Naval Research Laboratory
Washington, DC 20375-5320

NRL/5540/MR—2021/3

Logical Analysis of One Formalization of Exploitation

DR. GERARD ALLWEIN

Center for High Assurance Computer Systems Branch
Information Technology Division

August 30, 2021

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
30-08-2021	NRL Memorandum Report	1 Oct 2020 – 30 Sept 2021

4. TITLE AND SUBTITLE

Logical Analysis of One Formalization of Exploitation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
062235N

6. AUTHOR(S)

Dr. Gerard Allwein

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
6B23

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

IR-5543-21-15-U

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
875 N. Randolph Street
Arlington VA 22217-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report is a logical analysis of a formalization of an exploit called code reuse. There are instances of this exploit being used for attacks on existing systems. The basic scenario is to look at an application as consisting of blocks of code that can be strung together differently than the usual flow of control in the application. This is usually effected via altering the return addresses on the program stack. Looked upon in this way, an application consists of a collection of high level instructions, one per code block. The attacker can then string together these code blocks to suit his/her own purposes. The formalization, while not without its faults, represents an insightful method for understanding code reuse exploits. The analysis is performed using Distributed Logic.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Gerard Allwein
UU	UU	UU	UU	24	**19b. TELEPHONE NUMBER** *(include area code)* (202) 404-3748

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank.

CONTENTS

This page intentionally left blank

EXECUTIVE SUMMARY

This report is a logical analysis of a formalization of an exploit called *code reuse*. There are instances of this exploit being used for attacks on existing systems. The basic scenario is to look at an application as consisting of blocks of code that can be strung together differently than the usual flow of control in the application. This is usually effected via altering the return addresses on the program stack. Looked upon in this way, an application consists of a collection of high level instructions, one per code block. The attacker can then string together these code blocks to suit his/her own purposes.

The analysis is done in terms of Distributed Logic. Another NRL Memorandum Report, IR-5543-21-14-U explains some of the details of Distributed Logic. The PI also has published papers describing Distributed Logic in the bibliography.

This page intentionally left blank

LOGICAL ANALYSIS OF ONE FORMALIZATION OF EXPLOITATION

1. INTRODUCTION

There are many kinds of exploits available to an attacker for software applications. Code reuse is one such kind of exploit. Upon reading the paper [1], it begs the question of can we get a higher level understanding of the exploit. The PI has developed Distributed Logic [2, 3] for reasoning about distributed systems from the perspective of a collection of localities connected with some mathematical operators and relations, and something running at each locality in parallel with the things running at the other localities in a distributed system. The things running need not be computer programs but can anything describable in a modal logic. Distributed Logic shows how to connect them without getting outside the bounds of a formal logic, i.e., relying strictly upon mathematical models. The models for Distributed Logic constrain the kinds of mathematical models used in analyzing a system. As such, they present one set of guardrails to guide an analysis. The models can be quite general even as they constrain the class of models.

The basic scenario of code reuse is to look at an application as consisting of blocks of code that can be strung together differently than the usual flow of control in the application. This is usually effected via altering the return addresses on the program stack. Looked upon in this way, an application consists of a collection of high level instructions, one per code block. The attacker can then use these code blocks to suit his/her own purposes.

2. EXPLOITATION AS CODE REUSE

In the main example of [1], the *violated* or *broken* abstraction is some higher level programmer's abstraction of how the code ought to operate. The *obeyed* abstraction is a lower level machine code abstraction. This one is obeyed because the machine never reaches an invalid state. Invalid states are not defined relative to the application being run but relative to the machine's a rchitecture. These terms refer to other kinds of models for other kinds of exploits than their main exploit example. We restrict the use of these terms to those intended for [1].

They think of the states of the higher level abstraction as being mapped to the states of the lower level. They also tend to be a bit sloppy with their language. They think of a violation as occurring when a state in the lower level is reached that is not reverse mappable to the higher level. That is, a state is reached in the machine which is not in the image of the correct state in the upper level.

They set up a mapping M, which is a *simulation relation* represented as $M : S_V \rightarrow \mathcal{P}(S_O)$ for $\mathcal{P}$ the powerset functor. They do not describe the mapping as simulation relation; this is something the PI has recognized it to be from his prior research. There may be states outside of M's range and are termed *weird* states. The relation is generated by taking a state in the application and using it to describe a collection of states in the machine. A state in the application is merely the values in a collection of variables. A state in the machine includes that memory plus all the other bits and bobs of state in the machine, i.e., registers, memory, condition bits, etc.

Manuscript approved August 24, 2021.

Their set up from the paper is as follows

> Consider, for example, a model of $n < 256$ states, implemented as n values of a byte-sized variable in the memory of a process. The values of all other bytes in memory are irrelevant. Let us say the memory consists of N bytes, and thus has 2^{8N} states. Let the image of M consist of n subsets of $2^{8(N-1)}$ states each, and let there also be $256 - n$ subsets of $2^{8(N-1)}$ states in the induced partitioning that are not in M's image. Should the variable acquire a value outside of the n legitimate ones (say, due to an off-by-one arithmetic error), the underlying partition set could be thought of a "weird" state of the programmer-intended model, through which the actual computation goes.

Put another way, the one byte variable, call it v (v is a location in memory), induces an equivalence relation, $\mathcal{E}$ on the memory. A state x of the memory list of values for each of the N memory cells. For states x, y of the memory, define

$$\mathcal{E}_v xy \text{ iff } x \restriction v_i = y \restriction v_i$$

where $x \downarrow i$ sets the value of v to i and similarly to y. Since there are 256 values of i, there are 256 equivalence classes. If we only want to consider some n values of v as in the description, then we only want a subset of the equivalence classes induced by those n values of v. If there are n values of v that we wish to consider in the programmers model, then there are n equivalence classes in the programmers model and $256 - n$ that are "illegal".

They use the following diagram

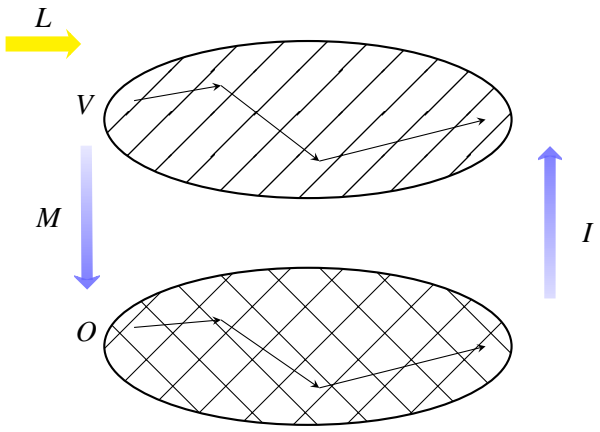


Figure 2.1: Programmer Intended (V) and actually obeyed (O) states and transitions.

where the sans serif L, M, I, V, and O in their diagram have been changed to L, M, I, V, and O to match the sequel of this report.

We will assume that any state change in the lower machine involves a state change to the location holding the variable v. This prevents us from needlessly modeling state changes not involving v. This assumption could easily be relaxed in the Distributed Logic analysis but would needlessly complicate that analysis.

2.1 First Paragraph of Example

Formally, they represent their model as follows, where they mean "transition relations" where they have "transition sets". These are really functions but in their set theoretic representation. M is morphism of both states and transitions.

> Assume there are two automata, V and O, with the state sets S_V and S_O and transition sets T_V and T_O respectively. Let them input the same language L.
>
> The mapping $\mathcal{M} : V \to \mathcal{P}(O)$, where $\mathcal{P}(O)$ is the powerset of O, connects V and O as (where I have augmented the presentation a bit to make it clearer). Also I have fixed what I believe to be a typo (more on that after the diagram)

$$
\begin{aligned}
\forall s_V \in S_V \quad & \mathcal{M}(s_V) \subset S_O \\
\forall s_V, s'_V \in S_V \quad & \mathcal{M}(s_V) \cap \mathcal{M}(s'_V) = \emptyset \\
\forall t_V \in T_V \ & \text{let } t_O = \mathcal{M}(t_V) \in T_O, \text{ and} \\
& \quad t_V(s_V) = s'_V, \text{ in} \\
& \quad\quad \mathcal{M}(t_V(s_V)) = t_O(\mathcal{M}(s_V))
\end{aligned}
$$

where I have turned $\mathcal{M}(t_V(s_V)) = t_O(\mathcal{M}(s_V))$ around from $t_O(\mathcal{M}(s_V)) = \mathcal{M}(t_V(s_V))$ so the left hand and right hand sides match the sides of the diagram below. The last two lines have the following depiction in terms of a simulation diagram with

$$
y = \mathcal{M}(s_V) \qquad y' = \mathcal{M}(t_V(s_V)) = t_O(\mathcal{M}(s'_V))
$$

Also, t_V is a pair of states in T_V treated as a function. That is, given the first state, it returns the second. A similar statement holds for t_O.

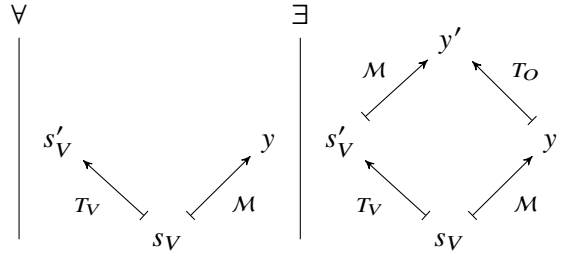


Figure 2.2: V-O Simulation

The diagram says that if the upper level makes a transition from s_V to s'_V and M relates s_V at the top level to y at the low level, then the low level makes a transition from y to y' and M relates s'_V to y.

Their original equation was

$$\mathcal{M}(t_V(s_V)) = t_O(\mathcal{M}(s'_V))$$

However, $t_O(M(s'_V))$ makes no sense because given $t_v(s_V) = s'_V$, we would have

$$\mathcal{M}(t_V(s_V)) = t_O(\mathcal{M}(t_v(s_V)))$$

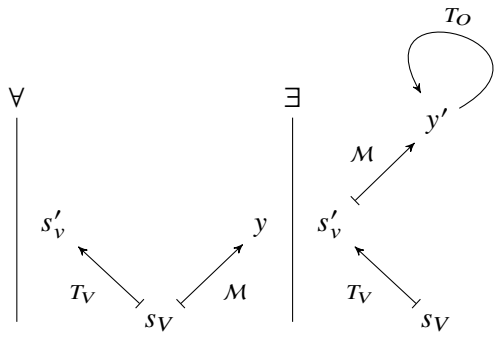


Figure 2.3: Bad V-O Simulation

2.2 Second Paragraph of Example

A rough depiction of these can be found in Figure 2.1.

Assume the partitioning P on S_O that agrees with that induced by $\mathcal{M}$ on the image $\mathcal{M}(S_V) \subset S_O$, i.e., $P = \{P_i\}$ such that $P_i = \mathcal{M}(s_i)$ for some $s_i \in S_V$, $P_i \cap P_j = \emptyset$ for $i \neq j$. If this partitioning can be naturally extended to $S_O \backslash \mathcal{M}(S_V)$, for example, if the sets P_i are natural translations of each other under some permutation of S_O (as in the above example), we'll have a partition $\bar{P} = \{\bar{P}_i\}$ of S_O, such that $P_i = \bar{P}_i$ for all $\bar{P}_i \subset \mathcal{M}(S_V)$, and $\bar{P}_i \cap \bar{P}_j = \emptyset$ for $i \neq j$.

We will let v exist both in the upper level S_V and the lower level S_O. Their use of the word "partitioning" is sloppy. The set of "partitions" P does not cover all of S_O and so do not constitute a partition of S_O. They will instead merely postulate a partitioning of S_O which includes P.

Their statement that $\bar{P}_i \subset \mathcal{M}(S_V)$ is suspect. Choose some j, then the set $P_j = M(s_j)$ and hence $P_j \in M(S_V)$. Since any P_j is similar to any $\bar{P}_i$ in the sense that $P_i, \bar{P}_j \subseteq S_O$, then $\bar{P}_j \in \mathcal{M}(S_V)$. We will assume this is what is meant.

The use of permutation seems to be an essential element of their model because they need to slice up the collection of lower level states not mapped from upper level via $\mathcal{M}$. Their permutation, we will call it $\mathcal{R}$,

must respect the equivalence classes determined by values of their state variable v. The permutation relation does not exist and there is no "permutation of S_O as in the above example".

They have an equivalence relation on some states of S_O and that equivalence relation induced P. This is the equivalence relation $\mathcal{E}_v$ defined above but restricted to identifying states with legal values of v. This will partition part of the set S_O into equivalence classes. They clearly intend for every equivalence class to be associated with some value i of the variable v. It need not be the case that two new equivalence classes over $S_O \backslash \mathcal{M}(S_V)$ disagree on the value of v, just that all states within an equivalence class agree on the same value of v.

Their definition of $\mathcal{M}$ is such that

$$P_i = \mathcal{M}v_i$$

We will use the v_i of S_V to define propositions such that $v = i$. Hence

$$x \models v = i \text{ iff } x \in \mathcal{M}v_i,$$

where we elide the parens, thus $\mathcal{M}v_i = \mathcal{M}(v_i)$. We are viewing $\mathcal{M}$ as a relation but represented as a function $\mathcal{M} : S_V \to \mathcal{P}S_O$. Hence $\mathcal{M}xy$ iff $y \in \mathcal{M}x$. We have

$$P_i \stackrel{def}{=} [m^\diamond\rangle(v = i).$$

That is, the term v_i denote propositions of V in a logical language and also denote states of S_V in a model of that language. $[m^\diamond\rangle(v = i)$ is the forward image of $v = i$ under $\mathcal{M}$:

$$\begin{aligned} y \models [m^\diamond\rangle(v = i) &\text{ iff } \exists x(\mathcal{M}xy \text{ and } x \models v_i) \\ &\text{iff } \mathcal{M}v_i y \\ &\text{iff } y \in \mathcal{M}v_i \end{aligned}$$

The P_i are required to be mutually disjoint:

$$\forall i, j((i \neq j \text{ and } 1 \leq i, j \leq n) \text{ implies } P_i \wedge P_j \supset \bot),$$

where $\bot$ is the proposition which is false everywhere. The P_i are also required to be natural translations of each other under a permutation $\mathcal{R}$. The permutation $\mathcal{R}$ is actually a parameter to their model. No prescription for it is given, so we are free to choose any $\mathcal{R}$ that respects the partition.

Since no permutation is actually given, choose anyone you like that permutes the states of S_O. As such, it will permute v_i to some v_j. A permutation must permute all of S_O; one choice is a permutation that leaves S_O as is except for the cell representing v. All that is required is that the P_i are such that a permutation must shift P_i to some P_j regardless of whether P_i or P_j is in the image of $\mathcal{M}$ as a map.

A permutation is also a function, we will let $\mathcal{R} : S_O \rightharpoonup S_O$ represent a permutation function as a relation. Being a function, it must satisfy

$$[r^\square\rangle Q \equiv [r^\diamond\rangle Q$$

where $\equiv$ stands for bi-implication. A permutation is $1-1$ and hence

$$\mathcal{R}xy \text{ and } \mathcal{R}zy \text{ implies } x = z.$$

Using the converse, where $\breve{\mathcal{R}}$ is the converse of the relation $\mathcal{R}$, i.e., if $\mathcal{R}xy$ then $\breve{\mathcal{R}}yx$, this becomes

$$\breve{\mathcal{R}}yx \text{ and } \breve{\mathcal{R}}yz \text{ implies } x = z.$$

Hence $\breve{\mathcal{R}}$ must be at least a partial function. It is also entire,

$$\forall y \exists x (\breve{\mathcal{R}}yx).$$

Together, this implies that $\breve{\mathcal{R}}$ is a function and hence we require it satisfy

$$[r^{\square}\rangle Q \equiv [r^{\diamond}\rangle Q,$$

where $[r^{\square}\rangle$ and $[r^{\diamond}\rangle$ are the backward looking operators interpreted by $\mathcal{R}$. The requirement that the P_i be natural translations of each other is the specification

$$\forall i \; \exists j \; (P_j \equiv [r^{\square}\rangle \, P_i).$$

The collection $\bar{P} = \{\bar{P}_i\}$ is the collection

$$\bar{P} = \{\bar{P}_i \mid \exists j (\bar{P}_j \equiv [r^{\square}\rangle \, P_i)\}$$

Finally,

$$\forall i (1 \leq i \leq n \text{ implies } P_i \equiv \bar{P}_i) \text{ and } \forall i, j (i \neq j \text{ implies } \bar{P}_i \wedge P_j \supset \bot).$$

The P_i that are not in the image of $\mathcal{M}$ are now considered new *abstract* states of S_V, we could simply equate them to the illegal values of v. However, they leave room in their representation to choose the permutation. Thus, even if the legal states are associated with legal values of v, the new abstract states are not.

2.3 Third Paragraph of Example

We quote (where we have changed $\subset$ to $\subseteq$ to adhere to accepted mathematical practice)

> Then we can formally extend S_V and $\mathcal{M}$ with the "preimage" of $\bar{P}_k : \bar{P}_k \subseteq S_O \backslash \mathcal{M}(S_V)$, writing $\bar{\mathcal{M}}$ and $\bar{S}_V$, so that $\bar{S}_V = S_V \cup W$ where

$$W = \{s_W : \bar{\mathcal{M}}(s_W) = \bar{P}_k, \bar{P}_k \subseteq S_O \backslash \mathcal{M}(S_V)\}.$$

Similarly, we extend T_V with transitions to and from W,

$$\bar{T}_V = T_V \cup \{s \rightarrow w : s \in S_V, w \in W\} \cup \{w \rightarrow s : s \in S_V, w \in W\} \cup \{w \rightarrow w' : w, w' \in W\},$$

as induced by the execution paths in S_O, T_O.

Their extension is not exactly well explained or well-defined. It reads as if there was some pre-existing state s_W when what is meant is there is a new formal state s_W that will be added to the collection for V. We get a new s_W for every stable (under the permutation relation) $\bar{P}_k \subset S_O \backslash \mathcal{M}(S_V)$. Let this map be $\mathcal{N}_W$, i.e., $\mathcal{N}(s_W) = \bar{P}_k$. Their second mathematical display should also read

$$\bar{T}_V = T_V \cup \{s \rightarrow s_w : s \in S_V, s_w \in W\} \cup \{s_w \rightarrow s : s \in S_V, s_w \in W\} \cup \{s_w \rightarrow s_{w'} : s_w, s'_w \in W\},$$

The new transitions are in $\bar{S}_V$. They assume that T_O is such that there are transitions that are not the image of transitions from S_V under $\mathcal{M}$. As such, they are another parameter to their model. Put another way, they must be chosen. Presumably, they are chosen to represent a particular exploit.

The relation $\mathcal{M}$ is extended to $\bar{\mathcal{M}}$ with

$$\bar{\mathcal{M}} = \mathcal{M} \cup \{\langle s_W, \bar{P}_k \rangle \mid \mathcal{N}(s_W) = \bar{P}_k\} = \mathcal{M} \cup \mathcal{N}.$$

The extension of T_V to $\bar{T}_V$ is better explained as

$$\bar{T}_V xy \text{ iff } \exists p, q \in S_O (T_O pq \text{ and } p \in \bar{\mathcal{M}} x \text{ and } q \in \bar{\mathcal{M}} y).$$

The relationships between $\mathcal{M}$ and $\bar{\mathcal{M}}$, and T_V and T_O, are subject to the axioms

$$[m^\diamond\rangle Q \supset [\bar{m}^\diamond\rangle Q \qquad [t_V^\diamond\rangle Q \supset [\bar{t}_V^\diamond\rangle Q$$

where Q ranges over propositions of O, and the t_V and t_O in the logic's distributed connectives refer to the next state relations T_V and T_O, not the individual pairs of those relations as in their setup.

2.4 Fourth Paragraph

We can think of the preimage mapping $I : \{\bar{P}_i\} \rightarrow W$ or the more general $\bar{I} : S_O \rightarrow \bar{S}_V$, which composes with $\mathcal{M}$ as

$$(\forall s \in S_V \cup W)(I(\mathcal{M}(s)) = s).$$

The mapping I is, in these terms, the essence of V's implementation by O (hence the choice of "I").

There prescription should read

$$(\forall s \in S_V \cup W)(I(\bar{\mathcal{M}}(s)) = s).$$

because what they are trying to express is $I \circ \bar{\mathcal{M}} = \mathrm{id}_{\bar{S}_V}$.

2.5 Fifth Paragraph

> With these extensions, we can describe the computations taken in machines V and O on an input $l \in L$ as the sequences of states and transitions $(s_{O_1}, t_{O_1}, \ldots, s_{O_i}, t_{O_i}, \ldots)$ for O and $(s_1, t_1, \ldots, s_i, t_i, \ldots)$ for V, where $s_i \in S_V, t_i \in T_V$ for all $i < i_0, s_{i_0} \in W, t_{i_0} \in \bar{T}_V$, and $s_i \in S_V \cup W, t_i \in \bar{T}_V$ for $i > i_0$. For these two sequences, $\mathcal{M}(s_i) = s_{O_i}, \mathcal{M}(t_i) = t_{O_i}$ for all i.

The location in the sequences, i_0, indicates where the computation goes off the rails and ventures into states that are not mapped from the original $\mathcal{M}$. Hence s_{i_0} is an formal state put into W by the extension to capture illicit computations. After that, succeeding states in V may fall into either S_V or W. Succeeding transitions will be in the extended $\bar{T}_V$.

The last statement appears to be in error, it should say $\bar{\mathcal{M}}(s_i) = s_{O_i}, \bar{\mathcal{M}}(t_i) = t_{O_i}$ for all i by using the extended mapping $\bar{\mathcal{M}}$ and not the unextended $\mathcal{M}$.

2.6 Sixth Paragraph

> Whether this construction appears natural or not, depends on the internal structure of the underlying state space S_O and its partitioning P induced by $\mathcal{M}$. Any natural symmetries on the subsets of S_O that map $\{P_i\}$ around and allow to extend it to a partition $\bar{P}$ of the entire S_O provide this structure. Then, on any path in the S_O state space we can lift any state $s \in S_O \backslash \mathcal{M}(S_V)$ to the preimage w_{P_k} such that $\bar{\mathcal{M}}(w_{P_k}) = P_k$.

The last statement leaves it to be inferred that the particular s such that $s \in S_O \backslash \mathcal{M}(S_V)$ implies that particular s is such that $s \in P_k$.

2.7 Seventh Paragraph

> The extension of the transition set T_V is tied with those of S_V and $\mathcal{M}$. Specifically, for a transition $s_O \to s'_O, s_O \in \mathcal{M}(S_V), s'_O \in S_O \backslash \mathcal{M}(S_V)$,
>
> $$\begin{aligned} \mathcal{M}(s_V) &= s_O, \\ \bar{\mathcal{M}}(w) &= s'_O \end{aligned}$$
>
> we put $\bar{t} \in \bar{T}_V$ defined as $\bar{t} : s \to w$.

This pokes in the missing explanation for $\bar{T}_V$ that I had put in above. However, it strikes me that it does not go far enough because surely the computation can progress from w further into a w' and that w' is not restricted to fall into S_V.

2.8 Final Two Paragraphs

They go on to state that underlying structure on S_O is required to get extensions of the upper level state space and transition function so that they are not contrived. However, in the their example the underlying structure of S_O is not really any underlying structure at all since it was derived from S_V via their lone state variable. This seems as though it can always be constructed given the state space of the upper level gets mapped to state space of the lower level and the size of the variables becomes fixed given the lower level machine.

3. EXPRESSING CODE EXPLOITATION IN LOGIC

In Floyd-Hoare analysis, triples of the form

$$U \; \{\mathcal{H}\} \; Q$$

are transformed into the logical statement

$$U \supset [h^{\square}\rangle \, Q$$

where $\mathcal{H}$ is a fragment of code thought of as an intensional description of a relation between the code's input and output. The necessity operator, $[h^{\square}\rangle \, Q$ is the imprint of $\mathcal{H}$ on a logic over the code's execution and in Floyd-Hoare analysis is usually depicted as $wp \, (\mathcal{H})$.

The propositions U and Q are similarly intensional descriptions of collections of states where we might view U as standing for the predicate $U(x)$ for x a variable over states. With x a free variable, this has the import of $\forall x(U(x))$. Incidentally, $U(x)$ is part of the standard interpretation for modal logic into classical logic.

The question then arises that if $\mathcal{H}$ is executed without its guard being true, i.e., with U violated, what can we expect out of the computation and how do we express this? One way to phrase this is to see what effect the computation has on Q given the violation. A possible way to express this is

$$\neg U \wedge [h^{\circ}\rangle \, Q.$$

Another possible way to get at the information of what $\mathcal{H}$ does in this situation is to use the residual operator for $[h^{\square}\rangle$, namely $[h^{\circ}\cdot\rangle$ (see Section 4.1). The result of $\mathcal{H}$ operating on a violated U can then be expressed as

$$[h^{\circ}\cdot\rangle \neg U.$$

In modal logic terms,

$$y \models^{\mathsf{h}} [h^{\circ}\cdot\rangle \neg U \text{ iff } \exists x(\mathcal{H}xy \text{ and } x \models^{\mathsf{h}} \neg U).$$

In prose,

> y makes $[h^\circ\rangle\neg U$ true just when there is some x that we determine by looking back along the computation $\mathcal{H}$ such that $\mathcal{H}$ operating on x results in y and x violates U.

The h appellation to $\models$ is an indication that we assume $\mathcal{H} : \mathsf{h} \rightharpoonup \mathsf{h}$, i.e., that $\mathcal{H}$ is a relation at a locality h. That it is not a distributed relation is indicated by the $\overset{\mathsf{h}}{\models}$ on both sides of the iff.

In distributed logic, there are two operators which are evaluated at the top layer of the semantics. These are called *spider connectives*. A spider is a form of realtime monitor and spider connectives come up in logical analysis of spiders.

$$U \xrightarrow{[h]} Q \overset{def}{=} \mathsf{A}\,(U \supset [h^\square\rangle\, Q) \qquad U \xrightarrow{\langle h\rangle} Q \overset{def}{=} \mathsf{E}\,(U \wedge [h^\circ\rangle\, Q)^1$$

It is but a short step to introduce

$$U \xrightarrow{\langle h\rangle} Q \overset{def}{=} \mathsf{E}\,([h^\circ\rangle U \wedge Q).$$

The first is evaluated with

$$\begin{aligned} z \models_{\mathsf{h}} \mathsf{A}\,(U \supset [h^\square\rangle\, Q) \text{ iff } & \forall x(x \models_{\mathsf{h}} U \text{ implies } x \models_{\mathsf{h}} [h^\square\rangle\, Q) \\ \text{iff } & \forall x(x \models_{\mathsf{h}} U \text{ implies } \forall y(\mathcal{H}xy \text{ implies } y \models_{\mathsf{h}} Q)) \\ \text{iff } & \forall x, y(x \models_{\mathsf{h}} U \text{ implies } (\mathcal{H}xy \text{ implies } y \models_{\mathsf{h}} Q)) \\ \text{iff } & \forall x, y((x \models_{\mathsf{h}} U \wedge \mathcal{H}xy) \text{ implies } y \models_{\mathsf{h}} Q) \end{aligned}$$

Notice that the evaluation of A throws out the initial state z and proceeds to evaluate over all states indexed by x. The second formula is evaluated with

$$\begin{aligned} z \models_{\mathsf{h}} \mathsf{E}\,([h^\circ\rangle U \wedge Q) \text{ iff } & \exists y(y \models_{\mathsf{h}} [h^\circ\rangle U \text{ and } y \models_{\mathsf{h}} Q) \\ \text{iff } & \exists y(\exists x(\mathcal{H}xy \text{ and } x \models_{\mathsf{h}} U) \text{ and } y \models_{\mathsf{h}} Q) \\ \text{iff } & \exists y, x(\mathcal{H}xy \text{ and } x \models_{\mathsf{h}} U \text{ and } y \models_{\mathsf{h}} Q) \end{aligned}$$

The operator E similarly throws out the initial state z and proceeds to evaluate the formula at the top layer of the semantics.

3.1 Bad Changes

If we concentrate merely on the low level machine and ignore the high level machine, there are several kinds of bad state change in the low level machine. Possible kinds of bad state change $s \xrightarrow{a} q$ for some action a are

- q is a state that s should never go to under a,

[1] The reason for the arrows is that these operators were suggested by research by the PI into fibration models for Distributed Logic.

- s is a state that should never be the source of a going to q,
- s in context Q_1 should never go under a to q in context Q_2,
- q in context Q_2 should never have been arrived at under a from a state s in context Q_1.

The first two can be reduced to the second two by choosing $\top$ for Q_1 and Q_2 where $\top$ represents the proposition *true* or, in set theory terms, the collection of all states at a locality (here, the locality is the low level machine). In Distributed Logic with spider connectives, we have where the relation $\mathcal{A}$ is associated with the action a:

- $\neg(Q_1 \xrightarrow{\langle a \rangle} Q_2)$
- $\neg(Q_2 \xrightarrow{\langle a \rangle} Q_1)$

Now we check

$$\begin{aligned} x \in \neg(Q_1 \xrightarrow{\langle a \rangle} Q_2) \text{ iff } & x \notin (Q_1 \xrightarrow{\langle a \rangle} Q_2) \\ \text{iff } & \neg \mathsf{E}\,(Q_1 \wedge [a^\circ\rangle\, Q_2) \\ \text{iff } & \neg\exists y(y \in Q_1 \wedge [a^\circ\rangle\, Q_2) \\ \text{iff } & \forall y(y \notin Q_1 \text{ or } y \notin [a^\circ\rangle\, Q_2) \\ \text{iff } & \forall y(y \notin Q_1 \text{ or } \forall z(\mathcal{A}yz \text{ implies } z \notin Q_2)) \end{aligned}$$

This seems to work. Next,

$$\begin{aligned} x \in \neg(Q_2 \xrightarrow{\langle a \rangle} Q_1) \text{ iff } & x \notin (Q_2 \xrightarrow{\langle a \rangle} Q_1) \\ \text{iff } & \neg \mathsf{E}\,(Q_2 \wedge [a^\circ\cdot\rangle Q_1) \\ \text{iff } & \neg\exists y(y \in Q_2 \wedge [a^\circ\cdot\rangle Q_1) \\ \text{iff } & \forall y(y \notin Q_2 \text{ or } y \notin [a^\circ\cdot\rangle Q_1) \\ \text{iff } & \forall y(y \notin Q_2 \text{ or } \forall z(\mathcal{A}zy \text{ implies } z \notin Q_1)) \end{aligned}$$

3.2 Generalized Simulations

We will change the notation a bit. The map $\mathcal{M}$ will be $\mathcal{F}$, t_v becomes $\mathcal{H}$, and t_O becomes $\mathcal{K}$. The location V becomes h and the location O becomes k. The basic simulation

$$[f^\circ\rangle\,[k^\square\rangle\, Q \overset{\mathsf{h}}{\supset} [h^\square\rangle\,[f^\circ\rangle\, Q$$

is satisfied when the compilation from high level to the machine level is valid. The appellation h of $\overset{\mathsf{h}}{\supset}$ indcates that the formula is a proposition at locality h. Q is a proposition in locality k. The subformula $[f^\circ\rangle\, Q$ pulls Q back along the relation $\mathcal{F}$ and hence $[f^\circ\rangle\, Q$ is a proposition in h. $[h^\square\rangle\,[f^\circ\rangle\, Q$ pulls $[f^\circ\rangle\, Q$ back along a

computation $\mathcal{H}$ in h. In terms of Floyd-Hoare, $[h^{\square}\rangle\,[f^{\circ}\rangle\,Q$ is $wp\,([f^{\circ}\rangle)(Q)$. Similarly, $[k^{\square}\rangle\,Q$ pulls Q back along $\mathcal{K}$ and is $wp\,(\mathcal{K})(Q)$ and is a proposition of k. $[f^{\circ}\rangle\,[k^{\square}\rangle\,Q$ pulls $[k^{\square}\rangle\,Q$ back along $\mathcal{F}$.

This can also be viewed as

$$[h^{\circ}\cdot\rangle\,[f^{\circ}\rangle\,Q \overset{\mathsf{h}}{\supset} [f^{\circ}\rangle\,[k^{\circ}\cdot\rangle Q.$$

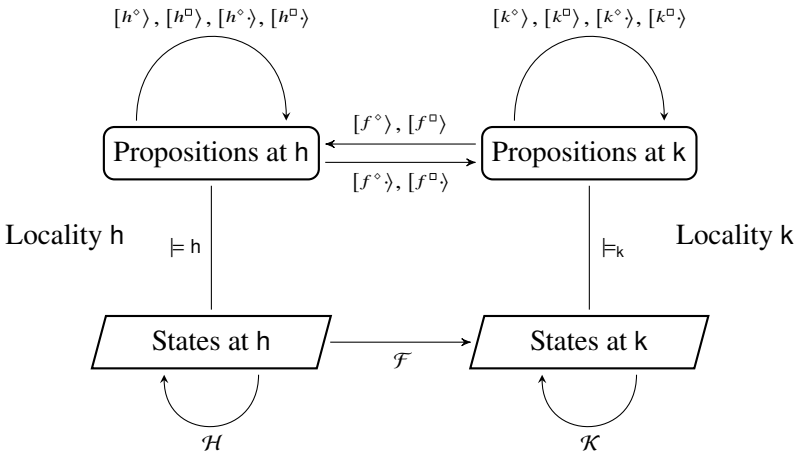
This formula is moving propositions from the machine level to the upper level.

Their restriction on $\mathcal{F}$ (formerly $\mathcal{M}$) is that it generates propositions at the machine level. The effect is to use propositions of the form

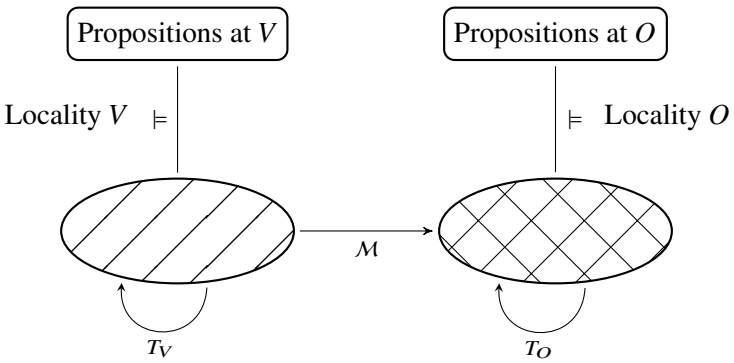
$$[f^{\square}\cdot\rangle U$$

for U a proposition at the high level. $[f^{\square}\cdot\rangle U$ is a proposition at the low level because $[f^{\square}\cdot\rangle$ is the forward image operator (connective) defined by $\mathcal{F}$.

The typical distributed logic diagram is



Their diagram, with the addition of propositions becomes,



We have at our disposal the following generalized simulation conditions where $\breve{\mathcal{H}}$ is the converse of the relation $\mathcal{H}$, i.e., if $\mathcal{H}xy$ then $\breve{\mathcal{H}}yx$:

Name	Frame Condition	Axiom
Forward Simulation	$\breve{\mathcal{H}} \cdot \mathcal{F} \subseteq \mathcal{G} \cdot \breve{\mathcal{K}}$	$[h^\circ\rangle [f^\circ\rangle Q \subseteq [g^\circ\rangle [k^\circ\rangle Q$
Proverse Simulation	$\mathcal{H} \cdot \mathcal{F} \subseteq \mathcal{G} \cdot \mathcal{K}$	$[h^\circ\rangle [f^\circ\rangle Q \subseteq [g^\circ\rangle [k^\circ\rangle Q$
Converse Simulation	$\mathcal{G} \cdot \breve{\mathcal{K}} \subseteq \breve{\mathcal{H}} \cdot \mathcal{F}$	$[g^\circ\rangle [k^\circ\cdot\rangle Q \subseteq [h^\circ\cdot\rangle [f^\circ\rangle Q$
Backward Simulation	$\breve{\mathcal{K}} \cdot \breve{\mathcal{F}} \subseteq \breve{\mathcal{G}} \cdot \breve{\mathcal{H}}$	$[k^\circ\cdot\rangle [f^\circ\cdot\rangle U \subseteq [g^\circ\cdot\rangle [h^\circ\cdot\rangle U$
Possibility Preservation	$\mathcal{H} \cdot \mathcal{G} \subseteq \mathcal{F} \cdot \mathcal{K}$	$[h^\circ\rangle [g^\circ\rangle Q \subseteq [f^\circ\rangle [k^\circ\rangle Q$

Taking the contrapositive of possibility preservation, pushing through the resulting negation, and generalizing to remove the negations yields necessity preservation, i.e.,

$$[f^\square\rangle [k^\square\rangle Q \subseteq [h^\square\rangle [g^\square\rangle Q.$$

These simulation and preservation axioms have rule forms, The rule forms allow us to move around the graph structure of the distributed logics in a way that the axiom forms do not. In these rules, S and R refer to propositions, not the collection of states at the upper level or the relation $\mathcal{R}$ used above:

Forward Simulation —

$$\frac{Q \xrightarrow{[k]} S \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\rangle Q \xrightarrow{[h]} [g^\circ\rangle S}$$

and

$$\frac{Q \xrightarrow{[k]} [g^\square\cdot\rangle R \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\rangle Q \xrightarrow{[h]} R} \qquad \frac{[f^\circ\rangle Q \xrightarrow{\langle h\rangle} R \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{Q \xrightarrow{\langle k\rangle} [g^\circ\cdot\rangle R}$$

Backward Simulation —

$$\frac{R \xrightarrow{[h\cdot]} U \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\cdot\rangle R \xrightarrow{[k\cdot]} [g^\circ\cdot\rangle U}$$

and

$$\frac{[g^\circ\rangle Q \xrightarrow{[h]} R \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{Q \xrightarrow{[k]} [f^\square\cdot\rangle R} \qquad \frac{Q \xrightarrow{\langle k\rangle} [f^\circ\cdot\rangle R \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{[g^\circ\rangle Q \xrightarrow{\langle h\rangle} R}$$

Proverse Simulation —

$$\frac{Q \xrightarrow{[k\cdot]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\rangle Q \xrightarrow{[h\cdot]} [g^\circ\rangle S}$$

and

$$\frac{[g^\circ\cdot\rangle U \xrightarrow{[k]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{U \xrightarrow{[h]} [f^\square\rangle S} \qquad\qquad \frac{U \xrightarrow{\langle h\rangle} [f^\circ\rangle S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[g^\circ\cdot\rangle U \xrightarrow{\langle k\rangle} S}$$

Converse Simulation —

$$\frac{U \xrightarrow{[h]} R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\cdot\rangle U \xrightarrow{[k]} [g^\circ\cdot\rangle R}$$

and

$$\frac{U \xrightarrow{[h]} [g^\square\rangle S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\circ\cdot\rangle U \xrightarrow{[k]} S} \qquad\qquad \frac{[f^\circ\cdot\rangle U \xrightarrow{\langle k\rangle} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{U \xrightarrow{\langle h\rangle} [g^\circ\rangle S}$$

Necessity Preservation —

$$\frac{Q \xrightarrow{[k]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\square\rangle Q \xrightarrow{[h]} [g^\square\rangle S}$$

3.3 Analysis of Exploitation

In their set up, they extend $\mathcal{M}$ to $\bar{\mathcal{M}}$. We will use $\mathcal{G}$ in place of $\bar{\mathcal{M}}$ subject to the axiom

$$[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle Q$$

This axiom says, once the evaluation conditions are unwound that $\mathcal{F} \subseteq \mathcal{G}$, or in their notation $\mathcal{M} \subseteq \bar{\mathcal{M}}$. Letting $\bar{\mathcal{H}}$ represent the extended T_V, namely $\bar{T}_V$, their set up satisfies

$$[h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [f^\circ\rangle[k^\circ\cdot\rangle Q, \quad [h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle[k^\circ\cdot\rangle Q, \quad [\bar{h}^\circ\cdot\rangle[g^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle[k^\circ\cdot\rangle Q, \quad [k^\circ\cdot\rangle[g^\circ\cdot\rangle U \overset{\mathsf{k}}{\supset} [g^\circ\cdot\rangle[\bar{h}^\circ\cdot\rangle U$$

where the low-level machine is simulating the high-level machine. The $\overset{\mathsf{h}}{\supset}$ indicates the formula is in locality h and $\overset{\mathsf{k}}{\supset}$ indicates the formula is in locality k. The propositional variable Q is in locality k and U is in h.

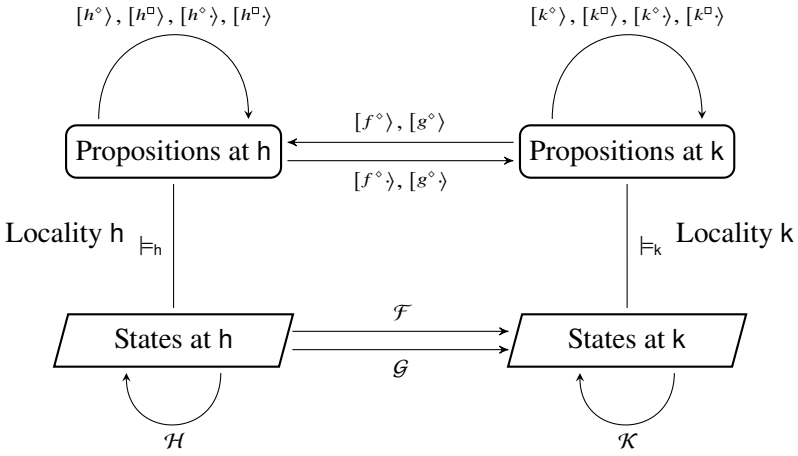
The first formula says the low level machine simulates the high level on the states S_V using $\mathcal{M}$. The second says the low level simulates the high level on the states S_V using $\bar{\mathcal{M}}$. The third says the low level

simulates the extended high level on the state $S_V \cup W$ using $\bar{\mathcal{M}}$. The last says the extended high level simulates the low level using the converse of the extended relation $\bar{\mathcal{M}}$.

In terms of Floyd-Hoare weakest preconditions, these formulas are

$$[f^\circ\rangle\,[k^\square\rangle\,Q \overset{\mathsf{h}}{\supseteq} [h^\square\rangle\,[f^\circ\rangle\,Q, \quad [f^\circ\rangle\,[k^\square\rangle\,Q \overset{\mathsf{h}}{\supseteq} [h^\square\rangle\,[g^\circ\rangle\,Q, \quad [g^\circ\rangle\,[k^\square\rangle\,Q \overset{\mathsf{h}}{\supseteq} [\bar{h}^\square\rangle\,[g^\circ\rangle\,Q, \quad [g^\circ\cdot\rangle\,[\bar{h}^\square\rangle\,U \overset{\mathsf{k}}{\supseteq} [k^\square\rangle\,[g^\circ\cdot\rangle\,U.$$

It helps to learn to read the backwards distributed operators, i.e., the ones with the speckles, $[\bar{h}^\circ\cdot\rangle$, $[k^\circ\cdot\rangle$. and $[g^\circ\cdot\rangle$ using the converse relations underlying their semantics.



The diagrams below labeled "Simulation" are had from the generalized simulation conditions above by identifying $\mathcal{G}$ with $\mathcal{F}$, and x, x' are in locality h representing the high level and y, y' are in locality k representing the low level:

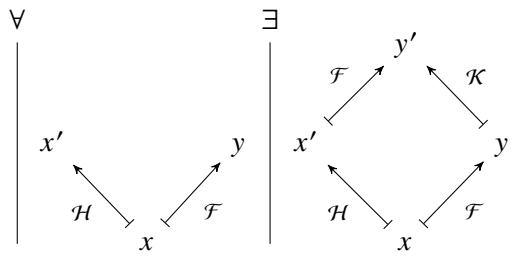


Figure 3.1: Simulation

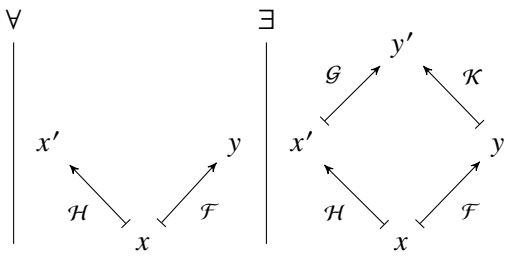


Figure 3.2: Generalized Simulation

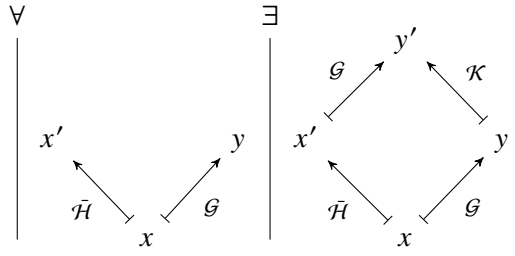


Figure 3.3: Simulation

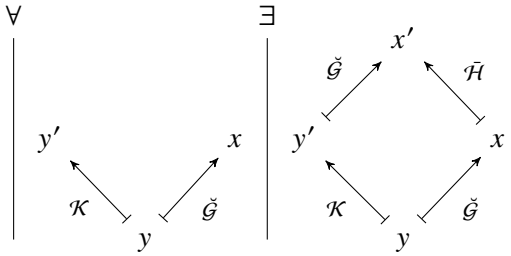


Figure 3.4: Backward Simulation

The formula $[\bar{h}^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle[k^\circ\cdot\rangle Q$ then says

- Looking back along the extended next state relation $\bar{\mathcal{H}}$ from x' to x and x is related to some y where y makes Q true, then
- x' is also related to some y' under the extended simulation relation $\mathcal{G}$ and y' is related looking backwards along $\mathcal{K}$ to some z where z makes Q true.
- The only way to force z to make Q true is for $y = z$.

Theorem 3.3.0.1 *These formulas are valid in any model satisfying the above diagrams (or their first-order logic equivalents)*

$$[h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [f^\circ\rangle[k^\circ\cdot\rangle Q, \quad [h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle[k^\circ\cdot\rangle Q, \quad [\bar{h}^\circ\cdot\rangle[g^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle[k^\circ\cdot\rangle Q, \quad [k^\circ\cdot\rangle[g^\circ\cdot\rangle U \overset{\mathsf{h}}{\supset} [g^\circ\cdot\rangle[\bar{k}^\circ\cdot\rangle U$$

Admittedly, the last step of making $y = z$ is not easily read off the formula but rather comes about from looking at the proof of this formula:

Proof:

1	$x' \models^{\mathsf{h}} [\bar{h}^\circ\cdot\rangle[f^\circ\rangle Q$	assume
2	$\bar{\mathcal{H}}xx'$ and $x \models^{\mathsf{h}} [f^\circ\rangle Q$	def. $[\bar{h}^\circ\cdot\rangle$ for some x, line 1
3	$\bar{\mathcal{H}}xx'$ and $\mathcal{F}xy$ and $y \models^{\mathsf{k}} Q$	def. $[f^\circ\rangle$ for some y, line 2
4	$\bar{\mathcal{H}}xx'$ and $\mathcal{F}xy$	$\wedge$-Elim, line 3
5	$\breve{\bar{\mathcal{H}}}x'x$ and $\mathcal{F}xy$	def. $\breve{\bar{\mathcal{H}}}$, line 4
6	$(\breve{\bar{\mathcal{H}}} \cdot \mathcal{F})x'y$	def. $\cdot$, line 5
7	$(\mathcal{G} \cdot \breve{\mathcal{K}})x'y$	simulation condition, line 6
8	$\mathcal{G}x'y'$ and $\breve{\mathcal{K}}y'y$	def. $\cdot$ for some y', line 7
9	$\mathcal{K}y'y$ and $y \models^{\mathsf{k}} Q$	$\wedge$-Elim, $\wedge$-Intro, lines 3, 8
10	$y' \models^{\mathsf{k}} [k^\circ\rangle Q$	def. $[k^\circ\cdot\rangle$, line 9
11	$\mathcal{G}x'y'$ and $y' \models^{\mathsf{k}} [k^\circ\cdot\rangle Q$	$\wedge$-Elim, $\wedge$-Intro, lines 8, 10
12	$x' \models^{\mathsf{h}} [g^\circ\rangle\,[k^\circ\cdot\rangle Q$	def. $[g^\circ\rangle$, line 11

By set theory,

$$[\bar{h}^\circ\cdot\rangle[f^\circ\rangle Q \subseteq_{\mathsf{h}} [g^\circ\rangle\,[k^\circ\cdot\rangle Q.$$

By the semantics of $\overset{\mathsf{h}}{\supset}$,

$$[\bar{h}^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle\,[k^\circ\cdot\rangle Q.$$

The proofs of the rest are similar. ∎

One can combine some axioms:

Theorem 3.3.0.2

$$\big(([h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [f^\circ\rangle\,[k^\circ\cdot\rangle Q) \text{ and } ([f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle Q)\big) \text{ implies } [h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle\,[k^\circ\cdot\rangle Q.$$

Proof: The step 7 in the proof of $[h^\circ\cdot\rangle[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [f^\circ\rangle\,[k^\circ\cdot\rangle Q$ can be augmented with the condition $\mathcal{F} \subseteq \mathcal{G}$. The formula $[f^\circ\rangle Q \overset{\mathsf{h}}{\supset} [g^\circ\rangle Q$ is shown to hold in all models where $\mathcal{F} \subseteq \mathcal{G}$. ∎

There are two kinds of incorrect state changes, [1] only identifies the first below:

- state change to a weird state, and
- incorrect state change between the P_i.

The first is captured via the formula below

$$\neg\big([g^\diamond\rangle\,[k^\diamond\cdot\rangle Q \overset{\mathsf{h}}{\supset} [h^\diamond\cdot\rangle[f^\diamond\rangle Q\big).$$

where Q is a collection of abnormal states. The formula on the left below is in non-standard notation. It is an easy step from that formula to its set theoretical equivalent and easier to read:

$$[g^\diamond\rangle\,[k^\diamond\cdot\rangle Q \overset{\mathsf{h}}{\not\supset} [h^\diamond\cdot\rangle[f^\diamond\rangle Q \qquad [g^\diamond\rangle\,[k^\diamond\cdot\rangle Q \not\subseteq_{\mathsf{h}} [h^\diamond\cdot\rangle[f^\diamond\rangle Q$$

Taking away the negation signs, these formulas are satisfied by a converse simulation where the low-level k is simulated by the high-level h. In words, and with the negation signs back in, they say that k can make a move where h cannot match it with a corresponding move. That is, k can move to a weird state but h cannot match it with a move to a normal state. Were the formula to be

$$[g^\diamond\rangle\,[k^\diamond\cdot\rangle Q \overset{\mathsf{h}}{\supset} [\bar{h}^\diamond\cdot\rangle[f^\diamond\rangle Q$$

with the extended $\bar{\mathcal{H}}$ in place of $\mathcal{H}$, this is somewhat nonsensical because $[f^\diamond\rangle$ is not defined over propositions of weird states. The formula

$$[g^\diamond\rangle\,[k^\diamond\cdot\rangle Q \overset{\mathsf{h}}{\supset} [\bar{h}^\diamond\cdot\rangle[g^\diamond\rangle Q$$

is however satisfied where the extended relation $\mathcal{G}$ is used in place of $\mathcal{F}$ and $[g^\diamond\rangle$ is defined over propositions of weird states.

The incorrect state change between the P_i is captured by

$$[h^\diamond\cdot\rangle[f^\diamond\rangle P_i \overset{\mathsf{h}}{\not\supset} [f^\diamond\rangle\,[k^\diamond\cdot\rangle P_i$$

for some i where $[k^\diamond\cdot\rangle P_i = P_j$ for some j.

We could make use of generalized simulation which uses either of the formulas

$$[f^\diamond\rangle\,[k^\square\rangle Q \supset [h^\square\rangle\,[g^\diamond\rangle Q \qquad [h^\diamond\cdot\rangle[f^\diamond\rangle Q \supset [g^\diamond\rangle\,[k^\diamond\cdot\rangle Q.$$

or any of the rules

$$\frac{Q \xrightarrow{[k]} S \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\diamond\rangle Q \xrightarrow{[h]} [g^\diamond\rangle S}$$

and

$$\frac{Q \xrightarrow{[k]} [g^\square\cdot\rangle R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{[f^\diamond\rangle Q \xrightarrow{[h]} R} \qquad\qquad \frac{[f^\diamond\rangle Q \xrightarrow{\langle h\rangle} R \qquad f,g : \mathsf{h} \rightharpoonup \mathsf{k}}{Q \xrightarrow{\langle k\rangle} [g^\diamond\cdot\rangle R}$$

These rules allow one to push and pull formulas to and from the localities h and k. Floyd-Hoare analysis uses triples of the form

$$U\ \{\mathcal{H}\}\ Q$$

where U is a precondition, $\mathcal{H}$ is a fragment of executable code, and Q is a postcondition. The code fragment $\mathcal{H}$ expresses a Kripke relation in intensional form; that is, it is a prescription in a syntax that relates input to output.

The Floyd-Hoare triples can be related to the first rule form. This rule form expresses pulling a Floyd-Hoare triple at the low level k up to the high level h against the underlying relations $\mathcal{F}$ and $\mathcal{G}$ where Q is a precondition in the lower machine, $[k^{\square}\rangle$ stands for a fragment of computation (i.e., an intensional description of a Kripke relation) on the lower level, S is the post condition, and $[k^{\square}\rangle S$ is the result of dragging S back through the computation $[k^{\square}\rangle$. Additionally, $[f^{\circ}\rangle$ pulls Q back up to the high level and $[g^{\circ}\rangle$ pulls S back up the high level. The underlying relations $\mathcal{F}$ and $\mathcal{G}$ constitute the generalized simulation. For a single relation expressing a simulation, let $\mathcal{F} = \mathcal{G}$ and consequently, $[f^{\circ}\rangle$ is the same operator as $[g^{\circ}\rangle$.

The premise of the rule instance is an Floyd-Hoare statement for the result of dragging S back through $[k^{\square}\rangle$ and demanding that Q imply $[k^{\square}\rangle S$ (on the low level). The conclusion is the result of pulling Q and S up the high level via $[f^{\circ}\rangle$ and $[g^{\circ}\rangle$ respectively, and then demanding that $[f^{\circ}\rangle Q$ implies the result of dragging $[g^{\circ}\rangle S$ back along $[h^{\square}\rangle$ where $[h^{\square}\rangle$ is a fragment of computation at the high level. Presumably, $[h^{\square}\rangle$ compiles to $[k^{\square}\rangle$.

In more common computer science notation where we use $\mathcal{F}^{-1}$ in place of $[f^{\circ}\rangle$ and $\mathcal{G}^{-1}$ in place of $[g^{\circ}\rangle$, the rule would be

$$\frac{Q \overset{\mathsf{k}}{\supset} wp(\mathcal{K}, S) \qquad f, g : \mathsf{h} \rightharpoonup \mathsf{k}}{\mathcal{F}^{-1}Q \overset{\mathsf{h}}{\supset} wp(\mathcal{H}, \mathcal{G}^{-1}S)}$$

4. APPENDIX

4.1 Residuation

Residuation occurs throughout non-standard logic. In modal logic, the backwards and forwards operators are residuated:

$$[h^{\circ}\cdot\rangle U \supset Q \text{ iff } U \supset [h^{\square}\rangle Q \qquad [h^{\circ}\rangle U \supset Q \text{ iff } U \supset [h^{\square}\cdot\rangle Q.$$

The first is of immediate interest since weakest preconditions are frequently stated as $U \supset [h^{\square}\rangle Q$ for the triple

$$U\{\mathcal{H}\}Q$$

where $\mathcal{H}$ stands for a relation (usually a function) stated in intensional form, i.e., a fragment of a program. Also, in distributed logic

$$U \xrightarrow{[h]} Q \overset{def}{=} \mathsf{A}\,(U \supset [h^{\square}\rangle Q).$$

Since $U \supset [h^{\square}\rangle\, Q$ usually occurs at the top level of any Floyd-Hoare semantics, $\mathsf{A}\,(U \supset [h^{\square}\rangle\, Q)$ is usually equivalent to $U \supset [h^{\square}\rangle\, Q$.

The residuated operator $[h^{\circ}\cdot\rangle$ is the strongest postcondition [4]. Note that A is not the same quantifier as $\boldsymbol{A}$ used in [4]. The former always evaluates at the top level of the semantics while the latter is the usual $\forall$.

REFERENCES

1. S. Bratus and A. Shubina, "Exploitation as Code Reuse: On the Need of Formalization," *Information Technology* **57**, 1–7 (2015).

2. G. Allwein, W. L. Harrison, and D. Andrews, "Simulation logic," *Logic and Logical Philosophy* **23**(3), 277–299 (2014).

3. G. Allwein and W. L. Harrison, "Distributed Modal Logic," in K. Bimbó, ed., *J. Michael Dunn on Information Based Logic: Outstanding Contributions to Logic*, pp. 331–362 (Springer-Verlag, 2016).

4. D. Gries, *The Science of Programming* (Springer-Verlag, 1981).