Fixed-Point Computations over Functions on Integers with Operations Min, Max and Plus

Yoshinori Tanabe and Masami Hagiya
Dept. of Computer Science, Graduate School of Inform. Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
tanabe@ci.i.u-tokyo.ac.jp and hagiya@is.s.u-tokyo.ac.jp

Abstract

Various kinds of graph problems, including shortest path computation, proof-number search, dataflow analysis, etc., can be solved by fixed-point computations over functions defined on natural numbers or integers. In this paper, we prove that fixed-point computations are possible for the algebra $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty, -\infty\}$, which has the operators min, max and plus. Since $\mathbb{Z}_\infty$ is not well-ordered, we formulate a kind of acceleration technique to guarantee termination of fixed-point computations.

1 Introduction

Fixed-point computations on various algebraic structures are required in many fields of computer science. The simplest example of such an algebraic structure is the boolean algebra, $\mathbf{2} = \{0, 1\}$, on which ordinary boolean operations are defined. Model-checking problems on Kripke structures [3] are reduced to fixed-point computations over functions from the set of states to $\mathbf{2}$, where each parameter of a function corresponds to the value of a propositional variable at a state of the target Kripke structure, which relates the parameters belonging to one state with those belonging to adjacent states. The μ and ν operators in the modal μ-calculus [6] correspond to the least and greatest fixed-points of a system of such boolean functions.

Model-checking problems can be easily generalized by adopting algebraic structures other than $\mathbf{2}$. Propositional variables are generalized to variables having value of the adopted algebraic structure, and boolean operations are replaced with operations on the algebraic structure. The modal μ-calculus can still be used as a language for expressing fixed-points, if the operators of the calculus are interpreted as operations on the algebraic structure. In fact, in our previous work [4], we adopted $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$, the set of natural numbers augmented with infinity, as an algebraic structure, and interpreted the operators $\vee$ and $\wedge$ as min and plus, respectively, over $\mathbb{N}_\infty$. The modal operators $\Diamond$ and $\Box$ were also interpreted accordingly. This kind of algebraic structure having min and plus is well known as min-plus algebra [8, 2]. As the values 0 and ∞ in $\mathbb{N}_\infty$ naturally correspond to 1 (true) and 0 (false) in $\mathbf{2}$, respectively, we interpreted the μ and ν operators as the greatest and least fixed-points. We then formulated algorithms for computing fixed-points over generalized Kripke structures.

Interestingly, various kinds of graph problems can be expressed in the above framework, including shortest path computation, proof-number search [1], dataflow analysis, etc. For example, we can extend the approach taken by Lacey et al., who used CTL, a sublogic of the modal μ-calculus, to express complex conditions on a control flow graph for the purpose of program transformation [7]. A control flow graph is regarded as a Kripke structure. We introduce propositional symbols **accessx** and **updatex**, which hold at a node of the control flow graph if the program variable **x** is accessed and updated on the node, respectively. Then we can construct a formula that expresses, for example, the minimum number of accesses to the variable **x** after each update of **x**.

For applications as the above one, it is natural to introduce the max operator in addition to min and plus. In the above example, it becomes possible to express the maximum number of accesses to a program variable. As this paper shows, however, fixed-point computations become more involved if max is introduced.

Even in our previous work, fixed-point computations were not trivial. Since $\mathbb{N}_\infty$ is well-ordered, the greatest fixed-point can be calculated in a natural way — starting from ∞ and repeat calculating the next value to obtain a decreasing chain of values. However, this simple strategy cannot be applied to compute the least fixed-point. In our previous work, we developed a kind of acceleration technique to guarantee termination.

In this paper, by carefully extending the acceleration technique, we show that fixed-point computations are also possible for the algebra $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty, -\infty\}$, which has the operators min, max and plus. We formulate an algorithm that computes fixed-points of functions defined on $\mathbb{Z}_\infty$. Our previous work is subsumed by embedding $\mathbb{N}_\infty$ into $\mathbb{Z}_\infty$. The efficiency of fixed-point computations is also improved. In some cases, the new algorithm is more efficient than the old one.

2 Target Functions

We define the set $\mathscr{F}$ of functions that our algorithm targets. Roughly speaking, an element of $\mathscr{F}$ is a function on finite power of $\mathbb{Z}_\infty$, composed of operations min, max, plus, minus, and fixed-points. Since we allow the fixed-point operations, the functions need to be monotone with respect to parameters over which the fixed-point is calculated. Therefore, we need to keep track of *positive* and *negative* parameters. Another small issue is value of operations when operands are ∞ or $-\infty$. While most of them can be defined naturally, some decision on the value of $\infty + (-\infty)$ is needed. We introduce two different operators $+_\uparrow$ and $+_\downarrow$, and define $\infty +_\uparrow (-\infty) = (-\infty) +_\uparrow \infty = \infty$, and $\infty +_\downarrow (-\infty) = (-\infty) +_\downarrow \infty = -\infty$. If $\{x, y\} \neq \{\infty, -\infty\}$, then $x +_\uparrow y = x +_\downarrow y = x + y$.

We introduce some notations. Let $n, m, k \in \mathbb{N}$. The set $\{n+1, n+2, \ldots, n+m\}$ is denoted by $I_{n,m}$, and $I_{0,n}$ is denoted by I_n. The constant function on ${\mathbb{Z}_\infty}^n$ that takes value $c \in \mathbb{Z}_\infty$ is denoted by γ_c^n. The projection function on ${\mathbb{Z}_\infty}^n$ to $T = \{t_1, \ldots, t_k\} \subseteq I_n$ is denoted by π_T^n, i.e., $\pi_T^n : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^k$, $\pi_T^n(x_1, \ldots, x_n) = (x_{t_1}, \ldots, x_{t_k})$. We write $\pi_{\{m\}}^n$ as π_m^n. The superscript n of γ and π is often omitted if no confusion occurs. For function f, function $\pi_T f$ is defined by $(\pi_T f)(x) = \pi_T(f(x))$. If $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_k)$, then $(x_1, \ldots, x_n, y_1, \ldots, y_k)$ is denoted by (x, y). We intentionally abuse this notation: if (T, S) is a partition of I_n, $y = \pi_T x$, and $z = \pi_S x$, then we write (y, z) to express x, if T and S are clear from the context. For functions $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ and $g : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^k$, $(f, g) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^{m+k}$ is defined by $(f, g)(x) = (f(x), g(x))$. For $c \in \mathbb{Z}_\infty$, the n-tuple of c, i.e., $x \in {\mathbb{Z}_\infty}^n$ such that $\pi_i x = c$ for all $i \in I_n$, is also denoted by c.

For $n, k \in \mathbb{N}$ and $c \in \mathbb{Z}_\infty$, let $z_{k,c}^n \in {\mathbb{Z}_\infty}^n$ be defined by $\pi_k z_{k,c}^n = c$ and $\pi_i z_{k,c}^n = 0$ for all $i \in I_n \setminus \{k\}$.

For $x, y \in {\mathbb{Z}_\infty}^n$, we write $x \leq y$ if $\pi_i x \leq \pi_i y$ for all $i \in I_n$. If $x \leq y$ and $x \neq y$, we write $x < y$. If $\pi_i x < \pi_i y$ for all $i \in I_n$, we write $x \ll y$. If (T, S) is a partition of I_n, $x \leq y$, and $\pi_S x = \pi_S y$, we write $x \leq_T y$. A function $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ is *monotone* (or *monotone increasing*) w.r.t. T if $x \leq_T y$ implies $f(x) \leq f(y)$, and *monotone decreasing* w.r.t. T if $x \leq_T y$ implies $f(x) \geq f(y)$.

For $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^n$, the k-th repetition of f is denoted by $f^{(k)}$. I.e., $f^{(k)} : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^n$, $f^{(0)}(x) = x$, and $f^{(k+1)}(x) = f(f^{(k)}(x))$.

We define the set $\mathscr{F}$ as the least set that satisfies the following conditions, together with the set $P(f)$ and $N(f)$ of positive and negative parameter indices of $f \in \mathscr{F}$, respectively.

- $\gamma_c \in \mathscr{F}$ and $P(\gamma_c) = N(\gamma_c) = \varnothing$.
- $\pi_T \in \mathscr{F}$, $P(\pi_T) = T$, and $N(\pi_T) = \varnothing$.
- If $f \in \mathscr{F}$, then $-f \in \mathscr{F}$, $P(-f) = N(f)$, and $N(-f) = P(f)$.
- If $f, g \in \mathscr{F}$, $P = P(f) \cup P(g)$, $N = N(f) \cup N(g)$, and $P \cap N = \varnothing$, then $h = (f, g)$, $f +_\uparrow g$, $f +_\downarrow g$, $\min(f, g)$, $\max(f, g) \in \mathscr{F}$, $P(h) = P$, and $N(h) = N$.

- If $f : {\mathbb{Z}_\infty}^{n+m} \to {\mathbb{Z}_\infty}^m$, $f \in \mathscr{F}$, and $I_{n,m} \cap N(f) = \varnothing$, then $h = \mathrm{LFP}(f), \mathrm{GFP}(f) \in \mathscr{F}$, $P(h) = P(f) \setminus I_{n,m}$, and $N(h) = N(f)$, where $\mathrm{LFP}(f) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ is defined so that for any $x \in {\mathbb{Z}_\infty}^n$, $\mathrm{LFP}(f)(x)$ is the least $y \in {\mathbb{Z}_\infty}^m$ such that $f(x,y) = y$. $\mathrm{GFP}(f)$ is defined similarly as the largest such y.

The existence of LFP and GFP in the definition can be proved by simultaneous induction with the fact that for all $f \in \mathscr{F}$, f is monotone increasing w.r.t. $P(f)$ and monotone decreasing w.r.t. $N(f)$. For LFP, starting with $z_0 = -\infty \in {\mathbb{Z}_\infty}^k$, an increasing sequence $(z_\alpha)_\alpha$ of ${\mathbb{Z}_\infty}^k$, indexed by ordinal numbers, is defined by $z_{\alpha+1} = f(x, z_\alpha)$ for $\alpha = 0, 1, \ldots$. It is clear that we have $z_{\alpha+1} = z_\alpha$ for some $\alpha \leq \omega^k$ and $\mathrm{LFP}(f)(x) = z_\alpha$. However, to complete the computation with finite repetitions, we need some acceleration technique, which is the main topic of this paper.

Because LFP and GFP are dual, we mainly concentrate on LFP. Assume $f \in \mathscr{F}$, $f : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^m$, $c \in {\mathbb{Z}_\infty}^m$, and $f(c) \geq c$. There exists the least y such that $y \geq c$ and $f(x,y) = y$. This y is denoted by $\mathrm{LFP}_c(f)$. For $\mathrm{LFP}(f)$, it is sufficient to compute $\mathrm{LFP}_c(f)$, because $\mathrm{LFP}(f) = \mathrm{LFP}_{-\infty}(f)$. We observe that if the operator max does not appear in the definition sequence of f, then $\mathrm{LFP}_c(f)$ can be computed relatively easily. To formalize the observation, we introduce a concept called "steplessness." Let $T \subseteq I_n$. A function $f : {\mathbb{Z}_\infty}^n \to \mathbb{Z}_\infty$ is *upward stepless* w.r.t. T if f is monotone w.r.t. T, and for all $k \in T$ and $x \in {\mathbb{Z}_\infty}^n$, $f(x) = f(x + z^n_{k,1})$ implies $f(x) = f(x + z^n_{k,c})$ for all $c \in \mathbb{N}_\infty$. Word "upward" and set T are omitted if no confusion occurs. A function $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ is *stepless* if $\pi_j f$ is stepless for all $j \in I_m$. Functions γ_c, π_T, $\min(x,y)$, and $x +_\downarrow y$ are stepless, but $\max(x,y)$ and $x +_\uparrow y$ are not. Stepless functions are closed under compositions: more precisely, if $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^k$ is stepless w.r.t. T, $g : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^n$, $\pi_i g$ is stepless w.r.t. S for all $i \in T$, and $\pi_i g = \pi_i$ for all $i \in I_n \setminus T$, then $f \circ g$ is stepless w.r.t. S.

The least fixed-point of a stepless function is computed using the following lemma, which can be proved in a similar manner as in the corresponding lemma in [5],

Lemma 1. *Assume $f : {\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^m$ is stepless, $c \in {\mathbb{Z}_\infty}^m$, and $f(c) \geq c$. Let $\bar{c} = \mathrm{LFP}_c(f)$, $T = \{i \in I_m \mid \pi_i c < \pi_i \bar{c}\}$, and $S = I_m \setminus T$. Thus, with respect to the partition (T,S), we have $c = (d,e)$, $\bar{c} = (\bar{d},e)$, and $d \ll \bar{d}$. Then, the following hold.*

(1) $T = \{i \in I_m \mid \pi_i c < \pi_i f^{(m)}(c)\}$.

(2) There is $i \in I_m$ such that $\pi_i \mathrm{LFP}_c(f) = \pi_i f(\infty, e)$

(3) $\mathrm{LFP}_c(f) = f^{(|T|)}(\infty, e)$.

For GFP, we define $\mathrm{GFP}_c(f)(x)$ to be the greatest y such that $y \leq c$ and $f(x,y) = y$, if $f(x,c) \leq c$. Function f is *downward stepless* w.r.t. T if for all $k \in T$, $f(x) = f(x - z^n_{k,1})$ implies $f(x) = f(x - z^n_{k,c})$ for all $c \in \mathbb{N}_\infty$. Then, the counterpart of Lemma 1 holds: if f is downward stepless, $f(c) \leq c$, $c = (d,e)$ and $\bar{c} = (\bar{d},e) = \mathrm{GFP}_d(f)$ w.r.t. a partition (T,S), and $\bar{d} \ll d$, then we have (1) $T = \{i \in I_m \mid \pi_i d > \pi_i f^{(m)}(d)\}$, (2) there is $i \in I_m$ such that $\pi_i \bar{d} = \pi_i f(\infty, e)$, and (3) $\bar{d} = f^{|T|}(\infty, e)$.

Although not all functions in $\mathscr{F}$ are stepless, we can approximate them with stepless functions. Assume $f \in \mathscr{F}$, $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$, and $c \in {\mathbb{Z}_\infty}^n$. We define the *under approximation* $\mathrm{ua}(f,c) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ and the *over approximation* $\mathrm{oa}(f,c) : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^m$ as shown in Figure 1. The intuition is as follows: we wish to define $\mathrm{ua}(f,c)$ to be an upward stepless function. Because pair, minimum, $+_\downarrow$ preserves steplessness (because it is closed under compositions), these operations can be handled naturally. LFP and GFP are also all right, because they are expressed as compositions if the operand is stepless, by Lemma 1 (3). Operation $+_\uparrow$ does not preserve steplessness, but it is almost the same as operation $+_\downarrow$, and the exceptional case can be covered by a constant function. Finally, for max, we simply choose one of the operands, by referring their values at c.

The following lemma can be proved without difficulty.

$$\mathrm{ua}(\gamma_e,c)=\gamma_e$$
$$\mathrm{ua}(\pi_T,c)=\pi_T$$
$$\mathrm{ua}(-f,c)=-\mathrm{oa}(f,c)$$
$$\mathrm{ua}((f,g),c)=(\mathrm{ua}(f,c),\mathrm{ua}(g,c))$$
$$\mathrm{ua}(\min(f,g),c)=\min(\mathrm{ua}(f,c),\mathrm{ua}(g,c))$$
$$\mathrm{ua}(\max(f,g),c)=\begin{cases}\mathrm{ua}(f,c) & \text{if } f(c)\geq g(c)\\ \mathrm{ua}(g,c) & \text{otherwise}\end{cases}$$
$$\mathrm{ua}(f+_{\downarrow}g,c)=\mathrm{ua}(f,c)+_{\downarrow}\mathrm{ua}(g,c)$$
$$\mathrm{ua}(f+_{\uparrow}g,c)=\begin{cases}\gamma_{-\infty} & \text{if } \{f(c),g(c)\}=\{\infty,-\infty\}\\ \mathrm{ua}(f,c)+_{\downarrow}\mathrm{ua}(g,c) & \text{otherwise}\end{cases}$$
$$\mathrm{ua}(\mathrm{LFP}(f),c)=\mathrm{LFP}_c(\mathrm{ua}(f,(c,\mathrm{LFP}(f)(c))))$$
$$\mathrm{ua}(\mathrm{GFP}(f),c)=\mathrm{GFP}_c(\mathrm{ua}(f,(c,\mathrm{GFP}(f)(c))))$$

$$\mathrm{oa}(\gamma_e,c)=\gamma_e$$
$$\mathrm{oa}(\pi_T,c)=\pi_T$$
$$\mathrm{oa}(-f,c)=-\mathrm{ua}(f,c)$$
$$\mathrm{oa}((f,g),c)=(\mathrm{oa}(f,c),\mathrm{oa}(g,c))$$
$$\mathrm{oa}(\min(f,g),c)=\begin{cases}\mathrm{oa}(f,c) & \text{if } f(c)\leq g(c)\\ \mathrm{oa}(g,c) & \text{otherwise}\end{cases}$$
$$\mathrm{oa}(\max(f,g),c)=\max(\mathrm{oa}(f,c),\mathrm{oa}(g,c))$$
$$\mathrm{oa}(f+_{\downarrow}g,c)=\begin{cases}\gamma_{\infty} & \text{if } \{f(c),g(c)\}=\{\infty,-\infty\}\\ \mathrm{oa}(f,c)+_{\uparrow}\mathrm{oa}(g,c) & \text{otherwise}\end{cases}$$
$$\mathrm{oa}(f+_{\uparrow}g,c)=\mathrm{oa}(f,c)+_{\uparrow}\mathrm{ua}(g,c)$$
$$\mathrm{oa}(\mathrm{LFP}(f),c)=\mathrm{LFP}_c(\mathrm{oa}(f,(c,\mathrm{LFP}(f)(c))))$$
$$\mathrm{oa}(\mathrm{GFP}(f),c)=\mathrm{GFP}_c(\mathrm{oa}(f,(c,\mathrm{GFP}(f)(c))))$$

Figure 1: Under/Over Approximation

Lemma 2. *Assume $f\in\mathscr{F}$, $f:{\mathbb{Z}_\infty}^n\to{\mathbb{Z}_\infty}^m$, and $c\in{\mathbb{Z}_\infty}^n$.*

(1) $\mathrm{ua}(f,c)$ *is upward stepless w.r.t. $P(f)$, and* $\mathrm{oa}(f,c)$ *is downward stepless w.r.t. $P(f)$.*

(2) $\mathrm{ua}(f,c)(c)=\mathrm{oa}(f,c)(c)=f(c)$.

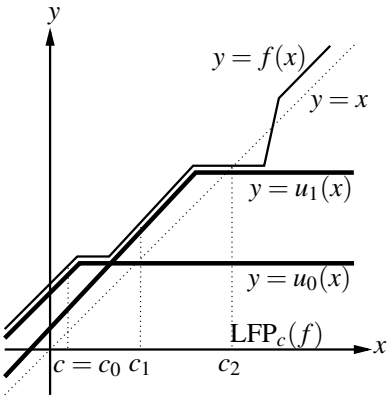
(3) $\mathrm{ua}(f,c)(x)\leq f(x)\leq\mathrm{oa}(f,c)(x)$ *for all $x\in{\mathbb{Z}_\infty}^n$.*

(4) If $f(c)\geq c$, then $\mathrm{LFP}_c(\mathrm{ua}(f,c))\leq\mathrm{LFP}_c(f)$. *If $f(c)\leq c$, then* $\mathrm{GFP}_c(\mathrm{oa}(f,c))\geq\mathrm{GFP}_c(f)$.

(5) $\{\mathrm{ua}(f,d)\mid d\in{\mathbb{Z}_\infty}^n\}$ *and* $\{\mathrm{oa}(f,d)\mid d\in{\mathbb{Z}_\infty}^n\}$ *are finite sets.*

3 Procedure

Based on the preparation in the previous section, we now describe the procedure to compute $\mathrm{LFP}_c(f)$ for $f\in\mathscr{F}$ and $c\in{\mathbb{Z}_\infty}^m$ such that $f:{\mathbb{Z}_\infty}^m\to{\mathbb{Z}_\infty}^m$ and $f(c)\geq c$. As a starting point, we consider the following naive procedure: starting with $c_0=c$, we compute $c_{n+1}=\mathrm{LFP}_{c_n}(\mathrm{ua}(f,c_n))$ until $c_n=c_{n+1}$. Because $\mathrm{ua}(f,c_n)$ is stepless, we can compute c_{n+1} using Lemma 1. Then, $\mathrm{LFP}_c(f)=c_n$.



The right figure illustrates the intuition behind the procedure. Note that $\bar{c}=\mathrm{LFP}_c(f)$ is the left-most intersection (on the right side of c) of the graph of $y=f(x)$ with that of $y=x$. Because of Lemma 2 (4), $c_1=\mathrm{LFP}_{c_0}(u_0)$ is smaller than or equal to $\bar{c}$, and c_1 is greater than or equal to c_0 by its definition. Thus, we have $c_0\leq c_1\leq\cdots$. The number of repetitions seems to be finite, because of Lemma 2 (5) and the fact that $\mathrm{ua}(f,c_n)$ becomes "constant" on the right side of c_{n+1}.

Unfortunately, the intuition is not correct. The procedure may not terminate if f has two or more parameters, as will be shown in Example 5. To resolve the problem, the under approximation should be taken component-wise, and if f does not move a component of c_n, we keep the previous approximation for that component.

The modified procedure is shown in Figure 2. For $f\in\mathscr{F}$, $\mathrm{LFP}_c(f)$ is computed in the left column. Because u_n is stepless, $\mathrm{LFP}_{c_n}(u_n)$ appearing in the left column is computed in the right column.

Procedure for $f \in \mathscr{F}$.
Compute $c_n \in {\mathbb{Z}_\infty}^m$ and define stepless function u_n until $c_{n+1} = c_n$:
$c_0 = c, \quad u_0 = \mathrm{ua}(f, c_0)$.
$c_{n+1} = \mathrm{LFP}_{c_n}(u_n)$.
$$\pi_i u_{n+1} = \begin{cases} \pi_i u_n & \text{if } \pi_i f(c_{n+1}) = \pi_i c_{n+1} \\ \mathrm{ua}(\pi_i f, c_{n+1}) & \text{otherwise} \end{cases}$$
Then, $\mathrm{LFP}_c(f) = c_n$.

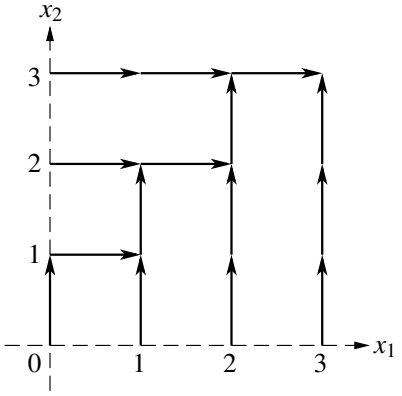
Procedure for stepless f.
Compute $d_n \in {\mathbb{Z}_\infty}^m$ until $T_{n+1} = T_n$, where $T_n = \{i \in I_n \mid \pi_i d_n > \pi_i c\}$:
$d_0 = c, \quad d_{n+1} = f(d_n)$.
Let $T = T_n$, $S = I_m \setminus T$, and $s = \pi_S c$. Compute $e_k \in {\mathbb{Z}_\infty}^m$ until $e_{k+1} = e_k$:
$e_0 = (\infty, s), \quad e_{k+1} = f(e_k)$
Then, $\mathrm{LFP}_c(f) = e_k$.

Figure 2: Procedure to Compute $\mathrm{LFP}_c(f)$

Example 3. Let $f : {\mathbb{Z}_\infty}^2 \to {\mathbb{Z}_\infty}^2$ be defined by $f(x_1, x_2) = (x_1, \min(x_1 +_\downarrow x_2, 10))$. We compute $\mathrm{LFP}_{(1,0)}(f)$. Because f is stepless, the right column of Figure 2 is used. The computation of the first half is as follows: $d_0 = (1,0)$, $T_0 = \varnothing$, $d_1 = (1,1)$, $T_1 = \{2\}$, $d_2 = (1,2)$. $T_2 = \{2\}$. Here, we have $T_1 = T_2 = \{2\}$. With this result, we start the second half: $e_0 = (1,\infty)$, $e_1 = (1,10)$, $e_2 = (1,10)$. Thus, we get the result: $\mathrm{LFP}_{(1,0)}(f) = (1,10)$.

Example 4. Assume that $n \in \mathbb{N}$ and $f_n : \mathbb{Z}_\infty \to \mathbb{Z}_\infty$ is defined by $f_n(x) = \max(n+1+\min(x-n,0),0)$. We compute $\mathrm{LFP}(f_n)$. First, $c_0 = -\infty$ and $u_0 = \gamma_0$, because $n+1+\min(-\infty-n,0) < 0$. Therefore, $c_1 = \mathrm{LFP}_{-\infty}(\gamma_0) = 0$. Because $f_n(c_1) = 1 \neq c_1$, we take $u_1(x) = \mathrm{ua}(f_n, 0)(x) = n+1+\min(x-n,0)$. Using the right column, we get $c_2 = \mathrm{LFP}_0(u_1) = n+1$. Repeating this step, we find $c_3 = n+1$, and conclude $\mathrm{LFP}(f_n) = n+1$.

Example 5. Let function $f : {\mathbb{Z}_\infty}^2 \to {\mathbb{Z}_\infty}^2$ be defined by $f(x_1,x_2) = (\min(x_1+1, \max(x_1,x_2)), \min(\max(x_2,x_1+1), x_2+1))$. A small calculation could show that $f(x_1,x_2) = (x_1, x_2+1)$ if $x_1 \geq x_2$, and $f(x_1,x_2) = (x_1+1, x_2)$ if $x_1 < x_2$, although it is not a part of the procedure. In the right figure, f is depicted by arrows connecting (x_1,x_2) and $f(x_1,x_2)$. It is clear that $\mathrm{LFP}_{(0,0)}(f) = (\infty,\infty)$.



If we apply the naive procedure shown in the beginning of this section, the computation does not terminate. We start with $c_0 = (0,0)$ and $u_0(x_1,x_2) = (x_1, \min(x_1+1, x_2+1))$. Then, $c_1 = \mathrm{LFP}_{c_0}(u_0) = (0,1)$, and $u_1(x_1,x_2) = (\min(x_1+1,x_2), x_2)$. In the next steps, we have $c_2 = (1,1)$, $u_2 = u_0$, $c_3 = (1,2)$, and $u_3 = u_1$. Thus, the computation continues for ever, calculating $c_{2n} = (n,n)$, $c_{2n+1} = (n,n+1)$, $u_{2n} = u_0$, and $u_{2n+1} = u_1$.

On the other hand, the modified procedure does terminate. The computation goes in the same way up to c_1. When we decide u_1, two values $c_1 = (0,1)$ and $f(c_1) = (1,1)$ are compared. Because their x_2-components are identical, we reuse the x_2-component of u_0 for that of u_1. The x_1-component is calculated as usual. Thus, we have $u_1(x_1,x_2) = (\min(x_1+1,x_2), \min(x_1+1, x_2+1))$, and because $\mathrm{LFP}_{c_1}(u_1) = (\infty,\infty)$, we reach the result in finite repetitions.

The partial correctness of the procedure is almost clear from the previous lemmas. When f is stepless, by Lemma 1 (1), $(T, I_m \setminus T)$ gives the required partition, where $T = T_n = T_{n+1}$. Then, Lemma 1 (3) guarantees that $\mathrm{LFP}_c(f) = e_k$. For $f \in \mathscr{F}$, Lemma 1 shows that the right column computes the fixed-point for stepless functions. By Lemma 2 (4), $(c_n)_n$ is an increasing sequence that does not exceed $\mathrm{LFP}_c(f)$. Therefore, if we reach n such that $c_{n+1} = c_n$, then $\mathrm{LFP}_c(f) = c_n$.

To prove the termination, i.e., there exists n such that $c_{n+1} = c_n$, we use the following technical lemma.

Lemma 6. *Assume (T,S) is a partition of I_m, $u=(v,w):{\mathbb{Z}_\infty}^m \to {\mathbb{Z}_\infty}^m$ is a stepless function, $(a_2,b_2) = \mathrm{LFP}_{(a_1,b_1)}u$, $v(a_2,b_3)=a_2$, $a_1 \ll a_2$, and $b_2 \ll b_3$. Then, there exists $i \in T$ such that $\pi_i a_2 = \pi_i v(\infty,\infty)$.*

Proof. Let $t=|T|$, $\bar{v}:{\mathbb{Z}_\infty}^t \to {\mathbb{Z}_\infty}^t$ be defined by $\bar{v}(a)=v(a,\infty)$, and $\bar{a}_2 = \mathrm{LFP}_{a_1}\bar{v}$. We have $a_2 \le \bar{a}_2$: this is because while a_2 is the T-part of the supremum of the sequence $(x_\alpha)_\alpha$ defined by $x_0=(a_1,b_1)$ and $x_{\alpha+1}=v(x_\alpha)$, $\bar{a}_2$ is the supremum of the sequence $(y_\alpha)_\alpha$ defined by $y_0=a_1$ and $y_{\alpha+1}=v(y_\alpha,\infty)$. We can show that $y_\alpha \ge \pi_T x_\alpha$ by induction on α. On the other hand, because $v(a_2,b_2)=v(a_2,b_3)$, $b_2 \ll b_3$, and v is stepless, we have $v(a_2,\infty)=a_2$, i.e., a_2 is a fixed point of $\bar{v}$. Therefore, $a_2=\bar{a}_2$. Because $\bar{v}$ is stepless and $a_1 \ll a_2$, by Lemma 1 (2), there exists $i \in T$ such that $\pi_i a_2 = \pi_i \bar{v}(\infty) = \pi_i v(\infty,\infty)$. □

We sketch a termination proof. Assume on the contrary that the sequence does not converge: $c_0 < c_1 < \cdots < c_n < c_{n+1} < \cdots$. Let $U = \{i \in I_m \mid \pi_i c_n < \pi_i c_{n+1} \text{ for infinitely many } n\}$ and $V = I_m \setminus U$. There exists $N \in \mathbb{N}$ such that for all $n \ge N$, $\pi_V c_n = \pi_V c_N$. Let $s = \pi_V c_N$. We write $c_n = (e_n, s)$.

We claim that for any $n' \ge N$, there exists $n \ge n'$ and $i \in U$ such that $e_{n'} \ll e_n$ and $\pi_i u_n(\infty,s) = \pi_i e_n$. Let k be the least k such that $e_{n'} \ll e_k$, and m be the largest m such that $e_m \ll e_k$. Let $T = \{i \in U \mid \pi_i e_{m+1} = \pi_i e_k\}$ and $S = U \setminus T$. By applying Lemma 6 with $u := u_m$ (with V-part fixed to s), $(a_1,b_1) := e_m$, $(a_2,b_2) := e_{m+1}$, and $b_3 := \pi_S e_k$, we confirm the claim by taking $n = m+1$.

By Lemma 2 (5), there exists $i \in T$ and $n,k \in \mathbb{N}$ such that $e_n \ll e_k$, $u_n = u_k$, $\pi_i u_n(\infty,s) = \pi_i e_n$, and $\pi_i u_k(\infty,s) = \pi_i e_k$, which is impossible.

4 Function Examples

In this section, we show that several functions defined on graphs are regarded as elements of $\mathscr{F}$, thus they can be computed by our algorithm. We do not claim that our procedure is more suitable to calculate values of these particular functions than known algorithms. Instead, these examples illustrate various quantitative properties are expressed as the fixed-point of a function in $\mathscr{F}$.

4.1 Shortest Path

Assume that $G = \{s_1, s_2, \ldots, s_n\}$ is a finite set of nodes and the lengths $d(i,j) \in \mathbb{N}_\infty$ of the connection between two adjacent nodes s_i and s_j are given. If s_i and s_j are not directly connected, $d(i,j)=\infty$. We fix $s = s_{i_0} \in G$, and define f by $\pi_i f(x) = \min(\{d(i,i_0)\} \cup \{d(i,j) + \pi_j x \mid j \ne i_0\})$. Then, the length of the shortest path between s_i to s is $\pi_i \mathrm{GFP}_x(f)$ (or $\pi_i \mathrm{LFP}_x(f)$: they coincide in this case). The intuitive meaning of this definition is that the shortest path from s_i is either the direct connection to s_{i_0} or a path to some adjacent state s_j connected with the shortest path between s_j and s_{i_0}, whichever is the shortest.

If (G,E) is a graph and $d(i,j)$ is defined by $d(i,j)=1$ if $(s_i,s_j) \in E$, $d(i,j)=\infty$ otherwise, then the same function computes the shortest hop count from s_i to s_{i_0}.

4.2 Proof-Number Search

Let us consider a finite tree with labels on nodes. The label is either 'true', 'false', or 'unknown' on a leaf node, and either 'MAX' or 'MIN' on an internal node. The proof number $\mathrm{proof}(n)$ of node n is defined as follows [1]: the value of $\mathrm{proof}(n)$ is 0 if the label is 'true', ∞ if 'false', and 1 if 'unknown'. For an internal node, $\mathrm{proof}(n) = \min\{\mathrm{proof}(n') \mid n' \in \mathrm{child}(n)\}$ if the label is 'MAX', and $\mathrm{proof}(n) = \sum\{\mathrm{proof}(n') \mid n' \in \mathrm{child}(n)\}$ if the label is 'MIN'.

The proof number can be computed as a value of function in $\mathscr{F}$: let $\{n_i \mid i=1,\ldots,N\}$ be an enumeration of the nodes of a tree. For each i, we define $f_i : {\mathbb{Z}_\infty}^N \to \mathbb{Z}_\infty$ to reflect the definition of the proof number. For example, if s_i is a leaf node with label 'false,' $f_i(x)=\infty$ for any x, and if s_i is an internal

node with label 'MIN,' $f_i(x) = \sum(x_j \mid s_j \in \mathrm{child}(s_i))$. Let $f : {\mathbb{Z}_\infty}^N \to {\mathbb{Z}_\infty}^N$ be such that $\pi_i f = f_i$ for all i. Then, it is obvious that $\mathrm{proof}(n_i) = \pi_i \mathrm{GFP}(f)$.

4.3 Data Flow Analysis

Lacey et al. used CTL to specify conditions on a control flow graph by regarding the graph as a Kripke structure for the purpose of program transformation [7]. We extended the approach by introducing a non-standard semantics of the modal μ-calculus [4]. For example, let us introduce a propositional symbol **accessx** that holds at a node in a control flow graph if the program variable **x** is accessed at the node. The minimum number of accesses to the variable **x** on an execution path starting from a node can then be expressed by the following formula under the non-standard semantics:

$$\nu X(((\mathbf{accessx} \wedge 1) \vee (\neg \mathbf{accessx})) \wedge (\mathbf{halt} \vee \Diamond X)).$$

Here, **halt** is an abbreviation for $\Box\bot$, which means that no outgoing transition exists.

Now, for given Kripke structure $\mathscr{K} = (S, R, L)$, we enumerate $S = \{s_1, \ldots, s_n\}$ and denote the set $\{s' \mid (s, s') \in R\}$ by sR. For $i \in I_n$, we define $a_i, h_i \in \mathbb{Z}_\infty$ by $a_i = 1$ if **x** is accessed at s_i, $a_i = 0$ otherwise, and $h_i = 0$ if there is no outgoing node at s_i, $h_i = \infty$ otherwise. Then, the above formula corresponds to function $\mathrm{LFP}(f)$ in $\mathscr{F}$, where $f : {\mathbb{Z}_\infty}^n \to {\mathbb{Z}_\infty}^n$ is defined as follows,

$$\pi_i f(x) = a_i + \min(h_i, \min\{\pi_j x \mid s_j \in s_i R\}).$$

5 Conclusion

5.1 Remark on Efficiency

Compared to the algorithm proposed in [4], the procedure in this paper not only covers wider range of functions, but also is more efficient in some cases. For example, The old algorithm requires $O(n)$ time to compute the least fixed-point of f_n in Example 4 (more precisely, their corresponding functions defined on $\mathbb{N}_\infty$), but the algorithm in Section 3 requires constant time.

Unfortunately, we have not yet obtained the time complexity of the algorithm. Even if we restrict ourselves to fixed-point free functions, the current termination proof shown in Section 3 does not provide the number of required repetitions. To evaluate the complexity of the algorithm remains as future work.

Acknowledgments

The authors would like to thank the reviewers for their careful reading and helpful comments.

This research has been partially supported by Grant-in-Aid for Scientific Research by Ministry of Education, Culture, Science and Technology, Scientific Research(C) 21500006, "Decision procedures of modal logics and their application to software verification."

References

[1] L. V. Allis, M. van der Meulen, and H. J. van den Herik. Proof-number search. *Artif. Intell.*, 66(1):91–124, 1994.

[2] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons, 1992.

[3] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[4] D. Ikarashi, Y. Tanabe, K. Nishizawa, and M. Hagiya. Modal μ-calculus on min-plus algebra $\mathbb{N}_\infty$. In *Proc. of 10th Wksh. on Programming and Programming Languages, PPL 2008 (March 2008)*, 2008. Available at `http://www.nue.riec.tohoku.ac.jp/ppl2008/program.html`.

[5] D. Ikarashi, Y. Tanabe, K. Nishizawa, and M. Hagiya. Modal μ-calculus on min-plus algebra $\mathbb{N}_\infty$. Revised version of [4], submitted, 2009. Available at `http://cent.xii.jp/tanabe.yoshinori/09/05/minplusPC.pdf`.

[6] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27(3):333–354, 1983.

[7] D. Lacey, N. D. Jones, E. Van Wyk, and C. C. Frederiksen. Compiler optimization correctness by temporal logic. *Higher-Order and Symb. Comput.*, 17(3):173–206, 2004.

[8] I. Simon. Limited subsets of a free monoid. In *Proc. of 19th Ann. Symp. on Foundations of Computer Science, FOCS '78 (Ann Arbor, MI, Oct. 1978)*, pp. 143–150. IEEE CS Press, 1978.