CHAPTER IV

Soundness and Relative Completeness

4.1 Soundness

In Chapter I (page 7), we said that soundness of the formal proof system implies the truth of the correctness conjecture for every assertive program (i.e., program-with-specification) that can be transformed to a mathematical statement that is true. By now we are referring to correct assertive programs (Definition 2.1, page 58) and true mathematical statements (Definition 2.2, page 59) as *valid*; hence, we have the following definition:

Definition 4.1 *A proof system for assertive programs is* sound *if and only if every program that (using only the rules of the proof system) can be transformed to a valid mathematical statement is, itself, valid.*

Recall from Section 3.2 that validity is *preserved* in the direction of a rule application if and only if the validity of the original implies the validity of the result. We have designed each of the rules to preserve validity in the program direction. Suppose we use the rules to transform an assertive program Prog into mathematical assertion H, and that H is valid (i.e., it is true in every model of its theory). Further suppose that every rule preserves validity when applied in the program direction. Then we can apply the rules in the reverse order to transform H into Prog. The result of applying each rule will be a valid assertive program. In particular, Prog is valid. Hence, assuming every rule preserves validity in the program direction, every assertive program that (using only the rules of the proof system) can be transformed to a valid mathematical statement is necessarily valid. Therefore, we have just proven the following lemma:

Lemma 4.1 *If every rule presented in Chapter III preserves validity in the program direction, then the proof system composed of those rules is sound.*

The following theorem states one of the two important results discussed in the present chapter.

Theorem 1 (Soundness) *The proof system composed of the rules of Chapter III is sound.*

Theorem 1 follows from Lemmas 4.1 and 4.2.

Lemma 4.2 *Every rule presented in Chapter III preserves validity in the program direction.*

We define invalidity to be *preserved* in the direction of a rule application if and only if the invalidity of the original implies the invalidity of the result. Using the contrapositive, we obtain the fact that invalidity is preserved in the direction of a rule application if and only if the validity of the result implies the validity of the original. Therefore, a rule preserves validity in a given direction if and only if it preserves invalidity in the opposite direction. Hence Lemma 4.2 follows immediately from Lemma 4.3.

Lemma 4.3 *Every rule presented in Chapter III preserves invalidity in the math direction.*

We prove Lemma 4.3 by establishing invalidity preservation in the math direction for each rule, one at a time, in Section 4.4.

4.2 Relative Completeness

Turning to the question of relative completeness, there are two problems that make the simplistic definition we made in Chapter I (page 7) not quite serviceable: when a "system is relatively complete, every program/specification pair satisfying the correctness conjecture is transformable to a mathematical assertion that is true." In the language of validity, this statement translates to the following possible definition: a proof system for assertive programs is relatively complete if and only if every valid program can be transformed (using only the rules of the proof system) to a valid mathematical statement.

This first problem is that we did not design the indexed method so that it could transform *all* assertive programs. It is designed specifically for transforming those programs that are a result of applying Krone's [25] procedure declaration rule. These programs begin with a **remember** statement and conclude with a **confirm** statement. Application of the bridge rule to such a program produces a program that conforms to the syntax of top level code. All the other rules of the indexed method transform

$C\backslash$ **alter all**
stow(0)
assume (**true**) $\Rightarrow$ ($x_0 < 8$)
whenever true do
loop
maintaining $x < 9$
while $x < 7$
stow(1)
Inc(x)
stow(2)
end loop
stow(3)
confirm $x_3 = 7$
end whenever

where $C \supseteq \{x : \text{Integer}\} \cup \left\{ \begin{array}{c} \textbf{procedure } \text{Inc(y: Integer)} \\ \textbf{ensures } y = \#y + 1 \end{array} \right\}$

Figure 74: A Valid Assertive Program

only top level code, producing only top level code. We, therefore, make the following definition.

Definition 4.2 *A* well-prepared assertive program *is an assertive program that conforms to the syntax of either (1) top level code or (2) $\mathcal{P}$ defined in the bridge rule (Figure 40).*

Then, provisionally, a proof system for well-prepared assertive programs is relatively complete if and only if every valid well-prepared program can be transformed (using only the rules of the proof system) to a valid mathematical statement.

The second problem is that the existence of internal assertions such as procedure specifications and loop invariants causes our proof system not to satisfy even this definition of relative completeness. For example, application of the **loop while** rule to the valid program shown in Figure 74 produces the invalid program shown in Figure 75.

First we argue that the program in Figure 74 is valid. It is valid if its interpretation takes every neutral environment to an environment that is not categorically false. Let env = [a, cs, os, ns, se, d] be the environment that results from interpreting the first

three statements of the program (**alter all**, **stow**(0), and **assume** (**true**) $\Rightarrow$ ($x_0 < 8$)) in an arbitrary neutral environment. Because interpreting each of those statements in a neutral environment never produces a categorically false environment, a $\neq$ CF. The result of interpreting the entire program is a vacuously true environment (and, therefore, not categorically false) if a = VT; we, therefore, must assume that a = NL. Then, due to the interpretation of the first three statements, especially the **assume** statement, we have cs(x) = ns(x, 0) < 8. Let $\mathcal{F}$ be the interpretation of the loop in the program. Because env is a neutral environment,

$$\mathcal{F}(\text{env}) = [\text{NL}, \text{cs}', \text{os}, \text{ns}', \text{se}, \text{d}] \tag{4.1}$$

$$\text{where } \text{cs}'(\xi) = \begin{cases} \text{cs}(\xi) & \text{if } \xi \neq \text{x} \\ 7 & \text{else if cs(x)} \leq 7 \\ \text{cs(x)} & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{and } \text{ns}'(\xi, i) = \begin{cases} \text{ns}(\xi, i) & \text{if } i \notin \{1, 2\} \vee 7 \leq \text{cs(x)} \\ \text{ns}(\xi, 0) & \text{else if } \xi \neq \text{x} \\ 6 & \text{else if } i = 1 \\ 7 & \text{otherwise} \end{cases} \tag{4.3}$$

Because cs(x) is an integer and cs(x) < 8, we have cs(x) $\leq$ 7, and, by equation 4.2, cs′(x) = 7. Hence, the interpretation of the sequence **stow**(3) **confirm** $x_3 = 7$ in the environment $\mathcal{F}$(env) necessarily yields a neutral environment. Therefore, the program in Figure 74 is valid.

The only rule applicable to the program in Figure 74 is the **loop while** rule. Figure 75 shows the result of this rule application. We can show the program in Figure 75 to be invalid. Let env = [NL, cs, os, ns, se, d] be a neutral environment such that the setup se has at least three states in its sequence that have the following properties. The first state of se maps x to the value 7, and the third state of se maps x to the value 8. The second state's value does not matter. Then, after execution of **stow**(0) in Figure 75, ns(x, 0) = 7 (i.e., $x_0 = 7$). Therefore, the assert status is still NL after execution of **stow**(1). The statement sequence inside the "**whenever** (**true**) $\wedge$ ($x_0 < 7$) **do**" statement is not executed, so the assert status is still NL after execution of **stow**(3). Due to the setup, we now have ns(x, 3) = 8 (i.e., $x_3 = 8$). Hence, the assert status is still NL after execution of "**assume** ($\neg(x_3 < 7)$) $\wedge$ ($x_3 < 9$)". We still have $x_3 = 8$, so execution of "**confirm** $x_3 = 7$" produces a categorically false environment. We have shown the program in Figure 75 to be invalid.

The proof rules of Chapter III can be applied in the math direction to rewrite the program in Figure 75 to a mathematical assertion. This assertion will be invalid (i.e., false) because, as we will learn in Section 4.4, all the rules preserve invalidity in the math direction. Once rewriting produces an invalid program, the programs produced

$C\backslash$ **alter all**
stow(0)
assume (**true**) $\Rightarrow$ ($x_0 < 8$)
confirm (**true**) $\Rightarrow$ ($x_0 < 9$)
alter all
stow(1)
whenever (**true**) $\wedge$ ($x_0 < 7$) **do**
  assume ($x_1 < 7$) $\wedge$ ($x_1 < 9$)
  Inc(x)
  stow(2)
  confirm $x_2 < 9$
end whenever
alter all
stow(3)
whenever true do
  assume ($\neg(x_3 < 7)$) $\wedge$ ($x_3 < 9$)
  confirm $x_3 = 7$
end whenever

Figure 75: An Invalid Assertive Program

by all further rewriting are also invalid. We have demonstrated that our rules cannot always preserve validity in the math direction. The valid program of Figure 74 cannot be transformed (using only the rules of the proof system) to a valid mathematical statement. The traditional proof systems do not differ from ours in this respect.

Not all is lost, however. Although the **maintaining** clause's claim about the loop in Figure 74 is true, it is not true enough! The loop itself accomplishes more than what its stated invariant advertises. Had the loop invariant been $x < 8$ rather than $x < 9$, the program would have been transformed by the rules to a true mathematical statement. Perhaps for every well-prepared valid assertive program Prog there exists a related valid program Prog′, which differs from Prog at most in the internal assertions (loop invariants and procedure specifications), such that Prog′ can be transformed (using only the rules of the proof system) to a valid mathematical statement. Such a proof system would still be quite useful. The authors of a valid assertive program could discover loop invariants and procedure specifications that are "tight" enough so that their program with these modified assertions could be shown to be valid—by using the rules to transform the modified program into a valid mathematical statement. This characterization is our definition of relative completeness:

Definition 4.3 *A proof system for well-prepared assertive programs is* relatively complete *if and only if for every well-prepared valid assertive program Prog there exists a related valid program Prog′, which differs from Prog at most in the internal assertions (loop invariants and procedure specifications), such that Prog′ can be transformed (using only the rules of the proof system) to a valid mathematical statement.*

The **loop while** rule is sort of an exception among our proof rules. Nearly all the other rules do, in fact, preserve validity in the math direction. The other exception is the procedure call rule, which can fail to preserve validity in the math direction when the procedure's postcondition is weaker than what Cook defines to be "the *post relation corresponding to*" the procedure's precondition and its body [4, p. 85]. By use of the minimum fixed-point operation, the procedure's body defines its procedure-function. This procedure-function plays an important role in the validity of the assertive program. When the procedure call rule removes the call and replaces it with an **assume** statement constructed from the procedure's weaker postcondition, the resulting program may be invalid.

A procedure that is declared and defined in the program is an *internal* procedure; others are *external*. The body of an internal procedure is part of the assertive program; the body of an external procedure is not. Because researchers have not yet produced a satisfactory relational semantics for assertive programs (see Section 5.3.1), we have adopted a well-understood functional semantics. This choice causes a technical problem for relative completeness with respect to external procedures. The

$C\backslash$ **alter all**
stow(0)
assume true
whenever true do
Make_1_Or_Minus_1(x)
stow(1)
Make_1_Or_Minus_1(y)
stow(2)
confirm $x_2 + y_2 \neq 0$
end whenever

where $C \supseteq \{x, y : \text{Integer}\} \cup \left\{ \begin{array}{l} \textbf{procedure}\ \text{Make\_1\_Or\_Minus\_1}(z: \text{Integer}) \\ \quad \textbf{ensures}\ (z = 1) \vee (z = -1) \end{array} \right\}$

Figure 76: An Assertive Program That Is Valid According to Functional Semantics

indexed method's rules, like Krone's rules, are designed so that external procedures can have relational behavior. We do not want programs to be considered correct whose validity depends crucially on the external procedures' adhering to functional semantics. For example, the program in Figure 76, given our functional semantics, and assuming that Make_1_Or_Minus_1 is an external procedure, is valid. Because the semantics of Make_1_Or_Minus_1 is functional, it either sets both x and y to 1, or it sets both x and y to -1. In either case $x + y \neq 0$. However, this program would not be valid if the semantics of Make_1_Or_Minus_1 were relational; the result of the procedure call could be 1 for x and -1 for y, making $x + y = 0$.

We want our proof rules to be sound whether the semantics is functional or relational; consequently, the indexed method, like Krone's method, is not relatively complete if the external procedures have functional semantics. Both methods are designed to be relatively complete if the external procedures have relational semantics. A careful proof that this is so will have to wait for the development of a satisfactory relational semantics. These technical problems do not arise if every external procedure called is functionally specified. Therefore, at present, we are in a position to provide a careful proof that the indexed method is relatively complete if all called procedures are either internal procedures or functionally specified external procedures.

Cook defines the assertion language to be *expressive* if, for every precondition and statement sequence, there is a formula of the language that expresses the post relation corresponding to the precondition and statement sequence [4, p. 86]. He

also shows that if the assertion language is expressive, then, for every **while** loop, there is a formula of the language that expresses the tightest loop invariant—that fully expresses the behavior of the loop [4, p. 87]. We claim, then, that given a valid assertive program Prog, we can construct from it a related valid program Prog′ that satisfies the conditions of Definition 4.3 above by (1) replacing the postcondition of each internal procedure with the post relation corresponding to the procedure's precondition and its body and (2) replacing each loop invariant with a tightest loop invariant. We will prove the following lemma in Section 4.5.

Lemma 4.4 (Related Valid Program Differing in Assertions Only) *Assuming the assertion language is expressive, if a well-prepared assertive program Prog is valid, then there exists a well-prepared assertive program Prog′ that can be obtained from Prog by (1) replacing the postcondition of each internal procedure with the post relation corresponding to the procedure's precondition and its body and (2) replacing each loop invariant with a tightest loop invariant; furthermore, Prog′ is also valid.*

Relative completeness will follow from Lemma 4.4 if we can show that every such Prog′ can be transformed (using only the rules of the proof system) to a valid mathematical statement. We can ignore the matter of validity for the moment and wonder whether every well-prepared assertive program can be transformed to a mathematical assertion. We answer this question in the affirmative by proving the following lemma in Section 4.5.

Lemma 4.5 *The process of rewriting any well-prepared assertive program in the math direction using the rules of Chapter III always terminates with a mathematical assertion.*

Returning to the matter of validity, we will prove each of the following lemmas in Section 4.5.

Lemma 4.6 *An application of the procedure call rule that involves a call either to (1) a functionally specified external procedure or (2) to an internal procedure, whose postcondition is the post relation corresponding to the procedure's precondition and its body, preserves invalidity in the program direction.*

Lemma 4.7 *An application of the* **loop while** *rule that involves a loop whose tightest loop invariant is expressed in the* **maintaining** *clause preserves invalidity in the program direction.*

Lemma 4.8 *Excluding the procedure call rule and the* **loop while** *rule (which are treated according to Lemmas 4.6 and 4.7), application of each rule of Chapter III preserves invalidity in the program direction.*

Recalling that preserving invalidity in the program direction is equivalent to preserving validity in the math direction, the preceding lemmas establish this one:

Lemma 4.9 *Let Prog′ be a well-prepared valid assertive program that (1) satisfies the conditions of Lemma 4.4 and (2) contains no calls to relationally specified external procedures. Then the process of rewriting Prog′ in the math direction using the rules of Chapter III always terminates with a valid mathematical assertion.*

Theorem 2 follows from Definition 4.3, Lemma 4.4, and Lemma 4.9.

Theorem 2 (Relative Completeness) *If the assertion language is expressive, then the proof system composed of the rules of Chapter III is relatively complete over the set of programs that contain no calls to relationally specified external procedures.*

Note that relative completeness is contingent upon the expressiveness of the assertion language.

The rest of this chapter is devoted to proving the lemmas that establish Theorems 1 and 2. Along the way, we need to state and prove auxiliary lemmas that are important in the proofs of the main lemmas. Because Theorems 1 and 2 form the critical heart of our thesis, we want to be very careful in establishing their truth. By being this careful, we leave very little to the imagination, so we take this opportunity to apologize for the repetitive nature of the proof cases.

4.3 General Auxiliary Lemmas

In this section, we present lemmas that are helpful for proving the lemmas involved in both soundness and relative completeness. We begin with the fact that the proof rules are well-formed.

4.3.1 Proof Rules Are Well-Formed

Recall from Section 3.2 that each of the rules that operates within phase 1, 2, or 3 is defined as being applicable, in the math direction, to some top level code, say $top\_lev\_code_{\mathcal{P}}$, only if there is some instantiation, say Inst, such that $top\_lev\_code_{\mathcal{P}} = \text{Inst}(\mathcal{P})$ and both $\text{Inst}(\mathcal{P})$ and $\text{Inst}(\mathcal{M})$ are syntactically correct top level code. We are thus assured, by definition, that the result of any rule application is (syntactically

correct) top level code. We would do well, however, to reserve judgment on the usefulness of the rules until we were at least convinced that, whenever an instantiation, say Inst, of $\mathcal{P}$ (i.e., Inst($\mathcal{P}$)) is top level code, Inst($\mathcal{M}$) is *also* top level code. Moreover, it is important to know that there exist many instantiations of $\mathcal{P}$ that produce top level code. Concerning application of the rules in the opposite direction—the program direction—we would want to be convinced of the same facts where the roles of $\mathcal{P}$ and $\mathcal{M}$ are interchanged. This collection of facts gives us confidence that the rules are generally applicable, are well-formed. These facts are stated in the following lemmas.

Lemma 4.10 *For each of the rules that operates within phase 1, 2, or 3, whenever an instantiation, say Inst, of $\mathcal{P}$ (i.e., Inst($\mathcal{P}$)) is top level code, Inst($\mathcal{M}$) is also top level code.*

Lemma 4.11 *For each of the rules that operates within phase 1, 2, or 3, whenever an instantiation, say Inst, of $\mathcal{M}$ (i.e., Inst($\mathcal{M}$)) is top level code, Inst($\mathcal{P}$) is also top level code.*

Lemma 4.12 *For each of the rules that operates within phase 1, 2, or 3, there exist infinitely many instantiations of $\mathcal{P}$ that produce top level code, and there exist infinitely many instantiations of $\mathcal{M}$ that produce top level code.*

Complete proofs of these lemmas require examining each of the eleven rules that operates within phase 1, 2, or 3. We provide here a sketch showing how to perform this examination, leaving the complete examination to the skeptical reader. Note that top level code is rewritten from a (possibly empty) finite sequence of nonterminal symbols that occur in a repeated waltz rhythm of three: stow section, then assume-confirm sequence, followed by a guarded block (see Figure 32). This observation will be used frequently. For example, we can show that if there is at least one instantiation of a rule's $\mathcal{P}$ that produces top level code, there are infinitely many such instantiations. Each rule involves either *prec_top_lev_code* or *top_lev_code*. These symbols can be instantiated to any top level code. Now top level code can simply be any arbitrary number of **assume** statements because it can be rewritten from exactly one triple of nonterminals; the stow section and guarded block can be empty; and the assume-confirm sequence can be any arbitrary number of **assume** statements. Because there is no fixed limit on the number of **assume** statements there may be in such a sequence, there are infinitely many ways to instantiate either *prec_top_lev_code* or *top_lev_code*.

As a prototypical example, we argue that there is at least one way to instantiate $\mathcal{P}$ of the rule for procedure call (see Figure 46) to top level code. This can be done if the part that lies strictly between *prec_top_lev_code* and *fol_top_lev_code* can be

instantiated to top level code. Superficially, that is easy because **alter all stow**(i) is a stow section; $ACseq_0$ is an assume-confirm sequence; and the **whenever** statement is a guarded block. We must show additionally that the part within the **whenever** statement can be rewritten from ⟨cd_suffix⟩—that it is a code suffix. Recall from Section 3.1 that a code suffix is a portion of internal code, and that internal code is a pattern of nonterminal symbols, cycling repeatedly through ⟨stow_sec⟩, ⟨ACseq⟩, and ⟨op_stmt⟩, beginning with ⟨stow_sec⟩ and concluding with ⟨ACseq⟩. A code suffix begins and ends with ⟨ACseq⟩. Furthermore, a code suffix must be rewritten according to the restricted grammar productions of Figure 35. Stow sections consist of one **stow** statement, and assume-confirm sequences contain zero or more **confirm** statements.

The code suffix within our **whenever** statement begins with an empty assume-confirm sequence. The operational statement (⟨op_stmt⟩) is a procedure call, and the stow section is **stow**(j). This completes a whole cycle; so, if we instantiate *cd_suffix* to any code suffix, we have instantiated the part within the **whenever** statement to a code suffix. Finally, we observe that an instantiation meeting the above conditions can be found that also satisfies the non-context-free syntactic restrictions by organizing the indexes of **stow** statements to be everywhere increasing, and by arranging that references to index i occur only after **stow**(i). Therefore, there is at least one way to instantiate $\mathcal{P}$ to top level code. Then, by the result of the previous paragraph, there are infinitely many ways to instantiate $\mathcal{P}$ to top level code. We have justified Lemma 4.12 for $\mathcal{P}$ of the procedure call rule.

Does Lemma 4.10 hold for the case of the procedure call rule? Let Inst be an instantiation of $\mathcal{P}$, and suppose Inst($\mathcal{P}$) is top level code. We are to show that Inst($\mathcal{M}$) is top level code. The statement "**confirm** (Br_Cd) $\Rightarrow$ (pre[x $\rightsquigarrow$ ac_i, y $\rightsquigarrow$ ad_i, z $\rightsquigarrow$ z_i])" extends the assume-confirm sequence started by $ACseq_0$. Because Inst($\mathcal{P}$) is top level code, if an indexed variable of Br_Cd has index h, then **stow**(h) occurs earlier in the program than this new **confirm** statement. All variables in pre[x $\rightsquigarrow$ ac_i, y $\rightsquigarrow$ ad_i, z $\rightsquigarrow$ z_i] are indexed by i, and **stow**(i) precedes the **confirm** statement. Therefore, this **confirm** statement satisfies non-context-free syntactic restriction number 2 (see p. 68). This **confirm** statement is followed immediately by an empty guarded block. A stow section, **alter all stow**(j), comes next, and is succeeded by an empty assume-confirm sequence. The guarded block that follows is a **whenever** statement.

We must be sure that the statement sequence within this **whenever** statement is a code suffix. It is; we can take the initial **assume** statement to be the first statement of the assume-confirm sequence with which *cd_suffix* begins. The variables of

this **assume** statement are all indexed by either i or j. So this **assume** statement satisfies non-context-free syntactic restriction number 2 because the statements **stow**(i) and **stow**(j) appear earlier in the program. As the syntax requires, the **whenever** statement is followed by some top level code: *fol_top_lev_code*. Therefore, Lemma 4.10 holds for the case of the procedure call rule.

A similar argument shows Lemma 4.11 to hold for the case of the procedure call rule. These lemmas can be shown to hold for the cases of each of the other ten rules that operate in phases 1, 2, and 3.

We come now to the question whether the rule that governs Step 0 of Figure 31—the bridge rule—is well-formed. The following lemma addresses the most interesting aspect of this question.

Lemma 4.13 *Let $\mathcal{P}$ and $\mathcal{M}$ be as they are defined in the bridge rule. Let Inst be an instantiation of $\mathcal{P}$ such that* p_body *is a procedure body (i.e., can be rewritten from* ⟨*p_body*⟩*). Then there exists an instantiation Inst′ such that Inst′($\mathcal{P}$) = Inst($\mathcal{P}$) and the indexes produced by the relation Stows_added can be chosen so that Inst′($\mathcal{M}$) is top level code.*

Proof. Given this lemma's hypotheses, we show that there exists an instantiation Inst′ such that Inst′($\mathcal{P}$) = Inst($\mathcal{P}$) and the indexes produced by the relation Stows_added can be chosen so that Inst′($\mathcal{M}$) is top level code. The stow section it (Inst′($\mathcal{M}$)) begins with is **alter all stow**(i). The assume-confirm sequence is empty, and it concludes with a non-empty guarded block, a **whenever** statement. We must be sure that the statement sequence within this **whenever** statement is a code suffix. We can take the **assume** statement to be the first statement of the assume-confirm sequence with which Stows_added(*p_body*) begins. Stows_added(*p_body*) needs to be—and can be taken to be—a code kernal (⟨cd_kern⟩). Thus, it concludes with an operational statement (⟨op_stmt⟩). Therefore, it can be followed by a stow section (**stow**(j)) and an assume-confirm sequence (a **confirm** statement) to complete the code suffix. Note that non-context-free syntactic restriction number 2 is satisfied because (1) all the variables in the **assume** statement are indexed by i and **stow**(i) appears earlier in the program and (2) all the variables in the **confirm** statement are indexed by either i or j and both **stow**(i) and **stow**(j) appear earlier in the program. The indexes produced by the relation Stows_added need to be chosen so that non-context-free syntactic restriction number 1 is satisfied. This can be done if j and i are such that $j - i$ is at least as large as the number of operational statements in *p_body*. So we let Inst′ = Inst except for the instantiations of i and j. We assure that Inst′(j) − Inst′(i) is at least as large as the number of operational statements in *p_body*. Finally, we chose the indexes produced by the relation Stows_added to be an increasing sequence of integers strictly between Inst′(i) and Inst′(j). □

Finally, we consider the question whether the rule that governs Step 4 of Figure 31—the rule of inference bridging predicate logic and the indexed method—is well-formed. By definition, H is a syntactically correct assertion in context C if and only if **confirm** H is a syntactically correct **confirm** statement in context C. The one remaining question is whether **confirm** H is syntactically correct top level code. It is. It begins with an empty stow section, which is followed by an assume-confirm sequence that contains exactly one **confirm** statement. It concludes with an empty guarded block. We are now justified in believing the rules of the indexed method to be well-formed.

4.3.2 Factoring Statement Sequences

The applicability of the proof rules is defined with respect to the syntax of top level code. However, the meaning of any assertive program—in particular, the meaning of any top level code—is defined with respect to the syntax of Section 2.1. According to this syntax, each assertive program—including top level code—is simply a sequence of statements. Some of these statements contain other statement sequences, and so on. It is this simpler syntax that is relevant to arguments about preserving invalidity in the math or program directions because these arguments are based on the meanings—the semantics—of the original and resulting assertive programs. In these arguments, we will be analyzing the semantics of assertive programs as concatenations of two or more statement sequences. Thus, it is important that we know, as the following lemma states, that the interpretation of the concatenation of two sequences of statements is the interpretation of the second sequence in the environment resulting from interpreting the first sequence.

Lemma 4.14 (Factoring)

$$\mathcal{I}(SS1\ SS2)(env) = \mathcal{I}(SS2)(\mathcal{I}(SS1)(env)) \tag{4.4}$$

This lemma is easily established from equation 2.23 by induction on the length of SS1.

4.3.3 Shallow Lemmas

The lemmas in this section are shallow in that their proofs are easy; they are not far removed from the definitions. They are, however, convenient for the proofs in Sections 4.4 and 4.5. The first of these lemmas follows easily from equation 2.26 by induction on the length of SS.

Lemma 4.15 (CF Is a Stuck State)

$$AE(env) = CF \Rightarrow AE(\mathcal{I}(SS)(env)) = CF \tag{4.5}$$

Similarly, Lemma 4.16 follows from equation 2.25 by induction on the length of SS.

Lemma 4.16 (VT Is a Stuck State)

$$AE(env) = VT \Rightarrow AE(\mathcal{I}(SS)(env)) = VT \tag{4.6}$$

Suppose S1 is a **loop while** statement and env_{NL} is some neutral environment. Then it is possible for the function $\mathcal{I}$(S1), which is obtained via the minimum fixed point operator, to be undefined at env_{NL}. Then, of course, the composition of any function f with $\mathcal{I}$(S1) is also undefined at env_{NL}. That is to say, if $\mathcal{I}(\text{S1})(\text{env}_{\text{NL}})$ is undefined, then $f(\mathcal{I}(\text{S1})(\text{env}_{\text{NL}}))$ is undefined. Lemma 4.17 depends on this fact.

Lemma 4.17 (Never from Undefined to CF)

$$AE(\mathcal{I}(SS2)(\mathcal{I}(SS1)(env))) = CF \Rightarrow \mathcal{I}(SS1)(env) \text{ is defined.} \tag{4.7}$$

Proof. The assertion that $\text{AE}(\mathcal{I}(\text{SS2})(\mathcal{I}(\text{SS1})(\text{env}))) = \text{CF}$ implies that $\mathcal{I}(\text{SS2})(\mathcal{I}(\text{SS1})(\text{env}))$ is defined. Suppose, by way of contradiction, that $\mathcal{I}(\text{SS1})(\text{env})$ is undefined. Then, because it is a composition of functions, $\mathcal{I}(\text{SS2})(\mathcal{I}(\text{SS1})(\text{env}))$ would be undefined. □

Just as a later environment being categorically false means that all earlier environments must have been defined (Lemma 4.17), we state in Lemma 4.18 that a later environment being categorically false means that no earlier environment was vacuously true.

Lemma 4.18 (Never from VT to CF)

$$AE(\mathcal{I}(SS)(env)) = CF \Rightarrow AE(env) \neq VT \tag{4.8}$$

Proof. Suppose $\text{AE}(\mathcal{I}(\text{SS})(\text{env})) = \text{CF}$. Then $\text{AE}(\mathcal{I}(\text{SS})(\text{env})) \neq \text{VT}$. By the contrapositive of Lemma 4.16, we have $\text{AE}(\text{env}) \neq \text{VT}$, the desired conclusion. □

Lemma 4.19 states that if a later environment is categorically false but the earlier one was not, then the earlier environment was neutral.

Lemma 4.19 (From Non-CF to CF Means From NL to CF)

$$(AE(env) \neq CF \wedge AE(\mathcal{I}(SS)(env)) = CF) \Rightarrow AE(env) = NL \tag{4.9}$$

Proof. Suppose $\text{AE}(\mathcal{I}(\text{SS})(\text{env})) = \text{CF}$. Then, from Lemma 4.17, we have that env is defined. From Lemma 4.18, we have that $\text{AE}(\text{env}) \neq \text{VT}$. From the hypothesis we have that $\text{AE}(\text{env}) \neq \text{CF}$. Therefore, by elimination, we must have $\text{AE}(\text{env}) = \text{NL}$. □

4.3.4 Deeper Lemmas

The proofs of the lemmas in this section are a little more difficult in that most of them involve mathematical induction. Lemma 4.20 tells us that later setups are suffixes of earlier setups.

Lemma 4.20 (Effect on Setup) *If SPE($\mathcal{I}$(SS)(env)) = se, then there exists a setup se_0 such that SPE(env) = $se_0 \circ se$, where $\circ$ is the symbol for sequence concatenation.*

Proof. Checking the semantic definitions, we observe that executions of all statements other than the **alter all** statement leave the setup unchanged. Execution of the **alter all** statement shortens any nonempty setup by removing the front state. Our proof is by induction on the length of SS. The base case is a length of zero; i.e., SS = ε, the empty sequence of statements. In this case, the lemma is satisfied by choosing se_0 to be the empty sequence of states, Λ. For the induction step, suppose $0 < n$ and that the lemma holds for all SSs whose length is less than n. We suppose that the length of the statement sequence S1 SS2 is n, and prove the lemma for S1 SS2. So we suppose SPE($\mathcal{I}$(S1 SS2)(env)) = se. Then, by equation 2.23, we have SPE($\mathcal{I}$(SS2)($\mathcal{I}$(S1)(env))) = se. By the induction hypothesis, there exists se_1 such that SPE($\mathcal{I}$(S1)(env)) = $se_1 \circ$ se. If S1 is not an **alter all** statement or AE($\mathcal{I}$(S1)(env)) $\neq$ NL, then se_0 satisfies the lemma if we take it to equal se_1. If S1 is an **alter all** statement and AE($\mathcal{I}$(S1)(env)) = NL, then se_0 satisfies the lemma if we take it to equal $\langle st \rangle \circ se_1$, where $\langle st \rangle$ is a one-element sequence of some state st. □

The next lemma states that subsequent interpretations of Br_Cd are identical to its initial interpretation.

Lemma 4.21 (Interpretations of Br_Cd Are Stable) *Let $\mathcal{R}$ be an intermediate result obtained by applying the proof rules in the math direction to a programmer-written program. Furthermore, let $\mathcal{R}$ have the following form:*

$$\mathcal{R} \stackrel{\text{def}}{=} C\backslash \begin{array}{l} \textit{prec\_top\_lev\_code} \\ \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ \quad \textit{guarded\_code} \\ \textbf{end whenever} \\ \textit{interm\_top\_lev\_code} \\ \textit{fol\_top\_lev\_code} \end{array} \tag{4.10}$$

Let env be some environment. Then

$$\mathcal{I}(\text{Br\_Cd})(\mathcal{I}\left(\begin{array}{l} \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ \quad \textit{guarded\_code} \\ \textbf{end whenever} \\ \textit{interm\_top\_lev\_code} \end{array}\right)(env)) = \mathcal{I}(\text{Br\_Cd})(env) \tag{4.11}$$

Proof. There are six non-compound statements in the language: procedure call, **confirm**, **assume**, **remember**, **stow**, and **alter all**. Except for **stow**, interpretation of each non-compound statement leaves the index-state unchanged. By the syntactic restriction that indices of **stow** statements are everywhere increasing and Lemma 4.22, which follows, there is an index i such that (1) if **stow**(h) is a statement of *guarded_code* or *interm_top_lev_code*, then $i < h$ and (2) every free variable of Br_Cd is an indexed variable, and every indexed variable ξ_n appearing in Br_Cd is such that $n < i$. Note that interpretation of **stow**(j) changes the index-state only at j. Therefore, for all n such that $n < i$, if S1 is a statement of *guarded_code* or *interm_top_lev_code*, then $\mathrm{ISE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}))(n) = \mathrm{ISE}(\mathrm{env})(n)$. Induction on program structure can be used to show from this fact that, for $n < i$,

$$\mathrm{ISE}(\mathcal{I}\left(\begin{array}{l}\textbf{whenever}\ \mathrm{Br\_Cd}\ \textbf{do}\\ \quad \textit{guarded\_code}\\ \textbf{end whenever}\\ \textit{interm\_top\_lev\_code}\end{array}\right)(\mathrm{env}))(n) = \mathrm{ISE}(\mathrm{env})(n) \tag{4.12}$$

This lemma's conclusion follows immediately. □

Lemma 4.22 (Br_Cds Refer Only to Variables with Earlier Indices) *Let $\mathcal{R}$ be an intermediate result obtained by applying the proof rules in the math direction to a programmer-written program. Furthermore, let $\mathcal{R}$ have the following form:*

$$\begin{array}{lll}\mathcal{R} & \stackrel{\mathrm{def}}{=} & C\backslash\ \textit{prec\_top\_lev\_code}\\ & & \textbf{alter all}\\ & & \textbf{stow}(i)\\ & & \mathit{ACseq}_0\\ & & \textbf{whenever}\ \mathrm{Br\_Cd}\ \textbf{do}\\ & & \quad \textit{guarded\_code}\\ & & \textbf{end whenever}\\ & & \textit{fol\_top\_lev\_code}\end{array} \tag{4.13}$$

Then every free variable of Br_Cd is an indexed variable, and every indexed variable ξ_h appearing in Br_Cd is such that $h < i$.

Proof. We use induction on the number of rule applications to show that every intermediate result obtained by applying the proof rules in the math direction to a programmer-written program satisfies the lemma's conditions. The base case is when exactly one rule is applied. That would have to be the bridge rule. The result of applying the bridge rule has exactly one **whenever** statement. The Br_Cd of this statement is "**true**"; it contains no free variables. So, the conditions of the lemma

are satisfied vacuously. Therefore, any result of applying the bridge rule satisfies the lemma's conditions.

The induction step is to assume that an intermediate result satisfies the lemma's conditions and, then, to show the result of applying each rule in the math direction satisfies the lemma's conditions. We consider each rule in turn.

By the induction hypothesis, $\mathcal{P}$ of the rule for **assume** satisfies the lemma's conditions. The **whenever** statement of $\mathcal{M}$ is in the same position as the one in $\mathcal{P}$ from which it was derived; it follows a sequence of **assume**s and **confirm**s that, in turn, follows an **stow**(i) statement. Furthermore, the **whenever** statement of $\mathcal{M}$ has the same branch condition, "Br_Cd", as the one in $\mathcal{P}$. Hence, $\mathcal{M}$ of the rule for **assume** satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for procedure call satisfies the lemma's conditions. The **whenever** statement of $\mathcal{M}$ follows an empty sequence of **assume**s and **confirm**s that, in turn, follows an **stow**(j) statement. The **whenever** statement of $\mathcal{M}$ has the same branch condition, "Br_Cd", as the one in $\mathcal{P}$. Furthermore, by the syntactic restriction that the indices in **stow** statements are everywhere increasing, $i < j$. By the transitivity of the less-than relation, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < j$. Hence, $\mathcal{M}$ of the rule for procedure call satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with eight statements. Let WE1 stand for the first, and WE2 for the second, of the **whenever** statements. Because $\mathcal{P}$ satisfies the lemma's conditions, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < i$. We also have $i < j$; hence, $h < j$. Boolean program expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of $\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]$ are indexed by i. Therefore, every indexed variable ξ_g appearing in "$(\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])$)$(\mathrm{env}_1)$" is such that $g < j$. Hence, WE1 satisfies the lemma's conditions. Because $i < n$, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < n$. Hence, WE2 satisfies the lemma's conditions. Therefore, $\mathcal{M}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the presence of an **else** clause satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with eleven statements. Let WE1 stand for the first, WE2 for the second, and WE3 for the third, of the **whenever** statements. Because $\mathcal{P}$ satisfies the lemma's conditions, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < i$. We also have $i < j$; hence, $h < j$. Boolean program

expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of $\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]$ are indexed by i. Therefore, every indexed variable ξ_g appearing in "$(\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]))(\mathrm{env}_1)$" is such that $g < j$. Hence, WE1 satisfies the lemma's conditions. We have $i < l$; hence, $h < l$. Therefore, every indexed variable ξ_g appearing in "$(\mathrm{Br\_Cd}) \wedge \neg(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]))(\mathrm{env}_1)$" is such that $g < l$. Hence, WE2 satisfies the lemma's conditions. Because $i < n$, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < n$. Hence, WE2 satisfies the lemma's conditions. Therefore, $\mathcal{M}$ of the rule for selection in the presence of an **else** clause satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the **loop while** rule satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with seven statements. Let WE1 stand for the first, and WE2 for the second, of the **whenever** statements. Because $\mathcal{P}$ satisfies the lemma's conditions, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < i$. We also have $i < j$; hence, $h < j$. Boolean program expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of $\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]$ are indexed by i. Therefore, every indexed variable ξ_g appearing in "$(\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]))(\mathrm{env}_1)$" is such that $g < j$. Hence, WE1 satisfies the lemma's conditions. Because $i < l$, we have that every indexed variable ξ_h appearing in Br_Cd is such that $h < l$. Hence, WE2 satisfies the lemma's conditions. Therefore, $\mathcal{M}$ of the **loop while** rule satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for **confirm** satisfies the lemma's conditions. The **whenever** statement of $\mathcal{M}$ is in the same position as the one in $\mathcal{P}$ from which it was derived; it follows a sequence of **assume**s and **confirm**s that, in turn, follows an **stow**(i) statement. Furthermore, the **whenever** statement of $\mathcal{M}$ has the same branch condition, "Br_Cd", as the one in $\mathcal{P}$. Hence, $\mathcal{M}$ of the rule for **confirm** satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for empty guarded blocks satisfies the lemma's conditions. Every statement of $\mathcal{M}$ is a statement of $\mathcal{P}$. Hence, $\mathcal{M}$ of the rule for empty guarded blocks satisfies the lemma's conditions.

Due to its additional syntactic restriction, neither $\mathcal{P}$ nor $\mathcal{M}$ of the rule for **alter all** contains any **whenever** statements. Hence, $\mathcal{M}$ of the rule for **alter all** (vacuously) satisfies the lemma's conditions.

One argument serves for the remaining three rules: the rule for consecutive **assume** statements, the **assume-confirm** rule, and the rule for consecutive **confirm** statements. By the induction hypothesis, the rule's $\mathcal{P}$ satisfies the lemma's conditions. Every **whenever** statement of $\mathcal{M}$ is a statement of $\mathcal{P}$. Hence, the rule's $\mathcal{M}$ satisfies the lemma's conditions.

The induction step is now complete. Hence the proof of this lemma (4.22) is finished. □

Consider compound statements, such as selection or iteration statements, that may appear within **whenever** statements. These compound statements contain **stow**(h) statements. When a compound statement is still intact within a **whenever** statement, it has yet to be transformed by the application (in the math direction) of a proof rule. Therefore, no variable that is subscripted by h appears anywhere in the program. We state this fact in Lemma 4.23.

Lemma 4.23 (Internal Indices Are Sealed) *Let $\mathcal{R}$ be an intermediate result obtained by applying the proof rules in the math direction to a programmer-written program. Furthermore, let $\mathcal{R}$ have the following form:*

$$\begin{array}{lll} \mathcal{R} & \stackrel{\text{def}}{=} & C\backslash \quad \textit{prec\_top\_lev\_code} \\ & & \quad \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ & & \quad\quad \textit{guarded\_code} \\ & & \quad \textbf{end whenever} \\ & & \quad \textit{fol\_top\_lev\_code} \end{array} \tag{4.14}$$

Let CST be any compound statement of guarded_code. Let CI be the set of indices h that appear in **stow**(h) *statements within CST. Let S1 be any statement (including statements within compound statements) of prec_top_lev_code, guarded_code, cd_suffix, or fol_top_lev_code. Then, if indexed variable ξ_h occurs in S1, $h \notin CI$.*

Proof. We use induction on the number of rule applications to show that every intermediate result obtained by applying the proof rules in the math direction to a programmer-written program satisfies the lemma's conditions. The base case is when exactly one rule is applied. That would have to be the bridge rule. Because *p_body* is programmer-written, it contains no subscripted variables. By the definition of the Stows_added relation, Stows_added(*p_body*) contains no subscripted variables. Hence, the result of applying the bridge rule, $\mathcal{M}$, has subscripted variables only in two statements. These variables are subscripted only by i and j. By the syntactic restriction that the indices in **stow** statements are everywhere increasing, no compound statement of Stows_added(*p_body*) can contain either a **stow**(i) statement or a **stow**(j) statement. Therefore, any result of applying the bridge rule satisfies the lemma's conditions.

The induction step is to assume that an intermediate result satisfies the lemma's conditions and, then, to show the result of applying each rule in the math direction satisfies the lemma's conditions. We consider each rule in turn.

Application of some of the rules introduces no new indexed variables into $\mathcal{M}$. These rules are: the rule for **assume**, the rule for empty guarded blocks, the rule for

alter all, the rule for consecutive **assume** statements, the **assume-confirm** rule, and the rule for consecutive **confirm** statements. By the induction hypothesis, the rule's $\mathcal{P}$ satisfies the lemma's conditions. Hence, the rule's $\mathcal{M}$ satisfies the lemma's conditions. We consider each of the remaining rules in turn.

By the induction hypothesis, $\mathcal{P}$ of the rule for procedure call satisfies the lemma's conditions. The only indexed variables introduced into $\mathcal{M}$ are indexed by i and j, but neither **stow**(i) nor **stow**(j) is within a compound statement. Hence, $\mathcal{M}$ of the rule for procedure call satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions. The only indexed variables introduced into $\mathcal{M}$ are indexed by i, j, k, or n, but none of **stow**(i), **stow**(j), **stow**(k), or **stow**(n) is within a compound statement of $\mathcal{M}$ (even though **stow**(j) and **stow**(k) were within a compound statement of $\mathcal{P}$). Hence, $\mathcal{M}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the presence of an **else** clause satisfies the lemma's conditions. The only indexed variables introduced into $\mathcal{M}$ are indexed by i, j, k, l, m, or n, but none of **stow**(i), **stow**(j), **stow**(k), **stow**(l), **stow**(m), or **stow**(n) is within a compound statement of $\mathcal{M}$ (even though **stow**(j), **stow**(k), **stow**(l), and **stow**(m) were within a compound statement of $\mathcal{P}$). Hence, $\mathcal{M}$ of the rule for selection in the presence of an **else** clause satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the **loop while** rule satisfies the lemma's conditions. The only indexed variables introduced into $\mathcal{M}$ are indexed by i, j, k, or l, but none of **stow**(i), **stow**(j), **stow**(k), or **stow**(l) is within a compound statement of $\mathcal{M}$ (even though **stow**(j) and **stow**(k) were within a compound statement of $\mathcal{P}$). Hence, $\mathcal{M}$ of the **loop while** rule satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for **confirm** satisfies the lemma's conditions. The only indexed variable introduced into $\mathcal{M}$ is indexed by i, but **stow**(i) is not within a compound statement. Hence, $\mathcal{M}$ of the rule for **confirm** satisfies the lemma's conditions.

The induction step is now complete. Hence the proof of this lemma (4.23) is finished. □

The old state and the declaration meanings of an environment remain unchanged when a statement that may appear in top level code is interpreted. We state and prove this fact below in Lemma 4.25. Lemma 4.24 states that the interpretation of sequences of such statements also preserves both the old state and the declaration meanings. That is to say, interpretation of top level code preserves both the old state and the declaration meanings.

Lemma 4.24 (Sequences of Statements That Preserve os and d) *Let env be an environment. If, for any environment env′, OSE($\mathcal{I}$(S1)(env′)) = OSE(env′) and DME($\mathcal{I}$(S1)(env′)) = DME(env′) for any statement S1 of SS for which $\mathcal{I}$(S1) is defined at env′, then, if $\mathcal{I}$(SS) is defined at env, OSE($\mathcal{I}$(SS)(env)) = OSE(env) and DME($\mathcal{I}$(SS)(env)) = DME(env).*

Proof. We use induction on the length of SS. The base case is a length of zero; i.e., SS = ε, the empty sequence of statements. Because the interpretation of the empty sequence of statements is the identity function, $\mathcal{I}$(SS)(env) = env. Substitution of equals then yields OSE($\mathcal{I}$(SS)(env)) = OSE(env) and DME($\mathcal{I}$(SS)(env)) = DME(env); we have finished the base case. For the induction step, suppose $0 < n$ and that the lemma holds for all SSs whose length is less than n. We suppose that the length of the statement sequence ST1 SS2 is n, and prove the lemma for ST1 SS2. Let env be an arbitrary environment. Suppose $\mathcal{I}$(ST1 SS2) is defined at env. Further suppose, given any environment env′, that OSE($\mathcal{I}$(S1)(env′)) = OSE(env′) and DME($\mathcal{I}$(S1)(env′)) = DME(env′) for any statement S1 of ST1 SS2 for which $\mathcal{I}$(S1) is defined at env′. Then, given any environment env″, OSE($\mathcal{I}$(S1)(env″)) = OSE(env″) and DME($\mathcal{I}$(S1)(env″)) = DME(env″) for any statement S1 of SS2 for which $\mathcal{I}$(S1) is defined at env″. By equation 2.23, $\mathcal{I}$(ST1) is defined at env, and $\mathcal{I}$(SS2) is defined at $\mathcal{I}$(ST1)(env). Also, by equation 2.23, OSE($\mathcal{I}$(ST1 SS2)(env)) = OSE($\mathcal{I}$(SS2)($\mathcal{I}$(ST1)(env))). Thus, we have OSE($\mathcal{I}$(ST1)(env)) = OSE(env) and DME($\mathcal{I}$(ST1)(env)) = DME(env). Setting env″ = $\mathcal{I}$(ST1)(env), we have OSE($\mathcal{I}$(SS2)($\mathcal{I}$(ST1)(env))) = OSE($\mathcal{I}$(ST1)(env)). Therefore, the chain of equalities yields OSE($\mathcal{I}$(ST1 SS2)(env)) = OSE(env). Identical reasoning gives DME($\mathcal{I}$(ST1 SS2)(env)) = DME(env), concluding the induction step and the proof. □

Lemma 4.25 (Statements Within Top Level Code Preserve os and d) *Let S1 be any statement that can occur within top level code. (Top level code is defined in Section 3.1 and summarized in Figure 32. The statement S1 may be a statement of the top-level statement sequence, or it may, recursively, be a statement within a compound statement. The restrictions of Section 3.1 assure that S1 is not a* **remember** *statement.) If env is any environment at which $\mathcal{I}$(S1) is defined, then OSE($\mathcal{I}$(S1)(env)) = OSE(env) and DME($\mathcal{I}$(S1)(env)) = DME(env).*

Proof. If AE(env) ≠ NL, then $\mathcal{I}$(S1)(env) = env. Substitution of equals gives OSE($\mathcal{I}$(S1)(env)) = OSE(env) and DME($\mathcal{I}$(S1)(env)) = DME(env). Therefore, for the remainder of this proof, we assume AE(env) = NL. We use induction on the structure of statement S1. In the base case, S1 is one of the five non-compound statements: procedure call, **confirm**, **assume**, **stow**, and **alter all**. The

definitions of the interpretations of these five statements yield immediately that $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$.

In the induction step, S1 is one of the four compound statements: **if-then**, **if-then-else**, **loop while**, and **whenever**. The induction hypothesis is that the lemma is true for each statement within S1. In the following treatments of each of the four compound statements, we suppose that env is any environment at which $\mathcal{I}(\mathrm{S1})$ is defined.

Let S1 be the statement "**if** *b_p_e* **then** SS **end if**".
Case $\mathcal{I}(\mathrm{b\_p\_e})(env) =$ **false**. In this case, $\mathcal{I}(\mathrm{S1})(\mathrm{env}) = \mathrm{env}$. Substitution of equals gives $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$.
Case $\mathcal{I}(\mathrm{b\_p\_e})(env) =$ **true**. In this case, $\mathcal{I}(\mathrm{S1})(\mathrm{env}) = \mathcal{I}(\mathrm{SS})(\mathrm{env})$. By the induction hypothesis, if ST1 is a statement of SS, env′ is some environment, and $\mathcal{I}(\mathrm{ST1})$ is defined at env′, then $\mathrm{OSE}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{OSE}(\mathrm{env}')$ and $\mathrm{DME}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{DME}(\mathrm{env}')$. Then, by Lemma 4.24, $\mathrm{OSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{SS})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$. Substitution of equals gives $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$.

Let S1 be the statement "**if** *b_p_e* **then** SS1 **else** SS2 **end if**".
Case $\mathcal{I}(\mathrm{b\_p\_e})(env) =$ **true**. In this case, $\mathcal{I}(\mathrm{S1})(\mathrm{env}) = \mathcal{I}(\mathrm{SS1})(\mathrm{env})$. By the induction hypothesis, if ST1 is a statement of SS1, env′ is some environment, and $\mathcal{I}(\mathrm{ST1})$ is defined at env′, then $\mathrm{OSE}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{OSE}(\mathrm{env}')$ and $\mathrm{DME}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{DME}(\mathrm{env}')$. Then, by Lemma 4.24, $\mathrm{OSE}(\mathcal{I}(\mathrm{SS1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{SS1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$. Substitution of equals gives $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$.
Case $\mathcal{I}(\mathrm{b\_p\_e})(env) =$ **false**. In this case, $\mathcal{I}(\mathrm{S1})(\mathrm{env}) = \mathcal{I}(\mathrm{SS2})(\mathrm{env})$. By the induction hypothesis, if ST1 is a statement of SS2, env′ is some environment, and $\mathcal{I}(\mathrm{ST1})$ is defined at env′, then $\mathrm{OSE}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{OSE}(\mathrm{env}')$ and $\mathrm{DME}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{DME}(\mathrm{env}')$. Then, by Lemma 4.24, $\mathrm{OSE}(\mathcal{I}(\mathrm{SS2})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{SS2})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$. Substitution of equals gives $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$.

Let S1 be the statement "**loop maintaining** Inv **while** *b_p_e* **do** SS **end loop**". By the induction hypothesis, if ST1 is a statement of SS, env′ is some environment, and $\mathcal{I}(\mathrm{ST1})$ is defined at env′, then $\mathrm{OSE}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{OSE}(\mathrm{env}')$ and $\mathrm{DME}(\mathcal{I}(\mathrm{ST1})(\mathrm{env}')) = \mathrm{DME}(\mathrm{env}')$. Then, by Lemma 4.24, $\mathrm{OSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\mathcal{I}(\mathrm{SS})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$. In the first three cases of the definition of Λ_{W}, $\mathrm{OSE}(\Lambda_{\mathrm{W}}(\mathrm{WL})(\mathrm{env})) = \mathrm{OSE}(\mathrm{env})$ and $\mathrm{DME}(\Lambda_{\mathrm{W}}(\mathrm{WL})(\mathrm{env})) = \mathrm{DME}(\mathrm{env})$. Because $\mathcal{I}(\mathrm{S1})$ is defined at env, $\mathrm{MFP}(\Lambda_{\mathrm{W}})$ is defined at env. Therefore, because $\mathrm{MFP}(\Lambda_{\mathrm{W}})$ is a fixed point of Λ_{W}, $\mathrm{OSE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{env})) =$

OSE(env) and DME(MFP(Λ_W)(env)) = DME(env). By Lemma 4.26 (to follow), OSE($\mathcal{I}$(S1)(env)) = OSE(env) and DME($\mathcal{I}$(S1)(env)) = DME(env).

Let S1 be the statement "**whenever** Br_Cd **do** SS **end whenever**".
Case $\mathcal{I}(Br\_Cd)(env)$ = **false**. In this case, $\mathcal{I}$(S1)(env) = env. Substitution of equals gives OSE($\mathcal{I}$(S1)(env)) = OSE(env) and DME($\mathcal{I}$(S1)(env)) = DME(env).
Case $\mathcal{I}(Br\_Cd)(env)$ = **true**. In this case, $\mathcal{I}$(S1)(env) = $\mathcal{I}$(SS)(env). By the induction hypothesis, if ST1 is a statement of SS, env′ is some environment, and $\mathcal{I}$(ST1) is defined at env′, then OSE($\mathcal{I}$(ST1)(env′)) = OSE(env′) and DME($\mathcal{I}$(ST1)(env′)) = DME(env′). Then, by Lemma 4.24, OSE($\mathcal{I}$(SS)(env)) = OSE(env) and DME($\mathcal{I}$(SS)(env)) = DME(env). Substitution of equals gives OSE($\mathcal{I}$(S1)(env)) = OSE(env) and DME($\mathcal{I}$(S1)(env)) = DME(env).

The induction step and the proof are now complete. □

Lemma 4.26 *Let F be a function from environments to environments such that OSE(F(env)) = OSE(env) and DME(F(env)) = DME(env). Then OSE(Fgt(F(Rem(env)))) = OSE(env) and DME(Fgt(F(Rem(env)))) = DME(env).*

Proof. By the definitions of Rem and Fgt, for any environment env′, DME(Rem(env′)) = DME(env′) and DME(Fgt(env′)) = DME(env′). Therefore, DME(Fgt(F(Rem(env)))) = DME(F(Rem(env))) = DME(Rem(env)) = DME(env). Let ξ be an arbitrary current variable name. Let k be an arbitrary natural number. Then $\#^{k+1}\xi$ is an arbitrary old variable name. By definition of Fgt, OSE(Fgt(F(Rem(env))))($\#^{k+1}\xi$) = OSE(F(Rem(env)))($\#\#^{k+1}\xi$) = OSE(F(Rem(env)))($\#^{k+2}\xi$). By the lemma's hypothesis concerning F, OSE(F(Rem(env)))($\#^{k+2}\xi$) = OSE(Rem(env))($\#^{k+2}\xi$). By definition of Rem, OSE(Rem(env))($\#^{k+2}\xi$) = OSE(env)($\#^{k+1}\xi$). The chain of equalities gives us OSE(Fgt(F(Rem(env))))($\#^{k+1}\xi$) = OSE(env)($\#^{k+1}\xi$). Therefore, OSE(Fgt(F(Rem(env)))) = OSE(env). □

Lemma 4.27 follows immediately from Lemmas 4.24 and 4.25.

Lemma 4.27 (Statement Sequences Preserve os and d) *Let env be an environment. Let SS be a sequence of statements, every one of which can occur within top level code. (SS contains no* **remember** *statements.) If $\mathcal{I}(SS)$ is defined at env, then OSE($\mathcal{I}(SS)$(env)) = OSE(env) and DME($\mathcal{I}(SS)$(env)) = DME(env).*

Definition 4.4 *A statement sequence SS is* unaffected by old state *(or* unaffected by os*) if and only if for any two environments that differ at most in their old states, say*

$$env_1 = [a, cs, os_1, ns, se, d] \tag{4.15}$$

$$env_2 = [a, cs, os_2, ns, se, d], \tag{4.16}$$

1. *$\mathcal{I}(SS)(env_1)$ is defined if and only if $\mathcal{I}(SS)(env_2)$ is defined, and*

2. *if $\mathcal{I}(SS)(env_1)$ is defined, say*

$$\mathcal{I}(SS)(env_1) = [a', cs', os_1, ns', se', d], \tag{4.17}$$

then

$$\mathcal{I}(SS)(env_2) = [a', cs', os_2, ns', se', d]. \tag{4.18}$$

Lemma 4.28 (Sequences of Statements That Are Unaffected by os) *If each statement S1 of statement sequence SS is unaffected by old state, then SS is unaffected by old state.*

Proof. We use induction on the length of SS. The base case is a length of zero; i.e., SS $= \varepsilon$, the empty sequence of statements. Let env_1 and env_2 be defined as in Definition 4.4. Because the interpretation of the empty sequence of statements is the identity function, both $\mathcal{I}(\text{SS})(\text{env}_1)$ and $\mathcal{I}(\text{SS})(\text{env}_2)$ are defined, $\mathcal{I}(\text{SS})(\text{env}_1) = \text{env}_1$, and $\mathcal{I}(\text{SS})(\text{env}_2) = \text{env}_2$. Because $\mathcal{I}(\text{SS})(\text{env}_2)$ satisfies the condition of Definition 4.4, SS is unaffected by old state. For the induction step, suppose $0 < n$ and that the lemma holds for all SSs whose length is less than n. We suppose that the length of the statement sequence ST1 SS2 is n, and prove the lemma for ST1 SS2. Let env_1 and env_2 be defined as in Definition 4.4. Because ST1 is not affected by old state, $\mathcal{I}(\text{ST1})(\text{env}_1)$ is defined if and only if $\mathcal{I}(\text{ST1})(\text{env}_2)$ is defined. By equation 2.23 and the composition of functions, if $\mathcal{I}(\text{ST1})(\text{env}_1)$ is not defined, then neither $\mathcal{I}(\text{ST1 SS2})(\text{env}_1)$ nor $\mathcal{I}(\text{ST1 SS2})(\text{env}_2)$ is defined. Suppose $\mathcal{I}(\text{ST1})(\text{env}_1)$ is defined, say

$$\mathcal{I}(\text{ST1})(\text{env}_1) = [\text{a}', \text{cs}', \text{os}_1, \text{ns}', \text{se}', \text{d}]. \tag{4.19}$$

Then

$$\mathcal{I}(\text{ST1})(\text{env}_2) = [\text{a}', \text{cs}', \text{os}_2, \text{ns}', \text{se}', \text{d}]. \tag{4.20}$$

By the induction hypothesis, SS2 is unaffected by old state. Therefore, $\mathcal{I}(\text{SS2})(\mathcal{I}(\text{ST1})(\text{env}_1))$ is defined if and only if $\mathcal{I}(\text{SS2})(\mathcal{I}(\text{ST1})(\text{env}_2))$ is defined. Suppose $\mathcal{I}(\text{SS2})(\mathcal{I}(\text{ST1})(\text{env}_1))$ is defined, say

$$\mathcal{I}(\text{SS2})(\mathcal{I}(\text{ST1})(\text{env}_1)) = [\text{a}'', \text{cs}'', \text{os}_1, \text{ns}'', \text{se}'', \text{d}]. \tag{4.21}$$

Then

$$\mathcal{I}(\text{SS2})(\mathcal{I}(\text{ST1})(\text{env}_2)) = [\text{a}'', \text{cs}'', \text{os}_2, \text{ns}'', \text{se}'', \text{d}]. \tag{4.22}$$

By equation 2.23, ST1 SS2 is unaffected by old state. □

Lemma 4.29 (Top Level Code Is Unaffected by os) *Each statement that can occur within top level code is unaffected by old state.*

Proof. Let S1 be a statement that can occur within top level code. If AE(env) $\neq$ NL, then $\mathcal{I}$(S1)(env) = env. In this case, S1 satisfies the definition of being unaffected by old state. Therefore, for the remainder of this proof, we assume AE(env) = NL. We use induction on the structure of statement S1. In the base case, because S1 is not a **remember** statement, S1 is one of the five non-compound statements: procedure call, **confirm**, **assume**, **stow**, and **alter all**. In top level code, **confirm** and **assume** statements do not contain old variables; i.e., none of the statements of top level code have the form **confirm** ⟨old_assert⟩ or **assume** ⟨old_assert⟩. Therefore, examination of the definitions of the interpretations of these five statements reveals that S1 is unaffected by old state.

In the induction step, S1 is one of the four compound statements: **if-then**, **if-then-else**, **loop while**, and **whenever**. The induction hypothesis is that each statement within S1 is unaffected by old state. The arguments for the **if-then-else** and **whenever** statements are similar to the argument for the **if-then** statement. We give here only the arguments for the **if-then** and **loop while** statements. Let env_1 and env_2 be defined as in Definition 4.4.

Let S1 be the statement "**if** *b_p_e* **then** SS **end if**".
Case $\mathcal{I}(\text{b\_p\_e})(env_1)$ = **false**. Because *b_p_e* contains no old variables, $\mathcal{I}(b\_p\_e)(\text{env}_2)$ = **false**. Therefore, both $\mathcal{I}(\text{S1})(\text{env}_1)$ and $\mathcal{I}(\text{S1})(\text{env}_2)$ are defined, $\mathcal{I}(\text{S1})(\text{env}_1) = \text{env}_1$, and $\mathcal{I}(\text{S1})(\text{env}_2) = \text{env}_2$. Because $\mathcal{I}(\text{S1})(\text{env}_2)$ satisfies the condition of Definition 4.4, S1 is unaffected by old state.
Case $\mathcal{I}(\text{b\_p\_e})(env_1)$ = **true**. Because *b_p_e* contains no old variables, $\mathcal{I}(b\_p\_e)(\text{env}_2)$ = **true**. Therefore, $\mathcal{I}(\text{S1})(\text{env}_1) = \mathcal{I}(\text{SS})(\text{env}_1)$ and $\mathcal{I}(\text{S1})(\text{env}_2) = \mathcal{I}(\text{SS})(\text{env}_2)$. By the induction hypothesis and Lemma 4.28, SS is unaffected by old state. By substitution of equals in Definition 4.4, S1 is unaffected by old state.

Let S1 be the statement "**loop maintaining** Inv **while** *b_p_e* **do** SS **end loop**". By the induction hypothesis and Lemma 4.28, SS is unaffected by old state; *b_p_e* contains no old variables. Each old variable in Inv contains exactly one # sign. Let ξ be an arbitrary current variable name. By definition of function Rem, $\text{OSE}(\text{Rem}(\text{env}_1))(\#\xi) = \text{cs}(\xi) = \text{OSE}(\text{Rem}(\text{env}_2))(\#\xi)$. Therefore, if $\text{MFP}(\Lambda_{\text{W}})(\text{Rem}(\text{env}_1))$ is not defined, then $\text{MFP}(\Lambda_{\text{W}})(\text{Rem}(\text{env}_2))$ is not defined. Furthermore, if $\text{MFP}(\Lambda_{\text{W}})(\text{Rem}(\text{env}_1))$ is defined, say

$$\text{MFP}(\Lambda_{\text{W}})(\text{Rem}(\text{env}_1)) = [\text{a}', \text{cs}', \text{os}'_1, \text{ns}', \text{se}', \text{d}], \tag{4.23}$$

then

$$\text{MFP}(\Lambda_{\text{W}})(\text{Rem}(\text{env}_2)) = [\text{a}', \text{cs}', \text{os}'_2, \text{ns}', \text{se}', \text{d}]. \tag{4.24}$$

Hence,

$$\mathrm{Fgt}(\mathrm{MFP}(\Lambda_W)(\mathrm{Rem}(\mathrm{env}_1))) = [\mathrm{a}', \mathrm{cs}', \mathrm{os}_1, \mathrm{ns}', \mathrm{se}', \mathrm{d}] \quad (4.25)$$

$$\mathrm{Fgt}(\mathrm{MFP}(\Lambda_W)(\mathrm{Rem}(\mathrm{env}_2))) = [\mathrm{a}', \mathrm{cs}', \mathrm{os}_2, \mathrm{ns}', \mathrm{se}', \mathrm{d}]. \quad (4.26)$$

Therefore, S1 is unaffected by old state.

The induction step and the proof are now complete. □

Lemma 4.30 follows immediately from Lemmas 4.28 and 4.29.

Lemma 4.30 (Statement Sequences Are Unaffected by os) *If SS is a sequence of statements, every one of which can occur within top level code, then SS is unaffected by old state.*

Interpretation of internal code—statement sequences internal to **whenever**, selection, or iteration statements—is unaffected by the index state or the setup. Lemmas 4.31 and 4.32 capture this fact.

Lemma 4.31 *Let S1 be a statement that may appear in internal code. Let* env_1 *and* env_2 *be two environments. If* $\mathcal{I}(S1)$ *is defined at* env_1, $AE(env_1) = AE(env_2)$, $CSE(env_1) = CSE(env_2)$, $OSE(env_1) = OSE(env_2)$, *and* $DME(env_1) = DME(env_2)$, *then* $\mathcal{I}(S1)$ *is defined at* env_2, $AE(\mathcal{I}(S1)(env_1)) = AE(\mathcal{I}(S1)(env_2))$, $CSE(\mathcal{I}(S1)(env_1)) = CSE(\mathcal{I}(S1)(env_2))$, $OSE(\mathcal{I}(S1)(env_1)) = OSE(\mathcal{I}(S1)(env_2))$, *and* $DME(\mathcal{I}(S1)(env_1)) = DME(\mathcal{I}(S1)(env_2))$.

Proof. According to Section 3.1 (especially Figure 34), S1 may be **stow**(⟨nat_num⟩), **confirm** ⟨cur_assert⟩, or an operational statement. If $\mathrm{AE}(\mathrm{env}_1) \neq \mathrm{NL}$, then $\mathrm{AE}(\mathrm{env}_2) \neq \mathrm{NL}$. We would then have $\mathcal{I}(\mathrm{S1})(\mathrm{env}_1) = \mathrm{env}_1$ and $\mathcal{I}(\mathrm{S1})(\mathrm{env}_2) = \mathrm{env}_2$. The lemma follows immediately by substitution of equals.

Therefore, for the remainder of this proof, we assume $\mathrm{AE}(\mathrm{env}_1) = \mathrm{NL}$. (Hence, $\mathrm{AE}(\mathrm{env}_2) = \mathrm{NL}$.) We use induction on the structure of statement S1. In the base case, S1 is one of the three non-compound statements that can occur in internal code: procedure call, **confirm**, and **stow**.

Let S1 be a procedure call. By the semantics of procedure call, $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{OSE}(\mathrm{env}_1) = \mathrm{OSE}(\mathrm{env}_2) = \mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$ and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{DME}(\mathrm{env}_1) = \mathrm{DME}(\mathrm{env}_2) = \mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Again, by the semantics of procedure call, because $\mathrm{CSE}(\mathrm{env}_1) = \mathrm{CSE}(\mathrm{env}_2)$ and $\mathrm{DME}(\mathrm{env}_1) = \mathrm{DME}(\mathrm{env}_2)$, $\mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$ and $\mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Hence, the lemma holds for procedure call.

Let S1 be **confirm** Q. Because Q is a current assertion—an assertion that contains only current variables—$\mathrm{CSE}(\mathrm{env}_1) = \mathrm{CSE}(\mathrm{env}_2)$ implies $\mathcal{I}(\mathrm{Q})(\mathrm{env}_1) = \mathcal{I}(\mathrm{Q})(\mathrm{env}_2)$.

Therefore, by the semantics of keyConfirm Q, $\mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Again, by the semantics of keyConfirm Q, $\mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{CSE}(\mathrm{env}_1) = \mathrm{CSE}(\mathrm{env}_2) = \mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{OSE}(\mathrm{env}_1) = \mathrm{OSE}(\mathrm{env}_2) = \mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{DME}(\mathrm{env}_1) = \mathrm{DME}(\mathrm{env}_2) = \mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Hence, the lemma holds for **confirm** Q.

Let S1 be **stow**(i). By the semantics of **stow**(i), $\mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{AE}(\mathrm{env}_1) = \mathrm{AE}(\mathrm{env}_2) = \mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, $\mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{CSE}(\mathrm{env}_1) = \mathrm{CSE}(\mathrm{env}_2) = \mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{OSE}(\mathrm{env}_1) = \mathrm{OSE}(\mathrm{env}_2) = \mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{DME}(\mathrm{env}_1) = \mathrm{DME}(\mathrm{env}_2) = \mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Hence, the lemma holds for **stow**(i).

In the induction step, S1 is one of the three compound statements that can occur in internal code: **if-then**, **if-then-else**, and **loop while**. The induction hypothesis is that the lemma is true for each statement within S1.

Let S1 be one of the two selection statements. Because all of *b_p_e*'s variables are current variables and $\mathrm{CSE}(\mathrm{env}_1) = \mathrm{CSE}(\mathrm{env}_2)$, $\mathcal{I}(b\_p\_e)(\mathrm{env}_1) = \mathcal{I}(b\_p\_e)(\mathrm{env}_2)$. So, without loss of generality, there is a statement sequence SS′ such that $\mathcal{I}(\mathrm{S1})(\mathrm{env}_1) = \mathcal{I}(\mathrm{SS}')(\mathrm{env}_1)$ and $\mathcal{I}(\mathrm{S1})(\mathrm{env}_2) = \mathcal{I}(\mathrm{SS}')(\mathrm{env}_2)$. SS′ is either the empty statement sequence or a sequence of statements within S1. If it is the empty sequence, the lemma follows immediately from the semantics of the empty sequence and substitution of equals. If it is a sequence of statements within S1, the lemma follows from the induction hypothesis and a simple induction on the length of SS′. Hence, the lemma holds if S1 is a selection statement.

Let S1 be the statement "**loop maintaining** Inv **while** *b_p_e* **do** SS **end loop**". Because $\mathcal{I}(\mathrm{S1})$ is defined at env_1, $\mathrm{MFP}(\Lambda_{\mathrm{W}})$ is defined at $\mathrm{Rem}(\mathrm{env}_1)$. Therefore, if env is some environment resulting from zero or more iterations of the loop beginning in environment $\mathrm{Rem}(\mathrm{env}_1)$, then $\mathcal{I}(\mathrm{SS})$ is defined at env. By the induction hypothesis and a simple induction on the length of SS, if env_3 and env_4 are two environments such that $\mathcal{I}(\mathrm{SS})$ is defined at env_3, $\mathrm{AE}(\mathrm{env}_3) = \mathrm{AE}(\mathrm{env}_4)$, $\mathrm{CSE}(\mathrm{env}_3) = \mathrm{CSE}(\mathrm{env}_4)$, $\mathrm{OSE}(\mathrm{env}_3) = \mathrm{OSE}(\mathrm{env}_4)$, and $\mathrm{DME}(\mathrm{env}_3) = \mathrm{DME}(\mathrm{env}_4)$, then $\mathcal{I}(\mathrm{SS})$ is defined at env_4, $\mathrm{AE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_3)) = \mathrm{AE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_4))$, $\mathrm{CSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_3)) = \mathrm{CSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_4))$, $\mathrm{OSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_3)) = \mathrm{OSE}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_4))$, and $\mathrm{DME}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_3)) = \mathrm{DME}(\mathcal{I}(\mathrm{SS})(\mathrm{env}_4))$. Under the same assumptions about env_3 and env_4, because the variables in Inv are only old variables and current variables and the variables in *b_p_e* are only current variables, $\mathcal{I}(\mathrm{Inv})(\mathrm{env}_3) = \mathcal{I}(\mathrm{Inv})(\mathrm{env}_4)$ and $\mathcal{I}(b\_p\_e)(\mathrm{env}_3) = \mathcal{I}(b\_p\_e)(\mathrm{env}_4)$. Now $\mathrm{AE}(\mathrm{Rem}(\mathrm{env}_1)) = \mathrm{AE}(\mathrm{Rem}(\mathrm{env}_2))$, $\mathrm{CSE}(\mathrm{Rem}(\mathrm{env}_1)) = \mathrm{CSE}(\mathrm{Rem}(\mathrm{env}_2))$, $\mathrm{OSE}(\mathrm{Rem}(\mathrm{env}_1)) = \mathrm{OSE}(\mathrm{Rem}(\mathrm{env}_2))$, and $\mathrm{DME}(\mathrm{Rem}(\mathrm{env}_1)) = \mathrm{DME}(\mathrm{Rem}(\mathrm{env}_2))$. Therefore, $\mathrm{MFP}(\Lambda_{\mathrm{W}})$ is defined at $\mathrm{Rem}(\mathrm{env}_2)$, $\mathrm{AE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_1))) = \mathrm{AE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_2)))$,

$\mathrm{CSE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_1))) = \mathrm{CSE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_2)))$, $\mathrm{OSE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_1))) = \mathrm{OSE}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_2)))$, and $\mathrm{DME}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_1))) = \mathrm{DME}(\mathrm{MFP}(\Lambda_{\mathrm{W}})(\mathrm{Rem}(\mathrm{env}_2)))$. By the definition of the function Fgt, $\mathcal{I}(\mathrm{S1})$ is defined at env_2, $\mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{AE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, $\mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{CSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, $\mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{OSE}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$, and $\mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_1)) = \mathrm{DME}(\mathcal{I}(\mathrm{S1})(\mathrm{env}_2))$. Hence, the lemma holds if S1 is a **loop-while** statement. The induction step and the proof are now finished. □

Lemma 4.32 (Interpretation of Internal Code) *Let SS be internal code. Let env_1 and env_2 be two environments. If $\mathcal{I}(SS)$ is defined at env_1, $AE(env_1) = AE(env_2)$, $CSE(env_1) = CSE(env_2)$, $OSE(env_1) = OSE(env_2)$, and $DME(env_1) = DME(env_2)$, then $\mathcal{I}(SS)$ is defined at env_2, $AE(\mathcal{I}(SS)(env_1)) = AE(\mathcal{I}(SS)(env_2))$, $CSE(\mathcal{I}(SS)(env_1)) = CSE(\mathcal{I}(SS)(env_2))$, $OSE(\mathcal{I}(SS)(env_1)) = OSE(\mathcal{I}(SS)(env_2))$, and $DME(\mathcal{I}(SS)(env_1)) = DME(\mathcal{I}(SS)(env_2))$.*

Proof. This lemma follows easily from Lemma 4.31 by induction on the length of SS. □

4.3.5 Negative-Branch-Condition Independence

The negative-branch-condition independence lemma is key for our ability to establish that each rule preserves invalidity in the math direction. The interpretations of the original and the result of a rule application in a given environment do not always result in equal environments. Recall that the rules are constructed not to preserve semantics but merely to preserve invalidity. The negative-branch-condition independence lemma states that interpretation of a particular section of code beginning in environments that are "nearly" equal results in environments that are "nearly" equal. The definition of Equal_except_at establishes what it means for two environments to be "nearly" equal.

Definition 4.5 *Equal_except_at is a predicate.*

$$Equal\_except\_at : (\wp(Integers) \times Environments \times Environments) \rightarrow Boolean \quad (4.27)$$

Equal_except_at(A, env_1, env_2) *if and only if all of the following hold:*

$$AE(env_1) = AE(env_2) \quad (4.28)$$

$$CSE(env_1) = CSE(env_2) \quad (4.29)$$

$$h \notin A \Rightarrow ISE(env_1)(h) = ISE(env_2)(h) \quad (4.30)$$

$$SPE(env_1) = SPE(env_2) \quad (4.31)$$

$$OSE(env_1) = OSE(env_2) \quad (4.32)$$

$$DME(env_1) = DME(env_2) \quad (4.33)$$

Furthermore, if $\mathcal{I}(SS)$ is undefined at env_1 and $\mathcal{I}(SS)$ is undefined at env_2, then Equal_except_at$(A, \mathcal{I}(SS)(env_1), \mathcal{I}(SS)(env_2))$ where SS is a statement sequence.

Two environments are "nearly" equal if (1) all their components except possibly their index-states are equal and (2) their index-states agree everywhere except on a certain set of integers. In all uses we make of this definition, the set of integers, A, is finite. The interpretations of a statement sequence in two environments are "nearly" equal if (1) both interpretations are defined and "nearly" equal or (2) both interpretations are undefined.

If the interpretations of a statement in two environments that are nearly equal are always nearly equal as well, we say that the statement is *uncritical* of that notion of near equality. Lemma 4.33 states that the interpretations of a statement sequence in two environments that are nearly equal remain nearly equal, assuming each statement of the sequence is uncritical of that notion of near equality.

Lemma 4.33 (Sequences of Uncritical Statements) *Let SN be a set of integers and SS a statement sequence. Let env_1 and env_2 be two environments. If Equal_except_at(SN, env_1, env_2) implies Equal_except_at$(SN, \mathcal{I}(S1)(env_1), \mathcal{I}(S1)(env_2))$ for any statement S1 of SS, then Equal_except_at$(SN, \mathcal{I}(SS)(env_3), \mathcal{I}(SS)(env_4))$ for any two environments env_3 and env_4 such that Equal_except_at(SN, env_3, env_4).*

Proof. We use induction on the length of SS. The base case is a length of zero; i.e., SS $= \varepsilon$, the empty sequence of statements. Because the interpretation of the empty sequence of statements is the identity function, $\mathcal{I}$(SS)(env) = env. So, because Equal_except_at(SN, env_3, env_4), Equal_except_at(SN, $\mathcal{I}$(SS)(env_3), $\mathcal{I}$(SS)(env_4)). For the induction step, suppose $0 < n$ and that the lemma holds for all SSs whose length is less than n. We suppose that the length of the statement sequence ST1 SS2 is n, and prove the lemma for ST1 SS2. Suppose that Equal_except_at(SN, env_1, env_2) implies Equal_except_at(SN, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)) for any statement S1 of ST1 SS2. Then Equal_except_at(SN, env_1, env_2) implies Equal_except_at(SN, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)) for any statement S1 of SS2. Equal_except_at(SN, env_1, env_2) also implies Equal_except_at(SN, $\mathcal{I}$(ST1)(env_1), $\mathcal{I}$(ST1)(env_2)). By the induction hypothesis, Equal_except_at(SN, $\mathcal{I}$(SS2)(env_3), $\mathcal{I}$(SS2)(env_4)) for any two environments env_3 and env_4 such that Equal_except_at(SN, env_3, env_4). This last relation is satisfied if we let $\text{env}_3 = \mathcal{I}$(ST1)($\text{env}_1$) and $\text{env}_4 = \mathcal{I}$(ST1)($\text{env}_2$). So, Equal_except_at(SN, $\mathcal{I}$(SS2)($\mathcal{I}$(ST1)(env_1)), $\mathcal{I}$(SS2)($\mathcal{I}$(ST1)(env_2))) for any two environments env_1 and env_2 such that Equal_except_at(SN, env_1, env_2). By the definition of the interpretation of a sequence of statements, we have

Equal_except_at(SN, $\mathcal{I}$(ST1 SS2)(env$_1$), $\mathcal{I}$(ST1 SS2)(env$_2$)) for any two environments env$_1$ and env$_2$ such that Equal_except_at(SN, env$_1$, env$_2$). By a change of two variables, that is Equal_except_at(SN, $\mathcal{I}$(ST1 SS2)(env$_3$), $\mathcal{I}$(ST1 SS2)(env$_4$)) for any two environments env$_3$ and env$_4$ such that Equal_except_at(SN, env$_3$, env$_4$). The induction step and the proof are now complete. □

Lemma 4.34 states that a statement is uncritical of a certain notion of near equality if the statement has no variables indexed by the integers where the environments differ.

Lemma 4.34 (Uninvolved Indices) *Let SN be a set of integers. Let S1 be a statement that contains no variables indexed by an integer in SN. That is to say, if ξ_h is an indexed variable that occurs in S1, then $h \notin$ SN. Let SB $\subseteq$ SN. Let env$_1$ and env$_2$ be two environments. If Equal_except_at(SB, env$_1$, env$_2$), then Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)).*

Proof. If AE(env$_1$) $\neq$ NL, then, because Equal_except_at(SB, env$_1$, env$_2$), AE(env$_2$) $\neq$ NL. We would then have $\mathcal{I}$(S1)(env$_1$) = env$_1$ and $\mathcal{I}$(S1)(env$_2$) = env$_2$. Then, by substitution of equals, if Equal_except_at(SB, env$_1$, env$_2$), then Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)).

Therefore, for the remainder of this proof, we assume AE(env$_1$) = NL. We use induction on the structure of statement S1. In the base case, S1 is one of the six non-compound statements: procedure call, **confirm**, **assume**, **remember**, **stow**, and **alter all**. We assume the lemma's hypotheses to prove Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)).

Let S1 be a procedure call. Not Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)) only if env$_1$ and env$_2$ differ in their assert status, their current state, or their declaration meanings. But they do not differ in those places because Equal_except_at(SB, env$_1$, env$_2$). Hence, Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)).

Let S1 be a **confirm** or **assume** statement. Not Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)) only if env$_1$ and env$_2$ differ in their assert status, their current state, their old state, or their index state at an integer h such that, for some ξ, ξ_h is a variable of S1. However, if h is such that, for some ξ, ξ_h is a variable of S1, then $h \notin$ SN. Hence $h \notin$ SB. Because Equal_except_at(SB, env$_1$, env$_2$), env$_1$ and env$_2$ do not differ in their assert status, their current state, their old state, or their index state at h. Therefore, Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)).

Let S1 be a **remember** statement. Not Equal_except_at(SB, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)) only if env$_1$ and env$_2$ differ

in their assert status, their current state, or their old state. But they do not differ in those places because Equal_except_at(SB, env_1, env_2). Hence, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

Let S1 be a **stow** statement. Not Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)) only if env_1 and env_2 differ in their assert status, their current state, or their index state at some integer $i \notin$ SB. But they do not differ in those places because Equal_except_at(SB, env_1, env_2). Hence, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

Let S1 be an **alter all** statement. Not Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)) only if env_1 and env_2 differ in their assert status or their setup. But they do not differ in those places because Equal_except_at(SB, env_1, env_2). Hence, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

In the induction step, S1 is one of the four compound statements: **if-then**, **if-then-else**, **loop while**, and **whenever**. The induction hypothesis is that the lemma is true for each statement within S1. We assume the lemma's hypotheses to prove Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

Let S1 be the statement "**if** *b_p_e* **then** SS **end if**". There are no indexed variables in *b_p_e*. Hence, because Equal_except_at(SB, env_1, env_2), $\mathcal{I}$(*b_p_e*)(env_1) = $\mathcal{I}$(*b_p_e*)(env_2).
Case $\mathcal{I}$(b_p_e)(env_1) = **false**. Then $\mathcal{I}$(*b_p_e*)(env_2) = **false**. Hence, we have $\mathcal{I}$(S1)(env_1) = env_1 and $\mathcal{I}$(S1)(env_2) = env_2. Therefore, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).
Case $\mathcal{I}$(b_p_e)(env_1) = **true**. Then $\mathcal{I}$(*b_p_e*)(env_2) = **true**. Hence, we have $\mathcal{I}$(S1)(env_1) = $\mathcal{I}$(SS)(env_1) and $\mathcal{I}$(S1)(env_2) = $\mathcal{I}$(SS)(env_2). By the induction hypothesis, if ST1 is a statement of SS, then Equal_except_at(SB, $\mathcal{I}$(ST1)(env_1), $\mathcal{I}$(ST1)(env_2)). By Lemma 4.33, Equal_except_at(SB, $\mathcal{I}$(SS)(env_1), $\mathcal{I}$(SS)(env_2)). Therefore, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

Let S1 be the statement "**if** *b_p_e* **then** SS1 **else** SS2 **end if**". There are no indexed variables in *b_p_e*. Hence, because Equal_except_at(SB, env_1, env_2), $\mathcal{I}$(*b_p_e*)(env_1) = $\mathcal{I}$(*b_p_e*)(env_2).
Case $\mathcal{I}$(b_p_e)(env_1) = **true**. Then $\mathcal{I}$(*b_p_e*)(env_2) = **true**. Hence, we have $\mathcal{I}$(S1)(env_1) = $\mathcal{I}$(SS1)(env_1) and $\mathcal{I}$(S1)(env_2) = $\mathcal{I}$(SS1)(env_2). By the induction hypothesis, if ST1 is a statement of SS1, then Equal_except_at(SB, $\mathcal{I}$(ST1)(env_1), $\mathcal{I}$(ST1)(env_2)). By Lemma 4.33, Equal_except_at(SB, $\mathcal{I}$(SS1)(env_1), $\mathcal{I}$(SS1)(env_2)). Therefore, Equal_except_at(SB, $\mathcal{I}$(S1)(env_1), $\mathcal{I}$(S1)(env_2)).

Case $\mathcal{I}(\text{b\_p\_e})(env_1) = \textbf{false}$. Then $\mathcal{I}(b\_p\_e)(\text{env}_2) = \textbf{false}$. Hence, we have $\mathcal{I}(\text{S1})(\text{env}_1) = \mathcal{I}(\text{SS2})(\text{env}_1)$ and $\mathcal{I}(\text{S1})(\text{env}_2) = \mathcal{I}(\text{SS2})(\text{env}_2)$. By the induction hypothesis, if ST1 is a statement of SS2, then $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{ST1})(\text{env}_1), \mathcal{I}(\text{ST1})(\text{env}_2))$. By Lemma 4.33, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{SS2})(\text{env}_1), \mathcal{I}(\text{SS2})(\text{env}_2))$. Therefore, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{S1})(\text{env}_1), \mathcal{I}(\text{S1})(\text{env}_2))$.

Let S1 be the statement "**loop maintaining** Inv **while** *b_p_e* **do** SS **end loop**". Let env_3 and env_4 be any two environments such that $\text{Equal\_except\_at}(\text{SB}, \text{env}_3, \text{env}_4)$. By the definitions of the functions Rem and Fgt, $\text{Equal\_except\_at}(\text{SB}, \text{Rem}(\text{env}_3), \text{Rem}(\text{env}_4))$ and $\text{Equal\_except\_at}(\text{SB}, \text{Fgt}(\text{env}_3), \text{Fgt}(\text{env}_4))$. Therefore, we have shown $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{S1})(\text{env}_1), \mathcal{I}(\text{S1})(\text{env}_2))$ and are done, when we show $\text{Equal\_except\_at}(\text{SB}, \text{MFP}(\Lambda_{\text{W}})(\text{env}_3), \text{MFP}(\Lambda_{\text{W}})(\text{env}_4))$. Neither Inv nor *b_p_e* contain indexed variables. Therefore, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{Inv})(\text{env}_3), \mathcal{I}(\text{Inv})(\text{env}_4))$ and $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(b\_p\_e)(\text{env}_3), \mathcal{I}(b\_p\_e)(\text{env}_4))$. By the induction hypothesis and Lemma 4.33, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{SS})(\text{env}_3), \mathcal{I}(\text{SS})(\text{env}_4))$. Because $\text{MFP}(\Lambda_{\text{W}})$ is a fixed point of Λ_{W}, $\text{MFP}(\Lambda_{\text{W}})$ is defined at env_3 if and only if $\text{MFP}(\Lambda_{\text{W}})$ is defined at env_4, and $\text{Equal\_except\_at}(\text{SB}, \text{MFP}(\Lambda_{\text{W}})(\text{env}_3), \text{MFP}(\Lambda_{\text{W}})(\text{env}_4))$; we are finished with the **loop while** statement.

Let S1 be the statement "**whenever** Br_Cd **do** SS **end whenever**". If ξ_h is an indexed variable that occurs in Br_Cd, then $h \notin \text{SN}$. Hence, because $\text{Equal\_except\_at}(\text{SB}, \text{env}_1, \text{env}_2)$ and $\text{SB} \subseteq \text{SN}$, $\mathcal{I}(\text{Br\_Cd})(\text{env}_1) = \mathcal{I}(\text{Br\_Cd})(\text{env}_2)$.
Case $\mathcal{I}(Br\_Cd)(env_1) = \textbf{false}$. Then $\mathcal{I}(\text{Br\_Cd})(\text{env}_2) = \textbf{false}$. Hence, we have $\mathcal{I}(\text{S1})(\text{env}_1) = \text{env}_1$ and $\mathcal{I}(\text{S1})(\text{env}_2) = \text{env}_2$. Therefore, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{S1})(\text{env}_1), \mathcal{I}(\text{S1})(\text{env}_2))$.
Case $\mathcal{I}(Br\_Cd)(env_1) = \textbf{true}$. Then $\mathcal{I}(\text{Br\_Cd})(\text{env}_2) = \textbf{true}$. Hence, we have $\mathcal{I}(\text{S1})(\text{env}_1) = \mathcal{I}(\text{SS})(\text{env}_1)$ and $\mathcal{I}(\text{S1})(\text{env}_2) = \mathcal{I}(\text{SS})(\text{env}_2)$. By the induction hypothesis, if ST1 is a statement of SS, then $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{ST1})(\text{env}_1), \mathcal{I}(\text{ST1})(\text{env}_2))$. By Lemma 4.33, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{SS})(\text{env}_1), \mathcal{I}(\text{SS})(\text{env}_2))$. Therefore, $\text{Equal\_except\_at}(\text{SB}, \mathcal{I}(\text{S1})(\text{env}_1), \mathcal{I}(\text{S1})(\text{env}_2))$.

The induction step and the proof are now complete. □

The negative-branch-condition independence lemma is a consequence of Lemma 4.35, which we are able to prove by induction on the number of rule applications.

Lemma 4.35 *Let $\mathcal{R}$ be an intermediate result obtained by applying the proof rules in the math direction to a programmer-written program. Furthermore, let $\mathcal{R}$ have the*

following form:

$$\mathcal{R} \stackrel{\text{def}}{=} C\backslash \begin{array}{l} \textit{prec\_top\_lev\_code} \\ \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ \quad \textit{guarded\_code} \\ \textbf{end whenever} \\ \textit{fol\_top\_lev\_code} \end{array} \tag{4.34}$$

Let GI be the set of indices h that appear in **stow**(h) *statements anywhere in guarded_code. Let $GS \subseteq GI$. Let env_1 and env_2 be two environments. Let S1 be any statement of fol_top_lev_code. If Equal_except_at(GS, env_1, env_2) and $\neg\mathcal{I}(\text{Br\_Cd})(env_1)$, then Equal_except_at$(GS, \mathcal{I}(S1)(env_1), \mathcal{I}(S1)(env_2))$. If $h \in GI$ and ξ_h occurs in S1, then each of the following is true. S1 is not within a* **whenever** *statement. S1 is either an* **assume** *statement or a* **confirm** *statement.*

Proof. We use induction on the number of rule applications to show that every intermediate result obtained by applying the proof rules in the math direction to a programmer-written program satisfies the lemma's conditions. The base case is when exactly one rule is applied. That would have to be the bridge rule. The result of applying the bridge rule is a three-statement program; the last statement is the only **whenever** statement. In this case, then, *fol_top_lev_code* is empty. Therefore, the lemma's statement S1 does not exist. So, the conditions of the lemma are satisfied vacuously. Therefore, any result of applying the bridge rule satisfies the lemma's conditions.

The induction step is to assume that an intermediate result satisfies the lemma's conditions and, then, to show the result of applying each rule in the math direction satisfies the lemma's conditions. The **whenever** statement of the lemma could be chosen to occur in *prec_top_lev_code* of the rule; it could be chosen to be one of the **whenever** statements explicitly mentioned in the rule; or it could be chosen to occur in *fol_top_lev_code* of the rule. In the case that the **whenever** statement of the lemma occurs in *fol_top_lev_code* of the rule, the induction hypothesis ensures that the result of the rule's application also satisfies the lemma's conditions—because the rule makes no changes to *fol_top_lev_code*. Therefore, as we examine each rule in turn, we need to consider just two cases:

1. the **whenever** statement of the lemma occurs in *prec_top_lev_code* of the rule, and

2. the **whenever** statement of the lemma is one of the **whenever** statements explicitly mentioned in the rule.

In the sequel, we refer to these cases by number. Now we examine each rule in turn.

By the induction hypothesis, $\mathcal{P}$ of the rule for **assume** satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with an **assume** statement and a **whenever** statement. Let ST1 stand for the **assume** statement: **assume** (Br_Cd) $\Rightarrow$ (H). Let ST2 stand for the **whenever** statement.

Case 1. We must show that both ST1 and ST2 satisfy the conditions the lemma places on S1. The statement "**assume** H" of $\mathcal{P}$ is within a **whenever** statement; therefore, by the induction hypothesis, if ξ_h occurs in H, then $h \notin$ GI. Because it appears as the condition of a **whenever** statement, and not in an **assume** or **confirm** statement, by the induction hypothesis, if ξ_h occurs in Br_Cd, then $h \notin$ GI. Therefore, if ξ_h occurs in ST1, then $h \notin$ GI. Suppose env_1 and env_2 satisfy the lemma's assumptions. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(ST1)(env_1), $\mathcal{I}$(ST1)(env_2)). Hence, the lemma's conditions are satisfied for ST1. Because ST2 is derived from a **whenever** statement of $\mathcal{P}$ by removing one **assume** statement, by the induction hypothesis, if ξ_h occurs in ST2, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(ST2)(env_1), $\mathcal{I}$(ST2)(env_2)). Hence, the lemma's conditions are satisfied for ST2. Therefore, all the lemma's conditions are satisfied for this case.

Case 2. The set of indices that appear in **stow**s in ST2 is the same as the set for the **whenever** statement of $\mathcal{P}$ from which it was derived. So, by the induction hypothesis, all the lemma's conditions are satisfied for this case. Hence, $\mathcal{M}$ of the rule for **assume** satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for procedure call satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with a **confirm** statement, an **alter all** statement, a **stow**(j) statement, and a **whenever** statement. Let ST1 stand for the **confirm** statement: **confirm** (Br_Cd) $\Rightarrow$ (pre[x $\rightsquigarrow$ ac_i, y $\rightsquigarrow$ ad_i, z $\rightsquigarrow$ z_i]). Let ST2 stand for the **whenever** statement.

Case 1. We must show that ST1, **alter all**, **stow**(j), and ST2 satisfy the conditions the lemma places on S1. By reasons given in the preceding paragraph, if ξ_h occurs in Br_Cd, then $h \notin$ GI. The preconditions and postconditions of procedures do not contain indexed variables. In particular, pre contains no indexed variables. Hence, all variables in pre[x $\rightsquigarrow$ ac_i, y $\rightsquigarrow$ ad_i, z $\rightsquigarrow$ z_i] are indexed by i. Because **stow**(i) is not within a **whenever** statement of *prec_top_lev_code*, $i \notin$ GI. Therefore, if ξ_h occurs in ST1, then $h \notin$ GI. ST2 is derived from a **whenever** statement of $\mathcal{P}$ by replacing a procedure call and a **stow**(j) statement with an **assume** statement all of whose variables are indexed by either i or j. Because **stow**(j) is not within a **whenever** statement

of *prec_top_lev_code*, $j \notin$ GI. Because $\{i, j\} \cap \text{GI} = \emptyset$, by the induction hypothesis, if ξ_h occurs in ST2, then $h \notin$ GI. Neither **alter all** nor **stow**(j) contains any indexed variables. Let S1 be any of ST1, **alter all**, **stow**(j), or ST2. Then, if ξ_h occurs in S1, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)). Therefore, all the lemma's conditions are satisfied for this case.

Case 2. The set of indices that appear in **stow**s in ST2 is a subset of the set for the **whenever** statement of $\mathcal{P}$ from which it was derived. So, by the induction hypothesis and the definition of subset, all the lemma's conditions are satisfied for this case. Hence, $\mathcal{M}$ of the rule for procedure call satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with eight statements. Let WE1 stand for the first, and WE2 for the second, of the **whenever** statements. Let SU1 stand for the first, and SU2 for the second, of the **assume** statements.

Case 1. We must show that all eight statements satisfy the conditions the lemma places on S1. We do so by showing that if S1 is one of these eight statements, then, if ξ_h occurs in S1, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)). Therefore, all the lemma's conditions would be satisfied for this case. Let S1 be one of **stow**(j), **stow**(n), or **alter all**. Then, because S1 contains no indexed variables, we have, vacuously, that, if ξ_h occurs in S1, then $h \notin$ GI. Because it appears as the condition of a **whenever** statement, and not in an **assume** or **confirm** statement, by the induction hypothesis, if ξ_h occurs in Br_Cd, then $h \notin$ GI. Boolean program expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of MExp(*b_p_e*)[y $\rightsquigarrow$ y$_i$] are indexed by i. Because none of **stow**(i), **stow**(j), **stow**(k), or **stow**(n) occur in *prec_top_lev_code*, $\{i, j, k, n\} \cap \text{GI} = \emptyset$. All variables of the statement "**assume** x$_j$ = x$_i$" are indexed by i or j. All other parts of WE1 are derived from statements within a selection statement, which, in turn, is within a **whenever** statement of $\mathcal{P}$. For these reasons and the induction hypothesis, if ξ_h occurs in WE1, then $h \notin$ GI. WE2 is derived from a **whenever** statement of $\mathcal{P}$ by removing a selection statement and a **stow**(n) statement. So, by the induction hypothesis, if ξ_h occurs in WE2, then $h \notin$ GI. Let S1 be one of SU1 or SU2. Then, because SU1 and SU2 are composed of Br_Cd, MExp(*b_p_e*)[y $\rightsquigarrow$ y$_i$], and, respectively, x$_n$ = x$_k$ and x$_n$ = x$_i$, if ξ_h occurs in S1, then $h \notin$ GI. We have checked all eight statements. Therefore, all the lemma's conditions are satisfied for this case.

Case 2. We divide this case into two subcases: (a) WE1 is the **whenever** statement of the lemma and (b) WE2 is the **whenever** statement of the lemma.

Case 2(a). The set, GW1, of indices that appear in **stow**s in WE1 is a subset

of the set for the **whenever** statement of $\mathcal{P}$ that contains the selection statement from which WE1 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE1. By Lemma 4.22, if ξ_h occurs in Br_Cd, then $h \notin$ GW1. By Lemma 4.23, if ξ_h occurs in *cd_suffix*, then $h \notin$ GW1. Hence, if ξ_h occurs in WE2, then $h \notin$ GW1. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(WE2)(env$_1$), $\mathcal{I}$(WE2)(env$_2$)), where GS $\subseteq$ GW1. Therefore, WE2 satisfies the lemma's conditions for WE1. Because they have no indexed variables, **alter all** and **stow**(n) both satisfy the lemma's conditions for WE1. If ξ_h occurs in SU2, then $h \notin$ GW1. So, SU2 satisfies the lemma's conditions for WE1. We are left to consider SU1. Note that $k \in$ GW1, and there are values of ξ (current variable names) such that ξ_k occurs in SU1. We must show, therefore, that SU1 is not within a **whenever** statement; it is not. We must also show that SU1 is either an **assume** statement or a **confirm** statement; it is an **assume** statement. Finally, suppose GS $\subseteq$ GW1; Equal_except_at(GS, env$_1$, env$_2$); and $\neg\mathcal{I}$((Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y$_i$]))(env$_1$). We must show Equal_except_at(GS, $\mathcal{I}$(SU1)(env$_1$), $\mathcal{I}$(SU1)(env$_2$)). We noted above that if ξ_h occurs in Br_Cd, then $h \notin$ GW1. So, because $i \notin$ GW1, if ξ_h occurs in (Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y$_i$]), then $h \notin$ GW1. Therefore, $\mathcal{I}$((Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y$_i$]))(env$_2$) = $\mathcal{I}$((Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y$_i$]))(env$_1$) = **false**. So, $\mathcal{I}$(SU1)(env$_1$) = env$_1$ and $\mathcal{I}$(SU1)(env$_2$) = env$_2$. Therefore, Equal_except_at(GS, $\mathcal{I}$(SU1)(env$_1$), $\mathcal{I}$(SU1)(env$_2$)), and we are done.
Case 2(b). The set, GW2, of indices that appear in **stow**s in WE2 (i.e., the **stow**s of *cd_suffix*) is a subset of the set for the **whenever** statement of $\mathcal{P}$ from which WE2 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE2.
Hence, $\mathcal{M}$ of the rule for selection in the absence of an **else** clause satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for selection in the presence of an **else** clause satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with eleven statements. Let WE1 stand for the first, WE2 for the second, and WE3 for the third, of the **whenever** statements. Let SU1 stand for the first, and SU2 for the second, of the **assume** statements.
Case 1. We must show that all eleven statements satisfy the conditions the lemma places on S1. We do so by showing that if S1 is one of these eleven statements, then, if ξ_h occurs in S1, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)). Therefore, all the lemma's conditions would be satisfied for this case. Let S1 be one of **stow**(j), **stow**(l), **stow**(n), or **alter all**. Then, because S1 contains no indexed variables, we have, vacuously, that,

if ξ_h occurs in S1, then $h \notin$ GI. Because it appears as the condition of a **whenever** statement, and not in an **assume** or **confirm** statement, by the induction hypothesis, if ξ_h occurs in Br_Cd, then $h \notin$ GI. Boolean program expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of MExp(*b_p_e*)[y $\rightsquigarrow$ y_i] are indexed by i. Because none of **stow**(i), **stow**(j), **stow**(k), **stow**(l), **stow**(m), or **stow**(n) occur in *prec_top_lev_code*, $\{i, j, k, l, m, n\} \cap \text{GI} = \emptyset$. All variables of the statement "**assume** $x_j = x_i$" are indexed by i or j. All other parts of WE1 are derived from statements within a selection statement, which, in turn, is within a **whenever** statement of $\mathcal{P}$. For these reasons and the induction hypothesis, if ξ_h occurs in WE1, then $h \notin$ GI. All variables of the statement "**assume** $x_l = x_i$" are indexed by i or l. All other parts of WE2 are derived from statements within a selection statement, which, in turn, is within a **whenever** statement of $\mathcal{P}$. For these reasons and the induction hypothesis, if ξ_h occurs in WE2, then $h \notin$ GI. WE3 is derived from a **whenever** statement of $\mathcal{P}$ by removing a selection statement and a **stow**(n) statement. So, by the induction hypothesis, if ξ_h occurs in WE3, then $h \notin$ GI. Let S1 be one of SU1 or SU2. Then, because SU1 and SU2 are composed of Br_Cd, MExp(*b_p_e*)[y $\rightsquigarrow$ y_i], and, respectively, $x_n = x_k$ and $x_n = x_m$, if ξ_h occurs in S1, then $h \notin$ GI. We have checked all eleven statements. Therefore, all the lemma's conditions are satisfied for this case.

Case 2. We divide this case into three subcases: (a) WE1 is the **whenever** statement of the lemma; (b) WE2 is the **whenever** statement of the lemma; and (c) WE3 is the **whenever** statement of the lemma.

Case 2(a). The set, GW1, of indices that appear in **stow**s in WE1 is a subset of the set for the **whenever** statement of $\mathcal{P}$ that contains the selection statement from which WE1 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE1. By Lemma 4.22, if ξ_h occurs in Br_Cd, then $h \notin$ GW1. By Lemma 4.23, if ξ_h occurs in *cd_suffix*, then $h \notin$ GW1. Hence, if ξ_h occurs in WE3, then $h \notin$ GW1. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(WE3)(env$_1$), $\mathcal{I}$(WE3)(env$_2$)), where GS $\subseteq$ GW1. Therefore, WE3 satisfies the lemma's conditions for WE1. If ξ_h occurs in (Br_Cd) $\wedge \neg$(MExp(*b_p_e*)[y $\rightsquigarrow$ y_i]), then $h \notin$ GW1. By the syntactic restriction that indexes in **stow**s are everywhere increasing and Lemma 4.23, if ξ_h occurs within WE2, then $h \notin$ GW1. Hence, if ξ_h occurs in WE2, then $h \notin$ GW1. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(WE2)(env$_1$), $\mathcal{I}$(WE2)(env$_2$)), where GS $\subseteq$ GW1. Therefore, WE2 satisfies the lemma's conditions for WE1. Because they have no indexed variables, **alter all**, **stow**(l), and **stow**(n) all satisfy the lemma's conditions for WE1. If ξ_h occurs in SU2, then $h \notin$ GW1. So, SU2 satisfies the lemma's conditions for WE1. We are left to consider SU1. Note

that $k \in$ GW1, and there are values of ξ (current variable names) such that ξ_k occurs in SU1. We must show, therefore, that SU1 is not within a **whenever** statement; it is not. We must also show that SU1 is either an **assume** statement or a **confirm** statement; it is an **assume** statement. Finally, suppose GS $\subseteq$ GW1; Equal_except_at(GS, env_1, env_2); and $\neg\mathcal{I}((\text{Br\_Cd}) \wedge (\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_1)$. We must show Equal_except_at(GS, $\mathcal{I}(\text{SU1})(\text{env}_1), \mathcal{I}(\text{SU1})(\text{env}_2)$). We noted above that if ξ_h occurs in Br_Cd, then $h \notin$ GW1. So, because $i \notin$ GW1, if ξ_h occurs in (Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y_i]), then $h \notin$ GW1. Therefore, $\mathcal{I}((\text{Br\_Cd}) \wedge (\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_2) = \mathcal{I}((\text{Br\_Cd}) \wedge (\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_1) =$ **false**. So, $\mathcal{I}(\text{SU1})(\text{env}_1) = \text{env}_1$ and $\mathcal{I}(\text{SU1})(\text{env}_2) = \text{env}_2$. Therefore, Equal_except_at(GS, $\mathcal{I}(\text{SU1})(\text{env}_1), \mathcal{I}(\text{SU1})(\text{env}_2)$), and we are done.

Case 2(b). The set, GW2, of indices that appear in **stow**s in WE2 is a subset of the set for the **whenever** statement of $\mathcal{P}$ that contains the selection statement from which WE2 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE2. By Lemma 4.22, if ξ_h occurs in Br_Cd, then $h \notin$ GW2. By Lemma 4.23, if ξ_h occurs in *cd_suffix*, then $h \notin$ GW2. Hence, if ξ_h occurs in WE3, then $h \notin$ GW2. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}(\text{WE3})(\text{env}_1), \mathcal{I}(\text{WE3})(\text{env}_2)$), where GS $\subseteq$ GW2. Therefore, WE3 satisfies the lemma's conditions for WE2. Because they have no indexed variables, **alter all** and **stow**(n) both satisfy the lemma's conditions for WE2. If ξ_h occurs in SU1, then $h \notin$ GW2. So, SU1 satisfies the lemma's conditions for WE2. We are left to consider SU2. Note that $m \in$ GW2, and there are values of ξ (current variable names) such that ξ_m occurs in SU2. We must show, therefore, that SU2 is not within a **whenever** statement; it is not. We must also show that SU2 is either an **assume** statement or a **confirm** statement; it is an **assume** statement. Finally, suppose GS $\subseteq$ GW2; Equal_except_at(GS, env_1, env_2); and $\neg\mathcal{I}((\text{Br\_Cd}) \wedge \neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_1)$. We must show Equal_except_at(GS, $\mathcal{I}(\text{SU2})(\text{env}_1), \mathcal{I}(\text{SU2})(\text{env}_2)$). We noted above that if ξ_h occurs in Br_Cd, then $h \notin$ GW2. So, because $i \notin$ GW2, if ξ_h occurs in (Br_Cd) $\wedge \neg$(MExp(b_p_e)[y $\rightsquigarrow$ y_i]), then $h \notin$ GW2. Therefore, $\mathcal{I}((\text{Br\_Cd}) \wedge \neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_2) = \mathcal{I}((\text{Br\_Cd}) \wedge \neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_1) =$ **false**. So, $\mathcal{I}(\text{SU2})(\text{env}_1) = \text{env}_1$ and $\mathcal{I}(\text{SU2})(\text{env}_2) = \text{env}_2$. Therefore, Equal_except_at(GS, $\mathcal{I}(\text{SU2})(\text{env}_1), \mathcal{I}(\text{SU2})(\text{env}_2)$), and we are done.

Case 2(c). The set, GW3, of indices that appear in **stow**s in WE3 (i.e., the **stow**s of *cd_suffix*) is a subset of the set for the **whenever** statement of $\mathcal{P}$ from which WE3 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE3.

Hence, $\mathcal{M}$ of the rule for selection in the presence of an **else** clause satisfies the

lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the **loop while** rule satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with seven statements. Let WE1 stand for the first, and WE2 for the second, of the **whenever** statements.

Case 1. We must show that all seven statements satisfy the conditions the lemma places on S1. We do so by showing that if S1 is one of these seven statements, then, if ξ_h occurs in S1, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(S1)(env$_1$), $\mathcal{I}$(S1)(env$_2$)). Therefore, all the lemma's conditions would be satisfied for this case. Let S1 be one of **stow**(j), **stow**(l), or **alter all**. Then, because S1 contains no indexed variables, we have, vacuously, that, if ξ_h occurs in S1, then $h \notin$ GI. Because it appears as the condition of a **whenever** statement, and not in an **assume** or **confirm** statement, by the induction hypothesis, if ξ_h occurs in Br_Cd, then $h \notin$ GI. Boolean program expressions, in particular *b_p_e*, do not contain indexed variables. So all the variables of MExp(*b_p_e*)[y $\rightsquigarrow$ y$_i$] are indexed by i. Because none of **stow**(i), **stow**(j), **stow**(k), or **stow**(l) occur in *prec_top_lev_code*, $\{i, j, k, l\} \cap \text{GI} = \emptyset$. All variables of the statement "**assume** (MExp(*b_p_e*)[y $\rightsquigarrow$ y$_j$]) $\wedge$ (Inv[x $\rightsquigarrow$ x$_j$, #x $\rightsquigarrow$ x$_i$])" are indexed by i or j. All variables of the statement "**confirm** Inv[x $\rightsquigarrow$ x$_k$, #x $\rightsquigarrow$ x$_i$]" are indexed by i or k. All other parts of WE1 are derived from statements within a **loop while** statement, which, in turn, is within a **whenever** statement of $\mathcal{P}$. For these reasons and the induction hypothesis, if ξ_h occurs in WE1, then $h \notin$ GI. WE2 is derived from a **whenever** statement of $\mathcal{P}$ by removing a **loop while** statement and a **stow**(l) statement, and inserting an **assume** statement. All variables of the statement "**assume** ($\neg$(MExp(*b_p_e*)[y $\rightsquigarrow$ y$_l$])) $\wedge$ (Inv[x $\rightsquigarrow$ x$_l$, #x $\rightsquigarrow$ x$_i$])" are indexed by i or l. By this fact and the induction hypothesis, if ξ_h occurs in WE2, then $h \notin$ GI. Because if ξ_h occurs in Br_Cd, then $h \notin$ GI, and because all variables of the expression "(Inv[x $\rightsquigarrow$ x$_i$, #x $\rightsquigarrow$ x$_i$])" are indexed by i, if ξ_h occurs in "**confirm** (Br_Cd) $\Rightarrow$ (Inv[x $\rightsquigarrow$ x$_i$, #x $\rightsquigarrow$ x$_i$])", then $h \notin$ GI. We have checked all seven statements. Therefore, all the lemma's conditions are satisfied for this case.

Case 2. We divide this case into two subcases: (a) WE1 is the **whenever** statement of the lemma and (b) WE2 is the **whenever** statement of the lemma.

Case 2(a). The set, GW1, of indices that appear in **stow**s in WE1 is a subset of the set for the **whenever** statement of $\mathcal{P}$ that contains the **loop while** statement from which WE1 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE1. By Lemma 4.22, if ξ_h occurs in Br_Cd, then $h \notin$ GW1. By Lemma 4.23, if ξ_h occurs in *cd_suffix*, then $h \notin$ GW1. $\{i, l\} \cap \text{GW1} = \emptyset$. Hence, if ξ_h occurs in WE2, then

$h \notin$ GW1. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(WE2)(env$_1$), $\mathcal{I}$(WE2)(env$_2$)), where GS $\subseteq$ GW1. Therefore, WE2 satisfies the lemma's conditions for WE1. Because they have no indexed variables, **alter all** and **stow**(l) both satisfy the lemma's conditions for WE1. So, we have finished this case.
Case 2(b). The set, GW2, of indices that appear in **stow**s in WE2 (i.e., the **stow**s of *cd_suffix*) is a subset of the set for the **whenever** statement of $\mathcal{P}$ from which WE2 was derived. So, by the induction hypothesis and the definition of subset, each statement of *fol_top_lev_code* satisfies the lemma's conditions for WE2.
Hence, $\mathcal{M}$ of the **loop while** rule satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for **confirm** satisfies the lemma's conditions. $\mathcal{M}$ is derived from $\mathcal{P}$ by replacing one **whenever** statement with a **confirm** statement and a **whenever** statement. Let ST1 stand for the **confirm** statement: **confirm** (Br_Cd) $\Rightarrow$ (H[x $\rightsquigarrow$ x$_i$]). Let ST2 stand for the **whenever** statement.
Case 1. We must show that both ST1 and ST2 satisfy the conditions the lemma places on S1. The statement "**confirm** $H\lfloor$x$\rfloor$" of $\mathcal{P}$ is within a **whenever** statement; therefore, by the induction hypothesis, if ξ_h occurs in H, then $h \notin$ GI. Every free current variable name in H is replaced—in H[x $\rightsquigarrow$ x$_i$]—by a variable subscripted with i. By the syntactic restriction that the indices appearing in **stow**s are everywhere increasing, $i \notin$ GI. Because it appears as the condition of a **whenever** statement, and not in an **assume** or **confirm** statement, by the induction hypothesis, if ξ_h occurs in Br_Cd, then $h \notin$ GI. Therefore, if ξ_h occurs in ST1, then $h \notin$ GI. Suppose env$_1$ and env$_2$ satisfy the lemma's assumptions. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(ST1)(env$_1$), $\mathcal{I}$(ST1)(env$_2$)). Hence, the lemma's conditions are satisfied for ST1. Because ST2 is derived from a **whenever** statement of $\mathcal{P}$ by removing one **confirm** statement, by the induction hypothesis, if ξ_h occurs in ST2, then $h \notin$ GI. By Lemma 4.34, Equal_except_at(GS, $\mathcal{I}$(ST2)(env$_1$), $\mathcal{I}$(ST2)(env$_2$)). Hence, the lemma's conditions are satisfied for ST2. Therefore, all the lemma's conditions are satisfied for this case.
Case 2. The set of indices that appear in **stow**s in ST2 is the same as the set for the **whenever** statement of $\mathcal{P}$ from which it was derived. So, by the induction hypothesis, all the lemma's conditions are satisfied for this case.
Hence, $\mathcal{M}$ of the rule for **confirm** satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for empty guarded blocks satisfies the lemma's conditions.
Case 1. Every statement of $\mathcal{M}$ is a statement of $\mathcal{P}$. Therefore, by the induction hypothesis, all the lemma's conditions are satisfied for this case.
Case 2. Because there are no **whenever** statements explicitly mentioned in $\mathcal{M}$, all

the lemma's conditions are satisfied (vacuously) for this case.
Hence, $\mathcal{M}$ of the rule for empty guarded blocks satisfies the lemma's conditions.

Due to its additional syntactic restriction, neither $\mathcal{P}$ nor $\mathcal{M}$ of the rule for **alter all** contains any **whenever** statements. Hence, $\mathcal{M}$ of the rule for **alter all** (vacuously) satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for consecutive **assume** statements satisfies the lemma's conditions.
Case 1. Given the lemma's hypotheses, we have Equal_except_at(GS, $\mathcal{I}$(**assume** H_1)(env$_1$), $\mathcal{I}$(**assume** H_1)(env$_2$)) and Equal_except_at(GS, $\mathcal{I}$(**assume** H_2)(env$_1$), $\mathcal{I}$(**assume** H_2)(env$_2$)). If AE(env$_1$) $\neq$ NL, then Equal_except_at(GS, $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$), $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_2$)). So, we suppose AE(env$_1$) = NL. If AE($\mathcal{I}$(**assume** H_1)(env$_1$)) $\neq$ NL, then $\mathcal{I}(H_1)$(env$_1$) = $\mathcal{I}(H_1)$(env$_2$) = **false**. In this case, $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$) equals env$_1$ everywhere except at the assert status where AE($\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$)) = VT. The same is true for env$_2$. Hence, Equal_except_at(GS, $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$), $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_2$)). So, we suppose AE($\mathcal{I}$(**assume** H_1)(env$_1$)) = NL. Then $\mathcal{I}(H_1)$(env$_1$) = $\mathcal{I}(H_1)$(env$_2$) = **true**. Hence, $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$) = $\mathcal{I}$(**assume** H_2)(env$_1$) and $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_2$) = $\mathcal{I}$(**assume** H_2)(env$_2$). Therefore, Equal_except_at(GS, $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_1$), $\mathcal{I}$(**assume** $(H_1) \wedge (H_2)$)(env$_2$)). The statement "**assume** $(H_1) \wedge (H_2)$" is an **assume** statement. By the induction hypothesis, it is not within a **whenever** statement. Therefore, all the lemma's conditions are satisfied for this case.
Case 2. Because there are no **whenever** statements explicitly mentioned in $\mathcal{M}$, all the lemma's conditions are satisfied (vacuously) for this case.
Hence, $\mathcal{M}$ of the rule for consecutive **assume** statements satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the **assume-confirm** rule satisfies the lemma's conditions.
Case 1. Given the lemma's hypotheses, we have Equal_except_at(GS, $\mathcal{I}$(**assume** H_1)(env$_1$), $\mathcal{I}$(**assume** H_1)(env$_2$)) and Equal_except_at(GS, $\mathcal{I}$(**confirm** H_2)(env$_1$), $\mathcal{I}$(**confirm** H_2)(env$_2$)). If AE(env$_1$) $\neq$ NL, then Equal_except_at(GS, $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_1$), $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_2$)). So, we suppose AE(env$_1$) = NL. If AE($\mathcal{I}$(**assume** H_1)(env$_1$)) $\neq$ NL, then $\mathcal{I}(H_1)$(env$_1$) = $\mathcal{I}(H_1)$(env$_2$) = **false**. In this case, $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_1$) = env$_1$ and $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_2$) = env$_2$. Hence, Equal_except_at(GS, $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_1$), $\mathcal{I}$(**confirm** $(H_1) \Rightarrow (H_2)$)(env$_2$)). So, we suppose AE($\mathcal{I}$(**assume** H_1)(env$_1$)) = NL. Then

$\mathcal{I}(H_1)(\text{env}_1) = \mathcal{I}(H_1)(\text{env}_2) = \textbf{true}$. Hence, $\mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\text{env}_1) = \mathcal{I}(\textbf{confirm } H_2)(\text{env}_1)$ and $\mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\text{env}_2) = \mathcal{I}(\textbf{confirm } H_2)(\text{env}_2)$. Therefore, Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\text{env}_1), \mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\text{env}_2)$). The statement "**confirm** $(H_1) \Rightarrow (H_2)$" is a **confirm** statement. By the induction hypothesis, it is not within a **whenever** statement. Therefore, all the lemma's conditions are satisfied for this case.
Case 2. Because there are no **whenever** statements explicitly mentioned in $\mathcal{M}$, all the lemma's conditions are satisfied (vacuously) for this case.
Hence, $\mathcal{M}$ of the **assume-confirm** rule satisfies the lemma's conditions.

By the induction hypothesis, $\mathcal{P}$ of the rule for consecutive **confirm** statements satisfies the lemma's conditions.
Case 1. Given the lemma's hypotheses, we have Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } H_1)(\text{env}_1), \mathcal{I}(\textbf{confirm } H_1)(\text{env}_2)$) and Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } H_2)(\text{env}_1), \mathcal{I}(\textbf{confirm } H_2)(\text{env}_2)$). If $\text{AE}(\text{env}_1) \neq \text{NL}$, then Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1), \mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_2)$). So, we suppose $\text{AE}(\text{env}_1) = \text{NL}$. If $\text{AE}(\mathcal{I}(\textbf{confirm } H_1)(\text{env}_1)) \neq \text{NL}$, then $\mathcal{I}(H_1)(\text{env}_1) = \mathcal{I}(H_1)(\text{env}_2) = \textbf{false}$. In this case, $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1)$ equals env_1 everywhere except at the assert status where $\text{AE}(\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1)) = \text{CF}$. The same is true for env_2. Hence, Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1), \mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_2)$). So, we suppose $\text{AE}(\mathcal{I}(\textbf{confirm } H_1)(\text{env}_1)) = \text{NL}$. Then $\mathcal{I}(H_1)(\text{env}_1) = \mathcal{I}(H_1)(\text{env}_2) = \textbf{true}$. Hence, $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1) = \mathcal{I}(\textbf{confirm } H_2)(\text{env}_1)$ and $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_2) = \mathcal{I}(\textbf{confirm } H_2)(\text{env}_2)$. Therefore, Equal_except_at(GS, $\mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_1), \mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\text{env}_2)$). The statement "**confirm** $(H_1) \wedge (H_2)$" is a **confirm** statement. By the induction hypothesis, it is not within a **whenever** statement. Therefore, all the lemma's conditions are satisfied for this case.
Case 2. Because there are no **whenever** statements explicitly mentioned in $\mathcal{M}$, all the lemma's conditions are satisfied (vacuously) for this case.
Hence, $\mathcal{M}$ of the rule for consecutive **confirm** statements satisfies the lemma's conditions.

The induction step is now complete. Hence the proof of this lemma (4.35) is finished. □

Now we are ready to state and prove the negative-branch-condition independence lemma.

Lemma 4.36 (Negative-Branch-Condition Independence) *Let $\mathcal{R}$ be an intermediate result obtained by applying the proof rules in the math direction to a*

programmer-written program. Furthermore, let $\mathcal{R}$ *have the following form:*

$$\mathcal{R} \stackrel{\text{def}}{=} C\backslash \begin{array}{l} \textit{prec\_top\_lev\_code} \\ \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ \quad \textit{guarded\_code} \\ \textbf{end whenever} \\ \textit{fol\_top\_lev\_code} \end{array} \tag{4.35}$$

Let GI be the set of indices h *that appear in* **stow**(h) *statements anywhere in guarded_code. Let* $GS \subseteq GI$. *Let* env_1 *and* env_2 *be two environments. If* $\textit{Equal\_except\_at}(GS, env_1, env_2)$ *and* $\neg\mathcal{I}(\text{Br\_Cd})(env_1)$, *then* $\textit{Equal\_except\_at}(GS, \mathcal{I}(\textit{fol\_top\_lev\_code})(env_1), \mathcal{I}(\textit{fol\_top\_lev\_code})(env_2))$.

Proof. Given the hypotheses of this lemma, we must show $\text{Equal\_except\_at}(\text{GS}, \mathcal{I}(\textit{fol\_top\_lev\_code})(\text{env}_1), \mathcal{I}(\textit{fol\_top\_lev\_code})(\text{env}_2))$. If it were always true that *fol_top_lev_code* contained no variables indexed by a member of the set GI, an appeal to Lemmas 4.34 and 4.33 would complete our proof. But *fol_top_lev_code* can contain variables indexed by members of the set GI. However, we will show that the different states of those indices can have no effect on the interpretation of *fol_top_lev_code*. The reason is that any occurrence in *fol_top_lev_code* of a variable indexed by a member of GI must be in an assertion in such a way that it is "guarded" by Br_Cd. For example, occurrences in *fol_top_lev_code* of a variable indexed by a member of GI are frequently in the *consequent* of an **assume** statement at the top level of *fol_top_lev_code* that has the form "**assume** (Br_Cd) $\Rightarrow$ (*consequent*)". This lemma (4.36) follows from Lemmas 4.35 and 4.33. □

4.3.6 Internal-Index Independence

The internal-index independence lemma is key for our ability to establish invalidity preservation for the rules that handle compound statements such as **loop while** and selection. In contrast with the negative-branch-condition independence lemma, the internal-index independence lemma does not require the hypothesis that Br_Cd evaluates as false. The other important difference is that, here, we permit differences only on the set of indices h that appear in **stow**(h) statements within the compound statement.

Lemma 4.37 (Internal-Index Independence) *Let* $\mathcal{R}$ *be an intermediate result obtained by applying the proof rules to a programmer-written program toward the*

goal of obtaining an assertion. Furthermore, let $\mathcal{R}$ *have the following form:*

$$\begin{array}{rcll} \mathcal{R} & \stackrel{\text{def}}{=} & C\backslash & \textit{prec\_top\_lev\_code} \\ & & & \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ & & & \quad \textit{compound\_stmt} \\ & & & \quad \textbf{stow}(n) \\ & & & \quad \textit{cd\_suffix} \\ & & & \textbf{end whenever} \\ & & & \textit{fol\_top\_lev\_code} \end{array} \tag{4.36}$$

Let CI be the set of indices h *that appear in* **stow**(h) *statements within compound_stmt. Let* env_1 *and* env_2 *be two environments. If* $\mathit{Equal\_except\_at}(CI, env_1, env_2)$*, then* $\mathit{Equal\_except\_at}(CI, \mathcal{I}(\mathit{cd\_suffix})(env_1), \mathcal{I}(\mathit{cd\_suffix})(env_2))$*. Furthermore,* $\mathit{Equal\_except\_at}(CI, \mathcal{I}(\mathit{fol\_top\_lev\_code})(env_1), \mathcal{I}(\mathit{fol\_top\_lev\_code})(env_2))$*.*

Proof. Let S1 be any statement of *cd_suffix* or *fol_top_lev_code*. By Lemma 4.23, if indexed variable ξ_h occurs in S1, $h \notin$ CI. By Lemma 4.34, $\text{Equal\_except\_at}(\text{CI}, \mathcal{I}(\text{S1})(\text{env}_1), \mathcal{I}(\text{S1})(\text{env}_2))$. By Lemma 4.33, $\text{Equal\_except\_at}(\text{CI}, \mathcal{I}(\mathit{cd\_suffix})(\text{env}_1), \mathcal{I}(\mathit{cd\_suffix})(\text{env}_2))$ and $\text{Equal\_except\_at}(\text{CI}, \mathcal{I}(\mathit{fol\_top\_lev\_code})(\text{env}_1), \mathcal{I}(\mathit{fol\_top\_lev\_code})(\text{env}_2))$. □

4.4 Soundness Lemmas

In this section we establish Lemma 4.3 by proving a series of lemmas, one for each proof rule of Chapter III. Each lemma states that the rule preserves invalidity in the math direction. Skepticism may be greatest about the **if-then-else** statement; therefore, we present its lemma and proof first. The proof is more complicated than the others. Readers wishing a gentler introduction are invited to skip ahead to some shorter proofs before returning here.

Lemma 4.38 *The rule for selection in the presence of an* **else** *clause preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\text{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\text{env}_I^{\mathcal{P}})$; i.e., $\text{AE}(\text{env}_I^{\mathcal{P}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$. We will show the existence of an environment, $\text{env}_I^{\mathcal{M}}$, such that $\text{AE}(\text{env}_I^{\mathcal{M}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$, where $\text{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\text{env}_I^{\mathcal{M}})$. The proof is organized by cases. First we make the following two definitions.

$$\text{env}_i^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\text{env}_I^{\mathcal{P}}) \tag{4.37}$$

$$\text{env}_i^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\text{env}_I^{\mathcal{M}}) \tag{4.38}$$

We will be defining other environments as well. Figure 77 shows the positions of the environments defined for $\mathcal{P}$, and figure 78 shows the positions of the environments defined for $\mathcal{M}$.
Case $AE(env_i^{\mathcal{P}}) = CF$. In this case, we take $\text{env}_I^{\mathcal{M}} = \text{env}_I^{\mathcal{P}}$. Then $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$.
Case $AE(env_i^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{NL}$. We may, for this case, write $\text{env}_i^{\mathcal{P}} = [\text{NL}, \text{cs}_i, \text{ns}, \text{se}, \text{os}, \text{d}]$, where $\text{ns}(i) = \text{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\text{env}_I^{\mathcal{P}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \text{se}, \text{os}, \text{d}]$. As shown in Figure 77, we let $\text{env}_{\text{ew}}^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{whenever}\ \text{Br\_Cd}\ \textbf{do}\ \ldots\ \textbf{end whenever})(\text{env}_i^{\mathcal{P}})$.
Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. (We are still inside case $\text{AE}(\text{env}_i^{\mathcal{P}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{NL}$.) In this case, $\text{env}_{\text{ew}}^{\mathcal{P}} = \text{env}_i^{\mathcal{P}}$, and we take $\text{env}_I^{\mathcal{M}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \langle \text{cs}_i, \text{cs}_i, \text{cs}_i \rangle \circ \text{se}, \text{os}, \text{d}]$. Then $\text{env}_i^{\mathcal{M}} = [\text{NL}, \text{cs}_i, \text{ns}, \langle \text{cs}_i, \text{cs}_i, \text{cs}_i \rangle \circ \text{se}, \text{os}, \text{d}]$. As shown in Figure 78, we let $\text{env}_{\text{ew}}^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{alter all stow}(j)\ \ldots\ \textbf{whenever}\ \text{Br\_Cd}\ \textbf{do}\ \textit{cd\_suffix}\ \textbf{end whenever})(\text{env}_i^{\mathcal{M}})$. By Lemma 4.22, the value of $\mathcal{I}(\text{Br\_Cd})(\text{env})$ depends only on $\text{ISE}(\text{env})$ for any environment env. So, because $\text{ISE}(\text{env}_i^{\mathcal{M}}) = \text{ISE}(\text{env}_i^{\mathcal{P}})$, we have $\neg\mathcal{I}(\text{Br\_Cd})(\text{env}_i^{\mathcal{M}})$. Therefore, we have another equality for $\text{env}_{\text{ew}}^{\mathcal{M}}$, namely,

$$\text{env}_{\text{ew}}^{\mathcal{M}} = \mathcal{I}\left(\begin{array}{l}\textbf{alter all stow}(j)\\ \textbf{alter all stow}(l)\\ \textbf{alter all stow}(n)\end{array}\right)(\text{env}_i^{\mathcal{M}}). \tag{4.39}$$

Therefore, $\text{env}_{\text{ew}}^{\mathcal{M}} = [\text{NL}, \text{cs}_i, \text{ns}', \text{se}, \text{os}, \text{d}]$ where

$$\text{ns}'(h) = \begin{cases} \text{ns}(h) & \text{if } h \notin \{j, l, n\} \\ \text{cs}_i & \text{if } h \in \{j, l, n\} \end{cases}. \tag{4.40}$$

So, $\text{Equal\_except\_at}(\{j, l, n\}, \text{env}_{\text{ew}}^{\mathcal{P}}, \text{env}_{\text{ew}}^{\mathcal{M}})$. The set $\{j, l, n\}$ is a subset of the set of indices h that appear in $\textbf{stow}(h)$ statements anywhere within the **whenever** statement explicitly shown in the schema $\mathcal{P}$. In this case, by Lemma 4.36, $\text{Equal\_except\_at}(\{j, l, n\}, \mathcal{I}(fol\_top\_lev\_code)(\text{env}_{\text{ew}}^{\mathcal{P}}), \mathcal{I}(fol\_top\_lev\_code)(\text{env}_{\text{ew}}^{\mathcal{M}}))$.
Hence, $\text{Equal\_except\_at}(\{j, l, n\}, \text{env}_F^{\mathcal{P}}, \text{env}_F^{\mathcal{M}})$. Therefore, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$.
Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. (We are still inside case $\text{AE}(\text{env}_i^{\mathcal{P}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{NL}$.) In this case, $\text{env}_{\text{ew}}^{\mathcal{P}} = \mathcal{I}(\textbf{if}\ b\_p\_e\ \textbf{then}\ \ldots\ \textbf{end if stow}(n)\ cd\_suffix)(\text{env}_i^{\mathcal{P}})$. Let $\text{env}_n^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{if}\ b\_p\_e\ \textbf{then}\ \ldots\ \textbf{end if stow}(n))(\text{env}_i^{\mathcal{P}})$. In this case, we take $\text{env}_I^{\mathcal{M}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \langle \text{cs}_i, \text{cs}_i, \text{CSE}(\text{env}_n^{\mathcal{P}}) \rangle \circ \text{se}, \text{os}, \text{d}]$. Then $\text{env}_i^{\mathcal{M}} = [\text{NL}, \text{cs}_i, \text{ns}, \langle \text{cs}_i, \text{cs}_i, \text{CSE}(\text{env}_n^{\mathcal{P}}) \rangle \circ \text{se}, \text{os}, \text{d}]$. We define $\text{env}_j^{\mathcal{M}}$, $\text{env}_l^{\mathcal{M}}$,

$\mathcal{P} \stackrel{\text{def}}{=} C\backslash$ $\quad \text{env}_I^{\mathcal{P}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$

$\quad \text{env}_i^{\mathcal{P}}$

whenever Br_Cd **do**
  if *b_p_e* **then**
    stow(j)
    cd_kern_1
    stow(k)
    $ACseq_1$
  else
    stow(l)
    cd_kern_2
    stow(m)
    $ACseq_2$
  end if
  stow(n)

$\quad \text{env}_n^{\mathcal{P}}$

  cd_suffix
end whenever

$\quad \text{env}_{\text{ew}}^{\mathcal{P}}$

fol_top_lev_code

$\quad \text{env}_F^{\mathcal{P}}$

Figure 77: Environments Defined for $\mathcal{P}$ of the Rule for Selection in the Presence of an **else** Clause

$\mathcal{M} \stackrel{\text{def}}{=} C\backslash$

$\text{env}_I^{\mathcal{M}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$

$\text{env}_i^{\mathcal{M}}$

alter all
stow(j)

$\text{env}_j^{\mathcal{M}}$

whenever (Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y_i]) **do**
  assume $\text{x}_j = \text{x}_i$
  cd_kern_1
  stow(k)
  $ACseq_1$
end whenever
alter all
stow(l)

$\text{env}_l^{\mathcal{M}}$

whenever (Br_Cd) $\wedge \neg$(MExp(b_p_e)[y $\rightsquigarrow$ y_i]) **do**
  assume $\text{x}_l = \text{x}_i$
  cd_kern_2
  stow(m)
  $ACseq_2$
end whenever
alter all
stow(n)

$\text{env}_n^{\mathcal{M}}$

assume ((Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y_i])) $\Rightarrow$ $(\text{x}_n = \text{x}_k)$
assume ((Br_Cd) $\wedge \neg$(MExp(b_p_e)[y $\rightsquigarrow$ y_i])) $\Rightarrow$ $(\text{x}_n = \text{x}_m)$
whenever Br_Cd **do**
  cd_suffix
end whenever

$\text{env}_{\text{ew}}^{\mathcal{M}}$

fol_top_lev_code

$\text{env}_F^{\mathcal{M}}$

Figure 78: Environments Defined for $\mathcal{M}$ of the Rule for Selection in the Presence of an **else** Clause

and $\mathrm{env}_n^{\mathcal{M}}$ as follows.

$$\mathrm{env}_j^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{alter all stow}(j))(\mathrm{env}_i^{\mathcal{M}}) \tag{4.41}$$

$$\mathrm{env}_l^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}\left(\begin{array}{l}\textbf{whenever}\\ (\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])\ \textbf{do}\\ \ldots\\ \textbf{end whenever alter all stow}(l)\end{array}\right)(\mathrm{env}_j^{\mathcal{M}}) \tag{4.42}$$

$$\mathrm{env}_n^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}\left(\begin{array}{l}\textbf{whenever}\\ (\mathrm{Br\_Cd}) \wedge \neg(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])\ \textbf{do}\\ \ldots\\ \textbf{end whenever alter all stow}(n)\end{array}\right)(\mathrm{env}_l^{\mathcal{M}}) \tag{4.43}$$

Case $\mathcal{I}(\mathrm{b\_p\_e})(env_i^{\mathcal{P}})$. In this case, $\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_i^{\mathcal{P}})$. In each of the following environments that we defined above, namely, $\mathrm{env}_i^{\mathcal{M}}$, $\mathrm{env}_j^{\mathcal{M}}$, $\mathrm{env}_l^{\mathcal{M}}$, and $\mathrm{env}_n^{\mathcal{M}}$, we have, for $h \in \{i, j, l, n\}$, $\mathrm{ISE}(\mathrm{env}_h^{\mathcal{M}})(i) = \mathrm{ns}(i)$. Therefore, we also have, for $h \in \{i, j, l, n\}$, $\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_h^{\mathcal{M}})$. According to the semantics of **if-then-else**, $\mathrm{env}_n^{\mathcal{P}} = \mathcal{I}(\textbf{stow}(j)\ cd\_kern_1\ \textbf{stow}(k)\ ACseq_1\ \textbf{stow}(n))(\mathrm{env}_i^{\mathcal{P}})$. According to the semantics of **whenever**, $\mathrm{env}_l^{\mathcal{M}} = \mathcal{I}(\textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i\ cd\_kern_1\ \textbf{stow}(k)\ ACseq_1\ \textbf{alter all stow}(l))(\mathrm{env}_j^{\mathcal{M}})$. Because $\mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(j) = \mathrm{cs}_i = \mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}})(i) = \mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(i)$, $\mathrm{env}_j^{\mathcal{M}} = \mathcal{I}(\textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i)(\mathrm{env}_j^{\mathcal{M}})$, so $\mathrm{env}_l^{\mathcal{M}} = \mathcal{I}(cd\_kern_1\ \textbf{stow}(k)\ ACseq_1\ \textbf{alter all stow}(l))(\mathrm{env}_j^{\mathcal{M}})$. Because $\mathrm{CSE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{cs}_i = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}})$, by Lemma 4.32, $\mathrm{AE}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{AE}(\mathrm{env}_n^{\mathcal{P}})$, $\mathrm{CSE}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}})$, $\mathrm{OSE}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{OSE}(\mathrm{env}_n^{\mathcal{P}})$, and $\mathrm{DME}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{DME}(\mathrm{env}_n^{\mathcal{P}})$.

Case $AE(env_n^{\mathcal{P}}) = CF$. Then, by Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{AE}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{AE}(\mathrm{env}_n^{\mathcal{P}}) = \mathrm{CF}$.

Case $AE(env_n^{\mathcal{P}}) \neq CF$. By Lemma 4.19, $\mathrm{AE}(\mathrm{env}_n^{\mathcal{P}}) = \mathrm{NL}$. Hence, $\mathrm{AE}(\mathrm{env}_l^{\mathcal{M}}) = \mathrm{NL}$. In this case (i.e., $\mathcal{I}(b\_p\_e)(\mathrm{env}_i^{\mathcal{P}})$), according to the semantics of **whenever**, $\mathrm{env}_n^{\mathcal{M}} = \mathcal{I}(\textbf{alter all stow}(n))(\mathrm{env}_l^{\mathcal{M}})$. Therefore, $\mathrm{AE}(\mathrm{env}_n^{\mathcal{M}}) = \mathrm{NL}$ and, due to the setup, $\mathrm{CSE}(\mathrm{env}_n^{\mathcal{M}}) = \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}})$. Furthermore,

$$\begin{aligned}\mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(k) &= \mathrm{ISE}(\mathrm{env}_l^{\mathcal{M}})(k) && (4.44)\\ &= \mathrm{ISE}(\mathrm{env}_n^{\mathcal{P}})(k) && (4.45)\\ &= \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}}) && (4.46)\end{aligned}$$

and

$$\begin{aligned}\mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(n) &= \mathrm{CSE}(\mathrm{env}_n^{\mathcal{M}}) && (4.47)\\ &= \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}}) && (4.48)\\ &= \mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(k). && (4.49)\end{aligned}$$

We now have the following equalities for $\text{env}_F^{\mathcal{P}}$ and $\text{env}_F^{\mathcal{M}}$.

$$\text{env}_F^{\mathcal{P}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_n^{\mathcal{P}}) \tag{4.50}$$

$$\text{env}_F^{\mathcal{M}} = \mathcal{I}\left(\begin{array}{l}\textbf{assume } ((\text{Br\_Cd}) \wedge \\ \quad (\text{MExp}(\textit{b\_p\_e})[\text{y} \rightsquigarrow \text{y}_i])) \Rightarrow (\text{x}_n = \text{x}_k) \\ \textbf{assume } ((\text{Br\_Cd}) \wedge \\ \quad \neg(\text{MExp}(\textit{b\_p\_e})[\text{y} \rightsquigarrow \text{y}_i])) \Rightarrow (\text{x}_n = \text{x}_m) \\ \textbf{whenever } \text{Br\_Cd } \textbf{do} \\ \quad \textit{cd\_suffix} \\ \textbf{end whenever} \\ \textit{fol\_top\_lev\_code}\end{array}\right)(\text{env}_n^{\mathcal{M}}) \tag{4.51}$$

Because $\text{ISE}(\text{env}_n^{\mathcal{M}})(k) = \text{ISE}(\text{env}_n^{\mathcal{M}})(n)$, $\mathcal{I}(\text{MExp}(\textit{b\_p\_e})[\text{y} \rightsquigarrow \text{y}_i])(\text{env}_n^{\mathcal{M}})$, and $\mathcal{I}(\text{Br\_Cd})(\text{env}_n^{\mathcal{M}})$, we have

$$\text{env}_F^{\mathcal{M}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_n^{\mathcal{M}}). \tag{4.52}$$

The only place $\text{env}_n^{\mathcal{M}}$ and $\text{env}_n^{\mathcal{P}}$ differ is at $\text{ISE}(\text{env}_n^{\mathcal{M}})(l)$. That is to say, $\text{Equal\_except\_at}(\{l\}, \text{env}_n^{\mathcal{P}}, \text{env}_n^{\mathcal{M}})$. By two applications of Lemma 4.37, $\text{Equal\_except\_at}(\{l\}, \text{env}_F^{\mathcal{P}}, \text{env}_F^{\mathcal{M}})$. Therefore, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$.

Case $\neg\mathcal{I}(\text{b\_p\_e})(\textit{env}_i^{\mathcal{P}})$. In this case, $\neg\mathcal{I}(\text{MExp}(\textit{b\_p\_e})[\text{y} \rightsquigarrow \text{y}_i])(\text{env}_i^{\mathcal{P}})$. We also have, for $h \in \{i, j, l, n\}$, $\neg\mathcal{I}(\text{MExp}(\textit{b\_p\_e})[\text{y} \rightsquigarrow \text{y}_i])(\text{env}_h^{\mathcal{M}})$. According to the semantics of **if-then-else**, $\text{env}_n^{\mathcal{P}} = \mathcal{I}(\textbf{stow}(l)\ \textit{cd\_kern}_2\ \textbf{stow}(m)\ \textit{ACseq}_2\ \textbf{stow}(n))(\text{env}_i^{\mathcal{P}})$. According to the semantics of **whenever**, $\text{env}_l^{\mathcal{M}} = \mathcal{I}(\textbf{alter all stow}(l))(\text{env}_j^{\mathcal{M}})$. So $\text{AE}(\text{env}_l^{\mathcal{M}}) = \text{AE}(\text{env}_j^{\mathcal{M}}) = \text{AE}(\text{env}_i^{\mathcal{M}}) = \text{NL}$, and $\text{CSE}(\text{env}_l^{\mathcal{M}}) = \text{cs}_i = \text{CSE}(\text{env}_i^{\mathcal{P}})$. According to the semantics of **whenever**, $\text{env}_n^{\mathcal{M}} = \mathcal{I}(\textbf{assume } \text{x}_l = \text{x}_i\ \textit{cd\_kern}_2\ \textbf{stow}(m)\ \textit{ACseq}_2\ \textbf{alter all stow}(n))(\text{env}_l^{\mathcal{M}})$. Because $\text{ISE}(\text{env}_l^{\mathcal{M}})(l) = \text{cs}_i = \text{ISE}(\text{env}_i^{\mathcal{M}})(i) = \text{ISE}(\text{env}_l^{\mathcal{M}})(i)$, $\text{env}_l^{\mathcal{M}} = \mathcal{I}(\textbf{assume } \text{x}_l = \text{x}_i)(\text{env}_l^{\mathcal{M}})$, so $\text{env}_n^{\mathcal{M}} = \mathcal{I}(\textit{cd\_kern}_2\ \textbf{stow}(m)\ \textit{ACseq}_2\ \textbf{alter all stow}(n))(\text{env}_l^{\mathcal{M}})$. Because $\text{CSE}(\text{env}_l^{\mathcal{M}}) = \text{cs}_i = \text{CSE}(\text{env}_i^{\mathcal{P}})$, by Lemma 4.32, $\text{AE}(\text{env}_n^{\mathcal{M}}) = \text{AE}(\text{env}_n^{\mathcal{P}})$, $\text{OSE}(\text{env}_n^{\mathcal{M}}) = \text{OSE}(\text{env}_n^{\mathcal{P}})$, and $\text{DME}(\text{env}_n^{\mathcal{M}}) = \text{DME}(\text{env}_n^{\mathcal{P}})$.

Case $\textit{AE}(\textit{env}_n^{\mathcal{P}}) = \textit{CF}$. Then, by Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{AE}(\text{env}_n^{\mathcal{M}}) = \text{AE}(\text{env}_n^{\mathcal{P}}) = \text{CF}$.

Case $\textit{AE}(\textit{env}_n^{\mathcal{P}}) \neq \textit{CF}$. By Lemma 4.19, $\text{AE}(\text{env}_n^{\mathcal{P}}) = \text{NL}$. Hence, $\text{AE}(\text{env}_l^{\mathcal{M}}) = \text{NL}$. Furthermore, due to the setup, $\text{CSE}(\text{env}_n^{\mathcal{M}}) = \text{CSE}(\text{env}_n^{\mathcal{P}})$. We have the following sequence of equations.

$$\begin{aligned}\text{ISE}(\text{env}_n^{\mathcal{M}})(m) &= \text{CSE}(\text{env}_n^{\mathcal{P}}) && (4.53)\\ &= \text{CSE}(\text{env}_n^{\mathcal{M}}) && (4.54)\\ &= \text{ISE}(\text{env}_n^{\mathcal{M}})(n) && (4.55)\end{aligned}$$

Equations 4.50 and 4.51 still hold for $\mathrm{env}_F^{\mathcal{P}}$ and $\mathrm{env}_F^{\mathcal{M}}$. Because $\mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(m) = \mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(n)$, $\mathcal{I}(\neg(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i]))(\mathrm{env}_n^{\mathcal{M}})$, and $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_n^{\mathcal{M}})$, we have

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_n^{\mathcal{M}}). \tag{4.56}$$

The only place $\mathrm{env}_n^{\mathcal{M}}$ and $\mathrm{env}_n^{\mathcal{P}}$ differ is at $\mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(j)$. That is to say, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_n^{\mathcal{P}}, \mathrm{env}_n^{\mathcal{M}})$. By two applications of Lemma 4.37, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_F^{\mathcal{P}}, \mathrm{env}_F^{\mathcal{M}})$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.39 *The rule for selection in the absence of an* **else** *clause preserves invalidity in the math direction.*

Proof. The proof is similar to the proof that the rule for selection in the presence of an **else** clause preserves invalidity in the math direction. The important difference is in the case in which $\neg\mathcal{I}(b\_p\_e)(\mathrm{env}_i^{\mathcal{P}})$ (within the cases $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) \neq \mathrm{CF}$ and $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$). We supply here only the argument for this case. The environments defined in the complete proof are similar to those defined for the rule for selection in the presence of an **else** clause. Figure 79 shows the positions of the environments defined for $\mathcal{P}$, and figure 80 shows the positions of the environments defined for $\mathcal{M}$. Because $\mathcal{M}$ of the rule for selection in the absence of an **else** clause introduces two—and not three—**alter all** statements, we insert two—and not three—states in the setup. We take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_i, \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}})\rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_i, \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}})\rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$.
Case $\neg\mathcal{I}(\mathrm{b\_p\_e})(env_i^{\mathcal{P}})$. In this case, $\neg\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_i^{\mathcal{P}})$. We also have, for $h \in \{i, j, n\}$, $\neg\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_h^{\mathcal{M}})$. According to the semantics of **if-then**, $\mathrm{env}_n^{\mathcal{P}} = \mathcal{I}(\mathbf{stow}(n))(\mathrm{env}_i^{\mathcal{P}})$. Therefore, $\mathrm{Equal\_except\_at}(\{n\}, \mathrm{env}_n^{\mathcal{P}}, \mathrm{env}_i^{\mathcal{P}})$. In particular, $\mathrm{AE}(\mathrm{env}_n^{\mathcal{P}}) = \mathrm{AE}(env_i^{\mathcal{P}}) = \mathrm{NL}$. According to the semantics of **whenever**, $\mathrm{env}_n^{\mathcal{M}} = \mathcal{I}(\mathbf{alter\ all\ stow}(n))(\mathrm{env}_j^{\mathcal{M}})$. Therefore, $\mathrm{env}_n^{\mathcal{M}} = [\mathrm{NL}, \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}}), \mathrm{ns}_n^{\mathcal{M}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_n^{\mathcal{M}}(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \notin \{j, n\} \\ \mathrm{cs}_i & \text{if } h = j \\ \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}}) & \text{if } h = n \end{cases}. \tag{4.57}$$

Hence, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_n^{\mathcal{P}}, \mathrm{env}_n^{\mathcal{M}})$. We have the following sequence of equations.

$$\begin{aligned} \mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(i) &= \mathrm{ISE}(\mathrm{env}_n^{\mathcal{P}})(i) && (4.58)\\ &= \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}}) && (4.59)\\ &= \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}}) && (4.60)\\ &= \mathrm{CSE}(\mathrm{env}_n^{\mathcal{M}}) && (4.61)\\ &= \mathrm{ISE}(\mathrm{env}_n^{\mathcal{M}})(n) && (4.62) \end{aligned}$$

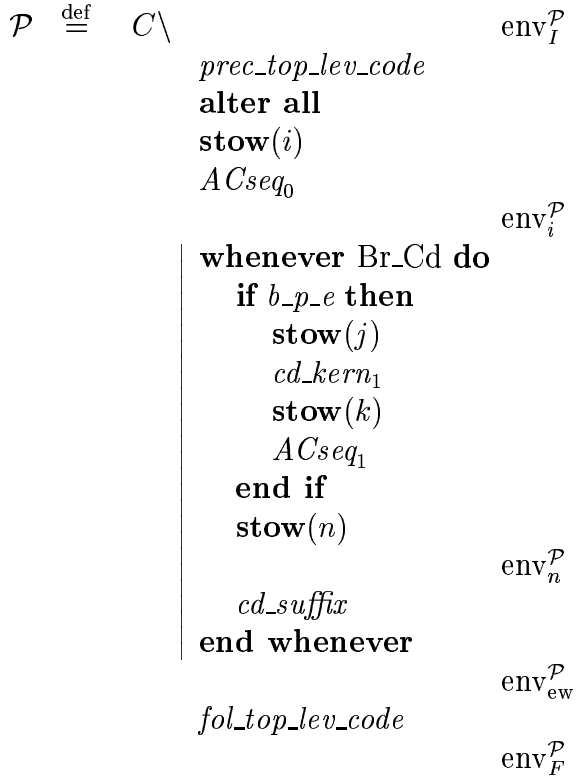

Figure 79: Environments Defined for $\mathcal{P}$ of the Rule for Selection in the Absence of an **else** Clause

$\mathcal{M} \stackrel{\text{def}}{=} C\backslash$ $\quad \text{env}_I^{\mathcal{M}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$
$\quad \text{env}_i^{\mathcal{M}}$

| **alter all**
| **stow**(j)
| $\quad \text{env}_j^{\mathcal{M}}$
| **whenever** (Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y_i]) **do**
| **assume** $\text{x}_j = \text{x}_i$
| *cd_kern*$_1$
| **stow**(k)
| $ACseq_1$
| **end whenever**
| **alter all**
| **stow**(n)
| $\quad \text{env}_n^{\mathcal{M}}$
| **assume** ((Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y_i])) $\Rightarrow$ ($\text{x}_n = \text{x}_k$)
| **assume** ((Br_Cd) $\wedge \neg$(MExp(b_p_e)[y $\rightsquigarrow$ y_i])) $\Rightarrow$ ($\text{x}_n = \text{x}_i$)
| **whenever** Br_Cd **do**
| *cd_suffix*
| **end whenever**
| $\quad \text{env}_{\text{ew}}^{\mathcal{M}}$

fol_top_lev_code
$\quad \text{env}_F^{\mathcal{M}}$

Figure 80: Environments Defined for $\mathcal{M}$ of the Rule for Selection in the Absence of an **else** Clause

Recall that $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_n^{\mathcal{P}})$. Because $\text{ISE}(\text{env}_n^{\mathcal{M}})(i) = \text{ISE}(\text{env}_n^{\mathcal{M}})(n)$, $\mathcal{I}(\neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i]))(\text{env}_n^{\mathcal{M}})$, and $\mathcal{I}(\text{Br\_Cd})(\text{env}_n^{\mathcal{M}})$, we have

$$\text{env}_F^{\mathcal{M}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_n^{\mathcal{M}}). \tag{4.63}$$

Recalling that $\text{Equal\_except\_at}(\{j\}, \text{env}_n^{\mathcal{P}}, \text{env}_n^{\mathcal{M}})$, by two applications of Lemma 4.37, $\text{Equal\_except\_at}(\{j\}, \text{env}_F^{\mathcal{P}}, \text{env}_F^{\mathcal{M}})$. Therefore, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$. □

Lemma 4.40 *The* **loop while** *rule preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\text{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\text{env}_I^{\mathcal{P}})$; i.e., $\text{AE}(\text{env}_I^{\mathcal{P}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$. We will show the existence of an environment, $\text{env}_I^{\mathcal{M}}$, such that $\text{AE}(\text{env}_I^{\mathcal{M}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$, where $\text{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\text{env}_I^{\mathcal{M}})$. The proof is organized by cases. First we make the following two definitions.

$$\text{env}_i^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\,ACseq_0)(\text{env}_I^{\mathcal{P}}) \tag{4.64}$$

$$\text{env}_i^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\,ACseq_0)(\text{env}_I^{\mathcal{M}}) \tag{4.65}$$

We will be defining other environments as well. Figure 81 shows the positions of the environments defined for $\mathcal{P}$, and figure 82 shows the positions of the environments defined for $\mathcal{M}$.

Case $AE(env_i^{\mathcal{P}}) = CF$. In this case, we take $\text{env}_I^{\mathcal{M}} = \text{env}_I^{\mathcal{P}}$. Then $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$.

Case $AE(env_i^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{NL}$. We may, for this case, write $\text{env}_i^{\mathcal{P}} = [\text{NL}, \text{cs}_i, \text{ns}, \text{se}, \text{os}, \text{d}]$, where $\text{ns}(i) = \text{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\text{env}_I^{\mathcal{P}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \text{se}, \text{os}, \text{d}]$. As shown in Figure 77, we let $\text{env}_{\text{ew}}^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{whenever}\ \text{Br\_Cd}\ \textbf{do}\ \ldots\ \textbf{end whenever})(\text{env}_i^{\mathcal{P}})$.

Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. This case is similar to the corresponding case in the proof of Lemma 4.38, and we do not repeat it here.

Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. (We are still inside case $\text{AE}(\text{env}_i^{\mathcal{P}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{NL}$.) In this case, $\text{env}_{\text{ew}}^{\mathcal{P}} = \mathcal{I}(\textbf{loop}\ \ldots\ \textbf{end loop stow}(l)\ \textit{cd\_suffix})(\text{env}_i^{\mathcal{P}})$. Let $\text{env}_l^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{loop}\ \ldots\ \textbf{end loop stow}(l))(\text{env}_i^{\mathcal{P}})$. Hence, $\text{env}_{\text{ew}}^{\mathcal{P}} = \mathcal{I}(\textit{cd\_suffix})(\text{env}_l^{\mathcal{P}})$.

Case $AE(env_l^{\mathcal{P}}) = CF$.

Case $\neg\mathcal{I}(Inv[x \rightsquigarrow x_i, \#x \rightsquigarrow x_i])(env_i^{\mathcal{P}})$. In this case we take $\text{env}_I^{\mathcal{M}} = \text{env}_I^{\mathcal{P}}$ so that $\text{env}_i^{\mathcal{M}} = \text{env}_i^{\mathcal{P}}$. Then we have $\neg\mathcal{I}((\text{Br\_Cd}) \Rightarrow (\text{Inv}[\text{x} \rightsquigarrow \text{x}_i, \#\text{x} \rightsquigarrow \text{x}_i]))(\text{env}_i^{\mathcal{M}})$. Therefore, $\text{AE}(\mathcal{I}(\textbf{confirm}\ (\text{Br\_Cd}) \Rightarrow (\text{Inv}[\text{x} \rightsquigarrow \text{x}_i, \#\text{x} \rightsquigarrow \text{x}_i]))(\text{env}_i^{\mathcal{M}})) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$.

```
P ≝ C\                                   env_I^P
      prec_top_lev_code
      alter all
      stow(i)
      ACseq_0
                                         env_i^P
    | whenever Br_Cd do
    |    loop
    |       maintaining Inv⌊x, #x⌋
    |       while b_p_e do
    |          stow(j)
    |          cd_kern
    |          stow(k)
    |          ACseq
    |    end loop
    |    stow(l)
    |                                    env_l^P
    |    cd_suffix
    | end whenever
                                         env_ew^P
      fol_top_lev_code
                                         env_F^P
```

Figure 81: Environments Defined for $\mathcal{P}$ of the **loop while** Rule

$\mathcal{M} \stackrel{\text{def}}{=} C\backslash$ $\quad \text{env}_I^{\mathcal{M}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$

$\quad \text{env}_i^{\mathcal{M}}$

confirm (Br_Cd) $\Rightarrow$ (Inv[x $\rightsquigarrow$ x$_i$, #x $\rightsquigarrow$ x$_i$])
alter all
stow(j)

$\quad \text{env}_j^{\mathcal{M}}$

whenever (Br_Cd) $\wedge$ (MExp(b_p_e)[y $\rightsquigarrow$ y$_i$]) **do**
  assume (MExp(b_p_e)[y $\rightsquigarrow$ y$_j$]) $\wedge$ (Inv[x $\rightsquigarrow$ x$_j$, #x $\rightsquigarrow$ x$_i$])
  cd_kern
  stow(k)
  ACseq
  confirm Inv[x $\rightsquigarrow$ x$_k$, #x $\rightsquigarrow$ x$_i$]
end whenever
alter all
stow(l)

$\quad \text{env}_l^{\mathcal{M}}$

whenever Br_Cd **do**
  assume ($\neg$(MExp(b_p_e)[y $\rightsquigarrow$ y$_l$])) $\wedge$ (Inv[x $\rightsquigarrow$ x$_l$, #x $\rightsquigarrow$ x$_i$])
  cd_suffix
end whenever

$\quad \text{env}_{\text{ew}}^{\mathcal{M}}$

fol_top_lev_code

$\quad \text{env}_F^{\mathcal{M}}$

Figure 82: Environments Defined for $\mathcal{M}$ of the **loop while** Rule

Case $\mathcal{I}(Inv[x \rightsquigarrow x_i, \#x \rightsquigarrow x_i])(env_i^{\mathcal{P}})$. (We are still inside case $\mathrm{AE}(\mathrm{env}_l^{\mathcal{P}}) = \mathrm{CF}$.) Therefore, there exists an environment $\mathrm{env}_e^{\mathcal{P}} \stackrel{\mathrm{def}}{=} [\mathrm{NL}, \mathrm{cs}_e, \mathrm{ns}_e, \mathrm{se}, \mathrm{os}_e, \mathrm{d}]$ where

$$\mathrm{ns}_e(h) = \mathrm{ns}(h) \text{ if } h < j \text{ or } k < h \tag{4.66}$$

$$\mathrm{os}_e(\psi) = \begin{cases} \mathrm{cs}_i(\xi) & \text{if } \psi = \#\xi \\ \mathrm{os}(\#^{k+1}\xi) & \text{if } \psi = \#^{k+2}\xi \end{cases} \tag{4.67}$$

that is the result of executing the loop zero or more iterations beginning in environment $\mathrm{env}_i^{\mathcal{P}}$ such that $\mathcal{I}(b\_p\_e)(\mathrm{env}_e^{\mathcal{P}})$, $\mathcal{I}(\mathrm{Inv}\lfloor \mathrm{x}, \#\mathrm{x} \rfloor)(\mathrm{env}_e^{\mathcal{P}})$, and execution of the loop's body beginning in environment $\mathrm{env}_e^{\mathcal{P}}$ produces either a categorically false environment or a neutral environment in which the loop invariant is interpreted to be false. That is to say, $\mathrm{AE}(\mathcal{I}(\mathbf{stow}(j)\ cd\_kern\ \mathbf{stow}(k)\ ACseq)(\mathrm{env}_e^{\mathcal{P}})) = \mathrm{CF}$ or $\neg\mathcal{I}(\mathrm{Inv}\lfloor \mathrm{x}, \#\mathrm{x} \rfloor)(\mathcal{I}(\mathbf{stow}(j)\ cd\_kern\ \mathbf{stow}(k)\ ACseq)(\mathrm{env}_e^{\mathcal{P}}))$. We take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_e \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_e \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Hence, $\mathcal{I}((\mathrm{Br\_Cd}) \Rightarrow (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_i, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_i^{\mathcal{M}})$. Therefore, $\mathrm{env}_j^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_e, \mathrm{ns}_j, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_j(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \neq j \\ \mathrm{cs}_e & \text{if } h = j \end{cases} . \tag{4.68}$$

Hence,

$$\mathrm{ns}_j(h) = \mathrm{ns}(h) \text{ if } h < j \text{ or } k < h. \tag{4.69}$$

All components except the index state and the old state of $\mathrm{env}_j^{\mathcal{M}}$ and $\mathrm{env}_e^{\mathcal{P}}$ are equal, and $\mathrm{ns}_j(h) = \mathrm{ns}_e(h)$ if $h < j$ or $k < h$. Due to the properties of $\mathrm{env}_e^{\mathcal{P}}$, and because the value of the old state has no effect on the interpretation of $\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i]$, *cd_kern* $\mathbf{stow}(k)$ *ACseq*, and $\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_k, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i]$, we know the following to be true. $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_j^{\mathcal{M}})$. $\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_j^{\mathcal{M}})$. $\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j])(\mathrm{env}_j^{\mathcal{M}})$. $\mathcal{I}(\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_j^{\mathcal{M}})$. $\mathcal{I}(\mathbf{whenever}\ (\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])\ \mathbf{do}\ \ldots\ \mathbf{end\ whenever})(\mathrm{env}_j^{\mathcal{M}}) = \mathcal{I}(\mathbf{assume}\ (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j]) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_kern\ \mathbf{stow}(k)\ ACseq\ \mathbf{confirm}\ \mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_k, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_j^{\mathcal{M}})$. $\mathrm{AE}(\mathcal{I}(\mathbf{assume}\ (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j]) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_kern\ \mathbf{stow}(k)\ ACseq)(\mathrm{env}_j^{\mathcal{M}})) = \mathrm{CF}$ or $\neg\mathcal{I}(\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_k, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathcal{I}(\mathbf{assume}\ (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j]) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_kern\ \mathbf{stow}(k)\ ACseq)(\mathrm{env}_j^{\mathcal{M}}))$. In either case, $\mathrm{AE}(\mathcal{I}(\mathbf{assume}\ (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j]) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_kern\ \mathbf{stow}(k)\ ACseq\ \mathbf{confirm}\ \mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_k, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_j^{\mathcal{M}})) = \mathrm{CF}$. Therefore, $\mathrm{AE}(\mathcal{I}(\mathbf{whenever}\ (\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])\ \mathbf{do}\ \ldots\ \mathbf{end\ whenever})(\mathrm{env}_j^{\mathcal{M}})) = \mathrm{CF}$. Hence, by Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.

Case AE($env_l^{\mathcal{P}}$) $\neq$ CF. In this case, by Lemmas 4.17 and 4.19, we have $\mathrm{env}_l^{\mathcal{P}}$ defined and $\mathrm{AE}(\mathrm{env}_l^{\mathcal{P}}) = \mathrm{NL}$. We may write $\mathrm{env}_l^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_l, \mathrm{ns}_l^{\mathcal{P}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_l^{\mathcal{P}}(h) = \mathrm{ns}(h) \text{ if } h < j \text{ or } l < h \tag{4.70}$$
$$\mathrm{ns}_l^{\mathcal{P}}(l) = \mathrm{cs}_l. \tag{4.71}$$

We take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_i, \mathrm{cs}_l \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_i, \mathrm{cs}_l \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Because the loop terminated in a neutral environment when interpreted in $\mathrm{env}_i^{\mathcal{P}}$, its invariant was satisfied at the beginning of each iteration. In particular it was satisfied at the beginning of the first iteration. Therefore, $\mathcal{I}(\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_i, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_j^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}_j^{\mathcal{M}}, \langle \mathrm{cs}_l \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where $\mathrm{ns}_j^{\mathcal{M}}(h) = \mathrm{ns}(h)$ if $h \neq j$.

Case loop had at least one iteration when interpreted in $env_i^{\mathcal{P}}$. In this case, $\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_j^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_l^{\mathcal{M}} = \mathcal{I}(\textbf{assume } (\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_j]) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_j, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_kern\ \textbf{stow}(k)\ ACseq\ \textbf{confirm}\ \mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_k, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i]\ \textbf{alter all stow}(l))(\mathrm{env}_j^{\mathcal{M}})$. Because the loop terminated in a neutral environment when interpreted in $\mathrm{env}_i^{\mathcal{P}}$, $\mathrm{env}_l^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_l, \mathrm{ns}'_l, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where $\mathrm{ns}'_l(h) = \mathrm{ns}(h)$ if $h < j$ or $l < h$.

Case loop had zero iterations when interpreted in $env_i^{\mathcal{P}}$. In this case, $\neg\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_j^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_l^{\mathcal{M}} = \mathcal{I}(\textbf{alter all stow}(l))(\mathrm{env}_j^{\mathcal{M}})$. Therefore, $\mathrm{env}_l^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_l, \mathrm{ns}''_l, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where $\mathrm{ns}''_l(h) = \mathrm{ns}(h)$ if $h < j$ or $l < h$.

We now return to the argument for the case $\mathrm{AE}(\mathrm{env}_l^{\mathcal{P}}) \neq \mathrm{CF}$. Whether the loop had zero or more iterations when interpreted in $\mathrm{env}_i^{\mathcal{P}}$, $\mathrm{env}_l^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_l, \mathrm{ns}_l^{\mathcal{M}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_l^{\mathcal{M}}(h) = \mathrm{ns}(h) \text{ if } h < j \text{ or } l < h \tag{4.72}$$
$$\mathrm{ns}_l^{\mathcal{M}}(l) = \mathrm{cs}_l. \tag{4.73}$$

Because $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}} = \mathcal{I}(cd\_suffix)(\mathrm{env}_l^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}})$, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_l^{\mathcal{P}})$. Because the loop terminated in a neutral environment when interpreted in $\mathrm{env}_i^{\mathcal{P}}$, $\mathcal{I}((\neg(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_l])) \wedge (\mathrm{Inv}[\mathrm{x} \rightsquigarrow \mathrm{x}_l, \#\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_l^{\mathcal{P}})$. Therefore, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_l^{\mathcal{M}})$. Let $A \stackrel{\mathrm{def}}{=} \{h | j \leq h \leq k\}$. Because $\mathrm{Equal\_except\_at}(A, \mathrm{env}_l^{\mathcal{P}}, \mathrm{env}_l^{\mathcal{M}})$, two applications of Lemma 4.37 give us $\mathrm{Equal\_except\_at}(A, \mathrm{env}_F^{\mathcal{P}}, \mathrm{env}_F^{\mathcal{M}})$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.41 *The bridge rule preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. We may write $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. We take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}, \mathrm{ns}, \langle \mathrm{cs} \rangle \circ \mathrm{se}, \mathrm{os}', \mathrm{d}]$ where

$$\mathrm{os}'(\psi) = \begin{cases} \mathrm{cs}(\xi) & \text{if } \psi = \#\xi \\ \mathrm{os}(\#^{k+1}\xi) & \text{if } \psi = \#^{k+2}\xi \end{cases} . \tag{4.74}$$

By the semantics of **whenever**,

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I} \left(\begin{array}{l} \textbf{alter all} \\ \textbf{stow}(i) \\ \textbf{assume } \mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i) \\ \mathrm{Stows\_added}(p\_body) \\ \textbf{stow}(j) \\ \textbf{confirm } \mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{x}_i, \mathrm{x} \rightsquigarrow \mathrm{x}_j] \end{array} \right) (\mathrm{env}_I^{\mathcal{M}}). \tag{4.75}$$

Let

$$\mathrm{env}_j^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I} \left(\begin{array}{l} \textbf{alter all} \\ \textbf{stow}(i) \\ \textbf{assume } \mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i) \\ \mathrm{Stows\_added}(p\_body) \\ \textbf{stow}(j) \end{array} \right) (\mathrm{env}_I^{\mathcal{M}}). \tag{4.76}$$

By Lemma 4.16, $\mathcal{I}(\mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z}))(\mathcal{I}(\textbf{remember})(\mathrm{env}_I^{\mathcal{P}}))$. Therefore, because $\mathrm{ISE}(\mathcal{I}(\textbf{alter all } \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}}))(i) = \mathrm{cs}$, $\mathcal{I}(\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i))(\mathcal{I}(\textbf{alter all } \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}}))$. Note also that $\mathrm{CSE}(\mathcal{I}(\textbf{alter all } \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}})) = \mathrm{cs}$.

Case $AE(\mathcal{I}(\textbf{remember assume } pre\lfloor x \rfloor \wedge \textbf{is\_initial}(z)\ \mathrm{p\_body})(env_I^{\mathcal{P}})) = CF$. Because *p_body* contains no indexed variables and the interpretation of any **stow** statement changes at most the index state of an environment, $\mathrm{AE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.

Case $AE(\mathcal{I}(\textbf{remember assume } pre\lfloor x \rfloor \wedge \textbf{is\_initial}(z)\ \mathrm{p\_body})(env_I^{\mathcal{P}})) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathcal{I}(\textbf{remember assume } \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}})) = \mathrm{NL}$. Note that $\mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(i) = \mathrm{cs}$. Because *p_body* contains no indexed variables and the interpretation of any **stow** statement changes at most the index state of an environment, $\mathrm{CSE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{CSE}(\mathcal{I}(\textbf{remember assume } \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))$.

Note also that $\mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(j) = \mathrm{CSE}(\mathrm{env}_j^{\mathcal{M}})$. In this case, it must be that $\neg\mathcal{I}(\mathrm{post}\lfloor\#\mathrm{x},\mathrm{x}\rfloor)(\mathcal{I}(\textbf{remember assume } \mathrm{pre}\lfloor\mathrm{x}\rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))$. Because $\mathrm{OSE}(\mathcal{I}(\textbf{remember assume } \mathrm{pre}\lfloor\mathrm{x}\rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))(\#\xi) = \mathrm{cs}(\xi)$, $\neg\mathcal{I}(\mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{x}_i, \mathrm{x} \rightsquigarrow \mathrm{x}_j])(\mathrm{env}_j^{\mathcal{M}})$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.42 *The rule for procedure call preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is organized by cases. First we make the following two definitions.

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{P}}) \tag{4.77}$$

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{M}}) \tag{4.78}$$

We will be defining other environments as well. Figure 83 shows the positions of the environments defined for $\mathcal{P}$ and for $\mathcal{M}$.

Case $AE(env_i^{\mathcal{P}}) = CF$. In this case, we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. Then $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.

Case $AE(env_i^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{NL}$. We may, for this case, write $\mathrm{env}_i^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$, where $\mathrm{ns}(i) = \mathrm{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. As shown in Figure 83, we let $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{whenever } \mathrm{Br\_Cd} \textbf{ do } \ldots \textbf{ end whenever})(\mathrm{env}_i^{\mathcal{P}})$.

Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. (We are still inside case $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) \neq \mathrm{CF}$, i.e., $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{NL}$.) In this case, $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}} = \mathrm{env}_i^{\mathcal{P}}$, and we take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_i \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_i \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Figure 83 shows our definition of $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}}$ with respect to $\mathrm{env}_i^{\mathcal{M}}$. By Lemma 4.22, the value of $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env})$ depends only on $\mathrm{ISE}(\mathrm{env})$ for any environment env. So, because $\mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{ISE}(\mathrm{env}_i^{\mathcal{P}})$, we have $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$. Therefore, we have another equality for $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}}$, namely,

$$\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}} = \mathcal{I}(\textbf{alter all stow}(j))(\mathrm{env}_i^{\mathcal{M}}). \tag{4.79}$$

Therefore, $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}', \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}'(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \neq j \\ \mathrm{cs}_i & \text{if } h = j \end{cases} . \tag{4.80}$$

So, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_{\mathrm{ew}}^{\mathcal{P}}, \mathrm{env}_{\mathrm{ew}}^{\mathcal{M}})$. The set $\{j\}$ is a subset of the set of indices h that appear in $\textbf{stow}(h)$ statements anywhere within the **whenever**

$\mathcal{P} \stackrel{\text{def}}{=} C\backslash$ $\quad \text{env}_I^{\mathcal{P}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$

$\quad \text{env}_i^{\mathcal{P}}$

whenever Br_Cd **do**
  P_nm(ac, ad)
  stow(j)

$\quad \text{env}_j^{\mathcal{P}}$

  cd_suffix
end whenever

$\quad \text{env}_{\text{ew}}^{\mathcal{P}}$

fol_top_lev_code

$\quad \text{env}_F^{\mathcal{P}}$

$\mathcal{M} \stackrel{\text{def}}{=} C\backslash$ $\quad \text{env}_I^{\mathcal{M}}$

prec_top_lev_code
alter all
stow(i)
$ACseq_0$

$\quad \text{env}_i^{\mathcal{M}}$

confirm (Br_Cd) $\Rightarrow$ (pre[x $\rightsquigarrow$ ac$_i$, y $\rightsquigarrow$ ad$_i$, z $\rightsquigarrow$ z$_i$])
alter all
stow(j)

$\quad \text{env}_j^{\mathcal{M}}$

whenever Br_Cd **do**
  assume b$_j$ = b$_i$ $\wedge$ (post[#x $\rightsquigarrow$ ac$_i$, x $\rightsquigarrow$ ac$_j$,
    #y $\rightsquigarrow$ ad$_i$, y $\rightsquigarrow$ ad$_j$,
    #z $\rightsquigarrow$ z$_i$, z $\rightsquigarrow$ z$_j$])
  cd_suffix
end whenever

$\quad \text{env}_{\text{ew}}^{\mathcal{M}}$

fol_top_lev_code

$\quad \text{env}_F^{\mathcal{M}}$

Figure 83: Environments Defined for $\mathcal{P}$ and $\mathcal{M}$ of the Rule for Procedure Call

statement explicitly shown in the schema $\mathcal{P}$. In this case, by Lemma 4.36, Equal_except_at($\{j\}, \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}^{\mathcal{P}}_{\mathrm{ew}}), \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}^{\mathcal{M}}_{\mathrm{ew}})$). Hence, Equal_except_at($\{j\}, \mathrm{env}^{\mathcal{P}}_F, \mathrm{env}^{\mathcal{M}}_F$). Therefore, $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_F) = \mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$.
Case $\mathcal{I}(\mathit{Br\_Cd})(\mathit{env}^{\mathcal{P}}_i)$. (We are still inside case $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_i) \neq \mathrm{CF}$, i.e., $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_i) = \mathrm{NL}$.) In this case, $\mathrm{env}^{\mathcal{P}}_{\mathrm{ew}} = \mathcal{I}(\mathrm{P\_nm}(\mathrm{ac}, \mathrm{ad})\ \mathbf{stow}(j)\ \mathit{cd\_suffix})(\mathrm{env}^{\mathcal{P}}_i)$, and $\mathrm{env}^{\mathcal{M}}_{\mathrm{ew}} = \mathcal{I}(\mathbf{assume}\ \mathrm{b}_j = \mathrm{b}_i \wedge (\mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{x} \rightsquigarrow \mathrm{ac}_j, \#\mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_j, \#\mathrm{z} \rightsquigarrow \mathrm{z}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_j])\ \mathit{cd\_suffix})(\mathrm{env}^{\mathcal{M}}_j)$. Let $\mathrm{env}^{\mathcal{P}}_j \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathrm{P\_nm}(\mathrm{ac}, \mathrm{ad})\ \mathbf{stow}(j))(\mathrm{env}^{\mathcal{P}}_i)$. Let $\mathrm{cs}_j \stackrel{\mathrm{def}}{=} \mathrm{CSE}(\mathrm{env}^{\mathcal{P}}_j)$. In this case, we take $\mathrm{env}^{\mathcal{M}}_I = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_j \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}^{\mathcal{M}}_i = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_j \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Because $\mathcal{P}$ is syntactically correct, the declaration-meaning d of environment $\mathrm{env}^{\mathcal{P}}_I$ contains d(P_nm). Let $\mathrm{d}(\mathrm{P\_nm}) \stackrel{\mathrm{def}}{=} [\mathrm{dp}, \mathrm{ep}, \mathrm{pf}, \mathrm{sf}]$. Because $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_I) = \mathrm{NL}$, we have that dp is the same as pre, ep is the same as post, and d(P_nm) is conformal.
Case $\mathit{AE}(\mathit{env}^{\mathcal{P}}_j) = \mathit{CF}$. Because d(P_nm) is conformal, if $\mathrm{dp}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}))$, then $\mathrm{sf}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z})) \neq \mathrm{CF}$. Therefore, $\neg\mathrm{dp}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}))$. Hence, $\neg\mathcal{I}(\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_i])(\mathrm{env}^{\mathcal{M}}_i)$. Therefore $\mathrm{AE}(\mathcal{I}(\mathbf{confirm}\ (\mathrm{Br\_Cd}) \Rightarrow (\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_i]))(\mathrm{env}^{\mathcal{M}}_i)) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_F) = \mathrm{CF}$.
Case $\mathit{AE}(\mathit{env}^{\mathcal{P}}_j) \neq \mathit{CF}$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_j) = \mathrm{NL}$. Therefore, $\mathrm{dp}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}))$, and $\mathrm{sf}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z})) = \mathrm{NL}$. Hence, $\mathcal{I}(\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_i])(\mathrm{env}^{\mathcal{M}}_i)$. Therefore, $\mathrm{env}^{\mathcal{M}}_j = [\mathrm{NL}, \mathrm{cs}_j, \mathrm{ns}_j, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_j(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \neq j \\ \mathrm{cs}_j & \text{if } h = j \end{cases} . \tag{4.81}$$

Hence, $\mathrm{env}^{\mathcal{M}}_j = \mathrm{env}^{\mathcal{P}}_j$. Because d(P_nm) is conformal, $\mathrm{ep}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}), \mathrm{cs}_j(\mathrm{ac}), \mathrm{cs}_j(\mathrm{ad}), \mathrm{cs}_j(\mathrm{z}))$. Furthermore, if $\xi \notin \{\mathrm{ac}, \mathrm{ad}, \mathrm{z}\}$, then $\mathrm{cs}_j(\xi) = \mathrm{cs}_i(\xi)$. Because $\mathrm{ns}_j(j) = \mathrm{cs}_j$ and $\mathrm{ns}_j(i) = \mathrm{cs}_i$, $\mathcal{I}(\mathbf{assume}\ \mathrm{b}_j = \mathrm{b}_i \wedge (\mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{x} \rightsquigarrow \mathrm{ac}_j, \#\mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_j, \#\mathrm{z} \rightsquigarrow \mathrm{z}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_j]))(\mathrm{env}^{\mathcal{M}}_j) = \mathrm{env}^{\mathcal{M}}_j$. Therefore, $\mathrm{env}^{\mathcal{M}}_{\mathrm{ew}} = \mathrm{env}^{\mathcal{P}}_{\mathrm{ew}}$, and $\mathrm{env}^{\mathcal{M}}_F = \mathrm{env}^{\mathcal{P}}_F$. Hence, $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_F) = \mathrm{CF}$. □

Lemma 4.43 *The rule for* **assume** *preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}^{\mathcal{P}}_I$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}^{\mathcal{P}}_F = \mathcal{I}(\mathcal{P})(\mathrm{env}^{\mathcal{P}}_I)$; i.e., $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_I) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}^{\mathcal{M}}_I$, such that $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_I) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_F) = \mathrm{CF}$, where $\mathrm{env}^{\mathcal{M}}_F = \mathcal{I}(\mathcal{M})(\mathrm{env}^{\mathcal{M}}_I)$. The proof is organized by cases. In each case we take $\mathrm{env}^{\mathcal{M}}_I = \mathrm{env}^{\mathcal{P}}_I$. First we make the following two definitions.

$$\mathrm{env}^{\mathcal{P}}_i \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \mathbf{alter\ all\ stow}(i)\mathit{ACseq}_0)(\mathrm{env}^{\mathcal{P}}_I) \tag{4.82}$$

$$\mathrm{env}^{\mathcal{M}}_i \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \mathbf{alter\ all\ stow}(i)\mathit{ACseq}_0)(\mathrm{env}^{\mathcal{M}}_I) \tag{4.83}$$

Consequently, $\mathrm{env}_i^{\mathcal{M}} = \mathrm{env}_i^{\mathcal{P}}$.
Case $AE(env_i^{\mathcal{P}}) = CF$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_i^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{NL}$.
Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. In this case, $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume } (\mathrm{Br\_Cd}) \Rightarrow (H)\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}((\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}})$. Therefore, $\mathcal{I}(\textbf{assume } (\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{env}_i^{\mathcal{M}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. In this case, $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textbf{assume } H\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume } (\mathrm{Br\_Cd}) \Rightarrow (H)\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}(\textbf{assume } (\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{assume } H)(\mathrm{env}_i^{\mathcal{M}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume } H\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.44 *The rule for* **confirm** *preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. First we make the following two definitions.

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{P}}) \tag{4.84}$$

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{M}}) \tag{4.85}$$

Consequently, $\mathrm{env}_i^{\mathcal{M}} = \mathrm{env}_i^{\mathcal{P}}$. Furthermore, $\mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}})(i) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}})$.
Case $AE(env_i^{\mathcal{P}}) = CF$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_i^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{NL}$.
Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{P}})$. In this case, $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm } (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}((\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_i^{\mathcal{M}})$. Therefore, $\mathcal{I}(\textbf{confirm } (\mathrm{Br\_Cd}) \Rightarrow (H)[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}}) =$

$\mathrm{env}_i^{\mathcal{M}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $\mathcal{I}(\mathit{Br\_Cd})(\mathit{env}_i^{\mathcal{P}})$. In this case, $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textbf{confirm } H[\mathrm{x}\rfloor \mathit{\ cd\_suffix\ fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm } (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ \mathit{cd\_suffix\ fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}(\textbf{confirm } (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{confirm } H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm } H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]\ \mathit{cd\_suffix\ fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}})(i) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}})$, $\mathcal{I}(\textbf{confirm } H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{confirm } H[\mathrm{x}\rfloor)(\mathrm{env}_i^{\mathcal{P}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.45 *The rule for empty guarded blocks preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. We take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$ and make the following two definitions.

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{P}}) \tag{4.86}$$

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{M}}) \tag{4.87}$$

Consequently, $\mathrm{env}_i^{\mathcal{M}} = \mathrm{env}_i^{\mathcal{P}}$.
Case $\mathit{AE}(\mathit{env}_i^{\mathcal{P}}) = \mathit{CF}$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $\mathit{AE}(\mathit{env}_i^{\mathcal{P}}) \neq \mathit{CF}$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{NL}$. Whether $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$ or $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$, $\mathcal{I}(\textbf{whenever } \mathrm{Br\_Cd}\ \textbf{do end whenever})(\mathrm{env}_i^{\mathcal{P}}) = \mathcal{I}(\varepsilon)(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{env}_i^{\mathcal{P}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{P}})$. Because $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.46 *The rule for* **alter all** *preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is

organized by cases. First we make the following four definitions.

$$\begin{aligned}
\mathrm{env}_1^{\mathcal{P}} &\stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec_top_lev_code})(\mathrm{env}_I^{\mathcal{P}}) && (4.88)\\
\mathrm{env}_a^{\mathcal{P}} &\stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{alter all})(\mathrm{env}_1^{\mathcal{P}}) && (4.89)\\
\mathrm{env}_s^{\mathcal{P}} &\stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{stow}(i))(\mathrm{env}_a^{\mathcal{P}}) && (4.90)\\
\mathrm{env}_1^{\mathcal{M}} &\stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec_top_lev_code})(\mathrm{env}_I^{\mathcal{M}}) && (4.91)
\end{aligned}$$

Consequently,

$$\begin{aligned}
\mathrm{env}_F^{\mathcal{P}} &= \mathcal{I}(\mathit{fol_top_lev_code})(\mathrm{env}_s^{\mathcal{P}}) && (4.92)\\
\mathrm{env}_F^{\mathcal{M}} &= \mathcal{I}(\mathit{fol_top_lev_code})(\mathrm{env}_1^{\mathcal{M}}). && (4.93)
\end{aligned}$$

Case $AE(env_1^{\mathcal{P}}) = CF$. In this case, we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. Hence, $\mathrm{env}_1^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. We may, for this case, write $\mathrm{env}_1^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_1, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. With the help of Lemmas 4.20 and 4.27, we may also write $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. In this case, we take $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}'_I, \mathrm{se}_I \circ \mathrm{tail}(\mathrm{se}), \mathrm{os}, \mathrm{d}]$, where

$$\mathrm{ns}'_I(h) = \begin{cases} \mathrm{ns}_I(h) & \text{if } h \neq i \\ \mathrm{first}(\mathrm{se}) & \text{if } h = i \end{cases} . \tag{4.94}$$

By Lemma 4.18, $\mathrm{AE}(\mathrm{env}_a^{\mathcal{P}}) \neq \mathrm{VT}$. Then, by the semantics of **alter all**, $\mathrm{AE}(\mathrm{env}_a^{\mathcal{P}}) = \mathrm{NL}$, $\mathrm{se} \neq \varepsilon$, and $\mathrm{env}_a^{\mathcal{P}} = [\mathrm{NL}, \mathrm{first}(\mathrm{se}), \mathrm{ns}, \mathrm{tail}(\mathrm{se}), \mathrm{os}, \mathrm{d}]$. By the semantics of $\textbf{stow}(i)$, $\mathrm{env}_s^{\mathcal{P}} = [\mathrm{NL}, \mathrm{first}(\mathrm{se}), \mathrm{ns}', \mathrm{tail}(\mathrm{se}), \mathrm{os}, \mathrm{d}]$, where

$$\mathrm{ns}'(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \neq i \\ \mathrm{first}(\mathrm{se}) & \text{if } h = i \end{cases} . \tag{4.95}$$

By the syntax restriction that references to index i occur only after $\textbf{stow}(i)$ (page 68), no statement of *prec_top_lev_code* contains a reference to any variable indexed with i. Because ns'_I differs from ns_I only at i, $\mathrm{env}_1^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_1, \mathrm{ns}'', \mathrm{tail}(\mathrm{se}), \mathrm{os}, \mathrm{d}]$, where

$$\mathrm{ns}''(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \neq i \\ \mathrm{ns}'_I(h) & \text{if } h = i \end{cases} . \tag{4.96}$$

Note that $\mathrm{ns}'' = \mathrm{ns}'$. Therefore, $\mathrm{env}_1^{\mathcal{M}}$ differs from $\mathrm{env}_s^{\mathcal{P}}$ only in the current state. Due to the additional syntactic restriction of the rule for **alter all**, *fol_top_lev_code* contains only **alter all**, **stow**, **assume**, and **confirm** statements. The **assume** and **confirm** statements refer only to indexed variables. Hence, **stow** is the only one

of these statements whose semantics is affected by the current state; however, every one of these **stow** statements is immediately preceded by an **alter all** statement. Therefore, $\mathrm{CSE}(\mathrm{env}_1^{\mathcal{M}})$ and $\mathrm{CSE}(\mathrm{env}_s^{\mathcal{P}})$ have no effect on the values, respectively, of $\mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_1^{\mathcal{M}})$ and $\mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_s^{\mathcal{P}})$. Therefore, $\mathrm{env}_F^{\mathcal{M}}$ and $\mathrm{env}_F^{\mathcal{P}}$ differ at most in the current state. In particular, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.47 *The rule for consecutive* **assume** *statements preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. First we make the following five definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.97}$$
$$\mathrm{env}_2^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_1)(\mathrm{env}_1^{\mathcal{P}}) \tag{4.98}$$
$$\mathrm{env}_b^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_2)(\mathrm{env}_2^{\mathcal{P}}) \tag{4.99}$$
$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.100}$$
$$\mathrm{env}_b^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } (H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}}) \tag{4.101}$$

Consequently,

$$\mathrm{env}_1^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{P}} \tag{4.102}$$
$$\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{P}}) \tag{4.103}$$
$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{M}}). \tag{4.104}$$

Case $AE(env_1^{\mathcal{P}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. By Lemma 4.18 and the semantics of the **assume** statement, $\mathrm{AE}(\mathrm{env}_2^{\mathcal{P}}) = \mathrm{NL}$. Therefore, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Similarly, $\mathrm{env}_b^{\mathcal{P}} = \mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Hence, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$ and $\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{P}})$. Therefore, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$ and $\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. Hence, $\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$. Therefore, by the semantics of the **assume** statement, $\mathrm{env}_b^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{M}}$. Hence, $\mathrm{env}_b^{\mathcal{M}} = \mathrm{env}_b^{\mathcal{P}}$. Therefore, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Hence, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.48 *The* **assume-confirm** *rule preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. First we make the following three definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.105}$$

$$\mathrm{env}_2^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_1)(\mathrm{env}_1^{\mathcal{P}}) \tag{4.106}$$

$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.107}$$

Consequently,

$$\mathrm{env}_1^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{P}} \tag{4.108}$$

$$\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textbf{confirm } H_2)(\mathrm{env}_2^{\mathcal{P}}) \tag{4.109}$$

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\mathrm{env}_1^{\mathcal{M}}). \tag{4.110}$$

Case $AE(env_1^{\mathcal{P}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. By Lemma 4.18 and the semantics of the **assume** statement, $\mathrm{AE}(\mathrm{env}_2^{\mathcal{P}}) = \mathrm{NL}$. Therefore, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Hence, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. By substitution of equals, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$. By the semantics of the **confirm** statement, we have $\neg\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. By substitution of equals, $\neg\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. Therefore, $\neg\mathcal{I}((H_1) \Rightarrow (H_2))(\mathrm{env}_1^{\mathcal{M}})$. By the semantics of the **confirm** statement, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.49 *The rule for consecutive* **confirm** *statements preserves invalidity in the math direction.*

Proof. We assume that $\mathcal{P}$ is invalid and show that $\mathcal{M}$ is invalid. Let $\mathrm{env}_I^{\mathcal{P}}$ be a witness to $\mathcal{P}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{M}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{M}} = \mathrm{env}_I^{\mathcal{P}}$. First we make the following five definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.111}$$

$$\mathrm{env}_2^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{confirm } H_1)(\mathrm{env}_1^{\mathcal{P}}) \tag{4.112}$$
$$\mathrm{env}_b^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{confirm } H_2)(\mathrm{env}_2^{\mathcal{P}}) \tag{4.113}$$
$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.114}$$
$$\mathrm{env}_b^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}}) \tag{4.115}$$

Consequently,

$$\mathrm{env}_1^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{P}} \tag{4.116}$$
$$\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{P}}) \tag{4.117}$$
$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{M}}). \tag{4.118}$$

Case $AE(env_1^{\mathcal{P}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{P}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$.
Case $AE(env_2^{\mathcal{P}}) = CF$. By the semantics of the **confirm** statement, $\neg\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. By substitution of equals, $\neg\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$. By the semantics of the **confirm** statement, $\mathrm{AE}(\mathrm{env}_b^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_2^{\mathcal{P}}) \neq CF$. Hence, $\mathrm{AE}(\mathrm{env}_2^{\mathcal{P}}) = \mathrm{NL}$. By the semantics of the **confirm** statement, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$ and $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. By substitution of equals, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$.
Case $AE(env_b^{\mathcal{P}}) = CF$. By the semantics of the **confirm** statement, $\neg\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. By substitution of equals, $\neg\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. By the semantics of the **confirm** statement, $\mathrm{AE}(\mathrm{env}_b^{\mathcal{M}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$.
Case $AE(env_b^{\mathcal{P}}) \neq CF$. Hence, $\mathrm{AE}(\mathrm{env}_b^{\mathcal{P}}) = \mathrm{NL}$. By the semantics of the **confirm** statement, $\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$ and $\mathrm{env}_b^{\mathcal{P}} = \mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{M}}$. By substitution of equals, $\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. Recall that $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$. Therefore, $\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$. By the semantics of the **confirm** statement, $\mathrm{env}_b^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{M}}$. Therefore, $\mathrm{env}_b^{\mathcal{M}} = \mathrm{env}_b^{\mathcal{P}}$ and $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Hence, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. □

Lemma 4.50 *The rule of inference bridging predicate logic and the indexed method preserves invalidity in the math direction.*

Proof. We suppose that **confirm** H is an invalid program. That is to say, there exists a neutral environment env such that $\mathrm{AE}(\textbf{confirm } H) = \mathrm{CF}$. By the semantics of the **confirm** statement, the component states of env make an assignment of values to the variables in H that renders H false in some model of context C's logic. Consequently, the mathematical assertion H is false (i.e., invalid) in context C. □

4.5 Relative Completeness Lemmas

4.5.1 Related Valid Program Differing in Assertions Only

Our first obligation regarding relative completeness is to establish Lemma 4.4.

Proof of Lemma 4.4. Because the lemma assumes the assertion language to be expressive, for each internal procedure there exists an assertion expressing the post relation corresponding to that procedure's precondition and its body. By the same assumption, for each loop there exists an assertion expressing its tightest loop invariant. Therefore, Prog′ of the lemma exists. It suffices to show that Prog′ is valid.

Consider procedure P_nm with precondition pre and postcondition (in Prog) post. In Prog′ P_nm has postcondition post′. By the definition of post′ and because Prog is valid, interpretation in Prog′—in a neutral environment—of P_nm's declaration does not produce a categorically false environment. Interpretation of any call to P_nm is the same in Prog and Prog′ because the only difference between the declaration meanings of P_nm in the two programs is in the effect predicate, which plays no role in the interpretation of a procedure call.

Consider a loop with invariant (in Prog) Inv. In Prog′ this loop has invariant Inv′. Because Inv′ is by definition an invariant for the loop, and because Prog is valid, interpretation of the loop in Prog′—in a neutral environment—does not produce a categorically false environment. Therefore, the interpretations of the two loops in identical environments produce identical environments.

The interpretations of all other statements are unaffected by the replacements made to produce Prog′. Therefore, Prog′ is also valid. □

4.5.2 System of Rewrite Rules Is Terminating

If we fix the direction of application to one orientation, say the math direction, then the proof rules of Chapter III can be regarded as rewrite rules. We establish that this system of rewrite rules is terminating by proving Lemma 4.5.

Proof of Lemma 4.5. A well-prepared assertive program is an assertive program that conforms to the syntax of either (1) top level code or (2) $\mathcal{P}$ defined in the bridge rule (Figure 40). These two cases are mutually exclusive. The bridge rule is always applicable (in the math direction) in the latter case—never in the former case. Let us refer to an an assertive program that conforms to the syntax of top level code as a *top level program.* Application of the bridge rule in the math direction always produces a top level program. It remains to show that the process of rewriting any top level program in the math direction using the remaining rules of Chapter III always terminates with a mathematical assertion.

Table 2: Multipliers for Function Meas

multiplier	*statement*
1	**whenever**
1	**alter all**
1	**stow**
1	**assume** at top level
1	**confirm** at top level
2	**assume** inside **whenever**
2	**confirm** inside **whenever**
5	procedure call
11	**loop**
12	**if**

We define a total function from the set of top level programs into the natural numbers. This function, which we call Meas, gives a measure on the size of assertive programs. Then we show that, for each rule of Chapter III (other than the bridge rule), $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$. Therefore, the process of rewriting assertive programs in the math direction must terminate in finitely many steps.

Meas's value for a given assertive program is defined by (1) counting the number of occurrences of each statement in the program, recursively, including statements within statements; (2) multiplying the count for each statement by the corresponding number in Table 2; and (3) summing the resulting ten products.

The rule for **assume**'s $\mathcal{M}$ (Figure 44) has one fewer **assume** statement inside a **whenever** statement and one more **assume** statement at the top level than its $\mathcal{P}$. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 2 \cdot 1 + 1 \cdot 1$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$. As we examine the remaining rules, we are comparing the rule's $\mathcal{M}$ with its $\mathcal{P}$.

The rule for procedure call's $\mathcal{M}$ (Figure 46) has one fewer procedure call, one more **confirm** statement at the top level, one more **alter all** statement, and one more **assume** statement inside a **whenever** statement. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 5 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 1$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

$\mathcal{M}$ of the rule for selection in the absence of an **else** clause (Figure 48) has one fewer **if** statement, two more **alter all** statements, one more **whenever** statement, one more **assume** statement inside a **whenever** statement, and two more **assume** statements at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 12 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 +$

$2 \cdot 1 + 1 \cdot 2$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 5$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

$\mathcal{M}$ of the rule for selection in the presence of an **else** clause (Figures 49 and 50) has one fewer **if** statement, three more **alter all** statements, two more **whenever** statements, two more **assume** statements inside a **whenever** statement, and two more **assume** statements at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 12 \cdot 1 + 1 \cdot 3 + 1 \cdot 2 + 2 \cdot 2 + 1 \cdot 2$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

The **loop while** rule's $\mathcal{M}$ (Figures 53 and 54) has one fewer **loop** statement, one more **confirm** statement at the top level, two more **alter all** statements, one more **whenever** statement, two more **assume** statements inside a **whenever** statement, and one more **confirm** statement inside a **whenever** statement. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 11 \cdot 1 + 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 + 2 \cdot 2 + 2 \cdot 1$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

The rule for **confirm**'s $\mathcal{M}$ (Figure 55) has one fewer **confirm** statement inside a **whenever** statement and one more **confirm** statement at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 2 \cdot 1 + 1 \cdot 1$; i.e., $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

$\mathcal{M}$ of the rule for empty guarded blocks (Figure 59) has one fewer **whenever** statement. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1 \cdot 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

The rule for **alter all**'s $\mathcal{M}$ (Figure 62) has one fewer **alter all** statement, and one fewer **stow** statement. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1 \cdot 1 - 1 \cdot 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

$\mathcal{M}$ of the rule for consecutive **assume** statements (Figure 64) has one fewer **assume** statement at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1 \cdot 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

The **assume-confirm** rule's $\mathcal{M}$ (Figure 66) has one fewer **assume** statement at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1 \cdot 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

$\mathcal{M}$ of the rule for consecutive **confirm** statements (Figure 68) has one fewer **confirm** statement at the top level. Therefore, $\text{Meas}(\mathcal{M}) = \text{Meas}(\mathcal{P}) - 1 \cdot 1$. Hence, $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$.

We have shown that, for each rule of Chapter III (other than the bridge rule), $\text{Meas}(\mathcal{M}) < \text{Meas}(\mathcal{P})$. Therefore, the process of rewriting assertive programs in the math direction must terminate in finitely many steps. We must show, finally, that the result with which the process terminates is a single **confirm** statement so that the rule of inference bridging predicate logic and the indexed method can be applied to it to produce a mathematical assertion.

There is always at least one **confirm** statement in the result of applying a rule in the math direction because (1) the process begins with the bridge rule's $\mathcal{P}$ whose last statement is a **confirm** statement and (2) only one rule, the rule for consecutive

confirm statements, removes a **confirm** statement when applied in the math direction, and it leaves at least one **confirm** statement in the result. If a top level program contains a nonempty **whenever** statement, then at least one of the following rules can be applied to it in the math direction: the rule for **assume**, the procedure call rule, a rule for selection, the **loop while** rule, or the rule for **confirm**. The rule for empty guarded blocks can be applied to a top level program if it contains an empty **whenever** statement. If a top level program contains no **whenever** statements but has at least one **alter all** statement, the rule for **alter all** is applicable. If it contains consecutive **assume** statements, there is an (appropriately named!) applicable rule. If the last statement is a **confirm** statement and the penultimate statement is an **assume** statement, the **assume-confirm** rule can be applied. If the program contains consecutive **confirm** statements, there is an applicable rule. If none of the preceding conditions holds for the program, then the program consists of exactly one **confirm** statement. Therefore, the result with which the process terminates is a single **confirm** statement. Hence, the process of rewriting any well-prepared assertive program in the math direction using the rules of Chapter III always terminates with a mathematical assertion. □

4.5.3 Preservation of Invalidity in the Program Direction

At this point we establish Lemma 4.8 by proving a series of lemmas, one for each proof rule of Chapter III, excluding the procedure call rule and the **loop while** rule. Each lemma states that the rule preserves invalidity in the program direction. Skepticism may be greatest about the **if-then-else** statement; therefore, we present its lemma and proof first. The proof is more complicated than the others. Readers wishing a gentler introduction are invited to skip ahead to some shorter proofs before returning here.

Lemma 4.51 *The rule for selection in the presence of an* **else** *clause preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\text{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\text{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\text{env}_I^{\mathcal{M}})$; i.e., $\text{AE}(\text{env}_I^{\mathcal{M}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$. We will show the existence of an environment, $\text{env}_I^{\mathcal{P}}$, such that $\text{AE}(\text{env}_I^{\mathcal{P}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$, where $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\text{env}_I^{\mathcal{P}})$. The proof is organized by cases. First we make the following two definitions.

$$\text{env}_i^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\text{env}_I^{\mathcal{M}}) \quad (4.119)$$

$$\text{env}_i^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\text{env}_I^{\mathcal{P}}) \quad (4.120)$$

We will be defining other environments as well. Figure 77 shows the positions of the environments defined for $\mathcal{P}$, and figure 78 shows the positions of the environments defined for $\mathcal{M}$.

Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. Then $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.

Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{NL}$. We may, for this case, write $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_j, \mathrm{cs}_l, \mathrm{cs}_n \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$, where $\mathrm{ns}(i) = \mathrm{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_j, \mathrm{cs}_l, \mathrm{cs}_n \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. As shown in Figure 78, we let $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{alter all stow}(j) \ \ldots \ \textbf{whenever}\ \mathrm{Br\_Cd}\ \textbf{do}\ \textit{cd\_suffix}\ \textbf{end whenever})(\mathrm{env}_i^{\mathcal{M}})$. Similarly, as shown in Figure 77, we let $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{whenever}\ \mathrm{Br\_Cd}\ \textbf{do}\ \ldots\ \textbf{end whenever})(\mathrm{env}_i^{\mathcal{P}})$. In this case, we take $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I^{\mathcal{P}}, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_I^{\mathcal{P}}(h) = \begin{cases} \mathrm{ns}_I(h) & \text{if } h \notin \{j, l, n\} \\ \mathrm{cs}_j & \text{if } h = j \\ \mathrm{cs}_l & \text{if } h = l \\ \mathrm{cs}_n & \text{if } h = n \end{cases} . \tag{4.121}$$

Then $\mathrm{env}_i^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}_i^{\mathcal{P}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_i^{\mathcal{P}}(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \notin \{j, l, n\} \\ \mathrm{cs}_j & \text{if } h = j \\ \mathrm{cs}_l & \text{if } h = l \\ \mathrm{cs}_n & \text{if } h = n \end{cases} . \tag{4.122}$$

Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. (We are still inside case $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) \neq \mathrm{CF}$, i.e., $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{NL}$.) Hence (by Lemma 4.22), $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$. Therefore, by the semantics of the **whenever** statement, $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}} = \mathrm{env}_i^{\mathcal{P}}$. Similarly, by Lemma 4.22 and the semantics of the **whenever** statement,

$$\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}} = \mathcal{I}\left(\begin{array}{l} \textbf{alter all stow}(j) \\ \textbf{alter all stow}(l) \\ \textbf{alter all stow}(n) \end{array}\right)(\mathrm{env}_i^{\mathcal{M}}). \tag{4.123}$$

Therefore, $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_n, \mathrm{ns}_i^{\mathcal{M}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where $\mathrm{ns}_i^{\mathcal{M}} = \mathrm{ns}_i^{\mathcal{P}}$. Because $\mathrm{AE}(\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$, *fol_top_lev_code* is not empty. The first statement of *fol_top_lev_code* is **alter all** because $\mathcal{M}$ is in phase 1 (as described in Figure 31). Because $\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}}$ differs from $\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}}$ only in the current state (of particular importance is the fact that $\mathrm{SPE}(\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}}) = \mathrm{SPE}(\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}})$), $\mathcal{I}(\textbf{alter all})(\mathrm{env}_{\mathrm{ew}}^{\mathcal{P}}) = \mathcal{I}(\textbf{alter all})(\mathrm{env}_{\mathrm{ew}}^{\mathcal{M}})$.

Hence, $\mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}^{\mathcal{P}}_{\mathrm{ew}}) = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}^{\mathcal{M}}_{\mathrm{ew}})$; i.e., $\mathrm{env}^{\mathcal{P}}_F = \mathrm{env}^{\mathcal{M}}_F$. Therefore, $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$.

Case $\mathcal{I}(\mathit{Br\_Cd})(\mathit{env}^{\mathcal{M}}_i)$. (We are still inside case $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_i) \neq \mathrm{CF}$, i.e., $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_i) = \mathrm{NL}$.) Hence (by Lemma 4.22), $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}^{\mathcal{P}}_i)$. Let $\mathrm{env}^{\mathcal{P}}_n \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{if } \mathit{b\_p\_e} \textbf{ then } \ldots \textbf{ end if stow}(n))(\mathrm{env}^{\mathcal{P}}_i)$. Therefore, $\mathrm{env}^{\mathcal{P}}_{\mathrm{ew}} = \mathcal{I}(\mathit{cd\_suffix})(\mathrm{env}^{\mathcal{P}}_n)$.

Case $\mathcal{I}(\mathrm{b\_p\_e})(\mathit{env}^{\mathcal{M}}_i)$. In each of the environments $\mathrm{env}^{\mathcal{M}}_i$, $\mathrm{env}^{\mathcal{M}}_j$, $\mathrm{env}^{\mathcal{M}}_l$, and $\mathrm{env}^{\mathcal{M}}_n$ (shown in Figure 78), we have, for $h \in \{i, j, l, n\}$, $\mathrm{ISE}(\mathrm{env}^{\mathcal{M}}_h)(i) = \mathrm{ns}(i)$. Therefore, we also have, for $h \in \{i, j, l, n\}$, $\mathcal{I}(\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}^{\mathcal{M}}_h)$. Because $\mathrm{ISE}(\mathrm{env}^{\mathcal{P}}_h)(i) = \mathrm{ns}(i)$, $\mathcal{I}(\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}^{\mathcal{P}}_i)$. According to the semantics of **if-then-else**, $\mathrm{env}^{\mathcal{P}}_n = \mathcal{I}(\textbf{stow}(j)\ \mathit{cd\_kern}_1\ \textbf{stow}(k)\ \mathit{ACseq}_1\ \textbf{stow}(n))(\mathrm{env}^{\mathcal{P}}_i)$. According to the semantics of **whenever**, $\mathrm{env}^{\mathcal{M}}_n = \mathcal{I}(\textbf{alter all stow}(j)\ \textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i\ \mathit{cd\_kern}_1\ \textbf{stow}(k)\ \mathit{ACseq}_1\ \textbf{alter all stow}(l)\ \textbf{alter all stow}(n))(\mathrm{env}^{\mathcal{M}}_i)$.

Case $\mathit{AE}(\mathit{env}^{\mathcal{M}}_n) = \mathit{CF}$. By Lemma 4.16, $\mathrm{AE}(\mathcal{I}(\textbf{alter all stow}(j)\ \textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i)(\mathrm{env}^{\mathcal{M}}_i)) \neq \mathrm{VT}$. Therefore, $\mathrm{cs}_j = \mathrm{ns}(i) = \mathrm{cs}_i$. In this case, we must have $\mathrm{AE}(\mathcal{I}(\textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i\ \mathit{cd\_kern}_1\ \textbf{stow}(k)\ \mathit{ACseq}_1)(\mathrm{env}^{\mathcal{M}}_j)) = \mathrm{CF}$. Hence, we also have $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_n) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$.

Case $\mathit{AE}(\mathit{env}^{\mathcal{M}}_n) \neq \mathit{CF}$. By Lemma 4.19, $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_n) = \mathrm{NL}$. By Lemma 4.16, $\mathrm{AE}(\mathcal{I}(\textbf{assume}\ ((\mathrm{Br\_Cd}) \wedge (\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])) \Rightarrow (\mathrm{x}_n = \mathrm{x}_k))(\mathrm{env}^{\mathcal{M}}_n)) \neq \mathrm{VT}$. Therefore, $\mathrm{CSE}(\mathrm{env}^{\mathcal{M}}_n) = \mathrm{cs}_n = \mathrm{CSE}(\mathcal{I}(\textbf{assume}\ \mathrm{x}_j = \mathrm{x}_i\ \mathit{cd\_kern}_1\ \textbf{stow}(k)\ \mathit{ACseq}_1)(\mathrm{env}^{\mathcal{M}}_j)) = \mathrm{CSE}(\mathrm{env}^{\mathcal{P}}_n)$. Hence, $\mathrm{Equal\_except\_at}(\{l\}, \mathrm{env}^{\mathcal{P}}_n, \mathrm{env}^{\mathcal{M}}_n)$. By two applications of Lemma 4.37, $\mathrm{Equal\_except\_at}(\{l\}, \mathrm{env}^{\mathcal{P}}_F, \mathrm{env}^{\mathcal{M}}_F)$. Therefore, $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$.

Case $\neg\mathcal{I}(\mathrm{b\_p\_e})(\mathit{env}^{\mathcal{M}}_i)$. In this case, we have, for $h \in \{i, j, l, n\}$, $\neg\mathcal{I}(\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}^{\mathcal{M}}_h)$. We also have $\neg\mathcal{I}(\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}^{\mathcal{P}}_i)$. According to the semantics of **if-then-else**, $\mathrm{env}^{\mathcal{P}}_n = \mathcal{I}(\textbf{stow}(l)\ \mathit{cd\_kern}_2\ \textbf{stow}(m)\ \mathit{ACseq}_2\ \textbf{stow}(n))(\mathrm{env}^{\mathcal{P}}_i)$. According to the semantics of **whenever**, $\mathrm{env}^{\mathcal{M}}_l = \mathcal{I}(\textbf{alter all stow}(j)\ \textbf{alter all stow}(l))(\mathrm{env}^{\mathcal{M}}_i)$ and $\mathrm{env}^{\mathcal{M}}_n = \mathcal{I}(\textbf{assume}\ \mathrm{x}_l = \mathrm{x}_i\ \mathit{cd\_kern}_2\ \textbf{stow}(m)\ \mathit{ACseq}_2\ \textbf{alter all stow}(n))(\mathrm{env}^{\mathcal{M}}_l)$.

Case $\mathit{AE}(\mathit{env}^{\mathcal{M}}_n) = \mathit{CF}$. By Lemma 4.16, $\mathrm{AE}(\mathcal{I}(\textbf{assume}\ \mathrm{x}_l = \mathrm{x}_i)(\mathrm{env}^{\mathcal{M}}_l)) \neq \mathrm{VT}$. Therefore, $\mathrm{cs}_l = \mathrm{ns}(i) = \mathrm{cs}_i$. In this case, we must have $\mathrm{AE}(\mathcal{I}(\textbf{assume}\ \mathrm{x}_l = \mathrm{x}_i\ \mathit{cd\_kern}_2\ \textbf{stow}(m)\ \mathit{ACseq}_2)(\mathrm{env}^{\mathcal{M}}_l)) = \mathrm{CF}$. Hence, we also have $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_n) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$.

Case $\mathit{AE}(\mathit{env}^{\mathcal{M}}_n) \neq \mathit{CF}$. By Lemma 4.19, $\mathrm{AE}(\mathrm{env}^{\mathcal{M}}_n) = \mathrm{NL}$. By Lemma 4.16, $\mathrm{AE}(\mathcal{I}(\textbf{assume}\ ((\mathrm{Br\_Cd}) \wedge \neg(\mathrm{MExp}(\mathit{b\_p\_e})[\mathrm{y} \rightsquigarrow \mathrm{y}_i])) \Rightarrow (\mathrm{x}_n = \mathrm{x}_m))(\mathrm{env}^{\mathcal{M}}_n)) \neq \mathrm{VT}$. Therefore, $\mathrm{CSE}(\mathrm{env}^{\mathcal{M}}_n) = \mathrm{cs}_n = \mathrm{CSE}(\mathcal{I}(\textbf{assume}\ \mathrm{x}_l = \mathrm{x}_i\ \mathit{cd\_kern}_2\ \textbf{stow}(m)\ \mathit{ACseq}_2)(\mathrm{env}^{\mathcal{M}}_l)) = \mathrm{CSE}(\mathrm{env}^{\mathcal{P}}_n)$. Hence, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}^{\mathcal{P}}_n, \mathrm{env}^{\mathcal{M}}_n)$. By two applications of Lemma 4.37, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}^{\mathcal{P}}_F, \mathrm{env}^{\mathcal{M}}_F)$. Therefore, $\mathrm{AE}(\mathrm{env}^{\mathcal{P}}_F) = \mathrm{CF}$. □

Lemma 4.52 *The rule for selection in the absence of an* **else** *clause preserves invalidity in the program direction.*

Proof. The proof is similar to the proof that the rule for selection in the presence of an **else** clause preserves invalidity in the program direction. The important difference is in the case in which $\neg\mathcal{I}(b\_p\_e)(\mathrm{env}_i^{\mathcal{M}})$ (within the cases $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) \neq \mathrm{CF}$ and $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$). We supply here only the argument for this case. The environments defined in the complete proof are similar to those defined for the rule for selection in the presence of an **else** clause. Figure 79 shows the positions of the environments defined for $\mathcal{P}$, and figure 80 shows the positions of the environments defined for $\mathcal{M}$. In the case that $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) \neq \mathrm{CF}$, because $\mathcal{M}$ of the rule for selection in the absence of an **else** clause introduces two—and not three—**alter all** statements, we may write $\mathrm{env}_i^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \langle \mathrm{cs}_j, \mathrm{cs}_n\rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$, where $\mathrm{ns}(i) = \mathrm{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{cs}_j, \mathrm{cs}_n\rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. In this case, we take $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I^{\mathcal{P}}, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_I^{\mathcal{P}}(h) = \begin{cases} \mathrm{ns}_I(h) & \text{if } h \notin \{j, n\} \\ \mathrm{cs}_j & \text{if } h = j \\ \mathrm{cs}_n & \text{if } h = n \end{cases} . \tag{4.124}$$

Then $\mathrm{env}_i^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}_i^{\mathcal{P}}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$ where

$$\mathrm{ns}_i^{\mathcal{P}}(h) = \begin{cases} \mathrm{ns}(h) & \text{if } h \notin \{j, n\} \\ \mathrm{cs}_j & \text{if } h = j \\ \mathrm{cs}_n & \text{if } h = n \end{cases} . \tag{4.125}$$

Case $\neg\mathcal{I}(\mathrm{b\_p\_e})(env_i^{\mathcal{M}})$. In this case, we have, for $h \in \{i, j, n\}$, $\neg\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_h^{\mathcal{M}})$. We also have $\neg\mathcal{I}(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])(\mathrm{env}_i^{\mathcal{P}})$. According to the semantics of **if-then**, $\mathrm{env}_n^{\mathcal{P}} = \mathcal{I}(\mathbf{stow}(n))(\mathrm{env}_i^{\mathcal{P}})$. According to the semantics of **whenever**, $\mathrm{env}_n^{\mathcal{M}} = \mathcal{I}(\mathbf{alter\ all\ stow}(j)\ \mathbf{alter\ all\ stow}(n))(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) \neq \mathrm{CF}$, $\mathrm{AE}(\mathrm{env}_n^{\mathcal{M}}) \neq \mathrm{CF}$. By Lemma 4.19, $\mathrm{AE}(\mathrm{env}_n^{\mathcal{M}}) = \mathrm{NL}$. By Lemma 4.16, $\mathrm{AE}(\mathcal{I}(\mathbf{assume}\ ((\mathrm{Br\_Cd}) \wedge \neg(\mathrm{MExp}(b\_p\_e)[\mathrm{y} \rightsquigarrow \mathrm{y}_i])) \Rightarrow (\mathrm{x}_n = \mathrm{x}_i))(\mathrm{env}_n^{\mathcal{M}})) \neq \mathrm{VT}$. Therefore, $\mathrm{CSE}(\mathrm{env}_n^{\mathcal{M}}) = \mathrm{ns}(i) = \mathrm{cs}_i = \mathrm{CSE}(\mathrm{env}_n^{\mathcal{P}})$. Hence, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_n^{\mathcal{P}}, \mathrm{env}_n^{\mathcal{M}})$. By two applications of Lemma 4.37, $\mathrm{Equal\_except\_at}(\{j\}, \mathrm{env}_F^{\mathcal{P}}, \mathrm{env}_F^{\mathcal{M}})$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.53 *The bridge rule preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and

$\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. We may write $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}', \mathrm{ns}, \langle \mathrm{cs} \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. We take $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. By the semantics of **whenever**,

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}\left(\begin{array}{l} \textbf{alter all} \\ \textbf{stow}(i) \\ \textbf{assume}\ \mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i) \\ \mathrm{Stows\_added}(p\_body) \\ \textbf{stow}(j) \\ \textbf{confirm}\ \mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{x}_i, \mathrm{x} \rightsquigarrow \mathrm{x}_j] \end{array}\right)(\mathrm{env}_I^{\mathcal{M}}). \tag{4.126}$$

Let

$$\mathrm{env}_j^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}\left(\begin{array}{l} \textbf{alter all} \\ \textbf{stow}(i) \\ \textbf{assume}\ \mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i) \\ \mathrm{Stows\_added}(p\_body) \\ \textbf{stow}(j) \end{array}\right)(\mathrm{env}_I^{\mathcal{M}}). \tag{4.127}$$

By Lemma 4.16, $\mathcal{I}(\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{x}_i] \wedge \textbf{is\_initial}(\mathrm{z}_i))(\mathcal{I}(\textbf{alter all}\ \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}}))$. Therefore, because $\mathrm{cs} = \mathrm{ISE}(\mathcal{I}(\textbf{alter all}\ \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}}))(i)$, $\mathcal{I}(\mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z}))(\mathcal{I}(\textbf{remember})(\mathrm{env}_I^{\mathcal{P}}))$. Note also that $\mathrm{cs} = \mathrm{CSE}(\mathcal{I}(\textbf{alter all}\ \textbf{stow}(i))(\mathrm{env}_I^{\mathcal{M}}))$.

Case $AE(env_j^{\mathcal{M}}) = CF$. Because *p_body* and $\mathrm{Stows\_added}(p\_body)$ are unaffected by old state (Lemma 4.30), *p_body* contains no indexed variables, and the interpretation of any **stow** statement changes at most the index state of an environment, $\mathrm{AE}(\mathcal{I}(\textbf{remember assume}\ \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}})) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.

Case $AE(env_j^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{NL}$. Note that $\mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(i) = \mathrm{cs}$. Because *p_body* and $\mathrm{Stows\_added}(p\_body)$ are unaffected by old state (Lemma 4.30), *p_body* contains no indexed variables, and the interpretation of any **stow** statement changes at most the index state of an environment, $\mathrm{CSE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{CSE}(\mathcal{I}(\textbf{remember assume}\ \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))$. Note also that $\mathrm{ISE}(\mathrm{env}_j^{\mathcal{M}})(j) = \mathrm{CSE}(\mathrm{env}_j^{\mathcal{M}})$. In this case, it must be that $\neg\mathcal{I}(\mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{x}_i, \mathrm{x} \rightsquigarrow \mathrm{x}_j])(\mathrm{env}_j^{\mathcal{M}})$. Because $\mathrm{OSE}(\mathcal{I}(\textbf{remember assume}\ \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))(\#\xi) = \mathrm{cs}(\xi)$, $\neg\mathcal{I}(\mathrm{post}\lfloor \#\mathrm{x}, \mathrm{x} \rfloor)(\mathcal{I}(\textbf{remember assume}\ \mathrm{pre}\lfloor \mathrm{x} \rfloor \wedge \textbf{is\_initial}(\mathrm{z})\ p\_body)(\mathrm{env}_I^{\mathcal{P}}))$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.54 *The rule for* **assume** *preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. First we make the following two definitions.

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{M}}) \quad (4.128)$$

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{P}}) \quad (4.129)$$

Consequently, $\mathrm{env}_i^{\mathcal{P}} = \mathrm{env}_i^{\mathcal{M}}$.
Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{NL}$.
Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. In this case, $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume}\ (\mathrm{Br\_Cd}) \Rightarrow (H)\ \mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$ and $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{P}})$. Because $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}((\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}})$. Therefore, $\mathcal{I}(\textbf{assume}\ (\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{env}_i^{\mathcal{M}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. In this case, $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textbf{assume}\ H\ \mathit{cd\_suffix}\ \mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume}\ (\mathrm{Br\_Cd}) \Rightarrow (H)\ \mathit{cd\_suffix}\ \mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}(\textbf{assume}\ (\mathrm{Br\_Cd}) \Rightarrow (H))(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{assume}\ H)(\mathrm{env}_i^{\mathcal{M}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{assume}\ H\ \mathit{cd\_suffix}\ \mathit{fol\_top\_lev\_code})(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.55 *The rule for* **confirm** *preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. First we make the following two definitions.

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{M}}) \quad (4.130)$$

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\mathit{ACseq}_0)(\mathrm{env}_I^{\mathcal{P}}) \quad (4.131)$$

Consequently, $\mathrm{env}_i^{\mathcal{P}} = \mathrm{env}_i^{\mathcal{M}}$. Furthermore, $\mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}})(i) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}})$.
Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{NL}$.
Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. In this case, $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm}\ (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}((\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_i^{\mathcal{M}})$. Therefore, $\mathcal{I}(\textbf{confirm}\ (\mathrm{Br\_Cd}) \Rightarrow (H)[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{env}_i^{\mathcal{M}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. In this case, $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$. By the semantics of **whenever**, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textbf{confirm}\ H\lfloor \mathrm{x} \rfloor\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm}\ (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{M}})$, $\mathcal{I}(\textbf{confirm}\ (\mathrm{Br\_Cd}) \Rightarrow (H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]))(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{confirm}\ H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textbf{confirm}\ H[\mathrm{x} \rightsquigarrow \mathrm{x}_i]\ cd\_suffix\ fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$. Because $\mathrm{ISE}(\mathrm{env}_i^{\mathcal{M}})(i) = \mathrm{CSE}(\mathrm{env}_i^{\mathcal{P}})$, $\mathcal{I}(\textbf{confirm}\ H[\mathrm{x} \rightsquigarrow \mathrm{x}_i])(\mathrm{env}_i^{\mathcal{M}}) = \mathcal{I}(\textbf{confirm}\ H\lfloor \mathrm{x} \rfloor)(\mathrm{env}_i^{\mathcal{P}})$. By two applications of equation 2.23, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.56 *The rule for empty guarded blocks preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. We take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$ and make the following two definitions.

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{M}}) \quad (4.132)$$

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(prec\_top\_lev\_code\ \textbf{alter all stow}(i)ACseq_0)(\mathrm{env}_I^{\mathcal{P}}) \quad (4.133)$$

Consequently, $\mathrm{env}_i^{\mathcal{P}} = \mathrm{env}_i^{\mathcal{M}}$.
Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, $\mathrm{AE}(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_i^{\mathcal{M}}) = \mathrm{NL}$. Whether $\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$ or $\neg\mathcal{I}(\mathrm{Br\_Cd})(\mathrm{env}_i^{\mathcal{P}})$, $\mathcal{I}(\textbf{whenever}\ \mathrm{Br\_Cd}\ \textbf{do end whenever})(\mathrm{env}_i^{\mathcal{P}}) = \mathcal{I}(\varepsilon)(\mathrm{env}_i^{\mathcal{P}}) = \mathrm{env}_i^{\mathcal{P}}$. By equation 2.23, $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{P}})$. Because $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_i^{\mathcal{M}})$, $\mathrm{env}_F^{\mathcal{M}} = \mathrm{env}_F^{\mathcal{P}}$. Therefore, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.57 *The rule for* **alter all** *preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. First we make the following four definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.134}$$

$$\mathrm{env}_a^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{alter all})(\mathrm{env}_1^{\mathcal{P}}) \tag{4.135}$$

$$\mathrm{env}_s^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{stow}(i))(\mathrm{env}_a^{\mathcal{P}}) \tag{4.136}$$

$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.137}$$

Consequently,

$$\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_s^{\mathcal{P}}) \tag{4.138}$$

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_1^{\mathcal{M}}). \tag{4.139}$$

Case $AE(env_1^{\mathcal{M}}) = CF$. In this case, we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. Hence, $\mathrm{env}_1^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{M}}$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. We may, for this case, write $\mathrm{env}_1^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_1, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. With the help of Lemmas 4.20 and 4.27, we may also write $\mathrm{env}_I^{\mathcal{M}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. In this case, we take $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \langle \mathrm{ns}(i) \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_1^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_1, \mathrm{ns}, \langle \mathrm{ns}(i) \rangle \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. By the semantics of **alter all**, $\mathrm{env}_a^{\mathcal{P}} = [\mathrm{NL}, \mathrm{ns}(i), \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. By the semantics of **stow**(i), $\mathrm{env}_s^{\mathcal{P}} = \mathrm{env}_a^{\mathcal{P}}$. Therefore, $\mathrm{env}_s^{\mathcal{P}}$ differs from $\mathrm{env}_1^{\mathcal{M}}$ only in the current state. Due to the additional syntactic restriction of the rule for **alter all**, *fol_top_lev_code* contains only **alter all**, **stow**, **assume**, and **confirm** statements. The **assume** and **confirm** statements refer only to indexed variables. Hence, **stow** is the only one of these statements whose semantics is affected by the current state; however, every one of these **stow** statements is immediately preceded by an **alter all** statement. Therefore, $\mathrm{CSE}(\mathrm{env}_1^{\mathcal{M}})$ and $\mathrm{CSE}(\mathrm{env}_s^{\mathcal{P}})$ have no effect on the values, respectively, of $\mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_1^{\mathcal{M}})$ and $\mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_s^{\mathcal{P}})$. Therefore, $\mathrm{env}_F^{\mathcal{M}}$ and $\mathrm{env}_F^{\mathcal{P}}$ differ at most in the current state. In particular, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.58 *The rule for consecutive* **assume** *statements preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and

$\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. First we make the following five definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.140}$$

$$\mathrm{env}_2^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_1)(\mathrm{env}_1^{\mathcal{P}}) \tag{4.141}$$

$$\mathrm{env}_b^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_2)(\mathrm{env}_2^{\mathcal{P}}) \tag{4.142}$$

$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{prec\_top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.143}$$

$$\mathrm{env}_b^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } (H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}}) \tag{4.144}$$

Consequently,

$$\mathrm{env}_1^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{M}} \tag{4.145}$$

$$\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{P}}) \tag{4.146}$$

$$\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathit{fol\_top\_lev\_code})(\mathrm{env}_b^{\mathcal{M}}). \tag{4.147}$$

Case $AE(env_1^{\mathcal{M}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. By Lemma 4.18 and the semantics of the **assume** statement, $\mathrm{AE}(\mathrm{env}_b^{\mathcal{M}}) = \mathrm{NL}$. Therefore, $\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$ and $\mathrm{env}_b^{\mathcal{M}} = \mathrm{env}_1^{\mathcal{M}}$. Hence, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$ and $\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. Therefore, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. Hence, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Therefore, $\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. Hence, $\mathrm{env}_b^{\mathcal{P}} = \mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Therefore, $\mathrm{env}_b^{\mathcal{P}} = \mathrm{env}_b^{\mathcal{M}}$. Hence, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.59 *The* **assume-confirm** *rule preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. First we make the following three definitions.

$$\mathrm{env}_1^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{top\_lev\_code})(\mathrm{env}_I^{\mathcal{P}}) \tag{4.148}$$

$$\mathrm{env}_2^{\mathcal{P}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\textbf{assume } H_1)(\mathrm{env}_1^{\mathcal{P}}) \tag{4.149}$$

$$\mathrm{env}_1^{\mathcal{M}} \stackrel{\mathrm{def}}{=} \mathcal{I}(\mathit{top\_lev\_code})(\mathrm{env}_I^{\mathcal{M}}) \tag{4.150}$$

Consequently,

$$\begin{aligned} \mathrm{env}_1^{\mathcal{M}} &= \mathrm{env}_1^{\mathcal{P}} && (4.151)\\ \mathrm{env}_F^{\mathcal{P}} &= \mathcal{I}(\textbf{confirm } H_2)(\mathrm{env}_2^{\mathcal{P}}) && (4.152)\\ \mathrm{env}_F^{\mathcal{M}} &= \mathcal{I}(\textbf{confirm } (H_1) \Rightarrow (H_2))(\mathrm{env}_1^{\mathcal{M}}). && (4.153) \end{aligned}$$

Case $AE(env_1^{\mathcal{M}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_1^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$. By the semantics of the **confirm** statement, $\neg\mathcal{I}((H_1) \Rightarrow (H_2))(\mathrm{env}_1^{\mathcal{M}})$. By the definition of implication, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$ and $\neg\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. By substitution of equals, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. By the semantics of the **assume** statement, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. By substitution of equals, $\neg\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. By the semantics of the **confirm** statement, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.60 *The rule for consecutive* **confirm** *statements preserves invalidity in the program direction.*

Proof. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. In each case we take $\mathrm{env}_I^{\mathcal{P}} = \mathrm{env}_I^{\mathcal{M}}$. First we make the following five definitions.

$$\begin{aligned} \mathrm{env}_1^{\mathcal{P}} &\stackrel{\text{def}}{=} \mathcal{I}(prec\_top\_lev\_code)(\mathrm{env}_I^{\mathcal{P}}) && (4.154)\\ \mathrm{env}_2^{\mathcal{P}} &\stackrel{\text{def}}{=} \mathcal{I}(\textbf{confirm } H_1)(\mathrm{env}_1^{\mathcal{P}}) && (4.155)\\ \mathrm{env}_b^{\mathcal{P}} &\stackrel{\text{def}}{=} \mathcal{I}(\textbf{confirm } H_2)(\mathrm{env}_2^{\mathcal{P}}) && (4.156)\\ \mathrm{env}_1^{\mathcal{M}} &\stackrel{\text{def}}{=} \mathcal{I}(prec\_top\_lev\_code)(\mathrm{env}_I^{\mathcal{M}}) && (4.157)\\ \mathrm{env}_b^{\mathcal{M}} &\stackrel{\text{def}}{=} \mathcal{I}(\textbf{confirm } (H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}}) && (4.158) \end{aligned}$$

Consequently,

$$\begin{aligned} \mathrm{env}_1^{\mathcal{M}} &= \mathrm{env}_1^{\mathcal{P}} && (4.159)\\ \mathrm{env}_F^{\mathcal{P}} &= \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_b^{\mathcal{P}}) && (4.160)\\ \mathrm{env}_F^{\mathcal{M}} &= \mathcal{I}(fol\_top\_lev\_code)(\mathrm{env}_b^{\mathcal{M}}). && (4.161) \end{aligned}$$

Case $AE(env_1^{\mathcal{M}}) = CF$. Therefore, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.

Case $AE(env_1^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_1^{\mathcal{M}}) = \mathrm{NL}$. By substitution of equals, $\mathrm{AE}(\mathrm{env}_1^{\mathcal{P}}) = \mathrm{NL}$.
Case $AE(env_b^{\mathcal{M}}) = CF$. By the semantics of the **confirm** statement, $\neg\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$.
Case $\neg\mathcal{I}(H_1)(env_1^{\mathcal{M}})$. In this case, $\mathrm{AE}(\mathrm{env}_2^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $\mathcal{I}(H_1)(env_1^{\mathcal{M}})$. By substitution of equals, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. By the semantics of the **confirm** statement, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. Because $\neg\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$, $\neg\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. By substitution of equals, $\neg\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. Therefore, $\mathrm{AE}(\mathrm{env}_b^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case $AE(env_b^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_b^{\mathcal{M}}) = \mathrm{NL}$. Therefore, $\mathcal{I}((H_1) \wedge (H_2))(\mathrm{env}_1^{\mathcal{M}})$. Hence, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{M}})$ and $\mathcal{I}(H_2)(\mathrm{env}_1^{\mathcal{M}})$. Therefore, $\mathcal{I}(H_1)(\mathrm{env}_1^{\mathcal{P}})$. Hence, $\mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}}$. By substitution of equals, $\mathcal{I}(H_2)(\mathrm{env}_2^{\mathcal{P}})$. Therefore, $\mathrm{env}_b^{\mathcal{P}} = \mathrm{env}_2^{\mathcal{P}} = \mathrm{env}_1^{\mathcal{P}} = \mathrm{env}_b^{\mathcal{M}}$. By substitution of equals, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Hence, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

Lemma 4.61 *The rule of inference bridging predicate logic and the indexed method preserves invalidity in the program direction.*

Proof. We suppose that H is invalid (i.e., false) in some model of context C's logic. That is to say, there is an assignment of values to the variables in H that renders H false in that model. Let env be a neutral environment whose component states make the same assignment of values to the variables in H. Then $\mathrm{AE}(\textbf{confirm } H) = \mathrm{CF}$. Therefore, **confirm** H is an invalid program. □

4.5.4 The loop while Rule

We provide here a proof that an application of the **loop while** rule that involves a loop whose tightest loop invariant is expressed in the **maintaining** clause preserves invalidity in the program direction.
Proof of Lemma 4.7. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\mathrm{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\mathrm{env}_I^{\mathcal{M}})$; i.e., $\mathrm{AE}(\mathrm{env}_I^{\mathcal{M}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{M}}) = \mathrm{CF}$. We will show the existence of an environment, $\mathrm{env}_I^{\mathcal{P}}$, such that $\mathrm{AE}(\mathrm{env}_I^{\mathcal{P}}) = \mathrm{NL}$ and $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$, where $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\mathrm{env}_I^{\mathcal{P}})$. The proof is organized by cases. First we make the following two definitions.

$$\mathrm{env}_i^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code } \textbf{alter all stow}(i)\, ACseq_0)(\mathrm{env}_I^{\mathcal{M}}) \quad (4.162)$$

$$\mathrm{env}_i^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code } \textbf{alter all stow}(i)\, ACseq_0)(\mathrm{env}_I^{\mathcal{P}}) \quad (4.163)$$

We will be defining other environments as well. Figure 81 shows the positions of the environments defined for $\mathcal{P}$, and figure 82 shows the positions of the environments defined for $\mathcal{M}$.

Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, we take $\text{env}_I^{\mathcal{P}} = \text{env}_I^{\mathcal{M}}$. Then $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$.

Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{NL}$. We may, for this case, write $\text{env}_i^{\mathcal{M}} = [\text{NL}, \text{cs}_i, \text{ns}, \langle \text{cs}_j, \text{cs}_l \rangle \circ \text{se}, \text{os}, \text{d}]$, where $\text{ns}(i) = \text{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\text{env}_I^{\mathcal{M}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \langle \text{cs}_j, \text{cs}_l \rangle \circ \text{se}, \text{os}, \text{d}]$. As shown in Figure 82, we let $\text{env}_{\text{ew}}^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{confirm}\ (\text{Br\_Cd}) \Rightarrow (\text{Inv}[\text{x} \rightsquigarrow \text{x}_i, \#\text{x} \rightsquigarrow \text{x}_i])\ \textbf{alter all stow}(j)\ \ldots\ \textbf{whenever}\ \text{Br\_Cd}\ \textbf{do assume}\ (\neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_l])) \wedge (\text{Inv}[\text{x} \rightsquigarrow \text{x}_l, \#\text{x} \rightsquigarrow \text{x}_i])\ cd\_suffix\ \textbf{end whenever})(\text{env}_i^{\mathcal{M}})$. Similarly, as shown in Figure 81, we let $\text{env}_{\text{ew}}^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textbf{whenever}\ \text{Br\_Cd}\ \textbf{do}\ \ldots\ \textbf{end whenever})(\text{env}_i^{\mathcal{P}})$.

Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. This case is similar to the corresponding case in the proof of Lemma 4.51, and we do not repeat it here.

Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. (We are still inside case $\text{AE}(\text{env}_i^{\mathcal{M}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{NL}$.) We take $\text{env}_I^{\mathcal{P}} = \text{env}_I^{\mathcal{M}}$. Therefore, $\text{env}_i^{\mathcal{P}} = \text{env}_i^{\mathcal{M}}$.

Case $AE(env_j^{\mathcal{M}}) = CF$. In this case, we must have $\neg\mathcal{I}(\text{Inv}[\text{x} \rightsquigarrow \text{x}_i, \#\text{x} \rightsquigarrow \text{x}_i]))(\text{env}_i^{\mathcal{M}})$. By the semantics of **whenever**, the loop is interpreted in environment $\text{env}_i^{\mathcal{P}}$. Because $\neg\mathcal{I}(\text{Inv}\lfloor \text{x}, \#\text{x} \rfloor)(\text{Rem}(\text{env}_i^{\mathcal{P}}))$, the interpretation of the loop in environment $\text{env}_i^{\mathcal{P}}$ is a categorically false environment. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$.

Case $AE(env_j^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_j^{\mathcal{M}}) = \text{NL}$.

Case $AE(env_l^{\mathcal{M}}) = CF$. The following statements are true in this case. $\mathcal{I}(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_i])(\text{env}_j^{\mathcal{M}})$. $\mathcal{I}((\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_j]) \wedge (\text{Inv}[\text{x} \rightsquigarrow \text{x}_j, \#\text{x} \rightsquigarrow \text{x}_i]))(\text{env}_j^{\mathcal{M}})$. $\text{AE}(\mathcal{I}(cd\_kern\ \textbf{stow}(k)\ ACseq\ \textbf{confirm}\ \text{Inv}[\text{x} \rightsquigarrow \text{x}_k, \#\text{x} \rightsquigarrow \text{x}_i])(\text{env}_j^{\mathcal{M}})) = \text{CF}$. Because Inv is the tightest loop invariant for the loop, $\mathcal{I}(\text{Inv}[\text{x} \rightsquigarrow \text{x}_j, \#\text{x} \rightsquigarrow \text{x}_i])(\text{env}_j^{\mathcal{M}})$, and the loop and its body are unaffected by old state (Lemma 4.30), cs_j must be the current state resulting from zero or more iterations of the loop when it is interpreted in environment $\text{env}_i^{\mathcal{P}}$ [4, p. 87]. Hence, execution of the loop's body beginning in environment $\text{env}_i^{\mathcal{P}}$ produces either a categorically false environment or a neutral environment in which the loop invariant is interpreted to be false. Therefore, $\text{AE}(\text{env}_l^{\mathcal{P}}) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$.

Case $AE(env_l^{\mathcal{M}}) \neq CF$. (We are still inside case $\text{AE}(\text{env}_j^{\mathcal{M}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_j^{\mathcal{M}}) = \text{NL}$.) In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_l^{\mathcal{M}}) = \text{NL}$. By Lemma 4.18, $\mathcal{I}((\neg(\text{MExp}(b\_p\_e)[\text{y} \rightsquigarrow \text{y}_l])) \wedge (\text{Inv}[\text{x} \rightsquigarrow \text{x}_l, \#\text{x} \rightsquigarrow \text{x}_i]))(\text{env}_l^{\mathcal{M}})$. Because Inv is the tightest loop invariant for the loop and the loop and its body are unaffected by old state (Lemma 4.30), cs_l is the current state of the first environment resulting from zero or more iterations of the loop when it is interpreted in environment

$\text{env}_i^{\mathcal{P}}$ in which the Boolean program expression *b_p_e* is interpreted to be false [4, p. 87]. In other words, cs_l must be the current state of environment $\text{env}_l^{\mathcal{P}}$. Let $A \stackrel{\text{def}}{=} \{h | j \leq h \leq k\}$. Therefore, Equal_except_at$(A, \text{env}_l^{\mathcal{P}}, \text{env}_l^{\mathcal{M}})$. Note that $\text{env}_F^{\mathcal{M}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_l^{\mathcal{M}})$ and $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\text{env}_l^{\mathcal{P}})$. Two applications of Lemma 4.37 give us Equal_except_at$(A, \text{env}_F^{\mathcal{P}}, \text{env}_F^{\mathcal{M}})$. Therefore, $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$. □

4.5.5 The Procedure Call Rule

We provide here a proof that an application of the procedure call rule that involves a call either to (1) a functionally specified external procedure or (2) to an internal procedure, whose postcondition is the post relation corresponding to the procedure's precondition and its body, preserves invalidity in the program direction.

Proof of Lemma 4.6. We assume that $\mathcal{M}$ is invalid and show that $\mathcal{P}$ is invalid. Let $\text{env}_I^{\mathcal{M}}$ be a witness to $\mathcal{M}$'s invalidity, and let $\text{env}_F^{\mathcal{M}} = \mathcal{I}(\mathcal{M})(\text{env}_I^{\mathcal{M}})$; i.e., $\text{AE}(\text{env}_I^{\mathcal{M}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{M}}) = \text{CF}$. We will show the existence of an environment, $\text{env}_I^{\mathcal{P}}$, such that $\text{AE}(\text{env}_I^{\mathcal{P}}) = \text{NL}$ and $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$, where $\text{env}_F^{\mathcal{P}} = \mathcal{I}(\mathcal{P})(\text{env}_I^{\mathcal{P}})$. The proof is organized by cases. First we make the following two definitions.

$$\text{env}_i^{\mathcal{M}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\,ACseq_0)(\text{env}_I^{\mathcal{M}}) \quad (4.164)$$

$$\text{env}_i^{\mathcal{P}} \stackrel{\text{def}}{=} \mathcal{I}(\textit{prec\_top\_lev\_code}\ \textbf{alter all stow}(i)\,ACseq_0)(\text{env}_I^{\mathcal{P}}) \quad (4.165)$$

We will be defining other environments as well. Figure 83 shows the positions of the environments defined for $\mathcal{P}$ and for $\mathcal{M}$.

Case $AE(env_i^{\mathcal{M}}) = CF$. In this case, we take $\text{env}_I^{\mathcal{P}} = \text{env}_I^{\mathcal{M}}$. Then $\text{AE}(\text{env}_i^{\mathcal{P}}) = \text{CF}$. By Lemma 4.15, $\text{AE}(\text{env}_F^{\mathcal{P}}) = \text{CF}$.

Case $AE(env_i^{\mathcal{M}}) \neq CF$. In this case, by Lemma 4.19, we have $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{NL}$. We may, for this case, write $\text{env}_i^{\mathcal{M}} = [\text{NL}, \text{cs}_i, \text{ns}, \langle \text{cs}_j \rangle \circ \text{se}, \text{os}, \text{d}]$, where $\text{ns}(i) = \text{cs}_i$. With the help of Lemmas 4.20 and 4.27, we may also write $\text{env}_I^{\mathcal{M}} = [\text{NL}, \text{cs}_I, \text{ns}_I, \text{se}_I \circ \langle \text{cs}_j \rangle \circ \text{se}, \text{os}, \text{d}]$.

Case $\neg\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. This case is similar to the corresponding case in the proof of Lemma 4.51, and we do not repeat it here.

Case $\mathcal{I}(Br\_Cd)(env_i^{\mathcal{M}})$. (We are still inside case $\text{AE}(\text{env}_i^{\mathcal{M}}) \neq \text{CF}$, i.e., $\text{AE}(\text{env}_i^{\mathcal{M}}) = \text{NL}$.) By definition of the applicability of the procedure call rule, the declaration-meaning d of environment $\text{env}_I^{\mathcal{M}}$ contains d(P_nm). Let $\text{d(P\_nm)} \stackrel{\text{def}}{=} [\text{dp}, \text{ep}, \text{pf}, \text{sf}]$. Because $\text{AE}(\text{env}_I^{\mathcal{M}}) = \text{NL}$, we have that dp is the same as pre, ep is the same as post, and d(P_nm) is conformal.

Case $AE(env_j^{\mathcal{M}}) = CF$. We take $\text{env}_I^{\mathcal{P}} = \text{env}_I^{\mathcal{M}}$. Therefore, $\text{env}_i^{\mathcal{P}} = \text{env}_i^{\mathcal{M}}$. In this case, $\neg\mathcal{I}(\text{pre}[\text{x} \rightsquigarrow \text{ac}_i, \text{y} \rightsquigarrow \text{ad}_i, \text{z} \rightsquigarrow \text{z}_i])(\text{env}_i^{\mathcal{M}})$. Therefore, $\neg\text{dp}(\text{cs}_i(\text{ac}), \text{cs}_i(\text{ad}), \text{cs}_i(\text{z}))$.

Hence, $\mathrm{AE}(\mathcal{I}(\mathrm{P\_nm}(\mathrm{ac},\mathrm{ad}))(\mathrm{env}_i^{\mathcal{P}})) = \mathrm{CF}$. Therefore, $\mathrm{AE}(\mathrm{env}_j^{\mathcal{P}}) = \mathrm{CF}$. By Lemma 4.15, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$.
Case AE($env_j^{\mathcal{M}}$) $\neq$ CF. In this case, by Lemma 4.19, we have $\mathrm{AE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{NL}$. We take $\mathrm{env}_I^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_I, \mathrm{ns}_I, \mathrm{se}_I \circ \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Then $\mathrm{env}_i^{\mathcal{P}} = [\mathrm{NL}, \mathrm{cs}_i, \mathrm{ns}, \mathrm{se}, \mathrm{os}, \mathrm{d}]$. Because $\mathrm{AE}(\mathrm{env}_j^{\mathcal{M}}) = \mathrm{NL}$, $\mathcal{I}(\mathrm{pre}[\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_i])(\mathrm{env}_i^{\mathcal{M}})$. Hence, $\mathrm{dp}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}))$. By Lemma 4.16, $\mathcal{I}(\mathrm{b}_j = \mathrm{b}_i \wedge (\mathrm{post}[\#\mathrm{x} \rightsquigarrow \mathrm{ac}_i, \mathrm{x} \rightsquigarrow \mathrm{ac}_j, \#\mathrm{y} \rightsquigarrow \mathrm{ad}_i, \mathrm{y} \rightsquigarrow \mathrm{ad}_j, \#\mathrm{z} \rightsquigarrow \mathrm{z}_i, \mathrm{z} \rightsquigarrow \mathrm{z}_j]))(\mathrm{env}_j^{\mathcal{M}})$. Hence, $\mathrm{ep}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}), \mathrm{cs}_j(\mathrm{ac}), \mathrm{cs}_j(\mathrm{ad}), \mathrm{cs}_j(\mathrm{z}))$. Let $[w_1, w_2, w_3] \stackrel{\mathrm{def}}{=} \mathrm{pf}(\mathrm{cs}_i(\mathrm{ac}), \mathrm{cs}_i(\mathrm{ad}), \mathrm{cs}_i(\mathrm{z}))$. Because P_nm is either (1) a functionally specified external procedure or (2) to an internal procedure, whose postcondition is the post relation corresponding to the procedure's precondition and its body, $[w_1, w_2, w_3] = [\mathrm{cs}_j(\mathrm{ac}), \mathrm{cs}_j(\mathrm{ad}), \mathrm{cs}_j(\mathrm{z})]$. Therefore, $\mathrm{CSE}(\mathrm{env}_j^{\mathcal{P}}) = \mathrm{cs}_j$, and, furthermore, $\mathrm{env}_j^{\mathcal{P}} = \mathrm{env}_j^{\mathcal{M}}$. Because $\mathrm{env}_F^{\mathcal{P}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\mathrm{env}_j^{\mathcal{P}})$ and $\mathrm{env}_F^{\mathcal{M}} = \mathcal{I}(\textit{cd\_suffix fol\_top\_lev\_code})(\mathrm{env}_j^{\mathcal{M}})$, $\mathrm{env}_F^{\mathcal{P}} = \mathrm{env}_F^{\mathcal{M}}$. Hence, $\mathrm{AE}(\mathrm{env}_F^{\mathcal{P}}) = \mathrm{CF}$. □

4.6 Relaxing a Simplifying Assumption

We made a simplifying assumption in formulating the proof rules of Chapter III. The presentation of the rules is shorter than it otherwise would have been because we assumed that only the first statement within a **whenever** statement would be replaced by application of a proof rule in the math direction. However, the more flexible method informally described in Chapter I would be better represented by rules that permitted any statement within a **whenever** statement to be replaced. We argue here that, if we relax this simplifying assumption by adding rules that permit any statement within a **whenever** statement to be replaced, all the lemmas and theorems of this chapter would still hold true. That is to say, we made our simplifying assumption without loss of generality.

Figure 84 shows an additional rule for procedure call that permits any call within a **whenever** statement to be replaced—as long as that call is not the first statement within a **whenever** statement. One would use the existing procedure call rule from Chapter III to replace a call that is the first statement within a **whenever** statement. We can add three other rules to our proof system to handle operational statements that are not first within a **whenever** statement—one for the **loop while** statement and two for the two forms of the selection statement.

The proofs that invalidity is preserved in the two directions for these four new rules will contain the same arguments as the corresponding proofs for the existing "first-statement" rules. The new proofs will also use the following two additional

$$\mathcal{P} \stackrel{\text{def}}{=} C\backslash \quad \begin{array}{l} \textit{prec\_top\_lev\_code} \\ \textit{stow\_sec} \\ \textit{ACseq}_0 \\ \mid\ \textbf{whenever}\ \text{Br\_Cd}\ \textbf{do} \\ \quad \textit{cd\_kern} \\ \quad \textbf{stow}(i) \\ \quad \textit{ACseq}_1 \\ \quad \text{P\_nm}(\text{ac}, \text{ad}) \\ \quad \textbf{stow}(j) \\ \quad \textit{cd\_suffix} \\ \mid\ \textbf{end whenever} \\ \textit{fol\_top\_lev\_code} \end{array} \tag{4.166}$$

$$\mathcal{M} \stackrel{\text{def}}{=} C\backslash \quad \begin{array}{l} \textit{prec\_top\_lev\_code} \\ \textit{stow\_sec} \\ \textit{ACseq}_0 \\ \mid\ \textbf{whenever}\ \text{Br\_Cd}\ \textbf{do} \\ \quad \textit{cd\_kern} \\ \quad \textbf{stow}(i) \\ \quad \textit{ACseq}_1 \\ \mid\ \textbf{end whenever} \\ \mid\ \textbf{confirm}\ (\text{Br\_Cd}) \Rightarrow (\text{pre}[\text{x} \rightsquigarrow \text{ac}_i, \text{y} \rightsquigarrow \text{ad}_i, \text{z} \rightsquigarrow \text{z}_i]) \\ \mid\ \textbf{alter all} \\ \mid\ \textbf{stow}(j) \\ \mid\ \textbf{assume}\ (\text{Br\_Cd}) \Rightarrow (\text{b}_j = \text{b}_i \wedge (\text{post}[\#\text{x} \rightsquigarrow \text{ac}_i, \text{x} \rightsquigarrow \text{ac}_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \#\text{y} \rightsquigarrow \text{ad}_i, \text{y} \rightsquigarrow \text{ad}_j, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \#\text{z} \rightsquigarrow \text{z}_i, \text{z} \rightsquigarrow \text{z}_j])) \\ \mid\ \textbf{whenever}\ \text{Br\_Cd}\ \textbf{do} \\ \quad \textit{cd\_suffix} \\ \mid\ \textbf{end whenever} \\ \textit{fol\_top\_lev\_code} \end{array} \tag{4.167}$$

Additional Syntactic Restriction: The same as in Figure 46.

Figure 84: The Rule for a Procedure Call That Is Not the First Statement Within a **whenever** Statement

observations. In the case that the branch condition Br_Cd evaluates as false, interpretation of the first **whenever** statement in $\mathcal{M}$ is the same as interpretation of the empty sequence of statements. In the case that the branch condition Br_Cd evaluates as true, interpretation of the first **whenever** statement in $\mathcal{M}$ is the same as interpretation of *cd_kern* **stow**(i) $ACseq_1$.

We can still prove all the other lemmas and theorems for this extended proof system. Three of the lemmas deserve special mention here. In the example of the additional proof rule for procedure call (Figure 84), all variables subscripted with i introduced by this rule appear after the **whenever** statement containing the **stow**(i) statement; importantly, they also appear only in **confirm** and **assume** statements in the consequents of assertions whose antecedents are the branch condition (Br_Cd) of the **whenever** statement. This fact means that we can prove Lemma 4.35 and, consequently, the negative-branch-condition independence lemma (4.36). Because the additional rules introduce an extra **whenever** statement, the multipliers for function Meas (Table 2) used in the proof of Lemma 4.5 in Section 4.5.2 need to be changed. The multipliers for procedure call, **loop**, and **if** need to be increased by one to six, twelve, and thirteen, respectively. Because we can still prove all lemmas and theorems for this extended proof system, our simplifying assumption is without loss of generality. The indexed method can process operational statements in any order.

4.7 Summary

In this chapter we have supplied the detailed proofs for Theorems 1 and 2. These two theorems validate the second and third points of our three-part thesis. They establish that the indexed method for reasoning about program behavior is both sound and relatively complete.