Maximal and Compositional Pattern-Based Loop Invariants

Definitions and Proofs

Virginia Aponte[1], Pierre Courtieu[1], and Yannick Moy[2]

[1] CNAM, 292 rue Saint-Martin F-75141 Paris Cedex 03 - FRANCE @cnam.fr
{maria-virginia.aponte_garcia,pierre.courtieu}@cnam.fr
[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 beeen 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 assignements 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_l) 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 ($\mathcal{B}$) 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; ϵ, 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.

(a)		(b)
$\mathcal{L}$ ::= **loop** i **in** α .. ω **exit** e_b **do** B **end**	loop	**loop** i **in** 1..10 **exit** A[i] = 0 **do**
$\mathcal{B}$ ::= **skip** $\mid \mathcal{G}(\parallel \mathcal{G})^*$	body	{ A[i] < 0 → B[b] := A[i]}
$\mathcal{G}$ ::= $\{s_l(;s_l)^*\}$	group	‖ { A[i] < 0 → b := b+1 }
s_l ::= $e_b \rightarrow e_l := e_a$	assignment	‖ { ¬(A[i] < 0) → C[c] := A[i]}
e_l ::= $x \mid A[e_a]$	location expr	‖ { ¬(A[i] < 0) → c := c+1 }
$e_a \in$ **Aexp**, e_b**Bexp**		‖ { **true** → A[i] := erased }
		end

Fig. 1. (a) Formal syntax of loop programs (b) A loop program example

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, $g_1 \wedge g_2$ 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 $g_1 \wedge g_2$ 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 α to its final value ω. We use $\ell_{(\alpha,\omega,\epsilon)}\{B\}$ to abbreviate **loop** i **in** $\alpha..\omega$ **exit** ϵ **do** B **end**, written $\ell_{(\alpha,\omega)}\{B\}$ when $\epsilon = \mathit{false}$. $\overrightarrow{G}$ denotes a loop body $G_1 \parallel \ldots \parallel G_n$ (for some n) and $\overrightarrow{G} \parallel B$ is the parallel composition of groups $G_1, \ldots G_n$ with groups occuring in B. Similarly, $\{\overrightarrow{g_k \rightarrow l_k := e_k}\}$ denotes a group made of the guarded assignments $\{g_1 \rightarrow l_1 := e_1; \ldots; g_n \rightarrow l_n := e_n\}$. $\mathcal{G}(B)$ 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 occuring 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 ι_V is defined as $\iota_V \Leftrightarrow \bigwedge_{v \in V} v = v_0$ 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 ϕ, ψ, ι and $\wp$ denote formulas. The loop index i may occur in the formula ϕ or in the expression e, 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 $\exists v_1 \dots v_n.\phi$ 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 $e^{'V} = e[V \leftarrow V']$ and $\phi^{'V} = \phi[V \leftarrow V']$.

2.2 Strongest Postcondition Semantics

A Semantic Formulation of sp Given a set of locations, and a set of values, we consider states σ 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 $\mathcal{L}$ given by the relation $< C, \sigma > \rightsquigarrow \sigma'$. Our theoretical framework relies on the following semantic definition [4] of the Strongest Postcondition Predicate Transformer sp, and on some of its properties, as given by corollary 1 and taken from [4].

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

$$\sigma' \vDash \mathrm{sp}(C, P) \Leftrightarrow \exists \sigma.(< C, \sigma > \rightsquigarrow \sigma' \wedge \sigma \vDash P)$$

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

$$\begin{array}{rl} [\text{SP-POST}] & \vDash_{par} \{P\}\ C\ \{\mathrm{sp}(C, P)\} \\ [\text{SP-STRG}] & \vDash_{par} \{P\}\ C\ \{Q\} \Rightarrow (\mathrm{sp}(C, P) \Rightarrow Q) \\ [\text{SP-MONO}] & (P \Rightarrow Q) \Rightarrow (\mathrm{sp}(C, P) \Rightarrow \mathrm{sp}(C, Q)) \end{array}$$

A Syntactic formulation of sp on $\mathcal{L}$ loop bodies We express the semantics of our intermediate language $\mathcal{L}$ through a formal definition of sp. In Definition 2, we give a syntactic formulation of 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 sp). *Let ϕ be a formula, $\overrightarrow{G_k}$ a loop body, and $V = V_w(\overrightarrow{G_k})$. We define $\mathrm{sp}(\overrightarrow{G_k}, \phi)$ as:*

$$\mathrm{sp}(\boldsymbol{skip}, \phi) = \phi \qquad \mathrm{sp}(\overrightarrow{G_k}, \phi) = \exists V'.\ \big(\phi^{'V} \bigwedge_k \mathrm{Psp}(G_k, V)\big)$$

$$\mathrm{Psp}(\{\overrightarrow{g_k \to x := e_k}\}, V) = \bigwedge_k \big(g_k^{'V} \Rightarrow x = e_k^{'V}\big) \ \wedge\ \left(\left(\bigwedge_k \neg g_k^{'V}\right) \Rightarrow x = x'\right)$$

$$\begin{aligned} \mathrm{Psp}(\{\overrightarrow{g_k \to A[a_k] := e_k}\}, V) = & \bigwedge_k \big(g_k^{'V} \Rightarrow A[a_k^{'V}] = e_k^{'V}\big) \\ & \wedge \forall j. \left(\bigwedge_k \neg\big(g_k^{'V} \wedge j = a_k^{'V}\big)\right) \Rightarrow A[j] = A'[j]. \end{aligned}$$

Corollary 2 (Renaming of External Variables in sp**).** *Let* $L = \ell_{(\alpha,\omega,\epsilon)}\{\overrightarrow{G} \parallel B\}$ *be a well-formed loop. Let* $V_G = V_w(\overrightarrow{G})$ *and* $V_B = V_w(B)$*. Then,*

$$\mathrm{Psp}(B, V_B \cup V_G) = (\mathrm{Psp}(B, V_B))'^{V_G}.$$

Proof. By definition, $\mathrm{Psp}(B, V_B \cup V_G)$ 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 $\mathrm{Psp}(B, V_B)$. From the Psp definition, its easy to see that this formula is equal to $\mathrm{Psp}(B, V_B \cup V_G)$ except for all variables in V_G that are renamed by their primed version. □

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 $\wp_L$ over the complete loop. Our notion of locality is generic with respect to $\wp_L$.

3.1 (Inductive) ι_L-Loop Invariants

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

Definition 3 ((Inductive) ι_L-Loop Invariant). *Assume* ι_L *covers a formula* ϕ*.* ϕ *is an* ι_L*-loop invariant on the loop* $L = \ell_{(\alpha,\omega,\epsilon)}\{B\}$*, iff*
(a) $(i = \alpha \wedge \iota_L) \Rightarrow \phi$*; and (b)* $\models_{\mathrm{par}} \{\alpha \le i \le \omega \wedge \neg\epsilon \wedge \phi\}\ B;\ i := i + 1\ \{\phi\}$.

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

Lemma 3.1 (ι_L-Loop Invariant Definition via sp**)** ψ *is an* ι_L*-loop invariant on loop* $L = \ell_{(\alpha,\omega,\epsilon)}\{B\}$ *iff:*
(a) ι_L *covers* ψ*;* *(b)* $i = \alpha \wedge \iota_L \Rightarrow \psi(\alpha)$*;*
(c) $\mathrm{sp}(B, \alpha \le i \le \omega \wedge \neg\epsilon \wedge \psi(i)) \Rightarrow \psi(i + 1)$.

Proof. Let us assume ψ is an ι_L-invariant. From the ι_L-invariant definition, condition (a) and (b) follow immediately. Moreover, we know that the Hoare triple $\{\alpha \le i \le \omega \wedge \neg\epsilon \wedge \psi(i)\}\ B; i := i + 1\ \{\psi(i)\}$ holds. As, $i \notin V_w(B)$, we necessarily have: $\{\alpha \le i \le \omega \wedge \neg\epsilon \wedge \psi(i)\}\ B\ \{\psi(i + 1)\}\ i := i + 1\ \{\psi(i)\}$, otherwise

ψ would not be an inductive invariant. Using SP-STRG on the triple $\{\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i)\}\ B\ \{\psi(i+1)\}$ we obtain $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i)) \Rightarrow \psi(i+1)$ as desired.
Assume now that hypothesis (ab), (b), and (c) hold, and let σ_1 be a state such that $\sigma_1 \vDash_{par} \mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i))$. Then, by the definition of sp we have $\exists \sigma_0.\ < C, \sigma_0 > \rightsquigarrow \sigma_1 \wedge \sigma_0 \vDash \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i)$. On the other hand, from hypothesis (c), we obtain that $\sigma_1 \vDash_{par} \psi(i+1)$ holds. By the definition of Partial Correctness satisfaction, we obtain that $\{\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i)\}\ B\ \{\psi(i+1)\}$ holds. Clearly, $\{\psi(i+1)\}\ i := i+1\ \{\psi(i)\}$ holds as well, which yields $\{\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i)\}\ B; i := i+1\ \{\psi(i)\}$. This, together with (a) and (b), shows that ψ is an ι_L-invariant for L. □

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 occuring within B (*i.e.* $\mathcal{G}(B_r) \subseteq \mathcal{G}(B)$). 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 $\wp_L$ 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: ϕ is inductive relative to another formula $\wp_L$, on loop L, when the inductive step of the proof of ϕ holds under the assumption $\wp_L$.

Definition 4 (Relative Inductive Invariant). *A property ϕ is $\wp_L$-inductive on loop L, if*
(1) ι_L covers $\wp_L \wedge \phi$ and $(i = \alpha \wedge \iota_L) \Rightarrow \phi$;
(2) $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \wp_L(i) \wedge \phi(i)) \Rightarrow \phi(i+1)$.

ϕ is a $\wp_L$-*local loop invariant* on loop L_r, if ϕ only refers to variables locally modified in L_r, and if ϕ holds inductively on L_r relatively to the property $\wp_L$.

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

Informally, the Theorem 1 says that whenever a property $\wp_L$, used to deduce that a local property ϕ holds on a reduced loop, is itself an inductive invariant on the entire loop, then $\wp_L \wedge \phi$ is an inductive invariant of the entire loop.

Theorem 1 (Compositionality of $\wp_L$-Local Invariants). *Assume the loops $L = \ell_{(\alpha,\omega,\epsilon)}\{\overrightarrow{G} \parallel B\}$ and $L_B = \ell_{(\alpha,\omega,\epsilon)}\{B\}$ are well-formed. Assume that (h_1) ϕ is a $\wp_L$-local loop invariant on L_B; (h_2) $\wp_L$ is an ι_L-invariant on L. Then, $\wp_L \wedge \phi$ is an ι_L-invariant on L.*

Proof. Following Lemma 3.1, $\wp_L \wedge \phi$ is an ι_L-invariant of L, if: (a) ι_L covers $\wp_L \wedge \phi$; (b) $i = \alpha \Rightarrow \wp_L \wedge \phi$; and (c) $A \Rightarrow \phi(i+1) \wedge \wp_L(i+1)$

where $A = \mathrm{sp}(\overrightarrow{G} \parallel B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \wp_L(i) \wedge \phi(i))$. From ($h_1$), conditions (a) and (b) hold by definition. By (h_2) and Lemma 3.1 we know that $A \Rightarrow \wp_L(i+1)$. Thus, we only need to prove $A \Rightarrow \phi(i+1)$. We unfold the sp definition in A and deduce:
$A \Rightarrow \exists V'.(\alpha \leq i \leq \omega \wedge \neg\epsilon(i)^{'V} \wedge \wp_L(i)^{'V} \wedge \phi(i)^{'V} \wedge \mathrm{Psp}(B, V))$,

where $V_G = V_w(\overrightarrow{G})$, $V_B = V_w(B)$ and $V = V_G \cup V_B$. By corollary 2, we can replace $\mathrm{Psp}(B, V)$ by $(\mathrm{Psp}(B, V_B))^{'V_G}$. 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 $(P^{'V_B})^{'V_G}$ and we obtain (1) below, where $C \equiv \mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \wp_L(i) \wedge \phi(i))$, which can be expanded to $C \equiv \exists V'_B.\alpha \leq i \leq \omega \wedge \neg\epsilon^{'V_B} \wedge \wp_L^{'V_B} \wedge \phi^{'V_B} \wedge \mathrm{Psp}(B, V_B)$. On the other hand, by (h_1) we also have (2) below:

$$A \Rightarrow \exists V'_G.C^{'V_G} \quad (1) \qquad\qquad C \Rightarrow \phi(i+1) \quad (2)$$

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):

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

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: $\exists V'_G.(C^{'V_G}[\vec{x} \leftarrow \vec{a}])$. Therefore there exists $v_1 \ldots v_n$ such that $C^{'V_G}[\vec{x} \leftarrow \vec{a}][V'_G \leftarrow \vec{v_i}]$ holds, which is identical to $C^{'V_G}[V'_G \leftarrow \vec{v_i}][\vec{x} \leftarrow \vec{a}]$ which is itself identical to $C[V_G \leftarrow \vec{v_i}][\vec{x} \leftarrow \vec{a}]$. 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 $\phi(i+1)[\vec{x} \leftarrow \vec{a}]$. □

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 $\wp_L$-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 $\wp_L$.

Definition 6 (Initial Value by ι_l). *The initial value of expression $e(i)$ by initialisation ι_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 ι_L:* $\quad e_0(i) \stackrel{def}{=} e(i)[x \leftarrow \iota_L(x)]$.

[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 $\wp_L$-stable (denoted $\wp_L$-st) in loop L if there exists an ι_L-loop invariant $\wp_L$ on L such that:*

$$\wp_L(i) \Rightarrow (e(i) = e_0(i)).$$

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

4.2 $\wp_L$-Loop Patterns

Given a preexisting inductive loop invariant $\wp_L$, we define loop patterns relative to $\wp_L$, or $\wp_L$-*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 $\mathcal{L}$; C_n is a list of constraints requiring $\wp_L$-st property on generic sub-expressions $e_1, e_2 \ldots$ of L_n; ϕ_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 ϕ_n with matched constructions from L. We establish in Lemmas 4.1, 4.2 and 4.3 that the local property ϕ_n is indeed a $\wp_L$-local invariant on the reduced loop $L_{\downarrow L_n}$, 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 ι_L-invariant $\wp_L$ to obtain a richer ι_L-invariant for loop L.

1. Search Pattern	
$L_1 = \ell_{(\alpha,\omega,\epsilon)}\{\mathbf{skip}\}$	ϵ is $\wp_L$-s.
$\phi_1(i) = \forall j. \alpha \le j < i \Rightarrow \neg\epsilon_0(j)$	

2. Single Map Pattern	
$L_2 = \ell_{(\alpha,\omega)}\{B_2\}$ $B_2 = \texttt{true} \to A[i] := e(i)$	$e(i)$ is $\wp_L$-s.
$\phi_2(i, A) = \begin{array}{l} \forall j.(\alpha \le j < i \Rightarrow A[j] = e_0(j)) \\ \wedge\ \forall j. j \ge i \Rightarrow A[j] = A_0[j] \end{array}$	

3. Filter Pattern	
$L_3 = \ell_{(\alpha,\omega)}\{B_3\}$ $B_3 = \{g(i) \to A[v] := e(i)\}$ $\quad \parallel \{g(i) \to v := v + 1\}$	g, e are $\wp_L$-s.
$\phi_3(i, v, A) = \forall j.(\alpha \le j < i \wedge g_0(j) \Rightarrow \exists k.(v_0 \le k < v \wedge A[k] = e_0(j)))$ $\wedge\ \forall k_1, k_2. v_0 \le k_1 \le k_2 < v \Rightarrow \exists j_1, j_2. \begin{pmatrix} \alpha \le j_1 \le j_2 < i \wedge A[k_1] = e_0(j_1) \\ g_0(j_1) \wedge g_0(j_2) \wedge A[k_2] = e_0(j_2) \end{pmatrix}$ $\wedge\ \forall j.(j \ge v \Rightarrow A[j] = A_0[j])$	

Fig. 2. Three $\wp_L$-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 $\iota_L, \wp_L$ such that ϵ is $\wp_L$-st, $\phi_1(i)$ is a $\wp_L$-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 ϕ_1 is a $\wp_L$-invariant. By Definition 4 this amounts to prove that (1) $i = \alpha \wedge \iota_L \Rightarrow \phi_1(i)$ and (2) $\mathrm{sp}(\mathbf{skip}, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \wp_L(i) \wedge \phi_1(i)) \Rightarrow \phi_1(i+1)$. Since $i = \alpha$ implies that $\alpha \leq j < i$ is false, then $\phi_1(i)$ is vacuosly true and (1) holds. Let us prove (2) by unfolding the definition of sp:

$$\mathrm{sp}(\mathbf{skip}, \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \wp_L(i) \wedge \phi_1(i)) = \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \wp_L(i) \wedge \phi_1(i)$$

which entails $\phi_1(i)$ and $\neg\epsilon_0(i)$ because ϵ is $\wp_L$-st. Therefore $\phi_1(i+1)$ holds. □

Lemma 4.2 (Single Map Pattern Invariant Locality) *Given $\iota_L, \wp_L$ such that $e(i)$ is $\wp_L$-st, $\phi_2(i, A)$ is a $\wp_L$-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 (ϵ is false in this pattern):

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

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

(a) $\wp_L(i, A')$ (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 ($e(i)$ is $\wp_L$-st) and (a) entails $e(i, A') = e_0(i, A_0)$. 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. □

Lemma 4.3 (Filter Pattern Invariant Locality) *For all $\iota_L, \wp_L$ such that $g(i)$ and $e(i)$ are $\wp_L$-st, $\phi_3(i, v, A)$ as given in figure 2 is a $\wp_L$-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 $P(i, v, A) \wedge Q(i, v, A) \wedge R(i, v, A)$. We also parameterize any non stable expressions by (i, v, A) and any stable expression by (i, A). For instance, $\wp_L$ and e are denoted $\wp_L(i, v, A)$ and $e(i, A)$. We also use the notation $E(i, v', A')$ for $E'^{\{v,A\}}$ 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 $V \stackrel{def}{=} V(\phi_3) = \{v, A\}$;
- $i = \alpha \wedge \iota_L \Rightarrow \phi_3(i, v, A)$, which follows from $i = \alpha \wedge \iota_L \Rightarrow \phi_3(\alpha, v_0, A_0)$ and that $\iota_L \Rightarrow (v = v_0 \wedge A = A_0)$;

– $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon(i,v,A) \wedge \wp_L(i,v,A) \wedge \phi_3(i,v,A)) \Rightarrow \phi_3(i+1)$ which we prove below.

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

(a) $\alpha \leq i \leq \omega$	(e) $g(i,A') \Rightarrow v = v'+1$
(b) $\neg\epsilon(i,v',A')$	(f) $\neg g(i,A') \Rightarrow v = v'$
(c) $\wp_L(i,v',A')$	(g) $g(i,A') \Rightarrow A[v'] = e(i,A')$
(d) $\phi_3(i,v',A')$	(h) $\forall j.(\neg(g(i,A') \wedge j = v')) \Rightarrow A[j] = A'[j]$

Notice moreover that the pattern constraints on $\wp_L$-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 $(\iota_{L_n}, \wp_{L_n})$ satifying each the constraints C_n for pattern P_n. Then, each ϕ_n is a $\wp_{L_n}$-local loop invariant on the loop L_n.*

Proof. Inmediate 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 ι_L-Loop Invariant). *ϕ is a maximal ι_L-loop invariant of loop L if (1) ϕ is an ι_L-loop invariant for L, and (2) for any other ι_L-loop invariant ψ of L, $\phi \Rightarrow \psi$ is an ι_L-loop invariant of L.*

Theorem 3 (Loop Invariant Maximality). *Let $L = \ell_{(\alpha,\omega,\epsilon)}\{B\}$ and assume that ϕ is some formula. ϕ is a maximal ι_L-invariant of L if*

(a) ι_L *covers* ϕ
(b) $i = \alpha \wedge \iota_L \Leftrightarrow i = \alpha \wedge \phi(i)$
(c) $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \phi(i)) \Leftrightarrow \alpha \leq i \leq \omega \wedge \phi(i+1)$

Proof. 1. Proving that ϕ is an ι_L-invariant on L: We proceed by showing that ϕ 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, ϕ is a ι_L-invariant on L.

2. Proving that ϕ is ι_L-maximal: Let ψ be an ι_L-invariant on L, let us prove that $\phi \Rightarrow \psi$ is an ι_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 ϕ by (1) and for ψ as condition (a) is verified on ψ.
- $(i = \alpha \wedge \iota_L) \Rightarrow (i = \alpha \wedge \phi(i))$ holds by (2) and $(i = \alpha \wedge \iota_L) \Rightarrow (\psi(i))$ holds as condition (b) of Lemma 3.1 is verified on ψ. Therefore, condition (b) is verified on $i = \alpha \wedge \iota_L) \Rightarrow (\phi(i) \Rightarrow \psi(i))$ 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:

$$\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge (\phi(i) \Rightarrow \psi(i))) \Rightarrow (\phi(i+1) \Rightarrow \psi(i+1)).$$

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 $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge (\phi(i) \Rightarrow \psi(i)))$ and $\phi(i+1)$. Let us prove this entails $\psi(i+1)$. By Technical Lemma 5.1, we have $\forall i.\ (\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \psi(i)$ and thus, $\forall i.\ (\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow (\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i))$. Therefore by SP-MONO, we have for all i:

$$\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \psi(i))$$

Since $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i))$ holds when $\alpha \leq i \leq \omega$ by (3), and having $\phi(i+1)$ by previous assumption, we obtain $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \psi(i))$. Finally, since ψ is an ι_L-invariant, we obtain from last result and as we know that condition (c) holds on ψ, 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 ι_L-invariant, then: $\forall i.(\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \psi(i)$.

Lemma 5.1 (Invariant maximality technical lemma) *Let ι_L be an initialisation on the loop $L = \ell_{(\alpha,\omega,\epsilon)}\{B\}$, where $i \notin V_w(B)$. Let ϕ be a ι_L-loop invariant such that:*

(1) ι_L *covers* ϕ.
(2) $i = \alpha \wedge \iota_L \Leftrightarrow i = \alpha \wedge \phi(i)$
(3) $\mathrm{sp}(B, \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \phi(i)) \Leftrightarrow \alpha \leq i \leq \omega \wedge \phi(i+1)$

Then for any ι_L*-invariant* ψ, $\forall i.(\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \psi(i)$

Proof. We prove $i \leq \omega \wedge \neg\epsilon \wedge \phi(i) \Rightarrow \psi(i)$ 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 ψ, we know that $(\iota_L \wedge i = \alpha) \Rightarrow \psi(i)$ holds. By (2) we have therefore: $(\phi(i) \wedge i = \alpha) \Rightarrow \psi(i)$ which implies for the base case $i = \alpha$ that:

$$(\alpha \leq i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \psi(i)$$

2. Inductive step
Assume:

$$i \leq \omega \wedge \neg\epsilon \wedge \phi(i) \Rightarrow \psi(i) \tag{Hi}$$

Let us prove $i+1 \leq \omega \wedge \neg\epsilon \wedge \phi(i+1) \Rightarrow \psi(i+1)$. Assume:

$$i+1 \leq \omega \wedge \neg\epsilon \wedge \phi(i+1) \tag{H1}$$

Let us prove that $\psi(i+1)$ holds. We can rewrite (Hi) as:

$$i \leq \omega \wedge \neg\epsilon \wedge \phi(i) \Rightarrow i \leq \omega \wedge \neg\epsilon \wedge \psi(i)$$

Applying SP-MONO on it we have:

$$\mathrm{sp}(B, i \leq \omega \wedge \neg\epsilon \wedge \phi(i)) \Rightarrow \mathrm{sp}(B, i \leq \omega \wedge \neg\epsilon \wedge \psi(i))$$

By (3) on the left hand side and as condition (c) of Lemma 3.1 holds on ψ, on the right hand side we obtain:

$$\alpha \leq i \leq \omega \wedge \phi(i+1) \Rightarrow \mathrm{sp}(B, i \leq \omega \wedge \neg\epsilon \wedge \psi(i)) \Rightarrow \psi(i+1)$$

Since $\alpha \leq i \leq \omega \wedge \phi(i+1)$ 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* ι_r *be an initialisation restricted to variables occurring in* L_r*, and* $\exists$ *a formula asserting constant values* $x = x_0$, $A = A_0$ *for all variables* x, A *external to* L_r. *We say that* ϕ_r *is locally maximal on* L_r *when* $\exists \wedge \phi_r$ *is a maximal* ι_r*-loop invariant of* L_r.

Lemma 5.2 (Search Pattern Invariant Local Maximality) *Let* L *be a well-formed loop.* ϕ_1 *is locally maximal on the reduced loop* L_1 *for* ϕ_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:

- $L_1 = \ell_{(\alpha,\omega,\epsilon)}\{\mathbf{skip}\}$, 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;
- $\daleth_1 = \iota_{V_{nw}(\epsilon)}$, initialisation restricted to external L_1 variables;
- $\Phi_1(i) = \daleth_1 \wedge \phi_1(i)$, the formula to be proved ι_1-maximal invariant;
- we take $\wp_L = \daleth_1$ as pre-existing loop invariant on the reduced loop.

We first must show that:

i. $\wp_L = \daleth_1$ is an inductive invariant on loop L_1, which is straightforward as $\daleth_1$ 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;
ii. ϵ is $\daleth_1$-st, which is inmediate, as $V(\epsilon) = V(\daleth_1)$ and there are no modified variables in this loop. Thus, we have $\daleth_1 \Rightarrow \epsilon(i) = \epsilon_0(i)$.

Also, as there are no modified variables in L_1, we have $\iota_1 = \daleth_1$. According to Definition 9, we must show that $\Phi_1(i)$ fulfills conditions of Theorem 3 to be an ι_1-maximal invariant on L_1:

1. ι_1 covers $\Phi_1(i)$, which is immediate as $V(\Phi_1) = V(\epsilon) = V(\iota_1)$;
2. $i = \alpha \wedge \iota_1(i) \Leftrightarrow i = \alpha \wedge \Phi_1(i)$
   When $i = \alpha$, $\phi_1(i)$ is vacuosly true. As $\daleth_1 = \iota_1$, we obtain inmediately $i = \alpha \wedge \iota_1 \Leftrightarrow i = \alpha \wedge \iota_1 \wedge \phi_1(i)$ as desired.
3. $\mathrm{sp}(\mathbf{skip}, \alpha \leq i \leq \omega \wedge \neg\epsilon(i) \wedge \Phi_1(i)) \Leftrightarrow \alpha \leq i \leq \omega \wedge \wedge\Phi_1(i+1)$.
   We unfold the sp definition on the left-hand side of (3) and replace $\epsilon(i)$ by $\epsilon_0(i)$ as ϵ is $\daleth_1$-st. As i does not occur in $\daleth_1$, we have that $\daleth_1(i) \Leftrightarrow \daleth_1(i+1)$. Finally, we have $\neg\epsilon_0(i) \wedge \forall j.(\alpha \leq j < i \Rightarrow \neg\epsilon_0(j)) \Leftrightarrow \phi_1(i+1)$, which ends the proof.

$\square$

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 Maximality) *Let L be a well-formed loop. ϕ_2 is locally maximal on the reduced loop of L_2, where L_2 and ϕ_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:

- $L_2 = \ell_{(\alpha,\omega)}\{B_2\}$,
- $B_2 = \mathtt{true} \rightarrow A[i] := e(i)$
- $\iota_2 = \iota_{V(L_2)}$, initialisation restricted to L_2 variables;
- $\daleth_2 = \iota_{V_{nw}(L_2)}$, initialisation restricted to external L_2 variables;
- $\Phi_2(i) = \daleth_2 \wedge \phi_2(i)$, the formula to be proved ι_2-maximal invariant.

– let us take $\wp_2 = \triangle_{A,i} \wedge \dashv_2$.

We first show that $\wp_2 = \dashv_2$ satisfies stability constraints for this pattern, namely, that $e(i)$ is $\wp_2$-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 $A[e_a]$ to array A is such that $e_a \geq i$, otherwise, as $A[i]$ is assigned in this loop, $e(i)$ would not be stable. $e(i)$ is clearly $\wp_2$-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 $\dashv_2$. Thus, we necessarily have $\dashv_2 \Rightarrow x = x_0$ or $\dashv_2 \Rightarrow B = B_0$, which yields $e(i)$ is $\wp_2$-st.

Notice now that the formula $\Phi_2(i) = \dashv_2 \wedge \phi_2(i)$ is actually equivalent to:

$$\begin{aligned}\Phi_2(i) &\Leftrightarrow \dashv_2 \wedge \forall j.(\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \wedge \forall j.(j \geq i) \Rightarrow A[j] = A_0[j] \\ &\Leftrightarrow \dashv_2 \wedge \triangle_{A,i} \wedge \forall j.(\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \\ &\Leftrightarrow \wp_2 \wedge \forall j.(\alpha \leq j < i \Rightarrow A[j] = e_0(j)) \qquad (\star)\end{aligned}$$

According to Definition 9, we must show that $\Phi_2 = \dashv_2 \wedge \phi_2$ is a maximal ι_2-invariant on the reduced loop L_2. We proceed by showing that Φ_2 fulfills the conditions of Theorem 3, namely:

1. ι_2 covers Φ_2, which is immediate as $V(\Phi_2) = V(\dashv_2) \cup V(\phi_2)$, and because by definition $V(\iota_2) = V(L_2)$.
2. $i = \alpha \wedge \iota_2 \Leftrightarrow i = \alpha \wedge \Phi_2(i)$
3. $\mathrm{sp}(B_2, \alpha \leq i \leq \omega \wedge \Phi_2(i)) \Leftrightarrow \alpha \leq i \leq \omega \wedge \Phi_2(i+1)$

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 $\iota_2 \Leftrightarrow (A = A_0) \wedge \dashv_2$. 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 vacuosly true. Therefore, $\phi_2(\alpha) \Leftrightarrow A = A_0$. We obtain:

$$i = \alpha \wedge \Phi_2(i) \Leftrightarrow i = \alpha \wedge \dashv_2 \wedge (A = A_0) \Leftrightarrow i = \alpha \wedge \iota_2$$

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

$$\begin{aligned}D &= \mathrm{sp}(\mathtt{true} \to A[i] := e(i), \alpha \leq i \leq \omega \wedge \dashv_2(i) \wedge \phi_2(i)) \\ R &= \alpha \leq i \leq \omega \wedge \dashv_2(i+1) \wedge \phi_2(i+1)\end{aligned}$$

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

$$\begin{aligned}D \Leftrightarrow \exists A'.\,(A[i] = e^{'\{A\}}(i) \wedge \forall j.(j \neq i \Rightarrow A[j] = A'[j]) \wedge \alpha \leq i \leq \omega \\ \wedge \dashv_2^{'\{A\}}(i) \wedge \forall j.(\alpha \leq j < i \Rightarrow A'[j] = e_0(j))) \\ \Leftrightarrow \exists A'.\,(D_1 \wedge D_2 \wedge D_3)\end{aligned}$$

where

$$
\begin{aligned}
D_1 = \quad & \forall j.(\omega \geq j \geq i \Rightarrow A'[j] = A_0[j]) \wedge (A[i] = e^{\prime\{A\}}(i)) \wedge \daleth_2 \\
D_2 = \quad & \forall j.(\alpha \leq j < i \Rightarrow (A[j] = A'[j] \wedge A'[j] = e_0(j))) \\
D_3 = \quad & \forall j.(\omega \geq j > i \Rightarrow (A'[j] = A_0[j] \wedge A[j] = A'[j])) \wedge A'[i] = A_0[i] \wedge \alpha \leq i \leq \omega
\end{aligned}
$$

Clearly, $D_2 \Leftrightarrow \forall j.(\alpha \leq j < i \Rightarrow A[j] = e_0(j))$. By $(\star)$ and D_1, we can replace $e(i)^{\prime\{A\}}$ by $e_0(i)$ within D_1. Moreover, A does not occur in $\daleth_2$. Thus, we obtain:

$$D_1 \Leftrightarrow \forall j.(\omega \geq j \geq i \Rightarrow A'[j] = A_0[j]) \wedge (A[i] = e_0(i)) \wedge \daleth_2(i)$$

Combining this result and D_2 we have also:

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

From D_3 we obtain :

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

On the other hand, as $\daleth_2$ does not contain i, we have $\daleth_2(i) \Leftrightarrow \daleth_2(i{+}1)$. Combining these results and unfolding R definition we obtain:

$$
\begin{aligned}
D \Leftrightarrow\ & \alpha \leq i \leq \omega \wedge \daleth_2 \wedge \forall j.(\alpha \leq j < i + 1 \Rightarrow A[j] = e_0(j)) \\
& \wedge \forall j.(\omega \geq j \geq i + 1 \Rightarrow A[j] = A_0[j]) \wedge \exists A'.(A'[i] = A_0[i]) \\
R \Leftrightarrow\ & \alpha \leq i \leq \omega \wedge \daleth_2 \wedge \forall j.(\alpha \leq j < i + 1 \Rightarrow A[j] = e_0(j)) \\
& \wedge \forall j.(\omega \geq j \geq i + 1 \Rightarrow A[j] = A_0[j])
\end{aligned}
$$

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. □

References

1. A. Bradley and Z. Manna. Property-directed incremental invariant generation. *Formal Aspects of Computing*, 20:379–405, 2008. 10.1007/s00165-008-0080-9.
2. C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12:576–580, October 1969.
3. L. Kovács and A. Voronkov. Finding loop invariants for programs over arrays using a theorem prover. In *Proceedings of the 2009 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC '09, Washington, DC, USA, 2009. IEEE Computer Society.
4. H. R. Nielson and F. Nielson. *Semantics with Applications: a formal introduction.* John Wiley & Sons, Inc., New York, NY, USA, 1992.