
ROPAS

Research On Program Analysis System
National Creative Research Initiative Center 1998-2003
Programming Research Laboratory, School of Computer Science & Engineering
Seoul National University

ROPAS MEMO
ROPAS-2006-33
July 27, 2006

Statically Detecting Uninitialized Array Element Usage in Perl Program

Seungho Han

School of Computer Science and Engineering
Seoul National University

Master thesis
Advisor: Kwangkeun Yi

Abstract

This thesis represent an idea of how to statically detect uninitialized array element usage in Perl programs. This is a common bug pattern that happens frequently in Perl programs. We try to detect sound alarms but we don't guarantee that our analysis can find all such bugs. To accomplish this goal we established a core Perl language which contains array features. In order to convince the analysis result we proved its correctness.

1 Introduction

Perl is a stable, cross platform programming language and open source software, which is created by Larry Wall. Perl is a very popular interpreter language and it takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.

Our goal is to detect uninitialized array element usage in Perl programs using static analysis technologies[1].

1.1 Problem

Perl is a popular programming language, which allows programmers to write programs in any style they want even if it isn't allowed in other programming languages. For example, C language programmers must annotate the size of array before using it, but Perl programmers can initialize an array any time they want and they can decide array keys and size at any time they want to use.

Authough, Perl's comfortable programming style seems acceptable and helpful , but this can also bring some defects. For instance, when a programmer accesses an uninitialized array

[1] CC77, CC79

element, Perl language will only manipulate error message silently without stoping it. but we know that this can cause a fatal logical bug.

In Perl language, among various bug patterns, accessing uninitialized array element is a common bug. It is because Perl language oddly allows programmers to initialize and assign new variables in an unodered sequence(Hash style). Which is totally different from other strict forms of languages, likewise C/C++/Java. These languages only allows array keys must be exist in ascending sequences.

Example 1 In the following example(Table 1), array @record get return values from $sth->fetchrow_array() function, but the return value isn't fully filled with. the third element contains **undef** value. In this situation Perl will emit an alarm at line 10.

Source: http://www.codecomments.com/archive211-2004-7-236273.html

```
1   #!/usr/bin/perl -w
2   use strict;
3   use DBI;
4
5   $ENV{"ORACLE_HOME"}="/orav101/oracle/8.0.6";
6   [snipet]
7   my @record=( );
8   while( @record= $sth->fetchrow_array( )) {
9     my $recordlist = ' ';
10    $recordlist=join(",",@record); # Problem here
11  } exit;
12  $dbh->disconnect;
```

Table 1.1

Return value: @record = (1, 2, undef, 4);. This kind bugs are very hard to detect before program runs, and only outbreak when they are executed.

Example 2 When write program in unstrict form(no warning/strict flag) then Perl will not report any alarm about accessing uninitialized array element usage bug. In this example(Table 2) array K has two keys is 0 and 2, but programmer make a slip to access key 1. The result is that Perl say nothing and continues its process uninterrupted.

```
1   #!/usr/bin/perl
2
3     K[0] = 10;
4     K[2] = 20;
5     ...
6     print "hello SNU\n";
7     print "K[1]";
8     print "hello Perl\n";
```

Table 1.2

Program output:

```
hello SNU
hello Perl
```

In Perl programming, as the code size increases a programmer or a team is almost impossible to recognize which keys are initialized keys and which keys are uninitialized ones. It happens mainly in team project which consist of many people. So, programmer can falsely access invalid array elements and they have to spend time to debug programs. This bugs can lead to many wrong executions which programmers never can imagine.

It is difficult to detect many bugs using traditional testing methods, which is also time consumming and needs work forces. However, these approach is widely used in many places. Consequently, this is the reason why software products are full of bugs and we don't have decisive solution to solve this problem.

1.2 Our Approach

It is excellent if there have some methods which can statically and automatically detect bugs before running program. If this is possible and reliable, which will greatly avoid dangers caused by invalid accesses, decrease testing time consumption and we can also get confidence from "No Error" message, which is usually impossible from normal tests.

1.2.1 How to Find Bugs

Although we don't find all possible bugs but we hope to find out as much real bugs as possible. There have many ways to dealing with array key analysis. our analysis automatically detects conservative errors about uninitialized array element usage in core Perl programs statically before running programs.

The key idea is that, we collect the array keys in program to an array key set and compares this to accessing array key set. For example, if accessing key set has no join point with array key set, then we will report true bug alarm.

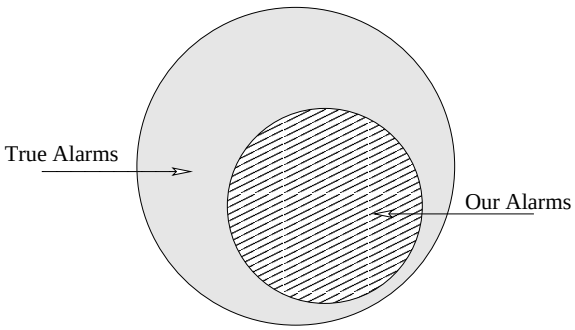

Figure 1: Alarm coverage

Our analysis detects sound bugs. In this thesis we collect array keys as a set, which we will represent as **Key Set**, **KS** for short. We join array values as a set either, which we will represent as **Value Set**, **VS** for short. If some KS joined after the join point, then we join **KS** and join their values as a set too.

1.2.2 Joining Rule

When two arrays $A_1 : \langle KS1, VS1\rangle$ and $A_2 : \langle KS2, VS2\rangle$ join at one point then the joining rule is like this.

- $|KS1| \cup |kS2| \leq 10 \wedge |VS1| \cup |VS2| \leq 10$

$$\langle KS_1, VS_2\rangle \sqcup \langle KS2, VS2\rangle = \langle KS_1 \cup KS_2, VS_1 \cup VS_2\rangle$$

- $|KS1| \cup |kS2| > 10 \vee |VS1| \cup |VS2| > 10$

$$\langle KS_1, VS_2\rangle \sqcup \langle KS2, VS2\rangle = \hat{\top}_{Array}$$

1.2.3 Case Examples

In this section we show an example of how does our analysis runs. For example, in Table 3, our analyzer will keep silent at line 10, because it access 4^{th} key and which is exists in key set.

```
1  x = readint;
2  if (x < 10)
3      a = &b;      # a = {b}, b is an array name
4  else
5      a = &c;      # a = {c}, c is an array name
6  end
7                   # a = {b, c}
8  b[2] = 20;       # b: <{2}, {20}>
9  c[3] = 30;       # c: <{3}, {30}>
10 print "*a[2]";   # *a: <{2,3}, {20,30}>
11 c[4] = 40;       # c: <{2,3,4}, {30,40}>
12 print "*a[5]";   # *a: <{2,3,4}, {20,30,40}> : Alarm
```

Table 1.3

1.3 Related work

There exist another Perl error checker program, called Lint.pm[2] . The B::Lint module is equivalent to an extended version of the -w option of perl. It is named after the program lint which carries out a similar process for C programs. However which can only detect syntax errors and can't detect semantic errors such as uninitalized array elements usage errors.

[2]http://search.cpan.org/ nwclark/perl-5.8.8/ext/B/B/Lint.pm

1.4 Thesis Summary

Chapter 2, introduces analysis framework and core Perl language on which our analysis works. In section 2.2, we will introduce collecting semantics domain and collecting semantics. In section 2.3, we will introduce abstract semantics domain and abstract semantics design. After that in section 2.4, we will show galois connection between two domains: collecting semantics domain and abstract semantics domain and proof its correctness. Chapter 3 shows analysis correctness proof. Which is separated by command proof and expression proof. In chapter 4, we will conclude this thesis, including summary, contributions and future work.

2 Analysis Design

Our analysis system is based on the framework of "Abstract Interpretation". Program's execution is defined by function $\mathcal{F} : D \to D$ in semantic domain D. Here, Semantic domain D is CPO(*complete partial order*), and semantic function $\mathcal{F}$ is continuous function:

$$\forall \texttt{chain}\ S \subseteq D : \mathcal{F}(\bigsqcup_{x \in S} x) = (\bigsqcup_{x \in S} \mathcal{F}(x))$$

Program's execution is defined by least fixed point of the function $\mathcal{F}$:

$$lfp\mathcal{F} = \bigsqcup_{i \in \mathbb{N}} \mathcal{F}^i(\bot)$$

We also define program's abstract execution, this means define semantic domain D's correspondence abstract domain $\hat{D}$, on which abstract semantic function $\hat{\mathcal{F}} : \hat{D} \to \hat{D}$ is defined. Here, abstract domain $\hat{D}$ is CPO and abstract semantic function $\hat{\mathcal{F}}$ is monotonic function.

In order to abstract program states in abstract framework, there must have abstracting function α abstract collecting semantic domain elements and concretizing function γ concretize abstract semantic domain elements. D and $\hat{D}$ must be galois connected: $D \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \hat{D}$. Furthermore, the concrete domain transition function $\mathcal{F}$ and abstract domain transition function $\hat{\mathcal{F}}$ must has following relation.

$$\begin{aligned} \alpha \circ \mathcal{F} &\sqsubseteq \hat{\mathcal{F}} \circ \alpha \\ \mathcal{F} \circ \gamma &\sqsubseteq \gamma \circ \hat{\mathcal{F}} \end{aligned}$$

2.1 Language Abstract Syntax

In order to analyze real Perl programs, we designed Perl's core part which consist of array handling expressions and some other simple conditional statements. Before analysis, we have to convert real Perl programs into our intermediate language.

In this abstract syntax definition, C and E represents commands and expressions respectively. In command syntax tree, assignment; sequence; **if** conditional command; **foreach** loop command; `while` loop command are the same as in Perl language. **pop** command selects

correspondent element from an array and assign to a variable. The **readint** expression read an integer from user and $@x$ means an empty array names x.

$$
\begin{array}{rcl}
C & \rightarrow & \texttt{skip} \\
 & | & x := E \mid *x := E \mid x[E] := E \mid *x[E] := E \\
 & | & C\ ;\ C \\
 & | & \texttt{if}\ E\ C\ C \\
 & | & \texttt{foreach}\ x\ E\ C \\
 & | & \texttt{While}\ E\ C \\
 & | & x := \texttt{pop}\ y\ E \mid x := \texttt{pop}\ *y\ E \\
E & \rightarrow & \texttt{readint} \mid n\ (n \in \mathbb{Z}) \\
 & | & E\ +\ E \mid -\ E \mid E\ <\ E \mid E\ =\ E \\
 & | & x \mid *x \mid \&x \mid @x \mid x[E] \mid *x[E]
\end{array}
$$

2.2 Collecting Semantics

2.2.1 Collecting Semantics Domain

In this domain memory *Mem* is the set of each state's memory m : $Mem = \{m_0, m_1, m_2, ..., m_n\}$ and each sub memories are defined as map of key to its value: $Var \rightarrow Val$. Value Val is consist of scalar value: *Scalar*, array value: *Array* and location: *Loc*. *Scalar* is singleton value and defined as $Z + Loc + \{\top\}$. Variable is consist of normal string characters as other programming languages mean, the special character $\top$ is used when we don't know the value at all. For instance, in **readint** expression. Location is the same as variable in this langauge used in point expressions. *Array* is the key point in this analysis, which is defined as a finite map $\mathbb{Z} \xrightarrow{fin} Scalar$. $\mathcal{C}$ is command C's semantic function. Which means when apply command and memory, then it will return new memory. $\mathcal{V}$ is expression semantic function. Which means when apply expression and memory, then it will return new memory.

$$
\begin{array}{rcccll}
m & \in & Mem & = & Var \rightarrow Val & (\texttt{Memory}) \\
v & \in & Val & = & Scalar + Array + Loc & (\texttt{Value}) \\
x, y & \in & Var & = & variable & (\texttt{Variable}) \\
s & \in & Scalar & = & \mathbb{Z} + Loc & (\texttt{Scalar value}) \\
l & \in & Loc & = & Var & (\texttt{Location}) \\
a, b & \in & Array & = & \mathbb{Z} \xrightarrow{\texttt{fin}} Scalar & (\texttt{Array}) \\
 & & \mathcal{C} & = & Cmd \rightarrow 2^{Mem} \rightarrow 2^{Mem} & (\texttt{Command function}) \\
 & & \mathcal{V} & = & Expr \rightarrow 2^{Mem} \rightarrow 2^{Val} & (\texttt{Expression function})
\end{array}
$$

2.2.2 Semantics

$$\mathcal{F} : (Cmd \rightarrow 2^{Mem} \rightarrow 2^{Mem}) \rightarrow (Cmd \rightarrow 2^{Mem} \rightarrow 2^{Mem})$$

$$\mathcal{V} : Exp \rightarrow 2^{Mem} \rightarrow 2^{Val}$$

Semantics operation & regulation

1. In assignment command, right hand side's value can not be an array it self.

2. In foreach command $\{cmd\}_k$ means repeat command *cmd* k times.

3. $a \dot{\mapsto} b$: Here a is set of variables and b is set of values. The operator $\dot{\mapsto}$ maps b to all elements in set a.

4. In **pop** command, $\uplus$ means, set union.
   For example, If set s is consist of two subset s_1 and s_2 then, $s = s_1 \uplus s_2$.

$$
\begin{array}{rcl}
\mathcal{F}\ C\ \texttt{skip}\ M & = & M \\
\mathcal{F}\ C\ x := E\ M & = & \{m\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in M, Array \notin (\mathcal{V}\ E\ \{m\})\} \\
\mathcal{F}\ C\ *x := E\ M & = & \{m\{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in M, Array \notin (\mathcal{V}\ E\ \{m\})\} \\
\mathcal{F}\ C\ x[E_1] := E_2\ M & = & \{m\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in M, \\
 & & Array \notin (\mathcal{V}\ E_2\ \{m\})\} \\
\mathcal{F}\ C\ *x[E_1] := E_2\ M & = & \{m\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} (\mathcal{V}\ E_2\ \{m\})]\} \mid \\
 & & m \in M, (\mathcal{V}\ E_2\ \{m\}) \notin Array\} \\
\mathcal{F}\ C\ C_1\ ;\ C_2\ M & = & C\ C_2\ (C\ C_1\ M) \\
\mathcal{F}\ C\ \texttt{if}\ E\ C_1\ C_2\ M & = & C\ C_1\ (\mathcal{B}\ E_1\ M) \cup C\ C_2\ (\neg\mathcal{B}\ E_1\ M) \\
\mathcal{F}\ C\ \texttt{foreach}\ x\ E\ C\ M & = & \{\{C\ C\ \{m_{i0}\{x \mapsto (\mathcal{V}\ E\ \{m_i\})[0]\} \mid m_{i0} \in m_k\}; \\
 & & \quad C\ C\ \{m_{i1}\{x \mapsto (\mathcal{V}\ E\ \{m_i\})[1]\} \mid m_{i1} \in m_k\}; \\
 & & \quad \ldots \\
 & & \quad C\ C\ \{m_{in}\{x \mapsto (\mathcal{V}\ E\ \{m_i\})[n]\} \mid m_{in} \in m_k\}\}_k \mid \\
 & & \quad (n = |\texttt{dom}(\mathcal{V}\ E\ \{m_i\})|,\ m_i \in m_k, m_k \in M, k = |\texttt{dom}(m_k)|) \\
 & & \} \\
\mathcal{F}\ C\ \texttt{while}\ E\ C\ M & = & \mathcal{B}\ \neg E(M \cup (C\ (\texttt{while}\ E\ C)\ (C\ C\ (\mathcal{B}\ E\ M)))) \\
\mathcal{F}\ C\ x := \texttt{pop}\ y\ E\ M & = & \{m\{x \mapsto s, y \mapsto z\} \mid s = m(y)[\mathcal{V}\ E\ \{m\}], \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in M\} \\
\mathcal{F}\ C\ x := \texttt{pop}\ *y\ E\ M & = & \{m\{x \mapsto s, m(y) \mapsto z\} \mid s = m(m(y))[\mathcal{V}\ E\ \{m\}], \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(m(y)),\ m \in M\} \\
\mathcal{B}\ E\ M & = & \cup\{M' \mid C\ E\ M' \not\ni 0, M' \subseteq M\} \\
\neg\mathcal{B}\ E\ M & = & \cup\{M' \mid C\ E\ M' \in \{0\}, M' \subseteq M\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{V}\ \texttt{readint}\ M & = & \{n\} \\
\mathcal{V}\ n\ M & = & \{n\} \\
\mathcal{V}\ E_1\texttt{+}E_2\ M & = & \{z_1 + z_2 \mid z_1 \in \mathcal{V}\ E_1\ M,\ z_2\ \in \mathcal{V}\ E_2\ M\} \\
\mathcal{V}\ \texttt{-}\ E\ M & = & \{-z \mid z \in \mathcal{V}\ E\ M\} \\
\mathcal{V}\ E_1\texttt{<}E_2\ M & = & \{z_1 < z_2 \mid z_1 \in \mathcal{V}\ E_1\ M,\ z_2\ \in \mathcal{V}\ E_2\ M\} \\
\mathcal{V}\ E_1\texttt{=}E_2\ M & = & \{z_1 = z_2 \mid z_1 \in \mathcal{V}\ E_1\ M,\ z_2\ \in \mathcal{V}\ E_2\ M\} \\
\mathcal{V}\ x\ M & = & \{m(x) \mid m \in M\} \\
\mathcal{V}\ \texttt{*}x\ M & = & \{m(m(x)) \mid m \in M\} \\
\mathcal{V}\ \texttt{\&}x\ M & = & \{x\} \\
\mathcal{V}\ \texttt{@}x\ M & = & x[] \\
\mathcal{V}\ x[E]\ M & = & \{m(x)[k] \mid m \in M, k \in \mathcal{V}\ E\ \{m\}\} \\
\mathcal{V}\ \texttt{*}x[E]\ M & = & \{m(m(x))[k] \mid m \in M, k \in \mathcal{V}\ E\ \{m\}\}
\end{array}
$$

2.3 Abstract Semantics

2.3.1 Abstract Domain

In abstract domain, $\hat{Mem}$ abstracted set of memories into a flat one. $\hat{Val}$ can be a $\hat{Scalar}$, $\hat{Array}$ or $\hat{Loc}$. $\hat{Scalar}$ is set of integer value: $2^{\mathbb{Z}}$, $\hat{Loc}$ or $\hat{\top}$. Since $\hat{Array}$ is the key point in this thesis, this is the most complexed and consist of 3 sets tuple: $2^{\mathbb{Z}} \times 2^{\mathbb{Z}} \times \{\hat{Scalar}\}$ or $2^{\mathbb{Z}} \times 2^{\mathbb{Z}} \times \{\hat{\top}\}$. The first power set is *DKS*; second power set if *PKS* and last one is set of values .$\hat{Loc}$ is set of variables.

$$
\begin{array}{rcrcll}
\hat{M} & \in & \hat{Mem} & = & Var \to \hat{Val} & (\texttt{Abstract memory}) \\
\hat{v} & \in & \hat{Val} & = & \hat{Scalar} + \hat{Array} + \hat{Loc} & (\texttt{Abstract value}) \\
\hat{s} & \in & \hat{Scalar} & = & \hat{\mathbb{Z}} + \hat{Loc} + \hat{\top} & (\texttt{Abstract scalar value}) \\
\hat{a} & \in & \hat{Array} & = & 2^{\mathbb{Z}}_{10} \times \hat{Scalar} + \hat{\top}_{Array} & (\texttt{Abstract array}) \\
\hat{l} & \in & \hat{Loc} & = & 2^{Var} & (\texttt{Abstract location}) \\
\hat{n} & \in & \hat{\mathbb{Z}} & = & 2^{\mathbb{Z}}_{10} + \hat{\top}_{\mathbb{Z}} & (\texttt{Abstract integer}) \\
 & & Var & = & Variable & (\texttt{Variable}) \\
 & & \hat{\mathcal{C}} & = & Cmd \to \hat{Mem} \to \hat{Mem} & (\texttt{Abstract command function}) \\
 & & \hat{\mathcal{V}} & = & Expr \to \hat{Mem} \to \hat{Val} & (\texttt{Abstract expression function})
\end{array}
$$

2.3.2 Semantics

$$\hat{\mathcal{F}} : (Cmd \to \hat{Mem} \to \hat{Mem}) \to (Cmd \to \hat{Mem} \to \hat{Mem})$$

$$\hat{\mathcal{V}} : Expr \to \hat{Mem} \to \hat{Val}$$

Semantic operation

[Caveat]: (Here, $KS : 2^{\mathbb{Z}}$ is a key set; $VS : \hat{Val}$ is value set)

- $\diamond$ (diamond) means, when abstracting array assignment, assigning key will be added to abstracted array of key set. Assigning value will be joined with respective array key. For example:

Array assignment $x[E_1] = E_2$,

We can abstract an array by $\hat{M}(x)$, and the result is a tuple of $\langle KS, VS \rangle$. the E_1 is the key set and can be evaluated by expression $\hat{\mathcal{V}}\ E_1\ \hat{M}$. We add this key set to KS, and the value $\hat{\mathcal{V}}\ E_2\ \hat{M}$ join with array $\hat{M}(x)$'s key.

Definition of $\diamond$: $\diamond \in \hat{Array} \times 2^{\mathbb{Z}} \times \hat{Val} \to \hat{Array}$

$a \diamond [b \mapsto c]$: (Here, a is an abstract array, b is an abstract key, c is an abstract value)

1. If $|KS \cup b| \leq 10$
$$\langle KS \cup b, VS \cup c \rangle$$
2. else if $|DKS \cup b| > 10$
$$\langle \top, \top, \top \rangle$$

$\hat{\cup}$ is a special join notation, which uses at the join point of the condition expression, $\hat{\cup}$'s function is defined by two operators: $\hat{\cup}_V, \cup$. Here, $\hat{\cup}_V$ is dealing with arrays, when the keys exist both on the definite key sets will be added to definite key set and rest keys are added to possible key sets. The values of joining arrays are join together. $\cup$ is a nomal union operator.
Type of $\hat{\cup}$: $\hat{Mem} \times \hat{Mem} \to \hat{Mem}$, $\hat{\cup}_V : \hat{Array} \times \hat{Array} \to \hat{Array}$

$$\hat{\cup}(m_1, m_2) = \begin{cases} \{x \mapsto v_1\ \hat{\cup}_V\ v_2\} & (v_1, v_2 \in \hat{Array}) \\ \{x \mapsto v_1 \cup v_2\} & (v_1, v_2 \in \hat{Scalar} + \hat{Loc}) \end{cases}$$

When two arrays A_1 and A_2 join at one point then the joining rule is like this, here I_1, I_2 and I_1', I_2' are KS, and value set respectively.

$$\langle I_1, I_2 \rangle\ \hat{\cup}_V\ \langle I_1', I_2' \rangle = \langle I_1 \cup I_1', I_2 \cup I_2' \rangle$$

$\pi_i(\hat{Array})$ is a loopup function. Since abstraction value of an array is consist of 2 sets, which are Key set and value set . and the π_1 selects KS, π_2 selects value set.
Type of $\pi_i : \hat{Array} \times \mathbb{Z} \to KS + VS$

$a \hat{\mapsto} b$: Here a is set of variables and b is set of values. The operator $\hat{\mapsto}$ maps b to all elements in set a.

$$
\begin{array}{rcl}
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,\texttt{skip}\,\hat{M} & = & \hat{M} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,x := E\,\hat{M} & = & \hat{M}\{x \mapsto \hat{\mathcal{V}}\,E\,\hat{M}\} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,*x := E\,\hat{M} & = & \hat{M}\{\hat{M}(x) \hat{\mapsto} \hat{\mathcal{V}}\,E\,\hat{M}\} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,x[E_1] := E_2\,M & = & \hat{M}\{x \mapsto \hat{M}(x) \diamond [(\hat{\mathcal{V}}\,E_1\,\hat{M}) \hat{\mapsto} (\hat{\mathcal{V}}\,E_2\,\hat{M})]\} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,*x[E_1] := E_2\,M & = & \hat{M}\{\hat{M}(x) \hat{\mapsto} \hat{M}(\hat{M}(x)) \diamond [(\hat{\mathcal{V}}\,E_1\,\hat{M}) \hat{\mapsto} (\hat{\mathcal{V}}\,E_2\,\hat{M})]\} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,C_1\,;\,C_2\,\hat{M} & = & \hat{\mathcal{C}}\,C_2\,(\hat{\mathcal{C}}\,C_1\,\hat{M}) \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,\texttt{if}\,E\,C_1\,C_2\,\hat{M} & = & (\hat{\mathcal{C}}\,C_1\,\hat{M}) \hat{\cup} (\hat{\mathcal{C}}\,C_2\,\hat{M}) \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,\texttt{foreach}\,x\,E\,C\,\hat{M} & = & \hat{\mathcal{C}}\,C\,\hat{M}\{x \mapsto (\hat{\mathcal{V}}\,E\,\hat{M})[0]\}; \\
 & & \hat{\mathcal{C}}\,C\,\hat{M}_1\{x \mapsto (\hat{\mathcal{V}}\,E\,\hat{M})[1]\}; \\
 & & \ldots \\
 & & \hat{\mathcal{C}}\,C\,\hat{M}_n\{x \mapsto (\hat{\mathcal{V}}\,E\,\hat{M})[n]\}; \\
 & & (n = \texttt{biggest value in}\ \pi_1(\hat{\mathcal{V}}\,E\,\hat{M})) \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,\texttt{while}\,E\,C\,\hat{M} & = & \hat{M} \sqcup (\hat{\mathcal{C}}\,\texttt{while}\,E\,C\,(\hat{\mathcal{C}}\,C\,\hat{M})) \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,x := \texttt{pop}\,y\,E\,\hat{M} & = & \hat{M}\{x \mapsto s \mid s = \pi_2(\hat{M}(y))\} \\
\hat{\mathcal{F}}\,\hat{\mathcal{C}}\,x := \texttt{pop}\,*y\,E\,\hat{M} & = & \hat{M}\{x \mapsto s \mid s = \pi_2(\hat{M}(\hat{M}(y)))\} \\
\hat{\mathcal{V}}\,\texttt{readint}\,\hat{M} & = & \top \\
\hat{\mathcal{V}}\,n\,\hat{M} & = & \alpha_V\{\mathcal{V}\,n\,M\} \\
\hat{\mathcal{V}}\,E_1 + E_2\,\hat{M} & = & (\hat{\mathcal{V}}\,E_1\,\hat{M}) \hat{+} (\hat{\mathcal{V}}\,E_2\,\hat{M}) \\
 & = & \alpha_V\{z_1 + z_2 \mid z_1 \in \gamma_V(\hat{\mathcal{V}}\,E_1\,\hat{M}),\ z_2 \in \gamma_V(\hat{\mathcal{V}}\,E_2\,\hat{M}\} \\
\hat{\mathcal{V}}\,-\,E\,\hat{M} & = & \hat{-} (\hat{\mathcal{V}}\,E\,\hat{M}) \\
 & = & \alpha_V\{-z \mid z \in \gamma_V(\hat{\mathcal{V}}\,E\,\hat{M})\} \\
\hat{\mathcal{V}}\,E_1\,<\,E_2\,\hat{M} & = & (\hat{\mathcal{V}}\,E_1\,\hat{M}) \hat{<} (\hat{\mathcal{V}}\,E_2\,\hat{M}) \\
 & = & \alpha_V\{z_1 < z_2 | z_1 \in \gamma_V(\hat{\mathcal{V}}\,E_1\,\hat{M}),\ z_2 \in \gamma_V(\hat{\mathcal{V}}\,E_2\,\hat{M})\} \\
\hat{\mathcal{V}}\,E_1\,=\,E_2\,\hat{M} & = & (\hat{\mathcal{V}}\,E_1\,\hat{M}) \hat{=} (\hat{\mathcal{V}}\,E_2\,\hat{M}) \\
 & = & \alpha_V\{z_1 = z_2 | z_1 \in \gamma_V(\hat{\mathcal{V}}\,E_1\,\hat{M}),\ z_2 \in \gamma_V(\hat{\mathcal{V}}\,E_2\,\hat{M})\} \\
\hat{\mathcal{V}}\,x\,\hat{M} & = & \alpha_V\{m(x) \mid m \in \gamma_M\hat{M}\} \\
\hat{\mathcal{V}}\,*x\,\hat{M} & = & \alpha_V\{m(m(x)) \mid m \in \gamma_M\hat{M}\} \\
\hat{\mathcal{V}}\,\&x\,\hat{M} & = & \alpha_V\{x\} \\
\hat{\mathcal{V}}\,x[E]\,\hat{M} & = & \alpha_V\{m(x)[\mathcal{V}\,E\,m] \mid m \in \gamma_M\hat{M}\} \\
\hat{\mathcal{V}}\,*x[E]\,\hat{M} & = & \alpha_V\{m(m(x))[\mathcal{V}\,E\,m] \mid m \in \gamma_M\hat{M}\}
\end{array}
$$

2.4 Galois Connection

Here, we define galois connection between collecting semantic domain and abstract semantic domain.

1. Abstract value:

$$2^{Val} \underset{\alpha_V}{\overset{\gamma_V}{\leftrightarrows}} \hat{Val}$$

is

$$\alpha_V(X) = \begin{cases} X & X \in 2^{\mathbb{Z}} \text{ and } |X| \leq 10 \\ \hat{\top}_{\mathbb{Z}} & X \in 2^{\mathbb{Z}} \text{ and } |X| > 10 \\ X & X \in 2^{Loc} \\ \langle KS, VS \rangle & (|KS| \leq 10 \text{ and } |VS| \leq 10) \\ & KS = \bigcup\{\mathtt{dom}(a) \mid a \in X\} \\ & VS = \bigcup\{\mathtt{range}(a) \mid a \in X\} \\ \hat{\top}_{Array} & (|KS| > 10 \text{ or } |VS| > 10) \\ \hat{\top} & o.w \end{cases}$$

and correspondent γ_V is

$$\gamma_V(X) = \begin{cases} X & X \in 2^{\mathbb{Z}} \text{ and } |X| \leq 10 \\ 2^{\mathbb{Z}} & X \in 2^{\mathbb{Z}} \text{ and } |X| > 10 \\ X & X \in 2^{Loc} \\ \{k \mapsto VS \mid k \in KS\} & (|KS| \leq 10 \text{ and } |VS| \leq 10) \\ \{k \mapsto \top \mid k \in \mathbb{Z}\} & X \in \hat{\top}_{Array} \\ \top & \hat{\top} \end{cases}$$

2. Semantics function C and $\hat{C}$ has following galois connection:

$$C = (Cmd \to 2^{Mem} \to 2^{Mem}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (Cmd \to \hat{Mem} \to \hat{Mem}) = \hat{C}$$

is connected by its components

$$Cmd \underset{id}{\overset{id}{\leftrightarrows}} Cmd \quad Var \underset{id}{\overset{id}{\leftrightarrows}} Var \quad 2^{Val} \underset{\alpha_V}{\overset{\gamma_V}{\leftrightarrows}} \hat{Val}$$

and defined by following relation

$$Mem = 2^{Var \overset{fin}{\to} Val} \underset{\alpha_M}{\overset{\gamma_M}{\leftrightarrows}} Var \overset{fin}{\to} \hat{Val} = \hat{Mem}$$

In order to define α_M, we have to devide the relation in two steps: α_1, α_2

$$Mem \overset{\alpha_1}{\to} Mem' \overset{\alpha_2}{\to} \hat{Mem} : \begin{cases} Mem = 2^{Var \overset{fin}{\to} Val} \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} Var \overset{fin}{\to} 2^{Val} = Mem' \\ Mem' = Var \overset{fin}{\to} 2^{Val} \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} Var \overset{fin}{\to} \hat{Val} = \hat{Mem} \end{cases}$$

here, α_1 is abstracting set of memories into a flattern memory (See 4.3.1),

$$\alpha_1(M) = \lambda x.\, \{m(x) \mid m \in M, x \in \mathtt{Dom}(m)\}$$

γ_1 concritise flattened memory to all product memories.

$$(A \texttt{ is set of value, } i \texttt{ is variable of the abstract memory})$$

$$Let, \begin{cases} A = \bigcup_{i \in I} A_i \\ \prod_{i \in I} A_i = \{f \mid f : I \to A \texttt{ is a function, and } f(i) \in A_i, \forall i \in I \end{cases}$$

$$\therefore \gamma_1(m') = \prod_{x \in \mathtt{dom}(m')} \gamma_V(m'(x))$$

$$Var \to 2^{Val} \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} Var \to \hat{Val}$$
$$\because \alpha_2(m) = \alpha_V \circ m \circ \gamma_{Var}$$

Then, from above equations, α_M is

$$\therefore \alpha_M(M) = \alpha_2 \circ \alpha_1 \circ M$$

and relation between C C and $\hat{C}$ $C$is:

$$Mem \to Mem \underset{\alpha_C}{\overset{\gamma_C}{\leftrightarrows}} \hat{Mem} \to \hat{Mem}$$

from this equation, α_C is

$$\alpha_C(x) = \alpha_M \circ x \circ \gamma_M$$

use α_M and α_C, we can conclude α is

$$\alpha(C) = \alpha_C \circ C \circ \gamma_{Cmd} = \alpha_C \circ C$$

2.4.1 Memory Abstraction Example

In collecting semantics, total memory Mem is consist of set of sub memories: $2^{Var \overset{fin}{\to} Val}$, like following example. However, in abstraction level, memory should be flattened to a single memory. i.e. same variable name in each sub memories are joined together in abstraction level. In this case, we will lose relations between variables their respective values.

[Memory abstraction example]:

$$\begin{aligned} Mem = \quad & \{ \\ & \{x \mapsto 1, y \mapsto 2\}, \\ & \{x \mapsto 2, y \mapsto 3, z \mapsto 8\}, \\ & \cdots, \\ & \{x \mapsto 4, y \mapsto 6\} \\ & \} \end{aligned}$$

After the first abstraction α_1, we'll get the following memory structure:
$Mem' = \{x \mapsto \{1, 2, \cdots, 4\}, y \mapsto \{2, 3, \cdots, 6\}, z \mapsto \{8\}\}$

Then, we will abstract second time using α_2, in this example which is actually an identity function in our analysis. So,

$$Mem' = \hat{Mem}$$

$$\hat{Mem} = \{x \mapsto \{1, 2, \cdots, 4\}, y \mapsto \{2, 3, \cdots, 6\}, z \mapsto \{8\}\}$$

□

2.4.2 Prove Galois Connection

Theorem 1 **If a function is composed by safety functions, then the function is safe too.** i.e. $A \underset{\alpha_A}{\overset{\gamma_A}{\leftrightarrows}} \hat{A}$, $B \underset{\alpha_B}{\overset{\gamma_B}{\leftrightarrows}} \hat{B}$, $C \underset{\alpha_C}{\overset{\gamma_C}{\leftrightarrows}} \hat{C}$ and monotone function $f : A \to B$, $\hat{f} : \hat{A} \to \hat{B}$, $g : B \to C$, $\hat{g} : \hat{B} \to \hat{C}$ has following relations:

$$\alpha_B \circ f \sqsubseteq \hat{f} \circ \alpha_A, \text{ and } \alpha_C \circ g \sqsubseteq \hat{g} \circ \alpha_B$$

then,

$$\alpha_C \circ (g \circ f) \sqsubseteq (\hat{g} \circ \hat{f}) \circ \alpha_A$$

is hold.

Proof – **Prove value abstraction relation is correct**

To Show

$$\forall x, \hat{x}.\ \alpha_V(x) \sqsubseteq \hat{x} \Leftrightarrow x \sqsubseteq \gamma_V(\hat{x})$$

We will proof case by case,

1. Case of $x \in 2^{\mathbb{Z}}$, $x \in 2^{Loc}$, we know from the galois connection definition that α_V and γ_V are identity functions. So,

   $$\alpha_V(x) \sqsubseteq \hat{x} \Leftrightarrow x \sqsubseteq \gamma_V(\hat{x})$$

   is hold.

2. Case of $x \in 2^{Array}$,
   (a) If size of DKS is ≤ 10
   From γ_V's definition, we can easily know that

   $$\{\{k \mapsto VS \mid k \in (DKS \cup p)\} \mid p \in 2^{PKS}\} \sqsubseteq$$
   $$\{\{k \mapsto VS \mid k \in (DKS \cup p \cup p')\} \mid p \in 2^{PKS}, p' \in 2^{\mathbb{Z}}\}$$

   So, we know that as the key set grows γ_V covers more. $\therefore \gamma_V$ is monotone function.

$\Rightarrow$ If $\alpha_V(x) \sqsubseteq \hat{x}$, then we can conclude that

$$\alpha_V(x) = \langle d, p, v\rangle \sqsubseteq \langle \hat{d}, \hat{p}, \hat{v}\rangle = \hat{x}$$

is hold.

To Show: whether $x \sqsubseteq \gamma_V(\hat{x})$ is hold.

We will show whether $\gamma_V(\hat{x}) = \gamma_V(\langle \hat{d}, \hat{p}, \hat{v}\rangle) \sqsupseteq \gamma_V(\langle d, p, v\rangle) \sqsupseteq x$ is hold.

$$\because \gamma_V(\langle d, p, v\rangle) \quad = \quad \{k \mapsto VS \mid k \in (DKS \cup p)\} \mid p \in 2^{PKS}\} \quad (\texttt{by def of } \gamma_V)$$

we define x as the set of arrays, and can be defined as $\{a \mid a \in x\}$. Single array a is defined as $\{k \mapsto v \mid k \in \texttt{dom}(a), v \in \texttt{codom}(a)\}$. As defined each array a must contains keys of DKS and may contains other keys(we name it P_{key}).

Since the P_{key} of an array are the belongs to PKS, as defined. So, each of them are the subset of 2^{PKS}.

$\therefore \gamma_V(\hat{x})$ covers set of array x.

$x \sqsubseteq \gamma_V(\hat{x})$ is hold.

$\Leftarrow$ If $x \in \gamma_V(\hat{x})$, then we can conclude that

$$\gamma_V(\hat{x}) = \gamma_V(\langle \hat{d}, \hat{p}, \hat{v}\rangle) \sqsupseteq x$$

is hold.

To Show: whether $\alpha_V(x) \sqsubseteq \hat{x}$

We will show whether $\alpha_V(x) = \langle d, p, v\rangle \sqsubseteq \langle \hat{d}, \hat{p}, \hat{v}\rangle = \hat{x}$

$$\texttt{Let say } \alpha_V(x) = \langle d, p, v\rangle$$

Since d and $\hat{d}$ both hold DKS, they are same. From hypothesis, we can conclude that the concritized array set covers original array set x. So, $p \sqsubseteq \hat{p}$. Also, from γ_V's definition, concritized array element's value are set of all original arrays' values. So, $v \sqsubseteq \hat{v}$

$\therefore d = \hat{d}, p \sqsubseteq \hat{p}, v \sqsubseteq \hat{v}$ is hold.

So, we can conclude that $\alpha_V(x) \sqsubseteq \hat{x}$

DONE; $\square$

(b) If size of DKS is > 10

$\Rightarrow$ If $\alpha_V(x) \sqsubseteq \hat{\top}_{Array}$

To Show: $x \sqsubseteq \gamma_V(\hat{\top}_{Array})$

From definition of γ_V,

$$\gamma_V(\hat{\top}_{Array}) = \{\{k \mapsto \top \mid k \in p\} \mid p \in 2^{\mathbb{Z}}\}$$

Which defines all the array sets in the system can define. Of cause covers original array x.

$\therefore x \sqsubseteq \gamma_V(\hat{\top}_{Array})$

$\Leftarrow$ If $x \sqsubseteq \gamma_V(\hat{\top}_{Array})$

To Show: $\alpha_V(x) \sqsubseteq \hat{\top}_{Array}$

From definition of α_V,

$$\alpha_V(x) = \hat{\top}_{Array}$$

$\therefore \alpha_V(x) \sqsubseteq \hat{\top}_{Array}$

(c) Case of $x \in \top$,

If $\alpha_V(\top) \sqsubseteq \hat{x}$, then, to show $\top \sqsubseteq \gamma_V(\hat{x})$

But there exists only one element that matches this situation, it must be $\hat{x} = \hat{\top}$.

If $\gamma_V(\hat{\top}) \sqsubseteq x$, then, to show $\alpha_V(x) \sqsubseteq \hat{\top}$

But there exists only one element that matches this situation, it must be $x = \top$.

□

So, galois connection between $2^{Val} \underset{\alpha_V}{\overset{\gamma_V}{\leftrightarrows}} \hat{Val}$ is hold.

– Galois connection between *Cmd* is,

$$Cmd \underset{id}{\overset{id}{\leftrightarrows}} Cmd$$

which is an identity relation. So its galois connected.

– Galois connection between *Var* is,

$$Var \underset{id}{\overset{id}{\leftrightarrows}} Var$$

which is an identity relation. So its galois connected.

– Galois connection between *Mem* and $\hat{Mem}$ is

$$Mem = 2^{Var \overset{fin}{\to} Val} \underset{\alpha_M}{\overset{\gamma_M}{\leftrightarrows}} Var \overset{fin}{\to} \hat{Val} = \hat{Mem}$$

and α_M can be reduced into two steps: α_1, α_2

$$Mem \overset{\alpha_1}{\to} Mem' \overset{\alpha_2}{\to} \hat{Mem} : \begin{cases} Mem = 2^{Var \overset{fin}{\to} Val} \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} Var \overset{fin}{\to} 2^{Val} = Mem' \\ Mem' = Var \overset{fin}{\to} 2^{Val} \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} Var \overset{fin}{\to} \hat{Val} = \hat{Mem} \end{cases}$$

Let's see if relation between *Mem* and *Mem'* is galois connected.

$\Rightarrow$ If $\alpha_1(M) \sqsubseteq \hat{M}$,

To Show: $M \sqsubseteq \gamma_1(\hat{M})$ is hold. from γ_1's definition, we can conclude that

$$\gamma_1(M') = \prod_{x \in \mathtt{dom}(M')} \gamma_V(M'(x))$$

As we know, $\hat{M}$ is greater than abstracted memory. This equation will produce set of memories by the rule of $\prod$, in chapter 4.3. The result is definitely greater than original memory *Mem*. So, it hold.

$\Leftarrow$ If $\gamma_1(\hat{M}) \sqsupseteq M$, there must be more sub memories than original ones. By the following equation,
To Show: $\alpha_1(M) \sqsubseteq \hat{M}$,

$$\alpha_1(M) = \lambda x.\{m(x) \mid m \in M, x \in \mathtt{Dom}(m)\}$$

result is grows as the greater input comes. Which abstract all the same variables into one flattened set, nomatter how many, if they are piled up one on another. And we don't lost any of the variable and its value. Because M doesn't have the variables which $\hat{M}$ has. So, $\alpha_1(M) \sqsubseteq \hat{M}$ must be true.

$\therefore$ galois connection in this case is hold.

– Relation between *Mem*′ and $\hat{Mem}$, 2^{Val} and $\hat{Val}$, C and $\hat{C}$ are function composited, but the composed functions are all galois connected.

$$Var \to 2^{Val} \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} Var \to \hat{Val}$$

$$Var \to 2^{Var} \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} Var \to \hat{Val}$$

$$Mem \to Mem \underset{\alpha_C}{\overset{\gamma_C}{\leftrightarrows}} \hat{Mem} \to \hat{Mem}$$

From **theory 1**, we can conclude that, all these relations are galois connected

$\star$ From this we can conclude that collecting semantics function C and abstract selamtics function $\hat{C}$ is galois connected. Proof DONE;

□

3 Correctness Proof

In this chapter, we prove the correctness of our analysis system. Which gives convince to us that the analysis result is correct.

3.1 Proposition

In order to use in the proof, we define following two propositions.

Proposition 1 Let $f \in A \to B$ is monotonic function, the galois connectin between two elements A and $\hat{A}$, B and $\hat{B}$ is $A \underset{\alpha_A}{\overset{\gamma_A}{\leftrightarrows}} \hat{A}$, and $B \underset{\alpha_B}{\overset{\gamma_B}{\leftrightarrows}} \hat{B}$,
then we can conclude:

$$\alpha_B(f\ a) \sqsubseteq (\alpha_{A\to B}\ f)(\alpha_A\ a)$$

because,

$$\begin{array}{rcl} \alpha_B(f\ a) & \sqsubseteq & \alpha_B(f\ (\gamma_A(\alpha_A\ a))) \\ & = & (\alpha_B \circ f \circ \gamma_A)(\alpha_A\ a) \\ & \sqsubseteq & (\alpha_{A\to B}\ f)(\alpha_A\ a) \end{array}$$

□

Proposition 2

If

$$\hat{\mathcal{V}}\ E \sqsupseteq \alpha(\mathcal{V}\ E)$$

is true for all E, then following proposition is true.

$$\hat{\mathcal{V}}\ E\ \hat{M} \sqsupseteq \alpha_V(\mathcal{V}\ E\ (\gamma_M \hat{M}))$$

Because, as as defined in the galois connection, we know that $\hat{\mathcal{V}}\ E$ is a monotone function. So, we can get following relation easily.

$$\begin{array}{rcl} \hat{\mathcal{V}}\ E\ \hat{M} & \sqsupseteq & \alpha(\mathcal{V}\ E)\ \hat{M} \\ & = & (\alpha_V \circ \mathcal{V}\ E \circ \gamma_M)\ \hat{M} \quad (\texttt{by}\ \alpha\texttt{'s def.}) \\ & = & \alpha_V(\mathcal{V}\ E(\gamma_M\ \hat{M})) \end{array}$$

□

3.2 Proof

Correctness proof is accomplished in two steps: command proof(3.2.1) and expression proof(3.2.2).

3.2.1 Command C

$$\begin{array}{rcl} \mathcal{F} & \in & (Cmd \to 2^{Mem} \to 2^{Mem}) \to (Cmd \to 2^{Mem} \to 2^{Mem}) \\ \hat{\mathcal{F}} & \in & (Cmd \to \hat{Mem} \to \hat{Mem}) \to (Cmd \to \hat{Mem} \to \hat{Mem}) \end{array}$$

To prove, $lfp\hat{\mathcal{F}}$ is $lfp\mathcal{F}$'s safe abstract, we have to show following is hold.

$$\alpha \circ \mathcal{F} \sqsubseteq \hat{\mathcal{F}} \circ \alpha$$

i.e., for all program C, show:

$$\alpha\ (\mathcal{F}\ C)\ C \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ C$$

[Semantics operation]

1. $S_1 \Cup S_2$: Means, join set S_1 and set S_2, but set S_1 is a set except set S_2.
2. $a \hat{\mapsto} b$: Here a is set of variables and b is set of values. The operator $\hat{\mapsto}$ maps b to all elements in set a.
3. $as \otimes [ks \hat{\mapsto} vs]$: Here as is array set, ks is key set, vs is value set. the $\otimes$ operator updates every array in as with $[ks \hat{\mapsto} vs]$.

4. $\diamond$: Defined in chapter 2.3.2.

- $C \rightarrow \texttt{skip}$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ \texttt{skip}\ \hat{M} & \\
rhs & = & \alpha_C \circ\ (\mathcal{F}\ C)\ \texttt{skip}\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
 & = & (\alpha_C(\mathcal{F}\ C)\ \texttt{skip})\ \hat{M} & \\
 & = & ((\alpha_M \circ (\mathcal{F}\ C)\ \texttt{skip}) \circ\ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha_C) \\
 & = & \alpha_M(\mathcal{F}\ C)\ \texttt{skip}\ (\gamma_M\ \hat{M}) & \\
 & \sqsubseteq & \alpha_M(\gamma_M\ \hat{M}) & (\texttt{by def. of}\ \mathcal{F}) \\
 & \sqsubseteq & \hat{M} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & = & (\hat{\mathcal{F}}\ \hat{C})\ \texttt{skip}\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ \texttt{skip}\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ \texttt{skip}\ \hat{M}$

- $C \rightarrow x := E$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ x := E\ \hat{M} & \\
rhs & = & \alpha_C \circ\ (\mathcal{F}\ C)\ x := E\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ x := E))\ \hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ x := E) \circ\ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ x := E\ (\gamma_M\ \hat{M})) & \\
 & = & \alpha_M(\{m\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\}) & (\texttt{by def. of}\ \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\} & (\texttt{by def. of}\ \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\} & \\
 & = & \alpha_2\alpha_1\{m' \uplus \{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by}\ \uplus) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\alpha_1\{\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is cont.}) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\alpha_1\{\{x \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is cont.}) \\
 & = & \alpha_2\alpha_1\{\gamma_M\hat{M}\}\ \uplus \alpha_2\{x \mapsto vs \mid & \\
 & & vs = \bigsqcup\{\mathcal{V}\ E\ \{m\} \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \alpha_1) \\
 & \sqsubseteq & \hat{M}\ \uplus \alpha_2\{x \mapsto vs \mid & \\
 & & vs = \bigsqcup\{\mathcal{V}\ E\ \{m\} \mid m \in \gamma_M\hat{M}\}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & = & \hat{M}\ \uplus\ \{x \mapsto \alpha_V(vs) \mid & \\
 & & vs = \bigsqcup\{\mathcal{V}\ E\ \{m\} \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \alpha_2) \\
 & = & \hat{M}\ \uplus\ \{x \mapsto \alpha_V(\mathcal{V}\ E\ \gamma_M\hat{M})\} & \\
 & \sqsubseteq & \hat{M}\ \uplus\ \{x \mapsto \hat{\mathcal{V}}\ E\ \hat{M}\} & (\texttt{by Prop.2}) \\
 & = & \hat{M}\{x \mapsto \hat{\mathcal{V}}\ E\ \hat{M}\}) & \\
 & = & (\hat{\mathcal{F}}\ \hat{C})\ x := E\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ x := E\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ x := E\ \hat{M}$

- $C \to *x := E$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ *x := E\ \hat{M} & \\
rhs & = & \alpha_C \circ\ (\mathcal{F}\ C)\ *x := E\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ *x := E))\ \hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ *x := E) \circ\ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ *x := E\ (\gamma_M\ \hat{M})) & \\
 & = & \alpha_M(\{m\{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\}) & (\texttt{by def. of}\ \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\} & (\texttt{by def. of}\ \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in (\gamma_M\ \hat{M})\} & \\
 & = & \alpha_2\alpha_1\{m' \uplus \{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by}\ \uplus) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\alpha_1\{\{m(x) \mapsto \mathcal{V}\ E\ \{m\}\} \mid m \in \gamma_M\hat{M}\} & (\uplus\ \texttt{is continuous}) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\{m(x) \mapsto vs \mid m \in \gamma_M\hat{M}\} & (\texttt{by}\ \alpha_1) \\
 & & (vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}) & \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\{m(x) \mapsto vs \mid m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is monotone}) \\
 & & (vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}) & \\
 & = & \alpha_2\alpha_1\{\gamma_M\hat{M}\}\ \uplus\ \alpha_2\{ls \dot{\mapsto} vs \mid & \\
 & & vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\} & \\
 & & ls = \{m(x) \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \dot{\mapsto}) \\
 & \sqsubseteq & \hat{M}\ \uplus\ \alpha_2\{ls \dot{\mapsto} vs \mid & \\
 & & vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\} & \\
 & & ls = \{m(x) \mid m \in \gamma_M\hat{M}\}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & = & \hat{M}\ \uplus\ \{\alpha_V(ls) \dot{\mapsto} \alpha_V(vs) \mid & \\
 & & vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\} & \\
 & & ls = \{m(x) \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \alpha_2) \\
 & = & \hat{M}\ \uplus\ \{ls \dot{\mapsto} \alpha_V(vs) \mid & \\
 & & vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\} & \\
 & & ls = \{m(x) \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \alpha_V) \\
 & = & \hat{M}\ \uplus\ \{ls \dot{\mapsto} \alpha_V(\mathcal{V}\ E\ \gamma_M\hat{M}) \mid & \\
 & & vs = \bigsqcup\{v \mid v = \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\} & \\
 & & ls = \{m(x) \mid m \in \gamma_M\hat{M}\}\} & \\
 & = & \hat{M}\ \uplus\ \{\hat{M}(x) \dot{\mapsto} \alpha_V(\mathcal{V}\ E\ \gamma_M\hat{M})\} & (ls = \hat{M}(x)) \\
 & \sqsubseteq & \hat{M}\ \uplus\ \{\hat{M}(x) \dot{\mapsto} \hat{\mathcal{V}}\ E\ \hat{M}\} & (\texttt{by Prop.2}) \\
 & = & \hat{M}\{\hat{M}(x) \dot{\mapsto} \hat{\mathcal{V}}\ E\ \hat{M}\} & \\
 & = & (\hat{\mathcal{F}}\ \hat{C})\ *x := E\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ *x := E\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ *x := E\ \hat{M}$

- $C \rightarrow x[E_1] := E_2$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ x[E_1] := E_2\ \hat{M} & \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ x[E_1] := E_2\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ x[E_1] := E_2))\ \hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ x[E_1] := E_2) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of } \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ x[E_1] := E_2\ (\gamma_M\ \hat{M})) & \\
 & = & \alpha_M\{m\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & (\texttt{by def. of } \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & (\texttt{by def. of } \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & \\
 & = & \alpha_2\alpha_1\{m' \Cup \{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) & \\
 & & \dot{\mapsto} (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & (\texttt{by } \Cup) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M \hat{M}\} \Cup & \\
 & & \alpha_2\alpha_1\{\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & (\alpha_2\alpha_1 \texttt{ is continuous}) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M \hat{M}\} \Cup & \\
 & & \alpha_2\alpha_1\{\{x \mapsto m(x)[(\mathcal{V}\ E_1\ \{m\}) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M \hat{M}\} & (\alpha_2\alpha_1 \texttt{ is monotone}) \\
 & = & \alpha_2\alpha_1\{\gamma_M \hat{M}\} \Cup & \\
 & & \alpha_2\{x \mapsto \{m(x)[(\mathcal{V}\ E_1\ m) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ m)] \mid m \in \gamma_M \hat{M}\}\} & (\texttt{by } \alpha_1) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M \hat{M}\} \ \Cup\ \alpha_2\{x \mapsto as \otimes [ks \dot{\mapsto} vs] \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(x)\}, m \in \gamma_M \hat{M}\} & (\alpha_2 \texttt{ is mono. \& } \otimes) \\
 & \sqsubseteq & \hat{M} \ \Cup\ \alpha_2\{x \mapsto as \otimes [ks \dot{\mapsto} vs] \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(x)\}, m \in \gamma_M \hat{M}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & \sqsubseteq & \hat{M} \ \Cup\ \{x \mapsto \alpha_V(as \otimes [ks \dot{\mapsto} vs]) \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(x)\}, m \in \gamma_M \hat{M}\} & (\texttt{by } \alpha_2) \\
 & = & \hat{M} \ \Cup\ \{x \mapsto \hat{M}(x) \diamond [\hat{\mathcal{V}}\ E_1\ \hat{M}) \mapsto & \\
 & & (\hat{\mathcal{V}}\ E_2\ \hat{M})]\} & (\texttt{by } \diamond) \\
 & \sqsubseteq & \hat{M}\{x \mapsto \hat{M}(x) \diamond [(\hat{\mathcal{V}}\ E_1\ \hat{M}) \mapsto (\hat{\mathcal{V}}\ E_2\ \hat{M})]\} & \\
 & = & (\hat{\mathcal{F}}\ \hat{C})\ x[E_1] := E_2\ \hat{M} & (\texttt{by def. of } \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ x[E_1] := E_2\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ x[E_1] := E_2\ \hat{M}$

- $C \to *x[E_1] := E_2$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ *x[E_1] := E_2\ \hat{M} & \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ *x[E_1] := E_2\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ *x[E_1] := E_2))\ \hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ *x[E_1] := E_2) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of } \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ *x[E_1] := E_2\ (\gamma_M\ \hat{M})) & \\
 & = & \alpha_M\{m\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) \mapsto & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by def. of } \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) & \\
 & & \dot{\mapsto}\ (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by def. of } \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) \mapsto & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & \\
 & = & \alpha_2\alpha_1\{m' \uplus \{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) & \\
 & & \dot{\mapsto}\ (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by } \uplus) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\alpha_1\{\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) \mapsto & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is continuous}) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\alpha_1\{\{m(x) \mapsto m(m(x))[(\mathcal{V}\ E_1\ \{m\}) \mapsto & \\
 & & (\mathcal{V}\ E_2\ \{m\})]\} \mid m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is monotone}) \\
 & = & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \uplus & \\
 & & \alpha_2\{\{m(x) \mapsto \{m(m(x))[(\mathcal{V}\ E_1\ m) \dot{\mapsto} & \\
 & & (\mathcal{V}\ E_2\ m)] \mid m \in \gamma_M\hat{M}\}\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by } \alpha_1) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \uplus\ \alpha_2\{\{m(x) \mapsto as \otimes [ks \dot{\mapsto} vs] \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(m(x)) \mid m \in \gamma_M\hat{M}\} \mid m \in \gamma_M\hat{M}\} & (\alpha_2\ \texttt{is mono. \& } \otimes) \\
 & \sqsubseteq & \hat{M} \uplus\ \alpha_2\{\{m(x) \mapsto as \otimes [ks \dot{\mapsto} vs] \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(m(x)) \mid m \in \gamma_M\hat{M}\} \mid m \in \gamma_M\hat{M}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & \sqsubseteq & \hat{M} \uplus\ \alpha_2\{\{ls \dot{\mapsto} as \otimes [ks \dot{\mapsto} vs] \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(m(x)) \mid m \in \gamma_M\hat{M}\}, & \\
 & & ls = \bigsqcup\{m(x) \mid m \in \gamma_M\hat{M}\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by } \dot{\mapsto}) \\
 & \sqsubseteq & \hat{M} \uplus\ \{\{\alpha_V(ls) \dot{\mapsto} \alpha_V(as \otimes [ks \dot{\mapsto} vs]) \mid & \\
 & & ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, & \\
 & & vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, & \\
 & & as = \bigsqcup\{m(m(x)) \mid m \in \gamma_M\hat{M}\}, & \\
 & & ls = \bigsqcup\{m(x) \mid m \in \gamma_M\hat{M}\} \mid m \in \gamma_M\hat{M}\} & (\texttt{by } \alpha_2)
\end{array}
$$

$$
\begin{array}{lll}
\sqsubseteq & \hat{M} \Cup \{\{ls \mapsto \alpha_V(as \otimes [ks \mapsto vs]) \mid \\
& ks = \bigsqcup\{v_1 \mid v_1 \in \mathcal{V}\ E_1\ \{m\}\}, \\
& vs = \bigsqcup\{v_2 \mid v_2 \in \mathcal{V}\ E_2\ \{m\}\}, \\
& as = \bigsqcup\{m(m(x)) \mid m \in \gamma_M \hat{M}\}, \\
& ls = \bigsqcup\{m(x) \mid m \in \gamma_M \hat{M}\} \mid m \in \gamma_M \hat{M}\} & (\texttt{by}\ \alpha_V) \\
= & \hat{M} \Cup \{\hat{M}(x) \mapsto \hat{M}(\hat{M}(x)) \diamond [\hat{\mathcal{V}}\ E_1\ \hat{M}) \mapsto \\
& (\hat{\mathcal{V}}\ E_2\ \hat{M})]\} & (\texttt{by}\ \diamond) \\
\sqsubseteq & \hat{M}\{\hat{M}(x) \mapsto \hat{M}(\hat{M}(x)) \diamond [(\hat{\mathcal{V}}\ E\ \hat{M}) \mapsto (\hat{\mathcal{V}}\ E\ \hat{M})]\} & \\
= & (\hat{\mathcal{F}}\ \hat{\mathcal{C}})\ x[E_1] := E_2\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ \mathcal{C})\ *x[E_1] := E_2\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{\mathcal{C}})\ *x[E_1] := E_2\ \hat{M}$

- $C \to C_1\ ;\ C_2$

$$
\begin{array}{lll}
lhs & = & \alpha\ (\mathcal{F}\ \mathcal{C})\ C_1\ ;C_2\ \hat{M} \\
rhs & = & \alpha_C \circ (\mathcal{F}\ \mathcal{C})\ C_1;C_2\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
& = & (\alpha_C((\mathcal{F}\ \mathcal{C})\ C_1;C_2))\ \hat{M} & \\
& = & (\alpha_M \circ ((\mathcal{F}\ \mathcal{C})\ C_1;C_2) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha_C) \\
& = & \alpha_M((\mathcal{F}\ \mathcal{C})\ C_1;C_2\ (\gamma_M\ \hat{M})) & \\
& = & \alpha_M(\mathcal{C}\ C_2(\mathcal{C}\ C_1\ (\gamma_M\ \hat{M}))) & (\texttt{by def. of}\ \mathcal{F}) \\
& = & \alpha_M(\mathcal{C}\ C_2(\mathcal{C}\ C_1\ (\gamma_M\ \hat{M}))) & \\
& \sqsubseteq & (\alpha\mathcal{C})C_2(\alpha_M(\mathcal{C}\ C_1(\gamma_M\ \hat{M}))) & (\texttt{by Prop.1 twice}) \\
& \sqsubseteq & (\alpha\mathcal{C})C_2((\alpha\mathcal{C})\ C_1(\alpha_M(\gamma_M\ \hat{M}))) & (\texttt{by Prop.1 twice}) \\
& \sqsubseteq & (\alpha\mathcal{C})C_2((\alpha\mathcal{C})\ C_1\ \hat{M}) & (\alpha_M\gamma_M \sqsubseteq id) \\
& = & \hat{\mathcal{C}}\ C_1;C_2\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ \mathcal{C})\ C_1;C_2\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{\mathcal{C}})\ C_1;C_2\ \hat{M}$

- $C \to \texttt{if}\ E\ C_1\ C_2$

$$
\begin{array}{lll}
lhs & = & \alpha\ (\mathcal{F}\ \mathcal{C})\ \texttt{if}\ E\ C_1\ C_2\ \hat{M} \\
rhs & = & \alpha_C \circ (\mathcal{F}\ \mathcal{C})\ \texttt{if}\ E\ C_1\ C_2\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
& = & (\alpha_C((\mathcal{F}\ \mathcal{C})\ \texttt{if}\ E\ C_1\ C_2))\ \hat{M} & \\
& = & (\alpha_M \circ ((\mathcal{F}\ \mathcal{C})\ \texttt{if}\ E\ C_1\ C_2) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha_C) \\
& = & \alpha_M((\mathcal{F}\ \mathcal{C})\ \texttt{if}\ E\ C_1\ C_2(\gamma_M\ \hat{M})) & \\
& = & \alpha_M(\mathcal{C}\ C_1\ (\mathcal{B}\ E_1\ (\gamma_M\ \hat{M})))\cup & \\
& & (\mathcal{C}\ C_2\ (\neg\mathcal{B}\ E_1\ (\gamma_M\ \hat{M}))) & (\texttt{by def. of}\ \mathcal{F}) \\
& = & \alpha_M(\mathcal{C}\ C_1\ (\mathcal{B}\ E_1\ (\gamma_M\ \hat{M})))\sqcup & \\
& & \alpha_M(\mathcal{C}\ C_2\ (\neg\mathcal{B}\ E_1\ (\gamma_M\ \hat{M}))) & (\alpha_M\ \texttt{is continuous}) \\
& = & ((\alpha\mathcal{C})\ C_1\ \alpha_M(\mathcal{B}\ E_1\ (\gamma_M\ \hat{M})))\sqcup & \\
& & ((\alpha\ \mathcal{C})\ C_2\ \alpha_M(\neg\mathcal{B}\ E_1\ (\gamma_M\ \hat{M}))) & (\texttt{by Prop.1 twice}) \\
& \sqsubseteq & ((\alpha\mathcal{C})\ C_1\ \alpha_M(\gamma_M\ \hat{M}))\sqcup & \\
& & ((\alpha\ \mathcal{C})\ C_2\ \alpha_M((\gamma_M\ \hat{M})) & (\alpha_M\ \texttt{is monotone}) \\
& \sqsubseteq & ((\alpha\mathcal{C})\ C_1\ \hat{M})\ \hat{\sqcup}\ ((\alpha\ \mathcal{C})\ C_2\ \hat{M}) & (\alpha_M\gamma_M \sqsubseteq id\ \&\ \hat{\sqcup}) \\
& = & \hat{\mathcal{C}}\ \texttt{if}\ C_1\ C_2\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ \texttt{if}\ \ E\ C_1\ C_2\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ \texttt{if}\ \ E\ C_1\ C_2\ \hat{M}$

- $C \rightarrow x = \texttt{pop}\ y\ E$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ x = \texttt{pop}\ y\ E\ \hat{M} & \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ x = \texttt{pop}\ y\ E\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ x = \texttt{pop}\ y\ E))\hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ x = \texttt{pop}\ y\ E) \circ \gamma_M)\hat{M} & (\texttt{by def. of}\ \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ x = \texttt{pop}\ y\ E\ (\gamma_M\hat{M})) & \\
 & = & \alpha_M\{m\{x \mapsto s, y \mapsto z\} \mid s = m(y)[\mathcal{V}\ E\ \{m\}], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & (\texttt{by def. of}\ \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{x \mapsto s, y \mapsto z\} \mid s = m(y)[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & (\texttt{by def. of}\ \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{x \mapsto s, y \mapsto z\} \mid s = m(y)[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & \\
 & = & \alpha_2\alpha_1\{m' \uplus \{x \mapsto s, y \mapsto z\} \mid s = m(y)[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & (\texttt{by}\ \Cup) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\}\ \Cup\ \alpha_2\alpha_1\{\{x \mapsto s, y \mapsto z\} \mid & \\
 & & s = m(y)[\mathcal{V}\ E\ \{m\}], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & (\Cup\ \texttt{is continuous}) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\}\ \Cup\ \alpha_2\alpha_1\{\{x \mapsto s, y \mapsto z\} \mid & \\
 & & s = m(y)[\mathcal{V}\ E\ \{m\}], & \\
 & & (s \rightarrow \mathcal{V}\ E\ \{m\}) \uplus m(z) = m(y), m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1\ \texttt{is monotone}) \\
 & = & \alpha_2\alpha_1\{\gamma_M\hat{M}\}\ \Cup\ \alpha_2\{x \mapsto s, y \mapsto z \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(y) \mid m \in \gamma_M\hat{M}\} & \\
 & & (k \mapsto s) \uplus \hat{m}(z) = \hat{m}(y)\} & (\texttt{by}\ \alpha_1) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\}\ \Cup\ \alpha_2\{x \mapsto s, y \mapsto y \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(y) \mid m \in \gamma_M\hat{M}\}\} & (\alpha_2\ \texttt{is monotone}) \\
 & \sqsubseteq & \hat{M}\ \Cup\ \alpha_2\{x \mapsto s \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(y) \mid m \in \gamma_M\hat{M}\}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & \sqsubseteq & \hat{M}\ \Cup\ \{x \mapsto \alpha_V(s) \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(y) \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by}\ \alpha_2) \\
 & = & \hat{M}\ \Cup\ \{x \mapsto s' \mid s' = \hat{M}(y)[\hat{\mathcal{V}}\ E\ \hat{M}]\} & (\texttt{by}\ \alpha_V) \\
 & = & (\hat{\mathcal{F}}\ \hat{C})\ x = \texttt{pop}\ y\ E\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ \texttt{pop}\ \ y\ E\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ \texttt{pop}\ \ y\ E\ \hat{M}$

- $C \rightarrow x = \texttt{pop} * y\ E$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ x = \texttt{pop} * y\ E\ \hat{M} & \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ x = \texttt{pop} * y\ E\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ x = \texttt{pop} * y\ E))\hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ x = \texttt{pop} * y\ E) \circ \gamma_M)\hat{M} & (\texttt{by def. of } \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ x = \texttt{pop} * y\ E\ (\gamma_M\hat{M})) & \\
 & = & \alpha_M\{m\{x \mapsto s, y \mapsto z\} \mid s = m(m(y))[\mathcal{V}\ E\ \{m\}], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & (\texttt{by def. of } \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{m\{x \mapsto s, y \mapsto z\} \mid s = m(m(y))[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & (\texttt{by def. of } \alpha_M) \\
 & = & \alpha_2\alpha_1\{m\{x \mapsto s, y \mapsto z\} \mid s = m(m(y))[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & \\
 & = & \alpha_2\alpha_1\{m' \uplus \{x \mapsto s, y \mapsto z\} \mid s = m(m(y))[\mathcal{V}\ E\ m], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & (\texttt{by } \uplus) \\
 & = & \alpha_2\alpha_1\{m' \mid m \in \gamma_M\hat{M}\} \ \uplus\ \alpha_2\alpha_1\{\{x \mapsto s, y \mapsto z\} \mid & \\
 & & s = m(m(y))[\mathcal{V}\ E\ \{m\}], & \\
 & & (\mathcal{V}\ E\ \{m\} \mapsto s) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & (\uplus \texttt{ is continuous}) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \ \uplus\ \alpha_2\alpha_1\{\{x \mapsto s, y \mapsto z\} \mid & \\
 & & s = m(m(y))[\mathcal{V}\ E\ \{m\}], & \\
 & & (s \rightarrow \mathcal{V}\ E\ \{m\}) \uplus m(m(z)) = m(m(y)), m \in \gamma_M\hat{M}\} & (\alpha_2\alpha_1 \texttt{ is monotone}) \\
 & = & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \ \uplus\ \alpha_2\{x \mapsto s, y \mapsto z \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(m(y)) \mid m \in \gamma_M\hat{M}\} & \\
 & & (k \mapsto s) \uplus \hat{m}(z) = \hat{m}(y)\} & (\texttt{by } \alpha_1) \\
 & \sqsubseteq & \alpha_2\alpha_1\{\gamma_M\hat{M}\} \ \uplus\ \alpha_2\{x \mapsto s, y \mapsto y \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(m(y)) \mid m \in \gamma_M\hat{M}\}\} & (\alpha_2 \texttt{ is monotone}) \\
 & \sqsubseteq & \hat{M} \ \uplus\ \alpha_2\{x \mapsto s \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(m(y)) \mid m \in \gamma_M\hat{M}\}\} & (\alpha_M\gamma_M \sqsubseteq id) \\
 & \sqsubseteq & \hat{M} \ \uplus\ \{x \mapsto \alpha_V(s) \mid s = \hat{m}(y) \odot [k] & \\
 & & k = \{v \mid v \in \mathcal{V}\ E\ \{m\}, m \in \gamma_M\hat{M}\}, & \\
 & & \hat{m}(y) = \{m(m(y)) \mid m \in \gamma_M\hat{M}\}\} & (\texttt{by } \alpha_2) \\
 & = & \hat{M} \ \uplus\ \{x \mapsto s' \mid s' = \hat{M}(\hat{M}(y))[\hat{\mathcal{V}}\ E\ \hat{M}]\} & (\texttt{by } \alpha_V) \\
 & = & (\mathcal{F}\ \hat{C})\ x = \texttt{pop} * y\ E\ \hat{M} & (\texttt{by def. of } \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ \texttt{pop} * y\ E\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ \texttt{pop} * y\ E\ \hat{M}$

- $C \to \texttt{foreach}\ x\ E\ C$

$$
\begin{array}{rcll}
lhs & = & \alpha\ (\mathcal{F}\ C)\ \texttt{foreach}\ x\ E\ C\ \hat{M} & \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ \texttt{foreach}\ x\ E\ C\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ \texttt{foreach}\ x\ E\ C))\ \hat{M} & \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ \texttt{foreach}\ x\ E\ C) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of } \alpha_C) \\
 & = & \alpha_M((\mathcal{F}\ C)\ \texttt{foreach}\ x\ E\ C\ (\gamma_M\ \hat{M})) & \\
 & = & \alpha_M\{\{C\ C\ \{m_{i0}\{x \mapsto (\mathcal{V}\ E\ m_i)[0]\} \mid m_{i0} \in m_k\}; & \\
 & & \quad C\ C\ \{m_{i1}\{x \mapsto (\mathcal{V}\ E\ m_i)[1]\} \mid m_{i1} \in m_k\}; & \\
 & & \quad \ldots & \\
 & & \quad C\ C\ \{m_{in}\{x \mapsto (\mathcal{V}\ E\ m_i)[n]\} \mid m_{in} \in m_k\}\}_k \mid & \\
 & & \quad (n = |\mathtt{dom}(\mathcal{V}\ E\ m_i)|,\ m_i \in m_k, m_k \in \gamma_M\hat{M})\} & (\texttt{by def. of } \mathcal{F}) \\
 & = & \alpha_2 \circ \alpha_1 \circ \{\{C\ C\ \{m_{i0}\{x \mapsto (\mathcal{V}\ E\ m_i)[0]\} \mid m_{i0} \in m_k\}; & \\
 & & \quad C\ C\ \{m_{i1}\{x \mapsto (\mathcal{V}\ E\ m_i)[1]\} \mid m_{i1} \in m_k\}; & \\
 & & \quad \ldots & \\
 & & \quad C\ C\ \{m_{in}\{x \mapsto (\mathcal{V}\ E\ m_i)[n]\} \mid m_{in} \in m_k\}\}_k \mid & \\
 & & \quad (n = |\mathtt{dom}(\mathcal{V}\ E\ m_i)|,\ m_i \in m_k, m_k \in \gamma_M\hat{M})\} & (\texttt{by def. of } \alpha_M) \\
 & = & \alpha_2\alpha_1\{\{C\ C\ \{m_{i0}\{x \mapsto (\mathcal{V}\ E\ m_i)[0]\} \mid m_{i0} \in m_k\}; & \\
 & & \quad C\ C\ \{m_{i1}\{x \mapsto (\mathcal{V}\ E\ m_i)[1]\} \mid m_{i1} \in m_k\}; & \\
 & & \quad \ldots & \\
 & & \quad C\ C\ \{m_{in}\{x \mapsto (\mathcal{V}\ E\ m_i)[n]\} \mid m_{in} \in m_k\}\}_k \mid & \\
 & & \quad (n = |\mathtt{dom}(\mathcal{V}\ E\ m_i)|,\ m_i \in m_k, m_k \in \gamma_M\hat{M})\} & \\
 & = & \{\alpha_2\alpha_1\{C\ C\ \{m_{i0}\{x \mapsto (\mathcal{V}\ E\ m_i)[0]\} \mid m_{i0} \in m_k\}; & \\
 & & \quad C\ C\ \{m_{i1}\{x \mapsto (\mathcal{V}\ E\ m_i)[1]\} \mid m_{i1} \in m_k\}; & \\
 & & \quad \ldots & \\
 & & \quad C\ C\ \{m_{in}\{x \mapsto (\mathcal{V}\ E\ m_i)[n]\} \mid m_{in} \in m_k\}\}_k \mid & \\
 & & \quad (n = |\mathtt{dom}(\mathcal{V}\ E\ m_i)|,\ m_i \in m_k, m_k \in M)\} & \\
 & = & \{\alpha_2\alpha_1\{C\ C\ \{m_{i0} \uplus \{x \mapsto (\mathcal{V}\ E\ m_i)[0]\} \mid m_{i0} \in m_k\}; & \\
 & & \quad C\ C\ \{m_{i1} \uplus \{x \mapsto (\mathcal{V}\ E\ m_i)[1]\} \mid m_{i1} \in m_k\}; & \\
 & & \quad \ldots & \\
 & & \quad C\ C\ \{m_{in} \uplus \{x \mapsto (\mathcal{V}\ E\ m_i)[n]\} \mid m_{in} \in m_k\}\}_k \mid & \\
 & & \quad (n = |\mathtt{dom}(\mathcal{V}\ E\ m_i)|,\ m_i \in m_k, m_k \in M)\} & (\texttt{by def. of } \uplus) \\
 & = & \{(\alpha\ C)\ C\ \alpha_2\alpha_1\{m_k \uplus \{x \mapsto (\mathcal{V}\ E\ m_k)[0]\}\}; & \\
 & & \quad (\alpha\ C)\ C\ \alpha_2\alpha_1\{m_{k1} \uplus \{x \mapsto (\mathcal{V}\ E\ m_k)[1]\}\}; & \\
 & & \quad \ldots & \\
 & & \quad (\alpha\ C)\ C\ \alpha_2\alpha_1\{m_{kn} \uplus \{x \mapsto (\mathcal{V}\ E\ m_k)[n]\} \mid & \\
 & & \quad\quad (n = |\mathtt{dom}(\gamma_M\hat{M})|,\ m_k \in \gamma_M\hat{M})\} & (\texttt{by Prop.1 twice}) \\
 & = & \{(\alpha\ C)\ C\ \alpha_2\alpha_1\{m_k \mid m_k \in \gamma_M\hat{M}\} \uplus & \\
 & & \quad \alpha_2\alpha_1\{\{x \mapsto (\mathcal{V}\ E\ m_k)[0]\} \mid m_k \in \gamma_M\hat{M}\}; & \\
 & & \quad (\alpha\ C)\ C\ \alpha_2\alpha_1\{m_{k1} \mid m_{k1} \in \gamma_M\hat{M}\} \uplus & \\
 & & \quad \alpha_2\alpha_1\{\{x \mapsto (\mathcal{V}\ E\ m_k)[1]\} \mid m_k \in \gamma_M\hat{M}\}; & \\
 & & \quad \ldots & \\
 & & \quad (\alpha\ C)\ C\ \alpha_2\alpha_1\{m_{kn} \mid m_{kn} \in \gamma_M\hat{M}\} \uplus & \\
 & & \quad \alpha_2\alpha_1\{\{x \mapsto (\mathcal{V}\ E\ m_k)[n]\} \mid m_k \in \gamma_M\hat{M}\}; & (\alpha_2\alpha_1\ \texttt{is continuous})
\end{array}
$$

$$
\begin{array}{rcll}
 & = & \{(\alpha\, C)\ C\ \alpha_2\alpha_1\{m_k \mid m_k \in \gamma_M\hat{M}\} \Cup \\
 & & \quad \alpha_2\{x \mapsto (\mathcal{V}\ E\ m_k)[0] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & (\alpha\, C)\ C\ \alpha_2\alpha_1\{m_{k1} \mid m_{k1} \in \gamma_M\hat{M}\} \Cup \\
 & & \quad \alpha_2\{x \mapsto (\mathcal{V}\ E\ m_k)[1] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & \cdots \\
 & & (\alpha\, C)\ C\ \alpha_2\alpha_1\{m_{kn} \mid m_{kn} \in \gamma_M\hat{M}\} \Cup \\
 & & \quad \alpha_2\{x \mapsto (\mathcal{V}\ E\ m_k)[n] \mid m_k \in \gamma_M\hat{M}\}; & (\texttt{by def. of } \alpha_1) \\
 & = & \{(\alpha\, C)\ C\ \alpha_2\alpha_1\{\gamma_M\hat{M}\} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[0] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & (\alpha\, C)\ C\ \alpha_2\alpha_1\{\gamma_M\hat{M}\} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[1] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & \cdots \\
 & & (\alpha\, C)\ C\ \alpha_2\alpha_1\{\gamma_M\hat{M}\} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[n] \mid m_k \in \gamma_M\hat{M}\}; & (\texttt{by def. of } \alpha_2) \\
 & \sqsubseteq & \{(\alpha\, C)\ C\ \hat{M} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[0] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & (\alpha\, C)\ C\ \hat{M} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[1] \mid m_k \in \gamma_M\hat{M}\}; \\
 & & \cdots \\
 & & (\alpha\, C)\ C\ \hat{M} \Cup \\
 & & \quad \{x \mapsto \alpha_V(\mathcal{V}\ E\ m_k)[n] \mid m_k \in \gamma_M\hat{M}\}; & (\alpha_M\gamma_M \sqsubseteq id) \\
 & \sqsubseteq & (\alpha\, C)\ C\ \hat{M} \Cup \{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[0]\}); \\
 & & (\alpha\, C)\ C\ \hat{M}_1 \Cup \{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[1]\}); \\
 & & \cdots \\
 & & (\alpha\, C)\ C\ \hat{M}_n \Cup \{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[n]\})) & (\texttt{by Prop.2}) \\
 & = & (\alpha\, C)\ C\ \hat{M}\{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[0]\}); \\
 & & (\alpha\, C)\ C\ \hat{M}_1\{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[1]\}); \\
 & & \cdots \\
 & & (\alpha\, C)\ C\ \hat{M}_n\{x \mapsto (\hat{\mathcal{V}}\ E\ \hat{M}))[n]\})) \\
 & = & \hat{C}\ \texttt{foreach}\ x\ E\ C\ \hat{M} & (\texttt{by def. of } \hat{\mathcal{F}})
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ C)\ \texttt{foreach}\ \ x\ E\ C\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{C})\ \texttt{foreach}\ \ x\ E\ C\ \hat{M}$

- $C \rightarrow \texttt{while}\ E\ C$

$$
\begin{array}{rcll}
lhs & = & \alpha(\mathcal{F}\ C)\ \texttt{while}\ E\ C\ \hat{M} \\
rhs & = & \alpha_C \circ (\mathcal{F}\ C)\ \texttt{while}\ E\ C\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & (\alpha_C((\mathcal{F}\ C)\ \texttt{while}\ E\ C))\ \hat{M} \\
 & = & (\alpha_M \circ ((\mathcal{F}\ C)\ \texttt{while}\ E\ C) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of } \alpha) \\
 & = & \alpha_M((\mathcal{F}\ C)\ \texttt{while}\ E\ C\ (\gamma_M\hat{M})) \\
 & = & \alpha_M(\mathcal{B}\ \neg E\ ((\gamma_M\hat{M}) \cup \\
 & & \quad (\mathcal{C}\ (\texttt{while}\ E\ C)\ \mathcal{C}\ C\ (\mathcal{B}\ E\ (\gamma_M\hat{M}))))) & (\texttt{by def. of } \mathcal{F})
\end{array}
$$

$$
\begin{array}{lll}
= & \alpha_M((\mathcal{B}\ \neg E\ (\gamma_M \hat{M}))\sqcup & \\
& (\mathcal{B}\ \neg E\ (\mathcal{C}\ (\texttt{while}\ E\ C)\ \mathcal{C}\ C\ (\mathcal{B}\ E\ (\gamma_M \hat{M}))))) & (\mathcal{B}\ \texttt{is continuous}) \\
= & \alpha_M(\mathcal{B}\ \neg E\ (\gamma_M \hat{M}))\sqcup & \\
& \alpha_M(\mathcal{B}\ \neg E\ (\mathcal{C}\ (\texttt{while}\ E\ C)\ \mathcal{C}\ C\ (\mathcal{B}\ E\ (\gamma_M \hat{M})))) & (\alpha_M\ \texttt{is continuous}) \\
\sqsubseteq & \alpha_M(\gamma_M \hat{M})\sqcup & \\
& \alpha_M(\mathcal{C}\ (\texttt{while}\ E\ C)\ \mathcal{C}\ C\ (\gamma_M \hat{M})) & (\alpha_M\ \texttt{is monotone}) \\
\sqsubseteq & \hat{M} \sqcup \alpha_M(\mathcal{C}\ (\texttt{while}\ E\ C)\ \mathcal{C}\ C\ (\gamma_M \hat{M})) & (\alpha_M \circ \gamma_M \sqsubseteq id) \\
= & \hat{M} \sqcup (\alpha\ \mathcal{C})\ (\texttt{while}\ E\ C)\ (\alpha_M(\mathcal{C}\ C\ (\gamma_M \hat{M}))) & (\texttt{by Prop.1 twice}) \\
= & \hat{M} \sqcup (\alpha\ \mathcal{C})\ (\texttt{while}\ E\ C)\ ((\alpha\ \mathcal{C})\ C\ \alpha_M(\gamma_M \hat{M})) & (\texttt{by Prop.1 twice}) \\
\sqsubseteq & \hat{M} \sqcup (\alpha\ \mathcal{C})\ (\texttt{while}\ E\ C)\ ((\alpha\ \mathcal{C})\ C\ \hat{M}) & (\alpha_M \circ \gamma_M \sqsubseteq id) \\
= & (\hat{\mathcal{F}}\ \hat{\mathcal{C}})\ \texttt{while}\ E\ C\ \hat{M} &
\end{array}
$$

$\therefore \alpha\ (\mathcal{F}\ \mathcal{C})\ \texttt{while}\ E\ \hat{M} \sqsubseteq (\hat{\mathcal{F}}\ \hat{\mathcal{C}})\ \texttt{while}\ E\ \hat{M}$

3.2.2 Expression E

$$(Mem \to 2^{Val}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\hat{Mem} \to \hat{Val})$$

Thus,

$$\alpha(\mathcal{V}) = \alpha_V \circ \mathcal{V} \circ \gamma_{\hat{Mem}}$$

For expression E, to show:

$$\alpha\ (\mathcal{V}\ E) \quad \sqsubseteq \quad (\hat{\mathcal{V}}\ E)$$

Proof this by induction hypothesis of E.

- $E \to \texttt{readint}$

$$
\begin{array}{rcl}
\hat{\mathcal{V}}\ \texttt{readint} & = & \alpha\{\top\} \\
\therefore \alpha\ (\mathcal{V}\ \texttt{readint}) & \sqsubseteq & \hat{\mathcal{V}}\ \texttt{readint}
\end{array}
$$

- $E \to n$

$$
\begin{array}{rcl}
(\hat{\mathcal{V}}\ n) & = & \alpha(\mathcal{V}\ n) \\
\therefore \alpha\ (\mathcal{V}\ n) & \sqsubseteq & (\hat{\mathcal{V}}\ n)
\end{array}
$$

- $E \to E_1 + E_2$
  by I.H
  $\alpha(\mathcal{V}\ E_1) \sqsubseteq \hat{\mathcal{V}}\ E_1$
  $\alpha(\mathcal{V}\ E_2) \sqsubseteq \hat{\mathcal{V}}\ E_2$

$$
\begin{array}{lcll}
\alpha(\mathcal{V}\ E_1 + E_2)\ \hat{M} & = & (\alpha_V \circ (\mathcal{V}\ E_1 + E_2) \circ \gamma_M)\ \hat{M} & (\texttt{by def. of}\ \alpha) \\
& = & \alpha_V(\mathcal{V}\ E_1 + E_2)(\gamma_M\ \hat{M}) & \\
& = & \alpha_V(\dot{+}\langle \mathcal{V}\ E_1\ \gamma_M \hat{M}, \mathcal{V}\ E_2\ \gamma_M \hat{M}\rangle & (\texttt{by def. of}\ \mathcal{V}) \\
& \sqsubseteq & \alpha_V(\dot{+}\langle \gamma_V(\hat{\mathcal{V}}\ E_1\ \hat{M}), \gamma_V(\hat{\mathcal{V}}\ E_2\ \hat{M})\rangle & (\dot{+}\ \texttt{is monotone and}\ \textit{i.h.}) \\
& = & \hat{\mathcal{V}}\ E_1 + E_2\ \hat{M} & (\texttt{by def. of}\ \hat{\mathcal{V}})
\end{array}
$$

$\therefore\ \hat{\mathcal{V}}\ E_1 + E_2 \sqsupseteq \alpha(\mathcal{V}\ E_1 + E_2)$

- $E \to -E$
  by I.H
  $\alpha(\mathcal{V}\,E) \sqsubseteq \hat{\mathcal{V}}\,E$

$$\begin{array}{rcll}
\alpha(\mathcal{V}\ -\ E)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\ -\ E) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\
& = & \alpha_V(\mathcal{V}\ -E)(\gamma_M\,\hat{M}) & \\
& = & \alpha_V(\dot{-}\langle \mathcal{V}\,E\,\gamma_M\hat{M}\rangle) & (\texttt{by def. of } \mathcal{V}) \\
& \sqsubseteq & \alpha_V(\dot{-}\langle \gamma_V(\hat{\mathcal{V}}\,E\,\hat{M})\rangle) & (\textit{dot}- \texttt{ is monotone and } \textit{i.h.}) \\
& = & \hat{\mathcal{V}}\ -E\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}})
\end{array}$$

  $\therefore \hat{\mathcal{V}}\ -E \sqsupseteq \alpha(\mathcal{V} - E)$

- $E \to E_1 < E_2$
  by I.H
  $\alpha(\mathcal{V}\,E_1) \sqsubseteq \hat{\mathcal{V}}\,E_1$
  $\alpha(\mathcal{V}\,E_2) \sqsubseteq \hat{\mathcal{V}}\,E_2$

$$\begin{array}{rcll}
\alpha(\mathcal{V}\,E_1 < E_2)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\,E_1 < E_2) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\
& = & \alpha_V(\mathcal{V}\,E_1 < E_2)(\gamma_M\,\hat{M}) & \\
& = & \alpha_V(\dot{<}\langle \mathcal{V}\,E_1\,\gamma_M\hat{M}, \mathcal{V}\,E_2\,\gamma_M\hat{M}\rangle & (\texttt{by def. of } \mathcal{V}) \\
& \sqsubseteq & \alpha_V(\dot{<}\langle \gamma_V(\hat{\mathcal{V}}\,E_1\,\hat{M}), \gamma_V(\hat{\mathcal{V}}\,E_2\,\hat{M})\rangle & (\dot{<} \texttt{ is monotone and } \textit{i.h.}) \\
& = & \hat{\mathcal{V}}\,E_1 < E_2\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}})
\end{array}$$

- $E \to E_1 = E_2$
  by I.H
  $\alpha(\mathcal{V}\,E_1) \sqsubseteq \hat{\mathcal{V}}\,E_1$
  $\alpha(\mathcal{V}\,E_2) \sqsubseteq \hat{\mathcal{V}}\,E_2$

$$\begin{array}{rcll}
\alpha(\mathcal{V}\,E_1 = E_2)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\,E_1 = E_2) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\
& = & \alpha_V(\mathcal{V}\,E_1 = E_2)(\gamma_M\,\hat{M}) & \\
& = & \alpha_V(\dot{=}\langle \mathcal{V}\,E_1\,\gamma_M\hat{M}, \mathcal{V}\,E_2\,\gamma_M\hat{M}\rangle & (\texttt{by def. of } \mathcal{V}) \\
& \sqsubseteq & \alpha_V(\dot{=}\langle \gamma_V(\hat{\mathcal{V}}\,E_1\,\hat{M}), \gamma_V(\hat{\mathcal{V}}\,E_2\,\hat{M})\rangle & (\dot{=} \texttt{ is monotone and } \textit{i.h.}) \\
& = & \hat{\mathcal{V}}\,E_1 = E_2\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}})
\end{array}$$

  $\therefore\ \hat{\mathcal{V}}\,E_1 = E_2 \sqsupseteq \alpha(\mathcal{V}\,E_1 = E_2)$

- $E \to x$

$$\begin{array}{rcll}
\alpha(\mathcal{V}\,x)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\,x) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\
& = & \alpha_V(\mathcal{V}\,x\,(\gamma_M\,\hat{M})) & \\
& = & \alpha_V\{m(x) \mid m \in (\gamma_M\,\hat{M})\} & (\texttt{by def. of } \mathcal{V}) \\
& = & \hat{\mathcal{V}}\,x\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}})
\end{array}$$

  $\therefore \alpha(\mathcal{V}\,x)\,\hat{M} \sqsubseteq \hat{\mathcal{V}}\,x\,\hat{M}$

- $E \to *x$

$$\begin{array}{rcll} \alpha(\mathcal{V} * x)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V} * x) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\ & = & \alpha_V(\mathcal{V} * x\,(\gamma_M\,\hat{M})) & \\ & = & \alpha_V\{m(m(x)) \mid m \in (\gamma_M\,\hat{M})\} & (\texttt{by def. of } \mathcal{V}) \\ & = & \hat{\mathcal{V}} * x\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}}) \end{array}$$

$\therefore \alpha(\mathcal{V} * x)\,\hat{M} \sqsubseteq \hat{\mathcal{V}} * x\,\hat{M}$

- $E \to \&x$

$$\begin{array}{rcll} \alpha(\mathcal{V}\,\&x)\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\,\&x) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\ & = & \alpha_V(\mathcal{V}\,\&x\,(\gamma_M\,\hat{M})) & \\ & = & \alpha_V\{x\} & (\texttt{by def. of } \mathcal{V}) \\ & = & \hat{\mathcal{V}}\,\&x\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}}) \end{array}$$

$\therefore \alpha(\mathcal{V}\,\&x)\,\hat{M} \sqsubseteq \hat{\mathcal{V}}\,\&x\,\hat{M}$

- $E \to x[E]$

$$\begin{array}{rcll} \alpha(\mathcal{V}\,x[E])\,\hat{M} & = & (\alpha_V \circ (\mathcal{V}\,x[E]) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\ & = & \alpha_V(\mathcal{V}\,x[E]\,(\gamma_M\,\hat{M})) & \\ & = & \alpha_V\{m(x)[\mathcal{V}\,E\,m] \mid m \in (\gamma_M\hat{M})\} & (\texttt{by def. of } \mathcal{V}) \\ & \sqsubseteq & \hat{\mathcal{V}}\,x[E]\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}}) \end{array}$$

$\therefore \alpha(\mathcal{V}\,x[E])\,\hat{M} \sqsubseteq \hat{\mathcal{V}}\,x[E]\,\hat{M}$

- $E \to *x[E]$

$$\begin{array}{rcll} \alpha(\mathcal{V} * x[E])\,\hat{M} & = & (\alpha_V \circ (\mathcal{V} * x[E]) \circ \gamma_M)\,\hat{M} & (\texttt{by def. of } \alpha) \\ & = & \alpha_V(\mathcal{V} * x[E]\,(\gamma_M\,\hat{M})) & \\ & = & \alpha_V\{m(m(x))[\mathcal{V}\,E\,m] \mid m \in (\gamma_M\hat{M})\} & (\texttt{by def. of } \mathcal{V}) \\ & = & \hat{\mathcal{V}} * x[E]\,\hat{M} & (\texttt{by def. of } \hat{\mathcal{V}}) \end{array}$$

$\therefore \alpha(\mathcal{V} * x[E])\,\hat{M} \sqsubseteq \hat{\mathcal{V}} * x[E]\,\hat{M}$

We have proved that our analysis system is correct, which means that the analysis result covers all the real situations.

Proof done;

4 Conclusion

4.1 Summary

Traditional testing methods hardly scaleup for large safety critical systems as found in avionics, automotive, healthcare, e-commerce and security industry. Which can not find many possible

bugs, also time consumming and human forces. As a result, computerized mordern societies are highly frahile to software bugs.

In real Perl programs, there can be many other kinds of bugs. But in order to improve correctness robust result, we have to specialize bug pattern in the analysis, like our analysis. And using uninitialized array element usage bugs is a very universal mistake in Perl society that which can bring unimaginable logical bugs.

Our analysis used a viable alternative static analysis technology to detect uninitialized array element usage in core Perl programs automatically before program runs, which can greately reduce our resources and can improve our confidence with "No Alarm" message.

Our analysis detects sound bugs. In this thesis we collect array keys as a set, which we will represent as **Key Set**, **KS** for short. We join array values as a set either, which we will represent as **Value Set**, **VS** for short. If some KS joined after the join point, then we join **KS** and join their values as a set too.

From this analysis, we have convinced that static analysis is a very useful method for detecting bugs before program runs and this process is done by computer automatically.

4.2 Contributions

In this thesis, we have applied static analysis in core Perl language and which can detect uninitialized array element usage bugs. And convinced us that abstract interpretation can bring us a key to conqure program bugs which traditional bug detection method can not do.

People can extend our idea to other many programming languages. For example, their own system's language and also can be a real huge programming language. Find their own bug patterns and detect them.

We have also proofed that our analysis result is correct. Means, our abstract analysis covers the real programming execution envirionment.

4.3 Future Work

We have done this analysis in a core Perl langauge, but we need to imporove our analysis to find bugs in real Perl programs in the future. To accomplish this goal,

1. Develop/Find Perl parser which can parsing many commercial perl sources.
2. Extend our core language to adapt more Perl language features.
3. Convert Perl programs to our core language and analysis in our language.
4. Reduce false alarms and detect more realiable bugs.

References

[1] Patric Cousot, Radhia Cousot, *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*, In Proceedings of ACM Symposium on Principles of Programming Languages, pages 238-252, January 1977

[2] Patrick Cousot and Radhia Cousot, *Systematic Design of Program Anslysis Frameworks*, Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Language, January 29-31, 1979

[3] Andrew M.Pitts, *Lecture Note on Denotational Semantics*. Part II of the Computer Science Tripos, 1999

[4] Patrick Cousot, *Abstract Interpretation: Achievements and Perspectives*, 2000

[5] Patrick Cousot, *Abstract Interpretation Based Formal Methods and Future Challenge*, 2000

[6] Patrick Cousot and Radhia Cousot, *Abstract Interpretation Frameworks*, 1992

[7] Patrick Cousot and Radhia Cousot, *Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation*, In PLILP '92: Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming, pages 269-295. Springer-Verlag, 1992.