Maximal and Compositional Pattern-Based Loop Invariants
Definitions and Proofs

Virginia Aponte\textsuperscript{1}, Pierre Courtieu\textsuperscript{1}, and Yannick Moy\textsuperscript{2}

\textsuperscript{1} CNAM, 292 rue Saint-Martin F-75141 Paris Cedex 03 - FRANCE \{maria-virginia.aponte.garcia,pierre.courtieu\}@cnam.fr
\textsuperscript{2} AdaCore, 46 rue d’Amsterdam, F-75009 Paris (France) \{moy\}@adacore.com

1 Introduction

We present a novel approach for the automatic generation of inductive loop invariants over non nested loops manipulating arrays. It is based in the detection of simple but frequent code patterns within loops and on the instantiation of corresponding invariants, which have been previously proved as correct. In this technical report we give definitions and some of the proofs associated with the theoretical framework of this approach.

2 A Language of Parallel Assignments

In this section we introduce the intermediate language \(\mathcal{L}\) and its formal semantics. \(\mathcal{L}\) is a refinement of the language introduced in [3] that groups in a single syntactic unit all the assignments performed on the same location.

2.1 Syntax

Fig. 1 presents the intermediate language \(\mathcal{L}\). In this language, programs are restricted to a non nested for-like loop (possibly having an extra exit condition) over scalar and one-dimensional array variables. Assignments in \(\mathcal{L}\) are performed in parallel.

Note that location expressions \((e_l)\) can be either scalar variables or array cells and that all statements \((s_i)\) of a group \((\mathcal{G})\) assign to the same variable: either the group (only) contains guarded statements \(g_k \rightarrow x := e_k\) assigning to some scalar variable \(x\); or it contains statements \(g_p \rightarrow A[a_p] := e_p\) assigning to the possibly different cells \(A[a_1], A[a_2] \ldots\) of some array variable \(A\). A loop body \((B)\) is an unordered collection of groups for different variables.

Expressions and variables \(n, k\) stand for (non negative) constants of the language; lower case letters \(x, a\) are scalar variables; upper-case letters \(A, C\) are array variables; \(v\) is any variable; \(e_a\) is an arithmetic expression; \(e, e_b, g\) are Boolean expressions; \(e\) is any expression. Subscripted variables \(x_0\) and \(A_0\) denote respectively the initial (abstract) value of variables \(x\) and \(A\).
Informal semantics Groups are executed simultaneously: expressions and guards are evaluated before assignments are executed. We assume groups and bodies to be write-disjoint, and loops to be well-formed. A group $G$ is write-disjoint if all its assignments update the same variable, and if for any two different guards $g_1, g_2$ in $G$, $g_1 \land g_2$ is unsatisfiable. A loop body $B = \{ G_1 \parallel \ldots \parallel G_n \}$ is write-disjoint if all $G_k$ update different variables and if they are all write-disjoint. A loop $L$ is well-formed if its body is write-disjoint. Thus, on each iteration, at most one assignment is performed for each variable. Conditions on guarded assignments are essentially the same as in the work of Kovacs and Voronkov [3], with a slightly different formalism. Note that, for simplicity, we require here unsatisfiability of $g_1 \land g_2$ for two guards within a group assigning to array $A$, even in the case where the updated cells on those guards are actually different.

Loop conventions $L$ denotes a loop, $B$ a body, and $i$ is always the loop index. The loop index is not a variable, so it cannot be assigned. For simplicity, we assume that $i$ is increased (and not decreased) after each run through the loop, from its initial value $\alpha$ to its final value $\omega$. We use $\ell_{(\alpha,\omega,e)}(B)$ to abbreviate

\[
\text{loop } i \text{ in } \alpha..\omega \text{ exit } e \text{ do } B \text{ end}
\]

written $\ell_{(\alpha,\omega)}(B)$ when $e = \text{false}$. $G$ denotes a loop body $G_1 \parallel \ldots \parallel G_n$ (for some $n$) and $G \parallel B$ is the parallel composition of groups $G_1, \ldots, G_n$ with groups occurring in $B$. Similarly, $\{ g_k \rightarrow c_k := e_k \}$ denotes a group made of the guarded assignments $\{ g_1 \rightarrow l_1 := e_1; \ldots; g_n \rightarrow l_n := e_n \}$. $G(B)$ denotes the set of groups occurring in $B$.

Loop variables $V(L)$ is the set of variables occurring in $L$ (note that $i \notin V(L)$). $V_w(L)$ is the set of variables assigned in $L$, referred to as local (to $L$). $V_{nw}(L)$ is the set of variables occurring in $L$ but not assigned in $L$, referred to as external (to $L$): $V_{nw}(L) = V(L) - V_w(L)$. Given a set of variables $V$, the initialisation predicate $i_V$ is defined as $i_V \Leftrightarrow \bigwedge_{v \in V} v = v_0$ asserting that all variables $v \in V$ have their initial (abstract) value $v_0$. Sets and formulas defined on the loop $L$ are similarly defined on the loop body $B$.

Quantifications, substitutions and fresh variables $\phi$, $\psi$, $\iota$ and $\varphi$ denote formulas. The loop index $i$ may occur in the formula $\phi$ or in the expression $e$, re-
respectively denoted \( \phi(i) \) or \( e(i) \), but it can be omitted when not relevant. Except for logical assertions (i.e. invariants, Hoare triples), formulas are implicitly universally quantified on the set of all their free variables, including \( i \). To improve readability, these quantifications are often kept implicit. We denote by \( \exists V.\phi \) the formula \( \exists v_1 \ldots v_n.\phi \) for all \( v_i \in V \), and by \( [V_1 \leftarrow V_2] \) the substitution of each variable of the set \( V_1 \) by the corresponding variable of the set \( V_2 \). Given a set of variables \( V \), \( V' \) denotes the set containing a fresh variable \( v' \) for each variable \( v \in V \). Given an expression \( e \), we denote \( e' = e[V \leftarrow V'] \) and \( \phi' = \phi[V \leftarrow V'] \).

### 2.2 Strongest Postcondition Semantics

**A Semantic Formulation of sp**

Given a set of locations, and a set of values, we consider states \( \sigma \) defined in the usual way, that is, as a partial function mapping locations to values. We also assume given an operational semantics over programs \( C \) from \( L \) given by the relation \( <C,\sigma> \rightarrow \sigma' \). Our theoretical framework relies on the following semantic definition [4] of the Strongest Postcondition Predicate \( \text{sp} \), and on some of its properties, as given by corollary 1 and taken from [4].

**Definition 1 (Predicate \( \text{sp}(C, P) \)).** For any statement \( C \) and predicate \( P \) we define the predicate \( \text{sp}(C, P) \) as being such that:

\[
\sigma' \models \text{sp}(C, P) \iff \exists \sigma. (\langle C, \sigma \rangle \rightarrow \sigma' \land \sigma \models P)
\]

**Corollary 1 (Properties on \( \text{sp}(C, P) \)).** Given formulas \( P, Q \) and statement \( C \), the following properties hold:

\[
\begin{align*}
\text{[sp-post]} & \quad \models \text{par}\{P\} C \{\text{sp}(C, P)\} \\
\text{[sp-strg]} & \quad \models \text{par}\{P\} C \{Q\} \Rightarrow (\text{sp}(C, P) \Rightarrow Q) \\
\text{[sp-mono]} & \quad (P \Rightarrow Q) \Rightarrow (\text{sp}(C, P) \Rightarrow \text{sp}(C, Q))
\end{align*}
\]

**A Syntactic formulation of \( \text{sp} \) on \( L \) loop bodies**

We express the semantics of our intermediate language \( L \) through a formal definition of \( \text{sp} \). In Definition 2, we give a syntactic formulation of \( \text{sp} \). It is worth noticing that Definition 2 requires replacing a variable \( v \) assigned in the loop body with a fresh logical variable \( v' \), standing for the value of \( v \) prior to the assignment.

**Definition 2 (Predicate Transformer \( \text{sp} \)).** Let \( \phi \) be a formula, \( G_k \) a loop body, and \( V = V_w(G_k) \). We define \( \text{sp}(G_k, \phi) \) as:

\[
\begin{align*}
\text{sp}(\text{skip}, \phi) & = \phi \\
\text{sp}(G_k, \phi) & = \exists V'. (\phi' \land P\text{sp}(G_k, V)) \\
P\text{sp}(\{ g_k \rightarrow x := e_k \}, V) & = \bigwedge_k (g'_k \Rightarrow x = e'_k) \land \left( \bigwedge_k \neg g'_k \Rightarrow x = x' \right) \\
P\text{sp}(\{ g_k \rightarrow A[a_k] := e_k \}, V) & = \bigwedge_k (g'_k \Rightarrow A[a'_k] = e'_k) \\
& \land \forall j. \left( \bigwedge_k \neg (g'_k \land j = a'_k) \Rightarrow A[j] = A'[j] \right).
\end{align*}
\]
Corollary 2 (Renaming of External Variables in \( sp \)). Let \( L = \ell_{(\alpha, \omega, \epsilon)} \{ G \} \) be a well-formed loop. Let \( V_G = V_w(G) \) and \( V_B = V_w(B) \). Then, \( Psp(B, V_B \cup V_G) = (Psp(B, V_B)) \setminus V_G \).

Proof. By definition, \( Psp(B, V_B \cup V_G) \) results in a formula where (a) all variables occurring in \( V_B \) are replaced by \( x' \) only on read expressions within \( B \), and (b) all variables \( x \in V_G \) occurring in \( B \) are replaced by \( x' \). As \( L \) is well-formed, we know that \( V_G \cap V_B = \emptyset \) and therefore, we can separate substitutions performed on \( V_G \)'s variables from those performed on \( V_B \)'s variables. Substitutions performed by (a) can be obtained from \( Psp(B, V_B) \). From the \( Psp \) definition, its easy to see that this formula is equal to \( Psp(B, V_B \cup V_G) \) except for all variables in \( V_G \) that are renamed by their primed version. \( \Box \)

3 Reduced Loops and Local Invariants

In this section, we define reduced loops, which are smaller versions of some loop \( L \), and local loop invariants. A local invariant over a reduced loop is local when it can strengthen a preexisting inductive invariant \( \varphi_L \) over the complete loop. Our notion of locality is generic with respect to \( \varphi_L \).

3.1 (Inductive) \( \ell_L \)-Loop Invariants

To define inductive loop invariants, we rely on the classical relation \( \models_{\text{par}} \) of satisfaction under partial correctness for Hoare triples \([2, 4]\). Invariants are defined relative to a given initialisation predicate \( \ell_L \) providing initial values to loop variables. We define \( \ell_L = \ell_V \), where \( V \) is the set of all variables occurring in \( L \). An \( \ell_L \)-loop invariant is an inductive loop invariant under \( \ell_L \) initial conditions. Also, we say that \( \ell_L \) covers \( \phi \) when \( V(\phi) \subseteq V(\ell_L) \). In the following, we assume that the initialisation predicate \( \ell_L \) covers all properties stated on \( L \).

Definition 3 ((Inductive) \( \ell_L \)-Loop Invariant). Assume \( \ell_L \) covers a formula \( \phi \), \( \phi \) is an \( \ell_L \)-loop invariant on the loop \( L = \ell_{(\alpha, \omega, \epsilon)} \{ B \} \), iff

\[ (a) \ (i = \alpha \land \ell_L) \Rightarrow \phi; \ \text{and} \ \ (b) \ \models_{\text{par}} \{ \alpha \leq i \leq \omega \land \neg \epsilon \land \phi \} B; \ i := i + 1 \ \{ \phi \}. \]

Assume we want to state that some \( \psi \) is an \( \ell_L \)-loop invariant of \( \ell_{(\alpha, \omega, \epsilon)} \{ B \} \). We shall use the following lemma.

Lemma 3.1 (\( \ell_L \)-Loop Invariant Definition via \( sp \)). \( \psi \) is an \( \ell_L \)-loop invariant on loop \( L = \ell_{(\alpha, \omega, \epsilon)} \{ B \} \) iff:

\[ (a) \ \ell_L \text{ covers } \psi; \ \ (b) \ i = \alpha \land \ell_L \Rightarrow \psi(\alpha); \ \ (c) \ sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)) \Rightarrow \psi(i + 1). \]

Proof. Let us assume \( \psi \) is an \( \ell_L \)-invariant. From the \( \ell_L \)-invariant definition, condition (a) and (b) follow immediately. Moreover, we know that the Hoare triple \( \{ \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i) \} B; i := i + 1 \ \{ \psi(i) \} \) holds. As, \( i \notin V_w(B) \), we necessarily have: \( \{ \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i) \} B \ \{ \psi(i + 1) \} i := i + 1 \ \{ \psi(i) \} \), otherwise
ψ would not be an inductive invariant. Using sp-strg on the triple \(\{\alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)\} B \{\psi(i+1)\}\) we obtain \(\text{sp}(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)) \Rightarrow \psi(i+1)\) as desired.

Assume now that hypothesis (ab), (b), and (c) hold, and let \(\sigma_1\) be a state such that \(\sigma_1 \models_{\text{par}} \text{sp}(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i))\). Then, by the definition of sp we have \(\exists \sigma_0, < C, \sigma_0 > \Rightarrow \sigma_1 \land \sigma_0 \models \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)\). On the other hand, from hypothesis (c), we obtain that \(\sigma_1 \models_{\text{par}} \psi(i+1)\) holds. By the definition of Partial Correctness satisfaction, we obtain that \(\{\alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)\} B \{\psi(i+1)\}\) holds. Clearly, \(\{\psi(i+1)\}\) \(i := i + 1 \{\psi(i)\}\) holds as well, which yields \(\{\alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)\} B; i := i + 1 \{\psi(i)\}\). This, together with (a) and (b), shows that ψ is an \(\ell_L\)-invariant for \(L\).

\[
\text{3.2 Local (Reduced) Loop Invariants}
\]

A reduced loop from a loop \(L = \ell_{(\alpha, \omega, \epsilon)}\{B\}\), is a loop with the same index range as \(L\) but whose body \(B_r\) is a collection of groups occurring within \(B\) (i.e. \(G(B_r) \subseteq G(B)\)). These loops either take the form \(L_r = \ell_{(\alpha, \omega, \epsilon)}\{B_r\}\) or \(L_r = \ell_{(\alpha, \omega)}\{B_r\}\). To deduce properties holding locally on \(L_r\), we assume given an inductive loop invariant \(\varphi_L\) holding on the entire loop, that states properties over variables external to \(L_r\). Thus, we use a global pre-established property on external variables in order to deduce local properties over local variables. The notion of relative-inductive invariants, borrowed from [1], captures this style of reasoning: \(\phi\) is inductive relative to another formula \(\varphi_L\) on loop \(L\), when the inductive step of the proof of \(\phi\) holds under the assumption \(\varphi_L\).

**Definition 4 (Relative Inductive Invariant).** A property \(\phi\) is \(\varphi_L\)-inductive on loop \(L\), if

1. \(\ell_L\) covers \(\varphi_L \land \phi\) and \((i = \alpha \land \ell_L) \Rightarrow \phi\);
2. \(\text{sp}(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \varphi_L(i) \land \phi(i)) \Rightarrow \phi(i + 1)\).

\(\phi\) is a \(\varphi_L\)-local loop invariant on loop \(L_r\), if \(\phi\) only refers to variables locally modified in \(L_r\), and if \(\phi\) holds inductively on \(L_r\) relatively to the property \(\varphi_L\).

**Definition 5 (\(\varphi_L\)-Local Loop Invariant).** \(\phi\) is a \(\varphi_L\)-local loop invariant for loop \(L_r\) if (a) \(V(\phi) \subseteq V_w(L_r)\); and (b) \(\phi\) is \(\varphi_L\)-inductive on \(L_r\).

Informally, the Theorem 1 says that whenever a property \(\varphi_L\), used to deduce that a local property \(\phi\) holds on a reduced loop, is itself an inductive invariant on the entire loop, then \(\varphi_L \land \phi\) is an inductive invariant of the entire loop.

**Theorem 1 (Compositionality of \(\varphi_L\)-Local Invariants).** Assume the loops \(L = \ell_{(\alpha, \omega, \epsilon)}\{G \parallel B\}\) and \(L_B = \ell_{(\alpha, \omega, \epsilon)}\{B\}\) are well-formed. Assume that

(a) \(\phi\) is a \(\varphi_L\)-local loop invariant on \(L_B\); (b) \(\varphi_L\) is an \(\ell_L\)-invariant on \(L\).

Then, \(\varphi_L \land \phi\) is an \(\ell_L\)-invariant on \(L\).

**Proof.** Following Lemma 3.1, \(\varphi_L \land \phi\) is an \(\ell_L\)-invariant of \(L\), if: (a) \(\ell_L\) covers \(\varphi_L \land \phi\); (b) \(i = \alpha \Rightarrow \varphi_L \land \phi\); and (c) \(A \Rightarrow \phi(i + 1) \land \varphi_L(i + 1)\).
where \( A = \text{sp}(\overrightarrow{G} \parallel B) \), \( \alpha \leq i \leq \omega \land \neg e \land \varphi_L(i) \land \phi(i) \). From \((h_1)\), conditions (a) and (b) hold by definition. By \((h_2)\) and Lemma 3.1 we know that \( A \Rightarrow \varphi_L(i+1) \). Thus, we only need to prove \( A \Rightarrow \phi(i+1) \). We unfold the \( \text{sp} \) definition in \( A \) and deduce:

\[
A \Rightarrow \exists V'.(\alpha \leq i \leq \omega \land \neg e(i)^V \land \varphi_L(i)^V \land \phi(i)^V \land \text{Psp}(B,V)),
\]

where \( V_G = V_B(\overrightarrow{G}), V_B = V_w(B) \) and \( V = V_G \cup V_B \). By corollary 2, we can replace \( \text{Psp}(B,V) \) by \((\text{Psp}(B,V_B))^V \). Moreover, by hypothesis of well-formedness on \( L \), we know that \( V_G \cap V_B = \emptyset \). Therefore, any predicate \( P^V \) can be written as \((P^V)^V \) and we obtain (1) below, where \( C \equiv \text{sp}(B,\alpha \leq i \leq \omega \land \neg e \land \varphi_L(i) \land \phi(i)) \), which can be expanded to \( C \equiv \exists V'_B, \alpha \leq i \leq \omega \land \neg e^V \land \varphi_L^V \land \phi^V \land \text{Psp}(B,V_B) \). On the other hand, by \((h_1)\) we also have (2) below:

\[
A \Rightarrow \exists V'_G.C^V, \quad (1) \quad C \Rightarrow \phi(i+1) \quad (2)
\]

To conclude, we need to rewrite (1) and (2) with explicit universal quantifications on the free variables \( \vec{x} \) of these formulas (see 2.1):

\[
\forall \vec{x}.A \Rightarrow \exists V'_G.C^V \quad (1') \quad \forall \vec{x}.C \Rightarrow \phi(i+1) \quad (2')
\]

We now prove that \( \forall \vec{x}.A \Rightarrow \phi(i+1) \). Suppose that for some \( \vec{a}, A[\vec{x} \leftarrow \vec{a}] \) holds, let us prove that \( \phi(i+1)[\vec{x} \leftarrow \vec{a}] \) also holds. By (1') we have: \( \exists V'_G.(C^V[\vec{x} \leftarrow \vec{a}]) \). Therefore there exists \( v_1 \ldots v_n \) such that \( C^V[\vec{x} \leftarrow \vec{a}][V'_G \leftarrow \vec{v}] \) holds, which is identical to \( C^V[\vec{x} \leftarrow \vec{v}] = C[\vec{x} \leftarrow \vec{v}] \) which is itself identical to \( C[V_G \leftarrow \vec{v}][\vec{x} \leftarrow \vec{a}] \). We can now apply (2') and deduce \( \phi(i+1)[V_G \leftarrow \vec{v}][\vec{x} \leftarrow \vec{a}] \). Since \( V_G \cap V(\phi) = \emptyset \) we finally have \( \phi(i+1)[\vec{x} \leftarrow \vec{a}] \). \( \square \)

# 4 Stable Loop Patterns

In this section, we introduce the stability property for expressions, and we give sufficient conditions for expressions to be stable. We define \( \varphi_L \)-stable loop patterns, as a particular instance of reduced loops restricted to stable expressions\(^3\). As examples, we present three concrete patterns and we provide the corresponding local invariants.

## 4.1 Stability

Informally, an expression \( e \) occurring in loop \( L \) is stable if, on any run through the loop, \( e \) is equal to its initial value \( e_0 \). Here, we are interested in being able to prove that \( e = e_0 \) under the assumption of a preexisting inductive loop invariant \( \varphi_L \).

**Definition 6 (Initial Value by \( \iota_L \)).** The initial value of expression \( e(i) \) by initialisation \( \iota_L \), noted \( e_0(i) \), is the result of replacing any variable \( x \) in \( e \), except \( i \), by its initial value \( x_0 \) according to \( \iota_L \): \( e_0(i) \overset{\text{def}}{=} e(i)[x \leftarrow \iota_L(x)] \).

\(^3\) More precisely, to expressions whose location expressions defined over external variables are stable.
Definition 7 (Stability). An expression \( e(i) \) is said to be \( \varphi_L \)-stable (denoted \( \varphi_L\)-st) in loop \( L \) if there exists an \( \iota_L \)-loop invariant \( \varphi_L \) on \( L \) such that:

\[
\varphi_L(i) \Rightarrow (e(i) = e_0(i)).
\]

The rationale behind stability is that, given a preexisting inductive loop invariant \( \varphi_L \), a \( \varphi_L \)-value preserving expression \( e \) can be replaced by its initial value \( e_0 \) when reasoning on the loop body using sp.

4.2 \( \varphi_L \)-Loop Patterns

Given a preexisting inductive loop invariant \( \varphi_L \), we define loop patterns relative to \( \varphi_L \), or \( \varphi_L \)-loop patterns, as triples \( P_n = (L_n,C_n,\phi_n) \). \( L_n \) is a loop scheme given by a valid loop construction in our intermediate language \( L \); \( C_n \) is a list of constraints requiring \( \varphi_L\)-st property on generic sub-expressions \( e_1,e_2,\ldots \) of \( L_n \); \( \phi_n \) is a local invariant referring only to variables local to \( L_n \).

Fig. 2 presents three concrete loop patterns. For each of them, the corresponding loop scheme is given in the upper-left entry, the constraints in the upper-right entry, and the invariant scheme in the bottom entry. To identify the pattern \( P_n \) within the source loop \( L \), \( L_n \) must match actual constructions occurring in \( L \), and the pattern constraints must be satisfied. In that case, we generate the corresponding local invariant by instantiating \( \phi_n \) with matched constructions from \( L \). We establish in Lemmas 4.1, 4.2 and 4.3 that the local property \( \phi_n \) is indeed a \( \varphi_L \)-local invariant on the reduced loop \( L_{iL_n} \), for each of the three loop patterns presented here. Thus, according to the compositional result given in Theorem 1, each generated local invariant can strengthen the preexisting \( \iota_L \)-invariant \( \varphi_L \) to obtain a richer \( \iota_L \)-invariant for loop \( L \).

<table>
<thead>
<tr>
<th>1. Search Pattern</th>
<th>2. Single Map Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_1 = \ell(\alpha,\omega){\text{skip}} ) , ( e ) is ( \varphi_L )-s.</td>
<td>( L_2 = \ell(\alpha,\omega){B_2} ) , ( e(i) ) is ( \varphi_L )-s.</td>
</tr>
<tr>
<td>( \phi_1(i) = \forall j. \alpha \leq j &lt; i \Rightarrow \neg e_0(j) )</td>
<td>( e(i) ) is ( \varphi_L )-s.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Filter Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_3 = \ell(\alpha,\omega){B_3} )</td>
</tr>
<tr>
<td>( B_3 = { g(i) \to A[v] := e(i) } )</td>
</tr>
<tr>
<td>( \parallel { g(i) \to v := v + 1 } )</td>
</tr>
<tr>
<td>( \phi_3(i,v,A) = \forall j. (\alpha \leq j &lt; i \land g_0(j) \Rightarrow \exists k. (v_0 \leq k &lt; v \land A[k] = e_0(j))) )</td>
</tr>
<tr>
<td>( \land \forall k_1, k_2, v_0 \leq k_1 \leq k_2 &lt; v \Rightarrow \exists j_1,j_2. \left( (\alpha \leq j_1 \leq j_2 &lt; i \land A[k_1] = e_0(j_1)) \right) )</td>
</tr>
<tr>
<td>( \land \forall j. (j \geq v \Rightarrow A[j] = A_0[j]) )</td>
</tr>
</tbody>
</table>

Fig. 2. Three \( \varphi_L \)-Loop Patterns
In the following we provide lemmas stating the locality of the pattern invariant schemes given in Fig. 2.

**Lemma 4.1 (Search Pattern Invariant Locality)** Given \( \iota_L, \varphi_L \) such that \( \epsilon \) is \( \varphi_L \)-st, \( \phi_1(i) \) is a \( \varphi_L \)-local loop invariant on \( L_1 \), for \( \phi_1(i), L_1 \) from Fig. 2.

*Proof.* By Definition 5 we need to prove that \( V(\phi_1) \subseteq V_w(L_1) \), which is trivial since \( V(\phi_1) = \emptyset \), and that \( \phi_1 \) is a \( \varphi_L \)-invariant. By Definition 4 this amounts to prove that (1) \( i = \alpha \land \iota_L \Rightarrow \phi_1(i) \) and (2) \( \text{sp}(\text{skip}, \alpha \leq i \leq \omega \land \neg \epsilon \land \varphi_L(i) \land \phi_1(i)) \Rightarrow \phi_1(i+1) \). Since \( i = \alpha \) implies that \( \alpha \leq j < i \) is false, then \( \phi_1(i) \) is vacuously true and (1) holds. Let us prove (2) by unfolding the definition of sp:

\[
\text{sp}(\text{skip}, \alpha \leq i \leq \omega \land \neg \epsilon(i) \land \varphi_L(i) \land \phi_1(i)) = \alpha \leq i \leq \omega \land \neg \epsilon(i) \land \varphi_L(i) \land \phi_1(i)
\]

which entails \( \phi_1(i) \) and \( \neg \epsilon_0(i) \) because \( \epsilon \) is \( \varphi_L \)-st. Therefore \( \phi_1(i+1) \) holds. \( \Box \)

**Lemma 4.2 (Single Map Pattern Invariant Locality)** Given \( \iota_L, \varphi_L \) such that \( \epsilon(i) \) is \( \varphi_L \)-st, \( \phi_2(i, A) \) is a \( \varphi_L \)-local invariant of \( L_2 \), for \( \phi_2(i, A) \) and \( L_2 \) as given in Fig. 2.

*Proof.* By Definitions 5 and 4 we have to prove \( (\epsilon \text{ is false in this pattern}) \):

- \( V(\phi_2) \subseteq V_w(L_2) \) which follows from \( V(\phi_2) = \{A\} \);
- \( i = \alpha \land \iota_L \Rightarrow \phi_2(i, A) \), which follows from \( i = \alpha \land \iota_L \Rightarrow \phi_2(\alpha, A_0) \) as \( \iota_L \Rightarrow (A = A_0) \);
- \( \text{sp}(B_2, \alpha \leq i \leq \omega \land \varphi_L(i, A) \land \phi_2(i, A)) \Rightarrow \phi_2(i+1) \) which we prove below.

Suppose that \( \text{sp}(B_2, \alpha \leq i \leq \omega \land \varphi_L(i, A) \land \phi_2(i, A)) \) holds and let us prove that \( \phi_2(i+1, A) \) holds. By the definition of sp there exists \( A' \) such that:

(a) \( \varphi_L(i, A') \)  
(b) \( \phi_2(i, A') \)  
(c) \( A[i] = e(i, A') \)  
(d) \( \forall j, j \neq i \Rightarrow A[j] = A'[j] \)

Moreover, the pattern constraint \( \epsilon(i) \) is \( \varphi_L \)-st and (a) entails \( \epsilon(i, A') = e_0(i, A_0) \).
By (c) and (d) we know that \( A \) and \( A' \) differ only on cell \( A[i] \) which contains \( e(i, A') \) which allows us to prove easily that \( \phi_2(i+1, A) \) also holds. \( \Box \)

**Lemma 4.3 (Filter Pattern Invariant Locality)** For all \( \iota_L, \varphi_L \) such that \( g(i) \) and \( e(i) \) are \( \varphi_L \)-st, \( \phi_3(i, v, A) \) as given in figure 2 is a \( \varphi_L \)-local loop invariant of the loop \( L_3 \) of figure 2.

*Proof.* In the following we denote \( \phi_3(i, v, A) \) as the conjunction \( P(i, v, A) \land Q(i, v, A) \land R(i, v, A) \). We also parameterize any non stable expressions by \( (i, v, A) \) and any stable expression by \( (i, A) \). For instance, \( \varphi_L \) and \( e \) are denoted \( \varphi_L(i, v, A) \) and \( e(i, A) \). We also use the notation \( E(i, v', A') \) for \( E'(i, v, A) \) for any expression \( E \).

The proof proceeds as it follows. By Definition 5 and 4 we must show:

- \( V(\phi_3) \subseteq V_w(L_3) \) which follows from \( V(\phi_3) = \{v, A\} \);
- \( i = \alpha \land \iota_L \Rightarrow \phi_3(i, v, A) \), which follows from \( i = \alpha \land \iota_L \Rightarrow \phi_3(\alpha, v_0, A_0) \) and that \( \iota_L \Rightarrow (v = v_0 \land A = A_0) \);
Maximal and Compositional Pattern-Based Loop Invariants

\[ \text{Since } \forall i, v, A. e(i, v, A) = \text{false } \text{we forget } e \text{ in the following. Suppose that } sp(B, \alpha \leq i \leq \omega \land \neg \epsilon(i, v, A) \land \varphi_L(i, v, A) \land \phi_3(i, v, A)) \Rightarrow \phi_3(i + 1) \text{ which we prove below.} \]

Since \( \forall i, v, A. e(i, v, A) = \text{false} \) we forget \( e \) in the following. Suppose that \( sp(B, \alpha \leq i \leq \omega \land \varphi_L(i, v, A) \land \phi_3(i, v, A)) \) holds and let us prove that \( \phi_3(i + 1, v, A) = P(i + 1, v, A) \land Q(i + 1, v, A) \land R(i + 1, v, A) \) holds. By definition of \( sp \) this implies that there exists \( v' \) and \( A' \) such that the following properties hold:

\( \begin{align*}
(\text{a}) & \quad \alpha \leq i \leq \omega \\
(\text{b}) & \quad \neg \epsilon(i, v', A') \\
(\text{c}) & \quad \varphi_L(i, v', A') \\
(\text{d}) & \quad \phi_3(i, v', A') \\
(\text{e}) & \quad g(i, A') \Rightarrow v = v' + 1 \\
(\text{f}) & \quad \neg g(i, A') \Rightarrow v = v' \\
(\text{g}) & \quad g(i, A') \Rightarrow A[v'] = e(i, A') \\
(\text{h}) & \quad \forall j. (\neg (g(i, A') \land j = v')) \Rightarrow A[j] = A'[j]
\end{align*} \)

Notice moreover that the pattern constraints on \( \varphi_L \)-stability of \( g(i) \) and \( e(i) \) together with (c) imply the following: (i) \( g(i, A') = g_0(i, A_0) \), and (j) \( e(i, A') = e_0(i, A_0) \). We now prove \( \phi_3(i + 1, v, A) \) by case on the truth of \( g(i, A') \) (which is equivalent to \( g_0(i, A') \) by pattern constraints).

- If \( g(i, A') \) is false it is easy to see that \( \phi_3(i, v, A) \rightarrow \phi_3(i + 1, v, A) \) (by contradiction for \( P(i + 1, v, A) \), trivial for \( Q(i + 1, v, A) \) and \( R(i + 1, v, A) \) since \( v' = v \) by (f)). Moreover by (f) and (h) we can replace \( v' \) and \( A' \) by \( v \) and \( A \) in (d) and have \( \phi_3(i, v, A) \).

- If \( g(i, A') \) is true, then by (e) we can replace \( v' \) by \( v - 1 \) in (d) and have \( \phi_3(i, v - 1, A') \). Moreover by (h) we know that \( A \) and \( A' \) differ only on cell \( A[v'] = A[v + 1] \) which contains \( e(i, A') \). Together with \( g(i, A') \) it is also easy to see that \( \phi_3(i + 1, v, A) \) also holds.

\[ \square \]

**Theorem 2 (Invariant Locality for Search, Map and Filter Pattern Invariants).** For \( n \in \{1, 2, 3\} \) assume that \( P_n = (L_n, C_n, \phi_n) \) corresponds to the patterns given in Fig. 2. Assume having three pairs \( (\varphi_{L_n}, \psi_{L_n}) \) satisfying each the constraints \( C_n \) for pattern \( P_n \). Then, each \( \phi_n \) is a \( \varphi_{L_n} \)-local loop invariant on the loop \( L_n \).

**Proof.** Immediate from Lemmas 4.1, 4.2 and 4.3. \( \square \)

5 Maximal Loop Invariants

In this section, we present maximality criteria on local loop invariants. A local invariant is maximal when it is stronger than any invariant on the reduced loop. For consistency, we compare loop invariants only if they are covered by the same initialisation predicate. Our notion of loop invariant maximality is independent of the language chosen to write those loops: it can be applied to any loop language equipped with a strong postcondition semantics. We show that the loop invariants we defined in Section 4, for the three concrete patterns we introduced, are indeed maximal.

**Definition 8 (Maximal \( \iota_L \)-Loop Invariant).** \( \phi \) is a maximal \( \iota_L \)-loop invariant of loop \( L \) if (1) \( \phi \) is an \( \iota_L \)-loop invariant for \( L \), and (2) for any other \( \iota_L \)-loop invariant \( \psi \) of \( L \), \( \phi \Rightarrow \psi \) is an \( \iota_L \)-loop invariant of \( L \).
Theorem 3 (Loop Invariant Maximality). Let \( L = \ell_{(\alpha, \omega, \epsilon)} \{B\} \) and assume that \( \phi \) is some formula. \( \phi \) is a maximal \( \iota_L \)-invariant of \( L \) if

(a) \( \iota_L \) covers \( \phi \)
(b) \( i = \alpha \land \iota_L \iff i = \alpha \land \phi(i) \)
(c) \( \sp(B, \alpha \leq i \leq \omega \land \neg \epsilon(i) \land \phi(i)) \iff \alpha \leq i \leq \omega \land \phi(i + 1) \)

Proof. 1. Proving that \( \phi \) is an \( \iota_L \)-invariant on \( L \): We proceed by showing that \( \phi \) fulfills conditions of Lemma 3.1. Condition (a) is a direct consequence of hypothesis (1); condition (b) follows from (2); condition (c) follows from (3). Therefore, by Lemma 3.1, \( \phi \) is a \( \iota_L \)-invariant on \( L \).

2. Proving that \( \phi \) is \( \iota_L \)-maximal: Let \( \psi \) be an \( \iota_L \)-invariant on \( L \), let us prove that \( \phi \Rightarrow \psi \) is an \( \iota_L \)-invariant on \( L \). It suffices to show that \( \phi \Rightarrow \psi \) fulfills the three conditions of Lemma 3.1. The first two are easy to show:

- verifying condition (a) on \( \phi \Rightarrow \psi \) is equivalent to ask \( V(\phi \Rightarrow \psi) \subseteq V(\iota_L) \), which holds since it holds for \( \phi \) by (1) and for \( \psi \) as condition (a) is verified on \( \psi \).
- (\( i = \alpha \land \iota_L \) \( \Rightarrow \) (\( i = \alpha \land \phi(i) \)) holds by (2) and (\( i = \alpha \land \iota_L \) \( \Rightarrow \) (\( \psi(i) \)) holds as condition (b) of Lemma 3.1 is verified on \( \psi \). Therefore, condition (b) is verified on \( i = \alpha \land \iota_L \) \( \Rightarrow \) (\( \phi(i) \Rightarrow \psi(i) \)) yielding that condition (b) holds on \( \phi \Rightarrow \psi \).

Let us prove the last condition, (c) on \( \phi \Rightarrow \psi \), i.e. for all \( i \):

\[
\sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land (\phi(i) \Rightarrow \psi(i))) \Rightarrow (\phi(i + 1) \Rightarrow \psi(i + 1)).
\]

First note that by (3), it suffices to show this property when \( \alpha \leq i \leq \omega \), as it holds vacuously on other values of \( i \). Assume \( \alpha \leq i \leq \omega \), and \( \sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land (\phi(i) \Rightarrow \psi(i))) \) and \( \phi(i + 1) \). Let us prove this entails \( \psi(i + 1) \). By Technical Lemma 5.1, we have \( \forall i. (\alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \Rightarrow \psi(i) \) and thus, \( \forall i. (\alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \Rightarrow (\alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)) \). Therefore by SP-MONO, we have for all \( i \):

\[
\sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \Rightarrow \sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i))
\]

Since \( \sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \) holds when \( \alpha \leq i \leq \omega \) by (3), and having \( \phi(i + 1) \) by previous assumption, we obtain \( \sp(B, \alpha \leq i \leq \omega \land \neg \epsilon \land \psi(i)) \). Finally, since \( \psi \) is an \( \iota_L \)-invariant, we obtain from last result and as we know that condition (c) holds on \( \psi \), the desired result \( \psi(i + 1) \).

The following technical lemma states that if the conditions (a), (b) and (c) of Theorem 3 hold for an \( \iota_L \)-invariant, then: \( \forall i. (\alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \Rightarrow \psi(i) \).

Lemma 5.1 (Invariant maximality technical lemma) Let \( \iota_L \) be an initialisation on the loop \( L = \ell_{(\alpha, \omega, \epsilon)} \{B\} \), where \( i \not\in V_B(B) \). Let \( \phi \) be a \( \iota_L \)-loop invariant such that:
(1) \( \iota_L \) covers \( \phi \).

(2) \( i = \alpha \land \iota_L \iff i = \alpha \land \phi(i) \)

(3) \( \text{sp}(B, \alpha \leq i \leq \omega \land \neg \epsilon(i) \land \phi(i)) \iff \alpha \leq i \leq \omega \land \phi(i+1) \)

Then for any \( \iota_L \)-invariant \( \psi \), \( \forall i. (\alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i)) \implies \psi(i) \)

Proof. We prove \( i \leq \omega \land \neg \epsilon \land \phi(i) \implies \psi(i) \) by induction on \( i \geq \alpha \). Without loss of generality we suppose \( \omega \geq \alpha \).

1. Case \( i = \alpha \)
   
   As condition (b) of Lemma 3.1 holds on \( \psi \), we know that \( (\iota_L \land i = \alpha) \implies \psi(i) \) holds. By (2) we have therefore: \( (\phi(i) \land i = \alpha) \implies \psi(i) \) which implies for the base case \( i = \alpha \) that:
   
   \[ \alpha \leq i \leq \omega \land \neg \epsilon \land \phi(i) ) \implies \psi(i) \]

2. Inductive step
   
   Assume:
   
   \[ i \leq \omega \land \neg \epsilon \land \phi(i) \implies \psi(i) \] (Hi)

   Let us prove \( i + 1 \leq \omega \land \neg \epsilon \land \phi(i+1) \implies \psi(i+1) \). Assume:
   
   \[ i + 1 \leq \omega \land \neg \epsilon \land \phi(i+1) \] (H1)

   Let us prove that \( \psi(i+1) \) holds. We can rewrite (Hi) as:
   
   \[ i \leq \omega \land \neg \epsilon \land \phi(i) \implies i \leq \omega \land \neg \epsilon \land \psi(i) \]

   Applying \( \text{sp-mono} \) on it we have:
   
   \[ \text{sp}(B, i \leq \omega \land \neg \epsilon \land \phi(i)) \implies \text{sp}(B, i \leq \omega \land \neg \epsilon \land \psi(i)) \]

   By (3) on the left hand side and as condition (c) of Lemma 3.1 holds on \( \psi \), on the right hand side we obtain:
   
   \[ \alpha \leq i \leq \omega \land \phi(i+1) \implies \text{sp}(B, i \leq \omega \land \neg \epsilon \land \psi(i)) \implies \psi(i+1) \]

   Since \( \alpha \leq i \leq \omega \land \phi(i+1) \) holds by (H1), we obtain the desired result \( \psi(i+1) \).

\[ \square \]

Definition 9 (Local Invariant Maximality). Let \( L = \ell_{(\alpha, \omega, \epsilon)} \{ \overrightarrow{G} \parallel B \} \) be a well-formed loop, and \( L_r = \ell_{(\alpha, \omega, \epsilon)} \{ B \} \). Let \( \iota_r \) be an initialisation restricted to variables occurring in \( L_r \), and \( \triangle \) a formula asserting constant values \( x = x_0 \), \( A = A_0 \) for all variables \( x, A \) external to \( L_r \). We say that \( \phi_r \) is locally maximal on \( L_r \) when \( \triangle \land \phi_r \) is a maximal \( \iota_r \)-loop invariant of \( L_r \).

Lemma 5.2 (Search Pattern Invariant Local Maximality) Let \( L \) be a well-formed loop, \( \phi_1 \) is locally maximal on the reduced loop \( L_1 \) for \( \phi_1 \), \( L_1 \) as defined in Fig. 2.

Proof. According to the Search Pattern definition and Definition 9, let us set the following predicates and notations:
\[ L_1 = \ell_{(\alpha, \omega, \epsilon)}\{\text{skip}\}, \] the pattern loop scheme,
- \( \phi_1(i) = \forall j. (\alpha \leq j < i \Rightarrow \neg\epsilon_0(j)) \), its local invariant,
- \( \iota_1 = \iota_{V(\epsilon)} \), initialisation restricted to \( L_1 \) variables;
- \( \iota_1 = \iota_{V_{\epsilon_{\omega}}(\epsilon)} \), initialisation restricted to external \( L_1 \) variables;
- \( \Phi_1(i) = \iota_1 \land \phi_1(i) \), the formula to be proved \( \iota_1 \)-maximal invariant;
- we take \( \varsigma_L = \varsigma_1 \) as pre-existing loop invariant on the reduced loop.

We first must show that:

- \( \varsigma_L = \varsigma_1 \) is an inductive invariant on loop \( L_1 \), which is straightforward as \( \varsigma_1 \) is only composed of equations \( x = x_0 \) and \( A = A_0 \) for variables occurring in \( L_1 \), and because there are no modified variables in this loop;
- \( \epsilon \) is \( \varsigma_1\)-st, which is immediate, as \( V(\epsilon) = V(\varsigma_1) \) and there are no modified variables in this loop. Thus, we have \( \varsigma_1 \Rightarrow \epsilon(i) = \epsilon_0(i) \).

Also, as there are no modified variables in \( L_1 \), we have \( \iota_1 = \varsigma_1 \). According to Definition 9, we must show that \( \Phi_1(i) \) fulfills conditions of Theorem 3 to be an \( \iota_1 \)-maximal invariant on \( L_1 \):

1. \( \iota_1 \) covers \( \Phi_1(i) \), which is immediate as \( V(\Phi_1) = V(\epsilon) = V(\iota_1) \);
2. \( \iota = \alpha \land \iota_1(i) \Leftrightarrow i = \alpha \land \Phi_1(i) \)
   When \( i = \alpha \), \( \phi_1(i) \) is vacuously true. As \( \varsigma_1 = \iota_1 \), we obtain immediately
   \( i = \alpha \land \iota_1 \Leftrightarrow i = \alpha \land \iota_1 \land \phi_1(i) \) as desired.
3. \( \text{sp} \{\text{skip}\}, \alpha \leq i < \omega \land \neg\epsilon(i) \land \Phi_1(i) \)
   \( \Leftrightarrow \alpha \leq i < \omega \land \Phi_1(i+1) \).

   We unfold the \( \text{sp} \) definition on the left-hand side of (3) and replace \( \epsilon(i) \) by \( \epsilon_0(i) \) as \( \epsilon \) is \( \varsigma_1\)-st. As \( i \) does not occur in \( \varsigma_1 \), we have that \( \varsigma_1(i) \Leftrightarrow \varsigma_1(i+1) \).

   Finally, we have \( \neg\epsilon_0(i) \land \forall j. (\alpha \leq j < i \Rightarrow \neg\epsilon_0(j)) \Leftrightarrow \phi_1(i+1) \), which ends the proof. \( \Box \)

**Remark** Remember that we ignore array bound considerations. Formally, this means that we assume in our formulas that every access to array cells \( A[a] \) is done for \( a \in [\alpha \ldots \omega] \). That is, arrays have exactly the same bounds as those of the loop index, and moreover, any expression \( a \) used as index on array \( A \) holds values within these bounds through loop execution.

**Lemma 5.3 (Single Map Local Maximaly)** Let \( L \) be a well-formed loop. \( \phi_2 \) is locally maximal on the reduced loop of \( L_2 \), where \( L_2 \) and \( \phi_2 \) are as defined in Fig. 2.

**Proof.** According to the Single Map Pattern definition and Definition 9, let us set the following predicates and notations:

- \( L_2 = \ell_{(\alpha, \omega)}\{B_2\} \)
- \( B_2 = \text{true} \rightarrow A[i] := e(i) \)
- \( \iota_2 = \iota_{V(L_2)} \), initialisation restricted to \( L_2 \) variables;
- \( \iota_2 = \iota_{V_{\epsilon_{\omega}}(L_2)} \), initialisation restricted to external \( L_2 \) variables;
- \( \Phi_2(i) = \iota_2 \land \phi_2(i) \), the formula to be proved \( \iota_2 \)-maximal invariant.
− let us take \( \varphi_2 = \Delta_{A,i} \land \Delta_2 \).

We first show that \( \varphi_2 = \Delta_2 \) satisfies stability constraints for this pattern, namely, that \( e(i) \) is \( \varphi_2 \)-st. First note, that by definition of Single Map Pattern, the expression \( e(i) \) must be initial stable in the reduced loop \( L_2 \). Without loss of generality, we assume \( e(i) \) such that any access \( A[e_i] \) to array \( A \) is such that \( e_i \geq i \), otherwise, as \( A[i] \) is assigned in this loop, \( e(i) \) would not be stable. \( e(i) \) is clearly \( \varphi_2 \)-st in \( L_2 \), by our previous hypothesis on \( e(i) \), and because any other location expression \( x \) or \( B[k] \) occurring in \( e \) necessarily corresponds to an external variable \( x \) or \( B \), which is not assigned within the reduced loop, and which by \((h1)\) is initialised in \( \Delta_2 \). Thus, we necessarily have \( \Delta_2 \Rightarrow x = x_0 \) or \( \Delta_2 \Rightarrow B = B_0 \), which yields \( e(i) \) is \( \varphi_2 \)-st.

Notice now that the formula \( \Phi_2(i) = \Delta_2 \land \phi_2(i) \) is actually equivalent to:

\[
\Phi_2(i) \Leftrightarrow \Delta_2 \land \forall j, (\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \land \forall j, (j \geq i \Rightarrow A[j] = A_0[j])
\]

\[
\Leftrightarrow \Delta_2 \land \Delta_{A,i} \land \forall j, (\alpha \leq j < i \Rightarrow A[j] = e_0(j))
\]

\[
\Leftrightarrow \varphi_2 \land \forall j, (\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \quad (\ast)
\]

According to Definition 9, we must show that \( \Phi_2 = \Delta_2 \land \phi_2 \) is a maximal \( \tau_2 \)-invariant on the reduced loop \( L_2 \). We proceed by showing that \( \Phi_2 \) fulfills the conditions of Theorem 3, namely:

1. \( \tau_2 \) covers \( \Phi_2 \), which is immediate as \( V(\Phi_2) = V(\Delta_2) \cup V(\phi_2) \), and because by definition \( V(\tau_2) = V(L_2) \).
2. \( i = \alpha \land \tau_2 \Leftrightarrow i = \alpha \land \Phi_2(i) \)
3. \( \text{sp}(B_2, \alpha \leq i \leq \omega \land \Phi_2(i)) \Leftrightarrow \alpha \leq i \leq \omega \land \Phi_2(i + 1) \)

Let us prove condition (2). As \( V(L_2) = V_w(L_2) \cup V_nw(L_2) \), and because \( A \) is the only modified variable in this loop, we know by hypothesis, that \( \tau_2 \Leftrightarrow (A = A_0) \land \Delta_2 \). When \( i = \alpha \), we have \( \forall j, (j \geq i \Rightarrow A[j] = A_0[j]) \Leftrightarrow A = A_0 \) and also that \( \forall j, (\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \) is vacuously true. Therefore, \( \phi_2(\alpha) \Leftrightarrow A = A_0 \).

We obtain:

\[
i = \alpha \land \Phi_2(i) \Leftrightarrow i = \alpha \land \Delta_2 \land (A = A_0) \Leftrightarrow i = \alpha \land \tau_2
\]

which achieves the proof of (2). We prove now condition (3). Let us call:

\[
D = \text{sp}(\text{true} \Rightarrow A[i] := e(i), \alpha \leq i \leq \omega \land \Delta_2(i) \land \phi_2(i))
\]

\[
R = \alpha \leq i \leq \omega \land \Delta_2(i + 1) \land \phi_2(i + 1)
\]

We must show \( D \Leftrightarrow R \). We develop \( D \) by unfolding the sp definition, and obtain

\[
D \Leftrightarrow \exists A'. (A[i] = e'(\{A\}(i) \land \forall j, (j \neq i \Rightarrow A[j] = A'[j])) \land \alpha \leq i \leq \omega
\]

\[
\land \Delta_2'(\{A\}(i) \land \forall j, (\alpha \leq j < i \Rightarrow A'[j] = e_0(j)))
\]

\[
\Leftrightarrow \exists A'. (D_1 \land D_2 \land D_3)
\]
where

\[ D_1 = \forall j. (\omega \geq j \geq i \Rightarrow A'[j] = A_0[j]) \land (A[i] = e'(A)(i)) \land \downarrow 2 \]
\[ D_2 = \forall j. (\alpha \leq j < i \Rightarrow (A[j] = A'[j] \land A'[j] = e_0(j))) \]
\[ D_3 = \forall j. (\omega \geq j > i \Rightarrow (A'[j] = A_0[j] \land A[j] = A'[j])) \land A'[i] = A_0[i] \land \alpha \leq i \leq \omega \]

Clearly, \( D_2 \Leftrightarrow \forall j. (\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \). By (*) and \( D_1 \), we can replace \( e(i)'(A) \) by \( e_0(i) \) within \( D_1 \). Moreover, \( A \) does not occur in \( \downarrow 2 \). Thus, we obtain:

\[ D_1 \Leftrightarrow \forall j. (\omega \geq j \geq i \Rightarrow A'[j] = A_0[j]) \land (A[i] = e_0(i)) \land \downarrow 2(i) \]

Combining this result and \( D_2 \) we have also:

\[ (A[i] = e_0(i)) \land \forall j. (\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \Leftrightarrow \forall j. (\alpha \leq j < i + 1 \Rightarrow A[j] = e_0(j)) \]

From \( D_3 \) we obtain:

\[ D_3 \Leftrightarrow (A'[i] = A_0[i]) \land \alpha \leq i \leq \omega \land \forall j. (\omega \geq j \geq i + 1 \Rightarrow A[j] = A_0[j]) \]

On the other hand, as \( \downarrow 2 \) does not contain \( i \), we have \( \downarrow 2(i) \Leftrightarrow \downarrow 2(i + 1) \). Combining these results and unfolding \( R \) definition we obtain:

\[ D \Leftrightarrow \alpha \leq i \leq \omega \land \downarrow 2 \land \forall j. (\alpha \leq j < i + 1 \Rightarrow A[j] = e_0(j)) \]
\[ \land \forall j. (\omega \geq j \geq i + 1 \Rightarrow A[j] = A_0[j]) \land \exists A'. (A'[i] = A_0[i]) \]
\[ R \Leftrightarrow \alpha \leq i \leq \omega \land \downarrow 2 \land \forall j. (\alpha \leq j < i + 1 \Rightarrow A[j] = e_0(j)) \]
\[ \land \forall j. (\omega \geq j \geq i + 1 \Rightarrow A[j] = A_0[j]) \]

By hypothesis, \( A_0[i] \) is defined as long as \( \alpha \leq i \leq \omega \) holds. Therefore \( \exists A'. (A'[i] = A_0[i]) \) is true, which ends the proof. \( \square \)

References