Finiteness Conditions for Fixed Point Iteration

(Extended Abstract)

Flemming Nielson, Hanne Riis Nielson

Computer Science Department, Aarhus University
Ny Munkegade, DK-8000 Aarhus C
Denmark

E-mail: {fnielson,hrnielson}@daimi.aau.dk

Abstract

This paper provides a link between the formulation of static program analyses using the framework of abstract interpretation (popular for functional languages) and using the more classical framework of data flow analysis (popular for imperative languages). In particular we show how the classical notions of fastness, rapidity and k-boundedness carry over to the abstract interpretation framework and how this may be used to bound the number of times a functional should be unfolded in order to yield the fixed point. This is supplemented with a number of results on how to calculate the bounds for iterative forms (as for tail recursion), for linear forms (as for one nested recursive call), and for primitive recursive forms. In some cases this improves the "worst case" results of [9], but more importantly it gives much better "average case" results.

1 Introduction

In a recent paper [9] we gave precise bounds on a number k such that

$$FIX\ H = H^k\bot$$

where

$$H : (A \to B) \to (A \to B)$$

is a continuous functional corresponding to some static program analysis (or abstract interpretation). Here A and B are finite complete lattices and

$$FIX = \lambda H. \bigsqcup\{H^n\bot \mid n \geq 0\}$$

is the least fixed point operator. The development of [9] was sufficiently general to apply to *imperative languages*, where $A = B$ is a natural choice, as well as *functional languages*, where $A = \mathbf{2}^p$ and $B = \mathbf{2}$ is a natural choice for first-order strictness analysis.

In this paper we investigate functionals H of the form

$$H\ h = g_0 \sqcup (G\ h)$$
$$\text{where e.g. } G\ h = g \circ h \circ g_1$$

and show that it is often possible to use special properties of g_0, g and g_1 to obtain considerably lower bounds than the "worst-case" results of [9][1]. Even for the case of iterative forms, where $g = id$, this may result in better bounds than those of [9, Section 4] provided that g_1 is sufficiently well-behaved.

However, one of the most interesting aspects of this work is that the kind of properties (of

[1]This is not intended to say that the "worst-case" bounds of [9] are wrong but rather that they may not arise for functionals H of the form considered in this paper.

g_0, g, g_1) considered in this paper are close to the kind of properties studied in classical flow analysis [5]; this includes properties like fastness and k-boundedness. Thus at long last we seem to be able to bridge the gap between the fixed point techniques of the functional world and the imperative world! This was left open in [9] and [5, p.129] simply states: "determination of program properties by application of approximating semantics is often called abstract interpretation; this approach is *formally* (although *not conceptually*) equivalent to the algebraic framework approach presented [in [5]]".

Overview of paper

We begin by reviewing the lattice-theoretic notions that we will need (Section 2). Then we formulate the notions of fastness and k-boundedness [5] in the notation of abstract interpretation and we explore a few of their consequences (Section 3).

In Section 4 we then study "iterative forms" which are functionals H of the form displayed above but with $g = id$. We consider three cases, depending on whether the functions from A to B are

- total (written $A \rightarrow_t B$)
- monotone (written $A \rightarrow_m B$)
- strict and additive[2] (written $A \rightarrow_{sa} B$)

In short we prove that the functional G is $\ell_\varphi(g_1)$-bounded where φ is one of t, m or sa and where ℓ_φ is some measure on functions. In practical terms this means that $FIX\ H = H^k\bot$ whenever $k \geq \ell_\varphi(g_1)$; thus only $\ell_\varphi(g_1)$ unfoldings of H are needed.

In Section 5 we then study how to extend these results to "linear forms" which are functionals H of the form displayed above but where g is *not* restrained to be *id*. A limitation of this development is that g must be strict and additive for the results of Section 3 to be applicable. We can then show that G is $\ell_\varphi(g) \cdot \ell_\varphi(g_1)$-bounded in the cases where φ is t or m. Unfortunately, the case where φ is sa eludes us.

[2]In [5] additivity is called "distributivity" and complete additivity (which is equivalent to strictness and additivity given the finiteness of A and B) is called "continuity".

The applicability of the results on "iterative forms" is further extended in Section 6. Here we show that the program transformation technique of "accumulator introduction" may often be used to transform a functional in "primitive recursive form" into one in "iterative form". Furthermore, if k unfoldings suffice for the transformed functional then also k unfoldings suffice for the original functional.

Finally, we compare our results with those of [9] and suggest directions for further research (Section 7). In the Appendix we characterize the "iterative forms", "linear forms" and "primitive recursive forms" in terms of the functional programs and analyses for which they naturally arise.

The full version appeared as [10].

2 Preliminaries

A *finite complete lattice* $(L, \sqsubseteq)$ is a finite set L, together with a partial order $\sqsubseteq$, such that each subset Y of L has a least upper bound; since a least upper bound is unique (if it exists) it makes sense to write $\bigsqcup Y$ for it. It is customary to write $\bot = \bigsqcup \emptyset$ for the least element and $\top = \bigsqcup L$ for the greatest element of L and to write $l_1 \sqcup l_2$ for $\bigsqcup\{l_1, l_2\}$. When $(L, \sqsubseteq)$ is a finite complete lattice each subset Y of L will also have a greatest lower bound; it is customary to write $l_1 \sqcap l_2$ for the greatest lower bound of the set $\{l_1, l_2\}$. We shall write

$$l_1 \sqsubset l_2 \text{ for } l_1 \sqsubseteq l_2 \land\ l_1 \neq l_2$$

and we shall say that a chain

$$l_0 \sqsubset l_1 \sqsubset \ldots \sqsubset l_k$$

has *length* k (rather than $k+1$). It is convenient to write

$\mathbf{C}(L)$ for the *cardinality* of L

$\mathbf{H}(L)$ for the *height* of L (i.e. the length of the longest chain)

Writing $L \times L'$ for the cartesian product and L^n for the n-fold product ($n \geq 1$) we have $\mathbf{C}(L \times L') = \mathbf{C}(L) \cdot \mathbf{C}(L')$, $\mathbf{C}(L^n) = \mathbf{C}(L)^n$, $\mathbf{H}(L \times L') = \mathbf{H}(L) + \mathbf{H}(L')$ and $\mathbf{H}(L^n) = n \cdot \mathbf{H}(L)$. Writing

$$\mathbf{2} = \begin{tikzcd} \bullet\, 1 \arrow[d, no head] \\ \bullet\, 0 \end{tikzcd}$$

we have $\mathbf{C}(\mathbf{2}) = 2$ and $\mathbf{H}(\mathbf{2}) = 1$.

A function $f : L \to L'$ from a finite complete lattice $L = (L, \sqsubseteq)$ to a finite complete lattice $L' = (L', \sqsubseteq)$ is *monotone* if

$$l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$$

and is *strict* if

$$f(\bot) = \bot$$

and is *additive* if

$$f(l_1 \sqcup l_2) = f(l_1) \sqcup f(l_2)$$

Fact 1: A monotone function f between finite complete lattices is *continuous*, i.e.

$$f(\bigsqcup\{l_n \mid n \geq 0\}) = \bigsqcup\{f(l_n) \mid n \geq 0\}$$

whenever $\forall n : l_n \sqsubseteq l_{n+1}$. □

It is well known that

$$FIX\ f = \bigsqcup\{f^n(\bot) \mid n \geq 0\}$$

denotes the *least fixed point* of f whenever f is continuous; it thus follows that if L is a finite complete lattice and $f : L \to L$ is monotone then $FIX\ f$ is the least fixed point of f.

Fact 2: A strict and additive function f between finite complete lattices is *completely additive*, i.e.

$$f(\bigsqcup Y) = \bigsqcup\{f(l) \mid l \in Y\}$$

for all subsets Y. □

We shall write $L \to_t L'$, $L \to_m L'$ and $L \to_{sa} L'$ for the sets of total, monotone, and strict and additive functions from L to L', respectively. The partial order is defined componentwise, i.e.

$$f \sqsubseteq f' \text{ if and only if } \forall l : f(l) \sqsubseteq f'(l)$$

and all of $L \to_t L'$, $L \to_m L'$ and $L \to_{sa} L'$ will be finite complete lattices if L and L' are. In all cases

$$(\bigsqcup \mathcal{F}) = \lambda l.\ \bigsqcup\{f(l) \mid f \in \mathcal{F}\}$$

is the formula for least upper bounds.

Finally, a finite complete lattice L is *distributive* [4] if

$$l_1 \sqcap (l_2 \sqcup l_3) = (l_1 \sqcap l_2) \sqcup (l_1 \sqcap l_3)$$

for all $l_1, l_2, l_3 \in L$. An alternative characterization was given in [9, Lemma 17].

Fact 3: If L and L' are distributive so are $L \times L'$, L^n, $L \to_t L'$, $L \to_m L'$ and $L \to_{sa} L'$. □

3 Finiteness conditions on functionals

Let A and B be arbitrary finite complete lattices and consider continuous (i.e. monotone) functionals

$$G,\ H : (A \to_\varphi B) \to (A \to_\varphi B)$$

where φ is any one of t, m or sa. Throughout this paper we shall assume that

$$H\ h = g_0 \sqcup (G\ h)$$

but in this section we shall not make any assumptions about the form of G. We are interested in determining the least fixed point, $FIX\ H$, of H but before doing so we need a few definitions from classical flow analysis (as surveyed in [5]).

As usual $G^i(h)$ denotes the i-fold iteration of G so that $G^0(h) = h$ and $G^{i+1}(h) = G(G^i(h)) = G^i(G(h))$. It is convenient to write Id for the functional defined by $Id(h) = h$. We then define

$$G^{[i]}(h) = \bigsqcup\{G^j(h) \mid 0 \leq j < i\}$$

This is well-defined given the assumptions on A and B and the results of Section 2. Concerning $G^{[i+1]}$ one may calculate that

$$G^{[i+1]}(h) = h \sqcup \cdots \sqcup G^i(h)$$

so that $G^{[i+1]} = Id \sqcup (G^{[i]} \circ G)$. We also have

Fact 4: If G is strict and additive then $G^{[i+1]} = Id \sqcup (G \circ G^{[i]})$. □

Fact 5: If $G \sqsupseteq Id$ then $G^{[i+1]} = G^i \sqsupseteq Id$; it follows that also $G^{[i+2]} = Id \sqcup (G \circ G^{[i+1]})$. □

Clearly $G \sqsubseteq G \sqcup Id$ so that also $G^{[i]} \sqsubseteq (G \sqcup Id)^{[i]}$ for all i. For equality we note

Fact 6: $G^{[i]} = (G \sqcup Id)^{[i]}$ if G is strict and additive (or if $G \sqsupseteq Id$). □

The interest in $G^{[i]}$ can now be motivated by:

Lemma 7: $FIX\ H = \bigsqcup\{(G \sqcup Id)^{[i]}(g_0) \mid i \geq 0\}$

Corollary 8: If G is strict and additive we have

$$FIX\ H = \bigsqcup\{G^{[i]}(g_0) \mid i \geq 0\}$$

(This also holds if $G \sqsupseteq Id$.) □

Following [5] we shall say that G is *k-bounded* if

$$\forall h \in A \rightarrow_\varphi B : \ G^{[k+1]}(h) = G^{[k]}(h)$$

There is a related concept called k-semi-boundedness but for strict (and additive) functionals it is equivalent to k-boundedness and so will not be of interest in this paper. The special case of 2-boundedness is known as *fastness*; in [5] this is motivated by the observation that for a *fast* problem only one iteration around the loop body will be needed.

Fact 9: If G is k-bounded then $G^{[i]}(h) = G^{[k]}(h)$ for all $i \geq k$ and for all $h \in A \rightarrow_\varphi B$. □

Turning to the consequences for the least fixed point, $FIX\ H$, of H we then have:

Fact 10: If G is strict and additive then $H^i\bot = G^{[i]}(g_0)$. □

Lemma 11: If G is strict and additive and G is k-bounded then

$$FIX\ H = G^{[k]}(g_0) = H^k(\bot)$$

The k-boundedness of G may be weakened to $G^{[k+1]}(g_0) = G^{[k]}(g_0)$. □

For the applicability of the results of the present paper, in particular Lemma 11, it is important that the functional G is

- k-bounded, and
- strict and additive

We will develop formulae for determining the constant k in Sections 4, 5 and 6.

Here we will conclude with a few examples of how to manufacture strict and additive functionals. Consider the definitions of functionals $G : (A \rightarrow_\varphi B) \rightarrow (A \rightarrow_\varphi B)$ shown in Figure 1. Then G is strict and additive in all cases. We shall use the first and the two last observations in Sections 4, 5 and 6 respectively.

4 Iterative forms

We now study a functional $G : (A \rightarrow_\varphi B) \rightarrow (A \rightarrow_\varphi B)$ defined by

$$G\ h = h \circ g_1$$

for $g_1 \in A \rightarrow_\varphi B$. Clearly G is continuous (i.e. monotone), strict and additive; this means that for H defined by $H\ h = g_0 \sqcup (G\ h)$ we will have $FIX\ H = G^{[k]}(g_0) = H^k(\bot)$ whenever G is k-bounded. In this section we shall define three measures on functions (denoted ℓ_t, ℓ_m and ℓ_{sa}) and we shall show that G will be $\ell_\varphi(g_1)$-bounded. This will be illustrated on an accumulator version of the factorial program.

The first case is where φ is t. Here we define

$$\begin{aligned} \ell_t(f,x) = &\\ &\min\{k \mid f^k(x) \in \{x, \ldots, f^{k-1}(x)\},\\ &\quad k > 0\}\\ \ell_t(f) = &\ \max\{\ell_t(f,x) \mid x \in A\} \end{aligned}$$

and note

$$\begin{array}{lll}
G\,h & = & h \circ g_1 \\
G\,h & = & g \circ h \quad \text{where } g \text{ is strict and additive} \\
G & = & G_1 \circ G_2 \quad \text{where } G_1 \text{ and } G_2 \text{ are strict and additive} \\
G & = & G_1 \sqcup G_2 \quad \text{where } G_1 \text{ and } G_2 \text{ are strict and additive} \\
G\,h & = & g \sqcap h \quad \text{where } A \rightarrow_{\varphi} B \text{ is distributive} \\
G\,h & = & \mathtt{tuple}(G_1\,h, G_2\,h) = \lambda l. < G_1\,h\,l, G_2\,h\,l > \\
 & & \quad \text{where } G_1 \text{ and } G_2 \text{ are strict and additive} \\
G\,h & = & g \circ h \circ g_1 \quad \text{where } g \text{ is strict and additive} \\
G\,h & = & g \circ \mathtt{tuple}(h \circ g_1, g_2) \\
 & & \quad \text{where } g \text{ is strict and additive in its left} \\
 & & \quad \text{argument, i.e. } g(\bot, l) = \bot \text{ and} \\
 & & \quad g(l_1 \sqcup l_2, l) = g(l_1, l) \sqcup g(l_2, l)
\end{array}$$

Figure 1: Typical forms of strict and additive G

Fact 12: $1 \leq \ell_t(f,x) \leq \ell_t(f) \leq \mathbf{C}(A)$ when $f : A \rightarrow_t A$ and $x \in A$. □

We then have

Lemma 13: The functional $G : (A \rightarrow_t B) \rightarrow (A \rightarrow_t B)$ defined by $G\ h = h \circ g_1$ is $\ell_t(g_1)$-bounded. □

Proof: Setting $k = \ell_t(g_1)$ we must prove that $G^{[k+1]} = G^{[k]}$ and for this it suffices to prove $G^k \sqsubseteq G^{[k]}$. So let $h \in A \rightarrow_t B$ and $w \in A$ be given and note that
$$G^k\ h\ w = h(g_1^k(w))$$
and that
$$G^{[k]}\ h\ w = \bigsqcup\{h(w), \ldots, h(g_1^{k-1}(w))\}$$
(where we used $k > 0$). From the definition of k there exists i such that $0 \leq i < k$ and $g_1^k(w) = g_1^i(w)$. Hence
$$G^k\ h\ w = h(g_1^i(w))$$
where $i \leq k - 1$ and the result is immediate. □

Example 14: An accumulator version of the factorial program may be written as follows

```
fac(n,a) = if n = 0 then a
                else fac(n-1,n*a)
```

where the initial call is `fac(n,1)`. We shall not go into the details of how to perform a strictness analysis for this program but simply postulate (or see the Appendix) that it may be obtained as follows:

$$\begin{aligned}
&A = \mathbf{2} \times \mathbf{2} \\
&B = \mathbf{2} \\
&g_0(n^{\#}, a^{\#}) = n^{\#} \sqcap a^{\#} \\
&g_1(n^{\#}, a^{\#}) = (n^{\#}, n^{\#} \sqcap a^{\#})
\end{aligned}$$

where $n^{\#} \sqcap a^{\#}$ denotes the greatest lower bound of $\{n^{\#}, a^{\#}\}$, i.e.

$$n^{\#} \sqcap a^{\#} = \begin{cases} 1 & \text{if } n^{\#} = 1 = a^{\#} \\ 0 & \text{otherwise} \end{cases}$$

Thus $G\ h = h \circ g_1$ and $H\ h = g_0 \sqcup (G\ h)$ and we must determine k such that $FIX\ H = G^{[k]}(g_0) = H^k\bot$.

The results of [9, Section 3] are applicable and ensure that one may take

$$k = \mathbf{H}(A \rightarrow_m B) = \mathbf{C}(A) \cdot \mathbf{H}(B) = 4 \cdot 1 = 4$$

Using Lemmas 13 and 11 we may improve this by taking
$$k = \ell_t(g_1) = 2$$

That $\ell_t(g_1) = 2$ follows easily from the idempotence of g_1, i.e. $g_1 \circ g_1 = g_1$.

This result is not optimal, however. Tabulating $H^i\bot\ w$ for $i \in \{0,1,2\}$ and $w \in A = \{(0,0),(0,1),(1,0),(1,1)\}$ one gets

w	(0,0)	(0,1)	(1,0)	(1,1)
$H^0\bot\ w$	0	0	0	0
$H^1\bot\ w$	0	0	0	1
$H^2\bot\ w$	0	0	0	1

and this shows that it is possible to use $k = 1$ in this example. □

The second case is where φ is m. Here we first need to define the set $LC(Y)$ of elements less than (or equal to) some element of Y, i.e.

$$LC(Y) = \{x \in A \mid \exists\, x' \in Y : x \sqsubseteq x'\}$$

We then define

$$\begin{aligned}
&\ell_m(f,x) = \\
&\quad \mathtt{min}\{k \mid f^k(x) \in LC\{x,\ldots,f^{k-1}(x)\}, \\
&\qquad\qquad k > 0\} \\
&\ell_m(f) = \ \mathtt{max}\{\ell_m(f,x) \mid x \in A\}
\end{aligned}$$

Fact 15: $1 \leq \ell_m(f,x) \leq \ell_m(f) \leq \mathbf{C}(A)$ and $\ell_m(f) \leq \ell_t(f)$ when $f : A \rightarrow_m A$ and $x \in A$. □

We then have

Lemma 16: The functional $G : (A \rightarrow_m B) \rightarrow (A \rightarrow_m B)$ defined by $G\ h = h \circ g_1$ is $\ell_m(g_1)$-bounded. □

Proof: Setting $k = \ell_m(g_1)$ we must prove that $G^{[k+1]} = G^{[k]}$ and for this it is sufficient to prove $G^k \sqsubseteq G^{[k]}$. So let $h \in A \rightarrow_m B$ and $w \in A$ be given and note that

$$G^k\ h\ w = h(g_1^k(w))$$

and that

$$G^{[k]}\ h\ w = \bigsqcup\{h(w),\ldots,h(g_1^{k-1}(w))\}$$

(where we used $k > 0$). From the definition of k there exists i such that $0 \leq i < k$ and $g_1^k(w) \sqsubseteq g_1^i(w)$. Hence

$$G^k\ h\ w \sqsubseteq h(g_1^i(w))$$

where $i \leq k-1$ and the result is immediate. □

Example 17: Continuing the previous example we now may take

$$k = \ell_m(g_1) = 1$$

which is the optimal result. To see that $\ell_m(g_1) = 1$ simply note that g_1 is *reductive*, i.e. $g_1 \sqsubseteq id$. □

The third case is where φ is sa. Here we define

$$\begin{aligned}
&\ell_{sa}(f,x) = \\
&\quad \mathtt{min}\{k \mid f^k(x) \in \bigsqcup\{x,\ldots,f^{k-1}(x)\}, \\
&\qquad\qquad k > 0\} \\
&\ell_{sa}(f) = \ \mathtt{max}\{\ell_{sa}(f,x) \mid x \in A\}
\end{aligned}$$

and note

Fact 18: $1 \leq \ell_{sa}(f,x) \leq \ell_{sa}(f) \leq \ \mathbf{H}(A)$ and $\ell_{sa}(f) \leq \ell_m(f)$ whenever $f \in A \rightarrow_{sa} A$ and $x \in A$; here we assume that $\mathbf{H}(A) > 0$ is indeed the case. □

We then have

Lemma 19: The functional $G : (A \rightarrow_{sa} B) \rightarrow (A \rightarrow_{sa} B)$ defined by $G\ h = h \circ g_1$ is $\ell_{sa}(g_1)$-bounded. □

Proof: Setting $k = \ell_{sa}(g_1)$ we must prove that $G^{[k+1]} = G^{[k]}$ and for this it is sufficient to prove $G^k \sqsubseteq G^{[k]}$. So let $h \in A \rightarrow_{sa} B$ and $w \in A$ be given and note that

$$G^k h\ w = h(g_1^k(w))$$

and that

$$\begin{aligned}
G^{[k]}h\ w &= \bigsqcup\{h(w),\ldots,h(g_1^{k-1}(w))\} \\
&= h(\bigsqcup\{w,\ldots,g_1^{k-1}(w)\})
\end{aligned}$$

where we have used that $k > 0$ and that h is completely additive. From the definition of k we have

$$g_1^k(w) \sqsubseteq \bigsqcup\{w,\ldots,g_1^{k-1}(w)\}$$

and (by monotonicity of h) the result follows. □

From Fact 18 and Lemmas 19 and 11 it follows that

$$FIX\ H = H^k\bot \text{ for } k = \mathbf{H}(A)$$

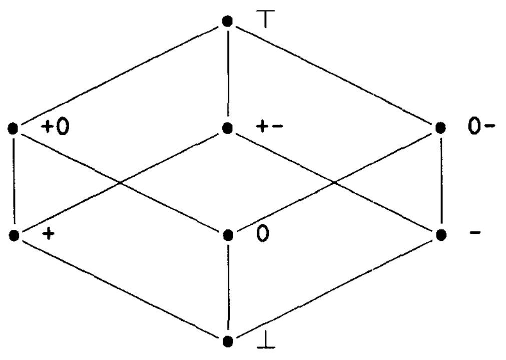
in the strict and additive case. It is interesting to note that this is the (tight) upper bound established in [9, Section 4] for functionals in iterative form.

Example 20: Continuing the previous example we note that

$$\ell_{sa}(g_1) = 1 = \ell_m(g_1);$$

however, g_1 is not additive so the above lemma is not applicable. □

Example 21: In certain cases the bound obtained from Lemma 19 will indeed be better than that obtained from Lemma 16. To illustrate this we shall consider a detection of signs analysis which is a typical example of an analysis in the strict and additive framework. The analysis will be based on the lattice **S** of signs depicted below:



Here + describes the positive natural numbers, 0 the natural number 0, +0 the non-negative natural numbers etc.

Now consider the function

```
f x = if sq x > 100 then x
      else f(sq(succ x)-3*(succ x))
```

The analysis of f will give rise to a functional

$$H : (\mathbf{S}\to\mathbf{S}) \to (\mathbf{S}\to\mathbf{S})$$

in iterative form, i.e.

$$H\ h = g_0 \sqcup h \circ g_1$$

Using the "obvious" interpretation of the primitives **sq**, >, **succ**, - and * we may obtain the following definitions of g_0 and g_1:

x	⊥	+	0	-	+0	+-	0-	⊤
$g_0(x)$	⊥	+	⊥	-	+0	+-	0-	⊤
$g_1(x)$	⊥	⊤	⊤	0+	⊤	⊤	⊤	⊤

We have

$$\ell_t(g_1) = \ell_m(g_1) = 3$$

so Lemma 16 gives that 3 iterations will suffice for computing the fixed point. However

$$\ell_{sa}(g_1) = 2$$

so Lemma 19 shows than an even better bound can be obtained. □

Sometimes simple "algebraic" properties of functions, like idempotency, may be used to obtain bounds on ℓ_φ. Figure 2 summarizes some such results; whenever

- $f \in A \to_\varphi A$
- f satisfies the condition in the left column

the corresponding table entry gives

- an upper bound on $\ell_\varphi(f)$

5 Linear forms

We now increase our level of ambition and study a functional

$$G : (A \to_\varphi B) \to (A \to_\varphi B)$$

	$\varphi = t$	$\varphi = m$	$\varphi = sa$
$f = f$ (worst-case)	$\mathbf{C}(A)$	$\mathbf{C}(A)$	$\mathbf{H}(A)$
$f = id$ (trivial case)	1	1	1
$f = f \circ f$ (idempotence)	2	2	2
$f \sqsupseteq id$ (extensive)	$\mathbf{H}(A) + 1$	$\mathbf{H}(A) + 1$	$\mathbf{H}(A)$
$f \sqsubseteq id$ (reductive)	$\mathbf{H}(A) + 1$	1	1
$f \circ f \sqsubseteq f$ (weak idempotence)	$\mathbf{C}(A)$	2	2
$f \circ f \sqsubseteq f \sqcup id$ (fast)	$\mathbf{C}(A)$	$\mathbf{C}(A)$	2
$f^k = f^i$ for $i < k$	k	k	k
$f^k \sqsubseteq f^i$ for $i < k$	$\mathbf{C}(A)$	k	k
$f^k \sqsubseteq f^{[k]}$	$\mathbf{C}(A)$	$\mathbf{C}(A)$	k

Figure 2: Example measures on functions

defined by

$$G\ h = g \circ h \circ g_1$$

for $g \in B \to_\varphi B$ and $g_1 \in A \to_\varphi A$. This functional is continuous (i.e. monotone) if g is monotone.

For the results of Section 3 to be applicable we need G not only to be continuous but also strict and additive and this holds when g is strict and additive; then the function H defined by $H\ h = g_0 \sqcup (G\ h)$ will have $FIX\ H = G^{[k]}(g_0) = H^k(\bot)$ whenever G is k-bounded. However, in this section we shall not formally assume that g is strict and additive as the results on k-boundedness of G do not depend on this.

The first case is where φ is t. Using the measure ℓ_t of the previous section we have

Lemma 22: The functional $G : (A \to_t B) \to (A \to_t B)$ defined by $G\ h = g \circ h \circ g_1$ is $\ell_t(g) \cdot \ell_t(g_1)$-bounded. □

The proof makes use of an auxiliary notion. For a function $f \in A \to_t B$ and an element $x \in A$ define

$$\varrho_t(f,x) = \max\{i \mid f^k(x) \in \{x, f(x), \ldots, f^{k-i}(x)\}, k = \ell_t(f,x)\}$$

and note that

$$1 \le \varrho_t(f,x) \le \ell_t(f,x).$$

Fact 23: If $k \ge \ell_t(f,x)$ then $f^k(x) = f^{k-\varrho_t(f,x)}(x)$. □

Fact 24: If $k - n \cdot \varrho_t(f,x) \ge \ell_t(f,x) - \varrho_t(f,x)$ and $n \ge 0$ then $f^k(x) = f^{k-n\cdot\varrho_t(f,x)}(x)$. □

We refer to [10] for a proof of Lemma 22.

The second case is where φ is m. Using the measure ℓ_m of the previous section we get

Lemma 25: The functional $G : (A \to_m B) \to (A \to_m B)$ defined by $G\ h = g \circ h \circ g_1$ (for $g \in B \to_m B$ and $g_1 \in A \to_m A$) is $\ell_m(g) \cdot \ell_m(g_1)$-bounded. □

Again we shall need som auxiliary results. Define

$$\varrho_m(f,x) = \max\{i \mid f^k(x) \in LC\{x, f(x), \ldots, f^{k-i}(x)\}, k = \ell_m(f,x)\}$$

and observe that

$$1 \le \varrho_m(f,x) \le \ell_m(f,x)$$

Fact 26: If $k \ge \ell_m(f,x)$ then $f^k(x) \sqsubseteq f^{k-\varrho_m(f,x)}(x)$. □

Fact 27: If $k - n \cdot \varrho_m(f,x) \ge \ell_m(f,x) - \varrho_m(f,x)$ and $n \ge 0$ then $f^k(x) \sqsubseteq f^{k-n\cdot\varrho_m(f,x)}(x)$. □

We refer to [10] for a proof of Lemma 25.

One can prove that that Lemma 22 can be strengthened to show that if $G : (A \to_t B) \to (A \to_t B)$ is defined by $G\ h = g \circ h \circ g_1$ for $g \in A \to_m B$ then G is $\ell_m(g) \cdot \ell_t(g_1)$-bounded. Knowledge of $g_1 \in A \to_m B$ cannot be used because the argument to G need not be monotone.

Example 28: Consider once again the detection of signs analysis of Example 21. The function

```
f x = if sq x > 100 then x
      else sq(f(x-3))
```

gives rise to a functional

$$H : (\mathbf{S}\to\mathbf{S}) \to (\mathbf{S}\to\mathbf{S})$$

in linear form, i.e.

$$H\ h = g_0 \sqcup g \circ h \circ g_1$$

The functions g_0, g and g_1 may be defined by

x	$\perp$	+	0	-	+0	+-	0-	$\top$
$g_0(x)$	$\perp$	+	$\perp$	-	+0	+-	0-	$\top$
$g(x)$	$\perp$	+	0	+	+0	+	+0	+0
$g_1(x)$	$\perp$	$\top$	-	-	$\top$	$\top$	- ·	$\top$

Note that g is strict and additive so that Lemma 11 applies. We get

$$\ell_t(g) = \ell_m(g) = 2$$
$$\ell_t(g_1) = \ell_m(g_1) = 2$$

so that both Lemmas 22 and 25 yield a bound of 4 on the number of iterations needed. This is substantially better than the results of [9] where we obtain a bound of 9 (when using that the detection of signs analysis is in the strict and additive framework). □

The third case is where φ is sa. Here we would like to show that G is $\ell_{sa}(g)\cdot\ell_{sa}(g_1)$-bounded but so far we have been unable to do so. Also we would like to strengthen Lemmas 22 and 25 to $\ell_{sa}(g)\cdot\ell_t(g_1)$ and $\ell_{sa}(g)\cdot\ell_m(g_1)$, respectively, provided that $g \in B \to_{sa} B$.

6 Primitive recursive forms

We now study a functional $G : (A \to_\varphi B) \to (A \to_\varphi B)$ defined by

$$G\ h = g \circ \mathtt{tuple}(h \circ g_1, g_2)$$

where $g \in B \times B \to_\varphi B$, $g_1 \in A \to_\varphi A$ and $g_2 \in A \to_\varphi B$. As already noted in Section 3, G is strict and additive if g is strict and additive in its left argument.

Rather than embarking on a detailed study of the iterands of

$$H = \lambda h. g_0 \sqcup (G\ h)$$

we shall transform H into another functional H', by using the well-known program transformation technique of "introducing an accumulator" [2, Section 6]. The functional $H' : (A \times B \to_\varphi B) \to (A \times B \to_\varphi B)$ will be in iterative form and is defined by

$$H'\ h' = g_0' \sqcup (G'\ h')$$
$$G'\ h' = h' \circ g_1'$$

where $g_0' \in A{\times}B \to_\varphi B$ and $g_1' \in A{\times}B \to_\varphi A{\times}B$ are defined by

$$g_0' = g \circ \mathtt{tuple}(g_0 \circ fst, snd)$$

$$g_1' = \mathtt{tuple}(g_1 \circ fst, g \circ \mathtt{tuple}(g_2 \circ fst, snd))$$

The formal relationship between H and H' is expressed by

Fact 29: Assume that $g \in B \times B \to_\varphi B$ satisfies

- g is *associative*, i.e. $g(w_1, g(w_2, w_3)) = g(g(w_1, w_2), w_3)$, and
- g is strict and additive in its left argument.

Then

$$H'^i \perp (w, w_1) = g(H^i \perp w, w_1)$$

holds for all $i \geq 0, w \in A$ and $w_1 \in B$. □

Fact 30: Assume that $g \in B \times B \to_\varphi B$ is associative, strict and additive in its left argument and that

- g has a *right-identity* $w_0 \in B$, i.e. $g(w_1, w_0) = w_1$.

We then have

- $H^i \bot\ w = H'^i \bot\ (w, w_0)$
- $FIX\ H\ w = FIX\ H'\ (w, w_0)$

for all $i \geq 0$ and $w \in A$. □

Fact 30 relates the fixed points of the two functionals but we shall also be interested in determining the number of unfoldings needed to compute the fixed points. We have

Lemma 31: Assume that g is associative, has a right identity and is strict and additive in its left argument. If k unfoldings suffice for H' then also k unfoldings suffice for H, i.e.

$$FIX\ H' = H'^k \bot \text{ implies } FIX\ H = H^k \bot$$

In particular, we have $FIX\ H = H^k \bot$ if G' is k-bounded. □

A similar result holds for the bounds of the functionals G and G'.

Lemma 32: Assume that g is associative, has a right identity and is strict and additive in its left argument. Then G and G' are strict and additive and G' is k-bounded implies G is k-bounded. □

Example 33: Consider the factorial function defined by

```
fac n = if n = 0 then 1
        else n * fac(n-1)
```

A strictness analysis (along the lines of the Appendix) will give rise to a functional

$$H : (\mathbf{2} \to_m \mathbf{2}) \to (\mathbf{2} \to_m \mathbf{2})$$

in primitive recursive form and with

$$g_0(n^\#) = 1$$
$$g_1 = g_2 = id$$
$$g(n^\#, m^\#) = n^\# \sqcap m^\#$$

To bound the number of iterations needed to compute the fixed point of H it follows from Lemmas 31 and 16 that we only have to determine $\ell_m(g_1')$. We have

$$\begin{aligned} g_1'(n^\#, m^\#) &= \mathtt{tuple}(g_1 \circ fst, \\ &\quad\ g \circ \mathtt{tuple}(g_2 \circ fst, snd))(n^\#, m^\#) \\ &= (n^\#, n^\# \sqcap m^\#) \end{aligned}$$

Since g_1' is reductive we have $\ell_m(g_1') = 1$.

This is clearly the optimal result. It is worth observing that it is also better than the bound of 2 obtainable from [9, Section 2]. Finally we should point out that the functional H' of the present example corresponds rather closely to the functional H of Examples 17 and 14. (Exact correspondance fails because g_0 is not strict.) □

7 Conclusion

We have considered the problem of bounding the number of iterations needed to compute the fixed point of a continuous functional

$$H : (A \to B) \to (A \to B)$$

defined on finite complete lattices A and B. We have considered three defining forms of H:

- iterative forms: $H\ h = g_0 \sqcup (h \circ g_1)$
- linear forms: $H\ h = g_0 \sqcup (g \circ h \circ g_1)$
- primitive recursive forms: $H\ h = g_0 \sqcup (g \circ \mathtt{tuple}(h \circ g_1, g_2))$

and three classes of functions from A to B:

- total functions
- monotone functions
- strict and additive functions

A related study was conducted in [9]. However, the main difference is that the bounds of [9] depended on measures of A and B whereas the bounds established here depend on measures of the functions g_1 etc. The results of the present paper may therefore carry over to the situation where neither A nor B are *finite* complete lattices.

Figure 3 summarizes all the information that can be obtained by combining the results established here with those of [9]. In the formulae using min we have arranged it so that the first argument is the result of the present paper and the second argument is the result from [9].

We should explain that $\mathbf{RJC}(A)$ is the number of non-bottom join-irreducible elements of A; if A is distributive we have $\mathbf{RJC}(A) = \mathbf{H}(A)$ and in general $\mathbf{H}(A) \leq \mathbf{RJC}(A) \leq \mathbf{C}(A)$. Finally we should remark that the $\mathbf{H}(A)+\mathbf{H}(B)$ entry in the table follows from $\ell_{sa}(g_1') \leq \mathbf{H}(A \times B)$ where we use the results already established for iterative forms.

Thus for the iterative forms the bounds of the present paper will always be at least as good as those of [9] whereas this need not be the case for linear forms and primitive recursive forms. However, the main point is that in the "average" case we expect $\ell_t(g_1)$ to be much less than $\mathbf{C}(A)$ etc., so that in the "average" case we are likely always to get an improvement over [9]. This suggests studying certain analyses, e.g. corresponding to fast analyses, where this always can be guaranteed.

Acknowledgements

The insights needed to write this paper were greatly assisted by Henrik Andersen's talk on model checking at the First DART-Workshop [7] and by subsequent discussions with him. This research has been supported in part by The Danish Research Councils, under the DART-project.

References

[1] A.V. Aho, R. Sethi, J.D. Ullman: *Compilers - Principles, Techniques and Tools*, Addison - Wesley (1986).

[2] R.M. Burstall, J. Darlington: A Transformation System for Developing Recursive Programs, *Journal of the ACM* **24** *1* (1977).

[3] A.J. Field, P.G. Harrison: *Functional Programming*, Addison - Wesley (1988).

[4] G. Grätzer: *Lattice Theory - First Concepts and Distributive Lattices*, W.H. Freeman and Company (1971).

[5] T.J. Marlowe, B.G. Ryder: Properties of Data Flow Frameworks - A Unified Model, *Acta Informatica* **28** (1990).

[6] A. Mycroft: Abstract Interpretation and Optimizing Transformations for Applicative Programs, University of Edinburgh Ph.D.-thesis (1981).

[7] F. Nielson (editor): Design, Analysis and Reasoning about Tools: Abstracts from the First Workshop, Aarhus University report DAIMI PB-367 (1991).

[8] F. Nielson: Two-Level Semantics and Abstract Interpretation, *Theoretical Computer Science* **69** (1989).

[9] H.R. Nielson, F. Nielson: Bounded Fixed Point Iteration, *Proceedings of the ACM Symposium on Principles of Programming Languages* (1992).

[10] F. Nielson, H.R. Nielson: Finiteness Conditions for Fixed Point Iteration, report DAIMI PB-384, Aarhus University, Denmark (February 1992).

[11] H.R. Nielson, F. Nielson: Transformations on Higher-Order Functions, *Proceedings FPCA'89* (1989).

[12] P. Wadler, R.J.M. Hughes: Projections for Strictness Analysis, *Proceedings FPCA'87, Springer Lecture Notes in Computer Science* **274** (1987).

	$A \rightarrow_t B$	$A \rightarrow_m B$	$A \rightarrow_{sa} B$
iterative form	$\ell_t(g_1) \leq \mathbf{C}(A)$	$\ell_m(g_1) \leq \mathbf{C}(A)$	$\ell_{sa}(g_1) \leq \mathbf{H}(A)$
linear form (g restricted)	$\mathtt{min}\{\ell_t(g) \cdot \ell_t(g_1), \mathbf{C}(A) \cdot \mathbf{H}(B)\}$	$\mathtt{min}\{\ell_m(g) \cdot \ell_m(g_1), \mathbf{C}(A) \cdot \mathbf{H}(B)\}$	$\mathbf{RJC}(A) \cdot \mathbf{H}(B)$
prim. rec. form (g restricted)	$\mathtt{min}\{\ell_t(g_1'), \mathbf{C}(A) \cdot \mathbf{H}(B)\}$	$\mathtt{min}\{\ell_m(g_1'), \mathbf{C}(A) \cdot \mathbf{H}(B)\}$	$\mathtt{min}\{\ell_{sa}(g_1'), \mathbf{RJC}(A) \cdot \mathbf{H}(B), \mathbf{H}(A)+\mathbf{H}(B)\}$
no restriction	$\mathbf{C}(A) \cdot \mathbf{H}(B)$	$\mathbf{C}(A) \cdot \mathbf{H}(B)$	$\mathbf{RJC}(A) \cdot \mathbf{H}(B)$

Figure 3: Summary of results

Appendix

In this Appendix we shall claim that the results of the present paper are widely applicable because the special forms allowed for the functionals H and G are likely to arise frequently. The general idea is that the functional H will be

- in iterative form if the function being analysed is *tail recursive* (corresponding to an iterative loop),
- in linear form if the function being analysed contains only one recursive call in its defining equation (somewhat analogous to the linear forms of [3]),
- in primitive recursive form if the function being analysed is *primitive recursive.*

To be able to substantiate these claims we must make several assumptions on how we analyse the various primitives. To this end we shall assume that

- function composition is interpreted as function composition (so that only *forward* analyses are considered),
- tupling is interpreted as tupling meaning that the abstraction of a pair is a pair of abstractions[3],
- the conditional given by

$$\mathsf{cond}(p, f_1, f_2)v = \begin{cases} f_1(v) & \text{if } p(v) = \text{true} \\ f_2(v) & \text{if } p(v) = \text{false} \\ \perp & \text{otherwise} \end{cases}$$

[3]"No tensor products."

is interpreted as

$$\mathsf{cond}^\#(p^\#, f_1^\#, f_2^\#) = (f_1^\# \circ p^\#_{true}) \sqcup (f_2^\# \circ p^\#_{false})$$

where $p^\#_{true}$ and $p^\#_{false}$ are filters defined from $p^\#$ so that typically

$$p^\#_b w = \begin{cases} w & \text{if } p^\# w \text{ is an abstraction of } b \\ \perp & \text{otherwise} \end{cases}$$

Under these assumptions we can validate our claims as illustrated below.

Example A1: A tail recursive function has the general form

$$\begin{aligned} \mathtt{f\ x} &= \mathtt{if\ p\ x\ then\ f_1\ x\ else\ f(f_2\ x)} \\ &= \mathsf{cond}(\mathtt{p}, \mathtt{f_1}, \mathtt{f} \circ \mathtt{f_2})\ \mathtt{x} \end{aligned}$$

The functional H obtained from the analysis will then be

$$\begin{aligned} Hh &= \mathsf{cond}^\#(p^\#, f_1^\#, h \circ f_2^\#) \\ &= (f_1^\# \circ p^\#_{true}) \sqcup h \circ (f_2^\# \circ p^\#_{false}) \end{aligned}$$

which is in iterative form so that the results of Section 4 apply. □

Example A2: For us a linear function has the general form

$$\begin{aligned} \mathtt{f\ x} &= \mathtt{if\ p\ x\ then\ f_1\ x\ else\ f_2(f(f_3\ x))} \\ &= \mathsf{cond}(\mathtt{p}, \mathtt{f_1}, \mathtt{f_2} \circ \mathtt{f} \circ \mathtt{f_3})\ \mathtt{x} \end{aligned}$$

The functional H obtained from the analysis is then

$$\begin{aligned} Hh &= \mathsf{cond}^\#(p^\#, f_1^\#, f_2^\# \circ h \circ f_3^\#) \\ &= (f_1^\# \circ p^\#_{true}) \sqcup (f_2^\# \circ h \circ (f_3^\# \circ p^\#_{false})) \end{aligned}$$

which is in linear form so that the results of Section 5 apply. □

Example A3: A primitive recursive function has the general form

$$\begin{aligned}\texttt{f x} &= \texttt{if p x then } \texttt{f}_1 \texttt{ x else } \texttt{f}_2(\texttt{f}(\texttt{f}_3 \texttt{ x}), \texttt{ x})\\ &= \texttt{cond}(\texttt{p}, \texttt{f}_1, \texttt{f}_2 \circ \texttt{tuple}(\texttt{f} \circ \texttt{f}_3, \texttt{ id}))\ \texttt{x}\end{aligned}$$

The functional H obtained from the analysis is then

$$\begin{aligned}Hh &= \texttt{cond}^{\#}(p^{\#}, f_1^{\#}, f_2^{\#} \circ \texttt{tuple}(h \circ f_3^{\#},\ id^{\#}))\\ &= (f_1^{\#} \circ p_{true}^{\#}) \sqcup (f_2^{\#} \circ\\ &\quad \texttt{tuple}(h \circ (f_3^{\#} \circ p_{false}^{\#}),\ id^{\#} \circ p_{false}^{\#}))\end{aligned}$$

which is in primitive recursive form so that the results of Section 6 do apply. □

The more debatable restriction on the form of the analyses is probably that for the conditional. One may note that it holds in classical flow analysis [1] as well as in many instances of abstract interpretation [8]. For simple strictness analysis [6] over the two point domain it is more common to have

$$\texttt{cond}'(h, h_1, h_2)w = h(w) \sqcap (h_1(w) \sqcup h_2(w))$$

Since the lattices of concern are distributive we have

$$\texttt{cond}'(h, h_1, h_2)w = (h(w) \sqcap h_1(w)) \sqcup (h(w) \sqcap h_2(w))$$

If h_1 and h_2 are strict we can bring $\texttt{cond}'$ into the desired form by setting

$$\texttt{cond}'(h, h_1, h_2)w = h_1(h_{true}(w)) \sqcup h_2(h_{false}(w))$$

where

$$h_{true}(w) = h_{false}(w) = \begin{cases} w & \text{if } h(w) = 1\\ \perp & \text{otherwise}\end{cases}$$

When h_1 and h_2 are not strict one may change the analysis to include a new "artificial" $\perp$-element in which all functions are strict. This is similar to the approach of projection based strictness analysis [12].

Finally we should like to stress that the requirements on the analyses and functions are *sufficient* but *not necessary* in order to apply the results.

Example A4: Consider strictness analysis of the Fibonacci function

```
fib n = if n ≤ 1 then 1
           else fib(n-1) + fib(n-2)
```

Even though this function is not in "linear form" the corresponding functional may be simplified to one that is in iterative form. □

Example A5: Also the restrictions on the analyses can sometimes be lifted. In a *backward* analysis it is often natural to take

$$\begin{aligned}h_1 \circ^{\#} h_2 &= h_2 \circ h_1\\ \texttt{cond}^{\#}(h, h_1, h_2)w &= h_1(w) \sqcup h_2(w) \sqcup h(1)\\ \texttt{tuple}^{\#}(h_1, h_2)(w_1, w_2) &= h_1(w_1) \sqcup h_2(w_2)\end{aligned}$$

If we apply this analysis to a primitive recursive function we will obtain a functional in linear form. An example of an analysis satisfying these conditions is the liveness analysis of [11]. □