-handled interrupts can interrupt long-handled handling, allowing for a shorter wait time and avoiding interrupt drops of short-handled interrupts.

The programs of Event-driven FRP can be translated to our interrupt calculus. However, interrupt calculus programs cannot be faithfully translated to Event-driven FRP. One example of this would be the *fancy handlers* example (Example 1.6 in Section 1.5).

2.4.4 The Giotto language and the embedded machine

The Giotto language is a time-triggered language for embedded programming introduced by Henzinger, Horowitz and Kirsch [22, 23]. Giotto programs are translated into a language called E code. E code runs on Embedded machine (E machine) [24, 25]. Embedded machine is a virtual machine, independent of platform and schedulers. In E-machine terms, a task is a periodic process that carries out a computation. Tasks can be invoked with arguments (typically drivers) and output its results to task ports.

The computation of tasks is not synchronous. Drivers are communication facilities that may provide sensor readings to tasks, or load tasks results to actuators, or provide task results as arguments to other tasks. E code schedules tasks on E machine such that the “time-safe” property holds. “Time-safe” means that the computation of a task t always finishes before drivers access t ’s output ports. There are two major steps of ensuring the “time-safe” property. (1) during the compilation, the compiler relies on the WCETs (worst case execution time) of tasks, triggers and drivers to compute whether the tasks can finish in time, assuming the underlying scheduling scheme. (2) the e code interpreter will guard the “time-safe” property at runtime. If any assumptions about WCETs prove to be incorrect, then the E machine will generate exceptions at runtime. Relying on information about shared ports between drivers and tasks in the ready queue, the e machine is able to tell whether there has been “time-safe” violation. Determinism of outcomes of tasks comes immediately from “time-safe” property.

The Giotto language and E machine provide one especially desirable feature: they enable real-time processes which produce predictable behavior, independent of platforms. However, Giotto and E machine are not designed to handle interrupts. The programming paradigm that Giotto uses is one in which all tasks are periodic within a *mode* (which is a set of tasks for some time). The interrupt calculus, on the other hand, does not require interrupts to occur periodically. It is clear that stack boundedness caused by interrupts is not an issue in E machine since the E machine does not consider the interrupt scenario.

There would appear to be no satisfactory translation between the Giotto language and interrupt calculus programs.

2.4.5 Esterel

The Esterel language [7] is a specification language for reactive systems. The central idea of the language is that it follows the synchrony hypothesis. The synchrony

hypothesis specifies that each *reaction* is assumed to be an instantaneous, atomic execution. The race between reactions (coming interrupts are handled before the current handler returns in the context of our interrupt calculus) is avoided by adopting such a hypothesis. Hence the stack boundedness problem does not appear in Esterel.

Esterel, on the other hand, is expressive and suitable for reactive system specifications. It has clear behavioral semantics which are based on *signals*, *events* and *histories* which our interrupt calculus does not have. It is for this reason that there appears to be no comprehensive translation between the two languages, with the exception of a few programs whose semantics fall within the intersection of both.

2.5 Future directions

The type system we give here type checks monotonic interrupt calculus programs. Chatterjee et al. [12] enrich the interrupt calculus by allowing conditional imr operations (bit-wise *and* and *or*) that are dependent on the imr bits, and give an exact stack size analysis for non-monotonic interrupt calculus programs which is based on the context-free graph [55, 56]. However, it remains unclear, as well as interesting, what a type system would look like if we would like to be able to accept non-monotonic interrupt calculus programs.

In addition, even though the Stack-Size Checking problem is PSPACE-complete, it remains an open issue whether there are better algorithms for the Type Inference problem than our PSPACE algorithm.

3 PERIODIC INTERRUPT CALCULUS

3.1 Background

The interrupt calculus described in Chapter 2 does not take into account the timing factor. Its semantics postulates that whenever the master bit of imr and one of the bits of imr which corresponds to interrupts are both enabled, then a handler must be called, thus implying that an interrupt occurs. However, this is not what happens in reality. Most interrupt-driven software handles interrupts in a periodic fashion. That is, the hardware receives an external device's interrupts from time to time with a fixed time interval; at the time of an interrupt arrival, the hardware calls the corresponding interrupt handler. This periodic mechanism is widely used in embedded systems. For example, a temperature sensor, as part of an embedded system, periodically records the current temperature, and when finished collecting data, it signals the microprocessor to receive the data. An interrupt occurs and a handler is called to process the data received. This period is set by the external sensor.

It is important to determine whether all interrupts generated by a device can be processed without missing any of them. Failing to do so may incur data loss, which in turn may result in the instability of the whole system.

In this chapter, we generalize our theoretical framework of the interrupt calculus introduced in Chapter 2 by incorporating periods of external interrupt devices. In particular, we identify the following important parameters regarding a specific interrupt device:

- Period r . The fixed minimum interval between arrivals of the interrupt defined *a priori*.

- Latency t . The time interval between the arrival of an interrupt and the start of handling the interrupt.

We say an interrupt is *missed* if $t \geq r$.

In this chapter, we focus on the semantics that include the *imr* OR operation and allows interrupts to be interleaved. We consider whether interrupts are missed or not (latency), which implies that the latencies of handlers are shorter than their periods $t \leq r$. We analyze the stack size under the impact of the latency constraints, a problem which is called the latency-space safety problem and which is formally defined in Section 3.2.4.

We will formally extend our interrupt calculus to include latencies of external interrupt devices, the result of which is called *periodic interrupt calculus*; then, based on the the periodic interrupt calculus, we define an abstract semantics which *safely approximates* the concrete semantics with respect to the latency-space safety. Our abstraction shows that it is necessary to preserve the *imr* values rather than abstract them because, otherwise, the approximation would not be safe. We will further define our types and the type system for the periodic interrupt calculus. We will use values in abstract semantics as the sources of types. Therefore, our type system is based on the abstract model.

An interesting by-product of our analysis is that it shows an equivalence relation between model checking and type checking. This relation not only confirms the speculation about whether model checking and type system can have the same expressive power to verify the stack boundedness and predictability of interrupt-driven programs, but also offers insights into the differences between the two as well as what kind of applications should choose which of the two tools.

The remainder of this chapter is organized in the following way. In Section 3.2, we formally present the *periodic interrupt calculus*, its syntax and semantics; we then describe an abstract semantics for the periodic interrupt calculus in Section 3.3, which, in essence, is an abstract model; in Section 3.4, we build our type systems on top of the abstract semantics. We then proceed to consider the type construction problem.

In Section 3.5, we illustrate our type construction process by giving a simple example; we formally show how to build types and type derivations from the abstract model in Section 3.6. We show that there is an interesting equivalence relation between type checking and model checking in 3.7 and discuss the advantages and disadvantages of type checking and model checking. Finally, we conclude the chapter by discussing related work in Section 3.8.

3.2 Periodic interrupt calculus

In this section, we introduce the syntax and semantics of the periodic interrupt calculus. We prove the nontermination property of the calculus programs. We also formally state the latency-space problem in this section.

3.2.1 Syntax

$$\begin{array}{ll}
 \text{(system)} & \kappa ::= (\bar{\tau}, p) \\
 \text{(program)} & p ::= (m, \bar{h}) \\
 \text{(main)} & m ::= \text{loop } s \mid s ; m \\
 \text{(handler)} & h ::= \text{iret} \mid s ; h \\
 \text{(statements)} & s ::= x = e \mid \text{imr} = \text{imr} \wedge \text{imr} \mid \text{imr} = \text{imr} \vee \text{imr} \mid \\
 & \quad \text{if0 } x \text{ then } s_1 \text{ else } s_2 \mid s_1 ; s_2 \mid \text{skip} \\
 \text{(expression)} & e ::= c \mid x \mid x + c \mid x_1 + x_2
 \end{array}$$

Figure 3.1. Syntax of periodic interrupt calculus

Figure (3.1) shows the syntax of the periodic interrupt calculus. A periodic interrupt calculus program consists of a number of devices and program p . We assume a fixed number of devices N , whose periods are specified by $\bar{\tau}$. We use u and v to range

over $1..N$, unless we specify otherwise. The overbar notation \bar{r} denotes a sequence $r_1 \dots r_N$, where $\forall u \in 1..N : r_u \geq 0$; and we will use the notation $\bar{r}(u) = r_u$. $\bar{r}(u)$ is the minimum time between the arrivals of consecutive interrupts from device u ,

A program p consists of a main program and N handlers. We use $p.m$ to refer to the main program m in (m, \bar{h}) . The overbar notation \bar{h} denotes a sequence $h_1 \dots h_N$; and we will use the notation $\bar{h}(u) = h_u$. We use x to range over a set of program variables, imr to range over bit strings, and c to range over integer constants. We call $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip** primitive statements. We use the notation \mathcal{L} to denote the set of all primitive statements plus **iret**. We use a to range over m and h .

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 or h into one of the seven forms:

$$\begin{aligned} \text{loop } s \quad \text{iret} \quad x = e; a \quad imr = imr \wedge imr; a \quad imr = imr \vee imr; a \\ (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \quad \text{skip}; a. \end{aligned}$$

3.2.2 Semantics

We use R to denote a *store*, a partial function mapping program variables to integers. We use σ to denote a *stack* generated by the grammar: $\sigma ::= \text{nil} \mid a :: \sigma$. We define the size of a stack as follows: $|\text{nil}| = 0$ and $|a :: \sigma| = 1 + |\sigma|$.

If $imr = b_0 b_1 \dots b_N$, where $b_i \in \{0, 1\}$, then we use the notation $imr(i) = b_i$. We use 0 to denote the imr value where all bits are 0. We use t_i to denote the imr value where all bits are 0's except that the i th bit is set to 1. We use \wedge to denote bitwise

logical conjunction, \vee to denote bitwise logical disjunction, \leq to denote bitwise logical implication, and \neg to denote bitwise logical negation.

We use \bar{T} to denote a *latency vector*. The \bar{T} is a vector of N numbers such that:

- If $\bar{T}(u) \geq 0$, then an interrupt from device u has been *pending* for *at least* $\bar{T}(u)$ units. In particular, if $\bar{T}(u) = 0$, then an interrupt from device u has just arrived.
- If $\bar{T}(u) < 0$, then no interrupt from device u is pending, and the next one will arrive in exactly $-\bar{T}(u)$ units.

We use the notation $\bar{T}_1 \leq \bar{T}_2$ to denote $\forall u \in 1..N : \bar{T}_1(u) \leq \bar{T}_2(u)$.

A program state is a 6-tuple $\langle \bar{h}, R, imr, \bar{T}, \sigma, a \rangle$. We use P to denote program states. If $P = \langle \bar{h}, R, imr, \bar{T}, \sigma, a \rangle$, then we use the notations $P.\bar{h} = \bar{h}$, $P.R = R$, $P.imr = imr$, $P.\sigma = \sigma$, $P.\bar{T} = \bar{T}$, $P.a = a$, and $P.stk = |P.\sigma|$. For $p = (m, \bar{h})$, the initial program state to execute p is $P_0 = \langle \bar{h}, \lambda x.0, 0, \bar{T}_0, nil, m \rangle$, where $\bar{T}_0(u) \leq \bar{0}$.

A small-step operational semantics for the language is a reflexive, transitive closure of the relation \rightarrow on program states, which is defined in Figure (3.2). The semantics uses the following auxiliary definitions:

$$\begin{aligned} eval_R(c) &= c \\ eval_R(x) &= R(x) \\ eval_R(x + c) &= R(x) + c \\ eval_R(x_1 + x_2) &= R(x_1) + R(x_2). \end{aligned}$$

We define the set of pending devices \mathcal{P} as follows:

$$\mathcal{P}(imr, \bar{T}) = \{u \mid imr(0) = 1 \text{ and } imr(u) = 1 \text{ and } \bar{T}(u) \geq 0 \text{ where } u \in 1..N\}$$

The notation η denotes the current program counter and is defined as follows:

$$\begin{aligned} \eta(a) = s & \quad \text{if } a = s; a' \text{ and } s \text{ is } \text{if0 } x \text{ then } s_1 \text{ else } s_2 \text{ or either} \\ & \quad x := e, imr = imr \wedge imr, imr = imr \vee imr \text{ or skip} \\ \eta(a) = a & \quad \text{if } a = \text{loop } s \text{ or } a = \text{iret} \\ \eta(a) = \eta(s_1; (s_2; a')) & \quad \text{if } a = s; a' \text{ and } s \text{ is } s_1; s_2; \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, a \rangle &\rightarrow \langle \bar{h}, R, imr \wedge \neg t_0, \theta_u(\bar{T}), a :: \sigma, \bar{h}(u) \rangle & (3.1) \\ &\text{if } u \in \mathcal{P}(imr, \bar{T}) \wedge \eta(a) \in \mathcal{L} \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, iret \rangle &\rightarrow \langle \bar{h}, R, imr \vee t_0, \theta(\bar{T}), \sigma', a \rangle & (3.2) \\ &\text{if } \mathcal{P}(imr, \bar{T}) = \emptyset \text{ and } \sigma = a :: \sigma' \end{aligned}$$

$$\langle \bar{h}, R, imr, \bar{T}, \sigma, \text{loop } s \rangle \rightarrow \langle \bar{h}, R, imr, \bar{T}, \sigma, s; \text{loop } s \rangle \quad (3.3)$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, x := e; a \rangle &\rightarrow \langle \bar{h}, R\{x \mapsto eval_R(e)\}, imr, \theta(\bar{T}), \sigma, a \rangle & (3.4) \\ &\text{if } \mathcal{P}(imr, \bar{T}) = \emptyset \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, imr = imr \wedge imr'; a \rangle &\rightarrow \langle \bar{h}, R, imr \wedge imr', \theta(\bar{T}), \sigma, a \rangle & (3.5) \\ &\text{if } \mathcal{P}(imr, \bar{T}) = \emptyset \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, imr = imr \vee imr'; a \rangle &\rightarrow \langle \bar{h}, R, imr \vee imr', \theta(\bar{T}), \sigma, a \rangle & (3.6) \\ &\text{if } \mathcal{P}(imr, \bar{T}) = \emptyset \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \rangle &\rightarrow \langle \bar{h}, R, imr, \bar{T}, \sigma, s_1; a \rangle & (3.7) \\ &\text{if } R(x) = 0 \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \rangle &\rightarrow \langle \bar{h}, R, imr, \bar{T}, \sigma, s_2; a \rangle & (3.8) \\ &\text{if } R(x) \neq 0 \end{aligned}$$

$$\begin{aligned} \langle \bar{h}, R, imr, \bar{T}, \sigma, \text{skip}; a \rangle &\rightarrow \langle \bar{h}, R, imr, \theta(\bar{T}), \sigma, a \rangle & (3.9) \\ &\text{if } \mathcal{P}(imr, \bar{T}) = \emptyset \end{aligned}$$

Figure 3.2. Semantics of the periodic interrupt calculus

We define the transfer functions $\theta(\bar{T})$ and $\theta_v(\bar{T})$ over latency vectors \bar{T} as follows:

$$\begin{aligned} \forall u \in 1..N : \theta(\bar{T})(u) &= \bar{T}(u) + 1 \\ \forall u \in 1..N : \theta_v(\bar{T})(u) &= \begin{cases} \bar{T}(u) + 1 & \text{if } u \neq v \\ \bar{T}(u) + 1 - \bar{r}(u) & \text{if } u = v \end{cases} \end{aligned}$$

For the sake of simplicity, we assume the primitive statements each individually execute in one unit of time. Statement `if0 x then s_1 else s_2` and main `loop s` are

instantaneous in the sense that they execute in 0 units of time and there is no handler invocation. Our treatment of `if0 x then s_1 else s_2` and `loop s` does not reduce the complexity of the problem; it is merely used to simplify later part of type construction.

If device u is pending and the instruction at the current program counter is not instantaneous, then Rule (3.1) is applied and handler $\bar{h}(u)$ is called. If multiple devices are pending, one of them is applied nondeterministically. If no device is pending, then one of Rules (3.2)-(3.9) is applied, depending upon the current program counter.

We use $P \rightarrow^n P'$ to denote that there are $n \geq 0$ steps involved in the transition from state P to P' ; we use $P \rightarrow^* P'$ if we do not care about the number n .

Imr Value Transfer Function

For the sake of convenience, we define the following transfer function over imr values for each primitive statement as well as for the interrupt return statement.

$$\begin{aligned} \chi_s(\text{imr}) &= \text{imr} && \text{if } s ::= \text{skip} \mid x := e \\ \chi_s(\text{imr}) &= \text{imr} \wedge \text{imr}' && \text{if } s ::= \text{imr} = \text{imr} \wedge \text{imr}' \\ \chi_s(\text{imr}) &= \text{imr} \vee \text{imr}' && \text{if } s ::= \text{imr} = \text{imr} \vee \text{imr}' \\ \chi_s(\text{imr}) &= \text{imr} \vee \mathbf{t}_0 && \text{if } s ::= \text{iret} \end{aligned}$$

It is straightforward to prove the following proposition.

Proposition 3.2.1 (Monotonicity of Imr Transfer Functions) *The imr transfer functions χ_s are monotone. That is, if $\text{imr}_1 \leq \text{imr}_2$, then $\chi_s(\text{imr}_1) \leq \chi_s(\text{imr}_2)$.*

3.2.3 Nontermination

We say that a program p *can terminate* if $P_0 \rightarrow^* P$ and there is no P' such that $P \rightarrow P'$.

We say that a program state $\langle \bar{h}, R, \text{imr}, \bar{T}, \sigma, a \rangle$ is *consistent* if and only if (1) $\sigma = \text{nil}$ and $a = m$; or (2) $\sigma = h^k :: \dots :: h^1 :: m :: \text{nil}$ and $a = h$, for $k \geq 0$, where $k = 0$ means $\sigma = m :: \text{nil}$.

Lemma 3.2.1 (Consistency Preservation) *If P is consistent and $P \rightarrow P'$, then P' is consistent.*

Proof A straightforward case analysis of $P \rightarrow P'$. ■

Lemma 3.2.2 (Progress) *If P is consistent, then there exists P' such that $P \rightarrow P'$.*

Proof From P is consistent, there are two cases of P .

1. $P = \langle \bar{h}, R, imr, \bar{T}, nil, m \rangle$. There are three cases depending upon the form of $\eta(m)$:

- (a) $\eta(m)$ is of the form `loop s`, then Rule (3.3) gives P' such that $P \rightarrow P'$.
- (b) $\eta(m)$ is of the form `if0 x then s1 else s2`, then Rules (3.7-3.8), depending upon whether $R(x) = 0$, gives P' such that $P \rightarrow P'$.
- (c) $\eta(m)$ is either of the form `x := e`, `imr = imr ∧ imr`, `imr = imr ∨ imr` or `skip`.

Then we have $\eta(m) \in \mathcal{L}$. There are two subcases:

- i. $u \in \mathcal{P}(imr, \bar{T})$, then Rule (3.1) gives P' such that $P \rightarrow P'$.
- ii. $\mathcal{P}(imr, \bar{T}) = \emptyset$, then Rules (3.4-3.6) and (3.9), depending upon what $\eta(m)$ is, ensure that there exists a state P' such that $P \rightarrow P'$.

2. $P = \langle \bar{h}, R, imr, \bar{T}, h^k :: \dots :: h^1 :: m :: nil, h \rangle$, $k \geq 0$. There are two cases depending upon the form of $\eta(h)$:

(a) $\eta(h)$ is of the form `iret`, then $\eta(h) \in \mathcal{L}$. There are two subcases:

- i. $u \in \mathcal{P}(imr, \bar{T})$, then Rule (3.1) gives P' such that $P \rightarrow P'$.
- ii. $\mathcal{P}(imr, \bar{T}) = \emptyset$, then from Rule (3.2)
 - either $k = 0$ and $s = m :: nil$, and hence $P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, nil, m \rangle$,
 - or $k > 0$ and hence $P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, \bar{T}, h^{k-1} :: \dots :: h^1 :: m :: nil, h^k \rangle$.

(b) $\eta(h)$ is either of the form `x := e`, `imr = imr ∧ imr`, `imr = imr ∨ imr` or `skip`, then we have $\eta(h) \in \mathcal{L}$. There are two subcases:

- i. $u \in \mathcal{P}(imr, \bar{T})$, then Rule (3.1) gives P' such that $P \rightarrow P'$.
- ii. $\mathcal{P}(imr, \bar{T}) = \emptyset$, then Rules (3.4-3.6) and (3.9), depending upon what $\eta(h)$ is, ensure that there exists a state P' such that $P \rightarrow P'$.
- (c) $\eta(h)$ is of the form *if0 x then s₁ else s₂*, then Rules (3.7-3.8), depending upon whether $R(x) = 0$, gives P' such that $P \rightarrow P'$.

■

Theorem 3.2.1 (Nontermination) *No program can terminate.*

Proof Suppose a program p can terminate, that is, suppose $P_0 \rightarrow^* P$ and there is no P' such that $P \rightarrow P'$. Notice first that P_0 is consistent by consistency criterion (1). From Lemma 3.2.1 and induction on the number of execution steps in $P_0 \rightarrow^* P$, we have that P is consistent. From Lemma 3.2.2 we have that there exists P' such that $P \rightarrow P'$, a contradiction. ■

3.2.4 Latency-space safety analysis

Given a natural number K , the latency-space safety analysis of a periodic interrupt calculus program is that, for each state of the program that is reachable from the initial state, the latencies of the state do not exceed the periods and the stack size of the state is also no greater than K . If the above safety property is satisfied, then we say that the program is *safe*. Formally, we define the latency-space safety analysis as: given $K \geq 0$,

$$P_0 \rightarrow^* P \Rightarrow (P.\bar{T} \leq \bar{r} \wedge P.stk \leq K)$$

3.3 Abstract semantics

In this section, we eliminate the store component of the concrete program state in the abstract semantics, which makes latency-space safety analysis decidable. In addition, we approximate the concrete latency vector (\bar{T}) by using the abstract latency

vector $\overline{\overline{T}}$. We call the result the abstract semantics. We further show that the abstract semantics *safely approximates* the concrete semantics with respect to the latency-space problem (soundness).

We will use the binary relation \leq to specify the relationship between a \overline{T} and its approximation $\overline{\overline{T}}$. The meaning is that $\overline{\overline{T}}$ is a conservative approximation of \overline{T} : for all $u \in 1..N$, we have $\overline{T}(u) \leq \overline{\overline{T}}(u)$.

An abstract program state is a 5-tuple $\langle \overline{h}, imr, \overline{\overline{T}}, \sigma, a \rangle$, where imr is the current imr value and $\overline{\overline{T}}$ is the abstract latency vector. \overline{h} is the group of handlers, σ is the program stack, a is the current statement. We use Q to denote abstract program states. If $Q = \langle \overline{h}, imr, \overline{\overline{T}}, \sigma, a \rangle$, then we use the notations $Q.\overline{h} = \overline{h}$, $Q.imr = imr$, $Q.\sigma = \sigma$, $Q.\overline{\overline{T}} = \overline{\overline{T}}$, $Q.a = a$, and $Q.stk = |\overline{\overline{T}}|$. For $p = (m, \overline{h})$, the initial abstract program state to execute p is $Q_0 = \langle \overline{h}, 0, \overline{\overline{T}}_0, nil, m \rangle$, where $\overline{\overline{T}}_0 = \overline{0}$.

We now formally define the approximation relation between the concrete states and the abstract states.

Definition 3.3.1 (Approximation Relation \mathcal{A}) *A binary relation \mathcal{A} over concrete and abstract states is an approximation relation $(P, Q) \in \mathcal{A}$ iff $P.\overline{h} = Q.\overline{h}$, $P.imr = Q.imr$, $P.\sigma = Q.\sigma$, $P.\overline{\overline{T}} \leq Q.\overline{\overline{T}}$ and $P.a = Q.a$.*

An abstract version of the pending device set $\widehat{\mathcal{P}}$ is defined as follows:

$$\widehat{\mathcal{P}}(imr, \overline{\overline{T}}) = \{u \mid imr(0) = 1 \text{ and } imr(u) = 1 \text{ and } \overline{\overline{T}}(u) \geq 0 \text{ where } u \in 1..N\}$$

For any $u \in 0..N$, since $\overline{\overline{T}}$ conservatively approximates \overline{T} , if $\overline{T}(u) \geq 0$, then $\overline{\overline{T}}(u) \geq 0$. We, thus, have the follow proposition regarding \mathcal{P} and $\widehat{\mathcal{P}}$:

Proposition 3.3.1 ($\widehat{\mathcal{P}}$ approximation) *If $\overline{\overline{T}} \leq \overline{\overline{T}}$, then $\mathcal{P}(imr, \overline{\overline{T}}) \subseteq \widehat{\mathcal{P}}(imr, \overline{\overline{T}})$.*

The abstract transfer functions $\widehat{\theta}(imr, \overline{\overline{T}})$ and $\widehat{\theta}_v(\overline{\overline{T}})$, where $v \in 1..N$, over an imr value and abstract latency vectors $\overline{\overline{T}}$, are defined as follows:

$$\forall u \in 1..N : \widehat{\theta}(imr, \overline{\overline{T}})(u) = \begin{cases} 0 & \text{if } u \in \widehat{\mathcal{P}}(imr, \overline{\overline{T}}) \\ \overline{\overline{T}}(u) + 1 & \text{otherwise} \end{cases}$$

$$\forall u \in 1..N : \widehat{\theta}_v(\overline{T})(u) = \begin{cases} \overline{T}(u) + 1 - \overline{r}(u) & \text{if } u = v \\ \overline{T}(u) + 1 & \text{otherwise} \end{cases}$$

Note that the abstract transfer functions take imr as an extra argument compared with their concrete versions because it has to be known which handler is enabled at the time when approximating the latency of the next step.

For those steps other than the interrupt (Rule 3.1), if handler u is pending to be called, then the best approximation of the latency of the device u after executing the step is 0 ($\widehat{\theta}(imr, \overline{T}(u)) = 0$ if $u \in \widehat{\mathcal{P}}(imr, \overline{T})$). This is because if a non-instantaneous step of Rule (3.2-3.9) is applied in the concrete semantics, then no interrupt device is pending; however, the reason that the device is pending in the abstract semantics is only because we over-approximate its latency by a positive number. Therefore, the the best approximation is 0.

A small step operational abstract semantics is a reflexive, transitive closure of the relation \hookrightarrow over program states, which is defined in Figure (3.3).

If device u is pending and the current program counter is not instantaneous, then Rule (3.1) might be applied, in which case handler $\overline{h}(u)$ is called. If multiple devices are pending, one of them may be applied nondeterministically. Otherwise, one of Rules (3.2)-(3.9) is applied, depending upon the current program counter.

We use $Q \hookrightarrow^n Q'$ to denote that there are $n \geq 0$ steps involved in the transition from state Q to Q' ; we use $Q \hookrightarrow^* Q'$ if we do not care about the number n .

Abstract State Consistency We say that an abstract program state $\langle \overline{h}, imr, \overline{T}, \sigma, a \rangle$ is *consistent* if and only if (1) $\sigma = \text{nil}$ and $a = m$; or (2) $\sigma = h^k :: \dots :: h^1 :: m :: \text{nil}$ and $a = h$, for $k \geq 0$, where $k = 0$ means $\sigma = m :: \text{nil}$.

Lemma 3.3.1 (Consistency Preservation) *If Q is consistent and $Q \hookrightarrow Q'$, then Q' is consistent.*

Proof The proof is a straightforward case analysis on $Q \hookrightarrow Q'$. We omit the details. ■

$$\langle \bar{h}, imr, \bar{T}, \sigma, a \rangle \hookrightarrow \langle \bar{h}, imr \wedge \neg \mathbf{t}_0, \hat{\theta}_u(\bar{T}), a :: \sigma, \bar{h}(u) \rangle \quad (3.10)$$

if $u \in \hat{\mathcal{P}}(imr, \bar{T}) \wedge \eta(a) \in \mathcal{L}$

$$\langle \bar{h}, imr, \bar{T}, \sigma, \text{iret} \rangle \hookrightarrow \langle \bar{h}, imr \vee \mathbf{t}_0, \hat{\theta}(imr, \bar{T}), \sigma', a \rangle \quad (3.11)$$

if $\sigma = a :: \sigma'$

$$\langle \bar{h}, imr, \bar{T}, \sigma, \text{loop } s \rangle \hookrightarrow \langle \bar{h}, imr, \bar{T}, \sigma, s; \text{loop } s \rangle \quad (3.12)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, x := e; a \rangle \hookrightarrow \langle \bar{h}, imr, \hat{\theta}(imr, \bar{T}), \sigma, a \rangle \quad (3.13)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, imr = imr \wedge imr'; a \rangle \hookrightarrow \langle \bar{h}, imr \wedge imr', \hat{\theta}(imr, \bar{T}), \sigma, a \rangle \quad (3.14)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, imr = imr \vee imr'; a \rangle \hookrightarrow \langle \bar{h}, imr \vee imr', \hat{\theta}(imr, \bar{T}), \sigma, a \rangle \quad (3.15)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \rangle \hookrightarrow \langle \bar{h}, imr, \bar{T}, \sigma, s_1; a \rangle \quad (3.16)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \rangle \hookrightarrow \langle \bar{h}, imr, \bar{T}, \sigma, s_2; a \rangle \quad (3.17)$$

$$\langle \bar{h}, imr, \bar{T}, \sigma, \text{skip}; a \rangle \hookrightarrow \langle \bar{h}, imr, \hat{\theta}(imr, \bar{T}), \sigma, a \rangle \quad (3.18)$$

Figure 3.3. Abstract semantics of the periodic interrupt calculus

Corollary 3.3.1 (Consistent Program State) *If $Q_0 \hookrightarrow^n Q$, then Q is consistent.*

Proof We prove by an induction on n . In the base case $n = 0$, we have Q_0 is consistent by definition. In the induction step, we have $Q_0 \hookrightarrow^{n+1} Q$, from which it follows $Q_0 \hookrightarrow^n Q'$ and $Q' \hookrightarrow Q$. From $Q_0 \hookrightarrow^n Q'$ and the induction hypothesis, we have Q' is consistent. From Q' is consistent, $Q' \hookrightarrow Q$ and Lemma 3.3.1, we have Q is consistent. ■

Abstract state space We define the abstract state space as the set of all reachable abstract states from the initial abstract state Q_0 .

Definition 3.3.2 *Define $\mathcal{R} = \{Q \mid Q_0 \hookrightarrow^* Q\}$.*

We next prove that if each latency vector of the interrupt calculus program is bounded by its respective periods and if the stack size is bound by $K \geq 0$, then \mathcal{R} is finite.

Theorem 3.3.1 *Given $K \geq 0$, if $Q_0 \hookrightarrow^* Q \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K)$, then \mathcal{R} is finite.*

Proof An abstract state Q consists of $imr, \overline{T}, \sigma$ and a . We next show that there are a finite number of abstract states by proving that every component of the abstract state is finite. There are a finite number of imr values. If $Q_0 \hookrightarrow^* Q \Rightarrow Q.\overline{T} \leq \overline{r}$, then there are a finite number of \overline{T} . There are a finite number of program points a . If $Q_0 \hookrightarrow^* Q \Rightarrow Q.stk \leq K$, then there are a finite number of σ because the size of σ is bound by K and the return points a are finite. ■

Soundness of Abstract Semantics

The soundness of abstract semantics over concrete semantics lies in the fact that the abstract semantics *conservatively* approximates the concrete semantics with respect to latency-space safety analysis in the sense that, if for any abstract Q that is reachable from Q_0 , we have $Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K$, then for any P that is reachable from P_0 , we have $P.\overline{T} \leq \overline{r} \wedge P.stk \leq K$. We next prove in Corollary (3.3.2) that abstract semantics is sound with respect to the latency-space analysis.

Lemma 3.3.2 (Single Step Approximation) *If $(P, Q) \in \mathcal{A}$ and $P \rightarrow P'$, then there exists a state Q' such that $Q \hookrightarrow Q'$ and $(P', Q') \in \mathcal{A}$.*

Proof There are nine cases depending on which one of Rules (3.1 - 3.9) is used in $P \rightarrow P'$.

- Rule (3.1).

We have $P = \langle \overline{h}, R, imr, \overline{T}, \sigma, a \rangle$, $P' = \langle \overline{h}, R, imr \wedge \neg t_0, \theta_v(\overline{T}), a :: \sigma, \overline{h}(v) \rangle$ where $v \in \mathcal{P}(imr, \overline{T})$, $\eta(a) \in \mathcal{L}$, and $Q = \langle \overline{h}, imr, \overline{T}, \sigma, a \rangle$. From $(P, Q) \in \mathcal{A}$, we have $\overline{T} \leq \overline{T}$. From $\overline{T} \leq \overline{T}$ and Proposition (3.3.1), we have $\mathcal{P}(imr, \overline{T}) \subseteq \widehat{\mathcal{P}}(imr, \overline{T})$.

From $v \in \mathcal{P}(imr, \bar{T})$ and $\mathcal{P}(imr, \bar{T}) \subseteq \widehat{\mathcal{P}}(imr, \bar{T})$, we have $v \in \widehat{\mathcal{P}}(imr, \bar{T})$. Let $Q' = \langle \bar{h}, imr \wedge \neg \mathbf{t}_0, \widehat{\theta}_v(\bar{T}), a :: \sigma, \bar{h}(v) \rangle$. From $v \in \widehat{\mathcal{P}}(imr, \bar{T})$, $\eta(a) \in \mathcal{L}$ and Rule (3.10), we have $Q \hookrightarrow Q'$. We need to prove $\forall u \in 1..N : \theta_v(\bar{T})(u) \leq \widehat{\theta}_v(\bar{T})(u)$. There are two subcases.

1. Subcase 1: $u = v$.

From $u = v$, we have $\theta_v(\bar{T})(u) = \bar{T}(u) + 1 - \bar{r}(u)$. From $\theta_v(\bar{T})(u) = \bar{T}(u) + 1 - \bar{r}(u)$ and $\bar{T}(u) \leq \bar{T}(u)$ we have $\theta_v(\bar{T})(u) \leq \bar{T}(u) + 1 - \bar{r}(u)$. From $\widehat{\theta}_v(\bar{T})(u) = \bar{T}(u) + 1 - \bar{r}(u)$, we have $\theta_v(\bar{T})(u) \leq \widehat{\theta}_v(\bar{T})(u)$.

2. Subcase 2: $u \neq v$.

From $v \in \widehat{\mathcal{P}}(imr, \bar{T})$, we have $imr(0) = 1$. From $u \neq v$, we have $\theta_v(\bar{T})(u) = \bar{T}(u) + 1$ and $\widehat{\theta}_v(\bar{T})(u) = \bar{T}(u) + 1$. From $\bar{T}(u) \leq \bar{T}(u)$, it follows $\bar{T}(u) + 1 \leq \bar{T}(u) + 1$. From $\theta_v(\bar{T})(u) = \bar{T}(u) + 1$, $\widehat{\theta}_v(\bar{T})(u) = \bar{T}(u) + 1$ and $\bar{T}(u) + 1 \leq \bar{T}(u) + 1$, we have $\theta_v(\bar{T})(u) \leq \widehat{\theta}_v(\bar{T})(u)$.

Thus, we have $\theta_v(\bar{T}) \leq \widehat{\theta}_v(\bar{T})$, from which it follows $(P', Q') \in \mathcal{A}$.

• Rule (3.2).

We have $P = \langle \bar{h}, R, imr, \bar{T}, \sigma, \text{iret} \rangle$ where $\sigma = a :: \sigma'$,

$P' = \langle \bar{h}, R, imr \vee \mathbf{t}_0, \theta(\bar{T}), \sigma', a \rangle$ and $Q = \langle \bar{h}, imr, \bar{T}, \sigma, \text{iret} \rangle$.

Let $Q' = \langle \bar{h}, imr \vee \mathbf{t}_0, \widehat{\theta}(imr, \bar{T}), \sigma', a \rangle$. From $\sigma = a :: \sigma'$ and Rule (3.11), we have $Q \hookrightarrow Q'$. From $P \rightarrow P'$ and Rule (3.2), we have $\mathcal{P}(imr, \bar{T}) = \emptyset$. From $(P, Q) \in \mathcal{A}$, we have $\bar{T} \leq \bar{T}$. We need to prove $\forall u \in 1..N : \theta(\bar{T})(u) \leq \widehat{\theta}(imr, \bar{T})(u)$. There are two cases.

1. $u \in \widehat{\mathcal{P}}(imr, \bar{T}(u))$, in which case $\widehat{\theta}(imr, \bar{T})(u) = 0$.

From $u \in \widehat{\mathcal{P}}(imr, \bar{T}(u))$, we have $imr(0) = 1$ and $imr(u) = 1$. From $\mathcal{P}(imr, \bar{T}) = \emptyset$, we have $u \notin \mathcal{P}(imr, \bar{T})$. From $imr(0) = 1$, $imr(u) = 1$ and $u \notin \mathcal{P}(imr, \bar{T})$, it follows that $\bar{T}(u) < 0$. From $\bar{T}(u) < 0$, we have $\bar{T}(u) + 1 \leq 0$. From $\theta(\bar{T})(u) = \bar{T}(u) + 1$ and $\bar{T}(u) + 1 \leq 0$, we

have $\theta(\overline{T})(u) \leq 0$. From $\widehat{\theta}(imr, \overline{\overline{T}})(u) = 0$ and $\theta(\overline{T})(u) \leq 0$, it follows $\theta(\overline{T})(u) \leq \widehat{\theta}(imr, \overline{\overline{T}})(u)$.

2. $u \notin \widehat{\mathcal{P}}(imr, \overline{\overline{T}}(u))$, in which case $\widehat{\theta}(imr, \overline{\overline{T}})(u) = \overline{\overline{T}}(u) + 1$.

From $\overline{T}(u) \leq \overline{\overline{T}}(u)$, it follows $\overline{T}(u) + 1 \leq \overline{\overline{T}}(u) + 1$. From $\theta(\overline{T})(u) = \overline{T}(u) + 1$, $\widehat{\theta}(imr, \overline{\overline{T}})(u) = \overline{\overline{T}}(u) + 1$ and $\overline{T}(u) + 1 \leq \overline{\overline{T}}(u) + 1$, we have $\theta(\overline{T})(u) \leq \widehat{\theta}(imr, \overline{\overline{T}})(u)$.

Thus, we have $\theta(\overline{T}) \leq \widehat{\theta}(imr, \overline{\overline{T}})$, from which it follows $(P', Q') \in \mathcal{A}$.

- Rule (3.4-3.6) and Rule (3.9).

The proofs are similar to that of Rule (3.2).

- Rule (3.3).

We have $P = \langle \overline{h}, R, imr, \overline{T}, \sigma, \text{loop } s \rangle$, $P' = \langle \overline{h}, R, imr, \overline{T}, \sigma, s; \text{loop } s \rangle$ and $Q = \langle \overline{h}, imr, \overline{\overline{T}}, \sigma, \text{loop } s \rangle$. Let $Q' = \langle \overline{h}, imr, \overline{\overline{T}}, \sigma, s; \text{loop } s \rangle$. From Rule (3.12), we have $Q \hookrightarrow Q'$. From $(P, Q) \in \mathcal{A}$, we have $\overline{T} \leq \overline{\overline{T}}$. From $\overline{T} \leq \overline{\overline{T}}$, we have $(P', Q') \in \mathcal{A}$.

- Rule (3.7 - 3.8).

The proofs are similar to that of Rule (3.3). ■

Lemma 3.3.3 (Multi-step Approximation) *If $(P, Q) \in \mathcal{A}$ and $P \rightarrow^n P'$, then there exists a state Q' such that $Q \hookrightarrow^n Q'$ and $(P', Q') \in \mathcal{A}$.*

Proof Proof by induction. We omit the base case where $n = 0$ because it is trivial. In the induction step, we suppose $(P, Q) \in \mathcal{A}$ and $P \rightarrow^n P' \rightarrow P''$. From $(P, Q) \in \mathcal{A}$, $P \rightarrow^n P'$ and the induction hypothesis, there exists a Q' such that $Q \hookrightarrow^n Q'$ and $(P', Q') \in \mathcal{A}$. From $(P', Q') \in \mathcal{A}$, $P' \rightarrow P''$ and Lemma (3.3.2), there exists a state Q'' such that $Q \hookrightarrow^{n+1} Q''$ and $(P'', Q'') \in \mathcal{A}$. ■

Corollary 3.3.2 (Soundness of Abstract Semantics) *If $Q_0 \hookrightarrow^n Q \Rightarrow (Q.\overline{\overline{T}} \leq \overline{r} \wedge Q.stk \leq K)$, then $P_0 \rightarrow^n P \Rightarrow (P.\overline{\overline{T}} \leq \overline{r} \wedge P.stk \leq K)$.*

Proof From $\forall u \in 1..N : \overline{T}_0(u) \leq 0$ and $\forall u \in 1..N : \overline{\overline{T}}_0(u) = 0$, we have $\overline{T}_0 \leq \overline{\overline{T}}_0$. From $P_0 = \langle \overline{h}, R, 0, \overline{T}_0, \text{nil}, m \rangle$, $Q_0 = \langle \overline{h}, R, 0, \overline{\overline{T}}_0, \text{nil}, m \rangle$ and $\overline{T}_0 \leq \overline{\overline{T}}_0$, we have $(P_0, Q_0) \in \mathcal{A}$. From $P_0 \rightarrow^n P$, $(P_0, Q_0) \in \mathcal{A}$ and Lemma (3.3.3), there exists a state Q such that $Q_0 \hookrightarrow^n Q$ and $(P, Q) \in \mathcal{A}$. From $(P, Q) \in \mathcal{A}$ and Definition (3.3.1), we have $P.\overline{T} \leq Q.\overline{\overline{T}} \wedge P.stk = Q.stk$. From $Q.\overline{\overline{T}} \leq \overline{r}$, $Q.stk \leq K$ and $P.\overline{T} \leq Q.\overline{\overline{T}} \wedge P.stk = Q.stk$. We have $P.\overline{T} \leq \overline{r} \wedge P.stk \leq K$. \blacksquare

3.4 Type system

The abstract semantics gives an abstraction of the concrete semantics. We will use it as the source of types. In this section, we introduce a type system for the periodic interrupt calculus. We show that if a periodic interrupt calculus program type checks, then the program *cannot go wrong* with respect to the latency-space problem (type soundness).

3.4.1 Types

We use K to range over the integers, and we use δ to range over the nonnegative integers. We call the pairs $(imr, \overline{\overline{T}})$ in abstract states *contexts*. We use the sets of contexts as types. A single context $(imr, \overline{\overline{T}})$ is a singleton type. Assuming that the set of all the contexts $(imr, \overline{\overline{T}})$ is uniquely indexed, we use i and j to range over the indexes; we use I and J to range over the index sets. In particular, for $u \in 1..N$, we use the notation I^u to denote the index set such that if $i \in I^u$, then the pair $(imr_i, \overline{\overline{T}}_i)$ is a context in which handler $\overline{h}(u)$ can be invoked; we use notation J_i^u to denote the index set such that if $j \in J_i^u$, then the pair $(imr_j, \overline{\overline{T}}_j)$ is a context in which handler $\overline{h}(u)$ which is invoked in the context $(imr_i, \overline{\overline{T}}_i)$ can return; we use the notation δ_i^u to denote the maximum stack growth over the course of the call of $\overline{h}(u)$ invoked in context $(imr_i, \overline{\overline{T}}_i)$.

Specifically, we define types as follows:

$$\begin{aligned}
(\text{Type of } s) \quad S & ::= \bigwedge_{i \in I} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{T}_j) \\
(\text{Type of } m) \quad M & ::= \bigvee_{i \in I} imr_i, \overline{T}_i. \\
(\text{Type of } h) \quad H & ::= \bigwedge_{i \in I} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{T}_j) \\
(\text{Type of handler } \overline{h}(u), u \in 1..N) \quad \tau & ::= \bigwedge_{i \in I^u} (imr_i, \overline{T}_i \xrightarrow{\delta_i^u} \bigvee_{j \in J_i^u} imr_j, \overline{T}_j)
\end{aligned}$$

We define the type of statement s as an intersection type:

$\bigwedge_{i \in I} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{T}_j)$, indicating that if statement s starts its execution in the context (imr_i, \overline{T}_i) , where $i \in I$, then it will finish its execution in one of the contexts $\{imr_j, \overline{T}_j \mid j \in J_i\}$, where J_i is the resultant index set of executing s in the context (imr_i, \overline{T}_i) . We define the type of main m as a union type: $\bigvee_{i \in I} imr_i, \overline{T}_i$, indicating that m can be safely executed if it starts its execution in one of the contexts $\{imr_i, \overline{T}_i \mid i \in I\}$. We define the type of h as an intersection type: $\bigwedge_{i \in I} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{T}_j)$, which bears a similar meaning to that of statements. For $u \in 1..N$, we define handler type τ as an intersection type: $\bigwedge_{i \in I^u} (imr_i, \overline{T}_i \xrightarrow{\delta_i^u} \bigvee_{j \in J_i^u} imr_j, \overline{T}_j)$, indicating that if the handler is invoked in the context (imr_i, \overline{T}_i) , where $i \in I^u$, then it will return in one of the contexts $\{imr_j, \overline{T}_j \mid j \in J_i^u\}$, in these calls stack growth will not exceed the number δ_i^u . We use notation $\overline{\tau}$ to denote handler types such that $\overline{\tau}(u)$ is the type of handler $\overline{h}(u)$.

For $u \in 1..N$, we define the well-formedness of $\overline{\tau}(u)$ as follows.

Definition 3.4.1 (Well-formedness of $\overline{\tau}(u)$)

$\overline{\tau}(u) = \bigwedge_{i \in I^u} (imr_i, \overline{T}_i \xrightarrow{\delta_i^u} \bigvee_{j \in J_i^u} imr_j, \overline{T}_j)$ is well formed if $I^u \neq \emptyset$ and $\forall i \in I^u : \delta_i^u \geq 0 \wedge J_i^u \neq \emptyset$.

We will use the following type judgments:

Type Judgment	Meaning
$\bar{\tau} \vdash_K (m, \bar{h})$	program $p = (m, \bar{h})$ type checks
$\bar{\tau} \vdash_K Q$	state P type checks
$\bar{\tau}, i \vdash_K \sigma$	stack σ type checks
$\bar{\tau} \vdash \bar{h}(u) : \bar{\tau}(u)$	handler h has type $\bar{\tau}(u)$
$\bar{\tau} \vdash_K m : M$	main part m has type M
$\bar{\tau} \vdash_K h : H$	Handler h has type H
$\bar{\tau} \vdash_K s : S$	Statement s has type S

The typing environment $\bar{\tau}$ carries the types of all handlers and is present in all judgments. The judgment $\bar{\tau} \vdash_K (m, \bar{h})$ means that program $p = (m, \bar{h})$ type checks in the environment $\bar{\tau}$. The integer K bounds the stack size to be at most K . We can view K as a “stack budget” in the sense that, any time an element is placed on the stack, the budget decrements by one, and, when an element is removed from the stack, the budget increments by one. The type system ensures that the budget does not go below zero. The judgment $\bar{\tau} \vdash_K Q$ means that the abstract program state Q type checks in the environment $\bar{\tau}$. The judgment $\bar{\tau}, i \vdash_K \sigma$ means that stack σ type checks in the environment $\bar{\tau}$ and in the context (imr_i, \bar{T}_i) . The judgment $\bar{\tau} \vdash \bar{h}(u) : \bar{\tau}(u)$ means that handler $\bar{h}(u)$ has type $\bar{\tau}(u)$ in the environment $\bar{\tau}$. We abbreviate the family of judgments $\bar{\tau} \vdash \bar{h}(u) : \bar{\tau}(u)$ as $\bar{\tau} \vdash \bar{h} : \bar{\tau}$. The judgment $\bar{\tau} \vdash_K m : M$ means that main part m has type M in the environment $\bar{\tau}$. The judgments $\bar{\tau} \vdash_K h : H$ and $\bar{\tau} \vdash_K s : S$ have similar meanings to that of $\bar{\tau} \vdash_K m : M$.

We define the well-formedness of type judgments for s , m and h as follows.

Definition 3.4.2 (Well-formedness of type judgments) *Define*

1. $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j)$ is well formed if $I \neq \emptyset$, $\forall i \in I : J_i \neq \emptyset$ and $K \geq 0$
2. $\bar{\tau} \vdash_K m : \bigvee_{i \in I} imr_i, \bar{T}_i$ is well formed if $I \neq \emptyset$ and $K \geq 0$
3. $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j)$ is well formed if $I \neq \emptyset$, $\forall i \in I : J_i \neq \emptyset$ and $K \geq 0$

3.4.2 Type rules

For the sake of convenience, we will use the following auxiliary notations in our later presentation of the chapter. ξ_s , ξ_{ret} and ζ_u are all defined on the context indexes.

- $\xi_s(i) = j$ such that $imr_j = \chi_s(imr_i)$ and $\bar{T}_j = \hat{\theta}(imr_i, \bar{T}_i)$, where s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip .
- $\xi_{\text{ret}}(i) = j$ such that $imr_j = imr_i \vee \mathbf{t}_0$ and $\bar{T}_j = \hat{\theta}(imr_i, \bar{T}_i)$
- $\zeta_u(i) = j$ such that $imr_j = imr_i \wedge \neg \mathbf{t}_0$ and $\bar{T}_j = \hat{\theta}_u(\bar{T}_i)$.

We define the type rules as follows, where $\text{safe}(\bar{\tau}, I, K)$ is defined in Definition (3.4.3).

$$\frac{\bar{\tau} \vdash_K m : (0, \bar{0}) \quad \bar{\tau} \vdash \bar{h} : \bar{\tau}}{\bar{\tau} \vdash_K (m, \bar{h})} \quad (3.19)$$

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K m : \bigvee_{i \in I} imr_i, \bar{T}_i}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \text{nil}, m \rangle} \quad [i' \in I] \quad (3.20)$$

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K h : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, h \rangle} \quad [i' \in I] \quad (3.21)$$

$$\frac{\bar{\tau} \vdash_{K+1} m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i}{\bar{\tau}, i' \vdash_K m :: \text{nil}} \quad [i' \in I] \quad (3.22)$$

$$\frac{\bar{\tau} \vdash_{K+1} h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_{K+1} \sigma}{\bar{\tau}, i' \vdash_K h :: \sigma} \quad [i' \in I] \quad (3.23)$$

$$\frac{\bar{\tau} \vdash_{\delta_i^u} \bar{h}(u) : \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)} \rightarrow \bigvee_{j \in J_i^u} \text{imr}_j, \bar{T}_j \quad \forall i \in I^u}{\bar{\tau} \vdash \bar{h}(u) : \bigwedge_{i \in I^u} \text{imr}_i, \bar{T}_i, \xrightarrow{\delta_i^u} \bigvee_{j \in J_i^u} \text{imr}_j, \bar{T}_j} \quad (3.24)$$

$$\frac{\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}{\bar{\tau} \vdash_K \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i} \quad [\forall i \in I : J_i \subseteq I] \quad (3.25)$$

$$\frac{\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \quad \bar{\tau} \vdash_K m : \bigvee_{i \in I'} \text{imr}_i, \bar{T}_i}{\bar{\tau} \vdash_K s; m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i} \quad [\forall i \in I : J_i \subseteq I'] \quad (3.26)$$

$$\frac{\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j \quad \bar{\tau} \vdash_K h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J''_i} \text{imr}_j, \bar{T}_j}{\bar{\tau} \vdash_K s; h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j} \quad [\forall i \in I : J'_i \subseteq I'] \quad (3.27)$$

where $\forall i \in I : J_i = \bigcup \{J''_j \mid j \in J'_i\}$

$$\bar{\tau} \vdash_K \text{iret} : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \quad [\text{safe}(\bar{\tau}, I, K)] \quad (3.28)$$

where $\forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \hat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} J_i^u\} \cup \{\xi_{\text{iret}}(i)\}$

$$\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \quad [\text{safe}(\bar{\tau}, I, K)] \quad (3.29)$$

where $\forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \hat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} J_i^u\} \cup \{\xi_s(i)\}$

and s is a primitive statement

$$\frac{\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \bar{T}_j \quad \bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \bar{T}_j}{\bar{\tau} \vdash_K \text{if0 } x \text{ then } s_1 \text{ else } s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t \cup J_i^f} \text{imr}_j, \bar{T}_j} \quad (3.30)$$

$$\begin{array}{c}
\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} imr_j, \bar{T}_j \\
\bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I'} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J''_i} imr_j, \bar{T}_j \\
\hline
\bar{\tau} \vdash_K s_1; s_2 : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j \quad [\forall i \in I : J'_i \subseteq I'] \quad (3.31) \\
\text{where } \forall i \in I : J_i = \bigcup \{J''_j \mid j \in J'_i\}
\end{array}$$

We define the condition $safe(\bar{\tau}, I, K)$ as follows.

Definition 3.4.3 (Latency-Space Safety Condition) Define $safe(\bar{\tau}, I, K)$ as

$$\begin{array}{l}
(1) \quad (i \in I \wedge u \in \hat{\mathcal{P}}(imr_i, \bar{T}_i)) \Rightarrow \left\{ \begin{array}{l} (i) \quad i \in I^u \text{ and} \\ (ii) \quad J_i^u \subseteq I \text{ and} \\ (iii) \quad \delta_i^u + 1 \leq K \text{ and} \\ (iv) \quad \hat{\theta}_u(\bar{T}_i) \leq \bar{\tau} \end{array} \right. \\
(2) \quad \forall i \in I : \hat{\theta}(imr_i, \bar{T}_i) \leq \bar{\tau}
\end{array}$$

Rule (3.19) is for type checking the periodic interrupt calculus program $p = (m, \bar{h})$. If both m and \bar{h} type check, then the program type checks. Rules (3.20)-(3.21) are for type checking abstract states. The side condition $i' \in I$ of both rules ensures that the context $(imr_{i'}, \bar{T}_{i'})$ of the state is covered by the type. In Rule (3.21), the second hypothesis ensures that it is safe for the handler to return to its calling contexts. This hypothesis involves type checking stacks, which is specified in Rules (3.22)-(3.23).

Rule (3.24) is for type checking handlers. Notice that the handler type is an intersection type, we, therefore, need to check all the component types of the intersection with their respective stack budget. The rule type checks the handler with $imr(0)$ turned off.

Rules (3.26)-(3.25) are for type checking the main part of the program.

Rules (3.27)-(3.28) are for type checking the handler part. The side condition of Rule (3.27) states that the context in which statement s finishes should be the context in which h starts executing. However, the context in which h starts executing is *not necessarily* the context in which statement s finishes because, before the actual execution of h , there may be invocations of other handlers which correspond to pending

devices, in which case, the context in which such an invocation of handler returns is also the context in which h starts executing.

Rule (3.28) has a latency-space safety condition as a side condition. Condition $\text{safe}(\bar{\tau}, I, K)$ is used to make sure that within context $(\text{imr}_i, \overline{\overline{T}}_i)$, $i \in I$, it is safe for a handler which corresponds to a pending device to be invoked and for the handler to return. In particular, the safety condition in the rule states the following conditions:

Item (1) Item (i) of the implication part states that if there is a pending device u in the context $(\text{imr}_i, \overline{\overline{T}}_i)$ then the context is one of the contexts in which handler $\bar{h}(u)$ starts executing. This is because, according to the abstract semantics, if a device u is pending in context $(\text{imr}_i, \overline{\overline{T}}_i)$ then handler $\bar{h}(u)$ may be invoked in the same context.

Item (ii) states that if device u is pending in the context $(\text{imr}_i, \overline{\overline{T}}_i)$, then the context $(\text{imr}_j, \overline{\overline{T}}_j)$, where $j \in J_i^u$, in which handler $\bar{h}(u)$ finishes is also the context in which `iret` starts executing. This is because, before executing `iret`, the handler which corresponds to a pending device may first start executing and then return; therefore, the contexts in which the handler returns should be included in the contexts in which `iret` starts.

Item (iii) guarantees that if there is a pending device u in the context $(\text{imr}_i, \overline{\overline{T}}_i)$, then the additional amount of stack space (stack budget) required by the execution of $\bar{h}(u)$ is limited within the upper bound $K - 1$. This item is crucial in ensuring that the boundedness of the additional stack space (stack growth) used by the call of $\bar{h}(u)$.

Item (iv) makes sure that if there is a pending device u in the context $(\text{imr}_i, \overline{\overline{T}}_i)$ in which case an interrupt that corresponds to device u can occur (Rule (3.10)), then the latency vector after the interrupt is also safe.

Item (2) ensures that the latency vector in the contexts after executing `iret` meets the latency requirement; that is, no latency on each device exceeds its period after executing `iret`.

Rule (3.28) uses the notation: $\forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)} J_i^u\} \cup \{\xi_{\text{iret}}(i)\}$ to define the contexts in which `iret` finishes. Note that J_i is recursively defined. (J_j ranges over a finite lattice, therefore the J_i always has a solution by fixed point construction.)

For any $i \in I$, the notation defines J_i as

1. $\xi_{\text{iret}}(i) \in J_i$. This is because in the context (imr_i, \overline{T}_i) , `iret` can always start its execution and finish in the context $(imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)})$, regardless of whether there are pending devices or not.
2. $J_j \subseteq J_i$, if $j \in \bigcup_{u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)} J_i^u$. This is because if there is a pending device u , then the context (imr_j, \overline{T}_j) in which $\overline{h}(u)$ finishes is also the context in which `iret` starts executing, in which case we have $j \in I$. By definition, J_j , the index set of the contexts in which `iret` finishes, should be included in J_i .

Rule (3.29) is for type checking primitive statements. Rule (3.30) is for type checking branch statements. Rule (3.31) is for type checking sequential statements.

3.4.3 Type soundness

In this subsection, we prove that the type system is sound with respect to the abstract semantics; that is, if a periodic interrupt calculus program type checks, then the program cannot go wrong with respect to the latency-space problem. The proofs are mainly concerned with the type preservation for each step of the abstract semantics.

Type preservation

We first prove some lemmas that will be used in the proof of Lemma (3.4.9).

Lemma 3.4.1 (Safety Guarantee, statements s) *If $\overline{\tau} \vdash_K s : \bigwedge_{i \in I} imr_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{T}_j$, then we have $\text{safe}(\overline{\tau}, I, K)$.*

Proof The proof is by induction on the derivation of $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$. There are three cases depending on which one of the Rules (3.29)-(3.31) is used to derive $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$.

1. Rule (3.29). The proof is trivial in this case because the side condition of Rule (3.29) contains $\text{safe}(\bar{\tau}, I, K)$.
2. Rule (3.30). We have

$$\frac{\begin{array}{l} \bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \bar{T}_j \\ \bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \bar{T}_j \end{array}}{\bar{\tau} \vdash_K \text{if0 } x \text{ then } s_1 \text{ else } s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t \cup J_i^f} \text{imr}_j, \bar{T}_j}$$

From $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \bar{T}_j$ and the induction hypothesis, we have $\text{safe}(\bar{\tau}, I, K)$.

3. Rule (3.31). We have

$$\frac{\begin{array}{l} \bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i'} \text{imr}_j, \bar{T}_j \\ \bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i''} \text{imr}_j, \bar{T}_j \end{array}}{\bar{\tau} \vdash_K s_1; s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}$$

where $J_i = \bigcup \{J_j'' \mid j \in J_i'\}$. From $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i'} \text{imr}_j, \bar{T}_j$ and the induction hypothesis, we have $\text{safe}(\bar{\tau}, I, K)$. ■

Lemma 3.4.2 (Safety Guarantee, handler h) *If $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$, then we have $\text{safe}(\bar{\tau}, I, K)$.*

Proof There are two cases depending on which one of the Rules (3.28) and (3.27) is used to derive $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$,

1. Rule (3.28). The proof is trivial in this case because the side condition of Rule (3.28) contains $\text{safe}(\bar{\tau}, I, K)$.

2. Rule (3.27). We have

$$\frac{\begin{array}{l} \bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j \\ \bar{\tau} \vdash_K h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J''_i} \text{imr}_j, \bar{T}_j \end{array}}{\bar{\tau} \vdash_K s; h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}$$

where $J_i = \bigcup \{J'_j \mid j \in J''_i\}$. From $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$ and Lemma (3.4.1), we have $\text{safe}(\bar{\tau}, I, K)$. ■

Lemma 3.4.3 (Safety Guarantee, main m) *If $\bar{\tau} \vdash_K m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$ then we have $\text{safe}(\bar{\tau}, I, K)$.*

Proof There are two cases depending on which one of the Rules (3.25), (3.26) is used to derive $\bar{\tau} \vdash_K m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$.

1. Rule (3.25). We have

$$\frac{\bar{\tau} \vdash s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}{\bar{\tau} \vdash_K \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i}$$

From $\bar{\tau} \vdash s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Lemma (3.4.1), we have $\text{safe}(\bar{\tau}, I, K)$.

2. Rule (3.26). We have

$$\frac{\begin{array}{l} \bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \\ \bar{\tau} \vdash_K m : \bigvee_{i \in I'} \text{imr}_i, \bar{T}_i \end{array}}{\bar{\tau} \vdash_K s; m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i}$$

From $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Lemma (3.4.1), we have $\text{safe}(\bar{\tau}, I, K)$. ■

Lemma 3.4.4 (K-weakening, Safety) *If $0 \leq K_1 \leq K_2$ and $\text{safe}(\bar{\tau}, I, K_1)$, then we have $\text{safe}(\bar{\tau}, I, K_2)$.*

Proof From Definition (3.4.3), we have $\text{safe}(\bar{\tau}, I, K_1)$ as

$$(1) \quad (i \in I \wedge u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\widehat{T}}_i)) \Rightarrow \begin{cases} (i) & i \in I^u \text{ and} \\ (ii) & J_i^u \subseteq I \text{ and} \\ (iii) & \delta_i^u + 1 \leq K_1 \text{ and} \\ (iv) & \widehat{\theta}_u(\overline{\widehat{T}}_i) \leq \bar{\tau} \end{cases}$$

$$(2) \quad \forall i \in I : \widehat{\theta}(\text{imr}_i, \overline{\widehat{T}}_i) \leq \bar{\tau}$$

From $K_1 \leq K_2$, we have $\delta_i^u + 1 \leq K_2$. Replacing Item (1).(iii) with $\delta_i^u + 1 \leq K_2$, we have

$$(1) \quad (i \in I \wedge u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\widehat{T}}_i)) \Rightarrow \begin{cases} (i) & i \in I^u \text{ and} \\ (ii) & J_i^u \subseteq I \text{ and} \\ (iii) & \delta_i^u + 1 \leq K_2 \text{ and} \\ (iv) & \widehat{\theta}_u(\overline{\widehat{T}}_i) \leq \bar{\tau} \end{cases}$$

$$(2) \quad \forall i \in I : \widehat{\theta}(\text{imr}_i, \overline{\widehat{T}}_i) \leq \bar{\tau}$$

which is $\text{safe}(\bar{\tau}, I, K_2)$. ■

Lemma 3.4.5 (K-weakening, Statement s) *If*

$$\bar{\tau} \vdash_{K_1} s : \bigwedge_{i \in I} \text{imr}_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{\widehat{T}}_j \text{ and } 0 \leq K_1 \leq K_2, \text{ then } \bar{\tau} \vdash_{K_2} s : \bigwedge_{i \in I} \text{imr}_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{\widehat{T}}_j.$$

Proof We prove by induction on the structure of the derivation of $\bar{\tau} \vdash_{K_1} s : \bigwedge_{i \in I} \text{imr}_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{\widehat{T}}_j$. There are three cases depending on which one of the Rules (3.29) - (3.31) are used in the last step of the derivation.

1. Rule (3.29).

We have

$$\bar{\tau} \vdash_{K_1} s : \bigwedge_{i \in I} \text{imr}_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{\widehat{T}}_j$$

and

- (i) $\forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)} J_i^u\} \cup \{\xi_s(i)\}$
- (ii) $\text{safe}(\overline{\tau}, I, K_1)$

where s is $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip . From (ii), $0 \leq K_1 \leq K_2$ and Lemma 3.4.4, we have $\text{safe}(\overline{\tau}, I, K_2)$. From (i), $\text{safe}(\overline{\tau}, I, K_2)$ and Rule (3.29), we have $\overline{\tau} \vdash_{K_2} s : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{T}_j$.

2. Rule (3.30).

We have

$$\frac{\begin{array}{l} \overline{\tau} \vdash_{K_1} s_1 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \overline{T}_j \\ \overline{\tau} \vdash_{K_1} s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \overline{T}_j \end{array}}{\overline{\tau} \vdash_{K_1} \text{if0 } x \text{ then } s_1 \text{ else } s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^t \cup J_i^f} \text{imr}_j, \overline{T}_j}$$

From $\overline{\tau} \vdash_{K_1} s_1 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \overline{T}_j$, $0 \leq K_1 \leq K_2$ and the induction hypothesis, we have

- (i) $\overline{\tau} \vdash_{K_2} s_1 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \overline{T}_j$

Similarly, from $\overline{\tau} \vdash_{K_1} s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \overline{T}_j$ and $0 \leq K_1 \leq K_2$ and the induction hypothesis, we have

- (ii) $\overline{\tau} \vdash_{K_2} s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \overline{T}_j$

From (i), (ii) and Rule (3.30), we have

$$\overline{\tau} \vdash_{K_2} \text{if0 } x \text{ then } s_1 \text{ else } s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i^t \cup J_i^f} \text{imr}_j, \overline{T}_j$$

3. Rule (3.31).

We have

$$\frac{\begin{array}{l} \overline{\tau} \vdash_{K_1} s_1 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{T}_j \\ \overline{\tau} \vdash_{K_1} s_2 : \bigwedge_{i \in I'} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i''} \text{imr}_j, \overline{T}_j \end{array}}{\overline{\tau} \vdash_{K_1} s_1; s_2 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{T}_j}$$

and $\bigcup_{i \in I'} J_i'' \subseteq \bigcup_{i \in I} J_i$ and $\forall i \in I : J_i' \subseteq I'$, where $J_i = \bigcup \{J_j'' \mid j \in J_i'\}$. From $\overline{\tau} \vdash_{K_1} s_1 : \bigwedge_{i \in I} \text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \overline{T}_j$, $0 \leq K_1 \leq K_2$ and the induction hypothesis, we have

$$(i) \quad \bar{\tau} \vdash_{K_2} s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$$

Similarly, from $\bar{\tau} \vdash_{K_1} s_2 : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and $0 \leq K_1 \leq K_2$ and the induction hypothesis, we have

$$(ii) \quad \bar{\tau} \vdash_{K_2} s_2 : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$$

From (i), (ii), $\bigcup_{i \in I'} J'_i \subseteq \bigcup_{i \in I} J_i$, $\forall i \in I : J'_i \subseteq I'$, and Rule (3.31), we have

$$\bar{\tau} \vdash_{K_2} s_1; s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$$

■

Lemma 3.4.6 (K-weakening, Handler h) *If*

$$\bar{\tau} \vdash_{K_1} h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j \text{ and } 0 \leq K_1 \leq K_2, \text{ then } \bar{\tau} \vdash_{K_2} h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j.$$

Proof We prove by induction on the structure of the derivation of $\bar{\tau} \vdash_{K_1} h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$. There are two cases depending on which one of the Rules (3.28), (3.27) are used in the last step of the derivation.

- Rule (3.28).

We have

$$\bar{\tau} \vdash_{K_1} \text{iret} : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$$

and

$$(i) \quad \forall i \in I : J_i = \bigcup \{ J_j \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} J_i^u \} \cup \{ \xi_{\text{iret}}(i) \}$$

$$(ii) \quad \text{safe}(\bar{\tau}, I, K_1)$$

From (ii), $0 \leq K_1 \leq K_2$ and Lemma (3.4.4), we have $\text{safe}(\bar{\tau}, I, K_2)$. From

$$(i), \text{safe}(\bar{\tau}, I, K_2) \text{ and Rule (3.28), we have } \bar{\tau} \vdash_{K_2} \text{iret} : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j.$$

- Rule (3.27).

We have

$$\frac{\begin{array}{l} \bar{\tau} \vdash_{K_1} s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j \\ \bar{\tau} \vdash_{K_1} h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j \end{array}}{\bar{\tau} \vdash_{K_1} s; h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}$$

and $\bigcup_{i \in I'} J'_i \subseteq \bigcup_{i \in I} J_i$ and $\forall i \in I : J'_i \subseteq I'$, where $J_i = \bigcup \{J'_j \mid j \in J'_i\}$. From $\bar{\tau} \vdash_{K_1} s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$, $0 \leq K_1 \leq K_2$ and Lemma (3.4.5), we have

$$(i) \quad \bar{\tau} \vdash_{K_2} s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$$

From $\bar{\tau} \vdash_{K_1} h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$ and $0 \leq K_1 \leq K_2$ and the induction hypothesis, we have

$$(ii) \quad \bar{\tau} \vdash_{K_2} h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$$

From (i), (ii), $\bigcup_{i \in I'} J'_i \subseteq \bigcup_{i \in I} J_i$, $\forall i \in I : J'_i \subseteq I'$ and Rule (3.27), we have $\bar{\tau} \vdash_{K_2} s; h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in \bigcup \{J'_j \mid j \in J'_i\}} \text{imr}_j, \bar{T}_j$.

■

Lemma 3.4.7 (J-weakening, statement) *If*

$\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$, $i' \in I$ and $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, then $i'' \in J_{i'}^u \Rightarrow (i'' \in I \wedge J_{i''} \subseteq J_{i'})$.

Proof We prove by induction on the structure of the derivation of s . There are three cases depending upon the form of s .

1. s is either of $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or **skip**.

From $\bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.29), we have $\text{safe}(\bar{\tau}, I, K)$ and

$$(i) \quad \forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} J_i^u\} \cup \{\xi_s(i)\}$$

From $\text{safe}(\bar{\tau}, I, K)$, $i' \in I$, $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, and Item (1).(ii) of Definition (3.4.3), we have $J_{i'}^u \subseteq I$, from which it follows $i'' \in J_{i'}^u \Rightarrow i'' \in I$.

From (i), $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, it follows $i'' \in J_{i'}^u \Rightarrow J_{i''} \subseteq J_{i'}$.

2. s is if0 x then s_1 else s_2 .

From $\bar{\tau} \vdash_K \text{if0 } x \text{ then } s_1 \text{ else } s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and

Rule (3.30), we have

- (i) $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \bar{T}_j$
- (ii) $\bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \bar{T}_j$
- (iii) $\forall i \in I : J_i = J_i^t \cup J_i^f$

From (i), $i' \in I$, $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$ and the induction hypothesis, we have

- (iv) $i'' \in J_{i'}^u \Rightarrow (i'' \in I \wedge J_{i''}^t \subseteq J_{i'})$

Similarly, from (ii), $i' \in I$, $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$ and the induction hypothesis, we have

- (v) $i'' \in J_{i'}^u \Rightarrow (i'' \in I \wedge J_{i''}^f \subseteq J_{i'})$

From (iv), (v) and (iii), we have $i'' \in J_{i'}^u \Rightarrow (i'' \in I \wedge J_{i''} \subseteq J_{i'})$.

3. s is $s_1; s_2$.

From $\bar{\tau} \vdash_K s_1; s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.31), we have

- (i) $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i'} \text{imr}_j, \bar{T}_j$
- (ii) $\bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i''} \text{imr}_j, \bar{T}_j$
- (iii) $\forall i \in I : J_i' \subseteq I'$
- (iv) $\forall i \in I : J_i = \bigcup \{J_j'' \mid j \in J_i'\}$

From (i), $i' \in I$ and $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$ and the induction hypothesis, we have

- (v) $i'' \in J_{i'}^u \Rightarrow i'' \in I$
- (vi) $i'' \in J_{i'}^u \Rightarrow J_{i''}' \subseteq J_{i'}$

From $i' \in I$ and (iv), we have

- (vii) $J_{i'} = \bigcup \{J_j'' \mid j \in J_{i'}'\}$

From (v) and (iv), we have

- (viii) $i'' \in J_{i'}^u \Rightarrow J_{i''} = \bigcup \{J_j'' \mid j \in J_{i''}'\}$

From (vi), (vii) and (viii), it follows $i'' \in J_{i'}^u \Rightarrow J_{i''} \subseteq J_{i'}$.

■

Lemma 3.4.8 (J-weakening, handler) *If*

$\bar{\tau} \vdash_K h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$, $i' \in I$ and $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, then $i'' \in J_{i'}^u \Rightarrow (i'' \in I \wedge J_{i''} \subseteq J_{i'})$.

Proof We prove by induction on the structure of the derivation of h . There are two cases depending upon the form of h .

1. h is *iret*.

From $\bar{\tau} \vdash_K \text{iret} : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.28), we have $\text{safe}(\bar{\tau}, I, K)$ and

$$(i) \quad \forall i \in I : J_i = \bigcup \{J_j \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} J_i^u\} \cup \{\xi_{\text{iret}}(i)\}$$

From $\text{safe}(\bar{\tau}, I, K)$, $i' \in I$, $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, and Item (1).(ii) of Definition (3.4.3), we have $J_{i'}^u \subseteq I$, from which it follows $i'' \in J_{i'}^u \Rightarrow i'' \in I$.

From (i), $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$, it follows $i'' \in J_{i'}^u \Rightarrow J_{i''} \subseteq J_{i'}$.

2. h is $s; h'$. From $\bar{\tau} \vdash_K s; h' : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.27),

we have

$$(i) \quad \bar{\tau} \vdash_K s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$$

$$(ii) \quad \bar{\tau} \vdash_K h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J''_i} \text{imr}_j, \bar{T}_j$$

$$(iii) \quad \forall i \in I : J'_i \subseteq I'$$

$$(iv) \quad \forall i \in I : J_i = \bigcup \{J''_j \mid j \in J'_i\}$$

From (i), $i' \in I$ and $u \in \widehat{\mathcal{P}}(\text{imr}_{i'}, \bar{T}_{i'})$ and Lemma (3.4.7), we have

$$(v) \quad i'' \in J_{i'}^u \Rightarrow i'' \in I$$

$$(vi) \quad i'' \in J_{i'}^u \Rightarrow J_{i''} \subseteq J'_{i'}$$

From $i' \in I$ and (iv), we have

$$(vii) \quad J_{i'} = \bigcup \{J''_j \mid j \in J'_{i'}\}$$

From (v) and (iv), we have

$$(viii) \quad i'' \in J_{i'}^u \Rightarrow J_{i''} = \bigcup \{J''_j \mid j \in J'_{i''}\}$$

From (vi), (vii) and (viii), it follows $i'' \in J_{i'}^u \Rightarrow J_{i''} \subseteq J_{i'}$.

■

Lemma 3.4.9 (Single Step Type Preservation) *Given $K \geq 0$ and Q is consistent, if $\bar{\tau} \vdash_K Q$, $Q.\bar{T} \leq \bar{\tau}$, $Q \hookrightarrow Q'$, then there exists a $K' \geq 0$ such that $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk$ and $Q'.\bar{T} \leq \bar{\tau}$.*

Proof We have nine cases depending upon which one of the Rules (3.10) - (3.18) is used.

1. Rule (3.10).

We have $\langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, a \rangle \hookrightarrow \langle \bar{h}, imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')}, a :: \sigma, \bar{h}(v) \rangle$ and $v \in \hat{\mathcal{P}}(imr_{i'}, \bar{T}_{i'})$. Since Q is consistent, there are two subcases.

Subcase 1: $Q = \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \text{nil}, m \rangle$ and $Q' = \langle \bar{h}, imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')}, m :: \text{nil}, \bar{h}(v) \rangle$.

From $\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \text{nil}, m \rangle$ and Rule (3.20), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K m : \bigvee_{i \in I} imr_i, \bar{T}_i}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, h \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K m : \bigvee_{i \in I} imr_i, \bar{T}_i$ and Lemma (3.4.3), we have $safe(\bar{\tau}, I, K)$. From $v \in \hat{\mathcal{P}}(imr_{i'}, \bar{T}_{i'})$, $safe(\bar{\tau}, I, K)$ and Item (1) of Definition (3.4.3), we have

- (i) $i' \in I^v$
- (ii) $J_{i'}^v \subseteq I$
- (iii) $\delta_{i'}^v + 1 \leq K$
- (iv) $\hat{\theta}_v(\bar{T}_{i'}) \leq \bar{\tau}$

From (i), $\bar{\tau} \vdash \bar{h} : \bar{\tau}$ and Rule (3.24), we have

$$(v) \quad \bar{\tau} \vdash_{\delta_{i'}^v} \bar{h}(v) : imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')} \rightarrow \bigvee_{j \in J_{i'}^v} imr_j, \bar{T}_j$$

From $\delta_{i'}^v \geq 0$ and (iii) above, we have $0 \leq \delta_{i'}^v \leq K - 1$. From (v) above, $0 \leq \delta_{i'}^v \leq K - 1$ and Lemma (3.4.6), we have

$$(vi) \quad \bar{\tau} \vdash_{K-1} \bar{h}(v) : imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')} \rightarrow \bigvee_{j \in J_{i'}^v} imr_j, \bar{T}_j$$

From (ii), we have

$$(vii) \quad \forall i'' \in J_{i'}^v : i'' \in I$$

From (vii), $\bar{\tau} \vdash_K m : \bigvee_{i \in I} imr_i, \bar{T}_i$ and Rule (3.22), we have

$$(viii) \quad \forall i'' \in J_{i'}^v : \bar{\tau}, i'' \vdash_K m :: \text{nil}$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, (viii), (vi) and Rule (3.21), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk = K - 1 \geq 0$. From (iv), we have $\bar{T}_{\zeta_v(i')} \leq \bar{r}$, which is $Q'.\bar{T} \leq \bar{r}$.

Subcase 2: $Q = \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, h \rangle$ and $Q' = \langle \bar{h}, imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')}, h :: \sigma, \bar{h}(v) \rangle$.
From $\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, h \rangle$ and Rule (3.21), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma \quad \bar{\tau} \vdash_K h : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{T}_{i'}, \sigma, h \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j$ and Lemma (3.4.2), we have $safe(\bar{\tau}, I, K)$. From $v \in \widehat{\mathcal{P}}(imr_{i'}, \bar{T}_{i'})$, $safe(\bar{\tau}, I, K)$ and Item (1) of Definition (3.4.3), we have

- (i) $i' \in I^v$
- (ii) $J_{i'}^v \subseteq I$
- (iii) $\delta_{i'}^v + 1 \leq K$
- (iv) $\widehat{\theta}_v(\bar{T}_{i'}) \leq \bar{r}$

From (i), $\bar{\tau} \vdash \bar{h} : \bar{\tau}$ and Rule (3.24), we have

$$(v) \quad \bar{\tau} \vdash_{\delta_{i'}^v} \bar{h}(v) : imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')} \rightarrow \bigvee_{j \in J_{i'}^v} imr_j, \bar{T}_j$$

From $\delta_{i'}^v \geq 0$ and (iii), we have $0 \leq \delta_k^v \leq K - 1$. From (v), $0 \leq \delta_k^v \leq K - 1$ and Lemma (3.4.6), we have

$$(vi) \quad \bar{\tau} \vdash_{K-1} \bar{h}(v) : imr_{\zeta_v(i')}, \bar{T}_{\zeta_v(i')} \rightarrow \bigvee_{j \in J_{i'}^v} imr_j, \bar{T}_j$$

From $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j$, $i' \in I$, $v \in \widehat{\mathcal{P}}(imr_{i'}, \bar{T}_{i'})$ and Lemma (3.4.8), we have

- (vii) $i''' \in J_{i'}^v \Rightarrow i''' \in I$
- (viii) $i''' \in J_{i'}^v \Rightarrow J_{i'''} \subseteq J_{i'}$

From $\forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma$ and (viii), we have

$$(ix) \quad \forall i''' \in J_{i'}^v : \forall i'' \in J_{i'''} : \bar{\tau}, i'' \vdash_K \sigma$$

From $\bar{\tau} \vdash_K h : \bigwedge_{i \in I} imr_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{T}_j$, (ix), (vii) and Rule (3.23), we have

$$(x) \quad \forall i''' \in J_{i'}^v : \bar{\tau}, i''' \vdash_K h :: \sigma$$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, (x), (vi) and Rule (3.21), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk = K - 1 \geq 0$. From (iv), we have $\overline{\widehat{T}}_{\zeta_v(i')} \leq \bar{\tau}$, which is $Q'.\overline{\widehat{T}} \leq \bar{\tau}$.

2. Rule (3.11).

We have $Q = \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, \sigma, iret \rangle \leftrightarrow \langle \bar{h}, imr_{\xi_{iret}(i')}, \overline{\widehat{T}}_{\xi_{iret}(i')}, \sigma', a \rangle$, where $\sigma = a :: \sigma'$. Since Q is consistent, there are two subcases.

Subcase 1: $Q = \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, m :: nil, iret \rangle$ and $Q' = \langle \bar{h}, imr_{\xi_{iret}(i')}, \overline{\widehat{T}}_{\xi_{iret}(i')}, nil, m \rangle$.

From $\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, h :: \sigma, iret \rangle$ and Rule (3.21), we have

$$\frac{\begin{array}{l} \bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K m :: nil \\ \bar{\tau} \vdash_K iret : \bigwedge_{i \in I} imr_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{\widehat{T}}_j \end{array}}{\vdash_K \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, m :: nil, iret \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K iret : \bigwedge_{i \in I} imr_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{\widehat{T}}_j$ and Rule (3.28),

we have

$$(i) \quad \forall i \in I : \xi_{iret}(i) \in J_i$$

$$(ii) \quad \forall i \in I : \widehat{\theta}(imr_i, \overline{\widehat{T}}_i) \leq \bar{\tau} \text{ (from Item (2) of } safe(\bar{\tau}, I, K))$$

From $i' \in I$ and (i), we have $\xi_{iret}(i') \in J_{i'}$. From $\forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K m :: nil$

and Rule (3.22), we have

$$(iii) \quad \bar{\tau} \vdash_{K+1} m : \bigvee_{i \in I'} imr_i, \overline{\widehat{T}}_i$$

$$(iv) \quad \forall i'' \in J_{i'} : i'' \in I'$$

From $\xi_{iret}(i') \in J_{i'}$ and (iv), we have $\xi_{iret}(i') \in I'$. From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $\xi_{iret}(i') \in I'$,

(iii) and Rule (3.20), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk = K + 1 \geq 0$.

From $i' \in I$ and (ii), we have $\overline{\widehat{T}}_{\xi_{iret}(i')} \leq \bar{\tau}$, which is $Q'.\overline{\widehat{T}} \leq \bar{\tau}$.

Subcase 2: $Q = \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, h :: \sigma, iret \rangle$ and $Q' = \langle \bar{h}, imr_{\xi_{iret}(i')}, \overline{\widehat{T}}_{\xi_{iret}(i')}, \sigma, h \rangle$.

From $\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, h :: \sigma, iret \rangle$ and Rule (3.21), we have

$$\frac{\begin{array}{l} \bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K h :: \sigma \\ \bar{\tau} \vdash_K iret : \bigwedge_{i \in I} imr_i, \overline{\widehat{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \overline{\widehat{T}}_j \end{array}}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \overline{\widehat{T}}_{i'}, h :: \sigma, iret \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K \text{iret} : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.28), we have

- (i) $\forall i \in I : \xi_{\text{iret}}(i) \in J_i$
- (ii) $\forall i \in I : \widehat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{\tau}$ (from Item (2) of $\text{safe}(\bar{\tau}, I, K)$)

From $i' \in I$ and (i), we have $\xi_{\text{iret}}(i') \in J_{i'}$. From $\forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K h :: \sigma$ and Rule (3.23), we have

- (iii) $\bar{\tau} \vdash_{K+1} h : \bigwedge_{i \in I'} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$
- (iv) $\forall i'' \in J_{i'} : \forall i''' \in J'_{i''} : \bar{\tau}, i''' \vdash_{K+1} \sigma$
- (v) $\forall i'' \in J_{i'} : i'' \in I'$

From $\xi_{\text{iret}}(i') \in J_{i'}$ and (iv), we have

- (vi) $\forall i''' \in J'_{\xi_{\text{iret}}(i')} : \bar{\tau}, i''' \vdash_{K+1} \sigma$

From $\xi_{\text{iret}}(i') \in J_{i'}$ and (v), we have $\xi_{\text{iret}}(i') \in I'$. From $\bar{\tau} \vdash \bar{h} : \bar{\tau}, \xi_{\text{iret}}(i') \in I'$, (vi), (iii) and Rule (3.21), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.\text{stk} - Q'.\text{stk} = K + 1 \geq 0$. From $i' \in I$ and (ii), we have $\bar{T}_{\xi_{\text{iret}}(i')} \leq \bar{\tau}$, which is $Q'.\bar{T} \leq \bar{\tau}$.

3. Rule (3.12).

We have $Q = \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, \text{loop } s \rangle$, $Q' = \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, s; \text{loop } s \rangle$ and $Q \hookrightarrow Q'$. From $Q.\bar{T} \leq \bar{\tau}$ and $Q'.\bar{T} = Q.\bar{T}$, we have $Q'.\bar{T} \leq \bar{\tau}$. From $\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, \text{loop } s \rangle$ and Rule (3.20), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i}{\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, \text{loop } s \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$ and Rule (3.25), we have

- (i) $\bar{\tau} \vdash s : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$
- (ii) $\forall i \in I : J_i \subseteq I$

From (i), (ii), $\bar{\tau} \vdash_K \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$ and Rule (3.26), we have

- (iii) $\bar{\tau} \vdash_K s; \text{loop } s : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$

From (iii), $\bar{\tau} \vdash \bar{h} : \bar{\tau}, i' \in I$ and Rule (3.20), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.\text{stk} - Q'.\text{stk} = K \geq 0$.

4. Rule (3.13).

We have $\langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, (x := e); a \rangle \hookrightarrow \langle \bar{h}, \text{imr}_{\xi_{x:=e}(i')}, \bar{T}_{\xi_{x:=e}(i')}, \sigma, a \rangle$. Since Q is consistent, there are two subcases.

Subcase 1: $Q = \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, (x := e); m \rangle$ and

$Q' = \langle \bar{h}, \text{imr}_{\xi_{x:=e}(i')}, \bar{T}_{\xi_{x:=e}(i')}, \text{nil}, m \rangle$. From $\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, (x := e); m \rangle$ and Rule (3.20), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K (x := e); m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i}{\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \text{nil}, x := e; m \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K (x := e); m : \bigvee_{i \in I} \text{imr}_i, \bar{T}_i$ and Rule (3.26), we have

- (i) $\bar{\tau} \vdash_K x := e : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$
- (ii) $\bar{\tau} \vdash_K m : \bigvee_{i \in I'} \text{imr}_i, \bar{T}_i$
- (iii) $\forall i \in I : J_i \subseteq I'$

From (i) and Rule (3.29), we have

- (iv) $\forall i \in I : \xi_{x:=e}(i) \in J'_i$
- (v) $\forall i \in I : \hat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{\tau}$ (from Item (2) of $\text{safe}(\bar{\tau}, I, K)$)

From $i' \in I$ and (iv), we have $\xi_{x:=e}(i') \in J'_{i'}$. From $i' \in I$ and (iii), we have $J'_{i'} \subseteq I'$. From $\xi_{x:=e}(i') \in J'_{i'}$ and $J'_{i'} \subseteq I'$, we have $\xi_{x:=e}(i') \in I'$. From (ii), $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, $\xi_{x:=e}(i') \in I'$ and Rule (3.20), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.\text{stk} - Q'.\text{stk} = K \geq 0$. From $i' \in I$ and (v), we have $\bar{T}_{\xi_{x:=e}(i')} \leq \bar{\tau}$, which is $Q'.\bar{T} \leq \bar{\tau}$.

Subcase 2: $Q = \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, (x := e); h \rangle$ and

$Q' = \langle \bar{h}, \text{imr}_{\xi_{x:=e}(i')}, \bar{T}_{\xi_{x:=e}(i')}, \sigma, h \rangle$. From $\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, (x := e); h \rangle$ and Rule (3.21), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma \quad \bar{\tau} \vdash_K (x := e); h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j}{\bar{\tau} \vdash_K \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, x := e; h \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K (x := e); h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i} \text{imr}_j, \bar{T}_j$ and Rule (3.27), we have

- (i) $\bar{\tau} \vdash_K x := e : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J'_i} \text{imr}_j, \bar{T}_j$
- (ii) $\bar{\tau} \vdash_K h : \bigwedge_{i \in I''} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J''_i} \text{imr}_j, \bar{T}_j$
- (iii) $\forall i \in I : J'_i \subseteq I''$
- (iv) $\forall i \in I : J_i = \bigcup \{J''_j \mid j \in J'_i\}$

From (i) and Rule (3.29), we have

- (v) $\forall i \in I : \xi_{x:=e}(i) \in J'_i$
- (vi) $\forall i \in I : \hat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{\tau}$ (from Item (2) of $\text{safe}(\bar{\tau}, I, K)$)

From $i' \in I$ and (v), we have $\xi_{x:=e}(i') \in J'_{i'}$. From $i' \in I$ and (iii), we have $J'_{i'} \in I''$. From $\xi_{x:=e}(i') \in J'_{i'}$ and $J'_{i'} \in I''$ we have $\xi_{x:=e}(i') \in I''$. From $i' \in I$ and (iv), we have

- (vii) $J_{i'} = \bigcup \{J''_j \mid j \in J'_{i'}\}$

From $\xi_{x:=e}(i') \in J'_{i'}$ and (vii), we have

- (viii) $J''_{\xi_{x:=e}(i')} \subseteq J_{i'}$

From $\forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma$ and (viii) we have

- (ix) $\forall i'' \in J''_{\xi_{x:=e}(i')} : \bar{\tau}, i'' \vdash_K \sigma$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}, \xi_{x:=e}(i) \in I''$, (ix), (ii) and Rule (3.21), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.\text{stk} - Q'.\text{stk} = K \geq 0$. From $i' \in I$ and (vi), we have $\bar{T}_{\xi_{x:=e}(i')} \leq \bar{\tau}$, which is $Q'.\bar{T} \leq \bar{\tau}$.

5. Rule (3.14),(3.15) and (3.18).

The proofs are similar to that of Rule (3.13).

6. Rule (3.16).

We have $\langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a \rangle \leftrightarrow \langle \bar{h}, \text{imr}_{i'}, \bar{T}_{i'}, \sigma, s_1; a \rangle$.

From $Q.\bar{T} \leq \bar{\tau}$ and $Q'.\bar{T} = Q.\bar{T}$, we have $Q'.\bar{T} \leq \bar{\tau}$. We next prove that there exists a $K' \geq 0$ such that $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.\text{stk} - Q'.\text{stk}$. Since Q is consistent, there are two subcases.

Subcase 1: $Q = \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, nil, (if0\ x\ then\ s_1\ else\ s_2); m \rangle$ and
 $Q' = \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, nil, s_1; m \rangle$.

From $\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, nil, (if0\ x\ then\ s_1\ else\ s_2); m \rangle$ and Rule (3.20), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \bar{\tau} \vdash_K (if0\ x\ then\ s_1\ else\ s_2); m : \bigvee_{i \in I} imr_i, \bar{\bar{T}}_i}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, nil, (if0\ x\ then\ s_1\ else\ s_2); m \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K (if0\ x\ then\ s_1\ else\ s_2); m : \bigvee_{i \in I} imr_i, \bar{\bar{T}}_i$ and Rule (3.26), we have

- (i) $\bar{\tau} \vdash_K if0\ x\ then\ s_1\ else\ s_2 : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{\bar{T}}_j$
- (ii) $\bar{\tau} \vdash_K m : \bigvee_{i \in I'} imr_i, \bar{\bar{T}}_i$

and $\forall i \in I : J_i \subseteq I'$. From (i) and Rule (3.30), we have

- (iii) $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i^t} imr_j, \bar{\bar{T}}_j$
- (iv) $\bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i^f} imr_j, \bar{\bar{T}}_j$

and $\forall i \in I : J_i = J_i^t \cup J_i^f$, from which it follows $\forall i \in I : J_i^t \subseteq J_i$. From $\forall i \in I : J_i \subseteq I'$ and $\forall i \in I : J_i^t \subseteq J_i$, we have $\forall i \in I : J_i^t \subseteq I'$. From (iii), (ii), $\forall i \in I : J_i^t \subseteq I'$ and Rule (3.26), we have

- (v) $\bar{\tau} \vdash_K s_1; m : \bigvee_{i \in I} imr_i, \bar{\bar{T}}_i$

From $i' \in I$, $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, (v) and Rule (3.20), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk = K \geq 0$.

Subcase 2: $Q = \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, \sigma, (if0\ x\ then\ s_1\ else\ s_2); h \rangle$ and
 $Q' = \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, \sigma, s_1; h \rangle$. Rule (3.21), we have

$$\frac{\bar{\tau} \vdash \bar{h} : \bar{\tau} \quad \forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma \quad \bar{\tau} \vdash_K (if0\ x\ then\ s_1\ else\ s_2); h : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{\bar{T}}_j}{\bar{\tau} \vdash_K \langle \bar{h}, imr_{i'}, \bar{\bar{T}}_{i'}, \sigma, (if0\ x\ then\ s_1\ else\ s_2); h \rangle}$$

and $i' \in I$. From $\bar{\tau} \vdash_K (if0\ x\ then\ s_1\ else\ s_2); h : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{\bar{T}}_j$ and Rule (3.27), we have

- (i) $\bar{\tau} \vdash_K if0\ x\ then\ s_1\ else\ s_2 : \bigwedge_{i \in I} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i} imr_j, \bar{\bar{T}}_j$
- (ii) $\bar{\tau} \vdash_K h : \bigwedge_{i \in I''} imr_i, \bar{\bar{T}}_i \rightarrow \bigvee_{j \in J_i''} imr_j, \bar{\bar{T}}_j$
- (iii) $\forall i \in I : J_i' \subseteq I''$
- (iv) $\forall i \in I : J_i = \bigcup \{J_j'' \mid j \in J_i'\}$

From (i) and Rule (3.30), we have

- (v) $\bar{\tau} \vdash_K s_1 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^t} \text{imr}_j, \bar{T}_j$
- (vi) $\bar{\tau} \vdash_K s_2 : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i^f} \text{imr}_j, \bar{T}_j$
- (vii) $\forall i \in I : J_i' = J_i^t \cup J_i^f$

From (vii), we have

- (viii) $\forall i \in I : J_i^t \subseteq J_i'$

From (viii) and (iii), we have

- (ix) $\forall i \in I : J_i^t \subseteq I''$

Let

- (x) $\forall i \in I : J_i''' = \bigcup \{J_j'' \mid j \in J_i^t\}$

From (v), (ii), (ix), (x) and Rule (3.27), we have

- (xi) $\bar{\tau} \vdash_K s_1; h : \bigwedge_{i \in I} \text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in J_i'''} \text{imr}_j, \bar{T}_j$

From (iv), (x) and (viii), we have

- (xii) $\forall i \in I : J_i''' \subseteq J_i$

From $i' \in I$ and (xii), we have $J_{i'}''' \subseteq J_{i'}$. From $\forall i'' \in J_{i'} : \bar{\tau}, i'' \vdash_K \sigma$ and $J_{i'}''' \subseteq J_{i'}$, we have

- (xiii) $\forall i'' \in J_{i'}''' : \bar{\tau}, i'' \vdash_K \sigma$

From $\bar{\tau} \vdash \bar{h} : \bar{\tau}$, (xiii), (xi), $i' \in I$ and Rule (3.21), we have $\bar{\tau} \vdash_{K'} Q'$ and $K' = K + Q.stk - Q'.stk = K \geq 0$.

7. Rule (3.17).

The proof is similar to that of Rule (3.16). ■

Lemma 3.4.10 (Multiple Step Type Preservation) *Given $K \geq 0$ and Q is consistent, if $\bar{\tau} \vdash_K Q$, $Q.\bar{T} \leq \bar{\tau}$, $Q \hookrightarrow^n Q'$, then there exists $K' \geq 0$ such that $\bar{\tau} \vdash_{K'} Q'$, $K' = K + Q.stk - Q'.stk$ and $Q'.\bar{T} \leq \bar{\tau}$.*

Proof We proceed by induction on n .

In the base case: $n = 0$, we have $Q = Q'$. The proof is trivial.

In the induction step, assume that the property holds true for n . Suppose $\bar{\tau} \vdash_K Q$,

$K \geq 0$, $Q.\overline{\overline{T}} \leq \overline{r}$ and $Q \rightarrow^n Q' \rightarrow Q''$. From the induction hypothesis, we have $\overline{r} \vdash_{K'} Q'$ and $K' \geq 0$, where $K' = K + |Q.stk| - |Q'.stk|$ and $Q'.\overline{\overline{T}} \leq \overline{r}$. From Lemma 3.3.1 we have that Q' is consistent. From $K' \geq 0$, Q' is consistent, $\overline{r} \vdash_{K'} Q'$, $Q'.\overline{\overline{T}} \leq \overline{r}$, $Q' \hookrightarrow Q''$ and Lemma 3.4.9, we have $\overline{r} \vdash_{K''} Q''$ and $K'' \geq 0$, where $K'' = K' + |Q'.stk| - |Q''.stk|$ and $Q''.\overline{\overline{T}} \leq \overline{r}$. From $K' = K + |Q.stk| - |Q'.stk|$ and $K'' = K' + |Q'.stk| - |Q''.stk|$, we have

$$\begin{aligned} K'' &= K' + |Q'.stk| - |Q''.stk| \\ &= K + |Q.stk| - |Q'.stk| + |Q'.stk| - |Q''.stk| \\ &= K + |Q.stk| - |Q''.stk| \end{aligned}$$

as desired. ■

Corollary 3.4.1 (Latency-Space Boundedness) *Given $K \geq 0$, if $\overline{r} \vdash_K Q_0$ and $Q_0 \hookrightarrow^* Q$, then $(Q.\overline{\overline{T}} \leq \overline{r}) \wedge (Q.stk \leq K)$.*

Proof Notice that Q_0 is consistent and $Q_0.\overline{\overline{T}} \leq \overline{r}$. From $K \geq 0$, $Q_0.\overline{\overline{T}} \leq \overline{r}$, $\overline{r} \vdash_K Q_0$, $Q_0 \hookrightarrow^* Q$ and Lemma (3.4.10), there exists $K' \geq 0$, $\overline{r} \vdash_K Q$ and $K' = K + Q_0.stk - Q.stk$ and $Q.\overline{\overline{T}} \leq \overline{r}$. From $Q_0.stk = 0$ and $K' = K + Q_0.stk - Q.stk$, we have $K' = K - Q.stk$. From $K' \geq 0$ and $K' = K - Q.stk$, we have $Q.stk \leq K$. ■

Theorem 3.4.1 *Given $K \geq 0$, if there exists \overline{r} such that $\overline{r} \vdash_K p$, then $Q_0 \hookrightarrow^* Q \Rightarrow (Q.\overline{\overline{T}} \leq \overline{r}) \wedge (Q.stk \leq K)$.*

Proof From $\overline{r} \vdash_K p$ and Rule (3.19), we have

- (i) $\overline{r} \vdash_K m : 0, \overline{0}$
- (ii) $\overline{r} \vdash \overline{h} : \overline{r}$

From (i), (ii) and Rule (3.20), we have $\overline{r} \vdash_K Q_0$. From $K \geq 0$ and $\overline{r} \vdash_K Q_0$ and $Q_0 \hookrightarrow^* Q$ and Corollary (3.4.1), we have $(Q.\overline{\overline{T}} \leq \overline{r}) \wedge (Q.stk \leq K)$. ■

3.5 An example

In this section, we illustrate how we construct types from an abstract model by running a simple example.

Consider the interrupt calculus program given in Figure (3.4). The main part of the program first enables both interrupt sources; it then enters an infinite loop. handler 1 enables the master bit of imr and then returns; handler 2 performs similar operations. The period for handler 1 is 4 time units; and the period for handler 2 is 20 time units. For the sake of convenience, we label each program statement by $m^0, m^1, h_1^0, h_1^1, h_2^0, h_2^1$. We will use these labels to represent the statements in the reachable states of the abstract model given in Figure (3.5).

Main:	Handler 1: $\bar{r}(1) = 4$	Handler 2: $\bar{r}(2) = 20$
m^0 : $imr = imr \vee 111$	h_1^0 : $imr = imr \vee 100$	h_2^0 : $imr = imr \vee 100$
m^1 : loop skip	h_1^1 : iret	h_2^1 : iret

Figure 3.4. The example program

Figure (3.5) shows an excerpt of all reachable states from the abstract model of the example program. The notation \xrightarrow{n} denotes a transition which uses rule n . Note that we only give the transition paths which entail the largest stack growth, with one exception for the transitions (3.48-3.53) which include all the possible paths of executing $h_1^0; h_1^1$ starting from the state $\langle 011, -1, -17, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle$. We will use transitions (3.48-3.53) to illustrate the construction of the types for handler 1.

Figure (3.6) illustrates the a segment of the growth of the stack size over the time of the example program following a path given in Figure (3.5) (The path is: 3.45-3.48, 3.50-3.57). Note that the stack size can grow as high as 3, rather than 2, following this path.

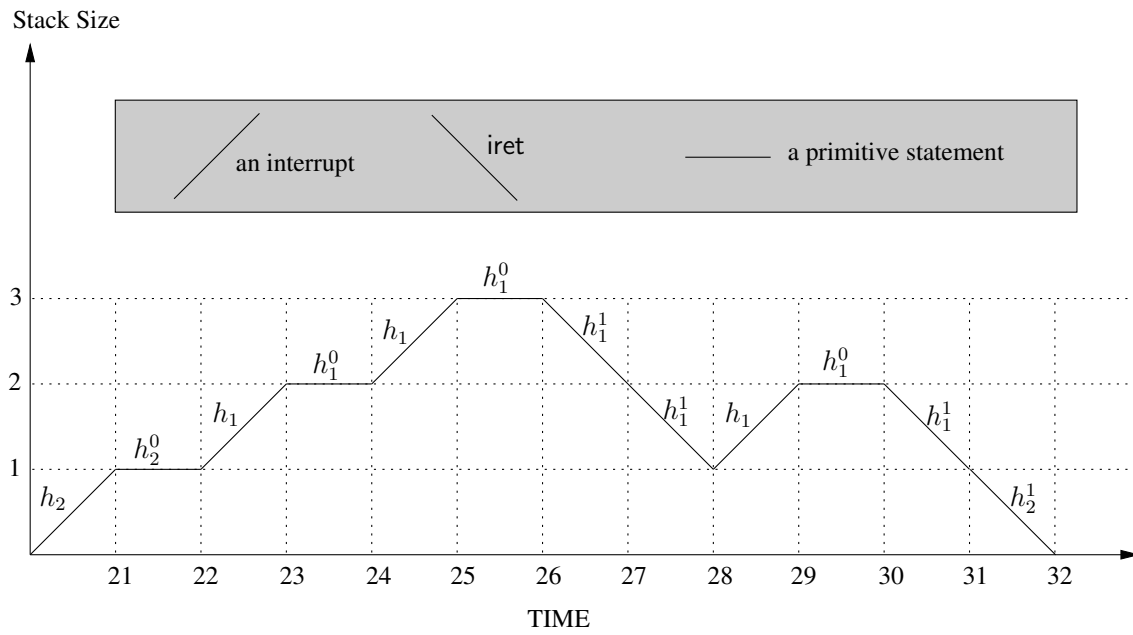
Since abstract states always carry \bar{h} , we omit it for the sake of simplicity. In addition, to make the presentation more clear, we assume $\bar{h}(0) = p.m$ in this section.

We now show how to construct the types for handlers \bar{r} . For each program point a and stack σ , we compute the context set $I^{\sigma,a}$ which contains the contexts that are reachable from the initial program state; we compute the context set $I_{imr, \bar{T}, u}^{\sigma,a}$ which contains the contexts that are reachable from the state $\langle imr, \bar{T}, \sigma, \bar{h}(u) \rangle$; we

$$\begin{aligned}
\langle 000, 0, 0, \text{nil}, m^0; m^1 \rangle &\hookrightarrow^* \langle 111, 0, 0, \text{nil}, m^1 \rangle & (3.32) \\
\langle 111, 0, 0, \text{nil}, m^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -3, 1, m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.33) \\
\langle 011, -3, 1, m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, -2, 2, m^1 :: \text{nil}, h_1^1 \rangle & (3.34) \\
\langle 111, -2, 2, m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -1, -17, h_1^1 :: m^1 :: \text{nil}, h_2^0; h_2^1 \rangle & (3.35) \\
\langle 011, -1, -17, h_1^1 :: m^1 :: \text{nil}, h_2^0; h_2^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, 0, -16, h_1^1 :: m^1 :: \text{nil}, h_2^1 \rangle & (3.36) \\
\langle 111, 0, -16, h_1^1 :: m^1 :: \text{nil}, h_2^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -3, -15, h_2^1 :: h_1^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.37) \\
\langle 011, -3, -15, h_2^1 :: h_1^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, -2, -14, h_2^1 :: h_1^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.38) \\
\langle 111, -2, -14, h_2^1 :: h_1^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, -1, -13, h_1^1 :: m^1 :: \text{nil}, h_2^1 \rangle & (3.39) \\
\langle 111, -1, -13, h_1^1 :: m^1 :: \text{nil}, h_2^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, 0, -12, m^1 :: \text{nil}, h_1^1 \rangle & (3.40) \\
\langle 111, 0, -12, m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -3, -11, h_1^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.41) \\
\langle 011, -3, -11, h_1^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, -2, -10, h_1^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.42) \\
\langle 111, -2, -10, h_1^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, -1, -9, m^1 :: \text{nil}, h_1^1 \rangle & (3.43) \\
\langle 111, -1, -9, m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, 0, -8, \text{nil}, m^1 \rangle & (3.44) \\
\langle 111, 0, 0, \text{nil}, m^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, 1, -19, m^1 :: \text{nil}, h_2^0; h_2^1 \rangle & (3.45) \\
\langle 011, 1, -19, m^1 :: \text{nil}, h_2^0; h_2^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, 2, -18, m^1 :: \text{nil}, h_2^1 \rangle & (3.46) \\
\langle 111, 2, -18, m^1 :: \text{nil}, h_2^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -1, -17, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.47) \\
\langle 011, -1, -17, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, 0, -16, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.48) \\
\langle 111, 0, -16, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, 0, -15, m^1 :: \text{nil}, h_2^1 \rangle & (3.49) \\
\langle 111, 0, -16, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -3, -15, h_1^1 :: h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.50) \\
\langle 011, -3, -15, h_1^1 :: h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, -2, -14, h_1^1 :: h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.51) \\
\langle 111, -2, -14, h_1^1 :: h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, -1, -13, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.52) \\
\langle 111, -1, -13, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, 0, -12, m^1 :: \text{nil}, h_2^1 \rangle & (3.53) \\
\langle 111, 0, -12, m^1 :: \text{nil}, h_2^1 \rangle &\stackrel{3.10}{\hookrightarrow} \langle 011, -3, -11, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle & (3.54) \\
\langle 011, -3, -11, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle &\stackrel{3.15}{\hookrightarrow} \langle 111, -2, -10, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle & (3.55) \\
\langle 111, -2, -10, h_2^1 :: m^1 :: \text{nil}, h_1^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, -1, -9, m^1 :: \text{nil}, h_2^1 \rangle & (3.56) \\
\langle 111, -1, -9, m^1 :: \text{nil}, h_2^1 \rangle &\stackrel{3.11}{\hookrightarrow} \langle 111, 0, -8, \text{nil}, m^1 \rangle & (3.57)
\end{aligned}$$

Figure 3.5. Excerpt of the reachable states of the example program

Figure 3.6. Stack size growth over time



compute the context set $J_{imr, \bar{T}}^{\sigma, a}$ which contains the contexts that are reachable from $\langle imr, \bar{T}, \sigma, a \rangle$ by executing s if a is $s; a'$ or by executing h if a is h ; in addition, we compute the stack growth $K_{imr, \bar{T}}^{\sigma, a}$ whose value is the largest stack growth from the state $\langle imr, \bar{T}, \sigma, a \rangle$ during the execution of s if a is $s; a'$ or during the execution of h if a is h .

We construct the types as follows:

$$\begin{aligned} \text{for any } s \text{ in program } & \text{types}(s) : \bigwedge_{(imr, \bar{T}) \in I_{imr, \bar{T}, u}^{\sigma, s; a}} imr, \bar{T} \rightarrow \bigvee_{(imr', \bar{T}') \in J_{imr, \bar{T}}^{\sigma, s; a}} imr', \bar{T}' \\ \text{for } m & \text{types}(m) : \bigvee_{(imr, \bar{T}) \in I_{0, \hat{0}, 0}^{nl, m}} imr, \bar{T} \\ \text{for } h & \text{types}(h) : \bigwedge_{(imr, \bar{T}) \in I_{imr, \bar{T}, u}^{\sigma, h}} imr, \bar{T} \rightarrow \bigvee_{(imr', \bar{T}') \in J_{imr, \bar{T}}^{\sigma, h}} imr', \bar{T}'. \end{aligned}$$

We construct $\bar{\tau}$ as follows: $\forall u \in 1..N$:

$$\bar{\tau}(u) = \bigwedge_{(imr, \bar{T}) \in I^{\sigma, a} \wedge (imr \wedge t_0, \hat{\theta}_u(\bar{T})) \in I^{a::\sigma, \bar{h}(u)}} imr, \bar{T} \xrightarrow{K_{imr \wedge t_0, \hat{\theta}_u(\bar{T})}^{\sigma, \bar{h}(u)}} \bigvee_{(imr', \bar{T}') \in J_{imr \wedge t_0, \hat{\theta}_u(\bar{T})}^{\sigma, \bar{h}(u)}} imr', \bar{T}'$$

We define the type judgment for any s in the program as: $\bar{\tau} \vdash_{K_{imr, \bar{T}, u}^{\sigma, \bar{h}(u)}} s : \text{types}(s)$; we define the type judgment for m as: $\bar{\tau} \vdash_{K_{0, \hat{0}, 0}^{nl, m}} m : \text{types}(m)$; we define the type judgment for h as: $\bar{\tau} \vdash_{K_{imr, \bar{T}, u}^{\sigma, h}} h : \text{types}(y)$.

We now illustrate how to construct $\bar{\tau}$ for the program in Figure (3.4). We will limit our focus to $\bar{\tau}(1)$ for handler 1. $\bar{\tau}(2)$ of handler 2 is similarly constructed. It is straightforward from the transition excerpt given in Figure (3.5) to derive the following:

a	σ	$(imr, \bar{T}) \in I^{\sigma, a}$	$\in I^{a::\sigma, \bar{h}(1)}$	$\in J^{\sigma, \bar{h}(1)}_{imr \wedge t_0, \hat{\theta}_1(\bar{T})}$	$K^{\sigma, \bar{h}(1)}_{imr, \bar{T}}$	From
m^1	nil	(111, 0, 0)	(011, -3, 1)	(111, 0, -8)	2	(3.33-3.44)
h_2^1	$h_1^1 :: m^1 :: \text{nil}$	(111, 0, -16)	(011, -3, -15)	(111, -1, -13)	0	(3.37-3.39)
h_1^1	$m^1 :: \text{nil}$	(111, 0, -12)	(011, -3, -11)	(111, -1, -9)	0	(3.41-3.43)
h_2^1	$m^1 :: \text{nil}$	(111, 2, -18)	(011, -1, -17)	(111, 0, -12)	1	(3.47-3.48, 3.50-3.53)
			(011, -1, -17)	(111, 0, -15)	1	(3.47-3.49)
		(111, 0, -2)	(011, -3, -11)	(111, -1, -9)	0	(3.54-3.56)
h_1^1	$h_2^1 :: m^1 :: \text{nil}$	(111, 0, -16)	(011, -3, -15)	(111, -1, -13)	0	(3.50-3.52)

Therefore, we have the following intersection type for handler 1:

$$\begin{aligned}
\bar{\tau}(1) = & \dots \bigwedge (111, 0, 0) \xrightarrow{2} \dots \vee (111, 0, -8) \vee \dots \\
& \bigwedge (111, 0, -16) \xrightarrow{0} \dots \vee (111, -1, -13) \vee \dots \\
& \bigwedge (111, 0, -12) \xrightarrow{0} \dots \vee (111, -1, -9) \vee \dots \\
& \bigwedge (111, 2, -18) \xrightarrow{1} \dots \vee (111, 0, -12) \vee (111, 0, -15) \vee \dots \\
& \bigwedge (111, 0, -2) \xrightarrow{0} \dots \vee (111, -1, -9) \vee \dots \\
& \bigwedge \dots
\end{aligned}$$

We next illustrate how to construct the type derivation for each program point. We again take handler 1 as an example. Consider the type derivation for $\bar{h}(1)$ in the context $(011, -1, -17) \xrightarrow{1} \dots \vee (111, 0, -12) \vee \dots$. The state transitions (3.48-3.53) which are reachable from the state $\langle 011, -1, -17, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle$ during the execution of $h_1^0; h_1^1$ are given in Figure (3.5). Note that transitions (3.48-3.53) include all the possible paths of executing $h_1^0; h_1^1$ starting from the state $\langle 011, -1, -17, h_2^1 :: m^1 :: \text{nil}, h_1^0; h_1^1 \rangle$. It is straightforward from transitions (3.48-3.53) in Figure (3.5) to derive the following:

	$(imr, \overline{T}) \in I_{011,-1,-17,1}^{h_2^1::m^1::nil,a}$	$(imr', \overline{T}') \in J_{imr,\overline{T}}^{h_2^1::m^1::nil,a}$	$K_{imr,\overline{T}}^{h_2^1::m^1::nil,a}$	From
$a = h_1^0; a'$	$(011, -1, -17)$	$(111, 0, -16)$	0	(3.48)
$a = h_1^1$	$(111, 0, -16)$	$(111, 0, -15)$	0	(3.49)
	$(111, 0, -16)$	$(111, 0, -12)$	1	(3.50-3.53)
	$(111, -1, -13)$	$(111, 0, -12)$	0	(3.53)

We, thus, have the following derivations

$$\overline{\tau} \vdash_0 h_1^0 : (111, -1, -17) \rightarrow (111, 0, -16) \quad (3.58)$$

$$\overline{\tau} \vdash_1 h_1^1 : (111, 0, -16) \rightarrow (111, 0, -12) \vee (111, 0, -15) \quad (3.59)$$

$$\bigwedge (111, -1, -13) \rightarrow (111, 0, -12)$$

From Rule (3.28), for derivation (3.59), we need to show that the following safety condition is observed:

1. (a) for the context $(111, 0, -16)$, we have the context that appears to the left side of the arrow in $\overline{\tau}(1)$: $\overline{\tau}(1) = \dots \bigwedge (111, 0, -16) \xrightarrow{0} (111, -1, -13) \bigwedge \dots$
 - (b) for the contexts that appear to the right side of the arrow in $\overline{\tau}(1)$: $\overline{\tau}(1) = \dots \bigwedge (111, 0, -16) \xrightarrow{0} (111, -1, -13) \bigwedge \dots$, we have $(111, -1, -13) \in I_{011,-1,-17,1}^{h_2^1::m^1::nil,h_1^1}$
 - (c) from $\overline{\tau}(1) = \dots \bigwedge (111, 0, -16) \xrightarrow{0} (111, -1, -13) \bigwedge \dots$, we have that the largest stack growth δ during the call of $\overline{h}(1)$ with the context $(111, 0, -16)$ is 0. From $K_{111,0,-16}^{h_2^1::m^1::nil,a} = 1$ and $\delta = 0$, we have $\delta + 1 \leq K_{111,0,-16}^{h_2^1::m^1::nil,h_1^1}$.
 - (d) it is trivial to show that $\widehat{\theta}_1(111, 0, -16) = (-3, -15) \leq (4, 20)$.
2. It is trivial to show that $\widehat{\theta}(111, 0, -16) = (0, -15) \leq (4, 20)$ and $\widehat{\theta}(111, -1, -13) = (0, -12) \leq (4, 20)$.

The safety condition of the derivation (3.58) can be similarly shown. We omit the details here.

Since $\{111, 0, -16\} \subseteq \{(111, 0, -16), (111, -1, -3)\}$, from (3.58), (3.59) and Rule (3.31), we can derive

$$\bar{\tau} \vdash_1 h_1^0; h_1^1 : (111, -1, -17) \rightarrow (111, 0, -12) \vee (111, 0, -15)$$

3.6 Type construction

The abstract semantics defines a transition graph \mathcal{R} of the dynamics of a periodic interrupt calculus program. In this section, we show how to construct types from the graph \mathcal{R} .

In subsection 3.6.1, we define a function $\psi_{\mathcal{R}}$ and show that the function $\psi_{\mathcal{R}}$ has a least fixed point $\mu\psi_{\mathcal{R}}$ by using Tarski's fixed point theorem. The intuition of the fixed point $\mu\psi_{\mathcal{R}}$ is that, for any abstract state in the graph \mathcal{R} and a segment of program code, $\mu\psi_{\mathcal{R}}$ can compute (1) the set of states that are reachable by executing the code segment; and (2) the largest stack growth during the execution of the code segment.

The property that, given an abstract state in \mathcal{R} and a segment of program code, $\mu\psi_{\mathcal{R}}$ computes the set of states in \mathcal{R} that are reachable by executing the code segment is defined as the soundness of $\mu\psi_{\mathcal{R}}$ on \mathcal{R} . We also define the stack-irrelevancy of $\mu\psi_{\mathcal{R}}$ on \mathcal{R} , which states the fact that it has nothing to do with the stack component of the starting state when $\mu\psi_{\mathcal{R}}$ computes (1) the set of states that are reachable by executing a code segment from a given abstract state, and (2) the largest stack growth during the execution of the code segment. We further define the completeness of $\mu\psi_{\mathcal{R}}$ on \mathcal{R} , which states that if an abstract state Q_1 is reachable from another abstract state Q_2 by executing a segment of code, then the abstract state Q_1 is included in the set of the reachable states of Q_2 which is computed with the help of the fixed-point function $\mu\psi_{\mathcal{R}}$. The properties of soundness, stack-irrelevancy and completeness are formally defined in subsection 3.6.2. In addition, we prove $\mu\psi_{\mathcal{R}}$ has the three properties on \mathcal{R} in the same subsection.

In subsection 3.6.3, we show how to construct types and type judgments with the help of the function $\mu\psi_{\mathcal{R}}$. In subsection 3.6.4, we prove that the constructed

types and type judgments are well-formed in the sense of Definition (3.4.1) and Definition (3.4.2). We show how to build the type derivations of statements, the main program and handlers out of \mathcal{R} , in subsection 3.6.5.

Since abstract states always carry \bar{h} , we omit it here for the sake of simplicity. In this section, we let $\bar{h}(0) = p.m.$

3.6.1 ψ function

Recall that \mathcal{R} (Section 3.3) is the set of all reachable abstract states from the initial abstract state Q_0 . We require that, given a number $K \geq 0$, $\forall Q \in \mathcal{R} : (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$. From Theorem (3.3.1), it follows that \mathcal{R} is finite.

For $u \in 0..N$, we define *effective code set* \mathcal{C}_u for each $\bar{h}(u)$.

Definition 3.6.1 (Effective code set \mathcal{C}_u) *Let $\bar{h}(u) \in \mathcal{C}_u$ if $\bar{h}(u)$ is not of the form loop s , and let $s; \text{loop } s \in \mathcal{C}_0$. If $a \in \mathcal{C}_u$ and*

- *a is of the form $s; a'$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $a' \neq \text{loop } s$, then let $a' \in \mathcal{C}_u$.*
- *a is of the form $s; a'$ and s is $\text{if0 } x \text{ then } s_1 \text{ else } s_2$, then let $s_1; a' \in \mathcal{C}_u$ and $s_2; a' \in \mathcal{C}_u$.*
- *a is of the form $s; a'$ and s is $s_1; s_2$, then let $s_1; a' \in \mathcal{C}_u$*
- *a is of the form h and h is $s; h'$, then let $h' \in \mathcal{C}_u$.*
- *a is of the form h and h is iret , then let $\text{iret} \in \mathcal{C}_u$.*

Let $\mathcal{C} = \bigcup \{ \mathcal{C}_u \mid u \in 0..N \}$. We use variable a to range over \mathcal{C} . Clearly, $\mathcal{C} \neq \emptyset$ and \mathcal{C} is finite. Let $\mathcal{S} = \{ \sigma \mid \langle \text{imr}, \bar{T}, \sigma, a \rangle \in \mathcal{R} \}$. We use variable σ to range over \mathcal{S} . Clearly, \mathcal{S} is finite and $\mathcal{S} \neq \emptyset$ (because we have $\text{nil} \in \mathcal{S}$.) Recall that in Section 3.4 we assume all the contexts $\{ \langle \text{imr}, \bar{T} \rangle \mid \langle \text{imr}, \bar{T}, \sigma, a \rangle \in \mathcal{R} \}$ are uniquely indexed; we now let $\mathcal{I} = \{ i \mid \langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \in \mathcal{R} \}$; we use variables i and j to range over \mathcal{I} . Specifically we denote the index of $(Q_0.\text{imr}, Q_0.\bar{T})$ by i_0 . We use variables I and J

to range over $2^{\mathcal{I}}$; We let $\mathcal{N}_K = \{0..K\}$. We use variable k to range over \mathcal{N}_K and M to range over $2^{\mathcal{N}_K}$.

Define function $f : \mathcal{I} \times \mathcal{S} \times \mathcal{C} \rightarrow ((2^{\mathcal{I}} \times \mathcal{N}_K))$; we use $f(i, \sigma, a).I$ to denote the first element of the pair $f(i, \sigma, a) = (I, k)$; use $f(i, \sigma, a).k$ to denote the second element of the pair $f(i, \sigma, a) = (I, k)$; define the set of all such functions as $D = \{2^{\mathcal{I} \times \mathcal{S} \times \mathcal{C}} \rightarrow 2^{2^{\mathcal{I}} \times \mathcal{N}_K}\}$.

Define an auxiliary function $\max = \lambda M.k : 2^{\mathcal{N}_K} \rightarrow \mathcal{N}_K$ as

$$M \neq \emptyset \wedge (\max(M) = k \text{ such that } k \in M \wedge \forall k' \in M : k \geq k')$$

Define function $\psi_{\mathcal{R}} : D \rightarrow D$ as follows:

Definition 3.6.2 ($\psi_{\mathcal{R}}$) Define $f' = \psi_{\mathcal{R}}f$: for all $i \in \mathcal{I}$, $\sigma \in \mathcal{S}$ and $a \in \mathcal{C}$:

1. if $a = s; a'$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip :

$$\begin{aligned} f'(i, \sigma, a) = (& \bigcup \{f(j, \sigma, s; a').I \mid j \in J\} \cup \{ \alpha(\xi_s(i)) \}, \\ & \max(\{f(\alpha(\zeta_u(i)), s; a'::_S\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\} \cup \\ & \{f(j, \sigma, s; a').k \mid j \in J\} \cup \{|\sigma|\})) \end{aligned}$$

where

$$J = \bigcup \{f(\alpha(\zeta_u(i)), s; a'::_S\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\}$$

2. if $a = s; a'$ and s is $\text{if0 } x \text{ then } s_1 \text{ else } s_2$:

$$\begin{aligned} f'(i, \sigma, a) = (& f(i, \sigma, s_1; a').I \cup f(i, \sigma, s_2; a').I, \\ & \max(\{f(i, \sigma, s_1; a').k, f(i, \sigma, s_2; a').k\}) \end{aligned}$$

3. if $a = s; a'$ and s is $s_1; s_2$:

$$\begin{aligned} f'(i, \sigma, a) = (& \bigcup \{f(j, \sigma, s_2; a').I \mid j \in J\}, \\ & \max(\{f(i, \sigma, s_1; (s_2; a')).k\} \cup \{f(j, \sigma, s_2; a').k \mid j \in J\})) \end{aligned}$$

where

$$J = f(i, \sigma, s_1; (s_2; a')).I$$

4. if $a = h$ and h is iret :

$$f'(i, \sigma, a) = (\bigcup \{f(j, \sigma, \text{iret}).I \mid j \in J\} \cup \{ \alpha(\xi_{\text{iret}}(i)) \},$$

$$\max(\{f(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\mathcal{T}}_i)\} \cup \{f(j, \sigma, \text{iret}).k \mid j \in J\} \cup \{|\sigma|\})$$

where

$$J = \bigcup \{f(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\mathcal{T}}_i)\}$$

5. if $a = h$ and h is $s; h'$:

$$f'(i, \sigma, a) = (\bigcup \{f(j, \sigma, h').I \mid j \in J\}, \max(\{f(j, \sigma, h').k\} \cup \{f(i, \sigma, s; h').k \mid j \in J\}))$$

where

$$J = f(i, \sigma, s; h').I$$

where operation $::_{\mathcal{S}}$ on \mathcal{C} and \mathcal{S} is defined as:

$$\forall a \in \mathcal{C} : \forall \sigma \in \mathcal{S} : a::_{\mathcal{S}}\sigma = \begin{cases} a :: \sigma & \text{if } a :: \sigma \in \mathcal{S} \\ \text{nil} & \text{if } a :: \sigma \notin \mathcal{S} \end{cases}$$

and operation α is defined as:

$$\forall i \in \mathcal{I} : \alpha(z) = \begin{cases} z & \text{if } z \in \mathcal{I} \\ i_0 & \text{if } z \notin \mathcal{I} \end{cases} \text{ where } z \text{ is of the form either } \zeta_u(i) \text{ or } \xi_s(i)$$

We define a reflexive, antisymmetric and transitive binary relation \leq over D as follows: $f_1 \leq f_2$ iff

$$\forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : \forall i \in \mathcal{I} : f_1(i, \sigma, a).I \subseteq f_2(i, \sigma, a).I \wedge f_1(i, \sigma, a).k \leq f_2(i, \sigma, a).k$$

Let f_{\top} be $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : f_{\top}(i, \sigma, a) = (\mathcal{I}, K)$ and let f_{\perp} be $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : f_{\perp}(i, \sigma, a) = (\emptyset, 0)$, and let $D' \subseteq D$, then we have $\bigvee_{f \in D'} f \leq f_{\top}$ and $f_{\perp} \leq \bigwedge_{f \in D'} f$. It is straightforward to show that (D, \leq) is a lattice.

We define f_0 as $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : f_0(i, \sigma, a) = (\emptyset, |\sigma|)$.

Given the relation \leq over D , it is straightforward to show that function $\psi_{\mathcal{R}}$ is monotone. It follows from Tarski's fixed-point theorem that $\psi_{\mathcal{R}}$ has a least fixed-point $\mu\psi_{\mathcal{R}}$.

Proposition 3.6.1 ($\mu\psi_{\mathcal{R}}$ Rewrite) For all $i \in \mathcal{I}$, $\sigma \in \mathcal{S}$ and $a \in \mathcal{C}$, we have

1. If $a = s; a'$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , then we have

$$\begin{aligned}\mu\psi_{\mathcal{R}}(i, \sigma, a).I &= \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, s; a').I \mid j \in J \} \cup \{ \alpha(\xi_s(i)) \} \\ \mu\psi_{\mathcal{R}}(i, \sigma, a).k &= \max(\{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a'::_{\mathcal{S}}\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \} \cup \\ &\quad \{ \mu\psi_{\mathcal{R}}(j, \sigma, s; a').k \mid j \in J \} \cup \{ |\sigma| \})\end{aligned}$$

where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a'::_{\mathcal{S}}\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$.

2. If $a = s; a'$ and s is $\text{if0 } x \text{ then } s_1 \text{ else } s_2$, then we have

$$\begin{aligned}\mu\psi_{\mathcal{R}}(i, \sigma, a).I &= \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').I \cup \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').I \\ \mu\psi_{\mathcal{R}}(i, \sigma, a).k &= \max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').k, \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').k \})\end{aligned}$$

3. if $a = s; a'$ and s is $s_1; s_2$, then we have

$$\begin{aligned}\mu\psi_{\mathcal{R}}(i, \sigma, a).I &= \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').I \mid j \in J \} \\ \mu\psi_{\mathcal{R}}(i, \sigma, a).k &= \max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).k \} \cup \{ \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').k \mid j \in J \})\end{aligned}$$

where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I$

4. If $a = h$ and h is iret :

$$\begin{aligned}\mu\psi_{\mathcal{R}}(i, \sigma, a).I &= \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).I \mid j \in J \} \cup \{ \alpha(\xi_{\text{iret}}(i)) \} \\ \mu\psi_{\mathcal{R}}(i, \sigma, a).k &= \max(\{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \} \cup \\ &\quad \{ \mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).k \mid j \in J \} \cup \{ |\sigma| \})\end{aligned}$$

where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$

5. If $a = h$ and h is $s; h'$:

$$\begin{aligned}\mu\psi_{\mathcal{R}}(i, \sigma, a).I &= \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, h').I \mid j \in J \} \\ \mu\psi_{\mathcal{R}}(i, \sigma, a).k &= \max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k \} \cup \{ \mu\psi_{\mathcal{R}}(j, \sigma, h').k \mid j \in J \})\end{aligned}$$

where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$

Proposition 3.6.2 *If $Q \in \mathcal{R} \Rightarrow Q.stk \leq K$, then $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : \mu\psi_{\mathcal{R}}(i, \sigma, a).k \leq K$.*

We now prove some lemmas regarding the fixed point $\mu\psi_{\mathcal{R}}$.

Lemma 3.6.1 *For all $i \in \mathcal{I}$, $\sigma \in \mathcal{S}$ and $a \in \mathcal{C}$, we have*

1. *if $a = s; a'$, s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\mathcal{T}}_i)$, then we have $\mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a'::_{\mathcal{S}}\sigma, \overline{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a').k$*
2. *if $a = s; a'$ and s is $\text{if0 } x \text{ then } s_1 \text{ else } s_2$, then we have $\mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a').k$ and $\mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a').k$*
3. *if $a = (s_1; s_2); a'$, then we have $\mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a').k$ and $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I \Rightarrow \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a').k$*
4. *if $a = \text{iret}$ and $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\mathcal{T}}_i)$, then we have $\mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \overline{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k$*
5. *if $h = s; h'$, then we have $\mu\psi_{\mathcal{R}}(i, \sigma, s; h').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$ and $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I \Rightarrow \mu\psi_{\mathcal{R}}(j, \sigma, h').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$*

Proof The proofs of each Item is immediately from corresponding Item of Proposition (3.6.1). ■

Lemma 3.6.2 *For all $i \in \mathcal{I}$, $\sigma \in \mathcal{S}$ and $a \in \mathcal{C}$, we have*

1. $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$
2. $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$

Proof We proceed by induction on the structure of $a \in \mathcal{C}$. There are five cases depending upon the form of a .

1. a is of the form: $s; a'$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or **skip**.

From Item (1) of Proposition (3.6.1), we have

- (i) $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).I = \bigcup\{\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \mid j \in J\} \cup \{\alpha(\xi_s(i))\}$
 where $J = \bigcup\{\mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a::_{\mathcal{S}}\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\widehat{T}}_i)\}$
- (ii) $\mu\psi_{\mathcal{R}}(i, \sigma, a).k = \max(\{\mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a::_{\mathcal{S}}\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\widehat{T}}_i)\} \cup \{f(j, \sigma, s; a').k \mid j \in J\} \cup \{|\sigma|\})$

From (i), we have $\alpha(\xi_s(i)) \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$, which is Item (1).

From (ii), we have $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).k \geq |\sigma|$, which is Item (2).

2. a is of the form: $s; a'$ and s is **if0** x **then** s_1 **else** s_2 .

From Item (2) of Proposition (3.6.1), we have

- (i) $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').I \cup \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').I$
- (ii) $\mu\psi_{\mathcal{R}}(i, \sigma, a).k = \max(\{\mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').k, \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').k\})$

From the induction hypothesis, we have

- (iii) $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').I \neq \emptyset$ and
 $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').I \neq \emptyset$
- (iv) $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').k \geq |\sigma|$ and
 $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').k \geq |\sigma|$

From (i) and (iii), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$, which is Item (1).

From (ii) and (iv), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$, which is Item (2).

3. a is of the form: $s; a'$ and s is $s_1; s_2$.

From Item (3) of Proposition (3.6.1), we have

- (i) $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \bigcup\{\mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').I \mid j \in J\}$
- (ii) $\mu\psi_{\mathcal{R}}(i, \sigma, a).k = \max(\{\mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).k, \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').k \mid j \in J\})$
 where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I$

From the induction hypothesis, we have

- (iii) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a')).I \neq \emptyset$
- (iv) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a')).k \geq |\sigma|$
- (v) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').I \neq \emptyset$

From (iii) and $J = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I$, we have $J \neq \emptyset$.

From $J \neq \emptyset$, (v) and (i), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$, which is Item (1).

From (iv) and (ii), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$, which is Item (2).

4. a is of the form: h and h is iret.

From Item (4) of Proposition (3.6.1), we have

- (i) $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).I \mid j \in J \} \cup \{ \alpha(\xi_{\text{iret}}(i)) \}$
 where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$
- (ii) $\mu\psi_{\mathcal{R}}(i, \sigma, a).k = \max(\{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{\mathcal{S}}\sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \} \cup \{ f(j, \sigma, \text{iret}).k \mid j \in J \} \cup \{ |\sigma| \})$

From (i), we have $\alpha(\xi_{\text{iret}}(i)) \in \mu\psi_{\mathcal{R}}(i, \sigma, a).I$, which is Item (1).

From (ii), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$, which is Item (2).

5. a is of the form: h and $h = s; h'$.

From Item (5) of Proposition (3.6.1), we have

- (i) $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, h').I \mid j \in J \}$
- (ii) $\mu\psi_{\mathcal{R}}(i, \sigma, a).k = \max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k, \mu\psi_{\mathcal{R}}(j, \sigma, h').k \mid j \in J \})$

where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$

From the induction hypothesis, we have

- (iii) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, s; h').I \neq \emptyset$
- (iv) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, s; h').k \geq |\sigma|$
- (v) $\forall j \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \mu\psi_{\mathcal{R}}(j, \sigma, h').I \neq \emptyset$

From (iii) and $J = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$, we have $J \neq \emptyset$.

From $J \neq \emptyset$, (v) and (i), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$, which is Item (1).

From (iv) and (ii), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$, which is Item (2).

■

3.6.2 Soundness, stack-irrelevancy and completeness of ψ

We define an anti-reflexive, transitive relation \subset over stack variable \mathcal{S} as follows: $\sigma \subset \sigma'$ iff $\sigma' = a_k :: \dots :: a_1 :: \sigma$, where $k \geq 1$. We define a reflexive, antisymmetric, transitive relation \subseteq over stack variable \mathcal{S} as follows: $\sigma \subseteq \sigma'$ iff $\sigma = \sigma'$ or $\sigma \subset \sigma'$.

We label the transition $Q \hookrightarrow Q'$ with $(Q'.\sigma$ if it is the Rule (3.10); label $Q \hookrightarrow Q'$ with $)^{Q.\sigma}$ if it is the Rule (3.11). We use variable π to denote state transition path. Given a path π , we use function $\Pi(\pi)$ to denote the set of labels along π . We abbreviate $l^{\sigma_1} \notin \Pi(\pi), \dots, l^{\sigma_k} \notin \Pi(\pi)$ as $l^{\sigma_1}, \dots, l^{\sigma_k} \notin \Pi(\pi)$, where l can be either (or).

Proposition 3.6.3 *We have*

1. If $\pi : Q \hookrightarrow^* Q'$ and $Q.\sigma \subset Q'.\sigma$, then for any σ such that $Q.\sigma \subset \sigma \subseteq Q'.\sigma$, we have $(\sigma \in \Pi(\pi))$.
2. If $\pi : Q \hookrightarrow^* Q'$ and $Q'.\sigma \subset Q.\sigma$, then for any σ such that $Q'.\sigma \subset \sigma \subseteq Q.\sigma$, we have $)^\sigma \in \Pi(\pi)$.
3. If $\pi : Q \hookrightarrow^* Q'$, $Q.\sigma = Q'.\sigma$ and $)^{Q.\sigma} \notin \Pi(\pi)$, then $(^{Q.\sigma} \notin \Pi(\pi)$.
4. If π' is a sub-path of π and then $\Pi(\pi') \subseteq \Pi(\pi)$.
5. If π is a concatenation of π_1 and π_2 , then $\Pi(\pi) = \Pi(\pi_1) \cup \Pi(\pi_2)$.

Proposition 3.6.4 *We have*

1. if path $\pi : \langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$, then $\exists i'$ along π such that $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \hookrightarrow^* \langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$.
2. if path $\pi : \langle imr_i, \overline{T}_i, a :: \sigma, s; h \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$, then $\exists i'$ along π such that $\langle imr_i, \overline{T}_i, a :: \sigma, s; h \rangle \hookrightarrow^* \langle imr_{i'}, \overline{T}_{i'}, a :: \sigma, h \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, a :: \sigma, h \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$.
3. For $u \in 0..N$, if $s; a \in \mathcal{C}_u$ and path $\pi : \langle imr_i, \overline{T}_i, \sigma, \overline{h}(u) \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$, then there exists i' along π such that $\langle imr_i, \overline{T}_i, \sigma, \overline{h}(u) \rangle \hookrightarrow^* \langle imr_{i'}, \overline{T}_{i'}, \sigma, s; a \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s; a \rangle \hookrightarrow^* \langle imr_j, \overline{T}_j, \sigma, a \rangle$.

Definition 3.6.3 We denote $Q \xrightarrow{\sigma} Q'$ if there exists a path π from Q to Q' such that

1. $Q.\sigma = Q'.\sigma = \sigma$ and $(\sigma,)^\sigma, ((Q'.a)::\sigma,)^{(Q'.a)::\sigma} \notin \Pi(\pi)$, or
2. $Q.\sigma = (Q'.a) :: \sigma$ and $Q'.\sigma = \sigma$ and $(\sigma,)^\sigma, ((Q'.a)::\sigma) \notin \Pi(\pi)$.

Definition 3.6.4 We denote $Q \xrightarrow{\sigma^+} Q'$ if there exists a path π from Q to Q' such that

1. $Q.\sigma = Q'.\sigma = \sigma$ and $(\sigma,)^\sigma \notin \Pi(\pi)$, or
2. $Q.\sigma = (Q'.a) :: \sigma$ and $Q'.\sigma = \sigma$ and $(\sigma,)^\sigma \notin \Pi(\pi)$.

Proposition 3.6.5 We have

1. if $Q \xrightarrow{\sigma} Q'$, then $Q \xrightarrow{\sigma^+} Q'$.
2. if $\langle imr_i, \overline{T}_i, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a_2 \rangle$, then $\exists i'$ such that $\langle imr_i, \overline{T}_i, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, a_2 \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, a_2 \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a_2 \rangle$.
3. if $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$, then either $i = j$ or $i \neq j \wedge \eta(a) \in \mathcal{L}$.

Lemma 3.6.3 If $\pi: \langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$ and $\sigma' \subset \sigma$, then we have $(\sigma',)^{\sigma'} \notin \Pi(\pi)$.

Proof Suppose for the sake of contradiction that $(\sigma' \in \Pi(\pi))$. Then there must be a state transition $\langle imr', \overline{T}', \sigma'', a \rangle \hookrightarrow \langle imr' \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}'), \sigma', \overline{h}(u) \rangle$ along π , for some $u \in 1..N$. Consider the sub-path π' of π : $\langle imr' \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}'), \sigma', \overline{h}(u) \rangle \hookrightarrow^* \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$. From $\sigma' \subset \sigma$, $\sigma \subseteq \sigma$, and Item (1) of Proposition (3.6.3), we have that $(\sigma \in \Pi(\pi'))$. From π' is sub-path of π , $(\sigma \in \Pi(\pi'))$ and Item (4) of Proposition (3.6.3), we have $(\sigma \in \Pi(\pi))$, which is a contradiction.

Suppose for the sake of contradiction that $(\sigma' \in \Pi(\pi))$. Then there must be a state transition $\langle imr', \overline{T}', \sigma', \text{iret} \rangle \hookrightarrow \langle imr' \vee \mathbf{t}_0, \widehat{\theta}(imr', \overline{T}'), \sigma'', a \rangle$ along π , where $\sigma' = a :: \sigma''$. Consider the sub-path of π' : $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \hookrightarrow^* \langle imr', \overline{T}', \sigma', \text{iret} \rangle$. From $\sigma' \subset \sigma$,

$\sigma \subseteq \sigma$, and Item (2) of Proposition (3.6.3), we have that there is a $)^\sigma \in \Pi(\pi')$. From π' is sub-path of π , $)^\sigma \in \Pi(\pi')$ and Item (4) of Proposition (3.6.3), we have $)^\sigma \in \Pi(\pi)$, which is a contradiction. \blacksquare

Lemma 3.6.4 *We have*

1. if $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$ and $\langle imr_2, \overline{T}_2, \sigma, a_2 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$ then $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$.
2. for $u \in 1..N$, if $\langle imr_1, \overline{T}_1, \sigma, a \rangle \hookrightarrow \langle imr_1 \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}_1), a :: \sigma, \overline{h}(u) \rangle$ and $\langle imr_1 \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}_1), a :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} \langle imr_2, \overline{T}_2, \sigma, a \rangle$, then $\langle imr_1, \overline{T}_1, \sigma, a \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, a \rangle$.
3. if $\sigma = a :: \sigma'$, $\langle imr_1, \overline{T}_1, \sigma, h_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, h_2 \rangle$ and $\langle imr_2, \overline{T}_2, \sigma, h_2 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{T}_3, \sigma', a \rangle$, then $\langle imr_1, \overline{T}_1, \sigma, h_1 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{T}_3, \sigma', a \rangle$.
4. if $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$ and $\langle imr_2, \overline{T}_2, \sigma, a_2 \rangle \xrightarrow{\sigma^+} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$, then $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$.

Proof 1. From $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$, we have path π_1 :

$\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \hookrightarrow^* \langle imr_2, \overline{T}_2, \sigma, a_2 \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi_1)$. From

$\langle imr_2, \overline{T}_2, \sigma, a_2 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$, we have path π_2 : $\langle imr_2, \overline{T}_2, \sigma, a_2 \rangle \hookrightarrow^*$

$\langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$ and $(\sigma,)^\sigma, (a_3 :: \sigma,)^{a_3 :: \sigma} \notin \Pi(\pi_2)$. Concatenating π_1 and π_2 , we

have path π_3 : $\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \hookrightarrow^* \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$. From $(\sigma,)^\sigma \notin \Pi(\pi_1)$,

$(\sigma,)^\sigma, (a_3 :: \sigma,)^{a_3 :: \sigma} \notin \Pi(\pi_2)$, π_3 is a concatenation of π_1 and π_2 and Item (5) of

Proposition (3.6.3), we have $(\sigma,)^\sigma, (a_3 :: \sigma,)^{a_3 :: \sigma} \notin \Pi(\pi_3)$, from which it follows

$\langle imr_1, \overline{T}_1, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{T}_3, \sigma, a_3 \rangle$.

2. From $\langle imr_1 \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}_1), a :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} \langle imr_2, \overline{T}_2, \sigma, a \rangle$, we have path π :

$\langle imr_1 \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}_1), a :: \sigma, \overline{h}(u) \rangle \hookrightarrow^* \langle imr_2, \overline{T}_2, \sigma, a \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. Con-

catenating $\langle imr_1, \overline{T}_1, \sigma, a \rangle \hookrightarrow \langle imr_1 \wedge \neg \mathbf{t}_0, \widehat{\theta}_u(\overline{T}_1), a :: \sigma, \overline{h}(u) \rangle$ and π , we have

path π' : $\langle imr_1, \overline{T}_1, \sigma, a \rangle \hookrightarrow^* \langle imr_2, \overline{T}_2, \sigma, a \rangle$. From $(\sigma,)^\sigma \notin \Pi(\pi)$, there is nei-

ther $(\sigma$ nor $)^\sigma$ along the transition

$\langle imr_1, \overline{\overline{T}}_1, \sigma, a \rangle \hookrightarrow \langle imr_1 \wedge \neg t_0, \widehat{\theta}_u(\overline{\overline{T}}_1), a :: \sigma, \overline{h}(u) \rangle$, π' is a concatenation of π and the transition and Item (5) of Proposition (3.6.3), we have $(\sigma,)^\sigma \notin \Pi(\pi')$, from which it follows $\langle imr_1, \overline{\overline{T}}_1, \sigma, a \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, a \rangle$.

3. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$, we have path π_1 :
 $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \hookrightarrow^* \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi_1)$. From $\sigma = a :: \sigma'$, we have $\sigma' \subset \sigma$. From $\sigma' \subset \sigma$, $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$ and Lemma (3.6.3), $(\sigma',)^{\sigma'} \notin \Pi(\pi_1)$. From $\langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$, we have path π_2 : $\langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle \hookrightarrow^* \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$ and $(\sigma',)^{\sigma'}, (\sigma \notin \Pi(\pi_2))$. Concatenating π_1 and π_2 , we have path π_3 : $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \hookrightarrow^* \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$. From $(\sigma',)^{\sigma'}, (\sigma,)^\sigma \notin \Pi(\pi_1)$, π_3 is a concatenation π_1 and π_2 and Item (5) of Proposition (3.6.3), we have $(\sigma',)^{\sigma'}, (\sigma \notin \Pi(\pi_3))$, from which it follows $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$.

4. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$, we have π_1 : $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \hookrightarrow^* \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi_1)$. From $\langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle \xrightarrow{\sigma^+} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$, we have π_2 : $\langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle \hookrightarrow^* \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi_2)$. Concatenating π_1 and π_2 , we have π_3 : $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \hookrightarrow^* \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$. From $(\sigma,)^\sigma \notin \Pi(\pi_1)$, $(\sigma,)^\sigma \notin \Pi(\pi_2)$, π_3 is concatenation of π_1 and π_2 and Item (5) of Proposition (3.6.3), we have $(\sigma,)^\sigma \notin \Pi(\pi_3)$, from which it follows $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$.

■

Corollary 3.6.1 *We have*

1. if $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$ and $\langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$ then $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$.
2. if $\sigma = a :: \sigma'$, $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$ and $\langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$, then $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$.

Proof 1. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$ and Item (1) of Proposition (3.6.5), we have

$\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle$,
 $\langle imr_2, \overline{\overline{T}}_2, \sigma, a_2 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$ and Item (1) of Lemma (3.6.4), we have
 $\langle imr_1, \overline{\overline{T}}_1, \sigma, a_1 \rangle \xrightarrow{\sigma} \langle imr_3, \overline{\overline{T}}_3, \sigma, a_3 \rangle$.

2. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$ and Item (1) of Proposition (3.6.5),
 we have

$\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$. From $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma^+} \langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle$,
 $\langle imr_2, \overline{\overline{T}}_2, \sigma, h_2 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$, $\sigma = a :: \sigma'$ and Item (3) of Lemma (3.6.4),
 we have $\langle imr_1, \overline{\overline{T}}_1, \sigma, h_1 \rangle \xrightarrow{\sigma'} \langle imr_3, \overline{\overline{T}}_3, \sigma', a \rangle$. ■

Let $f \in D$. We define the soundness of f on \mathcal{R} as follows.

Definition 3.6.5 (Soundness of f on \mathcal{R}) f is sound on \mathcal{R} iff for all $\sigma \in \mathcal{S}$, $a \in \mathcal{C}$
 and $i \in \mathcal{I}$, if $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$, then

1. $j \in f(i, \sigma, s; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$ if $a = s; a'$
2. $j \in f(i, \sigma, h).I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle$ if $a = h$ and $\sigma = a' :: \sigma'$

We simply say that f is sound if it is clear from the context which \mathcal{R} we refer to.

Lemma 3.6.5 Suppose $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$,

1. if $a = s; a'$ and s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$,
 then we have $\alpha(\xi_s(i)) = \xi_s(i)$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_{\xi_s(i)}, \overline{\overline{T}}_{\xi_s(i)}, \sigma, a' \rangle$.
2. if $a = s; a'$ and s is $if0\ x\ then\ s_1\ else\ s_2$, then we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma}$
 $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle$.
3. if $a = iret$ and $\sigma = a' :: \sigma'$, then we have $\alpha(\xi_{iret}(i)) = \xi_{iret}(i)$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \xrightarrow{\sigma'}$
 $\langle imr_{\xi_{iret}(i)}, \overline{\overline{T}}_{\xi_{iret}(i)}, \sigma', a' \rangle$.
4. if $u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i)$ and $\eta(a) \in \mathcal{L}$, then we have $a ::_{\mathcal{S}} \sigma = a :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$
 and $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle$.

Proof 1. From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle$ and applying either Rule (3.13) - (3.15), or (3.18), depending on what s is, we have

$$(i) \quad \langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \hookrightarrow \langle imr_{\xi_s(i)}, \overline{T}_{\xi_s(i)}, \sigma, a' \rangle.$$

From (i) and either Rule (3.13) - (3.15), or (3.18) does not introduce $(\sigma,)^\sigma$, $(a'::\sigma$ or $)^{a'::\sigma}$, we have

$$(ii) \quad \langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_{\xi_s(i)}, \overline{T}_{\xi_s(i)}, \sigma, a' \rangle.$$

From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, and (i), we have $\langle imr_{\xi_s(i)}, \overline{T}_{\xi_s(i)}, \sigma, a' \rangle \in \mathcal{R}$, from which it follows $\xi_s(i) \in \mathcal{I}$. From $\xi_s(i) \in \mathcal{I}$, we have $\alpha(\xi_s(i)) = \xi_s(i)$.

2. From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle$ and Rule (3.16), we have

$$(i) \quad \langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \hookrightarrow \langle imr_i, \overline{T}_i, \sigma, s_1; a' \rangle.$$

From (i) and Rule (3.16) does not introduce $(\sigma,)^\sigma$, $((s_1; a')::\sigma,)^{(s_1; a')::\sigma}$, we have $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_i, \overline{T}_i, \sigma, s_1; a' \rangle$.

From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle$ and Rule (3.17), we have

$$(ii) \quad \langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \hookrightarrow \langle imr_i, \overline{T}_i, \sigma, s_2; a' \rangle.$$

From (ii) and Rule (3.17) does not introduce $(\sigma,)^\sigma$, $((s_2; a')::\sigma,)^{(s_2; a')::\sigma}$, we have $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_i, \overline{T}_i, \sigma, s_2; a' \rangle$.

3. From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle$, $\sigma = a' :: \sigma'$ and Rule (3.11), we have

$$(i) \quad \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \hookrightarrow \langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle.$$

From Rule (3.11) does not introduce $(\sigma,)^\sigma$ or $(a::\sigma$ and (i), we have

$\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle$. From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and (i), we have

$\langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle \in \mathcal{R}$, from which it follows $\xi_{\text{iret}}(i) \in \mathcal{I}$. From $\xi_{\text{iret}}(i) \in \mathcal{I}$, we have $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$.

4. From $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, $\eta(a) \in \mathcal{L}$, $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle$ and Rule (3.10), we have

$$(i) \quad \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \hookrightarrow \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a :: \sigma, \overline{h}(u) \rangle.$$

From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and (i), we have $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$, from which it follows $a :: \sigma \in \mathcal{S}$ and $\zeta_u(i) \in \mathcal{I}$. From $a :: \sigma \in \mathcal{S}$, we have $a::s\sigma = a :: \sigma$. From $\zeta_u(i) \in \mathcal{I}$, we have $\alpha(\zeta_u(i)) = \zeta_u(i)$. ■

Lemma 3.6.6 *If f is sound on \mathcal{R} then $\psi_{\mathcal{R}}f$ is also sound on \mathcal{R} .*

Proof From Definition (3.6.5), we need to prove the following: for all $i \in \mathcal{I}$, $\sigma \in \mathcal{S}$ and $a \in \mathcal{C}$, if $\langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}$, then

1. $j \in \psi_{\mathcal{R}}f(i, \sigma, s; a').I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma}^* \langle imr_j, \overline{T}_j, \sigma, a' \rangle$ if $a = s; a'$
2. $j \in \psi_{\mathcal{R}}f(i, \sigma, h).I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, h \rangle \xrightarrow{\sigma'}^* \langle imr_j, \overline{T}_j, \sigma', a' \rangle$ if $a = h$ and $\sigma = a' :: \sigma'$

If $a = s; a'$, then we need to prove Item (1). There are three cases depending on the form of s .

1. s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**.

From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, $a = s; a'$ and s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip** and Item (1) of Lemma (3.6.5), we have $\alpha(\xi_s(i)) = \xi_s(i)$ and $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_{\xi_s(i)}, \overline{T}_{\xi_s(i)}, \sigma, a' \rangle$. From $\alpha(\xi_s(i)) = \xi_s(i)$ and Item (1) of Proposition (3.6.1), we have

$$(i) \quad \psi_{\mathcal{R}}f(i, \sigma, s; a').I = \bigcup \{f(j, \sigma, s; a').I \mid j \in J\} \cup \{\xi_s(i)\}$$

where $J = \bigcup \{f(\alpha(\zeta_u(i)), s; a' ::_{\mathcal{S}} \sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$. There are two subcases depending on whether $\widehat{\mathcal{P}}(imr_i, \overline{T}_i)$ is empty or not.

Subcase 1: $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$.

From $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$, $\alpha(\xi_s(i)) = \xi_s(i)$ and (i), we have $\psi_{\mathcal{R}}f(i, \sigma, s; a').I = \{\xi_s(i)\}$. From $\psi_{\mathcal{R}}f(i, \sigma, s; a').I = \{\xi_s(i)\}$ and $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_{\xi_s(i)}, \overline{T}_{\xi_s(i)}, \sigma, a' \rangle$, we have $j \in \psi_{\mathcal{R}}f(i, \sigma, s; a').I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a' \rangle$.

Subcase 2: $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) \neq \emptyset$.

Let $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$. From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, $\eta(s; a') \in \mathcal{L}$ and Item (4) of Lemma (3.6.5), we have $s; a' ::_{\mathcal{S}} \sigma = s; a' :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle$. From $s; a' ::_{\mathcal{S}} \sigma = s; a' :: \sigma$

and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we rewrite J as $J = \bigcup \{f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\}$. From $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle$ and $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \in \mathcal{R}$, we have $\langle \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$. From $\langle \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$ and f is sound, we have

$$(ii) \quad j \in f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \Rightarrow \\ \langle \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle \xrightarrow{\sigma} \langle \text{imr}_j, \bar{T}_j, \sigma, s; a' \rangle$$

From $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle$, (ii) and Item (2) of Lemma (3.6.4), we have

$$(iii) \quad j \in f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \Rightarrow \langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^+} \langle \text{imr}_j, \bar{T}_j, \sigma, s; a' \rangle$$

From $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$, (iii) and $J = \bigcup \{f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\}$, we have

$$(iv) \quad j \in J \Rightarrow \langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^+} \langle \text{imr}_j, \bar{T}_j, \sigma, s; a' \rangle$$

From $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \in \mathcal{R}$ and (iv), we have

$$(v) \quad j \in J \Rightarrow \langle \text{imr}_j, \bar{T}_j, \sigma, s; a' \rangle \in \mathcal{R}$$

From (v) and f is sound, we have

$$(vi) \quad j \in J \Rightarrow (j' \in f(j, \sigma, s; a').I \Rightarrow \langle \text{imr}_j, \bar{T}_j, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_{j'}, \bar{T}_{j'}, \sigma, a' \rangle)$$

From (iv), (vi) and Item (1) of Lemma (3.6.4), we have

$$(vii) \quad j \in J \Rightarrow (j' \in f(j, \sigma, s; a').I \Rightarrow \langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle \text{imr}_{j'}, \bar{T}_{j'}, \sigma, a' \rangle)$$

which is

$$(viii) \quad j' \in \bigcup \{f(j, \sigma, s; a').I \mid j \in J\} \Rightarrow \langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle \text{imr}_{j'}, \bar{T}_{j'}, \sigma, a' \rangle.$$

From $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle \text{imr}_{\xi_s(i)}, \bar{T}_{\xi_s(i)}, \sigma, a' \rangle$, (viii) and (i), we have

$$j \in \psi_{\mathcal{R}} f(i, \sigma, s; a').I \Rightarrow \langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_j, \bar{T}_j, \sigma, a' \rangle.$$

2. s is if0 x then s_1 else s_2 .

From Item (2) of Definition (3.6.2), we have

$$(i) \quad \psi_{\mathcal{R}} f(i, \sigma, s; a').I = f(i, \sigma, s_1; a').I \cup f(i, \sigma, s_2; a').I$$

From $\langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$ and Item (2) of Lemma (3.6.5), we have

$$(ii) \quad \langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \bar{T}_i, \sigma, s_1; a' \rangle$$

$$(iii) \quad \langle \text{imr}_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \bar{T}_i, \sigma, s_2; a' \rangle$$

from which it follows: $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$.

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$ and f is sound, we have

$$(iv) \quad j \in f(i, \sigma, s_1; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$$

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$ and f is sound, we have

$$(v) \quad j \in f(i, \sigma, s_2; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$$

From (ii), (iv) and Item (1) of Corollary (3.6.1), we have

$$(vi) \quad j \in f(i, \sigma, s_1; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$$

From (iii), (v) and Item (1) of Corollary (3.6.1), we have

$$(vii) \quad j \in f(i, \sigma, s_2; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$$

From (vi), (vii) and (i), we have $j \in \psi_{\mathcal{R}} f(i, \sigma, s; a').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$.

3. s is $s_1; s_2$.

From Item (3) of Definition (3.6.2), we have

$$(i) \quad \psi_{\mathcal{R}} f(i, \sigma, a).I = \bigcup \{ f(j, \sigma, s_2; a').I \mid j \in J \}$$

where $J = f(i, \sigma, s_1; (s_2; a')).I$. From

$\langle imr_i, \overline{\overline{T}}_i, \sigma, (s_1; s_2); a' \rangle = \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, (s_1; s_2); a' \rangle \in \mathcal{R}$, we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$. From

$\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$ and f is sound, we have

$$(ii) \quad j \in f(i, \sigma, s_1; (s_2; a')).I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, s_2; a' \rangle$$

From $J = f(i, \sigma, s_1; (s_2; a')).I$ and (ii), we have

$$(iii) \quad j \in J \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, s_2; a' \rangle$$

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$ and (iii), we have

$$(iv) \quad j \in J \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, s_2; a' \rangle \in \mathcal{R}$$

From (iv) and f is sound, we have

$$(v) \quad j \in J \Rightarrow$$

$$(j' \in f(j, \sigma, s_2; a').I \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, s_2; a' \rangle \xrightarrow{\sigma^*} \langle imr_{j'}, \overline{\overline{T}}_{j'}, \sigma, a' \rangle)$$

From (iii), (v) and Item (1) of Corollary (3.6.1), we have

(vi) $j \in J \Rightarrow$

$$(j' \in f(j, \sigma, s_2; a').I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a') \rangle \xrightarrow{\sigma} \langle imr_{j'}, \overline{T}_{j'}, \sigma, a' \rangle)$$

which is

(vii) $j' \in \bigcup \{f(j, \sigma, s_2; a').I \mid j \in J\} \Rightarrow$

$$\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a') \rangle \xrightarrow{\sigma} \langle imr_{j'}, \overline{T}_{j'}, \sigma, a' \rangle$$

From (vii) and (i), we have

$$j \in \psi_{\mathcal{R}} f(i, \sigma, (s_1; s_2); a').I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, (s_1; s_2); a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a' \rangle.$$

If $a = h$, then we need to prove Item (2). From $\langle imr_i, \overline{T}_i, \sigma, h \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have $\langle imr_i, \overline{T}_i, \sigma, h \rangle$ is consistent, from which it follows $\sigma = a' :: \sigma'$. There are two cases depending on the form of h .

1. h is iret.

From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$, $\sigma = a' :: \sigma'$ and Item (3) of Lemma (3.6.5), we have $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$ and $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle$. From $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$ and Item (4) of Definition (3.6.2), we have

(i) $\psi_{\mathcal{R}} f(i, \sigma, \text{iret}).I = \bigcup \{f(j, \sigma, \text{iret}).I \mid j \in J\} \cup \{ \xi_{\text{iret}}(i) \}$

where $J = \bigcup \{f(\alpha(\zeta_u(i)), \text{iret} ::_S \sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$. There are two subcases depending on whether $\widehat{\mathcal{P}}(imr_i, \overline{T}_i)$ is empty.

Subcase 1 : $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$.

From $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$ and (i), we have $\mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I = \{ \xi_{\text{iret}}(i) \}$. From $\mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I = \{ \xi_{\text{iret}}(i) \}$ and $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle$, we have $j \in \psi_{\mathcal{R}} f(i, \sigma, \text{iret}).I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a' \rangle$.

Subcase 2 : $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) \neq \emptyset$.

Let $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$. From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, $\eta(\text{iret}) \in \mathcal{L}$ and Item (4) of Lemma (3.6.5), we have $\text{iret} ::_S \sigma = \text{iret} :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle$. From $\text{iret} ::_S \sigma = \text{iret} :: \sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we rewrite J as $J = \bigcup \{f(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$. From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle$ and

$\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$, we have $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$. From $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$ and f is sound, we have

$$(ii) \quad j \in f(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u)).I \Rightarrow$$

$$\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle$$

From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle$, (ii) and Item (2) of Lemma (3.6.4), we have

$$(iii) \quad j \in f(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u)).I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle$$

From $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, (iii) and $J = \bigcup \{f(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$, we have

$$(iv) \quad j \in J \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle$$

From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and (iv), we have

$$(v) \quad j \in J \Rightarrow \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle \in \mathcal{R}$$

From (v), $\sigma = a' :: \sigma'$ and f is sound, we have

$$(vi) \quad j \in J \Rightarrow (j' \in f(j, \sigma, \text{iret}).I \Rightarrow \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{T}_{j'}, \sigma', a' \rangle)$$

From $\sigma = a' :: \sigma'$, (iv), (vi) and Item (3) of Lemma (3.6.4), we have

$$(vii) \quad j \in J \Rightarrow (j' \in f(j, \sigma, \text{iret}).I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{T}_{j'}, \sigma', a' \rangle)$$

which is

$$(viii) \quad j' \in \bigcup \{f(j, \sigma, \text{iret}).I \mid j \in J\} \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{T}_{j'}, \sigma', a' \rangle$$

From $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_{\xi_{\text{iret}}(i)}, \overline{T}_{\xi_{\text{iret}}(i)}, \sigma', a' \rangle$, (viii) and (i), we have

$$j \in \psi_{\mathcal{R}} f(i, \sigma, \text{iret}).I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a' \rangle.$$

2. h is $s; h'$.

From Item (5) of Definition (3.6.2), we have

$$(i) \quad \psi_{\mathcal{R}} f(i, \sigma, h).I = \bigcup \{f(j, \sigma, h').I \mid j \in J\}$$

where $J = f(i, \sigma, s; h').I$. From $\langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \in \mathcal{R}$ and f is sound, we have

$$(ii) \quad j \in f(i, \sigma, s; h').I \Rightarrow \langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, h' \rangle$$

From $J = f(i, \sigma, s; h').I$ and (ii), we have

$$(iii) \quad j \in J \Rightarrow \langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, h' \rangle$$

From $\langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \in \mathcal{R}$ and (iii), we have

$$(iv) \quad j \in J \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, h' \rangle \in \mathcal{R}$$

From (iv), $\sigma = a' :: \sigma'$ and f is sound, we have

$$(v) \quad j \in J \Rightarrow (j' \in f(j, \sigma, h').I \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, h' \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{\overline{T}}_{j'}, \sigma', a' \rangle)$$

From $\sigma = a' :: \sigma'$, (iii), (v) and Item (2) of Corollary (3.6.1), we have

$$(vi) \quad j \in J \Rightarrow (j' \in f(j, \sigma, h').I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, s; h' \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{\overline{T}}_{j'}, \sigma', a' \rangle)$$

which is

$$(vii) \quad j' \in \bigcup \{f(j, \sigma, h').I \mid j \in J\} \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, h' \rangle \xrightarrow{\sigma'} \langle imr_{j'}, \overline{\overline{T}}_{j'}, \sigma', a' \rangle$$

From (vii) and (i), we have

$$j \in \psi_{\mathcal{R}} f(i, \sigma, h).I \Rightarrow \langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle.$$

■

We define stack-irrelevancy of f on \mathcal{R} .

Definition 3.6.6 (Stack-irrelevancy on \mathcal{R}) f is stack-irrelevant on \mathcal{R} iff for all $i \in \mathcal{I}$, $a \in \mathcal{C}$ and for any $\sigma, \sigma' \in \mathcal{S}$, if $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma', a \rangle \in \mathcal{R}$, then we have

1. $f(i, \sigma, a).I = f(i, \sigma', a).I$
2. $f(i, \sigma, a).k - |\sigma| = f(i, \sigma', a).k - |\sigma'|$.

We next prove in Lemma (3.6.9) that fixed point $\mu\psi_{\mathcal{R}}f_0$ is sound and stack-irrelevant.

Proposition 3.6.6 If f is sound on \mathcal{R} and $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$, then

1. $j \in f(i, \sigma, s; a').I \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle \in \mathcal{R}$ if $a = s; a'$
2. $j \in f(i, \sigma, h).I \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle \in \mathcal{R}$ if $a = h$ and $\sigma = a' :: \sigma'$

Lemma 3.6.7 Suppose $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$,

1. if $a = s; a'$, s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , and f is sound, then we have

- (a) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow ((a::_S\sigma = a :: \sigma) \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$
 (b) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$
 (c) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow (j \in f(\zeta_u(i), s; a' :: \sigma, \overline{h}(u)).I \Rightarrow \langle \text{imr}_j, \overline{T}_j, \sigma, s; a' \rangle \in \mathcal{R})$

2. if $a = s; a'$ and s is if0 x then s_1 else s_2 , then we have $\langle \text{imr}_i, \overline{T}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$.

3. if $a = h$, h is ired and f is sound, then we have

- (a) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow ((\text{ired}::_S\sigma = \text{ired} :: \sigma) \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$
 (b) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{ired} :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$
 (c) $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow (j \in f(\zeta_u(i), \text{ired} :: \sigma, \overline{h}(u)).I \Rightarrow \langle \text{imr}_j, \overline{T}_j, \sigma, \text{ired} \rangle \in \mathcal{R})$

Proof 1. Suppose $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)$, $\eta(s; a') \in \mathcal{L}$ and Item (4) of Lemma (3.6.5), we have $a::_S\sigma = s; a' :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle$, we have $\langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$. From $\langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$, f is sound and Item (2) of Proposition (3.6.6), we have $j \in f(\zeta_u(i), s; a' :: \sigma, \overline{h}(u)).I \Rightarrow \langle \text{imr}_j, \overline{T}_j, \sigma, s; a' \rangle \in \mathcal{R}$.

2. From $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, s is if0 x then s_1 else s_2 and Item (2) of Lemma (3.6.5), we have $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \overline{T}_i, \sigma, s_1; a' \rangle$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \overline{T}_i, \sigma, s_2; a' \rangle$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \overline{T}_i, \sigma, s_1; a' \rangle$, we have $\langle \text{imr}_i, \overline{T}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle \text{imr}_i, \overline{T}_i, \sigma, s_2; a' \rangle$, we have $\langle \text{imr}_i, \overline{T}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$.

3. Suppose $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{ired} \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)$, $\eta(\text{ired}) \in \mathcal{L}$ and Item (4) of Lemma (3.6.5), we have $\text{ired}::_S\sigma = \text{ired} :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{ired} \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{ired} :: \sigma, \overline{h}(u) \rangle$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{ired} \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{ired} \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{ired} :: \sigma, \overline{h}(u) \rangle$,

we have $\langle imr_{\zeta_u(i)}, \overline{\widehat{T}}_{\zeta_u(i)}, iret :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$. From $\langle imr_{\zeta_u(i)}, \overline{\widehat{T}}_{\zeta_u(i)}, iret :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$, f is sound and Item (2) of Proposition (3.6.6), we have $j \in f(\zeta_u(i), iret :: \sigma, \bar{h}(u)).I \Rightarrow \langle imr_j, \overline{\widehat{T}}_j, \sigma, iret \rangle \in \mathcal{R}$.

■

Lemma 3.6.8 *If f is sound and stack-irrelevant on \mathcal{R} , then $\psi_{\mathcal{R}}f$ is also sound and stack-irrelevant.*

Proof From f is sound on \mathcal{R} and Lemma (3.6.6), we have $\psi_{\mathcal{R}}f$ is also sound. From Definition (3.6.6), we need to further prove that for all $i \in \mathcal{I}$ and $a \in \mathcal{C}$ and for any $\sigma, \sigma' \in \mathcal{S}$, if $\langle imr_i, \overline{\widehat{T}}_i, \sigma, a \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{\widehat{T}}_i, \sigma', a \rangle \in \mathcal{R}$, then we have

1. $\psi_{\mathcal{R}}f(i, \sigma, a).I = \psi_{\mathcal{R}}f(i, \sigma', a).I$
2. $\psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'|$

We prove by case analysis on the form of a . If the form of a is $s; a'$, there are five cases depending on the form of s .

1. If $a = s; a'$ and s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$.

From $\langle imr_i, \overline{\widehat{T}}_i, \sigma, s; a' \rangle \in \mathcal{R}$, s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, and Item (1) of Lemma (3.6.5), we have $\alpha(\xi_s(i)) = \xi_s(i)$.

From $\langle imr_i, \overline{\widehat{T}}_i, \sigma, s; a' \rangle \in \mathcal{R}$, s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, f is sound and Item (1) of Lemma (3.6.7), we have

$$(i) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow ((a ::_{\mathcal{S}} \sigma = a :: \sigma) \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$$

$$(ii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow \langle imr_{\zeta_u(i)}, \overline{\widehat{T}}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$$

$$(iii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow$$

$$(j \in f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \Rightarrow \langle imr_j, \overline{\widehat{T}}_j, \sigma, s; a' \rangle \in \mathcal{R}).$$

From $\langle imr_i, \overline{\widehat{T}}_i, \sigma', s; a' \rangle \in \mathcal{R}$, s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, f is sound and Item (1) of Lemma (3.6.7), we have

$$(iv) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow ((a ::_{\mathcal{S}} \sigma' = a :: \sigma') \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$$

$$(v) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow \langle imr_{\zeta_u(i)}, \overline{\widehat{T}}_{\zeta_u(i)}, s; a' :: \sigma', \bar{h}(u) \rangle \in \mathcal{R}$$

$$(vi) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow$$

$$(j \in f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).I \Rightarrow \langle imr_j, \overline{\widehat{T}}_j, \sigma', s; a' \rangle \in \mathcal{R}).$$

From $\alpha(\xi_s(i)) = \xi_s(i)$, (i) and Item (1) of Definition (3.6.2), we have

$$(vii) \quad \psi_{\mathcal{R}}f(i, \sigma, a).I = \bigcup \{f(j, \sigma, s; a').I \mid j \in J\} \cup \{ \xi_s(i) \}$$

$$(viii) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k = \max(\{f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\} \cup \{f(j, \sigma, s; a').k \mid j \in J\} \cup \{|\sigma|\})$$

where $J = \bigcup \{f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\}$. From $\alpha(\xi_s(i)) = \xi_s(i)$, (iv) and Item (1) of Definition (3.6.2), we have

$$(ix) \quad \psi_{\mathcal{R}}f(i, \sigma', a).I = \bigcup \{f(j, \sigma', s; a').I \mid j \in J'\} \cup \{ \xi_s(i) \}$$

$$(x) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k = \max(\{f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).k \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\} \cup \{f(j, \sigma', s; a').k \mid j \in J'\} \cup \{|\sigma'|\})$$

where $J' = \bigcup \{f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\}$. From (ii),(v) and f is stack-irrelevant, we have

$$(xi) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I = f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).I$$

$$(xii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow$$

$$(f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).k - |s; a' :: \sigma| = f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).k - |s; a' :: \sigma'|)$$

From (xi), $J = \bigcup \{f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\}$ and $J' = \bigcup \{f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\}$, we have $J = J'$.

From (iii), (vi), (xi) and f is stack-irrelevant, we have

$$(xiii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow$$

$$(j \in f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \Rightarrow f(j, \sigma, s; a').I = f(j, \sigma', s; a').I)$$

$$(xiv) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i) \Rightarrow (j \in f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).I \Rightarrow$$

$$f(j, \sigma, s; a').k - |\sigma| = f(j, \sigma', s; a').k - |\sigma'|)$$

From (xiii), (vii), (ix) and $J = J'$, we have $\psi_{\mathcal{R}}f(i, \sigma, a).I = \psi_{\mathcal{R}}f(i, \sigma', a).I$, which is Item (1).

From (viii), by subtracting $|\sigma|$ on both sides of the equation, we have

$$(xv) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| =$$

$$\max(\{(f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).k - |\sigma|) \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\widehat{T}}_i)\} \cup$$

$$\{(f(j, \sigma, s; a').k - |\sigma|) \mid j \in J\} \cup \{0\})$$

From (x), by subtracting $|\sigma'|$ on both sides of the equation, we have

$$(xvi) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'| = \\ \max(\{(f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).k - |\sigma'|) \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i)\} \cup \\ \{(f(j, \sigma', s; a').k - |\sigma'|) \mid j \in J'\} \cup \{0\})$$

From (xii), by subtracting 1 on both sides of the equation, we have

$$(xvii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i) \Rightarrow \\ f(\zeta_u(i), s; a' :: \sigma, \bar{h}(u)).k - |\sigma| = f(\zeta_u(i), s; a' :: \sigma', \bar{h}(u)).k - |\sigma'|$$

From (xiv),(xv), (xvi), (xvii) and $J = J'$, we have $\psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'|$, which is Item (2).

2. If $a = s; a'$ and s is if0 x then s_1 else s_2 .

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \in \mathcal{R}$, s is if0 x then s_1 else s_2 and Item (2) of Lemma (3.6.7), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma', s; a' \rangle \in \mathcal{R}$, s is if0 x then s_1 else s_2 and Item (2) of Lemma (3.6.7), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; a' \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma', s_2; a' \rangle \in \mathcal{R}$.

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; a' \rangle \in \mathcal{R}$, $\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; a' \rangle \in \mathcal{R}$ and f is stack-irrelevant, we have

$$(i) \quad f(i, \sigma, s_1; a').I = f(i, \sigma', s_1; a').I \\ (ii) \quad f(i, \sigma, s_1; a').k - |\sigma| = f(i, \sigma', s_1; a').k - |\sigma'|$$

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_2; a' \rangle \in \mathcal{R}$, $\langle imr_i, \overline{\overline{T}}_i, \sigma', s_2; a' \rangle \in \mathcal{R}$ and f is stack-irrelevant, we have

$$(iii) \quad f(i, \sigma, s_2; a').I = f(i, \sigma', s_2; a').I \\ (iv) \quad f(i, \sigma, s_2; a').k - |\sigma| = f(i, \sigma', s_2; a').k - |\sigma'|$$

From Item (2) of Definition (3.6.2), we have

$$(v) \quad \psi_{\mathcal{R}}f(i, \sigma, a).I = f(i, \sigma, s_1; a').I \cup f(i, \sigma, s_2; a').I \\ (vi) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k = \max(\{f(i, \sigma, s_1; a').k, f(i, \sigma, s_2; a').k\}) \\ (vii) \quad \psi_{\mathcal{R}}f(i, \sigma', a).I = f(i, \sigma', s_1; a').I \cup f(i, \sigma', s_2; a').I \\ (viii) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k = \max(\{f(i, \sigma', s_1; a').k, f(i, \sigma', s_2; a').k\})$$

From (i) and (iii), we have

$$(ix) \quad f(i, \sigma, s_1; a').I \cup f(i, \sigma, s_2; a').I = f(i, \sigma', s_1; a').I \cup f(i, \sigma', s_2; a').I$$

From (ix),(v),(vii), we have $\psi_{\mathcal{R}}f(i, \sigma, a).I = \psi_{\mathcal{R}}f(i, \sigma', a).I$, which is Item (1).

From (ii) and (iv), we have

$$(x) \quad \max(\{f(i, \sigma, s_1; a').k - |\sigma|, f(i, \sigma, s_2; a').k - |\sigma|\}) = \\ \max(\{f(i, \sigma', s_1; a').k - |\sigma'|, f(i, \sigma', s_2; a').k - |\sigma'|\})$$

From (vi), by subtracting $|\sigma|$ on both sides of the equation, we have

$$(xi) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \\ \max(\{f(i, \sigma, s_1; a').k - |\sigma|, f(i, \sigma, s_2; a').k - |\sigma|\})$$

From (viii), by subtracting $|\sigma'|$ on both sides of the equation, we have

$$(xii) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'| = \\ \max(\{f(i, \sigma', s_1; a').k - |\sigma'|, f(i, \sigma', s_2; a').k - |\sigma'|\})$$

From (x), (xi) and (xii), we have $\psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'|$, which is Item (2).

3. If $a = s; a'$ and s is $s_1; s_2$.

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, (s_1; s_2); a' \rangle = \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle$ and

$\langle imr_i, \overline{\overline{T}}_i, \sigma, (s_1; s_2); a' \rangle \in \mathcal{R}$, we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$, f is sound and Item (1) of Proposition (3.6.6), we have

$$(i) \quad j \in J \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, s_2; a' \rangle \in \mathcal{R}$$

where $J = f(i, \sigma, s_1; (s_2; a')).I$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma', (s_1; s_2); a' \rangle =$

$\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; (s_2; a') \rangle$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma', (s_1; s_2); a' \rangle \in \mathcal{R}$, we have

$\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; (s_2; a') \rangle \in \mathcal{R}$. From

$\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; (s_2; a') \rangle \in \mathcal{R}$, f is sound and Item (1) of Proposition (3.6.6), we have

$$(ii) \quad j \in J' \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma', s_2; a' \rangle \in \mathcal{R}.$$

where $J' = f(i, \sigma', s_1; (s_2; a')).I$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a') \rangle \in \mathcal{R}$,

$\langle imr_i, \overline{\overline{T}}_i, \sigma', s_1; (s_2; a') \rangle \in \mathcal{R}$ and f is stack-irrelevant, we have

$$(iii) \quad f(i, \sigma, s_1; (s_2; a')).I = f(i, \sigma', s_1; (s_2; a')).I$$

$$(iv) \quad f(i, \sigma, s_1; (s_2; a')).k - |\sigma| = f(i, \sigma', s_1; (s_2; a')).k - |\sigma'|$$

From $J = f(i, \sigma, s_1; (s_2; a')).I$, $J' = f(i, \sigma', s_1; (s_2; a')).I$ and (iii), we have

$J = J'$.

From $J = J'$, (i), (ii) and f is stack-irrelevant, we have

$$(v) \quad j \in J \Rightarrow f(j, \sigma, s_2; a').I = f(j, \sigma', s_2; a').I$$

$$(vi) \quad j \in J \Rightarrow f(j, \sigma, s_2; a').k - |\sigma| = f(j, \sigma', s_2; a').k - |\sigma'|$$

From Item (3) of Definition (3.6.2), we have

$$(vii) \quad \psi_{\mathcal{R}}f(i, \sigma, a).I = \bigcup \{f(j, \sigma, s_2; a').I \mid j \in J\}$$

$$(viii) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k = \max(\{f(i, \sigma, s_1; (s_2; a')).k\} \cup \{f(j, \sigma, s_2; a').k \mid j \in J\})$$

$$(ix) \quad \psi_{\mathcal{R}}f(i, \sigma', a).I = \bigcup \{f(j, \sigma', s_2; a').I \mid j \in J'\}$$

$$(x) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k = \max(\{f(i, \sigma', s_1; (s_2; a')).k\} \cup \{f(j, \sigma', s_2; a').k \mid j \in J'\})$$

From $J = J'$, (vii),(ix) and (v), we have $\psi_{\mathcal{R}}f(i, \sigma, a).I = \psi_{\mathcal{R}}f(i, \sigma', a).I$, which is Item (1).

From (viii), by subtracting $|\sigma|$ on both sides of the equation, we have

$$(xi) \quad \psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \max(\{f(i, \sigma, s_1; (s_2; a')).k - |\sigma|\} \cup \{f(j, \sigma, s_2; a').k - |\sigma| \mid j \in J\})$$

From (x), by subtracting $|\sigma'|$ on both sides of the equation, we have

$$(xii) \quad \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'| = \max(\{f(i, \sigma', s_1; (s_2; a')).k - |\sigma'|\} \cup \{f(j, \sigma', s_2; a').k - |\sigma'| \mid j \in J'\})$$

From $J = J'$, (iv), (vi), (xi) and (xii), we have $\psi_{\mathcal{R}}f(i, \sigma, a).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', a).k - |\sigma'|$, which is Item (2).

4. If $a = h$ and h is iret.

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle$ is consistent, from which it follows $\sigma = a' :: \sigma''$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$, $\sigma = a' :: \sigma''$, and Item (3) of Lemma (3.6.5), we have $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$. (We have exactly the same $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$ from $\langle imr_i, \overline{\overline{T}}_i, \sigma', \text{iret} \rangle \in \mathcal{R}$.)

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$, f is sound and Item (3) of Lemma (3.6.7), we have

$$(i) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i) \Rightarrow ((\text{iret}::s\sigma = \text{iret} :: \sigma) \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$$

$$(ii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i) \Rightarrow \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$$

$$(iii) \quad u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i) \Rightarrow$$

$$(j \in f(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u)).I \Rightarrow \langle imr_j, \overline{\overline{T}}_j, \sigma, \text{iret} \rangle \in \mathcal{R}).$$

From $\langle imr_i, \overline{\overline{T}}_i, \sigma', \text{iret} \rangle \in \mathcal{R}$, f is sound and Item (3) of Lemma (3.6.7), we have

- (iv) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow ((\text{iret}::_{\mathcal{S}}\sigma' = \text{iret}::\sigma') \wedge (\alpha(\zeta_u(i)) = \zeta_u(i)))$
- (v) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, \text{iret}::\sigma', \overline{h}(u) \rangle \in \mathcal{R}$
- (vi) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow$

$$(j \in f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).I \Rightarrow \langle imr_j, \overline{T}_j, \sigma', \text{iret} \rangle \in \mathcal{R}).$$

From $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$, (i) and Item (4) of Definition (3.6.2), we have

- (vii) $\psi_{\mathcal{R}}f(i, \sigma, \text{iret}).I = \bigcup \{f(j, \sigma, \text{iret}).I \mid j \in J\} \cup \{ \xi_{\text{iret}}(i) \}$
- (viii) $\psi_{\mathcal{R}}f(i, \sigma, \text{iret}).k = \max(\{f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).k \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\} \cup \{f(j, \sigma, \text{iret}).k \mid j \in J\} \cup \{|\sigma|\})$

where $J = \bigcup \{f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$. From $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$, (iv) and Item (4) of Definition (3.6.2), we have

- (ix) $\psi_{\mathcal{R}}f(i, \sigma', \text{iret}).I = \bigcup \{f(j, \sigma', \text{iret}).I \mid j \in J'\} \cup \{ \xi_{\text{iret}}(i) \}$
- (x) $\psi_{\mathcal{R}}f(i, \sigma', \text{iret}).k = \max(\{f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).k \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\} \cup \{f(j, \sigma', \text{iret}).k \mid j \in J'\} \cup \{|\sigma'|\})$

where $J' = \bigcup \{f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$. From (ii),(v) and f is stack-irrelevant, we have

- (xi) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).I = f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).I$
- (xii) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow$

$$(f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).k - |\text{iret}::\sigma| = f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).k - |\text{iret}::\sigma'|)$$

From $J = \bigcup \{f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$, $J' = \bigcup \{f(\zeta_u(i), \text{iret}::\sigma', \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)\}$ and (xi), we have $J = J'$.

From (iii), (vi), (xi) and f is stack-irrelevant, we have

- (xiii) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow$
- $(j \in f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).I \Rightarrow f(j, \sigma, \text{iret}).I = f(j, \sigma', \text{iret}).I$
- (xiv) $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow (j \in f(\zeta_u(i), \text{iret}::\sigma, \overline{h}(u)).I \Rightarrow$

$$f(j, \sigma, \text{iret}).k - |\sigma| = f(j, \sigma', \text{iret}).k - |\sigma'|)$$

From (xiii), (vii), (ix) and $J = J'$, we have $\psi_{\mathcal{R}}f(i, \sigma, \text{iret}).I = \psi_{\mathcal{R}}f(i, \sigma', \text{iret}).I$, which is Item (1).

From (viii), by subtracting $|\sigma|$ on both sides of the equation, we have

$$(xv) \quad \psi_{\mathcal{R}}f(i, \sigma, \text{iret}).k - |\sigma| = \\ \max(\{(f(\zeta_u(i), \text{iret} :: \sigma, \bar{h}(u)).k - |\sigma|) \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\} \cup \\ \{f(j, \sigma, \text{iret}).k - |\sigma| \mid j \in J\} \cup \{0\})$$

From (x), by subtracting $|\sigma'|$ on both sides of the equation, we have

$$(xvi) \quad \psi_{\mathcal{R}}f(i, \sigma', \text{iret}).k - |\sigma'| = \\ \max(\{(f(\zeta_u(i), \text{iret} :: \sigma', \bar{h}(u)).k - |\sigma'|) \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)\} \cup \\ \{f(j, \sigma', \text{iret}).k - |\sigma'| \mid j \in J'\} \cup \{0\})$$

From (xii), by subtracting 1 on both sides of the equation, we have

$$(xvii) \quad u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \Rightarrow \\ f(\zeta_u(i), \text{iret} :: \sigma, \bar{h}(u)).k - |\sigma| = f(\zeta_u(i), \text{iret} :: \sigma', \bar{h}(u)).k - |\sigma'|$$

From (xiv), (xv), (xvi), (xvii) and $J = J'$, we have $\psi_{\mathcal{R}}f(i, \sigma, \text{iret}).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', \text{iret}).k - |\sigma'|$, which is Item (2).

5. If $a = h$ and h is $s; h'$.

From $\langle \text{imr}_i, \bar{T}_i, \sigma, s; h' \rangle \in \mathcal{R}$ and f is sound, we have

$$(i) \quad j \in J \Rightarrow \langle \text{imr}_j, \bar{T}_j, \sigma, h' \rangle \in \mathcal{R}$$

where $J = f(i, \sigma, s; h').I$. From $\langle \text{imr}_i, \bar{T}_i, \sigma', s; h' \rangle \in \mathcal{R}$ and f is sound, we have

$$(ii) \quad j \in J' \Rightarrow \langle \text{imr}_j, \bar{T}_j, \sigma', h' \rangle \in \mathcal{R}$$

where $J' = f(i, \sigma', s; h').I$. From $\langle \text{imr}_i, \bar{T}_i, \sigma, s; h' \rangle \in \mathcal{R}$, $\langle \text{imr}_i, \bar{T}_i, \sigma', s; h' \rangle \in \mathcal{R}$ and f is stack-irrelevant, we have

$$(iii) \quad f(i, \sigma, s; h').I = f(i, \sigma', s; h').I$$

$$(iv) \quad f(i, \sigma, s; h').k - |\sigma| = f(i, \sigma', s; h').k - |\sigma'|$$

From (i), (ii) and (iii), we have $J = J'$.

From $J = J'$, (i), (ii) and f is stack-irrelevant, we have

$$(v) \quad j \in J \Rightarrow f(j, \sigma, h').I = f(j, \sigma', h').I$$

$$(vi) \quad j \in J' \Rightarrow f(j, \sigma, h').k - |\sigma| = f(j, \sigma', h').k - |\sigma'|$$

From Item (5) of Definition (3.6.2), we have

$$(vii) \quad \psi_{\mathcal{R}}f(i, \sigma, h).I = \bigcup \{f(j, \sigma, h').I \mid j \in J\}$$

$$(viii) \quad \psi_{\mathcal{R}}f(i, \sigma, h).k = \max(\{f(i, \sigma, s; h').k\} \cup \{f(j, \sigma, h').k \mid j \in J\})$$

$$(ix) \quad \psi_{\mathcal{R}}f(i, \sigma', h).I = \bigcup \{f(j, \sigma', h').I \mid j \in J'\}$$

$$(x) \quad \psi_{\mathcal{R}}f(i, \sigma', h).k = \max(\{f(i, \sigma', s; h').k\} \cup \{f(j, \sigma', h').k \mid j \in J'\})$$

From $J = J'$, (vii),(ix) and (v), we have $\psi_{\mathcal{R}}f(i, \sigma, h).I = \psi_{\mathcal{R}}f(i, \sigma', h).I$, which is Item (1).

From (viii), by subtracting $|\sigma|$ on both sides of the equation, we have

$$(xi) \quad \psi_{\mathcal{R}}f(i, \sigma, h).k - |\sigma| =$$

$$\max(\{f(i, \sigma, s; h').k - |\sigma|\} \cup \{f(j, \sigma, h').k - |\sigma| \mid j \in J\})$$

From (x), by subtracting $|\sigma'|$ on both sides of the equation, we have

$$(xii) \quad \psi_{\mathcal{R}}f(i, \sigma', h).k - |\sigma'| =$$

$$\max(\{f(i, \sigma', s; h').k - |\sigma'|\} \cup \{f(j, \sigma', h').k - |\sigma'| \mid j \in J'\})$$

From $J = J'$, (iv), (vi), (xi) and (xii), we have

$$\psi_{\mathcal{R}}f(i, \sigma, h).k - |\sigma| = \psi_{\mathcal{R}}f(i, \sigma', h).k - |\sigma'|, \text{ which is Item (2).}$$

■

Lemma 3.6.9 $\mu\psi_{\mathcal{R}}$ is sound and stack-irrelevant.

Proof From $\mu\psi_{\mathcal{R}}$ is the fixed point $\mu\psi_{\mathcal{R}}f_0$, we have that $\mu\psi_{\mathcal{R}}f_0 = \psi_{\mathcal{R}}^m f_0$, for some $m \geq 0$, for which $\psi_{\mathcal{R}}^m f_0 = \psi_{\mathcal{R}}^{m+1} f_0$. It is sufficient to prove by induction on n that, for all $n \geq 0$, $\psi_{\mathcal{R}}^n f_0$ is sound and stack-irrelevant.

- Base case: $n = 0$. We have $\psi_{\mathcal{R}}^0 f_0 = f_0$. f_0 is trivially sound and stack-irrelevant.
- Induction Step. From the induction hypothesis, we have that $\psi_{\mathcal{R}}^n f_0$ is sound and stack-irrelevant. From $\psi_{\mathcal{R}}^n f_0$ is sound and stack-irrelevant and Lemma (3.6.8), we have that $\psi_{\mathcal{R}}^{n+1} f_0$ is also sound and stack-irrelevant.

■

Let $f \in D$. We define the completeness of f on \mathcal{R} as follows.

Definition 3.6.7 (Completeness of f on \mathcal{R}) f is complete on \mathcal{R} iff for all $\sigma \in \mathcal{S}$, $a \in \mathcal{C}$ and $i \in \mathcal{I}$, if $\langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}$, then

1. $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a' \rangle \Rightarrow j \in f(i, \sigma, s; a')$. I if $a = s; a'$
2. $\langle imr_i, \overline{T}_i, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a' \rangle \Rightarrow j \in f(i, \sigma, h)$. I if $a = h$ and $\sigma = a' :: \sigma'$

We simply say that f is complete if it is clear from the context which \mathcal{R} we refer to.

We now prove in Lemma (3.6.11) that the fixed point $\mu\psi_{\mathcal{R}}$ is complete.

Lemma 3.6.10 *We have*

1. if $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ and s is if0 x then s_1 else s_2 , then we have either $\langle imr_i, \overline{T}_i, \sigma, s_1; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ or $\langle imr_i, \overline{T}_i, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$.
2. if $\langle imr_i, \overline{T}_i, \sigma, (s_1; s_2); a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$, then $\exists i'$ such that $\langle imr_i, \overline{T}_i, \sigma, (s_1; s_2); a \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$.
3. if $\sigma = a :: \sigma'$ and $\langle imr_i, \overline{T}_i, \sigma, s; h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a \rangle$, then $\exists i'$ such that $\langle imr_i, \overline{T}_i, \sigma, s; h \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, h \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a \rangle$.

Proof 1. From if0 x then s_1 else s_2 is instantaneous and Rules (3.16) and (3.17),

we have the only two out-going edges from $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle$ are

- (i) $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma} \langle imr_i, \overline{T}_i, \sigma, s_1; a \rangle$
- (ii) $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma} \langle imr_i, \overline{T}_i, \sigma, s_2; a \rangle$

From $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$, and (i) and (ii), it follows either $\langle imr_i, \overline{T}_i, \sigma, s_1; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ or $\langle imr_i, \overline{T}_i, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$.

2. From $\langle imr_i, \overline{T}_i, \sigma, (s_1; s_2); a \rangle = \langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle$, and

$\langle imr_i, \overline{T}_i, \sigma, (s_1; s_2); a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$, we have

- (i) $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma, a \rangle$

From (i), we have path $\pi : \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \hookrightarrow^* \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$ and $(\sigma,)^\sigma, (a::\sigma,)^{a::\sigma} \notin \Pi(\pi)$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \hookrightarrow^* \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$ and Item (1) of Proposition (3.6.4), there exists i_1 along π such that we have sub-path $\pi_1: \langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \hookrightarrow^* \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle$ and sub-path $\pi_2: \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle \hookrightarrow^* \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$. From $(\sigma,)^\sigma \notin \Pi(\pi)$ and π is the concatenation of π_1 and π_2 , we have $(\sigma,)^\sigma \notin \Pi(\pi_1)$ and $(\sigma,)^\sigma, (a::\sigma,)^{a::\sigma} \notin \Pi(\pi_2)$. From $(\sigma,)^\sigma \notin \Pi(\pi_1)$, we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle$ and Item (2) of Proposition (3.6.5), there exists i' such that $\langle imr_i, \overline{\overline{T}}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s_2; a \rangle$ and $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle$. From $(\sigma,)^\sigma, (a::\sigma,)^{a::\sigma} \notin \Pi(\pi_2)$ and π_2 , we have $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$. From $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle$, $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$, and Item (1) of Lemma (3.6.4), we have $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$.

3. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$, we have path $\pi: \langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \hookrightarrow^* \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$ and $(\sigma,)^\sigma, (\sigma' \notin \Pi(\pi))$. From $\sigma = a :: \sigma'$, π and Item (2) of Proposition (3.6.4), there exists i_1 along π such that we have sub-path $\pi_1 : \langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \hookrightarrow^* \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle$ and sub-path $\pi_2 : \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle \hookrightarrow^* \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$. From $(\sigma,)^\sigma, (\sigma' \notin \Pi(\pi))$ and π is the concatenation of π_1 and π_2 , we have $(\sigma,)^\sigma \notin \Pi(\pi_1)$ and $(\sigma,)^\sigma, (\sigma' \notin \Pi(\pi_2))$. From $(\sigma,)^\sigma \notin \Pi(\pi_1)$ and π_1 , we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle$ and Item (2) of Proposition (3.6.5), there exists i' such that $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h \rangle$ and $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle$. From $(\sigma,)^\sigma, (\sigma' \notin \Pi(\pi_2))$ and π_2 , we have $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$. From $\sigma = a :: \sigma'$, $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h \rangle \xrightarrow{\sigma+} \langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle$, $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$ and Item (3) of Lemma (3.6.4), we have $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a \rangle$.

■

Lemma 3.6.11 $\mu\psi_{\mathcal{R}}$ is complete on \mathcal{R} .

Proof From Definition (3.6.7), we need to prove for all $\sigma \in \mathcal{S}$, $a \in \mathcal{C}$ and $i \in \mathcal{I}$, if $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$, then

1. $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle \Rightarrow j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$ if $a = s; a'$
2. $\langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle \Rightarrow j \in \mu\psi_{\mathcal{R}}(i, \sigma, h).I$ if $a = h$ and $\sigma = a' :: \sigma'$

We prove by induction on the structure of $a \in \mathcal{C}$. If the form of a is $s; a'$, we need to prove Item (1). If the form of a is h , we need to prove Item (2). There are five cases depending upon the form of s .

1. If $a = s; a'$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip .

We have

$$(i) \quad \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$$

It is clear that there are 0 or more $(s; a' :: \sigma,)^{s; a' :: \sigma}$ pairs along the path. Let n be the number of the pairs. We prove by induction on n .

Base case ($n = 0$). There is no $(s; a' :: \sigma,)^{s; a' :: \sigma}$ pair, which means no interrupts along the path. From Rule (3.13)-(3.15) and (3.18), depending on what s is, we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle$, where $j = \xi_s(i)$. From $\langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle \in \mathcal{R}$, we have $j = \alpha(\xi_s(i)) = \xi_s(i)$. From $j = \alpha(\xi_s(i)) = \xi_s(i)$ and Item (1) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$.

Induction Step. Suppose there are $n + 1$ of $(s; a' :: \sigma,)^{s; a' :: \sigma}$ pairs along the path.

Considering the first one, we have

$$(ii) \quad \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle,$$

$$\text{where } u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i)$$

$$(iii) \quad \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, s; a' :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} Q \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s; a' \rangle$$

$$(iv) \quad \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, s; a' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma, a' \rangle,$$

along which there are n of $(s; a' :: \sigma,)^{s; a' :: \sigma}$ pairs

From (iii) and the induction hypothesis on a , we have

$$(v) \quad i' \in \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a' :: \sigma, \bar{h}(u))$$

From (iv) and the induction hypothesis on n , we have

$$(vi) \quad j \in \mu\psi_{\mathcal{R}}(i', \sigma, s; a').I$$

From $\langle imr_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)}, s; a' :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$, we have

$$(vii) \quad \alpha(\zeta_u(i)) = \zeta_u(i) \text{ and } s; a' ::_S \sigma = s; a' :: \sigma$$

From (v), (vi), (vii), $u \in \widehat{\mathcal{P}}(imr_i, \bar{T}_i)$ and Item (1) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$.

2. If $a = s; a'$ and s is if0 x then s_1 else s_2 .

We have

$$(i) \quad \langle imr_i, \bar{T}_i, \sigma, s; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \bar{T}_j, \sigma, a' \rangle$$

From (i), s is if0 x then s_1 else s_2 and Item (1) of Lemma (3.6.10), we have either

$$(ii) \quad \langle imr_i, \bar{T}_i, \sigma, s_1; a' \rangle \xrightarrow{\sigma} \langle imr_j, \bar{T}_j, \sigma, a' \rangle$$

or

$$(iii) \quad \langle imr_i, \bar{T}_i, \sigma, s_2; a' \rangle \xrightarrow{\sigma} \langle imr_j, \bar{T}_j, \sigma, a' \rangle$$

If we have (ii), then from the induction hypothesis, we have

$$(iv) \quad j \in \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').I$$

From (iv) and Item (2) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$.

If we have (iii), then from the induction hypothesis, we have

$$(v) \quad j \in \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').I$$

From (v) and Item (2) of Proposition (3.6.1), we also have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$.

3. If $a = s; a'$ and s is $s_1; s_2$.

We have

$$(i) \quad \langle imr_i, \bar{T}_i, \sigma, (s_1; s_2); a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \bar{T}_j, \sigma, a' \rangle$$

From (i) and Item (2) of Lemma (3.6.10), we have $\exists i'$ such that

$$(ii) \quad \langle imr_i, \bar{T}_i, \sigma, (s_1; s_2); a' \rangle \xrightarrow{\sigma^*} \langle imr_{i'}, \bar{T}_{i'}, \sigma, s_2; a' \rangle$$

$$(iii) \quad \langle imr_{i'}, \bar{T}_{i'}, \sigma, s_2; a' \rangle \xrightarrow{\sigma^*} \langle imr_j, \bar{T}_j, \sigma, a' \rangle$$

From (ii) and the induction hypothesis, we have

$$(iv) \quad i' \in \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a').I$$

From (iii) and the induction hypothesis, we have

$$(v) \quad j \in \mu\psi_{\mathcal{R}}(i', \sigma, s_2; a').I$$

From (iv), (v) and Item (3) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a').I$.

4. If h is iret.

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have that $\langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle$ is consistent, from which it follows $\sigma = a' :: \sigma'$. We thus have

$$(i) \quad \langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle$$

It is clear that there are 0 or more $(\text{iret}::\sigma,)^{\text{iret}::\sigma}$ pairs along the path. Let n be the number of the pairs. We prove by induction on n .

Base case ($n = 0$). There is no $(\text{iret}::\sigma,)^{\text{iret}::\sigma}$ pair, which means no interrupts along the path. From Rule (3.11), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle$, where $j = \xi_{\text{iret}}(i)$. From $\langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle \in \mathcal{R}$, we have $j = \alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$. From $j = \alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$ and Item (4) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I$.

Induction Step. Suppose there are $n + 1$ of $(\text{iret}::\sigma,)^{\text{iret}::\sigma}$ pairs along the path.

Considering the first one, we have

$$(ii) \quad \langle imr_i, \overline{\overline{T}}_i, \sigma, \text{iret} \rangle \hookrightarrow \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle, \text{ where } u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i)$$

$$(iii) \quad \langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} Q \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \text{iret} \rangle$$

$$(iv) \quad \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \text{iret} \rangle \xrightarrow{\sigma} \langle imr_j, \overline{\overline{T}}_j, \sigma', a' \rangle,$$

along which there are n of $(\text{iret}::\sigma,)^{\text{iret}::\sigma}$ pairs

From (iii) and the induction hypothesis on a , we have

$$(v) \quad i' \in \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret} :: \sigma, \overline{h}(u))$$

From (iv) and the induction hypothesis on n , we have

$$(vi) \quad j \in \mu\psi_{\mathcal{R}}(i', \sigma, \text{iret}).I$$

From $\langle imr_{\zeta_u(i)}, \overline{\overline{T}}_{\zeta_u(i)}, \text{iret} :: \sigma, \overline{h}(u) \rangle \in \mathcal{R}$, we have

$$(vii) \quad \alpha(\zeta_u(i)) = \zeta_u(i) \text{ and } \text{iret}::_S\sigma = \text{iret} :: \sigma$$

From (v), (vi), (vii), $u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i)$ and Item (4) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I$.

5. If $h = s; h'$.

From $\langle imr_i, \overline{T}_i, \sigma, h \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have that $\langle imr_i, \overline{T}_i, \sigma, h \rangle$ is consistent, from which it follows $\sigma = a' :: \sigma'$. We thus have

$$(i) \quad \langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_j, \overline{T}_j, \sigma', a' \rangle$$

From (i) and Item (3) of Lemma (3.6.10), we have $\exists i'$ such that

$$(ii) \quad \langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, h' \rangle$$

$$(iii) \quad \langle imr_{i'}, \overline{T}_{i'}, \sigma, h' \rangle \xrightarrow{\sigma'} \langle imr_j, \overline{T}_j, \sigma', a' \rangle$$

From (ii) and the induction hypothesis, we have

$$(iv) \quad i' \in \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$$

From (iii) and the induction hypothesis, we have

$$(v) \quad j \in \mu\psi_{\mathcal{R}}(i', \sigma, h').I$$

From (iv), (v) and Item (5) of Proposition (3.6.1), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, h).I$. ■

Proposition 3.6.7 *We have*

1. if s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , and path $\pi: \langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$, then there are $n \geq 0$ of $(s; a :: \sigma,)^{s; a :: \sigma}$ pairs along path π .
2. if path $\pi: \langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle$, then there are $n \geq 0$ of $(\text{iret} :: \sigma,)^{\text{iret} :: \sigma}$ pairs along path π .

Lemma 3.6.12 *We have*

1. if s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$, then $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$.
2. if $\langle imr_i, \overline{T}_i, \sigma, \text{iret} \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, \text{iret} \rangle$, then $\mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k$.

Proof We prove the two items individually.

1. From $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$, we have path $\pi : \langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{*} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. From s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**, $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$ and Item (1) of Proposition (3.6.7), we have there are $n \geq 0$ of $(s; a::\sigma,)^{s; a::\sigma}$ pairs along path π . We prove by induction on n .

Base case $n = 0$. There are no interrupts along π , in which situation the path π is of 0 length: $i = j$. It follows $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$.

Induction step. Suppose there are $n + 1$ of $(s; a::\sigma,)^{s; a::\sigma}$ pairs along path π . Consider the first pair along the path π , we have

- (i) $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \hookrightarrow \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a :: \sigma, \overline{h}(u) \rangle$
for some $u \in 1..N$ such that $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$
- (ii) Sub-path $\pi_1: \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a :: \sigma, \overline{h}(u) \rangle \xrightarrow{*} \langle imr_{i_1}, \overline{T}_{i_1}, \sigma, s; a \rangle$
along which there is no $(s; a::\sigma,)^{s; a::\sigma}$
- (iii) Sub-path $\pi_2: \langle imr_{i_1}, \overline{T}_{i_1}, \sigma, s; a \rangle \xrightarrow{*} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$
along which there are n $(s; a::\sigma,)^{s; a::\sigma}$ pairs

From (i), we have $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $s; a::s\sigma = s; a :: \sigma$. From $(\sigma,)^\sigma \notin \Pi(\pi)$ and π_1 is a sub-path of π , we have $(\sigma,)^\sigma \notin \Pi(\pi_1)$. From (ii) and $(\sigma,)^\sigma \notin \Pi(\pi_1)$, we have $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} \langle imr_{i_1}, \overline{T}_{i_1}, \sigma, s; a \rangle$. From $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, s; a :: \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma} \langle imr_{i_1}, \overline{T}_{i_1}, \sigma, s; a \rangle$ and $\mu\psi_{\mathcal{R}}$ is complete (Lemma 3.6.11), we have $i_1 \in \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \overline{h}(u)).I$. From $\alpha(\zeta_u(i)) = \zeta_u(i)$, $s; a::s\sigma = s; a :: \sigma$ and Item (1) of Proposition (3.6.1), we have

- (iv) $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j', \sigma, s; a).I \mid j' \in J \} \cup \{ \alpha(\xi_s(i)) \}$
- (v) $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).k = \max(\{ \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \overline{h}(u)).k \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \} \cup \{ \mu\psi_{\mathcal{R}}(j', \sigma, s; a).k \mid j' \in J \} \cup \{ |\sigma| \})$

where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \}$. From (iv) and (v), we have

$$(vi) \quad \mu\psi_{\mathcal{R}}(j', \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$$

$$(vii) \quad \mu\psi_{\mathcal{R}}(j', \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$$

where $j' \in \bigcup \{ \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{\overline{T}}_i) \}$.

From $i_1 \in \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).I$ and (vi) and (vii), we have

$$(viii) \quad \mu\psi_{\mathcal{R}}(i', \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$$

$$(ix) \quad \mu\psi_{\mathcal{R}}(i', \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$$

From π_2 is sub-path of π and $(\sigma,)^\sigma \notin \Pi(\pi)$, we have $(\sigma,)^\sigma \notin \Pi(\pi_2)$, from which it follows $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, s; a \rangle$. From $\langle imr_{i_1}, \overline{\overline{T}}_{i_1}, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, s; a \rangle$, there are n of $(s; a :: \sigma,)^{s; a :: \sigma}$ pairs along π_2 and the induction hypothesis on n , we have

$$(x) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i', \sigma, s; a).I$$

$$(xi) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i', \sigma, s; a).k$$

From (viii) and (x), we have $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$. From (ix) and (xi), we have $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$.

2. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, iret \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, iret \rangle$, we have path $\pi : \langle imr_i, \overline{\overline{T}}_i, \sigma, iret \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, iret \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. From s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, $\langle imr_i, \overline{\overline{T}}_i, \sigma, iret \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, iret \rangle$ and Item (2) of Proposition (3.6.7), we have there are $n \geq 0$ of $(iret :: \sigma,)^{iret :: \sigma}$ pairs along path π . We prove by induction on n .

The rest of proof is similar to that of Item (1). We omit details. ■

Lemma 3.6.13 *We have*

1. if $\eta(s; a) \in \mathcal{L}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, s; a \rangle$, then $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$.
2. if $\eta(h) \in \mathcal{L}$ and $\langle imr_i, \overline{\overline{T}}_i, \sigma, h \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma', h \rangle$, then $\mu\psi_{\mathcal{R}}(j, \sigma, h).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, h).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, h).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$.

Proof If $i = j$, then the two Items of this Lemma are trivially true. We now suppose $i \neq j$ in the following proof.

1. From $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$, we have path $\pi : \langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{*} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. From $\eta(s; a) \in \mathcal{L}$, it follows $\eta(s; a)$ is a primitive statement. We prove by induction on the form of s . There are two cases.

- (a) s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**.

From s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**, $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; a \rangle$ and Item (1) of Lemma (3.6.12), we have $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$ and $\mu\psi_{\mathcal{R}}(j, \sigma, s; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$.

- (b) $s = s_1; s_2$ and $\eta(s_1; (s_2; a))$ is a primitive statement.

From $s; a = s_1; (s_2; a)$, we have $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_1; (s_2; a) \rangle$. From $\eta(s_1; (s_2; a))$ is a primitive statement, $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_1; (s_2; a) \rangle$ and the induction hypothesis, we have

$$(i) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a)).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).I$$

$$(ii) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).k$$

From Item (3) of Proposition (3.6.1), we have

$$(iii) \quad \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j', \sigma, s_2; a).I \mid j' \in J \}$$

$$(iv) \quad \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).k = \max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).k \} \cup \{ \mu\psi_{\mathcal{R}}(j'', \sigma, s_2; a).k \mid j'' \in J \})$$

where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).I$. From Item (3) of Proposition (3.6.1), we have

$$(v) \quad \mu\psi_{\mathcal{R}}(j, \sigma, (s_1; s_2); a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j', \sigma, s_2; a).I \mid j' \in J' \}$$

$$(vi) \quad \mu\psi_{\mathcal{R}}(j, \sigma, (s_1; s_2); a).k = \max(\{ \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a)).k \} \cup \{ \mu\psi_{\mathcal{R}}(j'', \sigma, s_2; a).k \mid j'' \in J' \})$$

where $J' = \mu\psi_{\mathcal{R}}(j, \sigma, s_1; (s_2; a)).I$. From (i), we have $J' \subseteq J$. From $J' \subseteq J$, (iii) and (v), we have $\mu\psi_{\mathcal{R}}(j, \sigma, (s_1; s_2); a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).I$.

From $J' \subseteq J$, (ii), (iv) and (vi), we have

$$\mu\psi_{\mathcal{R}}(j, \sigma, (s_1; s_2); a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).k.$$

2. From $\langle imr_i, \overline{T}_i, \sigma, h \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, h \rangle$, we have path $\pi : \langle imr_i, \overline{T}_i, \sigma, h \rangle \xrightarrow{*} \langle imr_j, \overline{T}_j, \sigma, h \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. From $\eta(h) \in \mathcal{L}$, it follows the form of h is either $iret$ or $s; h'$ where $\eta(s; h') \in \mathcal{L}$. We prove by case analysis on the form of h .

(a) $h = iret$.

From s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, $\langle imr_i, \overline{T}_i, \sigma, iret \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, iret \rangle$ and Item (2) of Lemma (3.6.12), we have

$$\mu\psi_{\mathcal{R}}(j, \sigma, iret).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, iret).I \text{ and } \mu\psi_{\mathcal{R}}(j, \sigma, iret).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, iret).k.$$

(b) $h = s; h'$ and $\eta(s; h') \in \mathcal{L}$.

From $\eta(s; h') \in \mathcal{L}$ and $\langle imr_i, \overline{T}_i, \sigma, s; h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s; h' \rangle$ and Item (1) of Lemma (3.6.13), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s; h').I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$$

$$(ii) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s; h').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k$$

From Item (5) of Proposition (3.6.1), we have

$$(iii) \quad \mu\psi_{\mathcal{R}}(i, \sigma, h).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j', \sigma, h').I \mid j' \in J \}$$

$$(iv) \quad \mu\psi_{\mathcal{R}}(i, \sigma, h).k =$$

$$\max(\{ \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k \} \cup \{ \mu\psi_{\mathcal{R}}(j', \sigma, h').k \mid j' \in J \})$$

where $J = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$. From Item (5) of Proposition (3.6.1), we have

$$(v) \quad \mu\psi_{\mathcal{R}}(j, \sigma, h).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j', \sigma, h').I \mid j' \in J' \}$$

$$(vi) \quad \mu\psi_{\mathcal{R}}(j, \sigma, h).k =$$

$$\max(\{ \mu\psi_{\mathcal{R}}(j, \sigma, s; h').k \} \cup \{ \mu\psi_{\mathcal{R}}(j', \sigma, h').k \mid j' \in J' \})$$

where $J' = \mu\psi_{\mathcal{R}}(j, \sigma, s; h').I$. From (i), we have $J' \subseteq J$. From $J' \subseteq J$, (iii)

and (v), we have $\mu\psi_{\mathcal{R}}(j, \sigma, a).I \subseteq \mu\psi_{\mathcal{R}}(i, \sigma, a).I$. From (ii), $J' \subseteq J$, (iv)

and (vi), we have $\mu\psi_{\mathcal{R}}(j, \sigma, h).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$.

■

3.6.3 Constructing types and type judgments

In this subsection, we define the sets that are critical to our type and type judgment constructions with the help of function $\mu\psi_{\mathcal{R}}$; and prove a number of useful lemmas regarding the sets; formally define the types and type judgments for statements, handlers and the main program.

We define the following sets

Definition 3.6.8 For $u \in 0..N$, define

1. $\mathbb{I}^{\sigma,a} = \{i \mid \langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}\}$
2. $\mathbb{I}_{i',u}^{\sigma,a} = \begin{cases} \{i \mid \langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, a \rangle\} & \text{if } i' \in \mathbb{I}^{\sigma, \overline{h}(u)} \\ \emptyset & \text{otherwise} \end{cases}$
3. $\mathbb{J}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$
4. $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$
5. $\mathbb{K}_{i',u}^{\sigma,a} = \begin{cases} \max(\{\mathbb{K}_i^{\sigma,a} \mid i \in \mathbb{I}_{i',u}^{\sigma,a}\}) & \text{if } \mathbb{I}_{i',u}^{\sigma,a} \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$

Proposition 3.6.8 We have

1. if $j \in \mathbb{I}_{i,u}^{\sigma,a}$, then $\langle imr_j, \overline{T}_j, \sigma, a \rangle \in \mathcal{R}$.
2. if $\mathbb{I}_{i,u}^{\sigma,a} \neq \emptyset$, then $a \in \mathcal{C}_u$.

Proposition 3.6.9 For $u \in 0..N$, if $i \in \mathbb{I}^{\sigma, \overline{h}(u)}$, then $\mathbb{I}_{i,u}^{\sigma, \overline{h}(u)} = \{i\}$.

Lemma 3.6.14 For $u \in 0..N$, if $i \in \mathbb{I}_{i',u}^{\sigma,a}$, then $\mathbb{K}_i^{\sigma,a} \leq \mathbb{K}_{i',u}^{\sigma,a}$.

Proof From $\mathbb{I}_{i',u}^{\sigma,a} \neq \emptyset$ and Item (4) of Definition (3.6.8), we have

$\mathbb{K}_{i',u}^{\sigma,a} = \max(\{\mathbb{K}_i^{\sigma,a} \mid i \in \mathbb{I}_{i',u}^{\sigma,a}\})$, from which it follows for $i \in \mathbb{I}_{i',u}^{\sigma,a}$, $\mathbb{K}_i^{\sigma,a} \leq \mathbb{K}_{i',u}^{\sigma,a}$. ■

Lemma 3.6.15 *If $i \in \mathbb{I}^{\sigma,a} \cap \mathbb{I}^{\sigma',a}$, then we have $\mathbb{J}_i^{\sigma,a} = \mathbb{J}_i^{\sigma',a}$ and $\mathbb{K}_i^{\sigma,a} = \mathbb{K}_i^{\sigma',a}$*

Proof From $i \in \mathbb{I}^{\sigma,a} \cap \mathbb{I}^{\sigma',a}$ we have $i \in \mathbb{I}^{\sigma,a}$ and $i \in \mathbb{I}^{\sigma',a}$. From $i \in \mathbb{I}^{\sigma,a}$ and Item (1) of Definition (3.6.8), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$. Similarly, from $i \in \mathbb{I}^{\sigma',a}$ and Item (1) of Definition (3.6.8), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma', a \rangle \in \mathcal{R}$. From Item (3) of Definition (3.6.8), we have $\mathbb{J}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$ and $\mathbb{J}_i^{\sigma',a} = \mu\psi_{\mathcal{R}}(i, \sigma', a).I$. From Item (4) of Definition (3.6.8), we have $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$ and $\mathbb{K}_i^{\sigma',a} = \mu\psi_{\mathcal{R}}(i, \sigma', a).k - |\sigma'|$. From Lemma (3.6.9), we have $\mu\psi_{\mathcal{R}}$ is stack-irrelevant. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, a \rangle \in \mathcal{R}$, $\langle imr_i, \overline{\overline{T}}_i, \sigma', a \rangle \in \mathcal{R}$, that $\mu\psi_{\mathcal{R}}$ is stack-irrelevant, and Definition (3.6.6), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \mu\psi_{\mathcal{R}}(i, \sigma', a).I$ and $\mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma| = \mu\psi_{\mathcal{R}}(i, \sigma', a).k - |\sigma'|$. From $\mu\psi_{\mathcal{R}}(i, \sigma, a).I = \mu\psi_{\mathcal{R}}(i, \sigma', a).I$, $\mathbb{J}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$ and $\mathbb{J}_i^{\sigma',a} = \mu\psi_{\mathcal{R}}(i, \sigma', a).I$, we have $\mathbb{J}_i^{\sigma,a} = \mathbb{J}_i^{\sigma',a}$. From $\mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma| = \mu\psi_{\mathcal{R}}(i, \sigma', a).k - |\sigma'|$, $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$ and $\mathbb{K}_i^{\sigma',a} = \mu\psi_{\mathcal{R}}(i, \sigma', a).k - |\sigma'|$, we have $\mathbb{K}_i^{\sigma,a} = \mathbb{K}_i^{\sigma',a}$. ■

We define the following auxiliary sets.

Definition 3.6.9 *For $u \in 1..N$, define*

1. $\mathbb{I}^u = \{i \mid i \in \mathbb{I}^{\sigma,a} \wedge \zeta_u(i) \in \mathbb{I}^{a::\sigma, \overline{h}(u)}\}$
2. $\mathbb{W}_i^u = \{a :: \sigma \mid i \in \mathbb{I}^{\sigma,a} \wedge \zeta_u(i) \in \mathbb{I}^{a::\sigma, \overline{h}(u)}\}$
3. $\mathbb{J}_i^u = \begin{cases} \mathbb{J}_{\zeta_u(i)}^{\sigma, \overline{h}(u)} & \text{for any } \sigma \in \mathbb{W}_i^u \text{ if } i \in \mathbb{I}^u \\ \emptyset & \text{otherwise} \end{cases}$
4. $\mathbb{K}_i^u = \begin{cases} \mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)} & \text{for any } \sigma \in \mathbb{W}_i^u \text{ if } i \in \mathbb{I}^u \\ 0 & \text{otherwise} \end{cases}$

Note that if $i \in \mathbb{I}^u$, then \mathbb{J}_i^u and \mathbb{K}_i^u are well defined sets because of Lemma (3.6.15), Clearly, for all $u \in 1..N$, we have $\mathbb{I}^u \subseteq \mathcal{I}$ and for all $i \in \mathbb{I}^u$, $\mathbb{W}_i^u \subseteq \mathcal{S}$.

Proposition 3.6.10 *If $i \in \mathbb{I}^u$, then there exists $\sigma \in \mathbb{W}_i^u$ such that $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\sigma, \overline{h}(u)}$ and $\mathbb{K}_i^u = \mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)}$.*

We now construct handler types from \mathcal{R} :

Definition 3.6.10 Define $\bar{\tau}_{\mathcal{R}}: \forall u \in 1..N: \bar{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in \mathbb{I}^u} (imr_i, \bar{T}_i \xrightarrow{\mathbb{K}_i^u} \bigvee_{j \in \mathbb{J}_i^u} imr_j, \bar{T}_j)$

We construct type judgments from \mathcal{R} :

Definition 3.6.11 For $u \in 0..N$, $\sigma \in \mathcal{S}$ and $i' \in \mathcal{I}$, define

1. Type judgment for statement s

$$\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',u}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',u}^{\sigma,s;a}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;a}} imr_j, \bar{T}_j) \text{ where } s; a \in \mathcal{C}$$

2. Type judgment for main

$$\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i_0,0}^{nil,m}} m : \bigvee_{i \in \mathbb{I}_{i_0,0}^{nil,m}} imr_i, \bar{T}_i$$

3. Type judgment for handlers

$$\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',u}^{\sigma,h}} h : \bigwedge_{i \in \mathbb{I}_{i',u}^{\sigma,h}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h}} imr_j, \bar{T}_j) \text{ where } h \in \mathcal{C}$$

3.6.4 Well-formedness of types and type judgments

In this subsection, we first prove some lemmas that will be used in the later part of the section. We then proceed to prove the well-formedness of the types and type judgments constructed as we did in subsection 3.6.3.

Lemma 3.6.16 If $Q \in \mathcal{R} \Rightarrow Q.stk \leq K$, then $\mathbb{K}_i^{\sigma,a} \leq K$

Proof From $Q \in \mathcal{R} \Rightarrow Q.stk \leq K$ and Proposition (3.6.2), we have $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : \mu\psi_{\mathcal{R}}(i, \sigma, a).k \leq K$. From Item (4) Definition (3.6.8), we have $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$. From $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$ and $\forall i \in \mathcal{I} : \forall \sigma \in \mathcal{S} : \forall a \in \mathcal{C} : \mu\psi_{\mathcal{R}}(i, \sigma, a).k \leq K$, we have $\mathbb{K}_i^{\sigma,a} \leq K - |\sigma|$. From $\mathbb{K}_i^{\sigma,a} \leq K - |\sigma|$ and $|\sigma| \geq 0$, we have $\mathbb{K}_i^{\sigma,a} \leq K$. ■

Lemma 3.6.17 $\mathbb{K}_i^{\sigma,a} \geq 0$

Proof From Item (4) Definition (3.6.8), we have $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$. From Item (2) of Lemma (3.6.2), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$. From $\mathbb{K}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).k - |\sigma|$ and $\mu\psi_{\mathcal{R}}(i, \sigma, a).k \geq |\sigma|$, we have $\mathbb{K}_i^{\sigma,a} \geq 0$. ■

Corollary 3.6.2 $\mathbb{K}_{i',u}^{\sigma,a} \geq 0$

Proof There are two cases depending on whether $\mathbb{I}_{i',u}^{\sigma,a} = \emptyset$. If $\mathbb{I}_{i',u}^{\sigma,a} = \emptyset$, then from Item (5) of Definition (3.6.8), we have $\mathbb{K}_{i',u}^{\sigma,a} = 0$. Suppose $i \in \mathbb{I}_{i',u}^{\sigma,a}$. From Lemma (3.6.17), we have $\mathbb{K}_i^{\sigma,a} \geq 0$. From $i \in \mathbb{I}^{\sigma,a}$ and Item (5) of Definition (3.6.8), we have $\mathbb{K}^{\sigma,a} = \max(\{\mathbb{K}_i^{\sigma,a} \mid i \in \mathbb{I}^{\sigma,a}\})$. From $\mathbb{K}_i^{\sigma,a} \geq 0$ and $\mathbb{K}^{\sigma,a} = \max(\{\mathbb{K}_i^{\sigma,a} \mid i \in \mathbb{I}^{\sigma,a}\})$, we have $\mathbb{K}^{\sigma,a} \geq 0$. ■

Lemma 3.6.18 $\mathbb{J}_i^{\sigma,a} \neq \emptyset$.

Proof From Item (3) of Definition (3.6.8), we have $\mathbb{J}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$. From Item (1) of Lemma (3.6.2), we have $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$. From $\mu\psi_{\mathcal{R}}(i, \sigma, a).I \neq \emptyset$ and $\mathbb{J}_i^{\sigma,a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$, we have $\mathbb{J}_i^{\sigma,a} \neq \emptyset$. ■

Lemma 3.6.19 For $u \in 0..N$, if $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$, then we have $\mathbb{J}_i^{\sigma,s;a} \subseteq \mathbb{I}_{i',u}^{\sigma,a}$.

Proof From Lemma (3.6.18), we have $\mathbb{J}_i^{\sigma,s;a} \neq \emptyset$. Consider $j \in \mathbb{J}_i^{\sigma,s;a}$. From $j \in \mathbb{J}_i^{\sigma,s;a}$ and Item (3) of Definition (3.6.8), we have $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$. From Lemma (3.6.9), we have $\mu\psi_{\mathcal{R}}$ is sound. From $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$ and Item (1) of Proposition (3.6.8), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \in \mathcal{R}$. From $\mu\psi_{\mathcal{R}}$ is sound, $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \in \mathcal{R}$, $j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$ and Item (1) of Definition (3.6.5), we have $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$. From $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$ and Item (2) of Definition (3.6.8), we have we have $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$ and $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle$. From $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle$, $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$ and Item (1) of Lemma (3.6.4), we have $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$. From $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^*} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$ and Item (1) of Proposition (3.6.5), we have $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$. From $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$, $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, a \rangle$ and Item (2) of Definition (3.6.8), we have $j \in \mathbb{I}_{i',u}^{\sigma,a}$. ■

Corollary 3.6.3 For $u \in 0..N$, if $\mathbb{I}_{i',u}^{\sigma,s;a} \neq \emptyset$, then we have $\mathbb{I}_{i',u}^{\sigma,a} \neq \emptyset$.

Lemma 3.6.20 $\mathbb{I}_{i_0,0}^{\text{nil},m} \neq \emptyset$.

Proof We prove by induction on the nesting level of m . Define the nesting level of m inductively as follows:

- the nesting level of $p.m$ 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 $n + 1$.

In base case ($n = 0$), we have the initial state Q_0 , where $Q_0.a = p.m$, it follows $\mathbb{I}_{i_0,0}^{\text{nil},p.m} = \{i_0\}$. In induction step, suppose $m = s;m'$ and the nesting level of m is n . From the nesting level of m is n and the induction hypothesis, we have $\mathbb{I}_{i_0,0}^{\text{nil},m} \neq \emptyset$. From $\mathbb{I}_{i_0,0}^{\text{nil},m} \neq \emptyset$ and Corollary (3.6.3), we have $\mathbb{I}_{i_0,0}^{\text{nil},m'} \neq \emptyset$. ■

Lemma 3.6.21 For $u \in 1..N$, if $\mathbb{I}^{a::\sigma,\bar{h}(u)} \neq \emptyset$, then $\exists i : i \in \mathbb{I}^{\sigma,a} \wedge \zeta_u(i) \in \mathbb{I}^{a::\sigma,\bar{h}(u)}$.

Proof Let $j \in \mathbb{I}^{a::\sigma,\bar{h}(u)}$. From $j \in \mathbb{I}^{a::\sigma,\bar{h}(u)}$ and Item (1) of Definition (3.6.8), we have $\langle \text{imr}_j, \bar{T}_j, a :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$. Consider a state immediately before $\langle \text{imr}_j, \bar{T}_j, a :: \sigma, \bar{h}(u) \rangle$ in \mathcal{R} . By inspecting Rules (3.10)-(3.18), we have that Rule (3.10) is the only one that can have $\langle \text{imr}_j, \bar{T}_j, a :: \sigma, \bar{h}(u) \rangle$ on the right hand side of the binary relation \leftrightarrow , from which it follows that we can choose i such that $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \leftrightarrow \langle \text{imr}_j, \bar{T}_j, a :: \sigma, \bar{h}(u) \rangle$ and $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \in \mathcal{R}$, where $\zeta_u(i) = j$. From $\langle \text{imr}_i, \bar{T}_i, \sigma, a \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $i \in \mathbb{I}^{\sigma,a}$. ■

Lemma 3.6.22 Suppose $u \in 1..N$. If $Q \in \mathcal{R}$ and $Q.a = s;a$ and $\forall Q' \in \mathcal{R} : u \notin \hat{\mathcal{P}}(Q'.\text{imr}, Q'.\bar{T})$, then $\exists Q'' \in \mathcal{R}$ such that $Q''.a = a \wedge Q''.\bar{T}(u) > Q.\bar{T}(u)$.

Proof Let $Q = \langle \text{imr}, \bar{T}, \sigma, s;a \rangle$. We prove by induction on the form of s . There are three cases.

1. s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip .

From $\langle \text{imr}, \bar{T}, \sigma, s;a \rangle \in \mathcal{R}$ and Rules (3.13)-(3.15), (3.18), we have $\langle \chi_s(\text{imr}), \bar{T}', \sigma, a \rangle \in \mathcal{R}$, where $\bar{T}' = \hat{\theta}(\text{imr}, \bar{T})$. From $\forall Q' \in \mathcal{R} :$

$u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$, we have $\overline{T}'(u) = \widehat{\theta}(imr, \overline{T})(u) = \overline{T}(u) + 1$, from which it follows $\overline{T}'(u) > \overline{T}(u)$.

2. s is if0 x then s_1 else s_2 .

From $\langle imr, \overline{T}, \sigma, s; a \rangle \in \mathcal{R}$ and Rule (3.16), we have $\langle imr, \overline{T}, \sigma, s_1; a \rangle \in \mathcal{R}$. From $\langle imr, \overline{T}, \sigma, s_1; a \rangle \in \mathcal{R}$ and $\forall Q' \in \mathcal{R} : u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$ and the induction hypothesis, we have $\exists Q'' \in \mathcal{R}$ such that $Q''.a = a \wedge Q''.\overline{T}(u) > \overline{T}(u)$.

3. s is $s_1; s_2$.

From $(s_1; s_2); a$ and $\langle imr, \overline{T}, \sigma, s; a \rangle \in \mathcal{R}$, we have $\langle imr, \overline{T}, \sigma, s_1; (s_2; a) \rangle \in \mathcal{R}$. From $\langle imr, \overline{T}, \sigma, s_1; (s_2; a) \rangle \in \mathcal{R}$ and $\forall Q' \in \mathcal{R} : u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$ and the induction hypothesis, we have $\exists Q'' \in \mathcal{R}$ such that $Q''.a = s_2; a \wedge Q''.\overline{T}(u) > \overline{T}(u)$. From $Q'' \in \mathcal{R}$ and $Q''.a = s_2; a$ and $\forall Q' \in \mathcal{R} : u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$ and the induction hypothesis, we have $\exists Q''' \in \mathcal{R}$ such that $Q'''.a = a \wedge Q'''.\overline{T}(u) > Q''.\overline{T}(u)$. From $Q'''.\overline{T}(u) > Q''.\overline{T}(u)$ and $Q''.\overline{T}(u) > \overline{T}(u)$, we have $Q'''.\overline{T}(u) > \overline{T}(u)$.

■

Lemma 3.6.23 For $u \in 1..N$, if \mathcal{R} is finite, then $\exists \sigma \in \mathcal{S}$ such that $\mathbb{I}^{\sigma, \overline{h}(u)} \neq \emptyset$.

Proof Let $u \in 1..N$. Suppose for the sake of contradiction that $\forall \sigma \in \mathcal{S}$, we have $\mathbb{I}^{\sigma, \overline{h}(u)} = \emptyset$. From Item (1) of Definition (3.6.8), we have $\nexists Q \in \mathcal{R}$ such that $Q.a = \overline{h}(u)$. From $\nexists Q \in \mathcal{R}$ such that $Q.a = \overline{h}(u)$, and by inspecting conditions of Rules (3.10)-(3.18), we have $\forall Q' \in \mathcal{R} : u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$. Let N be the set of natural numbers. We will construct an infinite state sequence Q^0, \dots, Q^n, \dots such that $\forall n \in N : Q^n \in \mathcal{R} \wedge Q^n.a = \text{loop } s$ and $i > j \Rightarrow Q^i.\overline{T}(u) > Q^j.\overline{T}(u)$, which contradicts the assumption that \mathcal{R} is finite. We construct the sequence inductively as follows:

Base case (Q^0). From Lemma (3.6.20), we have $\mathbb{I}_{i_0, 0}^{\text{nil}, \text{loop } s} \neq \emptyset$. Let $i \in \mathbb{I}_{i_0, 0}^{\text{nil}, \text{loop } s}$. From $i \in \mathbb{I}_{i_0, 0}^{\text{nil}, \text{loop } s}$ and Item (1) of Proposition (3.6.8), we have $\langle imr_i, \overline{T}_i, \text{nil}, \text{loop } s \rangle \in \mathcal{R}$. Let $Q^0 = \langle imr_i, \overline{T}_i, \text{nil}, \text{loop } s \rangle$.

Induction step (Q^{n+1}). Suppose $Q^n \in \mathcal{R} \wedge Q^n.a = \text{loop } s$. From $Q^n \in \mathcal{R}$ and

Rule (3.12), we have $\langle Q^n.imr, Q^n.\overline{T}, \text{nil}, s; \text{loop } s \rangle \in \mathcal{R}$. From $\langle Q^n.imr, Q^n.\overline{T}, \text{nil}, s; \text{loop } s \rangle \in \mathcal{R}$ and $\forall Q' \in \mathcal{R} : u \notin \widehat{\mathcal{P}}(Q'.imr, Q'.\overline{T})$ and Lemma (3.6.22), $\exists Q'' \in \mathcal{R}$ such that $Q''.a = \text{loop } s \wedge Q''.\overline{T}(u) > Q^n.\overline{T}$. Let $Q^{n+1} = Q''$. ■

Corollary 3.6.4 *Suppose \mathcal{R} is finite. For $u \in 1..N$, $\mathbb{I}^u \neq \emptyset$.*

Proof From \mathcal{R} is finite and Lemma (3.6.23), we have there exist $\sigma \in \mathcal{S}$ such that $\mathbb{I}^{\sigma, \overline{h}(u)} \neq \emptyset$. Let $i \in \mathbb{I}^{\sigma, \overline{h}(u)}$. From $i \in \mathbb{I}^{\sigma, \overline{h}(u)}$ and Item (1) of Definition (3.6.8), we have $\langle imr_i, \overline{T}_i, \sigma, \overline{h}(u) \rangle \in \mathcal{R}$. From $\langle imr_i, \overline{T}_i, \sigma, \overline{h}(u) \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have state $\langle imr_i, \overline{T}_i, \sigma, \overline{h}(u) \rangle$ is consistent, from which it follows $\sigma = a :: \sigma'$. From $\sigma = a :: \sigma'$ and $\mathbb{I}^{\sigma, \overline{h}(u)} \neq \emptyset$, we have $\mathbb{I}^{a::\sigma', \overline{h}(u)} \neq \emptyset$. From $\mathbb{I}^{a::\sigma', \overline{h}(u)} \neq \emptyset$ and Lemma (3.6.21), we have $\exists i : i \in \mathbb{I}^{\sigma', a} \wedge \zeta_u(i) \in \mathbb{I}^{a::\sigma', \overline{h}(u)}$. From $\exists i : i \in \mathbb{I}^{\sigma', a} \wedge \zeta_u(i) \in \mathbb{I}^{a::\sigma', \overline{h}(u)}$ and Item (1) of Definition (3.6.9), we have $\exists i : i \in \mathbb{I}^u$, ■

We now prove that the handler types are well formed in the sense of Definition (3.4.1).

Lemma 3.6.24 (Well-formedness of handler types) *Suppose $Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K)$. For $u \in 1..N$, $\overline{\tau}_{\mathcal{R}}(u) = \bigwedge_{i \in \mathbb{I}^u} (imr_i, \overline{T}_i \xrightarrow{\mathbb{K}_i^u} \bigvee_{j \in \mathbb{J}_i^u} imr_j, \overline{T}_j)$ is well formed.*

Proof From $Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K)$ and Theorem 3.3.1, we have \mathcal{R} is finite. From Definition (3.4.1), we need to prove, for $u \in 1..N$

- (i) $\mathbb{I}^u \neq \emptyset$
- (ii) $\forall i \in \mathbb{I}^u : \mathbb{K}_i^u \geq 0$
- (iii) $\forall i \in \mathbb{I}^u : \mathbb{J}_i^u \neq \emptyset$

The proof of Item (i) comes immediately from Corollary (3.6.4). Let $i \in \mathbb{I}^u$, then from Proposition (3.6.10), there exists $\sigma \in \mathbb{W}_i^u$ such that we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\sigma, \overline{h}(u)}$ and $\mathbb{K}_i^u = \mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)}$. From Lemma (3.6.17), we have $\mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)} \geq 0$. From $\mathbb{K}_i^u = \mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)}$ and $\mathbb{K}_{\zeta_u(i)}^{\sigma, \overline{h}(u)} \geq 0$, we have $\mathbb{K}_i^u \geq 0$, which is Item (ii). From $i \in \mathbb{I}^u$ and Item (3) of Definition

(3.6.9), we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\sigma, \bar{h}(u)}$. From $i \in \mathbb{I}^u$ and Lemma (3.6.18), we have $\mathbb{J}_{\zeta_u(i)}^{\sigma, \bar{h}(u)} \neq \emptyset$. From $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\sigma, \bar{h}(u)}$ and $\mathbb{J}_{\zeta_u(i)}^{\sigma, \bar{h}(u)} \neq \emptyset$, we have $\mathbb{J}_i^u \neq \emptyset$, which is Item (iii). ■

We now prove the well-formedness of type judgments for statements, handlers and the main program in the sense of Definition (3.4.2).

Lemma 3.6.25 (Well-formedness of type judgment, statements) *Suppose $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$. If $\mathbb{I}_{i',u}^{\sigma,s;a} \neq \emptyset$, then $\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',u}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',u}^{\sigma,s;a}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;a}} imr_j, \bar{T}_j)$ is well formed.*

Proof From $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$ and Theorem 3.3.1, we have \mathcal{R} is finite. From Item (1) of Definition (3.4.2), it suffices to prove:

- (i) $\forall i \in \mathbb{I}_{i',u}^{\sigma,s;a} : \mathbb{J}_i^{\sigma,s;a} \neq \emptyset$
- (ii) $\mathbb{K}_{i',u}^{\sigma,s;a} \geq 0$.

Proof of Item (i) comes immediately from Lemma (3.6.18). Proof of Item (ii) comes immediately from Corollary (3.6.2). ■

Lemma 3.6.26 (Well-formedness of type judgment, handler) *Suppose $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$. If $\mathbb{I}_{i',u}^{\sigma,h} \neq \emptyset$, then $\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',u}^{\sigma,h}} h : \bigwedge_{i \in \mathbb{I}_{i',u}^{\sigma,h}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h}} imr_j, \bar{T}_j)$ is well formed.*

Proof From $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$ and Theorem 3.3.1, we have \mathcal{R} is finite. From Item (3) of Definition (3.4.2), it suffices to prove:

- (i) $\forall i \in \mathbb{I}_{i',u}^{\sigma,h} : \mathbb{J}_i^{\sigma,h} \neq \emptyset$
- (ii) $\mathbb{K}_{i',u}^{\sigma,h} \geq 0$.

Proof of Item (i) comes immediately from Lemma (3.6.18). Proof of Item (ii) comes immediately from Corollary (3.6.2). ■

Lemma 3.6.27 (Well-formedness of type judgment, main) *Suppose $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$. $\bar{\tau}_{\mathcal{R}} \vdash_K m : \bigvee_{i \in \mathbb{I}_{i_0,0}^{nil,m}} imr_i, \bar{T}_i$ is well formed.*

Proof It is clear that $K \geq 0$. From Item (2) of Definition (3.4.2), we need to prove: $\mathbb{I}_{i_0,0}^{nil,m} \neq \emptyset$, which comes immediately from Lemma (3.6.20). ■

3.6.5 Constructing type derivations

In this subsection, we first prove lemmas that will be used in the proofs of type derivation constructions. We then proceed to show how to build the type derivations for statements, the main program and handlers.

Proposition 3.6.11 *We have*

1. $\mathbb{I}_{i',u}^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a} = \mathbb{I}_{i',u}^{\sigma,s_1;a} = \mathbb{I}_{i',u}^{\sigma,s_2;a}$
2. $\mathbb{I}_{i',u}^{\sigma,(s_1;s_2);a} = \mathbb{I}_{i',u}^{\sigma,s_1;(s_2;a)}$
3. $\mathbb{I}_{i_0,0}^{\sigma,s;\text{loop } s} = \mathbb{I}_{i_0,0}^{\sigma,\text{loop } s}$

Lemma 3.6.28 *For $u \in 0..N$, if $\eta(a) \in \mathcal{L}$, $i \in \mathbb{I}_{i',u}^{\sigma,a}$ and $u \in \widehat{\mathcal{P}}(\text{imr}_i, \widehat{T}_i)$, then we have $\mathbb{J}_i^u \neq \emptyset$ and $\mathbb{J}_i^u \subseteq \mathbb{I}_{i',u}^{\sigma,a}$.*

Proof From $i \in \mathbb{I}_{i',u}^{\sigma,a}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \in \mathcal{R}$, from which it follows $i \in \mathbb{I}^{\sigma,a}$. From $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(\text{imr}_i, \widehat{T}_i)$ and Item (4) of Lemma (3.6.5), we have $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle$. From $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \in \mathcal{R}$ and $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle$, we have $\langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$, from which it follows $\zeta_u(i) \in \mathbb{I}^{a::\sigma, \bar{h}(u)}$. From $i \in \mathbb{I}^{\sigma,a}$, $\zeta_u(i) \in \mathbb{I}^{a::\sigma, \bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$. From $i \in \mathbb{I}^{\sigma,a}$, $\zeta_u(i) \in \mathbb{I}^{a::\sigma, \bar{h}(u)}$ and Item (2) of Definition (3.6.9), we have $a :: \sigma \in \mathbb{W}_i^u$. From $i \in \mathbb{I}^u$, $a :: \sigma \in \mathbb{W}_i^u$ and Item (3) of Definition (3.6.9), we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$. From Lemma (3.6.18), we have $\mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)} \neq \emptyset$. From $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$ and $\mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)} \neq \emptyset$, we have $\mathbb{J}_i^u \neq \emptyset$.

Consider $j \in \mathbb{J}_i^u$. From $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$, we have $j \in \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$. From $j \in \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$ and Item (3) of Definition (3.6.8), we have $j \in \mu\psi_{\mathcal{R}}(\zeta_u(i), a :: \sigma, \bar{h}(u)).I$. From Lemma (3.6.9), we have $\mu\psi_{\mathcal{R}}$ is sound. From $\mu\psi_{\mathcal{R}}$ is sound, $\langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle \in \mathcal{R}$, $j \in \mu\psi_{\mathcal{R}}(\zeta_u(i), a :: \sigma, \bar{h}(u)).I$ and Item (2) of Definition (3.6.5), we have $\langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle \xrightarrow{\sigma} \langle \text{imr}_j, \widehat{T}_j, \sigma, a \rangle$. From $\langle \text{imr}_i, \widehat{T}_i, \sigma, a \rangle \hookrightarrow \langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle$, $\langle \text{imr}_{\zeta_u(i)}, \widehat{T}_{\zeta_u(i)}, a :: \sigma, \bar{h}(u) \rangle \xrightarrow{\sigma} \langle \text{imr}_j, \widehat{T}_j, \sigma, a \rangle$, Item (2)

of Lemma (3.6.4), we have $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$. From $i \in \mathbb{I}_{i',u}^{\sigma,a}$ and Item (2) of Definition (3.6.8), we have $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, a \rangle$. From $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, a \rangle$, $\langle imr_i, \overline{T}_i, \sigma, a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$, Item (4) of Lemma (3.6.4), we have $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$. From $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ and Item (2) of Definition (3.6.8), we have $j \in \mathbb{I}_{i',u}^{\sigma,a}$. ■

Lemma 3.6.29 *For $u \in 0..N$, if $\mathbb{I}_{i',u}^{\sigma,s;a} \neq \emptyset$ and $j \in \mathbb{I}_{i',u}^{\sigma,a}$ then $\exists i \in \mathbb{I}_{i',u}^{\sigma,s;a}$ such that $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$.*

Proof From $j \in \mathbb{I}_{i',u}^{\sigma,a}$ and Item (2) of Definition (3.6.8), we have $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$. From $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ and Item (1) of Definition (3.6.4), we have a path $\pi: \langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ and $(\sigma,)^\sigma \notin \Pi(\pi)$. From $\mathbb{I}_{i',u}^{\sigma,s;a} \neq \emptyset$ and Item (2) of Proposition (3.6.8), we have $s; a \in \mathcal{C}_u$. From $s; a \in \mathcal{C}_u$, $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$ and Item (3) of Proposition (3.6.4), there exists i along π such that we have sub-path $\pi_1: \langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, s; a \rangle$ and sub-path $\pi_2: \langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$. From π_1 is sub-path of π and $(\sigma,)^\sigma \notin \Pi(\pi)$, we have $(\sigma,)^\sigma \notin \Pi(\pi_1)$, from which it follows $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, s; a \rangle$. From $i' \in \mathbb{I}^{\sigma, \overline{h}(u)}$, $\langle imr_{i'}, \overline{T}_{i'}, \sigma, \overline{h}(u) \rangle \xrightarrow{\sigma^+} \langle imr_i, \overline{T}_i, \sigma, s; a \rangle$ and Item (2) of Definition (3.6.8), we have $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$. From π_2 is sub-path of π and $(\sigma,)^\sigma \notin \Pi(\pi)$, we have $(\sigma,)^\sigma \notin \Pi(\pi_2)$, from which it follows $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, a \rangle$. ■

Lemma 3.6.30 *We have*

1. $\mathbb{J}_i^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a} = \mathbb{J}^{\sigma, s_1; a} \cup \mathbb{J}^{\sigma, s_2; a}$
2. $\mathbb{J}_i^{\sigma, (s_1; s_2); a} = \bigcup \{ \mathbb{J}_j^{\sigma, s_2; a} \mid j \in \mathbb{J}_i^{\sigma, s_1; (s_2; a)} \}$
3. *if $h = s; h'$, then $\mathbb{J}_i^{\sigma, h} = \bigcup \{ \mathbb{J}_j^{\sigma, h'} \mid j \in \mathbb{J}_i^{\sigma, s; h'} \}$*

Proof 1. From Item (2) of Proposition (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(i, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a).I = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a).I \cup \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a).I$$

From Item (3) of Definition (3.6.8), we have

$$(ii) \quad \mathbb{J}_i^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a} = \mu\psi_{\mathcal{R}}(i, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a).I$$

$$(iii) \quad \mathbb{J}^{\sigma, s_1; a} = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a).I$$

$$(iv) \quad \mathbb{J}^{\sigma, s_2; a} = \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a).I$$

Replacing (i) with (ii), (iii) and (iv), we have $\mathbb{J}_i^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a} = \mathbb{J}^{\sigma, s_1; a} \cup \mathbb{J}^{\sigma, s_2; a}$.

2. From Item (3) of Proposition (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(i, \sigma, a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').I \mid j \in \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I \}$$

From Item (3) of Definition (3.6.8), we have

$$(ii) \quad \mathbb{J}_i^{\sigma, (s_1; s_2); a} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$$

$$(iii) \quad \mathbb{J}_i^{\sigma, s_1; (s_2; a)} = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a')).I$$

$$(iv) \quad \mathbb{J}_j^{\sigma, s_2; a} = \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a').I$$

Replacing (i) with (ii), (iii) and (iv), we have $\mathbb{J}_i^{\sigma, (s_1; s_2); a} = \bigcup \{ \mathbb{J}_j^{\sigma, s_2; a} \mid j \in \mathbb{J}_i^{\sigma, s_1; (s_2; a)} \}$.

3. From Item (5) of Proposition (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(i, \sigma, a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, h').I \mid j \in \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I \}$$

From Item (3) of Definition (3.6.8), we have

$$(ii) \quad \mathbb{J}_i^{\sigma, h} = \mu\psi_{\mathcal{R}}(i, \sigma, a).I$$

$$(iii) \quad \mathbb{J}_i^{\sigma, s; h'} = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$$

$$(iv) \quad \mathbb{J}_j^{\sigma, h'} = \mu\psi_{\mathcal{R}}(j, \sigma, h').I$$

Replacing (i) with (ii), (iii) and (iv), we have $\mathbb{J}_i^{\sigma, h} = \bigcup \{ \mathbb{J}_j^{\sigma, h'} \mid j \in \mathbb{J}_i^{\sigma, s; h'} \}$. ■

Lemma 3.6.31 *We have*

1. For $u \in 0..N$, if $\mathbb{I}_{i', u}^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a} \neq \emptyset$, then we have

$$\mathbb{K}_{i', u}^{\sigma, s_1; a} \leq \mathbb{K}_{i', u}^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a} \quad \text{and} \quad \mathbb{K}_{i', u}^{\sigma, s_2; a} \leq \mathbb{K}_{i', u}^{\sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a}$$

2. For $u \in 0..N$, if $\mathbb{I}_{i', u}^{\sigma, (s_1; s_2); a} \neq \emptyset$, then we have $\mathbb{K}_{i', u}^{\sigma, s_1; (s_2; a)} \leq \mathbb{K}_{i', u}^{\sigma, (s_1; s_2); a}$ and

$$\mathbb{K}_{i', u}^{\sigma, s_2; a} \leq \mathbb{K}_{i', u}^{\sigma, (s_1; s_2); a}$$

3. For $u \in 1..N$, if $\mathbb{I}_{i',u}^{\sigma,h} \neq \emptyset$, where $h = s; h'$, then we have $\mathbb{K}_{i',u}^{\sigma,s;h'} \leq \mathbb{K}_{i',u}^{\sigma,h}$ and $\mathbb{K}_{i',u}^{\sigma,h'} \leq \mathbb{K}_{i',u}^{\sigma,h}$

Proof 1. From Item (1) of Proposition (3.6.11), we have $\mathbb{I}_{i',u}^{\sigma,\text{if0 } x \text{ then } s_1 \text{ else } s_2;a} = \mathbb{I}_{i',u}^{\sigma,s_1;a} = \mathbb{I}_{i',u}^{\sigma,s_2;a}$. Let $I = \mathbb{I}_{i',u}^{\sigma,\text{if0 } x \text{ then } s_1 \text{ else } s_2;a}$. From $\mathbb{I}_{i',u}^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a} \neq \emptyset$, we have $I \neq \emptyset$. From $I \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$$\mathbb{K}_{i',u}^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a} = \max(\{\mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a} \mid i \in I\})$$

$$\mathbb{K}_{i',u}^{\sigma,s_1;a} = \max(\{\mathbb{K}_i^{\sigma,s_1;a} \mid i \in I\})$$

$$\mathbb{K}_{i',u}^{\sigma,s_2;a} = \max(\{\mathbb{K}_i^{\sigma,s_2;a} \mid i \in I\})$$

It suffices to prove that for any $i \in I$, we have $\mathbb{K}_i^{\sigma,s_1;a} \leq \mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a}$ and $\mathbb{K}_i^{\sigma,s_2;a} \leq \mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a}$. From Item (4) of Definition (3.6.8), we have

$$(i) \quad \mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a} = \mu\psi_{\mathcal{R}}(i, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a').k - |\sigma|$$

$$(ii) \quad \mathbb{K}_i^{\sigma,s_1;a} = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a').k - |\sigma|$$

$$(iii) \quad \mathbb{K}_i^{\sigma,s_2;a} = \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a').k - |\sigma|$$

From Item (2) of Lemma (3.6.1), we have

$$(iv) \quad \mu\psi_{\mathcal{R}}(i, \sigma, s_1; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a).k$$

$$(v) \quad \mu\psi_{\mathcal{R}}(i, \sigma, s_2; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (\text{if0 } x \text{ then } s_1 \text{ else } s_2); a).k$$

Subtracting $|\sigma|$ on both sides of inequalities (iv) and (v), and replacing with

(i), (ii) and (iii), we have $\mathbb{K}_i^{\sigma,s_1;a} \leq \mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a}$ and

$$\mathbb{K}_i^{\sigma,s_2;a} \leq \mathbb{K}_i^{\sigma,(\text{if0 } x \text{ then } s_1 \text{ else } s_2);a}.$$

2. From Item (2) of Proposition (3.6.11), we have $\mathbb{I}_{i',u}^{\sigma,(s_1;s_2);a} = \mathbb{I}_{i',u}^{\sigma,s_1;(s_2;a)}$. Let $I = \mathbb{I}_{i',u}^{\sigma,(s_1;s_2);a}$. From $\mathbb{I}_{i',u}^{\sigma,(s_1;s_2);a} \neq \emptyset$, we have $I \neq \emptyset$.

We first prove $\mathbb{K}_{i',u}^{\sigma,s_1;(s_2;a)} \leq \mathbb{K}_{i',u}^{\sigma,(s_1;s_2);a}$. From $I \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$$\mathbb{K}_{i',u}^{\sigma,(s_1;s_2);a} = \max(\{\mathbb{K}_i^{\sigma,(s_1;s_2);a} \mid i \in I\})$$

$$\mathbb{K}_{i',u}^{\sigma,s_1;(s_2;a)} = \max(\{\mathbb{K}_i^{\sigma,s_1;(s_2;a)} \mid i \in I\})$$

It suffices to prove that for any $i \in I$, we have $\mathbb{K}_i^{\sigma,s_1;(s_2;a)} \leq \mathbb{K}_i^{\sigma,(s_1;s_2);a}$. From Item (4) of Definition (3.6.8), we have

$$(i) \quad \mathbb{K}_i^{\sigma,(s_1;s_2);a} = \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).k - |\sigma|$$

$$(ii) \quad \mathbb{K}_i^{\sigma,s_1;(s_2;a)} = \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).k - |\sigma|$$

From Item (3) of Lemma (3.6.1), we have

$$(iii) \quad \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).k$$

Subtracting $|\sigma|$ on both sides of the inequality (iii) and replacing with (i) and (ii), we have $\mathbb{K}_i^{\sigma, s_1; (s_2; a)} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$.

We next prove $\mathbb{K}_{i', u}^{\sigma, s_2; a} \leq \mathbb{K}_{i', u}^{\sigma, (s_1; s_2); a}$. From $\mathbb{I}_{i', u}^{\sigma, s_1; (s_2; a)} \neq \emptyset$ and Corollary (3.6.3), we have $\mathbb{I}_{i', u}^{\sigma, s_2; a} \neq \emptyset$. From $\mathbb{I}_{i', u}^{\sigma, s_2; a} \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$$\mathbb{K}_{i', u}^{\sigma, s_2; a} = \max(\{\mathbb{K}_j^{\sigma, s_2; a} \mid j \in \mathbb{I}_{i', u}^{\sigma, s_2; a}\})$$

It suffices to prove that for any $j \in \mathbb{I}_{i', u}^{\sigma, s_2; a}$, there exists $i \in \mathbb{I}_{i', u}^{\sigma, (s_1; s_2); a}$ such that $\mathbb{K}_j^{\sigma, s_2; a} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$. From $j \in \mathbb{I}_{i', u}^{\sigma, s_2; a}$ and Item (4) of Definition (3.6.8), we have

$$(iv) \quad \mathbb{K}_j^{\sigma, s_2; a} = \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a).k - |\sigma|$$

From $\mathbb{I}_{i', u}^{\sigma, s_1; (s_2; a)} \neq \emptyset$, $j \in \mathbb{I}_{i', u}^{\sigma, s_2; a}$ and Lemma (3.6.29), there exists $i \in \mathbb{I}_{i', u}^{\sigma, s_1; (s_2; a)}$ such that $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_2; a \rangle$.

From $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_2; a \rangle$ and Item (2) of Proposition (3.6.5), there exists i' such that $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle$ and $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_2; a \rangle$. From Item (4) of Definition (3.6.8), we have

$$(v) \quad \mathbb{K}_{i'}^{\sigma, s_2; a} = \mu\psi_{\mathcal{R}}(i', \sigma, s_2; a).k - |\sigma|$$

From $\langle imr_i, \overline{T}_i, \sigma, s_1; (s_2; a) \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle$ and $\mu\psi_{\mathcal{R}}$ is complete on R (Lemma 3.6.11), we have $i' \in \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).I$.

From $i' \in \mu\psi_{\mathcal{R}}(i, \sigma, s_1; (s_2; a)).I$ and Item (3) of Lemma (3.6.1), we have

$$(vi) \quad \mu\psi_{\mathcal{R}}(i', \sigma, s_2; a).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, (s_1; s_2); a).k$$

Subtracting $|\sigma|$ on both sides of the inequality (vii) and replacing with (i) and (vi), we have $\mathbb{K}_{i'}^{\sigma, s_2; a} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$. From $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_2; a \rangle$ and Item (3) of Proposition (3.6.5), we have either $i' = j$, in which case it follows $\mathbb{K}_j^{\sigma, s_2; a} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$; or $i' \neq j \wedge \eta(s_2; a) \in \mathcal{L}$, which we prove below. From $\eta(s_2; a) \in \mathcal{L}$, $\langle imr_{i'}, \overline{T}_{i'}, \sigma, s_2; a \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, s_2; a \rangle$ and Item (1) of Lemma (3.6.13), we have

$$(vii) \quad \mu\psi_{\mathcal{R}}(j, \sigma, s_2; a).k \leq \mu\psi_{\mathcal{R}}(i', \sigma, s_2; a).k$$

Subtracting $|\sigma|$ on both sides of the inequality (vii) and replacing with (iv) and (v), we have $\mathbb{K}_j^{\sigma, s_2; a} \leq \mathbb{K}_{i'}^{\sigma, s_2; a}$. From $\mathbb{K}_j^{\sigma, s_2; a} \leq \mathbb{K}_{i'}^{\sigma, s_2; a}$ and $\mathbb{K}_{i'}^{\sigma, s_2; a} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$, we have $\mathbb{K}_j^{\sigma, s_2; a} \leq \mathbb{K}_i^{\sigma, (s_1; s_2); a}$.

3. Let $I = \mathbb{I}_{i', u}^{\sigma, h} = \mathbb{I}_{i', u}^{\sigma, s; h'}$. From $\mathbb{I}_{i', u}^{\sigma, h} \neq \emptyset$ and we have $I \neq \emptyset$.

We first prove $\mathbb{K}_{i', u}^{\sigma, s; h'} \leq \mathbb{K}_{i', u}^{\sigma, h}$. From $I \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$$\mathbb{K}^{\sigma, h} = \max(\{\mathbb{K}_i^{\sigma, h} \mid i \in I\})$$

$$\mathbb{K}^{\sigma, s; h'} = \max(\{\mathbb{K}_i^{\sigma, s; h'} \mid i \in I\})$$

It suffices to prove that for any $i \in I$, we have $\mathbb{K}_i^{\sigma, s; h'} \leq \mathbb{K}_i^{\sigma, h}$. From Item (4) of Definition (3.6.8), we have

$$(i) \quad \mathbb{K}_i^{\sigma, h} = \mu\psi_{\mathcal{R}}(i, \sigma, h).k - |\sigma|$$

$$(ii) \quad \mathbb{K}_i^{\sigma, s; h'} = \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k - |\sigma|$$

From Item (5) of Lemma (3.6.1), we have

$$(iii) \quad \mu\psi_{\mathcal{R}}(i, \sigma, s; h').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$$

Subtracting $|\sigma|$ on both sides of inequalities (iii) and replacing with (i) and (ii), we have $\mathbb{K}_i^{\sigma, s; h'} \leq \mathbb{K}_i^{\sigma, h}$.

We next prove $\mathbb{K}_{i', u}^{\sigma, h'} \leq \mathbb{K}_{i', u}^{\sigma, h}$. From $\mathbb{I}_{i', u}^{\sigma, s; h'} \neq \emptyset$ and Corollary (3.6.3), we have $\mathbb{I}_{i', u}^{\sigma, h'} \neq \emptyset$. From $\mathbb{I}_{i', u}^{\sigma, h'} \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$$\mathbb{K}_{i', u}^{\sigma, h'} = \max(\{\mathbb{K}_j^{\sigma, h'} \mid j \in \mathbb{I}_{i', u}^{\sigma, h'}\})$$

It suffices to prove that for any $j \in \mathbb{I}_{i', u}^{\sigma, h'}$, there exists $i \in \mathbb{I}_{i', u}^{\sigma, h}$ such that $\mathbb{K}_j^{\sigma, h'} \leq \mathbb{K}_i^{\sigma, h}$. From $j \in \mathbb{I}_{i', u}^{\sigma, h'}$ and Item (4) of Definition (3.6.8), we have

$$(iv) \quad \mathbb{K}_j^{\sigma, h'} = \mu\psi_{\mathcal{R}}(j, \sigma, h').k - |\sigma|$$

From $\mathbb{I}_{i', u}^{\sigma, s; h'} \neq \emptyset$, $j \in \mathbb{I}_{i', u}^{\sigma, h'}$ and Lemma (3.6.29), there exists $i \in \mathbb{I}_{i', u}^{\sigma, s; h'}$ such that $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, h' \rangle$. From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, h' \rangle$ and Item (2) of Proposition (3.6.5), there exists i' such that $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h' \rangle$ and $\langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{\overline{T}}_j, \sigma, h' \rangle$. From Item (4) of Definition (3.6.8), we have

$$(v) \quad \mathbb{K}_{i'}^{\sigma, h'} = \mu\psi_{\mathcal{R}}(i', \sigma, h').k - |\sigma|$$

From $\langle imr_i, \overline{\overline{T}}_i, \sigma, s; h' \rangle \xrightarrow{\sigma} \langle imr_{i'}, \overline{\overline{T}}_{i'}, \sigma, h' \rangle$ and $\mu\psi_{\mathcal{R}}$ is complete on R

(Lemma 3.6.11), we have $i' \in \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$. From $i' \in \mu\psi_{\mathcal{R}}(i, \sigma, s; h').I$ and Item (5) of Lemma (3.6.1), we have

$$(vi) \quad \mu\psi_{\mathcal{R}}(i', \sigma, h').k \leq \mu\psi_{\mathcal{R}}(i, \sigma, h).k$$

Subtracting $|\sigma|$ on both sides of the inequality (vi) and replacing with (i) and (vi), we have $\mathbb{K}_{i'}^{\sigma, h'} \leq \mathbb{K}_i^{\sigma, h}$. From $\langle imr_{i'}, \overline{T}_{i'}, \sigma, h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, h' \rangle$ and Item (3) of Proposition (3.6.5), we have either $i' = j$, in which case it follows $\mathbb{K}_j^{\sigma, h'} \leq \mathbb{K}_i^{\sigma, h}$; or $i' \neq j \wedge \eta(h') \in \mathcal{L}$, which we prove below. From $\eta(h') \in \mathcal{L}$, $\langle imr_{i'}, \overline{T}_{i'}, \sigma, h' \rangle \xrightarrow{\sigma^+} \langle imr_j, \overline{T}_j, \sigma, h' \rangle$ and Item (2) of Lemma (3.6.13), we have

$$(vii) \quad \mu\psi_{\mathcal{R}}(j, \sigma, h').k \leq \mu\psi_{\mathcal{R}}(i', \sigma, h').k$$

Subtracting $|\sigma|$ on both sides of the inequality (vii) and replacing with (iv) and (v), we have $\mathbb{K}_j^{\sigma, h'} \leq \mathbb{K}_{i'}^{\sigma, h'}$. From $\mathbb{K}_j^{\sigma, h'} \leq \mathbb{K}_{i'}^{\sigma, h'}$ and $\mathbb{K}_{i'}^{\sigma, h'} \leq \mathbb{K}_i^{\sigma, h}$, we have $\mathbb{K}_j^{\sigma, h'} \leq \mathbb{K}_i^{\sigma, h}$.

■

Lemma 3.6.32 Suppose $i \in \mathbb{I}_{i', u}^{\sigma, a}$,

1. If $a = s; a'$ and s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**, then we have $\alpha(\xi_s(i)) = \xi_s(i)$.
2. If $a = \mathbf{iret}$, then we have $\alpha(\xi_{\mathbf{iret}}(i)) = \xi_{\mathbf{iret}}(i)$.
3. If $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$ and $\eta(a) \in \mathcal{L}$, then we have $a ::_S \sigma = a :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\zeta_u(i) \in \mathbb{I}^{a :: \sigma, \overline{h}(u)}$.

Proof From $i \in \mathbb{I}_{i', u}^{\sigma, a}$ and Item (1) of Proposition (3.6.8), we have $\langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}$.

1. From $\langle imr_i, \overline{T}_i, \sigma, s; a' \rangle \in \mathcal{R}$, s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip** and Item (1) of Lemma (3.6.5), we have $\alpha(\xi_s(i)) = \xi_s(i)$.
2. From $\langle imr_i, \overline{T}_i, \sigma, \mathbf{iret} \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have state $\langle imr_i, \overline{T}_i, \sigma, \mathbf{iret} \rangle$ is consistent from which it follows $\sigma = a :: \sigma'$. From $\langle imr_i, \overline{T}_i, \sigma, \mathbf{iret} \rangle \in \mathcal{R}$, a is **iret**, $\sigma = a :: \sigma'$ and Item (3) of Lemma (3.6.5), we have $\alpha(\xi_{\mathbf{iret}}(i)) = \xi_{\mathbf{iret}}(i)$.

3. From $\langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}$, $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, $\eta(a) \in \mathcal{L}$ and Item (4) of Lemma (3.6.5), we have $a::_S\sigma = a::\sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\langle imr_i, \overline{T}_i, \sigma, a \rangle \leftrightarrow \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a::\sigma, \overline{h}(u) \rangle$. From $\langle imr_i, \overline{T}_i, \sigma, a \rangle \in \mathcal{R}$ and $\langle imr_i, \overline{T}_i, \sigma, a \rangle \leftrightarrow \langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a::\sigma, \overline{h}(u) \rangle$, we have $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a::\sigma, \overline{h}(u) \rangle \in \mathcal{R}$. From $\langle imr_{\zeta_u(i)}, \overline{T}_{\zeta_u(i)}, a::\sigma, \overline{h}(u) \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $\zeta_u(i) \in \mathbb{I}^{a::\sigma, \overline{h}(u)}$.

■

Lemma 3.6.33 *If $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, where s is $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$, then we have*

1. $\mathbb{J}_i^{\sigma,s;a} = \bigcup \{ \mathbb{J}_j^{\sigma,s;a} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_s(i) \}$.
2. $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \Rightarrow \mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma,s;a}$.

Proof From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$ and Item (1) of Proposition (3.6.8), we have $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \in \mathcal{R}$. From $\langle imr_i, \overline{T}_i, \sigma, s; a \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $i \in \mathbb{I}^{\sigma,s;a}$.

1. From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, s is $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or $skip$ and Item (1) of Lemma (3.6.32), we have $\alpha(\xi_s(i)) = \xi_s(i)$. From $\alpha(\xi_s(i)) = \xi_s(i)$ and Item (1) of Proposition (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, s; a).I \mid j \in J \} \cup \{ (\xi_s(i)) \}$$

where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a::_S\sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \}$. There are two subcases depending on whether $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$.

Subcase 1. $\widehat{\mathcal{P}}(imr_i, \overline{T}_i) = \emptyset$.

From (i), we have $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).I = \{ (\xi_s(i)) \}$. From $\mu\psi_{\mathcal{R}}(i, \sigma, s; a).I = \{ (\xi_s(i)) \}$ and $\mathbb{J}_i^{\sigma,s;a} = \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$, we have $\mathbb{J}_i^{\sigma,s;a} = \{ (\xi_s(i)) \}$.

Subcase 2. Suppose $u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$.

From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, Item (3) of Lemma (3.6.32), we have $s; a::_S\sigma = s; a::\sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\zeta_u(i) \in \mathbb{I}^{s; a::\sigma, \overline{h}(u)}$.

From $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a::_S\sigma, \overline{h}(u)).I \mid u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i) \}$, $s; a::_S\sigma = s; a::$

σ and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we have $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$. From Item (3) of Definition (3.6.8), we have

- (ii) $\mathbb{J}_i^{\sigma, s; a} = \mu\psi_{\mathcal{R}}(i, \sigma, s; a).I$
- (iii) $\mathbb{J}_j^{\sigma, s; a} = \mu\psi_{\mathcal{R}}(j, \sigma, s; a).I$ where $j \in J$
- (iv) $\mathbb{J}_{\zeta_u(i)}^{s; a :: \sigma, \bar{h}(u)} = \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).I$

Replacing (i) and J with (ii), (iii) and (iv), we have

$$(v) \quad \mathbb{J}_i^{\sigma, s; a} = \bigcup \{ \mathbb{J}_j^{\sigma, s; a} \mid j \in J \} \cup \{ \xi_s(i) \}$$

where $J = \bigcup \{ \mathbb{J}_{\zeta_u(i)}^{s; a :: \sigma, \bar{h}(u)} \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$. From $i \in \mathbb{I}^{\sigma, s; a}$, $\zeta_u(i) \in \mathbb{I}^{s; a :: \sigma, \bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$. From $i \in \mathbb{I}^{\sigma, s; a}$, $\zeta_u(i) \in \mathbb{I}^{s; a :: \sigma, \bar{h}(u)}$ and Item (2) of Definition (3.6.9), we have $s; a :: \sigma \in \mathbb{W}^u$. From $i \in \mathbb{I}^u$, $s; a :: \sigma \in \mathbb{W}^u$ and Item (3) of Definition (3.6.9), we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{s; a :: \sigma, \bar{h}(u)}$. From $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{s; a :: \sigma, \bar{h}(u)}$ and (v), we have $\mathbb{J}_i^{\sigma, s; a} = \bigcup \{ \mathbb{J}_j^{\sigma, s; a} \mid j \in J \} \cup \{ \xi_s(i) \}$, where $J = \bigcup \{ \mathbb{J}_i^u \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$, which is $\mathbb{J}_i^{\sigma, s; a} = \bigcup \{ \mathbb{J}_j^{\sigma, s; a} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_s(i) \}$.

2. Suppose $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$. From s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$ and Item (1) of Lemma (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), s; a ::_S \sigma, \bar{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$$

From s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , we have $\eta(s; a) \in \mathcal{L}$. From $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$, $\eta(s; a) \in \mathcal{L}$ and Item (3) of Lemma (3.6.32), we have $s; a ::_S \sigma = s; a :: \sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i) \in \mathbb{I}^{s; a :: \sigma, \bar{h}(u)}$. Replacing (i) with $s; a ::_S \sigma = s; a :: \sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we have

$$(ii) \quad \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k$$

From Item (4) of Definition (3.6.8), we have

- (iii) $\mathbb{K}_i^{\sigma, s; a} = \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k - |\sigma|$
- (iv) $\mathbb{K}_{\zeta_u(i)}^{s; a :: \sigma, \bar{h}(u)} = \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).k - |s; a :: \sigma|$
 $= \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).k - |\sigma| - 1$

Subtracting $|\sigma|$ from both sides of (ii), we have

$$(v) \quad \mu\psi_{\mathcal{R}}(\zeta_u(i), s; a :: \sigma, \bar{h}(u)).k - |\sigma| \leq \mu\psi_{\mathcal{R}}(i, \sigma, s; a).k - |\sigma|$$

Replacing (v) with (iii) and (iv), we have

$$(vi) \quad \mathbb{K}_{\zeta_u(i)}^{s;a::\sigma,\bar{h}(u)} + 1 \leq \mathbb{K}_i^{\sigma,s;a}$$

From $i \in \mathbb{I}^{\sigma,s;a}$, $\zeta_u(i) \in \mathbb{I}^{s;a::\sigma,\bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$. From $i \in \mathbb{I}^{\sigma,s;a}$, $\zeta_u(i) \in \mathbb{I}^{s;a::\sigma,\bar{h}(u)}$ and Item (2) of Definition (3.6.9), we have $s;a :: \sigma \in \mathbb{W}_i^u$. From $i \in \mathbb{I}^u$, $s;a :: \sigma \in \mathbb{W}_i^u$ and Item (4) of Definition (3.6.9), we have $\mathbb{K}_{\zeta_u(i)}^{s;a::\sigma,\bar{h}(u)} = \mathbb{K}_i^u$. From $\mathbb{K}_{\zeta_u(i)}^{s;a::\sigma,\bar{h}(u)} = \mathbb{K}_i^u$ and (vi), we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma,s;a}$.

■

Lemma 3.6.34 *If $i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}$, then we have*

$$1. \mathbb{J}_i^{\sigma,\text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma,\text{iret}} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_{\text{iret}}(i) \}.$$

$$2. u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \Rightarrow \mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma,\text{iret}}.$$

Proof From $i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$. From $\langle \text{imr}_i, \overline{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $i \in \mathbb{I}^{\sigma,\text{iret}}$.

$$1. \text{ From } i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}} \text{ and Item (2) of Lemma (3.6.32), we have } \alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i).$$

From $\alpha(\xi_{\text{iret}}(i)) = \xi_{\text{iret}}(i)$ and Item (4) of Proposition (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I = \bigcup \{ \mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).I \mid j \in J \} \cup \{ (\xi_{\text{iret}}(i)) \}$$

where $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::s\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \}$. There are two subcases depending on whether $\widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) = \emptyset$.

Subcase 1. $\widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) = \emptyset$.

From (i), we have $\mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I = \{ (\xi_{\text{iret}}(i)) \}$. From $\mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I = \{ (\xi_{\text{iret}}(i)) \}$

and $\mathbb{J}_i^{\sigma,\text{iret}} = \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I$, we have $\mathbb{J}_i^{\sigma,\text{iret}} = \{ (\xi_{\text{iret}}(i)) \}$.

Subcase 2. Suppose $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i)$.

From $i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}$, Item (3) of Lemma (3.6.32), we have $\text{iret}::s\sigma = \text{iret} :: \sigma$, $\alpha(\zeta_u(i)) = \zeta_u(i)$ and $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma,\bar{h}(u)}$.

From $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::s\sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \}$, $\text{iret}::s\sigma = \text{iret} :: \sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we have $J = \bigcup \{ \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret} :: \sigma, \bar{h}(u)).I \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{T}_i) \}$. From Item (3) of Definition (3.6.8), we have

- (ii) $\mathbb{J}_i^{\sigma, \text{iret}} = \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).I$
- (iii) $\mathbb{J}_j^{\sigma, \text{iret}} = \mu\psi_{\mathcal{R}}(j, \sigma, \text{iret}).I$ where $j \in J$
- (iv) $\mathbb{J}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} = \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret}::\sigma, \bar{h}(u)).I$

Replacing (i) and J with (ii), (iii) and (iv), we have

$$(v) \quad \mathbb{J}_i^{\sigma, \text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma, \text{iret}} \mid j \in J \} \cup \{ \xi_{\text{iret}}(i) \}$$

where $J = \bigcup \{ \mathbb{J}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$. From $i \in \mathbb{I}^{\sigma, \text{iret}}$, $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$. From $i \in \mathbb{I}^{\sigma, \text{iret}}$, $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$ and Item (2) of Definition (3.6.9), we have $\text{iret}::\sigma \in \mathbb{W}^u$. From $i \in \mathbb{I}^u$, $\text{iret}::\sigma \in \mathbb{W}^u$ and Item (3) of Definition (3.6.9), we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)}$. From $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)}$ and (v), we have $\mathbb{J}_i^{\sigma, \text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma, \text{iret}} \mid j \in J \} \cup \{ \xi_{\text{iret}}(i) \}$, where $J = \bigcup \{ \mathbb{J}_i^u \mid u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \}$, which is $\mathbb{J}_i^{\sigma, \text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma, \text{iret}} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_{\text{iret}}(i) \}$.

2. Suppose $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$. From a is iret , $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$ and Item (4) of Lemma (3.6.1), we have

$$(i) \quad \mu\psi_{\mathcal{R}}(\alpha(\zeta_u(i)), \text{iret}::_{S\sigma}, \bar{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k$$

From $u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$, $\eta(\text{iret}) \in \mathcal{L}$ and Item (3) of Lemma (3.6.32), we have $\text{iret}::_{S\sigma} = \text{iret}::\sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$. Replacing (i) with $\text{iret}::_{S\sigma} = \text{iret}::\sigma$ and $\alpha(\zeta_u(i)) = \zeta_u(i)$, we have

$$(ii) \quad \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret}::\sigma, \bar{h}(u)).k \leq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k$$

From Item (4) of Definition (3.6.8), we have

$$(iii) \quad \mathbb{K}_i^{\sigma, \text{iret}} = \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k - |\sigma|$$

$$(iv) \quad \mathbb{K}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} = \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret}::\sigma, \bar{h}(u)).k - |\text{iret}::\sigma| \\ = \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret}::\sigma, \bar{h}(u)).k - |\sigma| - 1$$

Subtracting $|\sigma|$ from both sides of (ii), we have

$$(v) \quad \mu\psi_{\mathcal{R}}(\zeta_u(i), \text{iret}::\sigma, \bar{h}(u)).k - |\sigma| \leq \mu\psi_{\mathcal{R}}(i, \sigma, \text{iret}).k - |\sigma|$$

Replacing (v) with (iii) and (iv), we have

$$(vi) \quad \mathbb{K}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} + 1 \leq \mathbb{K}_i^{\sigma, \text{iret}}$$

From $i \in \mathbb{I}^{\sigma, \text{iret}}$, $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$. From $i \in \mathbb{I}^{\sigma, \text{iret}}$, $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$ and Item (2) of Definition (3.6.9), we have

$\text{iret} :: \sigma \in \mathbb{W}_i^u$. From $i \in \mathbb{I}^u$, $\text{iret} :: \sigma \in \mathbb{W}_i^u$ and Item (4) of Definition (3.6.9), we have $\mathbb{K}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} = \mathbb{K}_i^u$. From $\mathbb{K}_{\zeta_u(i)}^{\text{iret}::\sigma, \bar{h}(u)} = \mathbb{K}_i^u$ and (vi), we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma, \text{iret}}$. ■

Lemma 3.6.35 For $u \in 0..N$, if $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{r}$, then

1. $\hat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{r}$
2. $v \in \hat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \Rightarrow \hat{\theta}_v(\bar{T}_i) \leq \bar{r}$

Proof From $i \in \mathbb{I}_{i',u}^{\sigma,s;a}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \bar{T}_i, \sigma, s; a \rangle \in \mathcal{R}$.

1. From s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $\langle \text{imr}_i, \bar{T}_i, \sigma, s; a \rangle \in \mathcal{R}$, by applying Rule (3.13)-(3.15),(3.18), depending on what s is, we have $\langle \text{imr}_{\xi_s(i)}, \bar{T}_{\xi_s(i)}, \sigma, a \rangle \in \mathcal{R}$. From $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{r}$ and $\langle \text{imr}_{\xi_s(i)}, \bar{T}_{\xi_s(i)}, \sigma, a \rangle \in \mathcal{R}$, we have $\bar{T}_{\xi_s(i)} \leq \bar{r}$, which is $\hat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{r}$.
2. From s is $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , we have $\eta(s; a) \in \mathcal{L}$. From $v \in \hat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)$, $\eta(s; a) \in \mathcal{L}$, $\langle \text{imr}_i, \bar{T}_i, \sigma, s; a \rangle \in \mathcal{R}$, by applying Rule (3.10), we have $\langle \text{imr}_{\zeta_v(i)}, \bar{T}_{\zeta_v(i)}, (s; a) :: \sigma, \bar{h}(v) \rangle \in \mathcal{R}$. From $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{r}$ and $\langle \text{imr}_{\zeta_v(i)}, \bar{T}_{\zeta_v(i)}, (s; a) :: \sigma, \bar{h}(v) \rangle \in \mathcal{R}$, we have $\bar{T}_{\zeta_v(i)} \leq \bar{r}$, which is $\hat{\theta}_v(\bar{T}_i) \leq \bar{r}$. ■

Lemma 3.6.36 For $u \in 1..N$, if $i \in \mathbb{I}_{i',u}^{\sigma, \text{iret}}$ and $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{r}$, then

1. $\hat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{r}$
2. $v \in \hat{\mathcal{P}}(\text{imr}_i, \bar{T}_i) \Rightarrow \hat{\theta}_v(\bar{T}_i) \leq \bar{r}$

Proof From $i \in \mathbb{I}_{i',u}^{\sigma, \text{iret}}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \bar{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$. From $\langle \text{imr}_i, \bar{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and Corollary (3.3.1), we have $\langle \text{imr}_i, \bar{T}_i, \sigma, \text{iret} \rangle$ is consistent, from which it follows $\sigma = a :: \sigma'$.

1. From $\langle imr_i, \overline{T}_i, \sigma, iret \rangle \in \mathcal{R}$, by applying Rule (3.11), we have $\langle imr_{\xi_{iret}(i)}, \overline{T}_{\xi_{iret}(i)}, \sigma', a \rangle \in \mathcal{R}$. From $Q \in \mathcal{R} \Rightarrow Q.\overline{T} \leq \overline{r}$ and $\langle imr_{\xi_{iret}(i)}, \overline{T}_{\xi_{iret}(i)}, \sigma', a \rangle \in \mathcal{R}$, we have $\overline{T}_{\xi_{iret}(i)} \leq \overline{r}$, which is $\widehat{\theta}(imr_i, \overline{T}_i) \leq \overline{r}$.
2. Notice that $\eta(iret) \in \mathcal{L}$. From $v \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)$, $\eta(iret) \in \mathcal{L}$, $\langle imr_i, \overline{T}_i, \sigma, iret \rangle \in \mathcal{R}$, by applying Rule (3.10), we have $\langle imr_{\zeta_v(i)}, \overline{T}_{\zeta_v(i)}, iret :: \sigma, \overline{h}(v) \rangle \in \mathcal{R}$. From $Q \in \mathcal{R} \Rightarrow Q.\overline{T} \leq \overline{r}$ and $\langle imr_{\zeta_v(i)}, \overline{T}_{\zeta_v(i)}, iret :: \sigma, \overline{h}(v) \rangle \in \mathcal{R}$, we have $\overline{T}_{\zeta_v(i)} \leq \overline{r}$, which is $\widehat{\theta}_v(\overline{T}_i) \leq \overline{r}$.

■

We are now ready to build and prove the type derivation for statements, main program and handlers. We first prove the type derivation for statements can be derived.

Lemma 3.6.37 (Type derivation for statement) *If $Q \in \mathcal{R} \Rightarrow Q.\overline{T} \leq \overline{r}$ and the type judgment for statement s (3.60) is well formed, then we can derive*

$$\overline{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s;a}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;a}} imr_j, \overline{T}_j) \quad (3.60)$$

Proof From the type judgment for statement s (3.60) is well formed and Item (1) of Definition (3.4.2), we have $\mathbb{I}_{i',v}^{\sigma,s;a} \neq \emptyset$. We prove by induction on s . There are three cases.

1. s is either $x := e$, $imr = imr \wedge imr$, $imr = imr \vee imr$ or **skip**.

To derive $\overline{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s;a}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;a}} imr_j, \overline{T}_j)$, we need to prove the safety condition $safe(\overline{r}_{\mathcal{R}}, \mathbb{I}_{i',v}^{\sigma,s;a}, \mathbb{K}_{i',v}^{\sigma,s;a})$ which consists of the following two items:

$$(a) \quad (i \in \mathbb{I}_{i',v}^{\sigma,s;a} \wedge u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)) \Rightarrow \begin{cases} (i) & i \in \mathbb{I}^u \\ (ii) & \mathbb{J}_i^u \subseteq \mathbb{I}_{i',v}^{\sigma,s;a} \\ (iii) & \mathbb{K}_i^u + 1 \leq \mathbb{K}_{i',v}^{\sigma,s;a} \\ (iv) & \widehat{\theta}_u(\overline{T}_i) \leq \overline{r} \end{cases}$$

$$(b) \quad \forall i \in \mathbb{I}_{i',v}^{\sigma,s;a} : \widehat{\theta}(imr_i, \overline{T}_i) \leq \overline{r}$$

$$(c) \quad \forall i \in \mathbb{I}_{i',v}^{\sigma,s;a} : \mathbb{J}_i^{\sigma,s;a} = \bigcup \{ \mathbb{J}_j^{\sigma,s;a} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(imr_i, \overline{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_s(i) \}$$

- Proof of Item (a).

- Proof of Item (i). From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \in \mathcal{R}$. From $\langle \text{imr}_i, \overline{\overline{T}}_i, \sigma, s; a \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $i \in \mathbb{I}^{\sigma,s;a}$. From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, $\eta(s; a) \in \mathcal{L}$, $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\overline{T}}_i)$ and Item (3) of Lemma (3.6.32), we have $\zeta_u(i) \in \mathbb{I}^{s;a::\sigma, \overline{h}(u)}$. From $i \in \mathbb{I}^{\sigma,s;a}$, $\zeta_u(i) \in \mathbb{I}^{s;a::\sigma, \overline{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$.
- Proof of Item (ii). From $\eta(s; a) \in \mathcal{L}$, $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\overline{T}}_i)$ and Lemma (3.6.28), we have $\mathbb{J}_i^u \subseteq \mathbb{I}'_{i',v}{}^{\sigma,s;a}$.
- Proof of Item (iii). From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$, s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\overline{T}}_i)$ and Item (2) of Lemma (3.6.33), we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma,s;a}$. From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$ and Lemma (3.6.14), we have $\mathbb{K}_i^{\sigma,s;a} \leq \mathbb{K}'_{i',v}{}^{\sigma,s;a}$. From $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma,s;a}$ and $\mathbb{K}_i^{\sigma,s;a} \leq \mathbb{K}'_{i',v}{}^{\sigma,s;a}$, we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}'_{i',v}{}^{\sigma,s;a}$.
- Proof of Item (iv). From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\overline{T}}_i)$ and $Q \in \mathcal{R} \Rightarrow Q.\overline{\overline{T}} \leq \overline{r}$ and Item (2) of Lemma (3.6.35), we have $\widehat{\theta}_u(\overline{\overline{T}}_i) \leq \overline{r}$.

- Proof of Item (b).

From $i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a}$ and s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip and $Q \in \mathcal{R} \Rightarrow Q.\overline{\overline{T}} \leq \overline{r}$ and Item (1) of Lemma (3.6.35), we have $\widehat{\theta}(\text{imr}_i, \overline{\overline{T}}_i) \leq \overline{r}$.

- Proof of Item (c).

From $\mathbb{I}'_{i',v}{}^{\sigma,s;a} \neq \emptyset$, s is either $x := e$, $\text{imr} = \text{imr} \wedge \text{imr}$, $\text{imr} = \text{imr} \vee \text{imr}$ or skip , Item (1) of Lemma (3.6.33), we have $\forall i \in \mathbb{I}'_{i',v}{}^{\sigma,s;a} : \mathbb{J}_i^{\sigma,s;a} = \bigcup \{ \mathbb{J}_j^{\sigma,s;a} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \overline{\overline{T}}_i)} \mathbb{J}_i^u \} \cup \{ \xi_s(i) \}$.

2. s is if0 x then s_1 else s_2 .

From s is if0 x then s_1 else s_2 and Item (1) of Proposition (3.6.11), we have $\mathbb{I}'_{i',v}{}^{\sigma,s;a} = \mathbb{I}'_{i',v}{}^{\sigma,s_1;a} = \mathbb{I}'_{i',v}{}^{\sigma,s_2;a}$. From $\mathbb{I}'_{i',v}{}^{\sigma,s;a} \neq \emptyset$, and $\mathbb{I}'_{i',v}{}^{\sigma,s;a} = \mathbb{I}'_{i',v}{}^{\sigma,s_1;a} = \mathbb{I}'_{i',v}{}^{\sigma,s_2;a}$, we have

$\mathbb{I}_{i',v}^{\sigma,s_1;a} \neq \emptyset$ and $\mathbb{I}_{i',v}^{\sigma,s_2;a} \neq \emptyset$. From $\mathbb{I}_{i',v}^{\sigma,s_1;a} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{\overline{T}} \leq \overline{\overline{r}} \wedge Q.stk \leq K)$ and Lemma (3.6.25), we have that the type judgment (i) below for s_1 is well formed. From $\mathbb{I}_{i',v}^{\sigma,s_2;a} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{\overline{T}} \leq \overline{\overline{r}} \wedge Q.stk \leq K)$ and Lemma (3.6.25), we have that the type judgment (ii) below for s_2 is well formed. From the type judgment (i) below for s_1 is well formed and the induction hypothesis, we have

$$(i) \quad \overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s_1;a}} s_1 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_1;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_1;a}} imr_j, \overline{\overline{T}}_j)$$

From the type judgment (ii) below for s_2 is well formed and the induction hypothesis, we have

$$(ii) \quad \overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s_2;a}} s_2 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_2;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_2;a}} imr_j, \overline{\overline{T}}_j)$$

From $\mathbb{I}_{i',v}^{\sigma,s;a} \neq \emptyset$ and Item (1) of Lemma (3.6.31), we have $\mathbb{K}_{i',v}^{\sigma,s_1;a} \leq \mathbb{K}_{i',v}^{\sigma,s;a}$ and $\mathbb{K}_{i',v}^{\sigma,s_2;a} \leq \mathbb{K}_{i',v}^{\sigma,s;a}$. From $\mathbb{K}_{i',v}^{\sigma,s_1;a} \leq \mathbb{K}_{i',v}^{\sigma,s;a}$, (i) and Lemma (3.4.5), we have

$$(iii) \quad \overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s_1 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_1;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_1;a}} imr_j, \overline{\overline{T}}_j)$$

From $\mathbb{K}_{i',v}^{\sigma,s_2;a} \leq \mathbb{K}_{i',v}^{\sigma,s;a}$, (ii) and Lemma (3.4.5), we have

$$(iv) \quad \overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s_2 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_2;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_2;a}} imr_j, \overline{\overline{T}}_j)$$

From (iii), (iv), $\mathbb{I}_{i',v}^{\sigma,s;a} = \mathbb{I}_{i',v}^{\sigma,s_1;a} = \mathbb{I}_{i',v}^{\sigma,s_2;a}$, and Rule (3.30), we have

$$(v) \quad \overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_1;a} \cup \mathbb{J}_i^{\sigma,s_2;a}} imr_j, \overline{\overline{T}}_j).$$

From s is $\text{if0 } x \text{ then } s_1 \text{ else } s_2$ and Item (1) of Lemma (3.6.30), we have

$$(vi) \quad \mathbb{J}_i^{\sigma,s;a} = \mathbb{J}_i^{\sigma,s_1;a} \cup \mathbb{J}_i^{\sigma,s_2;a}$$

From (v) and (vi), we have $\overline{\overline{r}}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s;a}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s;a}} (imr_i, \overline{\overline{T}}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;a}} imr_j, \overline{\overline{T}}_j)$.

3. s is $s_1; s_2$.

From Item (2) of Proposition (3.6.11), we have $\mathbb{I}_{i',v}^{\sigma,(s_1;s_2);a} = \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)}$. From $\mathbb{I}_{i',v}^{\sigma,(s_1;s_2);a} \neq \emptyset$ and $\mathbb{I}_{i',v}^{\sigma,(s_1;s_2);a} = \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)}$, we have $\mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} \neq \emptyset$. From $\mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} \neq \emptyset$ and Lemma (3.6.19), we have $\forall i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} : \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \subseteq \mathbb{I}_{i',v}^{\sigma,s_2;a}$. From Lemma (3.6.18), we have $\forall i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} : \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \neq \emptyset$. From $\forall i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} : \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \neq \emptyset$ and $\forall i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} : \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \subseteq \mathbb{I}_{i',v}^{\sigma,s_2;a}$, we have $\mathbb{I}_{i',v}^{\sigma,s_2;a} \neq \emptyset$. From $\mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{\overline{T}} \leq \overline{\overline{r}} \wedge Q.stk \leq K)$ and Lemma (3.6.25), we have that the type judgment (i) below for s_1 is well formed. From $\mathbb{I}_{i',v}^{\sigma,s_2;a} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{\overline{T}} \leq \overline{\overline{r}} \wedge Q.stk \leq K)$ and Lemma (3.6.25), we have that the type judgment (ii) below for s_2 is well formed. From the type judgment (i) below for

s_1 is well formed, and the induction hypothesis, we have

$$(i) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s_1;(s_2;a)}} s_1 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_1;(s_2;a)}} imr_j, \bar{T}_j)$$

From the type judgment (ii) below for s_2 is well formed, and the induction hypothesis, we have

$$(ii) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,s_2;a}} s_2 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_2;a}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_2;a}} imr_j, \bar{T}_j).$$

From $\mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)} \neq \emptyset$ and Item (2) of Lemma (3.6.31), we have $\mathbb{K}_{i',v}^{\sigma,s_1;(s_2;a)} \leq \mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}$ and $\mathbb{K}_{i',v}^{\sigma,s_2;a} \leq \mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}$. From $\mathbb{K}_{i',v}^{\sigma,s_1;(s_2;a)} \leq \mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}$, (i), and Lemma (3.4.5), we have

$$(iii) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}} s_1 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_1;(s_2;a)}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_1;(s_2;a)}} imr_j, \bar{T}_j)$$

From $\mathbb{K}_{i',v}^{\sigma,s_2;a} \leq \mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}$, (i), and Lemma (3.4.5), we have

$$(iv) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}} s_2 : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,s_2;a}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s_2;a}} imr_j, \bar{T}_j).$$

From Item (2) of Lemma (3.6.30), we have

$$(v) \quad \mathbb{J}_i^{\sigma,(s_1;s_2);a} = \bigcup \{ \mathbb{J}_j^{\sigma,s_2;a} \mid j \in \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \}$$

From (iii), (iv), $\forall i \in \mathbb{I}_{i',v}^{\sigma,s;a} : \mathbb{J}_i^{\sigma,s_1;(s_2;a)} \subseteq \mathbb{I}_{i',v}^{\sigma,s_2;a}$ (from Lemma (3.6.19)), (v) and

Rule (3.31), we have

$$\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,(s_1;s_2);a}} (s_1; s_2) : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,(s_1;s_2);a}} (imr_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,(s_1;s_2);a}} imr_j, \bar{T}_j).$$

■

We next prove the type derivation for main program can be derived.

Lemma 3.6.38 *If $Q \in \mathcal{R} \Rightarrow Q.stk \leq K$ and $\mathbb{I}_{i_0,0}^{nil,s;a} \neq \emptyset$, then we have $\mathbb{K}_{i_0,0}^{nil,s;a} \leq K$.*

Proof From $\mathbb{I}_{i_0,0}^{nil,s;a} \neq \emptyset$ and Item (5) of Definition (3.6.8), we have

$\mathbb{K}_{i_0,0}^{nil,s;a} = \max(\{ \mathbb{K}_i^{nil,s;a} \mid i \in \mathbb{I}_{i_0,0}^{nil,s;a} \})$. From $Q \in \mathcal{R} \Rightarrow Q.stk \leq K$ and Lemma (3.6.16), we have $\forall i \in \mathbb{I}_{i_0,0}^{nil,s;a} : \mathbb{K}_i^{nil,s;a} \leq K$. From $\forall i \in \mathbb{I}_{i_0,0}^{nil,s;a} : \mathbb{K}_i^{nil,s;a} \leq K$ and $\mathbb{K}_{i_0,0}^{nil,s;a} = \max(\{ \mathbb{K}_i^{nil,s;a} \mid i \in \mathbb{I}_{i_0,0}^{nil,s;a} \})$, we have $\mathbb{K}_{i_0,0}^{nil,s;a} \leq K$. main program

■

Lemma 3.6.39 (Type derivation for main) *If $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{\tau} \wedge Q.stk \leq K)$, then we can derive $\bar{\tau}_{\mathcal{R}} \vdash_K m : \bigvee_{i \in \mathbb{I}_{i_0,0}^{nil,m}} imr_i, \bar{T}_i$.*

Proof We prove by induction on m . There are two cases.

1. $m = \text{loop } s$.

From Item (3) of Proposition (3.6.11), we have $\mathbb{I}_{i_0,0}^{\text{nil,loop } s} = \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s}$. From Lemma (3.6.20), we have $\mathbb{I}_{i_0,0}^{\text{nil,loop } s} \neq \emptyset$. From Item $\mathbb{I}_{i_0,0}^{\text{nil,loop } s} = \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s}$ and $\mathbb{I}_{i_0,0}^{\text{nil,loop } s} \neq \emptyset$, we have $\mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} \neq \emptyset$. From $\mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.\text{stk} \leq K)$ and Lemma (3.6.25), we have type judgment (i) below for s is well formed. From the type judgment (i) below for s is well formed and Lemma (3.6.37), we have the type derivation for s :

$$(i) \quad \overline{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i_0,0}^{\sigma,s;\text{loop } s}} s : \bigwedge_{i \in \mathbb{I}_{i_0,0}^{\sigma,s;\text{loop } s}} (\text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;\text{loop } s}} \text{imr}_j, \overline{T}_j)$$

From $\mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} \neq \emptyset$ and $Q \in \mathcal{R} \Rightarrow Q.\text{stk} \leq K$ and Lemma (3.6.38), we have $\mathbb{K}_{i_0,0}^{\sigma,s;\text{loop } s} \leq K$. From $\mathbb{K}_{i_0,0}^{\sigma,s;\text{loop } s} \leq K$, (i) and Lemma (3.4.5), we have

$$(ii) \quad \overline{r}_{\mathcal{R}} \vdash_K s : \bigwedge_{i \in \mathbb{I}_{i_0,0}^{\sigma,s;\text{loop } s}} (\text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,s;\text{loop } s}} \text{imr}_j, \overline{T}_j)$$

From $\mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} \neq \emptyset$ and Lemma (3.6.19), we have $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} : \mathbb{J}_i^{\text{nil},s;\text{loop } s} \subseteq \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s}$. From $\mathbb{I}_{i_0,0}^{\text{nil,loop } s} = \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s}$ and $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} : \mathbb{J}_i^{\text{nil},s;\text{loop } s} \subseteq \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s}$, we have $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} : \mathbb{J}_i^{\text{nil},s;\text{loop } s} \subseteq \mathbb{I}_{i_0,0}^{\text{nil,loop } s}$. From (ii) and $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;\text{loop } s} : \mathbb{J}_i^{\text{nil},s;\text{loop } s} \subseteq \mathbb{I}_{i_0,0}^{\text{nil,loop } s}$ and Rule (3.25), we have

$$\overline{r}_{\mathcal{R}} \vdash_K \text{loop } s : \bigvee_{i \in \mathbb{I}_{i_0,0}^{\text{nil,loop } s}} \text{imr}_i, \overline{T}_i.$$

2. $m = s; m'$.

From Lemma (3.6.20), we have $\mathbb{I}_{i_0,0}^{\text{nil},s;m'} \neq \emptyset$. From $\mathbb{I}_{i_0,0}^{\text{nil},s;m'} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.\text{stk} \leq K)$ and Lemma (3.6.25), we have type judgment (i) below for s is well formed. From the type judgment (i) below for s is well formed and Lemma (3.6.37), we have the type derivation for s :

$$(i) \quad \overline{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i_0,0}^{\text{nil},s;m'}} s : \bigwedge_{i \in \mathbb{I}_{i_0,0}^{\text{nil},s;m'}} (\text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\text{nil},s;m'}} \text{imr}_j, \overline{T}_j)$$

From $\mathbb{I}_{i_0,0}^{\text{nil},s;m'} \neq \emptyset$ and $Q \in \mathcal{R} \Rightarrow Q.\text{stk} \leq K$ and Lemma (3.6.38), we have $\mathbb{K}_{i_0,0}^{\text{nil},s;m'} \leq K$. From $\mathbb{K}_{i_0,0}^{\text{nil},s;m'} \leq K$, (i) and Lemma (3.4.5), we have

$$(ii) \quad \overline{r}_{\mathcal{R}} \vdash_K s : \bigwedge_{i \in \mathbb{I}_{i_0,0}^{\text{nil},s;m'}} (\text{imr}_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\text{nil},s;m'}} \text{imr}_j, \overline{T}_j)$$

From $\mathbb{I}_{i_0,0}^{\text{nil},s;m'} \neq \emptyset$ and Lemma (3.6.19), we have $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;m'} : \mathbb{J}_i^{\text{nil},s;m'} \subseteq \mathbb{I}_{i_0,0}^{\text{nil},m'}$.

From the induction hypothesis, we have

$$(iii) \quad \bar{\tau}_{\mathcal{R}} \vdash_K m' : \bigvee_{i \in \mathbb{I}_{i_0,0}^{\text{nil},m'}} \text{imr}_i, \bar{T}_i$$

From (ii), (iii), $\forall i \in \mathbb{I}_{i_0,0}^{\text{nil},s;m'} : \mathbb{J}_i^{\text{nil},s;m'} \subseteq \mathbb{I}_{i_0,0}^{\text{nil},m'}$ and Rule (3.26), we have

$$\bar{\tau}_{\mathcal{R}} \vdash_K s; m' : \bigvee_{i \in \mathbb{I}_{i_0,0}^{\text{nil},s;m'}} \text{imr}_i, \bar{T}_i.$$

■

We finally prove the type derivation for handlers can be derived.

Lemma 3.6.40 (Type derivation for handler) *If $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{\tau}$ and the type judgment for h (3.61) is well formed, then we can derive*

$$\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,h}} h : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,h}} (\text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h}} \text{imr}_j, \bar{T}_j) \quad (3.61)$$

Proof From the type judgment for h (3.61) is well formed and Item (3) of Definition (3.4.2), we have $\mathbb{I}_{i',v}^{\sigma,h} \neq \emptyset$.

We prove by induction on h . There are two cases.

1. $h = \text{iret}$.

To prove $\bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,\text{iret}}} \text{iret} : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}} (\text{imr}_i, \bar{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,\text{iret}}} \text{imr}_j, \bar{T}_j)$, we need to prove the safety condition $\text{safe}(\bar{\tau}_{\mathcal{R}}, \mathbb{I}_{i',v}^{\sigma,\text{iret}}, \mathbb{K}_{i',v}^{\sigma,\text{iret}})$ which consists of the following two items:

$$(a) \quad (i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}} \wedge u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)) \Rightarrow \begin{cases} (i) & i \in \mathbb{I}^u \\ (ii) & \mathbb{J}_i^u \subseteq \mathbb{I}_{i',v}^{\sigma,\text{iret}} \\ (iii) & \mathbb{K}_i^u + 1 \leq \mathbb{K}_{i',v}^{\sigma,\text{iret}} \\ (iv) & \widehat{\theta}_u(\bar{T}_i) \leq \bar{\tau} \end{cases}$$

$$(b) \quad \forall i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}} : \widehat{\theta}(\text{imr}_i, \bar{T}_i) \leq \bar{\tau}$$

$$(c) \quad \forall i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}} : \mathbb{J}_i^{\sigma,\text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma,\text{iret}} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(\text{imr}_i, \bar{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_{\text{iret}}(i) \}$$

• Proof of Item (a).

– Proof of (i). From $i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}$ and Item (1) of Proposition (3.6.8), we have $\langle \text{imr}_i, \bar{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$. From $\langle \text{imr}_i, \bar{T}_i, \sigma, \text{iret} \rangle \in \mathcal{R}$ and Item (1) of Definition (3.6.8), we have $i \in \mathbb{I}^{\sigma,\text{iret}}$. From $i \in \mathbb{I}_{i',v}^{\sigma,\text{iret}}$, $\eta(\text{iret}) \in \mathcal{L}$, $u \in$

$\widehat{\mathcal{P}}(imr_i, \widehat{T}_i)$ and Item (3) of Lemma (3.6.32), we have $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$.

From $i \in \mathbb{I}^{\sigma, \text{iret}}$, $\zeta_u(i) \in \mathbb{I}^{\text{iret}::\sigma, \bar{h}(u)}$ and Item (1) of Definition (3.6.9), we have $i \in \mathbb{I}^u$.

– Proof of Item (ii). From $\eta(\text{iret}) \in \mathcal{L}$, $i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}}$, $u \in \widehat{\mathcal{P}}(imr_i, \widehat{T}_i)$ and Lemma (3.6.28), we have $\mathbb{J}_i^u \subseteq \mathbb{I}_{i',v}^{\sigma, \text{iret}}$.

– Proof of Item (iii). From $i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}}$ and Item (2) of Lemma (3.6.34), we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma, \text{iret}}$. From $i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}}$ and Lemma (3.6.14), we have $\mathbb{K}_i^{\sigma, \text{iret}} \leq \mathbb{K}_{i',v}^{\sigma, \text{iret}}$. From $\mathbb{K}_i^u + 1 \leq \mathbb{K}_i^{\sigma, \text{iret}}$ and $\mathbb{K}_i^{\sigma, \text{iret}} \leq \mathbb{K}_{i',v}^{\sigma, \text{iret}}$, we have $\mathbb{K}_i^u + 1 \leq \mathbb{K}_{i',v}^{\sigma, \text{iret}}$.

– Proof of Item (iv). From $i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}}$ and $u \in \widehat{\mathcal{P}}(imr_i, \widehat{T}_i)$ and $Q \in \mathcal{R} \Rightarrow Q.\widehat{T} \leq \bar{r}$ and Item (2) of Lemma (3.6.36), we have $\widehat{\theta}_u(\widehat{T}_i) \leq \bar{r}$.

- Proof of Item (b).

From $i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}}$ and $Q \in \mathcal{R} \Rightarrow Q.\widehat{T} \leq \bar{r}$ and Item (1) of Lemma (3.6.36), we have $\widehat{\theta}(imr_i, \widehat{T}_i) \leq \bar{r}$.

- Proof of Item (c).

From $\mathbb{I}_{i',v}^{\sigma, \text{iret}} \neq \emptyset$ and Item (1) of Lemma (3.6.34), we have $\forall i \in \mathbb{I}_{i',v}^{\sigma, \text{iret}} : \mathbb{J}_i^{\sigma, \text{iret}} = \bigcup \{ \mathbb{J}_j^{\sigma, \text{iret}} \mid j \in \bigcup_{u \in \widehat{\mathcal{P}}(imr_i, \widehat{T}_i)} \mathbb{J}_i^u \} \cup \{ \xi_s(i) \}$.

2. $h = s; h'$.

From $h = s; h'$, we have $\mathbb{I}_{i',v}^{\sigma, h} = \mathbb{I}_{i',v}^{\sigma, s; h'}$. From $\mathbb{I}_{i',v}^{\sigma, h} \neq \emptyset$ and $\mathbb{I}_{i',v}^{\sigma, h} = \mathbb{I}_{i',v}^{\sigma, s; h'}$, we have $\mathbb{I}_{i',v}^{\sigma, s; h'} \neq \emptyset$. From $\mathbb{I}_{i',v}^{\sigma, s; h'} \neq \emptyset$, $Q \in \mathcal{R} \Rightarrow (Q.\widehat{T} \leq \bar{r} \wedge Q.stk \leq K)$ and Lemma (3.6.25), we have type judgment (i) below for s is well formed. From the type judgment (i) below for s is well formed and Lemma (3.6.37), we have the type derivation for s :

$$(i) \quad \bar{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma, s; h'}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma, s; h'}} (imr_i, \widehat{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma, s; h'}} imr_j, \widehat{T}_j)$$

From $\mathbb{I}_{i',v}^{\sigma, h} \neq \emptyset$, $h = s; h'$ and Item (3) of Lemma (3.6.31), we have $\mathbb{K}_{i',v}^{\sigma, s; h'} \leq \mathbb{K}_{i',v}^{\sigma, h}$ and $\mathbb{K}_{i',v}^{\sigma, h'} \leq \mathbb{K}_{i',v}^{\sigma, h}$. From $\mathbb{K}_{i',v}^{\sigma, s; h'} \leq \mathbb{K}_{i',v}^{\sigma, h}$, (i) and Lemma (3.4.5), we have

$$(ii) \quad \bar{r}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma, h}} s : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma, s; h'}} (imr_i, \widehat{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma, s; h'}} imr_j, \widehat{T}_j)$$

From $\mathbb{I}_{i',v}^{\sigma, s; h'} \neq \emptyset$ and Corollary (3.6.3), we have $\mathbb{I}_{i',v}^{\sigma, h'} \neq \emptyset$. From $\mathbb{I}_{i',v}^{\sigma, h'} \neq \emptyset$,

$Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K)$ and Lemma (3.6.26), we have the judgment for h' is well formed. From the judgment for h' is well formed and the induction hypothesis, we have

$$(iii) \quad \overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,h'}} h' : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,h'}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h'}} imr_j, \overline{T}_j)$$

From $\mathbb{K}_{i',v}^{\sigma,h'} \leq \mathbb{K}_{i',v}^{\sigma,h}$, (iii) and Lemma (3.4.5), we have

$$(iv) \quad \overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,h}} h' : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,h'}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h'}} imr_j, \overline{T}_j)$$

From $h = s; h'$ and Item (3) of Lemma (3.6.30), we have

$$(v) \quad \mathbb{J}_i^{\sigma,h} = \bigcup \{ \mathbb{J}_j^{\sigma,h'} \mid j \in \mathbb{J}_i^{\sigma,s;h'} \}$$

From (ii), (iv), $\forall i \in \mathbb{I}_{i',v}^{\sigma,s;h'} : \mathbb{J}_i^{\sigma,s;h'} \subseteq \mathbb{I}_{i',v}^{\sigma,h'}$ (from Lemma (3.6.19)), (v) and

Rule (3.27), we have

$$\overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{i',v}^{\sigma,h}} h : \bigwedge_{i \in \mathbb{I}_{i',v}^{\sigma,h}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{\sigma,h}} imr_j, \overline{T}_j)$$

■

We are ready to prove the Theorem that states if the abstract model checks, then there exist types for the periodic interrupt calculus such that the program type checks.

Lemma 3.6.41 *If $Q \in \mathcal{R} \Rightarrow Q.\overline{T} \leq \overline{r}$, then $\overline{\tau}_{\mathcal{R}} \vdash \overline{h} : \overline{\tau}_{\mathcal{R}}$.*

Proof Let $u \in 1..N$. From Corollary (3.6.4), we have $\mathbb{I}^u \neq \emptyset$. Consider $i \in \mathbb{I}^u$. From Item (1) and (2) of Definition (3.6.9) and $\mathbb{I}^u \neq \emptyset$, we have $\zeta_u(i) \in \mathbb{I}^{a::\sigma,\overline{h}(u)}$, where $a :: \sigma \in \mathbb{W}_i^u$. From $\zeta_u(i) \in \mathbb{I}^{a::\sigma,\overline{h}(u)}$ and Proposition (3.6.9), we have $\mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)} = \{\zeta_u(i)\}$. From $\mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)} = \{\zeta_u(i)\}$, $Q \in \mathcal{R} \Rightarrow (Q.\overline{T} \leq \overline{r} \wedge Q.stk \leq K)$ and Lemma (3.6.26), we have $\overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} \overline{h}(u) : \bigwedge_{i \in \mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} (imr_i, \overline{T}_i \rightarrow \bigvee_{j \in \mathbb{J}_i^{a::\sigma,\overline{h}(u)}} imr_j, \overline{T}_j)$ is well formed. From $Q \in \mathcal{R} \Rightarrow Q.\overline{T} \leq \overline{r}$, $\overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} \overline{h}(u) : \bigwedge_{i' \in \mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} (imr_{i'}, \overline{T}_{i'} \rightarrow \bigvee_{j \in \mathbb{J}_{i'}^{a::\sigma,\overline{h}(u)}} imr_j, \overline{T}_j)$ is well formed and Lemma (3.6.40), we have the derivation

$$(i) \quad \overline{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} \overline{h}(u) : \bigwedge_{i' \in \mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}} (imr_{i'}, \overline{T}_{i'} \rightarrow \bigvee_{j \in \mathbb{J}_{i'}^{a::\sigma,\overline{h}(u)}} imr_j, \overline{T}_j)$$

From $\mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)} = \{\zeta_u(i)\}$ is a singleton set, and Item (5) of Definition (3.6.8), we have $\mathbb{K}_{\zeta_u(i)}^{a::\sigma,\overline{h}(u)} = \mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$. From $i \in \mathbb{I}^u$, $a :: \sigma \in \mathbb{W}_i^u$ and Item (4) of Definition (3.6.9), we have $\mathbb{K}_i^u = \mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$. From $\mathbb{K}_{\zeta_u(i)}^{a::\sigma,\overline{h}(u)} = \mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$ and $\mathbb{K}_i^u = \mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$, we have $\mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)} = \mathbb{K}_i^u$. Replacing $\mathbb{K}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$ in (i) with \mathbb{K}_i^u and $\mathbb{I}_{\zeta_u(i),u}^{a::\sigma,\overline{h}(u)}$ in (i) with $\{\zeta_u(i)\}$, we

have

$$(ii) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_i^u} \bar{h}(u) : imr_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)} \rightarrow \bigvee_{j \in \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}} imr_j, \bar{T}_j$$

From $i \in \mathbb{I}^u$, $a :: \sigma \in \mathbb{W}_i^u$ and Item (3) of Definition (3.6.9), we have $\mathbb{J}_i^u = \mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$.

Replacing $\mathbb{J}_{\zeta_u(i)}^{a::\sigma, \bar{h}(u)}$ in (ii) with \mathbb{J}_i^u , we have

$$(ii) \quad \bar{\tau}_{\mathcal{R}} \vdash_{\mathbb{K}_i^u} \bar{h}(u) : imr_{\zeta_u(i)}, \bar{T}_{\zeta_u(i)} \rightarrow \bigvee_{j \in \mathbb{J}_i^u} imr_j, \bar{T}_j$$

From (ii) and Rule (3.24), we have $\bar{\tau}_{\mathcal{R}} \vdash \bar{h}(u) : \bar{\tau}_{\mathcal{R}}(u)$. ■

Theorem 3.6.1 *If $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$, then there exists $\bar{\tau}_{\mathcal{R}}$ such that $\bar{\tau}_{\mathcal{R}} \vdash_K p$.*

Proof From $Q \in \mathcal{R} \Rightarrow (Q.\bar{T} \leq \bar{r} \wedge Q.stk \leq K)$ and Lemma (3.6.39), we have derivation

$$(i) \quad \bar{\tau}_{\mathcal{R}} \vdash_K p.m : \bigvee_{i \in \mathbb{I}_{i_0,0}^{nil,p,m}} imr_i, \bar{T}_i$$

From $\mathbb{I}_{i_0,0}^{nil,p,m} = \{i_0\}$ and $(imr_{i_0}, \bar{T}_{i_0}) = (0, \bar{0})$ and (i), we have

$$(ii) \quad \bar{\tau}_{\mathcal{R}} \vdash_K p.m : 0, \bar{0}$$

From $Q \in \mathcal{R} \Rightarrow Q.\bar{T} \leq \bar{r}$ and Lemma (3.6.41), we have derivation

$$(iii) \quad \bar{\tau}_{\mathcal{R}} \vdash \bar{h} : \bar{\tau}_{\mathcal{R}}$$

From (ii), (iii) and Rule (3.19), we have $\bar{\tau}_{\mathcal{R}} \vdash_K p$. ■

3.7 Equivalence relation

In the light of model checking, our abstract semantics is a *model (an abstract interpretation)* of the concrete periodic interrupt program. Furthermore, Theorem 3.4.1 states that the execution of the model is a type-preserving process; while Theorem 3.6.1 states that the types of the program can be based on the information derived from the process of model checking/abstract interpretation.

Therefore, Theorem 3.4.1 and Theorem 3.6.1 combined together reveal an important equivalence relation between the model checking and the type checking presented in this chapter, which we formally state as follows:

Theorem 3.7.1 (Equivalence) *Given a number $K > 0$ and program p . We have $(Q_p \hookrightarrow^* Q \Rightarrow Q.stk \leq K)$ if and only if there exists $\bar{\tau}$ and that $\bar{\tau} \vdash_K p$.*

Proof Combine Theorem 3.4.1 and Theorem 3.6.1. ■

3.7.1 Model checking vs. type checking

Software model checking and program type checking are the two major tools which are both used to ensure that the software (1) observes desired properties, and (2) is free of bugs. Each has its advantages and disadvantages.

Model checking has a number advantages. It does not usually require the user to input annotations along with models, while type systems need type information from the program. This property makes model checking more appealing than the type system approach. Model checking is generally less strict than type checking, while a type system approach has to encode the property into the types and make sure that the type preservation works correctly for the program code. This lends model checking more flexibility and power in its real-world applications, such as being capable of dealing with more programs, leading to encompassing larger problem/program space. In contrast, type systems tend to end up with fewer programs being type checked.

However, model checking suffers from some disadvantages. Model checking is generally done on the entire model rather than in a modular fashion. If some part of a program is modified, then one has to model check the whole program again to make sure that some property does not break or that bugs are not introduced with the modification [8,9]. This repeatedly running model checker for the sake of a small change of the program is often unacceptable in the real world. However, type checking is modular and can be done on just a slice of code. Therefore, type checking can achieve much more efficiency.

In addition, such model checking would offer little help with the maintenance and further development of those systems because model checking, in general, does not offer more insight into bugs than giving a counter-example on the model (an

error trace). Moreover, the bugs found by the model checker are sometimes spurious ones caused by the model abstraction [49]. While model checking is able to produce counter-examples of a bug, it is well-known that pinpointing where the program point is that causes the bug is very difficult to do [4, 49]. Type systems, on the other hand, can capture the bugs “on the spot” and provide sufficient information to programmers or designers about the nature of bugs. This should be a tremendous benefit to programmers and testers in helping them understand the program code during debugging.

An equivalence relation between the model checking and the type checking, such as the one shown in Theorem 3.7.1, allows us to further and better understand programs. Furthermore, it would allow us to choose a better approach whenever we face a problem and have both tools, the type system and the model checking, available together.

3.8 Related work

Our work is based on the research done by Naik and Palsberg [40]. Both of works are concerned with the timing property of periodical interrupt-driven programs which are based on Palsberg and Ma’s interrupt calculus [44]. We further develop a version of interrupt calculus by Naik and Palsberg which leads to differences.

1. Their work is solely concerned with interrupt deadline safety within each interrupt device’s period. In contrast, we consider the bounded stack size problem as well as interrupt latency. More precisely, we consider the problem of bounding stack size that is impacted by the constraint of meeting interrupt latencies.
2. There are differences in the semantics. (a) We allow *imr* OR \vee operations while theirs does not. This difference is critical because allowing OR operations on *imr* allows for more interrupt handlers to be called. The analysis is consequently more complex. (b) We consider latencies rather than deadlines (see Item 3). (c) As a consequence of (b), we have a special latency transfer function for the

interrupt rule, rather than for the ired rule as shown in their work. (d) We have a completely different abstract transfer function than their work to ensure that the abstract semantics is an approximation of the concrete semantics.

3. We consider handler latencies rather than deadlines. The deadline of a handler is the time interval between an interrupt arrival and the end of handling it. The latency of a handler is the time interval between an interrupt arrival and the start of handling it. Naik and Palsberg make the restriction that the deadlines of a handler should be shorter than its periods. This restriction leads to the fact that a handler will always finish its handling before the next interrupt arrival. In contrast, our latency analysis does not impose such a restriction, yet it still ensures that all interrupts are handled in a timely manner and that there can be no interrupt which is missed. It is in this sense that our latency analysis is more relaxed than their deadline analysis.
4. Their type system gives types, and those types contain all the contexts of that term may enter. In contrast, our types for each term makes sense only for the process of type checking a handler within a particular context in which the handler is called. This is because for each calling context of a handler, the type judgment for the handler contains the stack size budget. For this reason, our type construction mainly involves a delimited version of reachability analysis.

There is a large body of work related to ours, among which model checking and type systems are of particular interest.

3.8.1 Model checking

Brylow and Palsberg [9] study the deadline problem of interrupt-driven software. They build a control flow graph out of an interrupt-driven program, with each node containing an address, an imr value and a stack. Their method identifies program loops in the graph. In particular, they differentiate (1) an unbounded loop; (2) the

loop that depends on external input; (3) fake loops caused by graph abstraction; (4) a race condition between interrupt handlers. They then run a reachability algorithm to ensure that all computation paths meet their deadline. In essence, their method is a variant of model checking, in which the model is the graph abstracted away from the program by using address, imr, and stack information. Our work is similar to theirs because both consider the deadline analysis problem and both works involve an abstract model. However, our work is a type based approach. The type system we give not only ensures whether or not the deadlines of handling can be met, but also ensures bounded stack size. Furthermore, the type checking process is modular, which makes it more appealing in real-world applications than model checking. More importantly, our work unveils an equivalence relation between model checking and type checking. It shows us what kind of type system there might be that is equivalent to model checking. The equivalence relation allows us to choose an appropriate tool, between model checking and type checking, when facing an application. Therefore, the revelation of the relation itself may shed more light in the fields of software engineering and analysis than the type system itself.

Basu, Kumar, Pokorny and Ramakrishan [6] develop a resource-constrained model checking technique that differentiates stack-bounded runs of programs from stack-diverging runs of programs for abstractions of recursive programs (push-down systems). Both their work and ours can predict whether program execution requires unbounded stack size or not. Despite the surface similarity, their work and ours are different in a number of significant ways. First, we consider the problem of possible unbounded stack size that arises in a very different situation, namely, interrupt handling from their work, Second, we consider timing issues (latencies) for each handler. If the latency can not be met, then the interrupt program is considered to be incorrect. This constraint enables us to have more quality control over the software than theirs. Third, resource constrained model checking is implemented on the push-down system and Büchi automaton; in contrast, we employ a simple abstraction to build the model. Finally, their model checking does not bound the stack size *a priori*,

rather, it takes into account all the runs as long as they have finite stack size usage; while we give *a priori* stack size K and check against the K .

There is also a large body of work dedicated to the problem of how to automatically abstract models from program code and how to solve the state explosion and expressivity problem for software model checking [3, 11, 14, 20, 27]. Although our main focus is on what model checking process can be encoded in/equivalent to a type system and what such a type system should look like, it is worth pointing out that our abstract model of the periodic calculus is aligned with all the other work which has been done on this issue because the calculus is a straightforward abstraction of Zilog 80 assembly code. This fact makes it relatively effortless to construct the model and types directly from the assembly source code.

3.8.2 Type systems

We use intersection type [51, 52] to show the equivalence relation between type systems and model checking. Intersection types have been studied extensively in many contexts. Some recent developments on intersection types show that it has increasingly been of interest in the programming languages community. Mossin [38] uses intersection types to achieve precise redex prediction of type-based flow analysis. Palsberg and Pavlopoulou [45] establish a equivalence correspondence between polyvariant flow information and intersection and union types. Davies and Pfenning [16] study the intersection types in the presence of computational effects (mutable references, exceptions, etc.) and give a sound type system for a language with mutable references.

Our treatment of intersection types is a form of finite polymorphism with a flavor of value-sensitivity (imr values in our case). It differs from the bounded polymorphism [51, 52] in that our use of intersection types is value-sensitive, (values of imrs) rather than bounded quantification (not bounded imrs).

There is some work that uses type systems to bound the run-time space usage of programs in different contexts. Xi and Pfenning [61–63] show that dependent types over constraint domains (integer domain, etc.) can be used to eliminate the runtime array bound checking in practical programming. The application of dependent types shows a resemblance to ours in terms of commanding the bounded memory space of program execution. However, the differences are substantial. The dependent types are mainly concerned with the size of data types indexed over a certain domain while we focus on the stack size growth that arises from the handler calls in the strict context of interrupt-driven systems. Neither can dependent types meet our needs nor our type system cover array bound elimination.

Hughes, Pareto and Sabry [29, 31, 47] propose sized types to ensure bounded size of recursion and data structure that is potentially unbounded at runtime. Their approach can be viewed as a restricted application of dependent types that aims only at the recursive datatypes with recursion in mind. Their work was done as part of moving functional programming (ML) into the field of embedded/reactive systems. However, their work cannot be applied to the context of interrupt handling.

Our treatment of parametric polymorphism is similar, in spirit, to Agesen’s Cartesian product algorithm [2]. The similarity comes from the fact that we combine an *imr* value (*imr*) and a stack growth value (δ) into a context and intersect together all such contexts as the type for each handler and that, for each context, we treat it as a mono-morphic combination of *imr* and δ . Doing it in this way allows us to have better and more accurate control of the contexts in which the handlers are called. The value-sensitivity of our type systems is mainly a result of this choice.

4 CONCLUSION

The goal of this thesis has been to bound the stack size of interrupt-driven real-time embedded systems. The bounded stack problem is especially crucial to mission-critical real-time applications because a stack overflow will cause a system crash, which may result in a disaster or even the loss of human life. On the other hand, predicting a safe and tight upper of memory usage should be of great interest to system manufacturers because they can put as little hardware as would be possible into the chips without the fear of causing any software problems, thus reducing manufacturing costs and increasing profit margins.

Looking at things from a broader perspective, the techniques used in this thesis demonstrate a useful strategy for resource-constrained compilation. Our type judgments contain a natural number K which is capable of formulating the space requirement for a handler. We believe this technique is useful in its own right, and can be naturally extended to other applications.

Our designs of the interrupt calculus and periodic interrupt calculus are the first theoretical attempt to employ formal language techniques to analyze resource (space) boundedness problems in real-time, embedded, interrupt-driven systems.

Recent developments in type related research [36, 37, 43] have vastly and rapidly pushed its frontier beyond its original use in programming language design. As one of many in this trend, the interrupt calculus and its periodic cousin can be viewed as testing ground for future experiments.

While model checking, in general, can be used to solve the problem, we believe that type systems provide more desirable flexibility because of their capability for doing modular type checking. For example, modern integrated programming development environments (IDE) usually incorporate a sub-system which collects type information

on demand from the source code; even when the source code is not completely written. The system attempts to type check the partially complete code. If it finds a definite error, it will then alert the programmer of the error “on the spot”. In an ideal world, interrupt-driven programs should also be developed in such an environment. The compiler should be able to collect the type information in order to predict whether periods constraints are satisfied or not. This kind of facility will greatly increase the quality of the real-time embedded systems by locating the bugs and by predicting whether software specification could be met at compile time.

Our type systems for interrupt-driven programs can be characterized as context-sensitive (interrupt calculus and periodic interrupt calculus) and value-sensitive (periodic interrupt calculus). The value-sensitive nature of the type system lends us a helping hand in gaining some insight into how type systems can facilitate resource-aware compilation, that is, whether and how quantified resources – either space, time, energy/battery requirements or any other constraint – can be readily and soundly typed.

Types, in general, also serve as convenient program documentations and specifications. This is especially useful when maintaining programs and updating programs because any modification to the program should observe the constraints enforced by types. For example, the type system of the periodic interrupt calculus guarantees the stack size boundedness and also maintains the period constraints at the same time. If one would like to further modify the code, then the modification part should pass the type checking. Doing so will also greatly reduce testing time, therefore shortening the software development cycle.

LIST OF REFERENCES

LIST OF REFERENCES

- [1] Martín Abadi and Luca Cardelli. *A Theory of Objects*. Springer-Verlag, 1996.
- [2] Ole Agesen. The Cartesian product algorithm. In *Proceedings of ECOOP'95, Seventh European Conference on Object-Oriented Programming*, pages 2–26. Springer-Verlag (LNCS 952), 1995.
- [3] Thomas Ball, Rupak Majumdar, Todd Millstein, and Sriram Rajamani. Automatic predicate abstraction of C programs. In *Proceedings of PLDI'01, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 203–213, 2001.
- [4] Thomas Ball, Mayur Naik, and Sriram Rajamani. From symptom to cause: Localizing errors in counterexample traces. In *Conference Record of the 30th Annual ACM Symposium on Principles of Programming Languages*, 2003.
- [5] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, Amsterdam, 1981.
- [6] Samik Basu, K. Narayan Kumar, L. Robert Pokorny, and C. R. Ramakrishnan. Resource-constrained model checking of recursive programs. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 236–250, 2002.
- [7] Gerard Berry and Georges Gonthier. The esternel synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.
- [8] Dennis Brylow, Niels Damgaard, and Jens Palsberg. Static checking of interrupt-driven software. In *Proceedings of ICSE'01, 23rd International Conference on Software Engineering*, pages 47–56, Toronto, May 2001.
- [9] 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*, pages 198–207, Helsinki, Finland, September 2003.
- [10] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of Foundations of Software Science and Computation Structures*, pages 140–155. Springer-Verlag (LNCS 1378), Berlin, Germany, 1998.
- [11] 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.

- [12] 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.
- [13] 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.
- [14] James C. Corbett, Matthew B. Dwyer, John Hatcliff, Shawn Laubach, Corina S. Pasareanu, Robby, and Hongjun Zheng. Bandera : Extracting finite-state models from Java source code. In *Proceedings of ICSE'00, 22nd International Conference on Software Engineering*, pages 439–448, 2000.
- [15] Matteo Corti, Roberto Brega, and Thomas Gross. Approximation of worst-case execution time for preemptive multitasking systems. In *Proceedings of LCTES'00, Languages, Compilers, and Tools for Embedded Systems*, 2000.
- [16] Rowan Davies and Frank Pfenning. Intersection types and computational effects. *ACM SIGPLAN Notices*, 35(9):198–208, 2000.
- [17] Giorgio Delzanno and Andreas Podelski. Constraint-based deductive model checking. *International Journal on Software Tools for Technology Transfer*, 3(3):250–270, 2001.
- [18] Matthew B. Dwyer, John Hatcliff, Roby Joehanes, Shawn Laubach, Corina S. Pasareanu, Robby, Hongjun Zheng, and W Visser. Tool-supported program abstraction for finite-state verification. In *International Conference on Software Engineering*, pages 177–187, 2001.
- [19] Jakob Engblom and Andreas Ermedahl. Modeling complex flows for worst-case execution time analysis. In *Proceedings of RTSS'00, 21st IEEE Real-Time Systems Symposium*, 2000.
- [20] Cormac Flanagan. Automatic software model checking via clp. In *ESOP*, 2003.
- [21] Alexandre Frey. Satisfying systems of subtype inequalities in polynomial space. In *Proceedings of SAS'97, International Static Analysis Symposium*. Springer-Verlag (LNCS), 1997.
- [22] Thomas A. Henzinger, Benjamin Horowitz, and Christoph Meyer Kirsch. Embedded control systems development with giotto. In *LCTES/OM*, pages 64–72, 2001.
- [23] Thomas A. Henzinger, Benjamin Horowitz, and Christoph Meyer Kirsch. Giotto: A time-triggered language for embedded programming. In *Proceedings of EM-SOFT 2001*. Springer-Verlag (LNCS 2211), 2001.
- [24] Thomas A. Henzinger and Christoph M. Kirsch. The embedded machine: Predictable, portable real-time code. In *Proceedings of PLDI'02, ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 315–326, 2002.

- [25] Thomas A. Henzinger, Christoph M. Kirsch, Rupak Majumdar, and Slobodan Matic. Time-safety checking for embedded programs. In *Proceedings of EM-SOFT'02, Second International Workshop on Embedded Software*, pages 76–92. Springer-Verlag (LNCS 2491), 2002.
- [26] J. Roger Hindley. Types with intersection: An introduction. *Formal Aspects of Computing*, 4:470–486, 1991.
- [27] G. J. Holzmann and M. H. Smith. Software model checking: extracting verification models from source code. In *Proceedings of FORTE/PSTV'99*, November 1999.
- [28] Paul Hudak. *The Haskell School of Expression—Learning Functional Programming through Multimedia*. Cambridge University Press, 2000.
- [29] John Hughes and Lars Pareto. Recursion and dynamic data-structures in bounded space: Towards embedded ML programming. In *International Conference on Functional Programming*, pages 70–81, 1999.
- [30] John Hughes, Lars Pareto, and Amr Sabry. Proving the correctness of reactive systems using sized types. In *Proceedings of POPL'96, 23rd Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 410–423, 1996.
- [31] John Hughes, Lars Pareto, and Amr Sabry. Proving the correctness of reactive systems using sized types. In *Symposium on Principles of Programming Languages*, pages 410–423, 1996.
- [32] A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. An analysis of ML typability. *Journal of the ACM*, 41(2):368–398, 1994.
- [33] Harry G. Mairson. Decidability of ML typing is complete for deterministic exponential time. In *Seventeenth Symposium on Principles of Programming Languages*, pages 382–401, 1990.
- [34] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag (LNCS 92), 1980.
- [35] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part I/II. *Information and Computation*, 100(1):1–77, September 1992.
- [36] Greg Morrisett, Karl Crary, Neal Glew, and David Walker. Stack-based typed assembly language. In *ACM Workshop on Types in Compilation*, pages 95–118, Kyoto, Japan, March 1998.
- [37] Greg Morrisett, David Walker, Karl Crary, and Neal Glew. From system F to typed assembly language. In *Proceedings of POPL'98, 25th Annual SIGPLAN–SIGACT Symposium on Principles of Programming Languages*, pages 85–97, 1998.
- [38] Christian Mossin. Exact flow analysis. In *Fourth International Static Analysis Symposium (SAS)*, pages 250–264, Paris, France, 1997. Springer-Verlag.

- [39] 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.
- [40] Mayur Naik and Jens Palsberg. A type system equivalent to a model checker. Manuscript, 2003.
- [41] Peter G. Neumann. Risks to the public in computers and related systems. *SIGSOFT Softw. Eng. Notes*, 20(3):7–12, 1995.
- [42] Greger Ottosson and Mikael Sjödín. Worst-case execution time analysis for modern hardware architectures. In *ACM SIGPLAN 1997 Workshop on Languages, Compilers, and Tools for Real-Time Systems (LCT-RTS'97)*, 1997.
- [43] Jens Palsberg. Type-based analysis and applications. In *Proceedings of PASTE'01, ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools*, pages 20–27, Snowbird, Utah, June 2001. Invited paper.
- [44] 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.
- [45] Jens Palsberg and Christina Pavlopoulou. From polyvariant flow information to intersection and union types. In *Conference Record of POPL 98: The 25TH ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, California*, pages 197–208, 1998.
- [46] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [47] Lars Pareto. Sized types. M.S. Thesis, Chalmers University of Technology, 1996.
- [48] Lars Pareto. *Types for Crash Prevention*. PhD thesis, Chalmers University of Technology, 2000.
- [49] Corina S. Pasareanu, Matthew B. Dwyer, and Willem Visser. Finding feasible counter-examples when model checking abstracted Java programs. *Lecture Notes in Computer Science*, 2031, 2001.
- [50] Stefan Petters and Georg Färber. Making worst case execution time analysis for hard real-time tasks on state of the art processors feasible. In *Proceedings of 6th International Conference on Real-Time Computing Systems and Applications*, 1999.
- [51] Benjamin C. Pierce. *Programming with Intersection Types and Bounded Polymorphism*. PhD thesis, 1991.
- [52] Benjamin C. Pierce. Intersection types and bounded polymorphism. In M. Bezem and J. F. Groote, editors, *Proc. of 1st Int. Conf. on Typed Lambda Calculi and Applications, TLCA'93, Utrecht, The Netherlands, 16–18 March 1993*, volume 664, pages 346–360. Springer-Verlag, Berlin, 1993.

- [53] Andreas Podelski. Model checking as constraint solving. In *Static Analysis Symposium*, pages 22–37, 2000.
- [54] Andreas Podelski. Model checking as constraint solving. In *Proceedings of SAS'00, International Static Analysis Symposium*, pages 22–37. Springer-Verlag (LNCS 1824), 2000.
- [55] T. Reps, S. Horwitz, and M. Sagiv. Precise interprocedural dataflow analysis via graph reachability. In *POPL 95: Principles of Programming Languages*, pages 49–61. ACM, 1995.
- [56] Thomas Reps. Program analysis via graph reachability. *Information and Software Technology*, 40(11–12):701–726, November 1998. <http://www.cs.wisc.edu/wpis/papers/tr1386.ps>.
- [57] Jerzy Tiuryn. Subtype inequalities. In *LICS'92, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 308–315, 1992.
- [58] Zhanyong Wan, Walid Taha, and Paul Hudak. Real-time FRP. In *Proceedings of ICFP'01, ACM SIGPLAN International Conference on Functional Programming*, pages 146–156, 2001.
- [59] Zhanyong Wan, Walid Taha, and Paul Hudak. Event-driven FRP. In *Proceedings of PADL'02, Practical Aspects of Declarative Languages*, pages 155–172, 2002.
- [60] Wayne Wolf. *Computers as Components, Principles of Embedded Computing System Design*. Morgan Kaufman Publishers, 2000.
- [61] Hongwei Xi. Dead code elimination through dependent types. *Lecture Notes in Computer Science*, 1551:228–242, 1999.
- [62] Hongwei Xi and Frank Pfenning. Eliminating array bound checking through dependent types. In *SIGPLAN Conference on Programming Language Design and Implementation*, pages 249–257, 1998.
- [63] Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *Conference Record of POPL 99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, Texas*, pages 214–227, New York, NY, 1999.
- [64] Tian Zhao, Di Ma, and Jens Palsberg. Stack-size analysis of interrupt-driven software. Manuscript, 2002.

VITA

VITA

Di Ma was born in Beijing, P.R. China, in 1972. He received his bachelor's degree in Computer Science and Engineering from National University of Defense and Technology (formerly known as Changsha Institute of Technology) in June 1995. After working for three years in a couple of software companies, he came to Purdue University to study computer science in Fall 1998. In May 2000, he received his master's degree in computer science, after which he continued to pursue a Ph.D. degree, which he received in August 2004.