Danicic, Sebastian, Laurence, Michael and Hierons, Robert

Decidability of Strong Equivalence for Subschemas of a Class of Linear, Free, near-Liberal Program Schemas

Goldsmiths Department of Computing
Internal Report #1 Dec 2009

Decidability of Strong Equivalence for Subschemas of a Class of Linear, Free, near-Liberal Program Schemas

Sebastian Danicic[b] Robert M Hierons[c] Michael R Laurence[a]

[a]*Corresponding author: Mike Laurence,* **email** *m.laurence@gold.ac.uk,* **tel** *+44 (0) 20 7919 7091,* **fax** *+44 (0) 20 7919 7853,* **address** *Department of Computing, Goldsmiths College, University of London, London SE14 6NW, UK.*

[b] *Department of Computing, Goldsmiths College, University of London, London SE14 6NW, UK.*

[c]*Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH.*

Abstract

A program schema defines a class of programs, all of which have identical statement structure, but whose functions and predicates may differ. A schema thus defines an entire class of programs according to how its symbols are interpreted. Two schemas are *strongly equivalent* if they always define the same function from initial states to final states for every interpretation. A *subschema* of a schema is obtained from a schema by deleting some of its statements. A schema S is *liberal* if there exists an initial state in the Herbrand domain such that the same term is not generated more than once along any executable path through S. In this paper we introduce near-liberal schemas, in which this non-repeating condition applies only to terms not having the form $g()$ for a constant function symbol g. Given a schema S that is linear (no function or predicate symbol occurs more than once in S) and a variable v, we compute a set of function and predicate symbols in S which is a subset of those defined by Weiser's slicing algorithm and prove that if for every while predicate q in S and every constant assignment $w := g()$; lying in the body of q, no other assignment to w also lies in the body of q, our smaller symbol set defines a correct subschema of S with respect to the final value of v after execution. We also prove that if S is also free (every path through S is executable) and near-liberal, it is decidable which of its subschemas are strongly equivalent to S. For the class of pairs of schemas in which one schema is a subschema of the other, this generalises a recent result in which S was required to be linear, free and liberal.

```
u := h();
if p(u)    then  v := f(u);
           else  v := g();
```

Fig. 1. Schema S

Key words: free and liberal program schemas, Herbrand domain, program slicing, linear schemas, Weiser's algorithm

1 Introduction

A schema represents the statement structure of a program by replacing real functions and predicates by symbols representing them. A schema, S, thus defines a whole class of programs which all have the same structure. Each program can be obtained from S via a domain D and an *interpretation* i which defines a function $f^i : D^n \to D$ for each function symbol f of arity n, and a predicate function $p^i : D^m \to \{\mathsf{T}, \mathsf{F}\}$ for each predicate symbol p of arity m. As an example, Figure 1 gives a schema S, and the program P of Figure 2 is defined from S by interpreting the function symbols f, g, h and the predicate symbol p as given by P, with D being the set of integers. The subject of schema theory is connected with that of program transformation and was originally motivated by the wish to compile programs effectively[1]. In this paper we are concerned with the relevance of schema theory to program slicing; that is, the study of the effect on a program's rum-time behaviour caused by the deletion of code from the program. Since program slicing algorithms do not normally take into account the meanings of the functions and predicates of a program, a schema encodes all the information about any program which it defines that is available to slicing algorithms.

In this paper we are concerned with three binary relations on schemas, for a variable v.

- Two schemas S, T are *strongly v-equivalent* if for every interpretation, the programs defined by S and T define the same function from initial states to the final value of v. Here non-termination is treated as being a possible final final value of v.
- Two schemas S, T are *weakly v-equivalent* if for every interpretation, the programs defined by S and T define the same function from initial states to the final value of v, when the initial state set is restricted to those states for which the two programs both terminate.
- (v-slices of a schema for variable v.) A *subschema* of a schema S is defined to be any schema obtained by deleting statements from S. For any subschema T of a schema S, and any variable v, we say that T is a v-slice of S if for every interpretation i and any initial state, the program defined by T always terminates if that defined by S does (but not necessarily conversely), in which case the final value of v is the

$$
\begin{array}{l}
u := 1; \\
\mathit{if}\ u > 1 \quad \mathit{then}\ \ v := u + 1; \\
\qquad\qquad\quad \mathit{else}\ \ \ v := 2;
\end{array}
$$

Fig. 2. Program P

same for both programs.

We are interested in obtaining decidable syntactic conditions on schema pairs which imply one of these relations. Given a variable v and a schema S that is assumed to be *linear* (that is, no function or predicate symbol occurs more than once in it), Weiser's slicing algorithm[2,3] defines a set $\mathcal{N}_S(v)$ of symbols[1] occurring in S. This set is defined using the data dependence $\underset{S}{\rightsquigarrow}$, $\rightsquigarrow_S^{\text{final}}$ and control dependence $\searrow_S$ relations. These are defined as follows; $f \underset{S}{\rightsquigarrow} g$ holds if there is a path from the function symbol f to g which does not pass through any assignment to the variable assigned by f, and the function symbol g references this variable, and $f \rightsquigarrow_S^{\text{final}} v$ is defined analogously with respect to a variable v evaluated at the end of a path; for example, in the schema of Fig. 1, $h \underset{S}{\rightsquigarrow} f$, $h \underset{S}{\rightsquigarrow} p$, $h \rightsquigarrow_S^{\text{final}} u$, $g \rightsquigarrow_S^{\text{final}} v$ and $f \rightsquigarrow_S^{\text{final}} v$ hold. We write $p \searrow_S x$ if the predicate p contains the symbol x in its body or in its *if* or *else* parts.

For a variable v and a linear schema S, $\mathcal{N}_S(v)$ is the minimal set of symbols that is left-closed under the $\searrow_S$ and $\underset{S}{\rightsquigarrow}$ relations and contains every function symbol f for which $f \overset{\text{final}}{\underset{S}{\rightsquigarrow}} v$ holds. The authors have proved that a subschema T of S that contains all symbols in $\mathcal{N}_S(v)$ is weakly v-equivalent to S, and if T contains *only* these symbols, then T is a v-slice of S [4,7]. An analogous set $\mathcal{N}_S(\omega)$ can be defined using the while predicates of S instead of the variable v as a starting point in the recursive definition [5, Definition 19]. A subschema of S that contains all the symbols in this set terminates for precisely the same set of pairs of interpretations and initial states as S[5]. In this paper we define subsets $\mathcal{W}funcs_S(v) \subseteq \mathcal{N}_S(v)$ and $\mathcal{W}preds_S(v) \subseteq \mathcal{N}_S(v)$ of the function and predicate symbols respectively in S, which we call the reduced Weiser sets.

We prove three main results involving the equivalence and slicing relations listed above. In each case, we consider a schema S which is linear, and we also require that for every while predicate q in S and every constant assignment $w := g()$; lying in the body of q, no other assignment to w also lies in the body of q. Under these conditions, for any variable v, we prove the following.

(1) *(weak equivalence.)* We show that if T is a subschema of S containing the symbols in $\mathcal{W}funcs_S(v) \cup \mathcal{W}preds_S(v)$, then S and T are weakly v-equivalent.

[1] A *symbol* in this paper means a function or predicate symbol in a schema.

(2) *(weak equivalence plus termination preservation.)* We analogously define sets $\mathcal{W}funcs_S(p)$ and $\mathcal{W}preds_S(p)$ for a predicate symbol p guarding a while statement, and prove that if a subschema T of S contains symbols in $\mathcal{W}funcs_S(p) \cup \mathcal{W}preds_S(p)$ for every such predicate p, and also contains the symbols in $\mathcal{W}funcs_S(v) \cup \mathcal{W}preds_S(v)$, then besides satisfying weak equivalence, T is a v-slice of S.

(3) *(strong equivalence.)* Suppose that in addition to the linearity condition and the condition on constant assignments given above, S satisfies the following; given any path through a schema S, there is an interpretation and an initial state such that the program thus defined follows this path when executed (the freeness condition) and no term apart from terms having the form $g()$ for a constant function symbol g is generated more than once as it does so (the near-liberality condition). The freeness condition was first defined by Paterson [6]. The near-liberality condition, which we introduce in this paper, is a generalisation of liberality[6], in which the non-repeating condition applies to all terms without restriction. Under these hypotheses, we prove that if T is a subschema of S, then S and T are strongly v-equivalent if and only if T contains every symbol in $\mathcal{W}funcs_S(u) \cup \mathcal{W}preds_S(u)$ for each $u \in \{v\} \cup \{p| \, p$ is a while predicate in $S\}$. In particular, it is decidable whether S and T are strongly v-equivalent under these extra conditions on S.

Since no free liberal schema can contain a constant assignment lying in the body of a while predicate, all linear, free and liberal schemas lie in the larger class of schemas to which Result (3) applies. Figure 3 gives an example of a schema lying in this larger class, but which is not liberal, since any path passing more than once through the assignment $v := g_1()$ clearly assigns the same value to v on each occasion. Hence, for schema pairs in which one schema is a subschema of the other, Result (3) is in effect a strengthening of a result in [7,4], in which strong equivalence was shown to be decidable for pairs of schemas which were required to be linear, free and liberal.

As we will show, if S is the schema in Fig. 3, then $f \notin \mathcal{W}funcs_S(v)$ and thus Result (1) is a strengthening of the weak equivalence result in [4,7]. Fig. 5, discussed in Section 9, gives an example of a linear schema that is neither free nor liberal, but satisfies our conditions on assignments of arity zero.

1.1 Relevance of Schema Theory to Program Slicing

The field of (static) program slicing is largely concerned with the design of algorithms which when given a program, eliminate as much code as possible from the program, such that the subprogram consisting of the remaining code, when executed from the same initial state, will preserve some of the behaviour of the original program. One algorithm is thus better than another if it constructs a smaller subprogram for a given program. (For a fuller discussion of program slicing algorithms see [8,9].)

```
while $q(w)$ do {
    $w := h_1(w)$;
    $u := h_2(u)$;
    if $p(u)$ then {
        $v := g_1()$;
        $u := f(u)$;
    }
}
```

Fig. 3. Deleting the assignment $u := f(u)$; does not change the final value of v or prevent termination of any program representable by this free near-liberal linear schema, although f lies in Weiser's set $\mathcal{N}_S(v)$

The simplest form of behaviour-preservation is defined by the final value of a variable, which must be the same for the subprogram as for the program. In addition, a subprogram is normally required to terminate under all inputs for which the original program terminates, thus motivating our v-slice definition in the Introduction.

Most program slicing algorithms, when applied to programs without procedures of the kind considered in this paper, use Weiser's algorithm[2], which, given a program, computes the subprogram containing those symbols defined by the transitive closure of the control and backward data dependence relations.

Thus Weiser's algorithm does not take account of the meanings of the functions and predicates occurring in a program, nor does it exploit the knowledge that the same function or predicate occurs in two different places in a program. In effect, therefore, it takes a linear schema S defined by a program P as input, and computes a subprogram of S which satisfies the required semantic behaviour for *all* interpretations; not solely the interpretation which defines P from S. This reflects the fact that it is undecidable whether the deletion of a particular line of code from a *program* can affect the final value of a given variable after execution (otherwise the halting problem for Turing machines would be decidable) and hence no slicing algorithm can guarantee to give a minimal correct subprogram for every program.

However, slicing algorithms taking linear schemas as input may yield more information about a program than algorithms that merely use Weiser's algorithm. As an example, in the schema S of Figure 3, which will be discussed in further sections, it can be

```
$v := g()$;
if $p(u)$    then $v := g()$;
```

Fig. 4. Deleting the if statement gives a v-slice of this schema

seen that the subschema of S obtained by deleting the assignment with symbol f is a v-slice of S, since the removal of this assignment cannot prevent termination (which is determined solely by the value of w when referenced by q), nor can it prevent the path of execution from passing through g_1 at least once, though it may affect the number of times this happens. However, if the assignment with symbol g_1 is replaced by an assignment $v := g_2(v)$; to give a schema T, then the assignment $u := f(u)$; may not similarly be deleted from T, since this deletion may change the value of v after execution. As an example of an interpretation under which this occurs, suppose that h_1, h_2, f and g_2 are all interpreted as the function $v \mapsto v + 1$ in the domain of integers and $q(0), q(1), p(0), p(1)$ and $p(2)$ map to *true*, whereas $q(v)$ and $p(v)$ map to *false* if $v \geq 2$ or $v \geq 3$ respectively. Execution of S from the initial state in which all variables are set to zero results in a final value of 1 for v, whereas if the assignment $u := f(u)$; is deleted, then the execution path will pass through g_2 on both occassions that it enters the body of q giving a final value of 3 for v. However Weiser's algorithm will treat these two cases identically, and will require f to be in a v-slice in both cases. This is because $f \underset{S}{\rightsquigarrow} p$ and $f \underset{T}{\rightsquigarrow} p$ and $p \searrow_S g_1$ and $p \searrow_T g_2$ hold, and thus $f \in \mathcal{N}_S(v) = \mathcal{N}_T(v)$ follows.

Danicic [10] gives other examples of cases of linear schemas for which program slicing algorithms will not give minimal correct subschemas. If the linearity assumption is discarded, then non-minimality can be demonstrated even for loop-free schemas, such as the one in Figure 4, in which p and both occurrences of g lie in the Weiser symbol set defined by v, but the p-statement can clearly be deleted without changing the final value of v. These examples motivate the mathematical study of schemas, which may lead to the computation of smaller subschemas than conventional program slicing techniques can achieve.

1.2 Organisation of the paper

In the remainder of this section, we explain how the field of program slicing provides motivation for our results, and we also discuss the history of the study of schemas. In Section 2, we give formally our basic schema definitions. In Section 3, we give the formal definition of a subschema of a schema and the semantic definitions of schema equivalence and a v-slice for variable v. In Section 4 we formally define the data dependence relations $\underset{S}{\rightsquigarrow}$ and $\overset{\text{final}}{\underset{S}{\rightsquigarrow}}$ for a schema S. In Section 5, we define the notion of a p-couple for a predicate p; that is, a pair of interpretations which differ only on one p-predicate term. In Section 6, we define formally the classes of free, liberal and near-liberal schemas, and prove that it is decidable whether a linear schema is both free and near-liberal given that it satisfies the additional condition involving while predicates and constant assignments required for the main results of this paper. In Section 7, we give the formal definition of the reduced Weiser set of symbols, and prove that it is decidable whether a given symbol in a linear schema lies in this set. In Section 8 we obtain preliminary results in order to prove our main theorems, which are proved in Section 9. In Section 10, we discuss our conclusions.

1.3 Different classes of schemas

Many subclasses of schemas have been defined:

Structured schemas, in which *goto* statements are forbidden, and thus loops must be constructed using while statements. *All schemas considered in this paper are structured.*
Linear schemas, in which each function and predicate symbol occurs at most once.
Free schemas, where all paths are executable under some interpretation.
Conservative schemas, in which every assignment is of the form $v := f(v_1, \ldots, v_r)$; where $v \in \{v_1, \ldots, v_r\}$.
Liberal schemas, in which two assignments along any executable path can always be made to assign distinct values to their respective variables by a suitable choice of interpretation and initial state.

We now give examples of schemas satisfying these definitions, and first show that the freeness and liberality conditions on schemas are incomparable. To see this, consider the following two examples of linear schemas. The schema

$$\textit{while}\, p(v)\ \textit{do skip}$$

contains no assignments and is therefore liberal, but it is not free, since there is no choice of interpretation and initial state under which the executed path thus defined passes exactly once through the body of p, since the value of v, and hence the boolean value defined at p cannot change during execution. On the other hand the schema

```
while q(w) do {
                 w := f(w);
                 x := g();
               }
```

is free, since if f defines the function $w \mapsto w + 1$ over the domain of integers, then w never defines a repeated value when referenced by q, and so q can be interpreted so as to define an executed path that passes any desired number of times through q, but it is not liberal, since the variable x is always assigned the same value at occurrences of g along any executed path. The subschema obtained from it by deleting the assignment $x := g()$; (that is, $\textit{while}\, q(w)\ \textit{do}\ w := f(w);$) is both free and liberal, on the other hand. More generally, it can be shown that all conservative schemas are liberal.

The schema in Figure 3 can also be seen to be free, owing to the conservative (self-referencing) assignments with symbols f, h_1, h_2, which can be interpreted as the function $w \mapsto w+1$ over the domain of integers, thus ensuring that the variables u, w referenced by p and q respectively never repeat in value. It is not liberal however, since it has a path passing more than once through g_1, along which this assignment defines the same value to v on each occasion. More generally, it is easy to see that no schema having a constant assignment in the body of a while predicate can be both free and liberal, since if it is free, then there is an executable path passing twice through this assignment, which clearly assigns the same value to its variable on each occasion.

Paterson [6] gave a proof that it is decidable whether a schema is both liberal and free and since he also gave an algorithm transforming a schema S into a schema T such that T is both liberal and free if and only if S is liberal, it is clearly decidable whether a schema is liberal. It is an open problem whether freeness is decidable for the class of linear schemas. However he also proved, using a reduction from the Post Correspondence Problem, that it is not decidable whether an arbitrary schema is free.

1.4 Previous results on the decidability of strong equivalence between schemas

Most previous research on schemas has focused on strong equivalence, as defined in the introduction. Many authors call strong equivalence simply 'equivalence', as we do in this subsection. All results on the decidability of equivalence of schemas are either negative or confined to very restrictive classes of schemas. In particular Paterson [6] proved that equivalence is undecidable for the class of all (unstructured) schemas. He proved this by showing that the halting problem for Turing machines (which is, of course, undecidable) is reducible to the equivalence problem for the class of all schemas. Ashcroft and Manna showed [11] that an arbitrary schema can be effectively transformed into an equivalent structured schema, provided that statements such as $\mathit{while}\, \neg p(\mathbf{u})\, \mathit{do}\, T$ are permitted; hence Paterson's result shows that any class of schemas for which equivalence can be decided must not contain this class of schemas. Thus in order to achieve positive results on this problem, it is plainly necessary to define the relevant classes of schema with great care.
Positive results on the decidability of equivalence of schemas include the following; in an early result in schema theory, Ianov [12] introduced a restrictive class of schemas, the Ianov schemas, for which equivalence is decidable. This problem was later shown to be co-NP-complete [13,14]. Ianov schemas are monadic (that is, they contain only a *single* variable) and all function symbols are unary; hence Ianov schemas are conservative.

Paterson [6] proved that equivalence is decidable for a class of schemas called *progressive schemas*, in which every assignment references the variable assigned by the previous assignment along every legal path.

Sabelfeld [15] proved that equivalence is decidable for another class of schemas called *through schemas.* A through schema satisfies two conditions: firstly, that on every path from an accessible predicate p to a predicate q which does not pass through another predicate, and every variable x referenced by p, there is a variable referenced by q which defines a term containing the term defined by x, and secondly, distinct variables referenced by a predicate can be made to define distinct terms under some interpretation.

The authors have shown [4,7] that it is decidable whether linear, free, liberal schemas are equivalent.

In view of the evident difficulty of obtaining positive results on this problem, and the importance of program slicing, it seems sensible to concentrate on trying to decide equivalence for classes of schema pairs in which one schema is a subschema of the other, as in this paper.

2 Basic definitions for schemas

Throughout this paper, $\mathcal{F}$, $\mathcal{P}$, and $\mathcal{V}$ denote fixed infinite sets of *function symbols*, *predicate symbols*, and *variables* respectively. We assume a function

$$\mathit{arity} : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}.$$

The arity of a symbol x is the number of arguments referenced by x. Note that in the case when the arity of a function symbol g is zero, g may be thought of as a constant.

The set $\mathit{Term}(\mathcal{F}, \mathcal{V})$ of *terms* is defined as follows:

- each variable is a term,
- if $f \in \mathcal{F}$ is of arity n and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term.

We refer to a tuple $\mathbf{t} = (t_1, \ldots, t_n)$, where each t_i is a term, as a vector term. We call $p(\mathbf{t})$ a predicate term if $p \in \mathcal{P}$ and the number of components of the vector term $\mathbf{t}$ is $\mathit{arity}(p)$.

We also define F-terms and vF-terms recursively for $F \in \mathcal{F}^*$ and $v \in \mathcal{V}$. Any term $f(t_1, \ldots, t_n)$ is an f-term, and the term v is a v-term. If $g \in \mathcal{F}$ and at least one of the terms $t_1, \ldots, t_n$ is an F-term or vF-term, then the term $g(t_1, \ldots, t_n)$ is an Fg-term, or vFg-term, respectively. Thus any FF'-term is also an F'-term.

An an example, $f(g(v))$ for $v \in \mathcal{V}$ is an f-term, a gf-term, and a vgf-term. Note that function symbols in this terminology occur in the order in which they are encountered along a path generating a given term.

Definition 1 (schemas) We define the set of all *schemas* recursively as follows. *skip* is a schema. An assignment $y := f(\mathbf{x})$; where $y \in \mathcal{V}$, $f \in \mathcal{F}$, and $\mathbf{x}$ is a vector of $\mathit{arity}(f)$ variables, is a schema. From these all schemas may be 'built up' from the following constructs on schemas.

sequences; $S' = U_1 U_2 \ldots U_r$ is a schema provided that each U_i for $i \in \{1, \ldots, r\}$ is a schema.

if **schemas;** $S'' = \mathit{if}\, p(\mathbf{x})\, \mathit{then}\, \{T_1\}\, \mathit{else}\, \{T_2\}$ is a schema whenever $p \in \mathcal{P}$, $\mathbf{x}$ is a vector of $\mathit{arity}(p)$ variables, and T_1, T_2 are schemas. We call the schemas T_1 and T_2 the *true* and *false* parts of p.

while **schemas;** $S''' = \mathit{while}\, q(\mathbf{y})\, \mathit{do}\, \{T\}$ is a schema whenever $q \in \mathcal{P}$, $\mathbf{y}$ is a vector of $\mathit{arity}(q)$ variables, and T is a schema. We call T the *body* of the *while* predicate q in S'''.

Thus a schema is a word in a language over an infinite alphabet. We normally omit the braces { and } if this causes no ambiguity. Also, we may write $\mathit{if}\, p(\mathbf{x})\, \mathit{then}\, \{T_1\}$ instead of
$\mathit{if}\, p(\mathbf{x})\, \mathit{then}\, \{T_1\}\, \mathit{else}\, \{T_2\}$ if $T_2 = \mathit{skip}$.

We refer to elements of $\mathcal{F} \cup \mathcal{P}$ as *symbols.* If no symbol appears more than once in a schema S, then S is said to be *linear.*

We define $\mathit{Funcs}(S)$, $\mathit{Preds}(S)$ and $\mathit{Symbols}(S) = \mathit{Funcs}(S) \cup \mathit{Preds}(S)$ to be the sets of function symbols, predicate symbols and all symbols occurring in a schema S. If S is linear, we define $\mathit{ifPreds}(S)$ and $\mathit{whilePreds}(S)$ to be the sets of if predicate symbols and while predicate symbols in S.

A schema without predicates (that is, a schema which consists of a sequence of assignments and *skip*s) is called *predicate-free.*

If a linear schema S contains an assignment $y := f(\mathbf{x})$; then we define $y = \mathit{assign}_S(f)$ and $\mathbf{x} = \mathbf{refvec}_S(f)$. If $p \in \mathit{Preds}(S)$ then $\mathbf{refvec}_S(p)$ is defined similarly. We also define $\mathit{refVars}_S(x)$ for a symbol x in S to be the set of variables occurring in $\mathbf{refvec}_S(x)$.

Definition 2 (the $\searrow_S$ relation)
Let S be a schema. If p is a predicate in S and x is any symbol, we say that $p \searrow_S x$ holds if x occurs in the body of an occurrence of p (if p is a while predicate in S) or x lies in the true or false part of p (if p is an if predicate). We may strengthen this by writing $p \searrow_S x\,(Z)$ for $Z \in \{\mathsf{T}, \mathsf{F}\}$ to indicate the additional condition that x lies in the Z-part of p if $p \in \mathit{ifPreds}(S)$, or $p \in \mathit{whilePreds}(S)$ (if $Z = \mathsf{T}$).

The relation $\searrow_S$ is the transitive closure of the relation 'controls' in program analysis terminology, when applied to structured schemas as in this paper.

2.1 Paths through a Schema

The execution of a program defines a possibly infinite sequence of assignments and predicates. Each such sequence will correspond to a *path* through the associated schema. The set $\Pi^\omega(S)$ of paths through S is now given.

Definition 3 (the set $\Pi^\omega(S)$ of paths through S, path-segments of S) If L is any set, then we write L^* for the set of finite words over L and L^ω for the set containing both finite and infinite words over L. If σ is a word, or a set of words over an alphabet, then $pre(\sigma)$ is the set of all finite prefixes of (elements of) σ.

For each schema S the alphabet of S, written $alphabet(S)$ is the set containing all letters $\underline{y := f(\mathbf{x})}$ such that $y := f(\mathbf{x});$ is an assignment in S and $\underline{p(\mathbf{y}), Z}$ such that $p(\mathbf{y})$ occurs in S and $Z \in \{\mathsf{T}, \mathsf{F}\}$. We define $symbol(\underline{y := f(\mathbf{x})}) = f$ and $symbol(\underline{p(\mathbf{y}), Z}) = p$. We sometimes abbreviate $\underline{p(\mathbf{y}), Z}$ to $\underline{p, Z}$, where the vector $\mathbf{y}$ of variables need not be referred to.

The words in $\Pi(S) \subseteq (alphabet(S))^*$ are formed by concatenation from the words of subschemas of S as follows:

$\Pi(skip)$ is the set containing only the empty word.

For **assignments**, $\Pi(y := f(\mathbf{x});) = \{\underline{y := f(\mathbf{x})}\}$.

For **sequences**, $\Pi(S_1 S_2 \ldots S_r) = \Pi(S_1) \ldots \Pi(S_r)$.

For ***if*** **schemas**, $\Pi(\,if\, p(\mathbf{x})\, then\, \{T_1\}\, else\, \{T_2\})$ is the set of all concatenations of $\underline{p(\mathbf{x}), \mathsf{T}}$ with a word in $\Pi(T_1)$ and all concatenations of $\underline{p(\mathbf{x}), \mathsf{F}}$ with a word in $\Pi(T_2)$.

For ***while*** **schemas**, $\Pi(\,while\, q(\mathbf{y})\, do\, \{T\}) = (\underline{q(\mathbf{y}), \mathsf{T}}\,\Pi(T))^*\underline{q(\mathbf{y}), \mathsf{F}}$.

We define $\Pi^\omega(S) = \{\sigma \in (alphabet(S))^\omega | pre(\sigma) \subseteq pre(\Pi(S))\}$. Prefixes of $\Pi(S)$ are called *path-prefixes* through S. Any $\mu \in alphabet(S)^*$ is a *path-segment* (in S) if there are words μ', μ'' such that $\mu'\mu\mu'' \in \Pi(S)$. A *terminal* path-segment of S is a path-segment ν such that $\mu\nu \in \Pi(S)$ for some μ.

2.2 Semantics of schemas

The symbols upon which schemas are built are given meaning by defining the notions of a state and of an interpretation. It will be assumed that 'values' are given in a single set D, which will be called the *domain*. We are mainly interested in the case in which $D = Term(\mathcal{F}, \mathcal{V})$ (the Herbrand domain) and the function symbols represent the 'natural' functions with respect to $Term(\mathcal{F}, \mathcal{V})$, since our equivalence and semantic slicing definitions can be stated solely with respect to this domain.

Definition 4 (states, (Herbrand) interpretations and the natural state e) Given a domain D, a *state* is either $\bot$ (denoting non-termination) or a function $\mathcal{V} \to D$. If d is a state and $\mathbf{x} = (v_1, \ldots, v_m)$ is a vector of variables, then we define $d(\mathbf{x}) = (d(v_1), \ldots, d(v_m))$. The set of all states with domain D will be denoted by $\mathrm{State}(\mathcal{V}, D)$. An interpretation i defines, for each function symbol $f \in \mathcal{F}$ of arity n, a function $f^i : D^n \to D$, and for each predicate symbol $p \in \mathcal{P}$ of arity m, a function $p^i : D^m \to \{\mathsf{T}, \mathsf{F}\}$. The set of all interpretations with domain D will be denoted $\mathit{Int}(\mathcal{F}, \mathcal{P}, D)$.
We call the set $\mathit{Term}(\mathcal{F}, \mathcal{V})$ of terms the *Herbrand domain*, and we say that a function from $\mathcal{V}$ to $\mathit{Term}(\mathcal{F}, \mathcal{V})$ is a Herbrand state. An interpretation i for the Herbrand domain is said to be *Herbrand* if the functions $f^i : \mathit{Term}(\mathcal{F}, \mathcal{V})^n \to \mathit{Term}(\mathcal{F}, \mathcal{V})$ for each $f \in \mathcal{F}$ are defined as

$$f^i(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

for all n-tuples of terms $(t_1, \ldots, t_n)$.
We define the *natural state* $e : \mathcal{V} \to \mathit{Term}(\mathcal{F}, \mathcal{V})$ by $e(v) = v$ for all $v \in \mathcal{V}$.

Note that an interpretation i being Herbrand places no restriction on the mappings $p^i : (\mathit{Term}(\mathcal{F}, \mathcal{V}))^m \to \{\mathsf{T}, \mathsf{F}\}$ defined by i for each $p \in \mathcal{P}$.

Given a schema S and a domain D, an initial state $d \in \mathrm{State}(\mathcal{V}, D)$ with $d \neq \bot$ and an interpretation $i \in \mathit{Int}(\mathcal{F}, \mathcal{P}, D)$ we now define the final state $\mathcal{M}[\![S]\!]^i_d \in \mathrm{State}(\mathcal{V}, D)$ and the associated path $\pi_S(i, d) \in \Pi^\omega(S)$. In order to do this, we need to define the predicate-free schema associated with a path-prefix by considering the sequence of assignments through which it passes.

Definition 5 (the schema $\mathit{schema}(\sigma)$)
Given a word $\sigma \in (\mathit{alphabet}(S))^*$ for a schema S, we recursively define the predicate-free schema $\mathit{schema}(\sigma)$ by the following rules; $\mathit{schema}(\lambda) = \mathit{skip}$ if λ is the empty word, $\mathit{schema}(\sigma \underline{v := f(\mathbf{x})}) = \mathit{schema}(\sigma)\, v := f(\mathbf{x})$; and $\mathit{schema}(\sigma \underline{p(\mathbf{y}), X}) = \mathit{schema}(\sigma)$.

Lemma 6 *Let S be a schema. If $\sigma \in \mathit{pre}(\Pi(S))$, the set $\{m \in \mathit{alphabet}(S) | \sigma m \in \mathit{pre}(\Pi(S))\}$ is one of the following; a singleton containing an underlined assignment, a pair $\{\underline{p(\mathbf{y}), \mathsf{T}}, \underline{p(\mathbf{y}), \mathsf{F}}\}$ where $p \in \mathit{Preds}(S)$, or the empty set, and if $\sigma \in \Pi(S)$ then the last case holds.*

Lemma 6, which can be proved in a similar way to [5, Lemma 6], reflects the fact that at any point in the execution of a program, there is never more than one 'next step' which may be taken, and an element of $\Pi(S)$ cannot be a strict prefix of another. Thus we can define the partial function $\sigma \mapsto \mathit{nextsymbol}_S(\sigma)$ for any $\sigma \in \mathit{pre}(\Pi(S))$.

Definition 7 ($\mathit{nextsymbol}_S(\sigma)$ for a path-prefix σ) Let S be a schema. If $\sigma \in \mathit{pre}(\Pi(S)) - \Pi^\omega(S)$, then $\mathit{nextsymbol}_S(\sigma)$ is the unique element of $\mathit{Funcs}(S) \cup \mathit{Preds}(S)$ satisfying $\sigma l \in \mathit{pre}(\Pi(S))$ for $l \in \mathit{alphabet}(S)$ and $\mathit{symbol}(l) = \mathit{nextsymbol}_S(\sigma)$.

Definition 8 (semantics of predicate-free schemas) Given a state $d \neq \bot$, the final state $\mathcal{M}[\![S]\!]^i_d$ and associated path $\pi_S(i,d) \in \Pi^\omega(S)$ of a schema S are defined as follows:

For *skip*,

$$\mathcal{M}[\![\mathit{skip}]\!]^i_d = d$$

and

$\pi_{\mathit{skip}}(i,d)$ is the empty word.

For assignments,

$$\mathcal{M}[\![y := f(\mathbf{x});]\!]^i_d(v) \quad = \quad \begin{cases} d(v) & \text{if } v \neq y, \\ f^i(d(\mathbf{x})) & \text{if } v = y \end{cases}$$

and

$$\pi_{y := f(\mathbf{x});}(i,d) \quad = \quad \underline{y := f(\mathbf{x})},$$

and for sequences $S_1 S_2$ of predicate-free schemas,

$$\mathcal{M}[\![S_1 S_2]\!]^i_d \quad = \quad \mathcal{M}[\![S_2]\!]^i_{\mathcal{M}[\![S_1]\!]^i_d}$$

and

$$\pi_{S_1 S_2}(i,d) \quad = \quad \pi_{S_1}(i,d)\pi_{S_2}(i, \mathcal{M}[\![S_1]\!]^i_d).$$

This uniquely defines $\mathcal{M}[\![S]\!]^i_d$ and $\pi_S(i,d)$ if S is predicate-free.

In order to give the semantics of a general schema S, first the path, $\pi_S(i,d)$, of S with respect to interpretation, i, and initial state d is defined.

Definition 9 (the path $\pi_S(i,d)$) Given a schema S, an interpretation i, and a state, $d \neq \bot$, the path $\pi_S(i,d) \in \Pi^\omega(S)$ is defined by the following condition; for all $\sigma\, \underline{p(\mathbf{y}), X} \in \mathit{pre}(\pi_S(i,d))$, the equality $p^i(\mathcal{M}[\![\mathit{schema}(\sigma)]\!]^i_d(\mathbf{y})) = X$ holds.

In other words, the path $\pi_S(i,d)$ has the following property; if a predicate expression $p(\mathbf{y})$ along $\pi_S(i,d)$ is evaluated with respect to the predicate-free schema consisting of the sequence of assignments preceding that predicate in $\pi_S(i,d)$, then the value in $\{\mathsf{T},\mathsf{F}\}$ of the resulting predicate term given by i 'agrees' with the value given in $\pi_S(i,d)$.

By Lemma 6, this defines the path $\pi_S(i,d) \in \Pi^\omega(S)$ uniquely.

Definition 10 (the semantics of arbitrary schemas) If $\pi_S(i,d)$ is finite, we define

$$\mathcal{M}[\![S]\!]^i_d = \mathcal{M}[\![\mathit{schema}(\pi_S(i,d))]\!]^i_d$$

(which is already defined, since $\mathit{schema}(\pi_S(i,d))$ is predicate-free) otherwise $\pi_S(i,d)$ is infinite and we define $\mathcal{M}[\![S]\!]^i_d = \bot$. In this last case we may say that $\mathcal{M}[\![S]\!]^i_d$ is not terminating. Also, for schemas S, T and interpretations i and j we write $\mathcal{M}[\![S]\!]^i_d(\omega) = \mathcal{M}[\![T]\!]^j_d(\omega)$ to mean $\mathcal{M}[\![S]\!]^i_d = \perp \iff \mathcal{M}[\![T]\!]^j_d = \perp$. For convenience, if S is predicate-free and $d : \mathcal{V} \to \mathit{Term}(\mathcal{F}, \mathcal{V})$ is a state then we define unambiguously $\mathcal{M}[\![S]\!]_d = \mathcal{M}[\![S]\!]^i_d$; that is, we assume that the interpretation i is Herbrand if d is a Herbrand state; and we will write $\mathcal{M}[\![\mu]\!]_d$ to mean $\mathcal{M}[\![\mathit{schema}(\mu)]\!]_d$ for any $\mu \in \mathit{alphabet}(S)^*$.

Observe that $\mathcal{M}[\![S_1S_2]\!]^i_d = \mathcal{M}[\![S_2]\!]^i_{\mathcal{M}[\![S_1]\!]^i_d}$ and

$$\pi_{S_1S_2}(i, d) = \pi_{S_1}(i, d)\pi_{S_2}(i, \mathcal{M}[\![S_1]\!]^i_d)$$

hold for all schemas (not just predicate-free ones).

Given a schema S, let $\mu \in \mathit{pre}(\Pi(S))$. We say that μ passes through a predicate term $p(\mathbf{t})$ if μ has a prefix μ' ending in $\underline{p(\mathbf{y}), Y}$ for $Y \in \{\mathsf{T}, \mathsf{F}\}$ such that $\mathcal{M}[\![\mu']\!]_e(\mathbf{y}) = \mathbf{t}$ holds. We say that $p(\mathbf{t}) = Y$ is a *consequence* of μ in this case. As an example of this usage, if μ is the terminating path through the schema of Fig. 3 which passes exactly twice through the body of q, then $q(w) = \mathsf{T}$, $q(h_1(w)) = \mathsf{T}$ and $q(h_1(h_1(w))) = \mathsf{F}$ are all consequences of μ.

3 Subschemas of schemas, the semantic slicing criterion and the equivalence condition

We now formalise the notion of a subschema.

Definition 11 (subschemas of a schema) The set of subschemas of a schema S is the minimal set of schemas which satisfies the following rules;

- Every schema is a subschema of itself.
- *skip* is a subschema of any schema.
- $S_1 \ldots S_{m-1}S_{m+1} \ldots S_n$ is a subschema of $S_1 \ldots S_n$.
- If S'_m is a subschema of S_m, then $S_1 \ldots S'_m \ldots S_n$ is a subschema of $S_1 \ldots S_m \ldots S_n$.
- if T' is a subschema of T then *while* $p(\mathbf{u})$ *do* T' is a subschema of *while* $p(\mathbf{u})$ *do* T;
- if T' is a subschema of T then the if schema *if* $q(\mathbf{u})$ *then* S *else* T' is a subschema of *if* $q(\mathbf{u})$ *then* S *else* T (the true and false parts may be interchanged in this example);
- a subschema of a subschema of S is itself a subschema of S.

In order to present our main results, it is useful to define two types of equivalence between schemas; strong equivalence, which some authors refer to as simply equivalence, and weak equivalence, in which non-termination is excluded from consideration as a final state. In addition, we restrict consideration to the final value of a single variable.

Definition 12 (strong and weak u-equivalence for $u \in \mathcal{V}$) Let $u \in \mathcal{V}$ and let S, T be schemas. If $\mathcal{M}[\![S]\!]^i_d(u) = \mathcal{M}[\![T]\!]^i_d(u)$ always holds for any state d over any domain and any interpretation i with respect to that domain, then we say that S and T are strongly u-equivalent. If $\mathcal{M}[\![S]\!]_d^i(u) = \mathcal{M}[\![T]\!]_d^i(u)$ always holds when neither side is $\perp$, we say that S and T are weakly u-equivalent. If $\mathcal{M}[\![S]\!]_d^i(u) = \perp \iff \mathcal{M}[\![T]\!]_d^i(u) = \perp$ always holds, then we say that S and T are ω-equivalent.

Clearly any schemas S and T are strongly u-equivalent if and only if they are both weakly u-equivalent and ω-equivalent.

Definition 13 is of more relevance to program slicing than that of either form of equivalence, since the behaviour of a subprogram is not usually of interest in cases in which the original program fails to terminate.

Definition 13 (the semantic u-slice condition for $u \in \mathcal{V}$) Let T be a subschema of a schema S. Then given $u \in \mathcal{V}$, we say that T is a u-slice of S if given any domain D, any state $d : \mathcal{V} \to D$ and any $i \in \mathit{Int}(\mathcal{F}, \mathcal{P}, D)$, $\mathcal{M}[\![S]\!]_d^i \neq \perp \Rightarrow \mathcal{M}[\![S]\!]_d^i(u) = \mathcal{M}[\![T]\!]_d^i(u)$ holds.

Clearly every u-slice of a schema is weakly u-equivalent to it. As an example of these relations, let S be the schema

$$u := g();$$
$$\mathit{while}\, p(v)\ \mathit{do}\ v := f(v);$$

The schema $u := g()$; is a u-slice of S and the two schemas are therefore weakly u-equivalent, but they are not strongly u-equivalent, since there exists an interpretation and an initial state for which S fails to terminate but its subschema $u := g()$; clearly does; for example, this holds for any interpretation under which $p(v)$ always maps to T.

These equivalence and slicing conditions are stated in terms of every conceivable domain and initial state; however it is well known that the Herbrand domain is the only one that needs to be considered when considering many schema problems. Theorem 14, which is virtually a restatement of [16, Theorem 4-1], ensures that for slicing and equivalence purposes, we only need to consider Herbrand interpretations and the natural state e.

Theorem 14 *Let χ be a set of schemas, let D be a domain, let d be a function from the set of variables into D and let i be an interpretation using this domain. Then there is a Herbrand interpretation j such that the following hold.*

(1) *For all $S \in \chi$, the path $\pi_S(j, e) = \pi_S(i, d)$.*
(2) *If $S_1, S_2 \in \chi$ and v_1, v_2 are variables and $\rho_k \in \mathit{pre}(\pi_{S_k}(j, e))$ for $k = 1, 2$ and $\mathcal{M}[\![\rho_1]\!]_e(v_1) = \mathcal{M}[\![\rho_2]\!]_e(v_2)$, then also $\mathcal{M}[\![\rho_1]\!]_d^i(v_1) = \mathcal{M}[\![\rho_2]\!]_d^i(v_2)$ holds.*

As a consequence of Theorem 14, $D = \mathit{Term}(\mathcal{F}, \mathcal{V})$ and $d = e$ may be assumed in Definitions 12 and 13. Therefore, throughout the remainder of the paper, all interpretations will be assumed to be Herbrand.

4 The data dependence relations $\underset{S}{\rightsquigarrow}$ and $\overset{\textbf{final}}{\underset{S}{\rightsquigarrow}}$

Definition 15 formalises the data dependence relations between symbols and variables in a linear schema.

Definition 15 (the $\underset{S}{\rightsquigarrow}$ and $\overset{\textbf{final}}{\underset{S}{\rightsquigarrow}}$ relations and parameterised path-segments) Let S be a linear schema and let σ be a path-segment in S.

- We call σ an F-path-segment, or vF-path-segment for $F \in \mathcal{F}^*$ and $v \in \mathcal{V}$ if $\mathcal{M}[\![\sigma]\!]_e(u)$ for some $u \in \mathcal{V}$ is an F-term, or vF-term, respectively. We also call these path-segments an Fu-path-segment or vFu-path-segment respectively.
- We call $\underline{\sigma p, Z}$ an Fp-path-segment or Fp-path-segment in S if $\mathcal{M}[\![\sigma]\!]_e(u)$ is an F-term for some $u \in \mathcal{V}$ referenced by p in S. We define vFp-path-segments analogously.
- We write $f \underset{S}{\rightsquigarrow} g$ if S contains an fg-path-segment for $f \in \mathcal{F}$ and $g \in \mathcal{F} \cup \mathcal{P}$, and write $f \overset{\text{final}}{\underset{S}{\rightsquigarrow}} u$ if S contains a terminal path-segment which is an fu-path-segment for $u \in \mathcal{V}$.

5 Couples of interpretations

In order to establish which predicate symbols of a schema must be included in a subschema in order to preserve our desired behaviour, we define the notion of a p-couple for a predicate p. This is simply a pair of (Herbrand) interpretations which differ at exactly one predicate term. The motivation for Definition 16 is as follows; if the final value of a variable v with respect to a schema S differs for each element of a p-couple then this means that the predicate p must influence the final value of v and so must be kept in any subschema of S that preserves it. Thus, p-couples are used to reason about the set of predicates required to lie in a subschema.

Definition 16 (couples) Let i, j be interpretations and let $p \in \mathcal{P}$. We say that the set $\{i, j\}$ is a p-couple if there is a vector term $\mathbf{t}$ such that $p^i(\mathbf{t}) \neq p^j(\mathbf{t})$, and i and j agree at all other predicate terms. In this case we may also say that $\{i, j\}$ is a $p(\mathbf{t})$-couple. If a component of $\mathbf{t}$ is an F-term for $F \in \mathcal{F}^*$, then $\{i, j\}$ is an Fp-couple. Given any $u \in \mathcal{V}$ and schema S, we also say that $\{i, j\}$ is an Fpu-couple or $p(\mathbf{t})u$-couple for S if also $\mathcal{M}[\![S]\!]^i_e(u) \neq \mathcal{M}[\![S]\!]^j_e(u)$ and both sides terminate.

We also make analogous definitions if instead $u = \omega$; we say $\{i, j\}$ is a $p\omega$-couple for S if exactly one path in $\{\pi_S(i, e), \pi_S(j, e)\}$ terminates.

Note that a *pu*-couple is simply an Fpu-couple with F as the empty word. The existence of a *pu*-couple for a schema S 'witnesses' the fact that p affects the semantics of S, as defined by u.

Proposition 17 follows immediately from Definition 16.

Proposition 17 *If $u \in \mathcal{V} \cup \{\omega\}$ and schemas S, T are strongly u-equivalent (or u-equivalent if $u = \omega$) then a pu-couple for S is also a pu-couple for T.* $\square$

Definition 18 (head and tails of a couple) Let S be a schema. Let $u \in \mathcal{V} \cup \{\omega\}$, and let $q \in \mathit{Preds}(S)$. Let $I = \{i, j\}$ be a qu-couple for S and write

$$\pi_S(k, e) = \mu \underline{q, Z_k}\ \rho_k$$

for each $k \in I$ and $\{Z_i, Z_j\} = \{\mathsf{T}, \mathsf{F}\}$; that is, μ is the maximal common prefix of the paths $\pi_S(k, e)$. Then we define $\mathit{tail}_S(k, I) = \rho_k$ for each $k \in I$, and $\mu = \mathit{head}_S(I)$.

Observe that Definition 18 is given in terms of the natural state e. The motivation for Definition 18 is given by Lemma 23, which shows that given a *pu*-couple for a free near-liberal schema, under certain conditions a new *pu*-couple may be obtained from it by replacing its head by any path-prefix leading to p, while keeping the same tails.

For the remainder of this paper, we use the following terminology with interpretations. If i is an interpretation, $p(\mathbf{t})$ is a predicate term and $X \in \{\mathsf{T}, \mathsf{F}\}$, then $i(p(\mathbf{t}) = X)$ is the interpretation which maps every predicate term to the same value as i except $p(\mathbf{t})$, which it maps to X.

6 Free, liberal and near-liberal schemas

We now state formally the definitions of freeness and liberality mentioned in Section 1.3, and define the new near-liberality condition.

Definition 19 (free and liberal schemas) Let S be a schema.

- If for every $\sigma \in \mathit{pre}(\Pi(S))$ there is a Herbrand interpretation i such that $\sigma \in \mathit{pre}(\pi_S(i, e))$, then S is said to be *free.*
- If for every Herbrand interpretation i and any path-prefix $\mu\ \underline{v := f(\mathbf{a})}\ \nu\ \underline{w := g(\mathbf{b})} \in \mathit{pre}(\pi_S(i, e))$, we have

  $$\mathcal{M}[\![\mu\ \underline{v := f(\mathbf{a})}]\!]_e(v) \neq \mathcal{M}[\![\mu\ \underline{v := f(\mathbf{a})}\ \nu\ \underline{w := g(\mathbf{b})}]\!]_e(w),$$

  then S is said to be *liberal.* (If $f \neq g$ then of course this condition is trivially satisfied.)

Thus a schema S is free if for every path through S, there is a Herbrand interpretation which follows it with the natural state e as the initial state, or, equivalently, if on every path through S, the same predicate term is not generated more than once given e as the initial state; and a schema S is liberal if given any path through S passing through two assignments and a Herbrand interpretation which follows it with e as the initial state, the assignments give distinct values to the variables to which they assign. The definitions of freeness and liberality were first given in [6].

In this paper we weaken the definition of liberality by only requiring it to apply to assignments that are not of the form $v := g()$; for any constant $g \in \mathcal{F}$.

Definition 20 (near-liberal schemas) Let S be a schema. We say that S is near-liberal if for every Herbrand interpretation i, every $f \in \mathit{Funcs}(S)$ such that $\mathit{arity}(f) > 0$ and any path-prefix $\mu\ \underline{v := f(\mathbf{a})}\ \nu\ \underline{w := f(\mathbf{b})} \in \mathit{pre}(\pi_S(i, e))$, we have

$$\mathcal{M}[\![\mathit{schema}(\mu\ \underline{v := f(\mathbf{a})})]\!]_e(v) \neq \mathcal{M}[\![\mathit{schema}(\mu\ \underline{v := f(\mathbf{a})}\ \nu\ \underline{w := f(\mathbf{b})})]\!]_e(w).$$

Theorem 21 shows that it is decidable whether a schema lies in the class of schemas considered in this paper, and as a consequence of this theorem, the linear schema of Figure 3 is both free and near-liberal.

Theorem 21 *Let Ω be the set of all linear schemas S such that for all assignments $w := g()$; lying in the body of a while predicate r in S, no other assignment to w also lies in the body of r. Let $S \in \Omega$ and let F be the set of non-constant function symbols in S. Consider the following assertions about S.*

(1) *S is both free and near-liberal.*
(2) *For every path-segment $l\nu l$ through S such that $l \in \mathit{alphabet}(S)$ and $\mathit{symbol}(l) \in \mathcal{P} \cup F$ holds, and ν does not pass more than once through any letter $\underline{p, \mathsf{T}}$ for $p \in \mathit{whilePreds}(S)$, there is a variable v referenced by $\mathit{symbol}(l)$ such that the last assignment to v on $l\nu$ exists and is non-constant.*

Then (1) $\iff$ (2) holds. In particular, it is decidable whether a schema in Ω is free and near-liberal.

Proof. If (1) holds, then (2) must hold, since if there exists a path-segment $l\nu l$ for which (2) is false, then the same term or predicate term is defined at the two occurrences of l after any path-prefix $\mu\nu l\nu l$ through S, contradicting (1). Conversely, assume (2) holds. Before proving (1), we first prove that the conclusion stated for $l\nu$ in (2) holds for all path-segments $l\nu$, without assuming the restriction given on the number of times ν passes through any while predicate. Assume this is false for some path-segment $l\nu l$, with $|\nu|$ minimal; then since (2) as written is assumed to hold, ν must pass more than once through a letter $\underline{p, \mathsf{T}}$ for some $p \in \mathit{whilePreds}(S)$. Write $\nu = \nu_1\underline{p, \mathsf{T}}\nu_2\underline{p, \mathsf{T}}\nu_3$. By considering the path-segment $\nu_1\underline{p, \mathsf{T}}\nu_3$ and using our minimality hypothesis, we infer that either $l\nu_1$ or ν_3 passes through a non-constant assignment to a variable v

referenced by l, and hence so does ν. From the definition of $\Pi(S)$, the linearity of S, and the existence of the path-segment $l\nu l$, ν lies entirely in the body of a while predicate, and so from by our assumption on while predicates in S, no non-constant assignment to v on ν is later 'killed' along ν by a constant assignment to v, contradicting the assumption on $l\nu l$.

Now assume that (1) is false. Thus there exists a path-prefix $\mu l\nu l$ through S such that either $\mathit{symbol}(l) \in F$ (if S is not near-liberal) or $\mathit{symbol}(l) \in \mathcal{P}$ (if S is not free), and the same term or predicate term is defined at the two occurrences of l. Let μ be of minimal length with this property. We now know that $l\nu$ passes through a non-constant assignment to a variable referenced by l, and this assignment defines a term occurring in the (predicate) term defined at each occurrence of l, and thus μ also passes through this assignment, and the two occurrences of the assignment define the same term, contradicting the minimality condition on μ.

The decidability conclusion follows from the fact that for any linear schema S, only finitely many path-segments pass not more than once through any letter $\underline{p, \mathsf{T}}$ for $p \in \mathit{whilePreds}(S)$, and the set of such path-segments can be computed. □

Theorem 21 can almost certainly be strengthened by allowing Ω to contain all linear, free, near-liberal schemas, but the proof in this more general case would be longer.

The significance of the near-liberality condition is given by Lemma 22, which will be used to prove Lemma 23.

Lemma 22 *Let $S, \bar{S}, T_1, T_2$ be predicate-free schemas and assume that the following hold.*

- *For each $i \in \{1, 2\}$, both schemas ST_i and $\bar{S}T_i$ are near-liberal.*
- *For all $w \in \mathcal{V}$ and $g \in \mathcal{F}$ of arity zero, if $\mathcal{M}[\![S]\!]_e(w) = g()$ then either also $\mathcal{M}[\![\bar{S}]\!]_e(w) = g()$ or g does not occur in either schema T_i.*

Let $v_1, v_2 \in \mathcal{V}$. If $\mathcal{M}[\![ST_1]\!]_e(v_1) = \mathcal{M}[\![ST_2]\!]_e(v_2)$, then $\mathcal{M}[\![\bar{S}T_1]\!]_e(v_1) = \mathcal{M}[\![\bar{S}T_2]\!]_e(v_2)$ holds.

Proof. Assume $\mathcal{M}[\![ST_1]\!]_e(v_1) = \mathcal{M}[\![ST_2]\!]_e(v_2)$ holds. We will prove $\mathcal{M}[\![\bar{S}T_1]\!]_e(v_1) = \mathcal{M}[\![\bar{S}T_2]\!]_e(v_2)$ by induction on the number of assignments in T_1. The proof proceeds in stages.

- Suppose that neither schema ST_i contains an assignment to the respective variable v_i. Then clearly $v_1 = v_2$ and so $\mathcal{M}[\![\bar{S}T_1]\!]_e(v_1) = \mathcal{M}[\![\bar{S}]\!]_e(v_1) = \mathcal{M}[\![\bar{S}]\!]_e(v_2) = \mathcal{M}[\![\bar{S}T_2]\!]_e(v_2)$ holds.
- Suppose that for exactly one value of i, the schema ST_i contains an assignment to v_i. This contradicts $\mathcal{M}[\![ST_1]\!]_e(v_1) = \mathcal{M}[\![ST_2]\!]_e(v_2)$.

Thus we may assume that both schemas ST_i contain assignments to the respective variables v_i.

- Suppose that the last assignment to v_1 in ST_1 occurs in S. If T_2 does not contain an assignment to v_2, then the conclusion follows immediately. On the other hand, if T_2 does contain an assignment to v_2, then since $\mathcal{M}[\![ST_1]\!]_e(v_1) = \mathcal{M}[\![ST_2]\!]_e(v_2)$ holds, the last such assignment defines the same term in ST_2 as the last assignment to v_1 in ST_1 and hence S. Since ST_2 is near-liberal, these must be constant assignments, and so $\mathcal{M}[\![ST_1]\!]_e(v_1) = \mathcal{M}[\![S]\!]_e(v_1) = \mathcal{M}[\![T_2]\!]_e(v_2)$ is a constant term $g()$. Thus also $\mathcal{M}[\![\bar{S}T_2]\!]_e(v_2) = g()$ holds. By the hypotheses in the original statement of the Lemma, $\mathcal{M}[\![\bar{S}]\!]_e(v_1) = g()$ holds and the conclusion follows.
- Thus we may assume that T_1 and (similarly) T_2 contain assignments to v_1 and v_2 respectively. Let $v_i := f_i(\mathbf{u}_i);$ be the last assignment to v_i in T_i for each i. Clearly $f_1 = f_2$. Let u_1 and u_2 be the first components of $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively, and for each i, write T_i as $T_i'\, v_i := f_i(\mathbf{u}_i); T_i''$. By the inductive hypothesis applied to $S, \bar{S}$ and each T_i', $\mathcal{M}[\![\bar{S}T_1']\!]_e(u_1) = \mathcal{M}[\![\bar{S}T_2']\!]_e(u_2)$; the Lemma then follows from the analogous result for the other components of each $\mathbf{u}_i$. $\square$

The conclusion of Lemma 22 need not hold without the condition on constant terms $g()$; for example, if $v_1 = v_2 = v$, the schema $T_1 = S = v := g();$, T_2 is *skip* and $\bar{S}$ is $v := h();$ for a constant $h \in \mathcal{F}$, then all possible concatenations of these schemas are near-liberal, and $\mathcal{M}[\![ST_1]\!]_e(v) = \mathcal{M}[\![ST_2]\!]_e(v) = g()$, holds, but $\mathcal{M}[\![\bar{S}T_1]\!]_e(v) \neq \mathcal{M}[\![\bar{S}T_2]\!]_e(v)$ holds.

Lemma 22 is a generalisation of [7, Proposition 59], whose hypotheses required the schemas ST_i and $\bar{S}T_i$ for each $i \in \{1, 2\}$ to be liberal. Under this stronger assumption, the condition on constant terms $g()$ is automatically satisfied, since a liberal predicate-free schema cannot contain two assignments having the same constant function symbol.

Lemma 23 will be used to prove Lemmas 29 and 30.

Lemma 23 (Changing the head of a couple) *Let S be a free linear near-liberal schema and let $p \in Preds(S)$ and $u \in \mathcal{V} \cup \{\omega\}$. Suppose there is a pu-couple I for S. Let $\mu \underline{p, \mathsf{T}} \in pre(\Pi(S))$, and assume that for all $v \in \mathcal{V}$ and constant $g \in \mathcal{F}$, if $\mathcal{M}[\![\mu]\!]_e(v) = g()$ then either also $\mathcal{M}[\![head_S(I)]\!]_e(v) = g()$ or g does not occur along either $tail_S(i, I)$ for $i \in I$. Then there is a pu-couple I' for S such that $\mu = head_S(I')$ and $\{tail_S(k, I) |\, k \in I\} = \{tail_S(k, I') |\, k \in I'\}$.*
In particular, this conclusion holds if for all assignments $v := g();$ lying in the body of a while predicate q in S, no other assignment to v also lies in the body of q, and $head_S(I)$ has the form $\rho'\rho''\rho'''$ with $\mu = \rho'\rho'''$.

Proof. Write $I = \{i_1, i_2\}$ and assume each $\pi_S(i_k, e)$ has prefix $head_S(I)\underline{p, Z_k}$. Since S is free, there exist interpretations j_1, j_2 such that $\pi_S(j_k, e) = \mu\underline{p, Z_k}\, tail_S(i_k, I)$ for each k; and if $u \in \mathcal{V}$, then by Lemma 22 applied to the predicate-free schemas defined

by $head_S(I), \mu$ and each path-segment $tail_S(i_k, I)$ and the fact that I is a pu-couple for S, $\mathcal{M}[\![S]\!]_e^{j_1}(u) \neq \mathcal{M}[\![S]\!]_e^{j_2}(u)$ holds for any such pair $\{j_1, j_2\}$ of interpretations. Clearly this also holds if $u = \omega$. Thus we need only to show that j_1 and j_2 can be chosen such that they differ only on the predicate term $p(\mathcal{M}[\![\mu]\!]_e(\mathbf{refvec}_S(p)))$. Suppose this is impossible. Since S is free, this implies that each path-segment $tail_S(i_k, I)$ has a prefix $\sigma_k \underline{q, \mathsf{T}_k}$ with $\mathsf{T}_1 \neq \mathsf{T}_2$ and $\mathcal{M}[\![\mu\,\sigma_1]\!]_e \mathbf{refvec}_S(q) = \mathcal{M}[\![\mu\,\sigma_2]\!]_e \mathbf{refvec}_S(q)$. However, again by Lemma 22, the same equality with $head_S(I)$ in place of μ also holds, contradicting the existence of the original pu-couple I. Thus the pair j_1, j_2 exists.
If the hypotheses of the last paragraph of the Lemma hold and $\mathcal{M}[\![\mu]\!]_e(v) = g() \neq \mathcal{M}[\![head_S(I)]\!]_e(v)$ hold for some assignment $v := g()$; in S, then the path-segment ρ' passes through g and ρ'' must pass through an assignment to v with function symbol $\neq g$, and thus neither path-segment $\rho''' tail_S(i, I)$ for $i \in I$ passes through g, by the condition on while predicates, and so the original hypotheses are satisfied. $\square$

7 The Reduced Weiser symbol set

For a variable v and a linear schema S, Weiser's original slicing algorithm computed the minimal set of function and predicate symbols that is left-closed under the $\searrow_S$ and $\underset{S}{\leadsto}$ relations and contains every $f \in \mathcal{F}$ for which $f \overset{\text{final}}{\underset{S}{\leadsto}} v$. The authors have proved that a subschema T of S that contains all these symbols is weakly v-equivalent to S, and if T contains *only* the symbols in this minimal set, then T is a v-slice of S [4,7]. For ω, indicating termination behaviour, an analogous set can be defined using while predicates instead of a variable as a starting point in the recursive definition [5, Definition 19]. A subschema of S that contains all the symbols in this set is in this case ω-equivalent to S. The symbol sets given in Definition 25, which take account of the path-prefix leading to a symbol, are subsets of those defined by Weiser's algorithm, as can be proved by induction using their recursive definitions. We call these the Reduced Weiser sets.

It is convenient to make the following definitions.

Definition 24 ((p, X)-links and v-feeding path-segments)
Let S be a linear schema.

- If q is a while predicate in S, then $body_S(q)$ is the body of q in S.
- Let $p \in ifPreds(S)$ and $X \in \{\mathsf{T}, \mathsf{F}\}$. A (p, X)-link in S is a path-segment $\underline{p, X}\nu$ in S, for some terminating path ν in the X-part of p in S.
- If $p \in whilePreds(S)$, then the path-segment $\underline{p, \mathsf{F}}$ is called a (p, F)-link in S; and a path-segment in $(\underline{p, \mathsf{T}}\Pi(body_S(p)))^*\underline{p, \mathsf{F}}$ in which $\underline{p, \mathsf{T}}$ occurs at least once, is a (p, T)-link.

- Let $p, q \in \mathit{Preds}(S)$ and let $v \in \mathcal{V}$. We say that a path-segment μ in S v-feeds p to q if there exists $X \in \{\mathsf{T}, \mathsf{F}\}$ such that $\nu\mu\underline{q, \mathsf{T}}$ is a path-segment in S for some (p, X)-link ν and $\mathcal{M}[\![\mu]\!]_e(w)$ is a vF-term for some $F \in \mathcal{F}^*$ and q references the variable w.

We may refer to a (p, Z)-link, for either $Z \in \{\mathsf{T}, \mathsf{F}\}$, as a p-link.

Definition 25 (The Reduced Weiser sets of path-prefixes and symbols) Let S be a linear schema and let $x \in \mathit{pre}(\Pi(S))$ satisfy $\mathit{nextsymbol}_S(x) \in \mathit{Preds}(S)$. Then we recursively define $\mathcal{W}\mathit{paths}_S(x) \subseteq \mathit{pre}(\Pi(S))$ to be the minimal set satisfying $x \in \mathcal{W}\mathit{paths}_S(x)$ which is closed under the following transformations, where $u \in \mathcal{V}$ and $p \in \mathit{Preds}(S)$.

(1) If $\mu\rho\alpha \in \mathcal{W}\mathit{paths}_S(x)$ for a p-link ρ such that α u-feeds p to $\mathit{nextsymbol}_S(\mu\rho\alpha) \in \mathit{Preds}(S)$ in S and the last assignment to u on ρ exists and has function symbol f, such that either f has arity ≥ 1 or f does not occur on μ, then $\mu \in \mathcal{W}\mathit{paths}_S(x)$ holds.

(2) If $\mu\underline{p, Z}\sigma \in \mathcal{W}\mathit{paths}_S(x)$ such that $p \searrow_S \mathit{nextsymbol}_S(\mu\underline{p, Z}\sigma)$, then $\mu \in \mathcal{W}\mathit{paths}_S(x)$ holds.

We also define $\mathcal{W}\mathit{funcs}_S(x)$ to be the set of all function symbols occurring in all terms $\mathcal{M}[\![\mu]\!]_e(v)$ for $\mu \in \mathcal{W}\mathit{paths}_S(x)$ such that $\mathit{nextsymbol}_S(\mu)$ references v.

If $x \in \mathit{Preds}(S)$ then we define $\mathcal{W}\mathit{paths}_S(x) = \bigcup_{\mathit{nextsymbol}_S(\mu)=x} \mathcal{W}\mathit{paths}_S(\mu)$ and $\mathcal{W}\mathit{funcs}_S(x) = \bigcup_{\mathit{nextsymbol}_S(\mu)=x} \mathcal{W}\mathit{funcs}_S(\mu)$, and we define $\mathcal{W}\mathit{paths}_S(\omega) = \bigcup_{p \in \mathit{whilePreds}(S)} \mathcal{W}\mathit{paths}_S(p)$ and $\mathcal{W}\mathit{funcs}_S(\omega) = \bigcup_{p \in P} \mathcal{W}\mathit{funcs}_S(p)$, where $P = \{p \in \mathit{Preds}(S) | \, p \in \mathit{whilePreds}(S) \vee p \searrow_S q \in \mathit{whilePreds}(S)\}$.

We also define $\mathcal{W}\mathit{paths}_S(x) \subseteq \mathit{pre}(\Pi(S))$ for $x \in \mathcal{V}$ to be the union of all sets $\mathcal{W}\mathit{paths}_S(\mu)$ for μ satisfying the following, for some $p \in \mathit{Preds}(S)$ and $u \in \mathcal{V}$;

(1′) $\mu\rho\alpha \in \Pi(S)$ for a p-link ρ and uGx-path-segment α for $G \in \mathcal{F}^*$ such that the last assignment to u on ρ exists and has function symbol f, such that either f has arity ≥ 1 or f does not occur on μ.

We also define the set $\mathcal{W}\mathit{funcs}_S(x) \subseteq \mathit{Funcs}(S)$ for $x \in \mathcal{V}$ to be the set containing every function symbol occurring in any term $\mathcal{M}[\![\mu]\!]_e(x)$ for $\mu \in \Pi(S)$ or in any set $\mathcal{W}\mathit{funcs}_S(\mu)$ for μ satisfying (1′) and define $\mathcal{W}\mathit{preds}_S(x) = \{\mathit{nextsymbol}_S(\mu) | \, \mu \in \mathcal{W}\mathit{paths}_S(x)\}$ for any x for which $\mathcal{W}\mathit{paths}_S(x)$ is defined. We define $\mathcal{W}\mathit{symbols}_S(x) = \mathcal{W}\mathit{preds}_S(x) \cup \mathcal{W}\mathit{funcs}_S(x)$.

Thus if S is the schema in Figure 3, then f lies in Weiser's original symbol set defined by v, as mentioned in Section 1.1, but $f \notin \mathcal{W}\mathit{funcs}_S(v)$ holds, since every path-prefix through S ending at f passes through the constant assignment to v. On the other hand, if the assignment $v := g_1()$; is replaced by $v := g_2(v)$, (in which case the resulting schema is both free and liberal[6]) then f lies even in the smaller set $\mathcal{W}\mathit{funcs}_S(v)$.

Theorem 28, which states that it is decidable whether a given symbol of a linear schema lies in its Reduced Weiser set with respect to a variable, a predicate or ω, is the main result of this section.

Proposition 26 *Let S be a linear schema and let $\mu, \mu' \in pre(\Pi(S))$ satisfy $nextsymbol_S(\mu) = nextsymbol_S(\mu')$ and suppose $\mu = \rho'\rho''\rho'''$ and $\mu' = \rho'\rho'''$ hold. Let $x \in pre(\Pi(S))$. If μ can be obtained from x by a transformation of Type (1) or (2) from Definition 25, then for some $Z \in \{\mathsf{T}, \mathsf{F}\}$, x can be written as $x = \mu\, \underline{nextsymbol_S(\mu), Z}\, \nu$ and μ' can be obtained from $\mu'\, \underline{nextsymbol_S(\mu), Z}\, \nu$ by a transformation of the same type.*

Proof. This follows immediately from the transformation definitions. □

Lemma 27 *There exists a polynomial P such that the following hold for every linear schema S and every $x, y \in pre(\Pi(S))$ such that $x \in \mathcal{W}paths_S(y)$, where $n = |Funcs(S) \cup Preds(S)|$.*

(1) *There exists $y' \in pre(\Pi(S))$ such that $nextsymbol_S(y) = nextsymbol_S(y')$, $x \in \mathcal{W}paths_S(y')$ and $|y'| - |x| \leq P(n)$.*
(2) *There exist $x', y' \in pre(\Pi(S))$ such that $nextsymbol_S(x) = nextsymbol_S(x')$ and $nextsymbol_S(y) = nextsymbol_S(y')$, $x' \in \mathcal{W}paths_S(y')$ and $|y'| \leq P(n)$.*

Proof.

(1) Given any $y' \in pre(\Pi(S))$ such that $x \in \mathcal{W}paths_S(y')$ there is a sequence $\mu_0 = x, \mu_1, \ldots, \mu_m = y' \in pre(\Pi(S))$ such that each μ_i is obtained from μ_{i+1} by one of the applications of transformations of Type (1) or (2) in Definition 25. Define $\mu_{i+1} = \mu_i \underline{p_i, Z_i} \rho_i$. Assume that $|y'| - |x|$ is minimal subject to the condition that $nextsymbol_S(y) = nextsymbol_S(y')$ holds. We prove $|y'| - |x| \leq P(n)$ by using Proposition 26 to delete path-segments from within the path-prefixes μ_i for $i > 0$ in order to reduce their length without changing $nextsymbol_S(\mu_m)$, thus contradicting the minimality condition. The proof proceeds in stages.
   - We first show that $i < j \Rightarrow p_i \neq p_j$ holds. For if this is false for some $i < j$, then we may reduce the value of $|y'| - |x|$ as follows. Write $\mu_j = \mu_i \underline{p_i, Z_i} \sigma$. The path-segment $\underline{p_i, Z_i} \sigma$ may be deleted from each μ_k for $k > j$ to give μ'_k, and by Proposition 26, the sequence $\mu_0, \mu_1, \ldots, \mu_i, \mu'_{j+1} \ldots, \mu'_m$ satisfies the conditions of the original sequence, contradicting the minimality condition. Thus $m \leq n$ holds.
   - We now show that we may assume that the number of occurrences of a letter $\underline{r, \mathsf{T}}$ for $r \in whilePreds(S)$ in any ρ_i is bounded by a polynomial in n. Suppose that some μ_i is obtained from μ_{i+1} by a transformation of Type (2). Then ρ_i cannot pass more than once through any letter $\underline{r, \mathsf{T}}$ for $r \in whilePreds(S)$,

otherwise we may again use Proposition 26 to delete a path-segment within ρ_i from every μ_k with $k > i$, contradicting the minimality condition. On the other hand, suppose that some μ_i is obtained from μ_{i+1} by a transformation of Type (1). We may write $\rho_i = \alpha_i\beta_i$, where β_i is a $v_i g_1 \dots g_k v_{i+1}$-path-segment for $g_1, \dots, g_k \in \mathcal{F}$, p_{i+1} references $v_{i+1} \in \mathcal{V}$, and $\underline{p_i, Z_i}\alpha_i$ is a (p_i, Z_i)-link passing through an assignment to $v_i \in \mathcal{V}$. There are no repeated function symbols in $g_1, \dots, g_k$; otherwise a path-segment within β_i can be deleted from each of $\mu_{i+1}, \dots, \mu_m$ as before, using Proposition 26. Hence $k \le n$ holds. Similarly, each path-segment within β_i connecting any g_j to g_{j+1} does not pass more than once through any letter $\underline{r, \mathsf{T}}$ for $r \in \mathit{whilePreds}(S)$. The same assumption on while predicate letters may be made for α_i, which needs only to pass through the assignment to v_i, thus proving the bound.
- Thus we have proved the existence of a polynomial bound on the number of occurrences of letters $\underline{r, \mathsf{T}}$ for $r \in \mathit{whilePreds}(S)$ in μ_m. The existence of the polynomial P now follows by observing that if $z \in \mathcal{F} \cup \mathit{ifPreds}(S)$, and μ_m passes $j > 1$ times through z, then necessarily μ_m also passes at least j times through $\underline{r, \mathsf{T}}$ for some $r \in \mathit{whilePreds}(S)$.

(2) This is similar to (1), except that here we also use Proposition 26 to show that μ_0 does not pass more than once through any letter $\underline{r, \mathsf{T}}$ for $r \in \mathit{whilePreds}(S)$. □

Theorem 28 *Let S be a linear schema and let $q \in \mathit{Preds}(S) \cup \mathcal{V} \cup \{\omega\}$.*

(1) *Let $p \in \mathit{Preds}(S)$. Then it is decidable whether $p \in \mathcal{W}preds_S(q)$.*
(2) *Let $f \in \mathit{Funcs}(S)$. Then it is decidable whether $f \in \mathcal{W}funcs_S(q)$.*

Proof.

(1) Assume first that $q \in \mathit{Preds}(S)$. Then by Part (2) of Lemma 27 and the definition of $\mathcal{W}preds_S(q)$, there is a polynomial P such that $p \in \mathcal{W}preds_S(q)$ if and only if there exists $x, y \in \mathit{pre}(\Pi(S))$ such that $p = \mathit{nextsymbol}_S(x)$ and $q = \mathit{nextsymbol}_S(y)$ and $x \in \mathcal{W}paths_S(y)$ and $|y| \le P(n)$. Since there are finitely many elements $y \in \mathit{pre}(\Pi(S))$ such that $|y| \le P(n)$, and they can all be enumerated, the conclusion follows.
If instead $q = \omega$, then $p \in \mathcal{W}preds_S(q)$ if and only if $p \in \mathcal{W}preds_S(q')$ for some $q' \in \mathit{whilePreds}(S)$, and the decidability result follows immediately. If $q \in \mathcal{V}$, then define the schema $T = S$ *if* $r(q)$ *then skip* such that the symbol r does not occur in S and so T is linear. Then $p \in \mathcal{W}preds_S(q)$ if and only if $p \in \mathcal{W}preds_T(r)$, which we have shown to be decidable.

(2) Assume first that $q \in \mathit{Preds}(S)$. By Part (1) of Lemma 27, there is a polynomial P such that $f \in \mathcal{W}funcs_S(q)$ if and only if there exist $x, y \in \mathit{pre}(\Pi(S))$ such that $|y| - |x| \le P(n)$, $p = \mathit{nextsymbol}_S(x)$ and $q = \mathit{nextsymbol}_S(y)$ and x is an $fg_1 \dots g_k v$-path-segment for some $G \in \mathcal{F}^*$ and $v \in refset(p)$. We may use Proposition 26 to ensure that if this condition holds, then no function symbol

occurs more than once in G, which hence has length $\leq n$, no letter $\underline{r, \mathsf{T}}$ for $r \in \mathit{whilePreds}(S)$ occurs more than once in each path-segment connecting g_j to g_{j+1}. As in the proof of Part (1) of Lemma 27, this gives a computable upper bound on the length of y, thus allowing $f \in \mathcal{W}funcs_S(q)$ to be decided.
If instead $q \in \mathcal{V} \cup \{\omega\}$, then the proof is similar to that of Part (1) under this assumption. $\square$

8 A symbol's membership in a variable's Reduced Weiser set implies it may affect the variable's final value

The main result of this section is Theorem 32, in which we prove that membership of a symbol in the Reduced Weiser set of a schema S with respect to some $v \in \mathcal{V} \cup \{\omega\}$ means that it can affect the semantics of S as given by v. We do this by using the recursive definition of $\mathcal{W}funcs_S(v)$ and $\mathcal{W}preds_S(v)$, and this motivates the three preceding Lemmas which now follow.

Lemma 29 *Let S be a free linear near-liberal schema such that for all assignments $w := g()$; lying in the body of a while predicate r in S, no other assignment to w also lies in the body of r. Let $p, q \in \mathit{Preds}(S)$ and let $v \in \mathcal{V} \cup \{\omega\}$ and suppose there exists a $q(\mathbf{t})v$-couple I for S such that $head_S(I)$ has a prefix $\mu\underline{p, Y}$. Suppose that one of the following holds.*

(1) *There is a subterm $f(\mathbf{a})$ of one of the components of $\mathbf{t}$ which is not created by any assignment on μ, and $p \searrow_S f$ holds.*
(2) *$p \searrow_S q$.*

Then there exists a pv-couple H for S such that $head_S(H) = \mu$.

Proof. We will prove the Lemma for Case (1). Case (2) is analogous, but with f replaced by q. The proof proceeds in stages.

(a) We may assume that $p \searrow_S f\,(Y)$ holds, since otherwise $head_S(I)$ must have a prefix $\mu\underline{p, Y}\,\mu'\,\underline{p, \neg Y}$ such that $\underline{p, \neg Y}$ and hence f do not occur on μ', and so the conclusion of the Lemma follows from considering this longer prefix and $\neg Y$ in place of $\mu\underline{p, Y}$ and Y and using Lemma 23 to delete the path-segment $\underline{p, Y}\,\mu'$.
(b,$\mathcal{V}$) We now observe that if $v \in \mathcal{V}$, it may be assumed that the interpretations in I only map finitely many while predicate terms to T, and only map finitely many p-predicate terms to Y; this is clear from the termination of both paths defined by I and the fact that $p \in \mathit{whilePreds}(S) \Rightarrow Y = \mathsf{T}$ clearly follows from (a).
(b,ω) We now show that if $v = \omega$ then it may be assumed that the interpretations in I only map finitely many p-predicate terms to Y. Suppose that this finiteness condition does not already hold, and that $v = \omega$. Write $I = \{i, j\}$ where i

defines the non-terminating path through S. We define recursively the (possibly finite) set $\{p(\mathbf{x}_1), p(\mathbf{x}_2), \ldots\}$ of predicate terms, and the set $\{i_0 = i, i_1, i_2, \ldots\}$ of interpretations, as follows. Each path $\pi_S(i_n, e)$ passes through the predicate terms $\{p(\mathbf{x}_1), p(\mathbf{x}_2), \ldots, p(\mathbf{x}_{n+1})\}$ in order, and $p(\mathbf{x}_r) = \neg Y$ is a consequence of $\pi_S(i_n, e)$ for all $r \leq n$, and $p(\mathbf{x}_{n+1})$ is the first p-predicate term occurring on $\pi_S(i_n, e)$ that i_n maps to Y and does not occur on $\pi_S(j, e)$. We define $i_{n+1} = i_n(p(\mathbf{x}_{n+1}) = \neg Y)$. By induction on n, $\mu\underline{p, Y}$ is a prefix of each path $\pi_S(i_n, e)$. If $\pi_S(i_{n+1}, e)$ is the first terminating path in the sequence, then the conclusion of the Lemma follows from Lemma 23 using the $p\omega$-couple $\{i_n, i_{n+1}\}$ for S, so we may assume that every interpretation i_n defines a nonterminating path. We now replace I by $\{i', j'\}$, where i' maps each predicate term $p(\mathbf{x}_r)$ to $\neg Y$ and is otherwise the same as i, and similarly for j' and j. Clearly $\pi_S(j', e) = \pi_S(j, e)$, which terminates and has $\mu\underline{p, Y}$ as a prefix. If there are finitely many predicate terms $p(\mathbf{x}_r)$, then $i' = i_n$ for the last i_n defined; we have shown that $\pi_S(i_n, e)$ may be assumed to be nonterminating. If there are infinitely many predicate terms $p(\mathbf{x}_r)$, then again $\pi_S(i', e)$ is nonterminating since it passes through all such predicate terms. Hence $\{i', j'\}$ is a $p\omega$-couple for S such that $\mu\underline{p, Y}$ is a prefix of both its paths, and hence of $\mathit{head}_S(\{i', j'\})$. These paths pass only finitely often through $\underline{p, Y}$, by the construction of i', so we may assume i' and j' map finitely many p-predicate terms to Y.

(c) We now prove the Lemma by induction on the number of p-predicate terms that either interpretation in I maps to Y; we have shown in (b,$\mathcal{V}$) or (b,ω) that this number may be assumed to be finite. Let $p(\mathbf{s})$ be the predicate term defined at the occurrence of p after μ, and define $I' = \{i(p(\mathbf{s}) = \neg Y) | \, i \in I\}$. From (b,$\mathcal{V}$), and since S is free, the interpretations in I' both define terminating paths if $v \in \mathcal{V}$. Thus we may assume that $\mathcal{M}[\![S]\!]^i_e(v) = \mathcal{M}[\![S]\!]^{i(p(\mathbf{s})=\neg Y)}_e(v)$ for each $i \in I'$ and whether or not $v = \omega$, otherwise the Lemma follows immediately. Thus I' is also a $q(\mathbf{t})v$-couple I for S, and the term $f(\mathbf{a})$ must be created later on $\mathit{head}_S(I')$. Hence $\mathit{head}_S(I')$ has a prefix $\mu\underline{p, \neg Y}\tau\underline{p, Y}$ such that $\underline{p, Y}$ does not occur on τ. Thus the Lemma follows from considering this longer prefix in place of $\mu\underline{p, Y}$, using the inductive hypothesis applied to I' and using Lemma 23 to delete $\underline{p, \neg Y}\tau$. $\Box$

Lemma 30 is a strengthening of Part (1) of Lemma 29.

Lemma 30 *Let S be a free linear near-liberal schema such that for all assignments $v := g()$; lying in the body of a while predicate r in S, no other assignment to v also lies in the body of r. Let $p, q \in \mathit{Preds}(S)$, $u \in \mathcal{V}$ and $v \in \mathcal{V} \cup \{\omega\}$. Suppose that there exists a qv-couple I for S such that $\mathit{head}_S(I) = \mu\,\rho\,\alpha$, where ρ is a p-link and the path-segment α u-feeds p to q. Suppose also that there exists a p-link along which the last assignment to u exists and has function symbol f, such that either f has arity ≥ 1 or f does not occur on μ. Then there exists a pv-couple H for S such that $\mathit{head}_S(H) = \mu$.*

Proof. Let ρ' be the p-link whose existence is asserted in the penultimate sentence of the Lemma. We consider two cases separately.

(1) Suppose first that $\rho = \rho'$. Then the conclusion follows from Part (1) of Lemma 29 applied to the term defined by the symbol f, using the near-liberality condition on S if f has arity ≥ 1.

(2) For the general case we prove the Lemma using induction on the length of α. We will show that we can replace ρ by ρ' in $\mathit{head}_S(I)$ and then use Case (1). If $v \in \mathcal{V}$, then we may assume that both interpretations in I map finitely many predicate terms to T, and hence altering either interpretation at finitely many predicate terms preserves path termination, since S is free. For each $i \in I$, we define an interpretation $\phi(i)$ satisfying $\mu\rho'\alpha \in \mathit{pre}(\pi_S(\phi(i), e))$ by successively altering i at predicate terms encountered along $\rho'\alpha$. We now show that we may assume that these alterations of the interpretations in I never change the final value of v (or termination if $v = \omega$) and hence that $\mathcal{M}[\![S]\!]^i_e(v) = \mathcal{M}[\![S]\!]^{\phi(i)}_e(v)$ for each $i \in I$ holds.

- If altering an interpretation in I along ρ' changes the final value of v, then ρ' has a prefix $\sigma\underline{p', X}$ for $p' = p \vee p \searrow_S p'$ such that there exists a $p'v$-couple J satisfying $\mathit{head}_S(J) = \mu\sigma$, and hence the Lemma follows from Part (2) of Lemma 29 if $p \searrow_S p'$ or 23 if $p = p'$.
- If altering either $i \in I$ along $\rho'\alpha$ first changes the final value of v at a predicate within α, then α has a prefix $\gamma\underline{p', X}$ such that the predicate term defined at p' after $\mu\rho'\gamma$ differs from the one occurring after $\mu\rho\gamma$, otherwise the alteration would not be needed. Thus replacing ρ by ρ' in $\mu\rho\gamma$ changes the value of some variable referenced by p', and so there is a variable u' such that γ u'-feeds p to p' and $\mathcal{M}[\![\mu\rho]\!]_e(u') \neq \mathcal{M}[\![\mu\rho']\!]_e(u')$. Hence at least one element of the set $\{\mathcal{M}[\![\mu\rho]\!]_e(u'), \mathcal{M}[\![\mu\rho']\!]_e(u')\}$ differs from $\mathcal{M}[\![\mu]\!]_e(u')$. Thus the Lemma follows from the inductive hypothesis applied to p', u', the appropriate element of $\{\rho, \rho'\}$ and γ in place of α.

Thus we may assume that $\{\phi(i) | i \in I\}$ is also a qv-couple for S, and that $\mu\rho'\alpha$ is a prefix of both paths $\pi_S(\phi(i), e)$, and hence of $\mathit{head}_S(I)$. By Lemma 23, we may assume that $\mu\rho'\alpha = \mathit{head}_S(I)$ holds, and so the Lemma follows from Case (1). $\square$

Lemma 31 *Let S be a free linear schema and assume that $q \in \mathit{Preds}(S)$ and either $q \in \mathit{whilePreds}(S)$ or $q \searrow_S q'$ for some $q' \in \mathit{whilePreds}(S)$. Let $\mu\underline{q, \mathsf{T}} \in \mathit{pre}(\Pi(S))$. Then there exists a $q\omega$-couple I for S such that $\mathit{head}_S(I) = \mu$.*

Proof. We will assume that $q \in \mathit{whilePreds}(S)$; the other case is similar. Let ν be a non-terminating path of which $\mu\underline{q, \mathsf{T}}$ is a prefix, which does not subsequently pass through $\underline{q, \mathsf{F}}$. Thus ν does not pass through any predicate p satisfying $p \searrow_S q$. Let ρ be a terminating path having $\mu\underline{q, \mathsf{F}}$ as a prefix, which does not subsequently pass through $\underline{p, \mathsf{T}}$ for any $p \in \mathit{whilePreds}(S)$ such that $p \searrow_S q$ holds. Clearly the only predicates through which both μ and ρ pass occur on μ and at q just after μ, and S

is free, hence there exist interpretations i, j such that $\pi_S(i, e) = \nu$, $\pi_S(j, e) = \rho$ and i and j differ only at the q-predicate term occurring after μ. Thus defining $I = \{i, j\}$ proves the Lemma. $\square$

Theorem 32 *Let S be a free linear near-liberal schema such that for all assignments $w := g()$; lying in the body of a while predicate r in S, no other assignment to w also lies in the body of r. Let $v \in \mathcal{V} \cup \{\omega\}$.*

(1) *If $\gamma \in \mathcal{W}paths_S(v)$ holds and $nextsymbol_S(\gamma) = p$, then there exists a pv-couple I for S such that $head_S(I) = \gamma$.*
(2) *If $f \in \mathcal{W}funcs_S(v)$, then either $v \in \mathcal{V}$ and f occurs in a term $\mathcal{M}[\![\nu]\!]_e(v)$ for some $\nu \in \Pi(S)$, or there exists a pv-couple I for S for some predicate p referencing a variable u such that f occurs in the term $\mathcal{M}[\![head_S(I)]\!]_e(u)$.*

Proof. We first prove Part (1) of the Theorem. Let $\gamma \in \mathcal{W}paths_S(v)$ hold with $nextsymbol_S(\gamma) = p$. From the recursive definition of $\mathcal{W}paths_S(v)$, there exists $\mu\underline{q, \mathsf{T}} \in pre(\Pi(S))$ such that $\gamma \in \mathcal{W}paths_S(\mu)$ and the following hold, with μ as the first in the sequence of path-prefixes 'witnessing' that $\gamma \in \mathcal{W}paths_S(\mu)$ holds.

- If $v = \omega$, then either $q \in whilePreds(S)$ or q contains a while predicate in its body.
- If $v \in \mathcal{V}$ then Condition $(1')$ in Definition 25 holds with q and v in place of p and x respectively; thus, $\mu\rho\alpha \in \Pi(S)$ for a q-link ρ and uGv-path-segment α for $G \in \mathcal{F}^*$ such that the last assignment to the variable u on ρ exists and has function symbol f, such that either f has arity ≥ 1 or f does not occur on μ.

We now prove that for either value of v, Part (1) of the Theorem holds if $\gamma = \mu$.

- If $v = \omega$ then this follows from Lemma 31.
- If $v \in \mathcal{V}$ then we prove this by defining a new schema T to be S *if* $q'(v)$ *then* $v := h()$; for symbols q', h not occurring in S. Clearly T satisfies the hypotheses given for S, and it can be easily seen that there exists a $q'v$-couple J for T such that $head_T(I) = \mu\rho\alpha$. Thus by Lemma 30, there exists a qv-couple I for T such that $head_T(I) = \mu$, and clearly I is also a qv-couple for S, proving the result.

The proof of Part (1) in the general case follows by induction on the length of the sequence of iterations of Conditions (1,2) in Definition 25 that demonstrates that $\gamma \in \mathcal{W}paths_S(\mu)$ holds.

- If the last Condition in the sequence was (1), then the conclusion follows from the inductive hypothesis applied to the penultimate element of $\mathcal{W}paths_S(\mu)$ defined by the sequence and Lemma 30.
- If the last Condition in the sequence was (2), then the proof of Part (1) follows from the inductive hypothesis applied to the penultimate element of $\mathcal{W}paths_S(\mu)$ defined by the sequence and Part (2) of Lemma 29.

We now prove Part (2) of the Theorem. Let $f \in \mathcal{W}funcs_S(v)$. Assume that f does not occur in any term $\mathcal{M}[\![\nu]\!]_e(v)$ for some $\nu \in \Pi(S)$ and $v \in \mathcal{V}$. Then by the definition of $\mathcal{W}funcs_S(v)$, f occurs in a term $\mathcal{M}[\![\gamma]\!]_e(u)$ for $\gamma \in \mathcal{W}paths_S(v)$ with the predicate $nextsymbol_S(\gamma)$ referencing the variable u. Thus the conclusion follows straightforwardly from Part (1) of this Theorem. $\square$

9 Slicing Theorems

In this Section we prove our main theorem, Theorem 39. In order to do this, we first prove, in Theorem 38, that if a subschema of a schema S contains all elements of the reduced Weiser symbol set with respect to a variable or ω, then it preserves some of the behaviour of S. This motivates the definition and results which now follow.

Definition 33 *Let S be a linear schema and let T be a subschema of S. For any path-segment μ in S, we define $proj_T(\mu)$ to be the word obtained from μ by deleting every letter in μ whose symbol does not occur in T.*

It follows easily from the definition of $\Pi(S)$ that if $\mu \in \Pi(S)$ then $proj_T(\mu) \in \Pi(T)$ holds.

Proposition 34 *Let S_1 be a linear schema and let S_2 be a subschema of S. For each $k \in \{1, 2\}$, let ν_k be a terminating path in S_k. Then we can write*

$$\nu_k = \alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k}\ldots\alpha_{nk},$$

such that for each $r \leq n$, $proj_{S_2}(\alpha_{r1}) = \alpha_{r2}$ and there exists $\{Y_{r1}, Y_{r2}\} = \{\mathsf{T}, \mathsf{F}\}$ and $p_r \in Preds(S_2)$ and each ρ_{rk} is a (p_r, Y_{rk})-link in S_k.

Proof. This follows by induction on the total number of symbols and *skip*s in S_1. If S_1 has the form T_1T_2 or *if* $q(\mathbf{x})$ *then* T_1 *else* T_2, then the result follows easily from the inductive hypothesis and the definition of a subschema. If S_1 is an assignment or *skip*, or S_2 is *skip*, then again the result follows easily. Lastly, assume that $S_1 =$ *while* $q(\mathbf{y})T_1$ and that $S_2 =$ *while* $q(\mathbf{y})T_2$, where T_2 is a subschema of T_1. Let $m_k \in \mathbb{N}$ be such that $\underline{q, \mathsf{T}}$ occurs m_k times in each path ν_k. Thus we may write $\nu_k = \underline{q, \mathsf{T}}\mu_{1k}\underline{q, \mathsf{T}}\mu_{2k}\ldots\mu_{mk}\sigma_k$, where m is the minimum of m_1, m_2 and for each k, $\mu_{jk} \in \Pi(T_k)$, $m_k = m \Rightarrow \sigma_k = \underline{q, \mathsf{F}}$ and $m_k > m \Rightarrow \sigma_k \in \Pi(S_k) - \{\underline{q, \mathsf{F}}\}$. The result now follows by applying the inductive hypothesis to T_1, showing that each μ_{ik} has the correct form; note that if $m_1 = m_2$ then each α_{nk} will end in $\underline{q, \mathsf{F}}$; otherwise, each α_{nk} is the empty word. $\square$

Proposition 35 *Let S be a linear schema and let T be a subschema of S. Let ν be a terminal or non-terminating path through S and assume that $proj_T(\nu)$ is non-*

terminating if ν is. If $proj_T(\nu)$ has a prefix $\alpha_1\rho_2\alpha_2\rho_3\ldots\alpha_{n-1}\rho_n$, with each ρ_r a p_r-link in T for $p_r \in Preds(T)$, then ν has a prefix $\tilde{\alpha}_1\tilde{\rho}_2\tilde{\alpha}_2\tilde{\rho}_3\ldots\tilde{\alpha}_{n-1}\tilde{\rho}_n$, where each $proj_T(\tilde{\alpha}_r) = \alpha_r$, $proj_T(\tilde{\rho}_r) = \rho_r$, and each $\tilde{\rho}_r$ is a p_r-link in S.

Proof. The conclusion follows by induction on the total number of symbols and *skip*s in S. If S is an assignment, or T is *skip*, then the conclusion is immediate. If S has the form S_1S_2, *if* $p(\mathbf{x})$ *then* S_1 *else* S_2 or *while* $p(\mathbf{x})S_1$, then $T \neq$ *skip* has the same form but with S_i replaced by a subschema T_i. In these cases, the conclusion follows from the definitions of $\Pi(S)$ and $\Pi(T)$, the inductive hypothesis applied to T and the fact that any q-link in T for $q \neq p$ must lie within a subschema T_i. $\square$

The conclusion of Proposition 35 need not hold if $proj_T(\nu)$ terminates but ν does not; for example, let $S =$ *if* $p(\mathbf{x})$ *then while* $q(\mathbf{y})$*skip* and $T =$ *if* $p(\mathbf{x})$ *then skip*, and let ν be the non-terminating path through S. Clearly $proj_T(\nu) = \underline{p, \mathsf{T}}$, which is a p-link in T. Let α_1 be the empty word and let $\rho_2 = \underline{p, \mathsf{T}}$ and $n = 2$. In this case no p-link $\tilde{\rho}_2$ in S exists, since it would have to end in $\underline{q, \mathsf{F}}$, which does not occur in ν.

Lemma 36 *Let S_1 be a linear schema and let S_2 be a subschema of S. Let ν_1, ν_2 be paths through S_1 and S_2 respectively. Suppose that one of the following conditions holds.*

(1) *ν_1 is a terminal path in S_1 and ν_2 is a non-terminating path in S_2.*
(2) *ν_2 is a terminal path in S_2 and ν_1 is a non-terminating path in S_1 and S_2 contains every while predicate in S_1.*

Then for each $k \in \{1, 2\}$, ν_k has a prefix

$$\alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k}\ldots\alpha_{nk}\underline{q, Z_k},$$

such that for each $r \leq n$, $proj_{S_2}(\alpha_{r1}) = \alpha_{r2}$ and there exists $\{Y_{r1}, T_{r2}\} = \{\mathsf{T}, \mathsf{F}\}$ and $p_r \in Preds(S)$, each ρ_{rk} is a (p_r, Y_{rk})-link in S_k, $Z_1 \neq Z_2$ and q is either a while predicate in S_1 or contains a while predicate in one of its parts.

Proof. We first assume that that $S_1 = S_2$. In this special case the conclusion follows by induction on the length of the shorter, terminating path in $\{\nu_1, \nu_2\}$ minus that of $pre(\nu_1, \nu_2)$. By Lemma 6 and the fact that exactly one path terminates, each ν_k has a prefix $pre(\nu_1, \nu_2)\underline{q, Z_k}$ with $Z_1 \neq Z_2$. If q is either a while predicate in S_1 or contains a while predicate in one of its parts, then the conclusion follows immediately, with $pre(\nu_1, \nu_2) = \alpha_{1k}$. Otherwise, each ν_k has a prefix $pre(\nu_1, \nu_2)\underline{q, Z_k\sigma_k}$ for a (q, Z_k)-link $\underline{q, Z_k\sigma_k}$ in S_k. Define the path ν_1' by replacing $\underline{q, Z_1\sigma_1}$ by $\underline{q, Z_2\sigma_2}$ in ν_1 after $pre(\nu_1, \nu_2)$. The conclusion now follows from the inductive hypothesis applied to ν_1' and ν_2.

For the general case, observe that if (2) holds, then ν_1 and hence $proj_{S_2}(\nu_1)$ pass infinitely many times through a while predicate and so $proj_{S_2}(\nu_1)$ is non-terminating. Thus if either (1) or (2) holds, we may apply the conclusion for the case $S_1 = S_2$ to the paths $proj_{S_2}(\nu_1)$ and ρ_2 and then use Proposition 35 to prove the Theorem. $\square$

Lemma 37 *Let S_1 be a linear schema such that for all assignments $v := g()$; lying in the body of a while predicate q in S_1, no other assignment to v also lies in the body of q, and let S_2 be a subschema of S_1. For each $k \in \{1, 2\}$, let*

$$\nu_k = \alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k} \dots \rho_{nk}\alpha_{nk} \in pre(\Pi(S_k)),$$

such that for each $r \le n$, $proj_{S_2}(\alpha_{r1}) = \alpha_{r2}$ and there exists $\{Y_{r1}, T_{r2}\} = \{\mathsf{T}, \mathsf{F}\}$ and $p_r \in Preds(S)$, and each ρ_{rk} is a (p_r, Y_{rk})-link in S_k. Let $v \in \mathcal{V}$ and assume that $\mathcal{M}[\![\nu_1]\!]_e(v) \neq \mathcal{M}[\![\nu_2]\!]_e(v)$ holds. Suppose that S_2 contains every function symbol occurring in the term $\mathcal{M}[\![\nu_1]\!]_e(v)$. Then for some $r \le n$, there exists $f \in \mathcal{F}$ satisfying $p_r \searrow_{S_1} f$ such that $\alpha_{r1} \dots \rho_{n1}\alpha_{n1}$ is an $assign_{S_1}(f)Gv$-path-segment for some $G \in \mathcal{F}^*$ and either f has arity ≥ 1 or f does not occur on $\alpha_{11}\rho_{21} \dots \alpha_{r-1\,1}$.

Proof. This follows by induction, firstly on n and secondly on the length of α_{n1}. We consider three cases separately.

- Suppose that the last letter of α_{n1} is not an assignment to v. Write $\alpha_{n1} = \beta l$, where $l \in alphabet(S_1)$. Then the conclusion follows by replacing α_{n1} and α_{n2} by β and $proj_{S_2}(\beta)$ respectively and using the inductive hypothesis.
- Suppose that the last letter of α_{n1} (and hence of α_{n2}, since h occurs in $\mathcal{M}[\![\nu_1]\!]_e(v)$) is $\underline{v := h(\mathbf{u})}$. Then the conclusion follows by replacing each α_{nk} by its prefix of length $|\alpha_{n1}| - 1$ and replacing v by each component of $\mathbf{u}$ in turn and applying the inductive hypothesis.
- Lastly, suppose that α_{n1} (and hence α_{n2}) is the empty word. Assume the conclusion is false. We now show that

  $$\mathcal{M}[\![\alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k} \dots \alpha_{n-1\,k}\rho_{nk}]\!]_e(v) = \mathcal{M}[\![\alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k} \dots \alpha_{n-1\,k}]\!]_e(v)$$

  holds for each k. For $k = 1$ this is immediate from the falsity of the conclusion for $r = n$. For $k = 2$, the falsity of the conclusion for $r = n$ implies that the equality can only fail if ρ_{n2} passes through a constant $f \in \mathcal{F}$ assigning to v. However, by the falsity of the conclusion for $r = n$, this implies that f occurs on $\alpha_{11}\rho_{21} \dots \alpha_{r-1\,1}$, and hence $\mathcal{M}[\![\nu_1]\!]_e(v) = \mathcal{M}[\![\nu_2]\!]_e(v) = f()$ follows, contradicting the hypotheses. Thus we can delete the path-segments ρ_{nk} from the end of ν_k, reducing the value of n, and use the inductive hypothesis. □

Theorem 38 *Let S_1 be a linear schema such that for all assignments $w := g()$; lying in the body of a while predicate r in S, no other assignment to w also lies in the body of r and let S_2 be a subschema of S_1. Let i be an interpretation. Then the following hold.*

(1) *Let $v \in \mathcal{V}$. If S_2 contains every element of $\mathcal{W}symbols_{S_1}(v)$ and the paths $\pi_{S_1}(i, e)$ and $\pi_{S_2}(i, e)$ both terminate, then $\mathcal{M}[\![S_1]\!]^i_e(v) = \mathcal{M}[\![S_2]\!]^i_e(v)$ holds.*

(2) *If S_2 contains every element of $\mathcal{W}symbols_{S_1}(\omega)$, then $\pi_{S_1}(i, e)$ terminates $\iff$ $\pi_{S_2}(i, e)$ terminates.*

(3) *If for every $p \in whilePreds(S_2)$, S_2 contains every element of $\mathcal{W}symbols_{S_1}(p)$, then $\pi_{S_1}(i,e)$ terminates $\Rightarrow \pi_{S_2}(i,e)$ terminates.*

Proof. Assume that one of the assertions is false. In all cases, for each $k \in \{1,2\}$, there exist $\nu_k \in pre(\pi_{S_k}(i,e))$ such that

$$\nu_k = \alpha_{1k}\rho_{2k}\alpha_{2k}\rho_{3k}\alpha_{3k}\dots\alpha_{nk},$$

such that for each $r \leq n$, $proj_{S_2}(\alpha_{r1}) = \alpha_{r2}$ and there exists $\{Y_{r1}, Y_{r2}\} = \{\mathsf{T},\mathsf{F}\}$ and $p_r \in Preds(S_2)$ and each ρ_{rk} is a (p_r, Y_{rk})-link in S_k, and there exists $w \in \mathcal{V}$ such that $\mathcal{M}[\![\nu_1]\!]_e(w) \neq \mathcal{M}[\![\nu_2]\!]_e(w)$, and either $nextsymbol_{S_1}(\nu_1) = nextsymbol_{S_2}(\nu_2)$ lies in $Preds(S_2)$ and references w, or ν_1 and ν_2 are terminal. In all cases, S_2 contains every element of $\mathcal{W}funcs_{S_1}(\nu_1)$ and every predicate $nextsymbol_{S_1}(\mu)$ for $\mu \in \mathcal{W}paths_{S_1}(\nu_1)$. In Case (1), this follows from Proposition 34 and the fact that $w = v$ and $\mathcal{W}funcs_{S_1}(\nu_1) \subseteq \mathcal{W}funcs_{S_1}(v)$ and $\mathcal{W}paths_{S_1}(\nu_1) \subseteq \mathcal{W}paths_{S_1}(v)$; and in Cases (2,3) it follows from Lemma 36 and the fact that $\mathcal{W}funcs_{S_1}(\nu_1) \subseteq \mathcal{W}funcs_{S_1}(\omega)$ and $\mathcal{W}paths_{S_1}(\nu_1) \subseteq \mathcal{W}paths_{S_1}(\omega)$. Assume that n is minimal with these conditions. Lemma 37 now gives a contradiction. $\square$

Our main result, Theorem 39, is a summary of preceding results.

Theorem 39 *Let S be a linear schema such that for all constant assignments $w := g()$; lying in the body of a while predicate r in S, no other assignment to w also lies in the body of r, and let T be a subschema of S. Let $v \in \mathcal{V}$. Then the following hold.*

(1) *If T contains every symbol in $\mathcal{W}symbols_S(v)$, then S and T are weakly v-equivalent.*
(2) *If T contains every symbol in $\mathcal{W}symbols_S(v)$, and contains every symbol in $\mathcal{W}symbols_S(p)$ for each $p \in whilePreds(T)$, then T is a v-slice of S.*
(3) *If S is free and near-liberal, then S and T are strongly v-equivalent if and only if T contains every symbol in $\mathcal{W}symbols_S(u)$ for each $u \in \{v, \omega\}$. In particular, it is decidable whether S and T are strongly v-equivalent under these extra conditions on S.*

Proof. Part (3) follows from Parts (1) and (2) of Theorem 38 and both Parts of Theorem 32, with the decidability result following from Theorem 28. Part (2) is simply a restatement of Part (3) of Theorem 38. Part (1) is a restatement of Part (1) of Theorem 38. $\square$

As an additional example of the application of Parts (1) and (2) of Theorem 39, consider the linear schema S of Figure 5, which is neither free nor liberal. The subschema of S obtained by deleting the assignment to u is a v-slice of S. This follows from Part (2) of Theorem 39, since $k \notin \mathcal{W}symbols_S(v) \cup \mathcal{W}symbols_S(q)$ holds. On the other hand, if the assignments $w := h(w)$; and $u := k()$ are both deleted from S, then the resulting subschema T is weakly v-equivalent to S, by Part (1) of Theorem 39, since

```
while $q(w)$ do {
        $w := h(w)$;
        if $p(u)$ then {
                $v := g_1()$;
                $u := k()$;
                }
        }
```

Fig. 5. Deleting the assignment to u from this schema defines a v-slice of it.

$k \notin \mathcal{W}symbols_S(v)$, but T is not a v-slice of S, since if h is interpreted as the function $w \mapsto w + 1$ on the integers, and $q(1)$ is defined to be true, whereas $q(2)$ is defined to be false, then any program thus defined by S will terminate from any initial state for which $w = 1$, whereas for T this assertion is false. Since k lies in Weiser's set $\mathcal{N}_S(v)$, this example demonstrates the non-optimality of Weiser's algorithm even for very simple schemas.

We isolate the following consequence of Theorem 39.

Theorem 40 *Let S be a linear, free and near-liberal schema such that for all assignments $v := g()$; lying in the body of a while predicate r in S, no other assignment to v also lies in the body of r, and let T be a subschema of S. Then S and T are ω-equivalent if and only if T contains every symbol in $\mathcal{W}preds_S(\omega)$ and $\mathcal{W}funcs_S(\omega)$. In particular, it is decidable whether S and T are ω-equivalent.*

Proof. This is a special case of Part (3) of Theorem 39, where v is taken to be a variable not occurring in S. $\square$

Part (2) of Theorem 39 is a strengthening of [5, Theorem 20] for linear schemas, which states that the subschema of a schema S containing precisely those symbols in Weiser's original set with respect to a variable v, is a v-slice of S.

We mention that there is a strict ordering between the conditions given on the schema T in Theorem 39. To see this, let S be the schema in Figure 6. By Theorem 39, deleting the line *while* $p(u)$ *do* $u := f(u)$; gives a v-slice of S which is not strongly v-equivalent to S, (for example, if an interpretation always maps p to T and q to F, then the subschema will terminate, but S will not), and deleting the h-assignment (whether or not the p-statement is also deleted) gives a schema that is weakly v-equivalent to S, but is not a v-slice, since termination is not preserved for all interpretations.

```
while $p(u)$ do  $u := f(u)$;
while $q(w)$ do  {
                   $v := g_1()$;
                   $w := h(w)$;
                 }
```

Fig. 6. The predicate p in this schema lies in $\mathcal{W}preds_S(\omega)$ but not in $\mathcal{W}preds_S(v)$, and $q \in \mathcal{W}preds_S(v)$ and $h \in \mathcal{W}funcs_S(q)$, but h is not in $\mathcal{W}funcs_S(v)$

10 Conclusions and Suggestions for Further Work

Given a schema S that is linear, free, and near-liberal, such that for every while predicate q in S and every constant assignment $w := g()$; lying in the body of q in S, no other assignment to w also lies in the body of q, we have proved that it is decidable whether S is equivalent to a given subschema of S. We have shown, by Theorem 21, that the schema of Figure 3 lies in this class of schemas, but is not liberal, owing to its assignment $v := g_1()$;, and therefore, when applied to the subclass of schema pairs in which one schema is a subschema of the other, our main theorem is a true generalisation of the corresponding result for linear, free, liberal schemas[7,4].

Additionally, we have shown in Part (2) of Theorem 39 that for any linear schema S just having the 'non-sharing' condition on assignments, we can compute smaller weakly equivalent subschemas than those given by Weiser's slicing algorithm. Parts (1) and (2) of Theorem 39 can undoubtedly be generalised to arbitrary linear schemas by altering the definitions of the sets $\mathcal{W}preds_S(u)$ and $\mathcal{W}funcs_S(u)$ for $u \in \mathcal{V} \cup \{\omega\}$; we have not done this because the condition on constant assignments is needed for Part (3) of the Theorem.

It would be of interest to study the time complexity of computing the sets $\mathcal{W}preds_S(u)$ and $\mathcal{W}funcs_S(u)$, since a tractability result would increase the significance of our Theorem. Imposing syntactic conditions on the class of schemas considered, such as putting a constant upper bound on the depth of any while predicate, may make this possible.

Further work could focus on discarding the uniqueness requirement on constant assignments lying in the body of a while predicate. Lemmas 22 and 23, on which the later results rely, do not assume this hypothesis, so an attempt to generalise our main theorem without assuming it seems reasonably likely to succeed. In addition, the near-liberal condition that we have introduced in this paper can probably be relaxed by allowing a larger set of terms to be exempt from the liberality condition. For example, arbitrary constant terms, such as those like $f(g(), h())$, which contain no variables, could be treated the same way as terms of the form $g()$ in this paper.

Acknowledgements

This work was supported by a grant from the Engineering and Physical Sciences Research Council, Grant EP/E002919/1.

References

[1] S. Greibach, Theory of program structures: schemes, semantics, verification, Vol. 36 of Lecture Notes in Computer Science, Springer-Verlag Inc., New York, NY, USA, 1975.

[2] M. Weiser, Program slices: Formal, psychological, and practical investigations of an automatic program abstraction method, PhD thesis, University of Michigan, Ann Arbor, MI (1979).

[3] M. Weiser, Program slicing, in: 5^{th} International Conference on Software Engineering, San Diego, CA, 1981, pp. 439–449.

[4] S. Danicic, M. Harman, R. Hierons, J. Howroyd, M. R. Laurence, Equivalence of linear, free, liberal, structured program schemas is decidable in polynomial time, Theoretical Computer Science 373 (1-2) (2007) 1–18.

[5] M. R. Laurence, Characterising minimal semantics-preserving slices of function-linear, free, liberal program schemas, Journal of Logic and Algebraic Programming 72 (2) (2005) 157–172.

[6] M. S. Paterson, Equivalence problems in a model of computation, Ph.D. thesis, University of Cambridge, UK (1967).

[7] M. R. Laurence, S. Danicic, M. Harman, R. Hierons, J. Howroyd, Equivalence of linear, free, liberal, structured program schemas is decidable in polynomial time, Tech. Rep. ULCS-04-014, University of Liverpool, electronically available at `http://www.csc.liv.ac.uk/research/techreports/` (2004).

[8] F. Tip, A survey of program slicing techniques, Tech. Rep. CS-R9438, Centrum voor Wiskunde en Informatica, Amsterdam (1994).

[9] D. W. Binkley, K. B. Gallagher, Program slicing, in: M. Zelkowitz (Ed.), Advances in Computing, Volume 43, Academic Press, 1996, pp. 1–50.

[10] S. Danicic, Dataflow minimal slicing, PhD thesis, University of North London, UK, School of Informatics (Apr. 1999).

[11] E. A. Ashcroft, Z. Manna, Translating program schemas to while-schemas, SIAM Journal on Computing 4 (2) (1975) 125–146.

[12] Y. I. Ianov, The logical schemes of algorithms, in: Problems of Cybernetics, Vol. 1, Pergamon Press, New York, 1960, pp. 82–140.

[13] J. D. Rutledge, On Ianov's program schemata, J. ACM 11 (1) (1964) 1–9.

[14] H. B. Hunt, R. L. Constable, S. Sahni, On the computational complexity of program scheme equivalence, SIAM J. Comput 9 (2) (1980) 396–416.

[15] V. K. Sabelfeld, An algorithm for deciding functional equivalence in a new class of program schemes, Journal of Theoretical Computer Science 71 (1990) 265–279.

[16] Z. Manna, Mathematical Theory of Computation, McGraw–Hill, 1974.