A Note on three Programming Paradigms*

N.V. Shilov

A.P. Ershov Institute of Informatics Systems,
Lavren'ev av., 6, Novosibirsk 630090, Russia,
http://www.iis.nsk.su/persons/shilov/shilov.htm
shilov@iis.nsk.su

Abstract. This paper is about a puzzle to be solved in three programming paradigms: logic, functional and imperative. It can be considered as a case study of algorithm inversion, since we start with logic algorithm, that answers the question "Is balancing M times sufficient for detecting a single fake in a set of coins?", and finishes with imperative algorithm, that effectively computes the minimal number of balancing that is sufficient for detection the fake. Functional paradigm is used for developing an intermediate functional algorithm that also computes the minimal number of balancing, but inefficiently, while the efficient imperative algorithm is a "lazy memoization" of the functional one.

1 Introduction

1.1 A Puzzle

Let us start with the following fake-coin puzzle:

> A set of 15 coins consists of 14 valid coins and a false one; all valid coins have equal weights, while the false coin has a different weight, one of the valid coins is marked but all other coins (including the fake) are unmarked. Is it possible to identify the fake coin using a balance at most 3 times?

Of course, there is nothing to program in this puzzle: just to solve it "manually" (sorry, mentally) and write a program that outputs the appropriate answer "Yes" or "No". But the following parameterized version of the puzzle is a straightforward generalization of the above one:

> Write a program that inputs a number $N > 0$ of coins under question and a number $W \geq 0$ of marked valid coins ($N \geq W$), and outputs the least number of balancing M that is always sufficient for detecting the unique fake among N coins under question with aid of W valid ones. (Assume that all valid coins have one and the same weight while the false coin has a different weight.)

* Research is supported by grant of Russian Basic Research Foundation 08-01-00899-a.

We will discuss how to solve this parameterized puzzles in logic, functional and imperative programming in the further sections. But first we would like to provide a short remark about programming paradigms in general and their relations to programming languages.

1.2 What are "Programming Paradigms"?

Robert Floyd was the first who had explicitly used the concept of "Programming Paradigm" in his Turing Award Lecture in 1978 [1]. He referred to Thomas Kuhn's well-known book [2], published just 8 years before. According to T. Kuhn, a paradigm is a method, an approach to a problem formulation (statement) and the ways to solve the problem. R. Floyd had a similar understanding of programming paradigms. In particular, he advocated that a programming language should support one or several of programming paradigms, and wrote: "To the designer of programming languages, I say: unless you can support the paradigms I use when I program, or at least support my extending your language into one that does support my programming methods, I don't need your shiny new languages".

At present the number of essentially different paradigms of programming is already several dozens (see, for example, the list of "programming paradigms" at http://en.wikipedia.org/wiki/Programming paradigm), while the number of programming languages has overcome 2,500 (please refer poster "History of Programming Languages" by O'Reilly). Unfortunately, sometimes relations between programming languages and paradigms (that they support) are vague or not explicit. Due to this reason and due to the number of programming languages that support a paradigm (logic, functional and imperative in particular), we use in this paper logic, functional and imperative pseudocode for representing algorithms (instead of any particular logic, functional or imperative programming language).

1.3 Preliminaries

Let us start with discussion about information/knowledge that is available for a "balancing agent" (anyone who would like to detect the fake) after balancing coins several times.

Before the first balancing, the agent knows initial values of two variables: the first is variable U for the number of coins about which nothing is known, and the second is V for the number of coins which are known to be valid; their initial values are $(N - W)$ and W respectively.

During the first balancing the agent puts some coins at the first and the second pans of the balance; the outcome of the first balancing is either "pans are equal", "the first pan is lighter than the second", or "the first pan is heavier than the second".

The emerging knowledge after the first balancing are four integer values: an updated value of the variable U, the second is an updated value of the variable

V, and values of two new variables L and H that are for the numbers of coins that were balanced and were at lighter and, respectively, at heavier pans.

If to generalize the above observations, then the agent's knowledge about coins after a series of balancing can be represented by the following four non-negative integer values: U of the number of coins, about which nothing is known after the series, L of the number of coins that still have to be verified but were balancing and were at lighter pan, H of the number of coins that still have to be verified but were balancing and were at heavier pan, V of the number of coins, which are known (after this series) to be valid. Observe also that these values meet a natural constraint $U + L + H + V = N$.

Then we have to discuss how to represent balancing in terms of these four values. Let us observe that $L + H > 0$ implies that $U = 0$ (since $L + H > 0$ means that the fake has been balanced already, i.e. all other coins are valid). Due to this reason we can consider two disjoint cases:

1. $L + H = 0$ and $U > 0$,
2. $L + H > 0$ and $U = 0$.

In the first case any balancing consists in a choice of two numbers $u1$ and $u2$ of coins to be put at the first and the second pans, such that $0 < u1 + u2$ (since balancing set should include at least one coin in U), $u1 + u2 \leq U$ (since coins for balancing should be selected in U), $|u1 - u2| \leq V$ (since the oddity in coin numbers should be compensate by coins in V).

In the second case balancing consists in selection of numbers $l1$, $h1$, and $l2$, $h2$ of coins to put at the first and the second pans such that $0 < l1 + l2 + h1 + h2$ (since balancing set should include at least one coin in $L + H$), $l1 + l2 \leq L$ and $h1 + h2 \leq H$ (since coins for balancing should be selected in L and H), $|l1 + h1 - l2 - h2| \leq V$ (since the oddity in coin numbers should be compensate by coins in V).

1.4 Paper outlines

The rest of the paper is organized as follows. The next section 2 addresses logical approach to the parameterized puzzle. Then the section 3 develops a functional solution for the parameterized puzzle. Section 4 converts the functional solution into imperative one by memoization. Finally the last section 5 concludes by some related remarks.

2 Logic of Fake Detection

Problem formulation in logic paradigm is a sound axiomatization of desired data property, problem solution (logic algorithm or program) is a "transformation" of the axiomatization into a knowledge base which can be effectively used by some logic inference machine for proving correct answers for user queries. Below we will illustrate how the logic paradigm can work to solve our parameterized puzzle.

We would like to axiomatize the following data property: it is possible to detect a single fake in a set that consists of $N = U + L + H + V$ coins (where U, L, H, and V have meaning as in the Preliminaries) by balancing them M times at most. Let us denote the appropriate predicate by P and represent the corresponding data property by $P(U, L, H, V, M)$.

Our axiomatization AxP will consist of four axioms. The first axiom describes situation when the fake is detected already: $U+L+H < 1 \leftrightarrow P(U, L, H, V, M)$. It states that a single fake is already detected in the empty set $(U + L + H = 0)$ and in any singleton set of coins $(U + L + H = 1)$. The second axiom is also straightforward: $L+H > 0 \& P(0, L, H, V+U, M) \leftrightarrow P(U, L, H, V, M)$. We have discussed already its meaning: if the fake has been balanced already $(L+H > 0)$, then we can say without additional balancing that all other $(V + U)$ coins are valid.

The remaining two axioms establish connections between $P(U, L, H, V, M)$ and $P(..., ..., ..., ..., M - 1)$ and are based on the following observation:

a single fake can be detected in a set of coins
by balancing them M times

iff

there exists a balancing such that
in case of any possible outcome of this balancing
(a) pans are equal, (b) the first is lighter, (c) the first is heavier,
the fake can be detected by balancing coins $(M - 1)$ times.

In particular, the third axiom formalizes this observation in the case when $U > 0$ and $L = H = 0$:

$$P(U,0,0,V,M) \leftrightarrow \exists u1, u2\ (0 < u1 + u2 \leq U\ \&\ |u1 - u2| \leq V\ \& \\ P(U - u1 - u2, 0, 0, V + u1 + u2, M - 1)\ \& \\ P(0, u1, u2, V + (U - u1 - u2), M - 1)\ \& \\ P(0, u2, u1, V + (U - u1 - u2), M - 1)).$$

In verbal form: balancing M times is sufficient for detecting the fake ($P(U, 0, 0, V, M)$) iff there exists two sets of coins for balancing at the first and the second pans $(\exists u1, u2)$, that they are selected among "unknown" coins $(0 < u1+u2 \leq U)$ with aid of some oddity-compensating valid ones $(|u1 - u2| \leq V)$, in case of any possible outcome ('a', 'b' and 'c' from the above) balancing $(M - 1)$ times is sufficient for detecting the fake ('a' $\leftrightarrow P(U - u1 - u2, 0, 0, V + u1 + u2, M - 1)$, 'b' $\leftrightarrow P(0, u1, u2, V + (U - u1 - u2), M - 1)$, and 'c' $\leftrightarrow P(0, u2, u1, V + (U - u1 - u2), M - 1)$ respectively).

The fourth axiom formalizes this connection in the case when $U = 0$ but $L + H > 0$:

$$P(0,L,H,V,M) \leftrightarrow \exists l1, l2, h1, h2 \\ (0 < l1 + l2 + h1 + h2\ \&\ l1 + l2 \leq L\ \&\ h1 + h2 \leq H\ \&\ |l1 + h1 - l1 - l2| \leq V\ \& \\ P(0, L - l1 - l2, H - h1 - h2, V + l1 + l2 + h1 + h2, M - 1)\ \& \\ P(0, l1, h2, V + (L - l1) + (H - h2), M - 1)\ \& \\ P(0, l2, h1, V + (L - l2) + (H - h1), M - 1)).$$

In words: balancing M times is sufficient for detecting the fake ($P(0, L, H, V, M)$) if there exists two sets of coins for balancing at the first and the second pans ($\exists l1, l2, h1, h2$), that are selected among "light" and "heavy" coins ($0 < l1+l2+h1+h2$, and $l1+l2 \leq L$, $h1+h2 \leq H$) with aid of some oddity-compensating valid ones ($|l1+h1-l1-l2| \leq V$), in case of any possible outcome ('a', 'b' and 'c' from the above) balancing $(M-1)$ times is sufficient for detecting the fake ('a' $\leftrightarrow$ $P(0, L-l1-l2, H-h1-h2, V+l1+l2+h1+h2, M-1)$, 'b' $\leftrightarrow$ $P(0, l1, h2, V+(L-l1)+(H-h2), M-1)$, and 'c' $\leftrightarrow$ $P(0, l2, h1, V+(L-l2)+(H-h1), M-1)$ respectively). Maybe, we have to comment why 'b' $\leftrightarrow$ $P(0, l1, h2, V+(L-l1)+(H-h2), M-1)$ and 'c' $\leftrightarrow$ $P(0, l2, h1, V+(L-l2)+(H-h1), M-1)$. The reason is very simple: the fake coin can not occur at lighter and heavier pans in series. Really, if the fake was lighter, then it can occur at lighter pans only; otherwise it can occur at heavier pans only.

The axiomatization AxP is finished. It is sound and complete in the following sense.

Proposition 1. *For every tuple of non-negative integer values (U, L, H, V, M), the following equivalence holds: $AxP \vdash P(U, L, H, V, M)$ iff it is possible to detect a single fake in any set that consists of $N = U + L + H + V$ coins by balancing them M times at most (where U, L, H, and V have the same meaning as in the preliminaries).*

Proof (sketch) by induction on $S = U + L + H$. The basic cases when $S \leq 1$ are straightforward. Induction step follows from the following simple argument: for every tuple (U, L, H, V, M), if $(U + L + H) > 1$, then there exists a "corresponding" axiom in AxP with the left-hand side (w.r.t. '$\leftrightarrow$') that matches $P(U, L, H, V, M)$; in the right-hand side of the corresponding axiom P is applied to data (U', L', H', V', M') such that $(U' + L' + H') < (U + L + H)$. ■

This axiomatization AxP is a logic problem formulation for puzzles that are similar to our original non-parameterized one, i.e. queries of the following kind: Whether it is possible to detect fake in a set of coins by balancing them M times. The corresponding logic solution (a pseudocode of a logic algorithm) should be a knowledge base that comprises a list of facts and a list of executable inference rules, that can be used by logic inference machine for efficient deduction of valid queries.

In particular, the first axiom can be transformed into the following four facts: $P(0, 0, 0, V, M)$, $P(1, 0, 0, V, M)$, $P(0, 1, 0, V, M)$, and $P(0, 0, 1, M)$.

The second axiom can be converted into the following rule

$$P(U, L, H, V, M) \ :- \ L + H > 0 \ \& \ P(0, L, H, V + U, M)$$

and be placed in the list of rules at any place; but it makes sense to transform it into the following simpler rule

$$P(U, L, H, V, M) \ :- \ P(0, L, H, V + U, M)$$

and place this simpler rule into the end of the rule list (where it will be used after the rule that corresponds to the third axiom in the case when $L + H > 0$).

The remaining two axioms can be transformed it into the following rules:

$$P(U,0,0,V,M) \ :- \ \exists u1,u2 : 0 < u1+u2 \le U \ (|u1-u2| \le V \& \\ P(U-u1-u2,0,0,V+u1+u2,M-1) \ \& \\ P(0,u1,u2,V+(U-u1-u2),M-1) \ \& \\ P(0,u2,u1,V+(U-u1-u2),M-1)),$$

$$P(0,L,H,V,M) \ :- \ \exists l1,l2 : l1+l2 \le L; \ \exists h1,h2 : h1+h2 \le H \\ (0 < l1+l2+h1+h2 \ \& \ |l1+h1-l1-l2| \le V \ \& \\ P(0,L-l1-l2,H-h1-h2,V+l1+l2+h1+h2,M-1) \ \& \\ P(0,l1,h2,V+(L-l1)+(H-h2),M-1) \ \& \\ P(0,l2,h1,V+(L-l2)+(H-h1),M-1)),$$

and can be put at the top of the rule list in any order (since they match disjoint cases). These rule are executable since all existential quantifiers are bounded.

This logic program is able to solve our initial non-parameterized puzzle by proving $P(14,0,0,1,3)$, and solve other puzzles of several different types. For example, it can solve how many balancing is sufficient for detecting a single fake in a set of 39 coins with aid of additional marked valid coin; if one would like to solve the puzzle than he/she may query this program P(39, 0, 0, 1, ?), and logic inference machine can answer by "finding" some value M, such that $P(39,0,0,1,M)$ is provable in AxP; unfortunately, this value could be any value greater than 4 (39 for example), but not the minimal one (that is 4). It means that we have solved the non-parameterized puzzle, but still have to solve the parameterized one.

3 Functional approach to balancing

Problem formulation in functional paradigm is a set of equations ("equational theory") that should to be valid equalities for functions that we want to compute; functional solution (functional algorithm or program) is set of function definitions that can be effectively used to compute function values for given arguments that convert the equational theory into valid equalities after instantiation. Let us apply below the functional paradigm to our parameterized puzzle.

Proposition 2. *Let us consider a function*[1] $M : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ *that for every tuple of arguments* U, L, H, V *returns the least number of balancing that is sufficient for detecting a single fake in any set of* $N = U + L + H + V$ *coins (where* U, L, H *and* V *have the same meaning as in the preliminaries). Then the following four equalities are valid:*

1. *if* $U + L + H \le 1$ *then* $M(U,L,H,V) = 0$;
2. *if* $L + H > 0$ *then* $M(U,L,H,V) = M(0,L,H,V+U)$;
3. $M(U,0,0,V) = \ 1 + \min_{0<u1+u2\le U, \ |u1-u2|\le V}$
$$\max\{M(U-u1-u2,0,0,V+u1+u2), \\ M(0,u1,u2,V+U-u1-u2), \\ M(0,u2,u1,V+U-u1-u2)\};$$

[1] $\mathcal{N}$ is the set of natural numbers, not a natural number (the amount of coins in particular).

$$4.\ M(0,L,H,V) = 1 + \min_{l1+l2\le L,\ h1+h2\le H,\ 0<l1+l2+h1+h2,\ |l1+h1-l1-l2|\le V} \max\{M(0,L-l1-l2,H-h1-h2,V+l1+l2+h1+h2),\ M(0,l1,h2,V+(L-l1)+(H-h2)),\ M(0,l2,h1,V+(L-l2)+(H-h1))\}$$

Proof (sketch). These equalities follows from corresponding axioms for predicate p from the previous section 2. Validity of the first two equalities are straightforward, but validity of the last two equalities requires some comments. These equalities start with increment '1+' since corresponding axioms reduce number of balancing from M to $(M-1)$. They have min instead of existential quantifiers, since the function returns the least number of balancing. Finally, max stays on place of conjunction since it represents the worst in three possible outcomes (cases 'a', 'b' and 'c' in the previous section). ■

Proposition 3. *There exists unique function* $M : \mathcal{N}\times\mathcal{N}\times\mathcal{N}\times\mathcal{N}\to\mathcal{N}$ *that satisfies equalities 1 – 4 in the proposition 2.*

Proof (sketch) Existence has been proved in the proposition 2. Uniqueness in this case means equality of any function M' (that satisfies equalities 1 – 4 in the proposition 2) to the function M that has been specified in the formulation of the proposition 2. But equality of two functions M and M' means equality of function values $M(U,L,H,V)$ and $M'(U,L,H,V)$ for every quadruple (U,L,H,V) of argument values; equality of these values can be proved by induction on $S = U+L+H$. ■

So we can adopt equalities 1 – 4 in the proposition 2 as the equational theory for the parameterized fake-coin puzzle; together with type definition $M : \mathcal{N}\times\mathcal{N}\times\mathcal{N}\times\mathcal{N}\to\mathcal{N}$ these equational theory becomes a functional problem formulation.

The corresponding functional solution (functional algorithm) is the following single function definition that comprises seven clauses:

$$\begin{aligned}
M : (1,0,0,V) &= 0;\\
(0,1,0,V) &= 0;\\
(0,0,1,V) &= 0;\\
(0,0,0,V) &= 0;\\
(U,0,0,V) &= 1 + \min_{0<u1+u2\le U,\ |u1-u2|\le V} \max\{M(U-u1-u2,0,0,V+u1+u2),\\
&\qquad M(0,u1,u2,V+U-u1-u2),\ M(0,u2,u1,V+U-u1-u2)\};\\
(0,L,H,V) &= 1 + \min_{l1+l2\le L,\ h1+h2\le H,\ 0<l1+l2+h1+h2,\ |l1+h1-l1-l2|\le V} \max\{M(0,L\ l1-l2,H-h1-h2,V+l1+l2+h1+h2),\\
&\qquad M(0,l1,h2,V+(L-l1)+(H-h2)),\\
&\qquad M(0,l2,h1,V+(L-l2)+(H-h1))\};\\
(U,L,H,V) &= M(0,L,H,V+U).
\end{aligned}$$

Here first four clauses represent all solutions of the equation 1, fifth and sixth clauses define how to compute function values to solve equations 3 an 4 respectively. The last clause corresponds to the equation 2. Let us remark that this equation could be converted into the following clause $(U,L,H,V) = if\ L+H>0\ then\ M(0,L,H,V+U)$ and be placed in between the fourth and the

fifth clauses, but (like in logic solution) we transform it into $(U, L, H, V) = M(0, L, H, V + U)$ and place at the end of the definition since at this place it will be used only in the case when $L + H > 0$.

Finally let us exercise manually the above functional solution for some simple data. For example, for values $U = 4$, $L = H = 0$ and $V = 1$, that correspond to question how many balancing is sufficient to detect a single fake in a set of 4 coins with use of one additional valid coin[2]:

$$\begin{aligned} M(4,0,0,1) &= 1 + \min_{0<u1+u2\le 4,\ |u1-u2|\le 1} \\ &\quad \max\{M(4-u1-u2,0,0,1+u1+u2),\ M(0,u1,u2,5-u1-u2), \\ &\quad M(0,u2,u1,5-u1-u2)\} = \\ &= 1 + min_{(u1,u2)\in\{(0,1),(1,0),(1,1),(1,2),(2,1),(2,2)\}} \\ &\quad \max\{M(4-u1-u2,0,0,1+u1+u2),\ M(0,u1,u2,5-u1-u2), \\ &\quad M(0,u2,u1,5-u1-u2)\} = \\ &= 1 + min\{\max\{M(3,0,0,2), M(0,0,1,4), M(0,1,0,4)\}, \\ &\quad \max\{M(3,0,0,2), M(0,1,0,4), M(0,0,1,4)\}, \\ &\quad \max\{M(2,0,0,3), M(0,1,1,3), M(0,1,1,3)\}, \\ &\quad \max\{M(1,0,0,4), M(0,1,2,2), M(0,2,1,2)\}, \\ &\quad \max\{M(1,0,0,4), M(0,2,1,2), M(0,1,2,2)\}, \\ &\quad \max\{M(0,0,0,5), M(0,2,2,1), M(0,2,2,1)\}\}. \end{aligned}$$

Let us observe that in this exercise all but one functional calls are duplicated: $M(3,0,0,2)$, $M(0,1,1,3)$, $M(0,1,2,2)$, $M(0,2,1,2)$, and $M(0,2,2,1)$; the single call without duplicates is $M(2,0,0,3)$. Moreover, if to proceed further with our exercise, the number of duplicated (triplicated and so on) of calls will increase.

It leads to a natural idea to compute new emerging function calls once, save their values, and reuse saved values when a duplication call emerge. These technique is well-known optimization in functional programming as memoization [3]. Applying this optimization and omitting all these intermediate computations we can finish the above exercise as follows:

$$\begin{aligned} M(4,0,0,1) &= 1 + \min\{\max\{2,0,0\},\ \max\{2,0,0\}, \\ &\quad \max\{1,1,1\},\ \max\{0,1,1\},\ \max\{0,1,1\},\ \max\{0,2,2\}\} = 2. \end{aligned}$$

4 Categorical Imperative: compute efficiently!

Memorization is not unique technique that can be applied to optimization of the execution of the above functional algorithm. Lazy computations can help also. In particular, lazy memoization consists in saving all emerging function calls, then (when no new calls emerge) sort all saved calls (according to call dependence), execute each saved call once, save all computed values, and use saved values when they become mandatory (inevitable) in computations.

For instance, in the above exercise with functional call $M(4,0,0,1)$ we had several emerging functional calls: $M(0,1,1,3)$, $M(0,1,2,2)$, $M(0,2,1,2)$, $M(3, 0, 0, 2)$, $M(0,2,2,1)$, ... Of course, $M(0,1,1,3)$ is "simpler" than $M(0,1,2,2)$ and $M(0,2,1,2)$, since in the former case just 2 coins are not-verified, while

[2] We omit technical details of this exercise.

there are 3 coins to be verified in the later cases. Call $M(3,0,0,2)$ is simpler that $M(0,2,2,1)$ due to the same reason. In contrast, calls $M(0,1,2,2)$ and $M(0,2,1,2)$ are simple that $M(3,0,0,2)$, since in the former cases the balancing agent has some knowledge about coins while in the later case any information is unavailable. So, for the sake of efficiency, values of $M(0,1,1,3)$, $M(0,1,2,2)$, $M(0,2,1,2)$, $M(3,0,0,2)$, $M(0,2,2,1)$ have to be computed and saved in this order.

But why effectiveness should rely upon computation optimization via lazy memoization? Why not to use memory explicitly and compute in advance all values for all possible (and, maybe, some extra) function calls (but in the right sequence: the simplest — the first, the hardest — the last)? In our particular case (the parameterized fake-coin puzzle) some upper approximation for the set of all possible function call is easy to predict.

Proposition 4. *Let $M : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ be a function defined by the functional algorithm in the previous section 3. Then for all non-negative integers U, L, H, and V the set $FCM(U,L,H,V)$ of all function calls that occur in the computation of $M(U,L,H,V)$ (according to the definition in the section 3) is contained in one of the following sets*

- $\{M(0,L,H,V') \ : \ V' \le V+U\} \cup$
  $\cup\{M(0,L',H',V') \ : L' < L, \ H' < H, \ V' \le V+U+L+H-L'-H'\}$,
  if $U > 0$ and $L+H>0$;
- $\{M(0,L',H',V') \ : \ L' < L, \ H' < H, \ V' \le V+U+L+H-L'-H'\}$,
  if $U = 0$ and $L+H>0$;
- $\{M(U',0,0,V') \ : \ U' < U, \ V' \le V+U-U'\} \cup$
  $\cup \ \{M(0,L',H',V') \ : \ U \ge L'+H', \ V' \le V+U-L'-H')\}$,
  if $U > 0$ and $L+H=0$.

Proof (sketch) by induction on $S = U+L+H$. ■

Imperative problem formulation consists in a command that decrees to compute a set of data values in accordance explicit rules of data manipulation and transformation. Imperative problem solution (imperative algorithm or program) is a well-defined sequence of operators that are commands of transformations of values that are stored in individual elements of a computer memory ("memory cells").

A preliminary imperative formulation of the parameterized puzzle can be as follows: for a given integer value $N \ge 0$ fill-in a four-dimensional table $T[0..N, 0..N, 0..N, 0..N]$ by corresponding values of the function M specified in the proposition 2, i.e. by integers such that for every tuple (U,L,H,V) of non-negative integers, if $U+L+H+V \le N$, then $T[U,L,H,V]$ is the least number of balancing to detect a single fake in a set of $U+L+H+V$ coins (where U, L, H, V have the same meaning as in the preliminaries).

According to our functional solution of the parameterized fake-coin puzzle, this preliminary imperative formulation can be transformed into the following final one: for a given integer value $N \ge 0$ fill-in a four-dimensional table $T[0..N, 0..N, 0..N, 0..N]$ in accordance with following rules (whenever the sum

of the indexes $\le N$):
$T(1,0,0,V) = T(0,1,0,V) = T(0,0,1,V) = T(0,0,0,V) = 0$ for all $V \in [0..N]$;
$T(U,0,0,V) \;=\; 1 + \min_{0<u1+u2\le U,\ |u1-u2|\le V}$
$\max\{T(U-u1-u2,0,0,V+u1+u2),\ T(0,u1,u2,V+U-u1-u2),$
$T(0,u2,u1,V+U-u1-u2)\}$;
$T(0,L,H,V) \;=\; 1 + \min_{l1+l2\le L,\ h1+h2\le H,\ 0<l1+l2+h1+h2,\ |l1+h1-l1-l2|\le V}$
$\max\{T(0,L-l1-l2,H-h1-h2,V+l1+l2+h1+h2),$
$T(0,l1,h2,V+(L-l1)+(H-h2)),$
$T(0,l2,h1,V+(L-l2)+(H-h1))\}$;
$T(U,L,H,V) \;=\; T(0,L,H,V+U)$.

The above Proposition 4 leads to an idea how to fill-in the table in the right sequence: in the order of ascending of sum of indexes $U+L+H+V$, but for every value of the sum in $[0..N]$, elements $T[0,L,H,V]$ should be filled first, next — elements $T[U,0,0,V]$, finally — elements $T[U,L,H,V]$. This idea can be implemented by imperative solution as follows:

```
constn;
var u,l,h,v,s,l1,l2,h1,h2,u1,u2 : integer;
var t : integer array of [0..n,0..n,0..n,0..n];
begin
for v = 0 to n do
begin t(1,0,0,v) := 0; t(0,1,0,v) := 0; t(0,0,1,v) := 0; t(0,0,0,v) = 0 end;
for s = 2 to n do
    begin
    for l = 0 to s do
        for h = 0 to (s-l) do
            t(0,l,h,s-l-h) := 1+
                min_{l1+l2≤l, h1+h2≤h, 0<l1+l2+h1+h2, |l1+h1-l1-l2|≤(s-l-h)} max{
                    t(0,l-l1-l2,h-h1-h2,s-l-h+l1+l2+h1+h2),
                        t(0,l1,h2,s-l1-h2), t(0,l2,h1,s-l2-h1)};
    for u = 0 to s do
        t(u,0,0,s-u) := 1 + min_{0<u1+u2≤u, |u1-u2|≤(s-u)} max{
                t(u-u1-u2,0,0,s-u+u1+u2), t(0,u1,u2,s-u1-u2),
                                          t(0,u2,u1,s-u1-u2)};
    for u = 0 to s do
        for l = 0 to (s-u) do
            for h = 0 to (s-u-l) do
                t(u,l,h,v) := t(0,l,h,s-l-h)
    end
end.
```

Imperative solution is done. Let us remark, that it was extracted from a functional one by lazy memoization, which, in turn, was extracted from a logic solution.

5 Conclusion

We start this paper with logic solution for a puzzle "*Is balancing M-times sufficient for detecting the fake in a set of coins?*", and finishes with the imperative algorithm, that effectively computes the minimal number of balancing that is sufficient for detection the fake. Thus the paper can be considered as a kind of case-study of algorithm (a program) inversion.

At the same time some ideas (that have been exploited in this paper) can be generalized for inverting some logic and functional algorithms and programs. In particular, let us assume that a functional algorithm (program) defines a partial function $F : D \to R$ such that for every $X \in D$ it is easy to compute an upper approximation of the set of all function calls $FCF(X)$, that emerge in the computation of $F(X)$. Then the function F can be efficiently inverted in the following sense: for every $Y \in R$, for every finite $D' \subseteq D$ the set[3]

$$(F \restriction D')^{-}(Y) \;=\; \{X \in D' \;:\; F(X) = Y\}$$

can be computed by dynamic programming. For instance, function $M : \mathcal{N} \times \mathcal{N} \times \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ (that was discussed in section 3) enjoys this property (see Proposition 4) and can be inverted by means of the table T (computed in the section 4 by dynamic programming).

Let us also remark that the parameterized puzzle has been used in Asian Regional ACM International Collegiate Programming Contest in year 2000 (problem H). One can try the following (more complicated) version of the puzzle [4]:

> Write a program with 3 inputs: a number $U \geq 0$ of coins under question, a number $V \geq 0$ of marked valid coins, and a limit $K \geq 0$ for the number of balancing, that outputs either the string impossible, or another executable interactive program ALPHA (in the same language) with respect to existence of a strategy to identify a single false coin among U coins with use of additional V marked valid coins and weighing coins K times at most. Your program should output impossible iff there is no such strategy. Otherwise it should output the program ALPHA which implements a strategy in the following settings.
> - All $(U+V)$ coins are enumerated by consecutive numbers from 1 to $(U+V)$, all marked valid coins are enumerated by initial numbers from 1 up to V.
> - Every interactive session with ALPHA begins with user's initial decision on the coin number of the false coin in $[(V+1)..(V+U)]$ and whether it is lighter or heavier.
> - Every interactive session with ALPHA consists of a series of rounds and the number of rounds in the session can not exceed K.
> - In each round $i \in [1..K]$ the program ALPHA outputs two disjoint subsets of coin numbers to be placed on the first and the second pans of the balance and prompts the user with '?' for a reply.

[3] '$\restriction$' denotes domain restriction of a function.

- The user in his/her turn replies with '<', '=', or '>' in accordance with the initial decision on the number of the false coin and its weight.
- Every interactive session with ALPHA finishes with the final output string "False coin number is " followed by the coin number of the false coin.

Since the problem is to write a program which generates another program we would like to refer to the first program as a *metaprogram* and to the problem as the *metaprogram* problem respectively. This problem has been discussed in the context of propositional program logics and solved in [4].

References

1. Floyd R.W. *The paradigms of Programming.* Communications of ACM. v.22, 1979, p.455-460.
2. Kuhn T.S. *The structure of Scientific Revolutions.* Univ. of Chicago Press, 1970.
3. Astapov D. *Recursion + Memoization = Dynamic Programming.* (in Russian) Practice of Functional Programming, n.3, 2009, p. 17-33. Available at http://fprog.ru/2009/issue3/.
4. Shilov N.V. and Yi K. *How to find a coin: propositional program logics made easy.* Current Trends in Theoretical Computer Science, World Scientific, v. 2, 2004, p.181-214.

A A Puzzle for You...

Finally I would like to draw attention to another coin puzzles, which can be called Find Money Puzzle. Again, let us start with a non-parameterized puzzle[4] and finish with a parameterized version[5]. Non-parameterized puzzle (enjoy!):

> A set consists of 40 coins, three of them are fake and 37 are valid ones. All coins look identical, all valid coins have equal weight, but all fakes are lighter than the valid ones. Is it possible to select 18 valid by balancing coins 3 times at most?

Parameterized Find Money Puzzle (logic, functional, and imperative feasible algorithms are welcome by e-mail to author):

> Write a program that inputs a number $N \geq 0$ of coins, a number $L \in [0..N]$ of fake coins (the remaining $(N - L)$ coins are valid), a number $V \in [0..(N - L)]$, and outputs the least number of balancing M that is always sufficient for selecting V valid coins in this set. (Assume that all valid coins have one and the same weight while fake coins are lighter.)

[4] I know how to solve it.
[5] I do not know how to solve it.