ANALYSING DNA MICROARRAY DATA USING BOOLEAN TECHNIQUES

Ondřej Čepek [a] David Kronus [b] Petr Kučera [c]

RRR ,

[a]Faculty of Mathematics and Physics, Charles University in Prague, Malostranské náměstí 25, Prague, Czech Republic; (ondrej.cepek@mff.cuni.cz), and also Institute of Finance and Administration (VŠFS), Prague, Czech Republic; (cepek@vsfs.cz)

[b]Faculty of Mathematics and Physics, Charles University in Prague, Malostranské náměstí 25, Prague, Czech Republic; (david.kronus@matfyz.cz)

[c]Faculty of Mathematics and Physics, Charles University in Prague, Malostranské náměstí 25, Prague, Czech Republic; (kucerap@ktiml.ms.mff.cuni.cz)

Rutcor Research Report
RRR 08-2008 , May 2008

Analysing DNA Microarray Data Using Boolean Techniques

Ondřej Čepek David Kronus Petr Kučera

Abstract. We address in this manuscript a problem arising in molecular biology, namely a problem of discovering dependencies among gene expression levels. The problem is formulated in mathematical terms as a search for a fully defined three valued function defined on three valued variables which is partially specified by the DNA microarray measurments. This formulation as well as our solution methods are strongly motivated by results in the area of logical analysis of data (LAD) and in the area of partially defined Boolean functions (pdBfs), in particular by procedures for finding fully defined extensions of pdBfs. On one hand we present several algorithms which (under some assumptions) construct the desired three valued functional extension of the input data, and on the other hand we derive several proofs showing that (under different assumptions) finding such an extension is NP-hard.

Acknowledgements: The research was supported by Czech Science Foundation (GACR) under the contracts no. 201/07/0205 (first author), no. 201/05/H014 (second author) and no. 201/07/P168 (third author).

1 Introduction

One of the principal questions asked in molecular biology is how genes regulate each other or in other words how a status (expresion level) of some selected gene X depends on the expression levels of other genes. Answering this question means doing two things: 1) detecting the subset S of genes (among all measured genes) which influence the selected gene X and 2) discovering for each gene Y in the detected subset S in which way it influences the selected gene X. There are basically only two ways in which Y can influence X: either positively (if the expression level of Y goes up, so does the expression level of X) or negatively (if the expression level of Y goes up, the expression level of X goes down). The tool, which makes it possible to measure the expression levels of genes (in fact levels of thousands of genes in parallel), is the DNA microarray technology [19, 12]. There are many methods which attempt to use the DNA microarray data to discover dependencies among genes (in biological literature this is called *gene regulatory network modeling*) which use various tools: linear models [22, 13], Bayesian networks [18, 14], neural networks [23], edge-labelled directed graphs [10], graphical Gaussian modelling [21], and dynamical system models based on differential equations [15]. A modeling method which is of a particular interest in this paper is the so-called Boolean Network model originally introduced in [16]. This model has been expanded in recent years in several directions, all attempting to identify the structure of gene regulatory networks from expression data [17, 1, 3, 4, 2, 20]. However, all these Boolean Network models assume that the expression levels of all genes are explicitely available and, moreover, that they can be discretized into just two Boolean values (gene is either ON or OFF). This is not a realistic assumption as the DNA microarray data in fact contain expression level ratios (not the levels themselves) [9], and experimental evidence seems to suggest these ratios are clustered in three not two groups. Let us try to explain the last sentence in more detail.

The DNA microarray data consists of vectors, the length N of which corresponds to the number of measured genes (typically several thousand). Each vector is produced by a following experiment: the DNA is taken in a so called *wild state* in which no gene is artificially manipulated and then a selected gene i (where the number of a gene corresponds to its position in the vector of measurements) is *disrupted* (or suppressed) which means that its expression level is fixed to zero by some genetic manipulation. Such a disruption may of course cause expression levels of other genes to change as well. What is actually measured by the DNA microarray technology are the changes in the expression levels of individual genes during the disruption process, or more precisely logarithms of the ratios (for every gene j, $1 \leq j \leq N$) of the expression level $e_i(j)$ of gene j in the DNA with a disrupted gene i and the expression level of gene j in the starting wild state. We denote the value in component j of the resulting vector (measured in the experiment with gene i being disrupted) by $r_i(j)$. Hence

$$r_i(j) = \log \frac{e_i(j)}{e(j)} = \log e_i(j) - \log e(j) \ \ .$$

Let us note that $e_i(i) = 0$ always holds (the level of the disrupted gene is fixed at zero[1]), and this is the only expression level we always know for sure. All the other expression levels must be deduced from the ratios (which cannot be always done uniquely, as we shall see below). Finally let us note that a given DNA may have many different wild states (each individual wild state of the DNA may be determined by the exact conditions under which the particular experiment is taking place) and so the wild states in different experiments (each producing one vector of $r_i(j)$ values) may differ. Thus we can think of the DNA microarray data as a set of vectors, where each vector of ratios corresponds to a pair of unknown expression levels which should be deduced from the ratios, while there is no apparent dependency between the values of the same gene in different pairs.

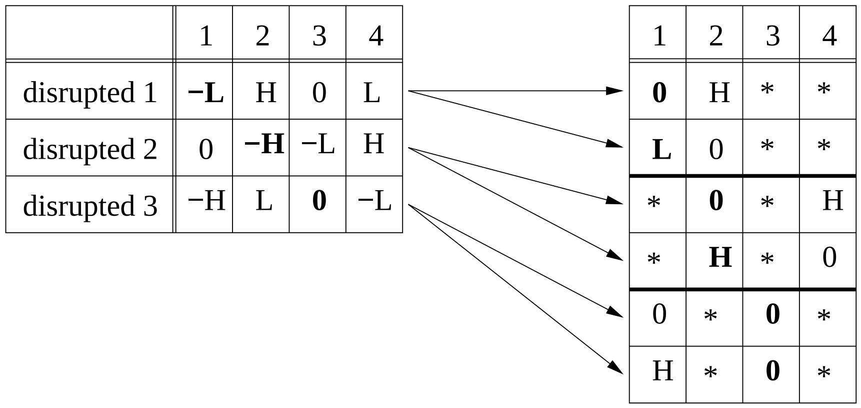
As mentioned above, the expression levels of genes can be discretized in three levels denoted $\{0, L, H\}$ where L stands for *low* and H for *high*. Consequently, the change in expression level can be of three types, namely no change, small change, or large change. However, since the change can be either positive or negative (expression level $e_i(j)$ of gene j goes either up or down compared to its wild state, i.e. the ratio $r_i(j)$ either increases or decreases) the values $r_i(j)$ are discretized in five levels $\{-H, -L, 0, L, H\}$. The correspondence between the discretized values of $r_j(i)$ on one hand and the discretized values of $e_j(i)$ and $e(i)$ can be specified as follows:

$r_j(i)$	$e_j(i)$	$e(i)$
H	H	0
L	L	0
	H	L
	0	0
0	L	L
	H	H
$-L$	0	L
	L	H
$-H$	0	H

Note, that if the discretized value of ratio $r_i(j)$ is H or $-H$ then the corresponding expression levels $e(j)$ and $e_i(j)$ are fully determined and hence the values $e(j)$ and $e_i(j)$ can be uniquely deduced from the value $r_i(j)$. In the following example in Figure 1 we consider

[1]In fact it is an "idealized zero", or in other words a value close enough to zero, which is not a problem because the values we will work with discretize intervals of real values. The reason is that if $e_i(i)$ is zero then $r_i(i)$ is minus infinity, which is not a measurable value. In practise $e_i(i)$ is zero when $r_i(i)$ is sufficiently negative.

three experiments and four genes. The fields in the right hand side table which contain an asterisk cannot be deduced uniquely. The values which can be determined uniquely are of two types: the expression levels of the disrupted gene (zero in the disrupted mutant DNA and an easily derivable value in the wild type) which are typed in boldface, and expression levels corresponding to H and $-H$ ratios.



	1	2	3	4
disrupted 1	**−L**	H	0	L
disrupted 2	0	**−H**	−L	H
disrupted 3	−H	L	**0**	−L

1	2	3	4
0	H	*	*
L	0	*	*
*	**0**	*	H
*	**H**	*	0
0	*	**0**	*
H	*	**0**	*

Figure 1: Interpretation of discretized ratios

Now let us assume that we want to find out whether the expression levels of the fourth gene depend on the expression levels of the first three. In other words we want to find a three-valued function f of three (or more, if we had more genes in our sample) three-valued variables which takes first three columns in the table as an input and outputs the fourth column. The table can be then viewed as a partial definition of the function where some rows are specified fully, some rows are specified partially (some values - missing bits - are replaced by asterisks) and some rows (combinations of input values) are missing form the table entirely. The sought function is then a fully defined *extension* of the partial definition. In fact there are three different natural problems we may consider:

1. Does there exist a feasible replacement of missing bits by $\{0, L, H\}$ values for which there exist an extending function?

2. Does there for every feasible replacement of missing bits exists an extending function (possibly different functions for different replacements)?

3. Does there exists a function which is an extending function for all feasible replacements of missing bits?

All of the above problems will be addressed in this paper. However, first we have to discuss what are the natural conditions such a function must fulfil and next we have to specify what do we understand by a feasible replacement of missing bits.

We already know from the discussion above that gene i can influence gene j in only two ways. Either an increase of the expression level of i always causes an increase of the expression level of j (possibly negligible or zero) or an increase of the expression level of i

always causes a decrease of the expression level of j (possibly negligible or zero). This means that the desired function f must be monotone in all its variables, or in other words f must be unate. However, such a goal is most likely computationally intractable, as it was already shown [11] that even in the binary case (two-valued function on two-valued variables) with no missing bits (no asterisks in the input table) it is NP-hard to decide whether there exists a unate extension of the input data. Therefore we simplify the task by looking only for functions that are positive in all variables, which will allow us to design effective algorithms solving the task. Although this restriction is rather strong, for a reasonably small number of variables (genes) these algorithms can be used to find also general unate extensions by subsequently considering all possible combinations of positive and negative monotonicities of input variables, modifying in each case the input vectors accordingly, and searching for a positive functional dependency on the modified data. It should be noted, that if there are no missing bits, the positive extension problem is computationally quite easy. The binary case of this problem has been already studied (e.g. in [6]) and it would not be hard to generalize the result in a straightforward way to the ternary case. So let us now look at the case with missing bits and the rules for their feasible replacements.

A similar extension problem called an *extension of a partially defined Boolean functions with missing data* was studied and solved in [7]. There are however, aside of the difference between binary (Boolean) and ternary values, two more issues in which the problem studied in [7] is different. First of all, the missing bits appear only in the input part of the table, i.e. the function value is always defined, which is not true in our ternary case (see example in Figure 1). Secondly, an arbitrary binary value can be substituted for every missing bit, i.e. there are no dependencies among the values of different missing bits. However, in our case the missing bits cannot be replaced arbitrarily because there are dependencies among the asterisks in the two vectors which were formed from a single vector of ratios. In fact there are three types of asterisks as can be seen from example in Figure 2:

1. The asterisks in a pair that corresponds to ratio value 0 must have the same value. Hence these asterisks, e.g. in the first and second line in the third column and in the third and fourth line in the first column, are of an "equal" type. We will denote such asterisks by $*_=$.

2. The asterisks in a pair that corresponds to ratio values $-L$ or $+L$ must have values that differ by "one step" (0 and L or L and H). Hence e.g the asterisk in the first line and the fourth column is of a "higher" type (we will denote it by $*_+$), and the asterisk second line and the fourth column is of a "lower" type (we will denote it by $*_-$).

Using the just defined types of asterisks we can rewrite the table from example in Figure 1 as in Figure 2.

The second table shows a feasible replacement of asterisks by ternary values and the third table shows an infeasible replacement in which the infeasible values are typed in boldface. Now let us summarize how the input data to our problem look like in general. It is a table of ternary values, where some values may be missing and each such value is replaced

1	2	3	4
0	H	$*_=$	$*_+$
L	0	$*_=$	$*_-$
$*_=$	0	$*_-$	H
$*_=$	H	$*_+$	0
0	$*_+$	0	$*_-$
H	$*_-$	0	$*_+$

(a) Original data

1	2	3	4
0	H	L	L
L	0	L	0
H	0	L	H
H	H	H	0
0	L	0	L
H	0	0	H

(b) Feasible assignment

1	2	3	4
0	H	**0**	**H**
L	0	**L**	**H**
L	0	0	H
0	H	L	0
0	**0**	0	0
H	**L**	0	L

(c) Infeasible assignment

Figure 2: Different asterisks and feasibility of a replacement

by one of the three types of asterisks. Furthermore, the rows in the table are paired to reflect the fact that each such pair corresponds to a single row of ratios. The goal is to find a positive ternary function which extends the partial definition given by the table. This problem broadly speaking belongs to the area of logical analysis of data (LAD) pioneered by Peter L. Hammer (see e.g. [5] for a review of LAD related papers) and more specifically to the study of partially defined functions. Our work is a generalization of extension problems on partially defined Boolean functions with missing data (pBmds) for which several variants were studied in [7] and [8].

In practice, the situation is even more complicated. The measured vectors of ratios contain hundreds or even thousands of genes, which makes any extension problem computationally intractable. However, according to practical experience every gene depends on a relatively small number of other genes (no more than 30). Moreover, techniques are available, which reduce the number of possible candidates for the influencing genes from hundreds or thousands to below one hundred. This makes it possible to reduce the original table with hundreds or thousands of columns to a table with at most one hundred columns. The task which remaines to be solved is how to reduce this number of columns further to the desired number around 30. Obviously, trying all subsets of 30 coulumns out of roughly 100 columns is not computationally feasible. Therefore our simplified approach is still just a first step to practical usability of the proposed model.

The paper is organized as follows. The next section introduces the necessary notation and formalizes the problems which are addressing in the rest of the paper. Section 3 studies the complexity of constructing various extensions with no monotonicity restriction on the extending functions. Section 4 then studies the same cases under the positivity constraint.

2 Definitions and basic properties

In this section we formalize the definitions outlined in the previous section. We consider three possible (discretized) expression levels of genes over $\mathbb{Z}_3$. For easier manipulation, we map L to 1 and H to 2 (and use the numeric values from now on) while the symbols O, L, H will be hereunder used as the names of sets of vectors.

Definition 2.1 A *partially defined 3-valued function* (or *pd3f* in short) on n (3-valued) variables is a 3-tuple $F = (O, L, H)$, where $O, L, H \subseteq \mathbb{Z}_3^n$. The symbol $F(a)$ for a vector $a \in O \cup L \cup H$ denotes the value assigned to a by F and is defined by $F(a) = 0$ if $a \in O$, $F(a) = 1$ if $a \in L$, and $F(a) = 2$ if $a \in H$.

Definition 2.2 The class of all three valued functions $f : \mathbb{Z}_3^n \mapsto \mathbb{Z}_3$ will be denoted by $\mathcal{C}_{3-all}$. The class of all positive three valued functions will be denoted by $\mathcal{C}_3^+$.

The way in which pd3fs emerge in our problem is the following: when we choose gene i and we want to decide whether it is functionally dependent on some other genes we look at the vectors of expression levels (obtained in pairs from the vectors of ratios), select the i-th component as the functional value (determining in which of the sets O, L, H the corresponding vector belongs) and take the values of other bits as the individual inputs of the function. This would work if we were given fully defined vectors of expression levels. However, since we have some asterisks (of three types) in the vectors of expression levels, and the vectors need to be paired to capture the missing data properly, we need the following definition to fully represent the situation in question.

Definition 2.3 Let $\mathbb{E} = \{0, 1, 2, *_=, *_-, *_+\}$ be an enlargement of $\mathbb{Z}_3$ by three symbols for missing bits. A *paired partially defined 3-valued function with missing bits and uncertain value* (or *p-p3mbuv* in short) on n variables is a pair $F = (\tilde{U}, \pi)$, where $\tilde{U} \subseteq \mathbb{E}^{n+1}$ is a set of vectors and $\pi : \tilde{U} \mapsto \tilde{U}$ is a pairing function satisfying $[\pi(u) = v \Leftrightarrow \pi(v) = u]$. Furthermore, if $u = \pi(v), v = \pi(u) \in \tilde{U}$ is a pair of vectors and $1 \leq i \leq n+1$ is an index then either $u_i, v_i \in \mathbb{Z}_3$, or both u_i and v_i are missing bits (or uncertain function values if $i = n+1$), in which case

$$[(u_i = *_=) \Leftrightarrow (v_i = *_=)] \wedge [(u_i = *_-) \Leftrightarrow (v_i = *_+)] \wedge [(u_i = *_+) \Leftrightarrow (v_i = *_-)].$$

Given vector $u \in \tilde{U}$, let us denote $I(u) = (u_1, \ldots, u_n) \in \mathbb{E}^n$ the *input part* of u and $F(u) = u_{n+1} \in \mathbb{E}$ the *output value* of u.

From now on we assume that the problem we are interested in takes a p-p3mbuv as its input. These are the pairs of vectors containing expression levels (and missing data marked by asterisks) which have been obtained from the vectors of discretized ratios using the procedure described in the previous section. The definition also expresses the fact that in paired vectors u, v there is an asterisk at u_i if and only if there is an asterisk at v_i. Moreover the type of asterisk at v_i is fully determined by u_i.

In the following we present a number of definitions and several simple lemmas. Our aim is to formalize the problem we are trying to solve. We start with definitions which help us easily describe assignments of values from $\mathbb{Z}_3$ to the missing components of vectors which are marked by asterisks.

Definition 2.4 Let $v \in \mathbb{E}^{n+1}$ and $\tilde{S} \subseteq \mathbb{E}^{n+1}$, then we define:

$$\begin{aligned}
ASI(v) &= \{(v,j) \mid v_j \in \{*_=, *_-, *_+\}, j = 1,2,\ldots,n\} \\
ASO(v) &= \{(v,n+1) \mid v_{n+1} \in \{*_=, *_-, *_+\}\} \\
AS(v) &= ASI(v) \cup ASO(v) = \{(v,j) \mid v_j \in \{*_=, *_-, *_+\}, j = 1,2,\ldots,n+1\} \\
ASI(\tilde{S}) &= \textstyle\bigcup_{v\in\tilde{S}} ASI(v) \\
ASO(\tilde{S}) &= \textstyle\bigcup_{v\in\tilde{S}} ASO(v) \\
AS(\tilde{S}) &= ASI(\tilde{S}) \cup ASO(\tilde{S}) = \textstyle\bigcup_{v\in\tilde{S}} AS(v)
\end{aligned}$$

Let $Q \subseteq AS(\tilde{S})$ and let $\alpha : Q \mapsto \mathbb{Z}_3$ be an assignment of values to the set Q of missing bits (and uncertain function values). For a vector $v \in \tilde{S}$ let v^α denote the vector obtained from v by replacing the $*$ components which belong to Q, by the values assigned to them, i.e.

$$v_j^\alpha = \begin{cases} v_j & \text{if } (v,j) \notin Q \\ \alpha(v,j) & \text{if } (v,j) \in Q \end{cases}$$

Definition 2.5 Let $\tilde{S} \subseteq \mathbb{E}^{n+1}$, and let $Q', Q'' \subseteq AS(\tilde{S})$ be two disjoint sets. Let $\alpha' : Q' \mapsto \mathbb{Z}_3$ and let $\alpha'' : Q'' \mapsto \mathbb{Z}_3$. Then we define a *composition of α' and α''* as an assignment $\alpha = \alpha' \circ \alpha'' : Q' \cup Q'' \mapsto \mathbb{Z}_3$ in a straightforward manner as follows: $a^\alpha = (a^{\alpha'})^{\alpha''}$ for every $a \in \tilde{S}$.

Let $\alpha : Q \mapsto \mathbb{Z}_3$ and let $Q' \subseteq Q \subseteq AS(\tilde{U})$, then we define a *restriction of α to Q'* denoted by $\alpha\restriction_{Q'}: Q' \mapsto \mathbb{Z}_3$, as an assignment, for which for every $v \in \tilde{S}$, and every $j = 1,\ldots,n,n+1$.

$$v_j^{\alpha\restriction_{Q'}} = \begin{cases} v_j & \text{if } (v,j) \notin Q' \\ \alpha(v,j) & \text{if } (v,j) \in Q' \end{cases}$$

Not all assignments fulfill the conditions that we have specified for the asterisk components. First of all there are some forbidden values from $\mathbb{Z}_3$ that a specific asterisk type cannot be assigned, e.g. asterisks $*_+$ cannot be assigned 0, and secondly the assignment must respect the pairing of vectors and hence, e.g., two corresponding asterisks $*_=$ in paired vectors must have the same value assigned. Moreover one of corresponding asterisks in paired vectors should be assigned a value from $\mathbb{Z}_3$ if and only if the other one is also assigned a value. In the following definitions we capture these conditions in a notion of a *feasible assignment.*

Definition 2.6 Let $a \in \mathbb{E}^{n+1}$, and $j \in \{1, \dots, n, n+1\}$. A *domain* of a position a_j is a subset of $\mathbb{Z}_3$ denoted by $\Delta(a, j)$ and defined as:

$$\Delta(a,j) = \begin{cases} \{0\} & \text{if } a_j = 0 \\ \{1\} & \text{if } a_j = 1 \\ \{2\} & \text{if } a_j = 2 \\ \{0,1,2\} & \text{if } a_j = *_= \\ \{0,1\} & \text{if } a_j = *_- \\ \{1,2\} & \text{if } a_j = *_+ \end{cases}$$

Definition 2.7 Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv and let $\alpha : Q \mapsto \mathbb{Z}_3$ be an assignment of values to a subset $Q \subseteq AS(\tilde{U})$. We say, that α is a *feasible assignment with respect to* F, if for every pair of vectors $u = \pi(v), v = \pi(u) \in \tilde{U}$ and every index $j = 1, 2, \dots, n, n+1$ all the following conditions are satisfied:

1. $[(u,j) \in Q \Leftrightarrow (v,j) \in Q]$ (i.e. α assigns values either to both or to none of the paired positions).
2. If $(u,j) \in Q$ and $u_j = v_j = *_=$, then $\alpha(u,j) = \alpha(v,j)$.
3. If $(u,j) \in Q$, $u_j = *_-$, and $v_j = *_+$, then $\alpha(u,j) = \alpha(v,j) - 1$.
4. If $(u,j) \in Q$, $u_j = *_+$, and $v_j = *_-$, then $\alpha(u,j) = \alpha(v,j) + 1$.

The set of all feasible assignments $\alpha : AS(\tilde{U}) \mapsto \mathbb{Z}_3$ with respect to p-p3mbuv $F = (\tilde{U}, \pi)$ will be denoted by $\mathcal{A}(\tilde{U}, \pi)$ or $\mathcal{A}(F)$.

Moreover, if $Q \subseteq ASI(\tilde{U})$, and $\alpha : Q \mapsto \mathbb{Z}_3$ is a feasible assignment with respect to p-p3mbuv $F = (\tilde{U}, \pi)$, then we say that α is a *feasible assignment to inputs of* F. The set of all feasible assignments to inputs of F, $\alpha : ASI(\tilde{U}) \mapsto \mathbb{Z}_3$, will be denoted by $\mathcal{A}_I(\tilde{U}, \pi)$ or $\mathcal{A}_I(F)$.

Similarly, if $Q \subseteq ASO(\tilde{U})$, and $\alpha : Q \mapsto \mathbb{Z}_3$ is a feasible assignment with respect to p-p3mbuv $F = (\tilde{U}, \pi)$, then we say that α is a *feasible assignment to outputs of* F. The set of all feasible assignments to outputs of F, $\alpha : ASO(\tilde{U}) \mapsto \mathbb{Z}_3$, will be denoted by $\mathcal{A}_O(\tilde{U}, \pi)$ or $\mathcal{A}_O(F)$.

The following lemma states a simple fact that two feasible assignments on disjoint sets of asterisks are independent.

Lemma 2.8 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv, and let $Q', Q'' \subseteq AS(\tilde{U})$ be disjoint. Let $\alpha' : Q' \mapsto \mathbb{Z}_3$ and $\alpha'' : Q'' \mapsto \mathbb{Z}_3$ be two feasible assignments. Then also assignment*

$\alpha = \alpha' \circ \alpha''$ is a feasible assignment. In particular, if $Q' = ASI(\tilde{U})$ and $Q'' = ASO(\tilde{U})$, then $\alpha' \in \mathcal{A}_I(F)$, $\alpha'' \in \mathcal{A}_O(F)$, and $\alpha \in \mathcal{A}(F)$.

Similarly let $Q \subseteq AS(\tilde{U})$, and let $\alpha : Q \mapsto \mathbb{Z}_3$ be a feasible assignment. Furthermore let $Q' \subseteq Q$ be a subset satisfying the first property in Definition 2.7 (i.e. $u = \pi(v), v = \pi(u)$ implies $[(u,j) \in Q' \Leftrightarrow (v,j) \in Q']$ for every j). then also $\alpha \lceil_{Q'}$ is a feasible assignment. In particular if $Q = AS(\tilde{U})$, then $\alpha \in \mathcal{A}(F)$. If $Q' = ASI(\tilde{U})$, then $\alpha' \in \mathcal{A}_I(F)$, and if $Q' = ASO(\tilde{U})$, then $\alpha' \in \mathcal{A}_O(F)$.

Proof : This is a mere observation which follows directly from the definition of feasibility. ■

For the purpose of finding functional dependencies in a p-p3mbuv it is important to know which vectors may become identical after an assignment of values to their asterisk components. If there is assignment which makes two vectors of input values identical then the corresponding functional value must also be identical otherwise there is no functional dependence. We define this notion as a *potential identity*.

Definition 2.9 Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables, $a, b \in \tilde{U}$ two vectors, and $j \in \{1, \ldots, n, n+1\}$ an index. We define the following notation

1. $a_j \approx b_j$ (an entry a_j is *potentially identical* to an entry b_j with respect to F) if there exists $\alpha \in \mathcal{A}(F)$, such that $a_j^\alpha = b_j^\alpha$.
2. $a \approx b$ (vectors a, b are *potentially identical* with respect to F) if there exists $\alpha \in \mathcal{A}(F)$, such that $a^\alpha = b^\alpha$.
3. $a_j \lessapprox b_j$ (an entry a_j is *potentially smaller* than an entry b_j with respect to F) if there exists $\alpha \in \mathcal{A}(F)$, such that $a_j^\alpha \leq b_j^\alpha$.
4. $a \lessapprox b$ (vector a is *potentially smaller* than vector b with respect to F) if there exists $\alpha \in \mathcal{A}(F)$, such that $a^\alpha \leq b^\alpha$, i.e. for every index $j \in \{1, \ldots, n, n+1\}$, $a_j^\alpha \leq b_j^\alpha$.
5. $a_j \approxeq b_j$ (an entry a_j is *identical* to an entry b_j with respect to F) if $a_j^\alpha = b_j^\alpha$ holds for every $\alpha \in \mathcal{A}(F)$.
6. $a \approxeq b$ (vectors a, b are *identical* with respect to F) if $a^\alpha = b^\alpha$ holds for every $\alpha \in \mathcal{A}(F)$.

Similarly, we say that a and b have *potentially identical inputs* if $I(a) \approx I(b)$, and they have *potentially identical outputs* if $F(a) \approx F(b)$. We say that *an input part of a is potentially smaller than an input part of b* if $I(a) \lessapprox I(b)$, and *an output value of a is potentially smaller than an output value of b*, if $F(a) \lessapprox F(b)$. Finally, a and b have *identical inputs* if $I(a) \approxeq I(b)$, and they have *identical outputs* if $F(a) \approxeq F(b)$.

The following properties are easy to observe:

Lemma 2.10 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables, $a, b \in \tilde{U}$ two vectors, and $j \in \{1, \ldots, n+1\}$ an index. The following statements hold:*

1. *$a_j \approx b_j$ if and only if one of the following properties is satisfied:*
   - *$a = \pi(b)$ and $a_j = b_j$, or*
   - *$a \neq \pi(b)$ and $\Delta(a,j) \cap \Delta(b,j) \neq \emptyset$.*
2. *$a \approx b$ if and only if $a_i \approx b_i$ holds for every index $i \in \{1,\dots,n+1\}$.*
3. *$a_j \lessapprox b_j$ if and only if one of the following properties is satisfied:*
   - *$a = \pi(b)$, $a_j, b_j \in \mathbb{Z}_3$, and $a_j \leq b_j$, or*
   - *$a = \pi(b)$, $a_j = *_-$ and $b_j = *_+$, or*
   - *$a \neq \pi(b)$ and $\min \Delta(a,j) \leq \max \Delta(b,j)$.*
4. *$a \lessapprox b$ if and only if $a_i \lessapprox b_i$ holds for every index $i \in \{1,\dots,n+1\}$.*
5. *$a_j \approxeq b_j$ if and only if one of the following properties is satisfied:*
   - *$a = \pi(b)$ and $a_j = b_j$, or*
   - *$a \neq \pi(b)$ and $(\exists e \in \mathbb{Z}_3)\ [\Delta(a,j) = \Delta(b,j) = \{e\}]$.*
6. *$a \approxeq b$ if and only if $a_i \approxeq b_i$ holds for every index $i \in \{1,\dots,n+1\}$.*

Proof : Follows directly from the above definitions. Note, that if $a = \pi(b)$ then the identity $a_j = b_j$ includes the case $a_j = b_j = *_=$ but excludes all cases when $a_j, b_j \in \{*_-, *_+\}$. ■

For the concept of robust extensions (defined at the end of this section), we shall need a weaker notion of potential identity and potential inequality, as well as a stronger notion of identity.

Definition 2.11 Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables, $a, b \in \tilde{U}$ two vectors, and $j \in \{1,\dots,n,n+1\}$ an index. We define the following notation

1. $a_j \sim b_j$ (an entry a_j is *robustly potentially identical* to an entry b_j with respect to F) if there exist $\alpha, \beta \in \mathcal{A}(F)$, such that $a_j^\alpha = b_j^\beta$.
2. $a \sim b$ (vectors a, b are *robustly potentially identical* with respect to F) if there exist $\alpha, \beta \in \mathcal{A}(F)$, such that $a^\alpha = b^\beta$.
3. $a_j \lesssim b_j$ (an entry a_j is *robustly potentially smaller* than an entry b_j with respect to F) if there exist $\alpha, \beta \in \mathcal{A}(F)$, such that $a_j^\alpha \leq b_j^\beta$.
4. $a \lesssim b$ (vector a is *robustly potentially smaller* than vector b with respect to F) if there exist $\alpha, \beta \in \mathcal{A}(F)$, such that $a^\alpha \leq b^\beta$, i.e. for every index $j \in \{1,\dots,n,n+1\}$, $a_j^\alpha \leq b_j^\beta$.
5. $a_j \simeq b_j$ (an entry a_j is *robustly identical* to an entry b_j with respect to F) if $a_j^\alpha = b_j^\beta$ holds for every $\alpha, \beta \in \mathcal{A}(F)$.

6. $a \simeq b$ (vectors a, b are *robustly identical* with respect to F) if $a^\alpha = b^\beta$ holds for every $\alpha, \beta \in \mathcal{A}(F)$.

Similarly, we say that a and b have *robustly potentially identical inputs* if $I(a) \sim I(b)$, and they have *robustly potentially identical outputs* if $F(a) \sim F(b)$. We say that *an input part of a is robustly potentially smaller than an input part of b* if $I(a) \lesssim I(b)$, and *an output value of a is robustly potentially smaller than an output value of b*, if $F(a) \lesssim F(b)$. Finally, a and b have *robustly identical inputs* if $I(a) \simeq I(b)$, and they have *robustly identical outputs* if $F(a) \simeq F(b)$.

The following properties are also easy to observe:

Lemma 2.12 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables, $a, b \in \tilde{U}$ two vectors, and $j \in \{1, \ldots, n+1\}$ an index. The following statements hold:*

1. *$a_j \sim b_j$ if and only if $\Delta(a, j) \cap \Delta(b, j) \neq \emptyset$.*
2. *$a \sim b$, if and only if $a_i \sim b_i$ holds for every index $i \in \{1, \ldots, n+1\}$.*
3. *$a_j \lesssim b_j$ if and only if $\min \Delta(a, j) \leq \max \Delta(b, j)$.*
4. *$a \lesssim b$, if and only if $a_i \lesssim b_i$ holds for every index $i \in \{1, \ldots, n+1\}$, .*
5. *$a_j \simeq b_j$, if and only if $(\exists e \in \mathbb{Z}_3)$ $[\Delta(a, j) = \Delta(b, j) = \{e\}]$.*
6. *$a \simeq b$, if and only if $a_i \simeq b_i$ holds for every index $i \in \{1, \ldots, n+1\}$.*

Proof : Follows directly from the above definitions. ■

If there are no missing bits (and uncertain function values) then the problem of finding an extension for a given pd3f $F = (O, L, H)$ in the class $\mathcal{C}_{3-all}$ (i.e. finding a fully defined 3-valued function f that agrees with F on all vectors in the sets (O, L, H)) is very easy. Clearly, analogically to the binary case of pdBf (see [7]), an extension of a pd3f $F = (O, L, H)$ in $\mathcal{C}_{3-all}$ exists, if and only if the sets O, L, H are pairwise disjoint. For the case of missing bits (and uncertain function values) there are several problems to consider (we are trying to extend the terminology from [7] to the 3-valued case here).

Definition 2.13 Given a p-p3mbuv $F = (\tilde{U}, \pi)$ on n variables and a class $\mathcal{C} \subseteq \mathcal{C}_{3-all}$ of 3-valued functions on n variables, we define problems CEP3($\mathcal{C}$) (consistent extension problem), FCP3($\mathcal{C}$) (fully consistent extension problem), and REP3($\mathcal{C}$) (robust extension problem) as follows:

CEP3($\mathcal{C}$) $[\exists \alpha \in \mathcal{A}(\tilde{U}, \pi)]$ $(\exists f \in \mathcal{C})$ $(\forall a \in \tilde{U})$ $[F(a^\alpha) = f(I(a^\alpha))]$?

FCP3($\mathcal{C}$) $[\forall \alpha \in \mathcal{A}(\tilde{U}, \pi)]$ $(\exists f \in \mathcal{C})$ $(\forall a \in \tilde{U})$ $[F(a^\alpha) = f(I(a^\alpha))]$?

REP3($\mathcal{C}$) $(\exists f \in \mathcal{C})$ $[\forall \alpha \in \mathcal{A}(\tilde{U}, \pi)]$ $(\forall a \in \tilde{U})$ $[F(a^\alpha) = f(I(a^\alpha))]$?

Looking at inputs and outputs separately, we consider the following variants FCOP3($\mathcal{C}$) (fully consistent output problem), and RCOP3($\mathcal{C}$) (robustly consistent output problem) as well:

FCOP3 $(\tilde{U}, \pi)$ has *fully consistent outputs* if for some $\alpha_O \in \mathcal{A}_O(\tilde{U}, \pi)$, $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension.

RCOP3 $(\tilde{U}, \pi)$ has *robustly consistent outputs* if for some $\alpha_O \in \mathcal{A}_O(\tilde{U}, \pi)$, $(\tilde{U}^{\alpha_O}, \pi)$ has a robust extension.

3 General functions

3.1 Problems CEP3, FCP3, and REP3

In this section we shall show, that problems FCP3 and REP3 are polynomially solvable for the class of all three valued functions, while problem CEP3 is NP-complete.

Theorem 3.1 *Problem CEP3($\mathcal{C}_{3-all}$) is NP-complete.*

Proof : Since the concept of a p-p3mbuv can be viewed as a generalization of concept of a pBmb, this result is a corollary of similar result in [7] (Theorem 9 therein). ∎

Now, we shall describe a polynomially checkable condition, under which a given p-p3mbuv has a fully consistent extension.

Theorem 3.2 *Problem FCP3($\mathcal{C}_{3-all}$) can be solved in polynomial time.*

Proof : The proof is similar to the case of pBmbs (see, Theorem 7 in [7]). We claim, that the following equivalence holds: A p-p3mbuv $F = (\tilde{U}, \pi)$ has a positive fully consistent extension if and only if

$$(\forall a, b \in \tilde{U})\ [I(a) \approx I(b) \Rightarrow F(a) \approxeq F(b)] \tag{1}$$

This condition can be clearly checked in polynomial time using Lemma 2.10. Since validity of this claim is not as obvious as in case of pBmbs, we shall prove it in more details.

Let us at first assume, that F has a fully consistent extension and let $a, b \in \tilde{U}$ be arbitrary such that $I(a) \approx I(b)$, therefore there is an assignment $\alpha_I \in \mathcal{A}_I(\tilde{U}, \pi)$, for which $I(a^{\alpha_I}) = I(b^{\alpha_I})$. Let us proceed by a contradiction and assume, that there is $\alpha_O \in \mathcal{A}_O(\tilde{U}, \pi)$ be such, that $F(a^{\alpha_O}) \neq F(b^{\alpha_O})$. Let $\alpha = \alpha_O \circ \alpha_I$. Since F has a fully consistent extension and since $\alpha \in \mathcal{A}(F)$ according to Lemma 2.8, there is a function $f : \mathbb{Z}_3 \mapsto \mathbb{Z}_3$, such that $F(a^{\alpha_O}) = F(a^{\alpha}) = f(I(a^{\alpha})) = f(I(a^{\alpha_I}) = f(I(b^{\alpha_I})) = f(I(b^{\alpha})) = F(b^{\alpha}) = F(b^{\alpha_0})$, which is a contradiction to our assumption that $F(a^{\alpha_O}) \neq F(b^{\alpha_O})$.

Now let us assume, that the condition (1) is satisfied and let $\alpha \in \mathcal{A}(F)$ be arbitrary, we shall find a function $f : \mathbb{Z}_3 \mapsto \mathbb{Z}_3$, which is consistent with F and α. Let

$$O = \{I(a^{\alpha}) \mid F(a^{\alpha}) = 0\}$$

$$\begin{aligned} L &= \{I(a^\alpha) \mid F(a^\alpha) = 1\} \\ H &= \{I(a^\alpha) \mid F(a^\alpha) = 2\} \end{aligned}$$

Our condition ensures, that O, L, H are pairwise disjoint. That means, that there exists an extension f of a pd3f (O, L, H). It can be observed, that f is a function consistent with F and α. ■

A similar condition can be found for robust extension, in fact, we only have to consider robust potential identity instead of a simple potential identity.

Theorem 3.3 *Problem REP3($\mathcal{C}_{3-all}$) can be solved in polynomial time.*

Proof : The proof is similar to the case of FCP3, and also to the case of pBmbs, as it was done in Theorem 7 in [7], we only have to use robust potential identity instead of potential identity, we also have to relax condition on difference of output values a bit. In particular, we claim, that the following equivalence holds. A p-p3mbuv $F = (\tilde{U}, \pi)$ has a robust extension if and only if

$$(\forall a, b \in \tilde{U})\ [I(a) \sim I(b) \Rightarrow F(a) \simeq F(b)]. \tag{2}$$

Let us at first assume, that F has a robust extension f and let $a, b \in \tilde{U}$ be arbitrary, $I(a) \sim I(b)$, therefore there exist $\alpha_I, \beta_I \in \mathcal{A}_I(F)$, for which $I(a^{\alpha_I}) = I(b^{\beta_I})$. Let us proceed by contradiction and let us assume, that there exist $\alpha_O, \beta_O \in \mathcal{A}_O(F)$, for which $F(a^{\alpha_O}) \neq F(b^{\beta_O})$. Let $\alpha = \alpha_I \circ \alpha_O, \beta = \beta_I \circ \beta_O$. Since f is a robust extension of F, we have that $f(I(a^\alpha)) = F(a^\alpha) \neq F(b^\beta) = f(I(b^\beta)$, which is a contradiction, since $I(a^\alpha) = I(b^\beta)$ implying $f(I(a^\alpha)) = f(I(b^\beta))$.

Now, let us suppose, that the condition (2) is satisfied. Let

$$\begin{aligned} O^\alpha &= \{I(a^\alpha) \mid F(a^\alpha) = 0\} \\ L^\alpha &= \{I(a^\alpha) \mid F(a^\alpha) = 1\} \\ H^\alpha &= \{I(a^\alpha) \mid F(a^\alpha) = 2\} \\ O &= \bigcup_{\alpha \in \mathcal{A}(F)} O^\alpha \\ L &= \bigcup_{\alpha \in \mathcal{A}(F)} L^\alpha \\ H &= \bigcup_{\alpha \in \mathcal{A}(F)} H^\alpha \end{aligned}$$

We can observe, that O, L, H are pairwise disjoint sets. This is ensured by condition (2), since if for some α, β and some $a, b \in \tilde{U}$ would be $a^\alpha = b^\beta$, then $I(a) \sim I(b)$ and according to condition (2) is $F(a) \simeq F(b)$ and hence $F(a^\alpha) = F(b^\beta)$. Therefore a^α, b^β fall into the same set within O, L, H. Thus a pd3f (O, L, H) has an extension f, which is also a robust extension of F. ■

3.2 Problems FCOP3 and RCOP3

In this subsection, we shall present an algorithm, which for a given p-p3mbuv $F = (\tilde{U}, \pi)$ enumerates with polynomial delay all feasible assignments to outputs $\alpha_O \in \mathcal{A}_O(F)$, for which there is a fully consistent extension of $(\tilde{U}^{\alpha_O}, \pi)$. Consequence of correctness of this algorithm will be, that problem FCOP3($\mathcal{C}_{3-all}$) is solvable in polynomial time.

Algorithm 3.4 FCOP3($\mathcal{C}_{3-all}$)

Input: *A p-p3mbuv $F = (\tilde{U}, \pi)$.*
Output: *All assignments $\alpha_O \in \mathcal{A}_O(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension.*

```
1:  Construct undirected graph $G = (\tilde{U}, E)$, where
        $$E = \{\{u, v\} \mid u, v \in \tilde{U} \wedge [I(u) \approx I(v) \vee [u = \pi(v) \wedge F(u) = F(v) = *_=]]\}.$$
2:  Find all connected components $C_1, \ldots, C_k$ of graph $G$.
3:  for $i$:=1 to $k$
4:  do
5:    $S_i := \bigcap_{u \in C_i} \Delta(u, n+1)$
6:    if $S_i = \emptyset$ or $(\exists u, v = \pi(u) \in C_i)$ $[F(u) = *_- \wedge F(v) = *_+]$
7:    then
8:      return empty set (no assignment)
9:    endif
10: enddo
11: Construct directed graph $D = (V, A)$, where $V = \{C_1, \ldots, C_k\}$ and
        $$A = \{(C_i, C_j) \mid (\exists u \in C_i)(\exists v \in C_j)\ [u = \pi(v) \wedge F(u) = *_- \wedge F(v) = *_+]\}.$$
12: Find all connected components $B_1, \ldots, B_l$ of underlying undirected graph of $D$.
13: for $j$:=1 to $l$
14: do
15:   Let $B_j = \{C_{j_1}, \ldots, C_{j_k}\}$.
16:   Let (w.l.o.g.) $C_{j_1}$ be with minimum $|S_{j_1}|$ among $C_{j_1}, \ldots, C_{j_k}$.
17:   $\mathcal{B}_j := \emptyset$ {$\mathcal{B}_j$ is a set of assignments.}
18:   for each $e \in S_{j_1}$
19:   do
20:     $c_j[j_1] := e$ {$c_j[i]$ will denote value assigned to all vectors belonging to $C_i$}
21:     Let $T$ be a stack containing $j_1$.
22:     while $Z \neq \emptyset$
23:     do
24:       $i$:=pop($T$)
25:       for each $(C_i, C_{i'}) \in A$
26:       do
27:         if $(c[i] = 2)$ or $(c[i']$ is defined and $c[i'] \neq c[i] + 1)$ or $(c[i] + 1 \notin S_{i'})$
```

```
28:             then
29:               break
30:             endif
31:             if $c[i']$ is undefined
32:             then
33:               $c[i'] := c[i] + 1$
34:               push $(T,\, i')$
35:             endif
36:           enddo
37:           for each $(C_{i'}, C_i) \in A$
38:           do
39:             if $(c[i] = 0)$ or $(c[i']$ is defined and $c[i'] \neq c[i] - 1)$ or $(c[i] - 1 \notin S_{i'})$
40:             then
41:               break
42:             endif
43:             if $c[i']$ is undefined
44:             then
45:               $c[i'] := c[i] - 1$
46:               push $(T,\, i')$
47:             endif
48:           enddo
49:         enddo
50:         Let $\alpha_O^{e,i}$ be assignment, which for all $i \in \{j_1, \ldots, j_k\}$ assigns $\alpha_O^{e,i}(u, n+1) = c[i]$
            for every $u \in C_i$.
51:         $\mathcal{B}_j := \mathcal{B}_j \cup \{\alpha_O^{e,i}\}$
52:       enddo
53:       if $\mathcal{B}_j = \emptyset$ then return empty set endif
54:     enddo
55:     for each $(\alpha_O^1, \ldots, \alpha_O^l) \in \prod_{j=1}^{l} \mathcal{B}_j$
56:     do
57:       output $\alpha_O^1 \circ \ldots \circ \alpha_O^l$
58:     enddo
```

Proof of correctness of Algorithm 3.4 is split into several lemmas. We shall start with showing that returning an empty set on line 8 is correct.

Lemma 3.5 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables. If Algorithm 3.4 returns empty set on line 8, then there is no $\alpha_O \in \mathcal{A}_O(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension.*

Proof : We shall at first show, that if $\alpha_O \in \mathcal{A}_O(F)$ is such, that $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension, and if C is an arbitrary connected component of G, then for every $u, v \in C$, $\alpha_O(u, n+1) = \alpha_O(v, n+1)$ (i.e. α_O is constant on C). This can be observed using a simple induction on the length of path from u to v in G. If $\{u, v\} \in E$, then either $F(u) = F(v) = *_=$, in which case obviously $\alpha_O(u, n+1) = \alpha_O(v, n+1)$ for any $\alpha_O \in \mathcal{A}_O(F)$, or $I(u) \approx I(v)$, in which case $\alpha_O(u, n+1) = \alpha_O(v, n+1)$ using condition 1 from the proof of Theorem 3.2 used on p-p3mbuv $(\tilde{U}^{\alpha_O}, \pi)$. In this case $I(u) \approx I(v)$ implies, that $F(a^{\alpha_O})$ is not potentially different from $F(b^{\alpha^O})$, which means, that they are same. Induction step is obvious.

It is easy to see, that set S_i contains exactly values, which can be assigned simultaneously outputs of all vectors in connected compontent C_i. If $S_i = \emptyset$ or there is a pair $u, v = \pi(v) \in C$, for which $F(u) = *_-$ and $F(v) = *_+$, than clearly no value can be simultaneously to assigned to outputs of vectors in connected components and that means, that no assignment $\alpha_O \in \mathcal{A}_O(F)$ for which $(\tilde{U}^{\alpha_O}, \pi)$ would have fully consistent extension, may exist. ∎

We continue with proving that line 53 also correctly returns an empty set.

Lemma 3.6 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables. If Algorithm 3.4 returns empty set on line 53, then there is no $\alpha_O \in \mathcal{A}_O(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension.*

Proof : Let $B_j = (C_{j_1}, \ldots, C_{j_k})$ be an arbitrary component of underlying undirected graph of D. Let $Q = \{(u, n+1) \mid (u \in \cup B_j) \wedge (u, n+1) \in \{*_=, *_-, *_+\}\}$. We shall show, that $\alpha : Q \mapsto \mathbb{Z}_3 \in \mathcal{B}_j$ if and only if α is feasible assignment. The proposition follows, since if no feasible assignment $\alpha : Q \mapsto \mathbb{Z}_3$ exists, then we correctly return empty set in step 53.

We claim, that if we choose an arbitrary element $C \in B_j$ and we set the output values of vectors in C, then the output values of remaining vectors in $\cup B_j$ are uniquely defined. In particular, let $C_i, C_{i'} \in B_j$ be two components, we shall show, that if we set output values of vectors in C_i, then the output values of vectors in $C_{i'}$ are uniquely defined. This follows by a simple induction from fact, that if $(C_i, C_{i'}) \in A$ or $(C_{i'}, C_i) \in A$, and we have assigned the value $e \in \mathbb{Z}_3$, then the value, which we have to assign to output values of vectors in $C_{i'}$ is either $e+1$ or $e-1$, if it belongs to $S_{i'}$, otherwise, no value can be assigned to output values of vectors in $C_{i'}$. This fact follows directly from the definition of graph D.

In *while* cycle on lines 22–49 we traverse graph D in depth first search manner and we assign the values in unique way, starting from C_{j_1}, and if we successfully traverse entire graph, we add assignment which we have found to $\mathcal{B}_j$.

From this it follows, that when $\alpha \in \mathcal{B}_j$, then it is feasible assignment to Q. If on the other hand α is a feasible assignment to Q, then it assigns some value to output values of vectors in C_{j_1}, and this value must be same for all vectors using arguments from proof of Lemma 3.5, moreover this value e belongs to S_{j_1}. Therefore our algorithm will consider this α and adds it into $\mathcal{B}_j$. ∎

It remains to show, that Algorithm 3.4 returns exactly those assignments $\alpha_O \in \mathcal{A}_O(\tilde{U}, \pi)$, for which a p-p3mbuv $(\tilde{U}, \pi)$ has a fully consistent extension. Output of these assignments is done on line 57, correctness of which is justified by the following lemma.

Lemma 3.7 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables. Let α_O be an arbitrary assignment, then α_O is output by Algorithm 3.4 on line 57, if and only if $\alpha_O \in \mathcal{A}_O(F)$ and $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension.*

Proof : Let us at first assume, that α_O is output by Algorithm 3.4. Let us assume, that $\alpha_O = \alpha_O^1 \circ \dots \alpha_O^l$, where $(\alpha_O^1, \dots, \alpha_O^l) \in \prod_{j=1}^{l} \mathcal{B}_j$ and let us start by observing, that $\alpha_O \in \mathcal{A}_O(F)$. This is because if $u, v = \pi(v)$ is a pair of vectors with $F(u) = F(v) = *_=$, then $\{u, v\} \in E$ and therefore u and v belong to the same component C of graph G. And our algorithm in step 50 ensures, that in α_O are outputs of both u and v assigned the same value. If $F(u) = *_-$ and $F(u) = *_+$, then u and v belong to different components of$G$. Let us denote them by C and C' respectively. In graph D, then $(C, C') \in A$. In which case during traversing connected component B of D, to which both C and C' belong, we assign the right values either in step 33, if we have come to C' from C, or in step 45, if we have come to C from C'. If at the time, we were examining neighbours of C or C', value of C' or C has been already decided, we ensure by test at line 27 or line 39, that the same value would be assigned this time. Hence, by Definition 2.7, α_O is a feasible assignment, which assigns values to all outputs and therefore $\alpha_O \in \mathcal{A}_O(F)$.

Let us continue by showing, that $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension. By the proof of Theorem 3.2, we know, that this is equivalent to the fact, that for every pair $u, v \in \tilde{U}$, such that $I(u) \approx I(v)$ is $F(u^{\alpha_O}) = F(u^{\alpha_O})$, which is clearly satisfied, since both u and v belong to the same connected component of G.

Now let us assume, that $\alpha_O \in \mathcal{A}_O(F)$ and that $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension. For a connected component B_j of underlying undirected graph of D, let us denote $Q_j = \{(u, n+1) \mid u \in B_j \wedge (u, n+1) \in \{*_-, *_+, *_=\}\}$ and let $\alpha_O^j = \alpha_O \restriction_{Q_j}$. We shall show that for an arbitrary j is $\alpha_O^j \in \mathcal{B}_j$ and therefore $(\alpha_O^1, \dots, \alpha_O^l) \in \prod_{j=1}^{l} \mathcal{B}_j$ and $\alpha_O = \alpha_O^1 \circ \dots \alpha_O^l$ is output at line 57. Let us recall the proof of Lemma 3.6, where we have shown, that an assignment $\alpha : Q_j \mapsto \mathbb{Z}_3 \in \mathcal{B}_j$ if and only if α is a feasible assignment. Since according to Lemma 2.8 α_O^j is a feasible assignment, it belongs to $\mathcal{B}_j$ and we are done. ■

Now, we have everything ready to prove correctness of Algorithm 3.4.

Theorem 3.8 *Given p-p3mbuv $F = (\tilde{U}, \pi)$ on n variables, Algorithm 3.4 outputs all assignments $\alpha_O \in \mathcal{A}_0(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a fully consistent extension. If no such assignment exists, Algorithm 3.4 returns empty set.*

Proof : The proof directly follows from lemmas 3.5, 3.6, and 3.7 ■

It remains to estimate time requirements of Algorithm 3.4.

Lemma 3.9 *Given p-p3mbuv $F = (\tilde{U}, \pi)$ on n variables, Algorithm 3.4 finishes in time $O(m \cdot m + |OUTPUT|)$, where $m = |\tilde{U}|$ and $|OUTPUT|$ denotes the size of output.*

Proof : Time requirements are quite straightforward. Construction of graph G takes $O(m \cdot n)$ time, because we need to read entire input. Finding connected components of G

takes $O(m \cdot m)$, since the number of edges of G is at most $m \cdot m$. Cycle at steps 3–10 takes $O(\sum_{i=1}^{k} |C_i|) = O(m)$ time. Construction of graph D can be done in linear time, i.e. in $O(m)$, since each vector from $\tilde{U}$ may contribute by at most one arc to A. Hence also finding connected components of its underlying undirected graph takes $O(m)$ time. Every run of a cycle at lines 13–52 takes $O(|B_j|)$, since depth first search of a graph takes linear time, therefore in the whole, cycle from 13–52 takes $O(m)$ time. Cycle, in which we ouptput the result, takes time $O(|OUTPUT|)$. After summing it up, we get desired $O(m \cdot m + |OUTPUT|)$ time requirements. The fact, that we get output with polynomial delay is pretty obvious. ■

Algorithm 3.4 can be easily modified in order to solve problem RCOP3($\mathcal{C}_{all}$).

Theorem 3.10 *Given p-p3mbuv $F = (\tilde{U}, \pi)$ on n variables, Algorithm 3.4 can be modified, so that it outputs all assignments $\alpha_O \in \mathcal{A}_0(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a robust extension. If no such assignment exists, this modified Algorithm 3.4 returns empty set. Time requirements of modified algorithm are the same as requirements of Algorithm 3.4*

Proof : Only definition of set of edges of graph G in step 1 needs to be modified. In modified version, we set:

$$E = \{\{u, v\} \mid u, v \in \tilde{U} \wedge [I(u) \sim I(v) \vee [u = \pi(v) \wedge F(u) = F(v) = *_=]]\}.$$

I.e. instead of potential identity we use robustly potential identity. It is not hard to check, that after this modification, Algorithm 3.4 solves problem RCOP3($\mathcal{C}_{all}$) in the same time as it did for FCOP3($\mathcal{C}_{all}$). We leave the details to the reader. ■

4 Positive functions

In this section we shall concentrate on positive functions. Throughout the section, we shall need the following condition which specifies, when a pd3f has a positive extension.

Lemma 4.1 *A positive extension of a pd3f $F = (O, L, H)$ exists, if and only if the following property is satisfied:*

$$(\forall a, b \in O \cup L \cup H)\ [(a \leq b) \Rightarrow (F(a) \leq F(b))]$$

Proof : The property is a simple extension of binary case (see [6]), and we therefore leave the details to the reader. ■

A simple corollary, which we shall use throughout the section is following. Assume that we have a p-p3mbuv $F = (\tilde{U}, \pi)$ and a feasible assignment $\alpha \in \mathcal{A}(F)$, which assigns a value to every asterisk position in F. Then we can naturally associate a pd3f with F^α, which has an extension, if and only if the following property is satisfied:

$$(\forall a, b \in \tilde{U})\ [(I(a^\alpha) \leq I(b^\alpha)) \Rightarrow (F(a^\alpha) \leq F(b^\alpha))]$$

We shall start with problem CEP3, which is NP-complete, unlike its binary pBmb version. Then we consider FCP3 and REP3 and finally we shall solve problems FCOP3 and RCOP3, all these problems are polynomially solvable.

4.1 Problem CEP3

Problem CEP3($\mathcal{C}_3^+$) seems to be harder to solve - we show in the next theorem that it is probably intractable in the general case.

Theorem 4.2 *Problem CEP3($\mathcal{C}_3^+$) is NP-complete.*

The problem is obviously in NP since when we get $\alpha \in \mathcal{A}(F)$ for p-p3mbuv $F = (\tilde{U}, \pi)$ we can verify (using Lemma 4.1) in polynomial time that pd3f function $(\tilde{O}^\alpha, \tilde{L}^\alpha, \tilde{H}^\alpha)$ has a positive extension.

To show that the problem is NP-hard we use the problem SAT restricted to instances where the input CNF consists only of positive and negative clauses. We shall denote this problem SAT$_{\text{PN}}$. It is NP-hard because a general CNF $\mathcal{F}$ on variables $x_1, \ldots, x_n$ can be transformed into a CNF $\mathcal{F}$ which has only positive and negative clauses and which is satisfiable iff $\mathcal{F}$ is satisfiable. The transformation introduces new variables $y_1, \ldots, y_n$ such that $x_i \equiv \overline{y}_i$ for every $i \in \{1, \ldots, n\}$. This can be achieved by adding the following two clauses (one positive and one negative) for every i:

$$(x_i \vee y_i) \wedge (\overline{x}_i \vee \overline{y}_i) \ ,$$

which ensure that variables x_i and y_i have complementary values in every evaluation satisfying $\mathcal{F}$. Now for each clause $C \in \mathcal{F}$ we can substitute suitable subset of its literals by complementary literals formed by corresponding varibles y_i, e.g., we can substite all negative literals $\overline{x}_i$ by positive literals y_i to form a positive clause C'. Then $\mathcal{F}$ will consist of all these C' together with the pairs of clauses ensuring $x_i \equiv \overline{y}_i$ and hence $\mathcal{F}$ will contain only positive and negative clauses.

Now we shall take CNF $\mathcal{F}$ on n variables which forms an instance of SAT$_{\text{PN}}$ and transform it to an instance of CEP3($\mathcal{C}_3^+$). We will denote by $P(\mathcal{F})$ the set of positive clauses of $\mathcal{F}$ and by $N(\mathcal{F})$ the set of negative clauses of $\mathcal{F}$. By $\chi_{0,2}(C)$, where C is a clause of $\mathcal{F}$, we denote the 0,2 characteristic vector of C, i.e., a vector v of length n containing only 0 and 2 and such that

$$v_i = \begin{cases} 2 & \text{if } x_i \text{ forms a literal in } C \\ 0 & \text{if } x_i \text{ does not form a literal in } C \end{cases}$$

We form p-p3mbuv $F = (\tilde{U}, \pi)$ on $n+1$ variables, i.e. with vectors of length $n+2$, as follows:

$$\begin{aligned} \tilde{O} &= \{(\overline{\chi_{0,2}(C)}, 0, 0) \mid C \in P(\mathcal{F})\} \\ \tilde{L} &= \{k = (*_-, \ldots, *_-, 0, 1), l = (*_+, \ldots, *_+, 2, 1)\} \\ \tilde{H} &= \{(\chi_{0,2}(C), 2, 2) \mid C \in N(\mathcal{F})\} \\ \tilde{U} &= \tilde{O} \cup \tilde{L} \cup \tilde{H} \ . \end{aligned}$$

The pairing function π just associates the two vectors in $\tilde{L}$ and it can be arbitrary for all other vectors (as there are no uncertain values in them).

We will finish the proof by showing that the p-p3mbuv $(\tilde{U}, \pi)$ has a positive consistent extension iff $\mathcal{F}$ is satisfiable. According to Lemma 4.1 and definition of consistent extension p-p3mbuv $(\tilde{U}, \pi)$ has a positive consistent extension iff there is $\alpha \in \mathcal{A}(F)$ such that

$$(\forall a, b \in \tilde{U})\ [(I(a^\alpha) \leq I(b^\alpha)) \Rightarrow (F(a^\alpha) \leq F(b\alpha))]\ . \tag{3}$$

Condition (3) is equivalent to the following one:

$$(\forall a, b \in \tilde{U})\ [(F(a^\alpha) > F(b^\alpha)) \Rightarrow (\exists i \in \{1, \ldots, n+1\})\ (a_i^\alpha > b_i^\alpha)]\ . \tag{4}$$

Because of the artificial $(n+1)$-st component of the vectors of F we immediately know that condition (4) is satisified in all of the following cases:

- $a \in \tilde{H}$, $b \in \tilde{O}$,
- $a \in \tilde{H}$, $b = k$,
- $a = l$, $b \in \tilde{O}$.

Therefore there is a positive consistent extension of F iff there is $\alpha \in \mathcal{A}(F)$ such that the following two conditions are satisfied:

$$(\forall a \in \tilde{H})\ (\exists i \in \{1, \ldots, n\})\ (l_i^\alpha < a_i) \tag{5}$$

$$(\forall b \in \tilde{O})\ (\exists j \in \{1, \ldots, n\})\ (k_j^\alpha > b_j) \tag{6}$$

Let us fix $\alpha \in \mathcal{A}(F)$. Since $\pi(k) = l$ we have for every $i \in \{1, \ldots, n\}$ that $k_i^\alpha = l_i^\alpha - 1$. Hence if some i satisfies condition (5) for some $a \in \tilde{H}$ (i.e., $a_i = 2$ and $l_i^\alpha = 1$) then this i cannot satisfy condition (6) for any $b \in \tilde{O}$ because k_i^α must be 0. Also, vice versa, i satisfying condition (6) for some $b \in \tilde{O}$ cannot satisfy condition (5) for any $a \in \tilde{H}$. This altogether gives us that every i can satisfy either condition (5) for all $a \in \tilde{H}$ such that $a_i = 2$ or condition (6) for all $b \in \tilde{O}$ such that $b_i = 2$.

This is equivalent to the fact that every variable x_i in CNF $\mathcal{F}$ can either satisfy all negative clauses containing literal $\overline{x}_i$ (by evaluating x_i to 0) or it can satisfy all positive clauses containing literal x_i (by evaluating x_i to 1). Hence satisfying conditions (5) and (6) for all $a \in \tilde{H}$ and $b \in \tilde{O}$ is equivalent to satisfying all clauses of CNF $\mathcal{F}$. This completes the proof of NP-hardness of CEP3(C_3^+). ■

4.2 Problems FCP3 and REP3

In this section we shall show, that problems FCP3 and REP3 are polynomially solvable for the class of positive three valued functions. We start with problem FCP3.

Theorem 4.3 *Problem FCP3($\mathcal{C}_3^+$) can be solved in polynomial time.*

Proof : We claim, that a p-p3mbuv $F = (\tilde{U}, \pi)$ has a fully consistent extension if and only if the following condition is satisfied:

$$(\forall a, b \in \tilde{U})\ (\forall \alpha_O \in \mathcal{A}_O(F))\ [I(a) \lessapprox I(b) \Rightarrow F(a^{\alpha_O}) \leq F(b^{\alpha_O})]. \tag{7}$$

This condition can easily be checked.

Let us at first suppose, that F has a fully consistent positive extension and let $a, b \in \tilde{U}$ be arbitrary, such that $I(a) \lessapprox I(b)$, let $\alpha_I \in \mathcal{A}_I(F)$ be such, that $I(a^{\alpha_I}) \leq I(b^{\alpha_I})$. Let $\alpha_O \in \mathcal{A}_O(F)$ be arbitrary and let $\alpha = \alpha_I \circ \alpha_O$, according to Lemma 2.8 is $\alpha \in \mathcal{A}(F)$ and $I(a^\alpha) \leq I(b^\alpha)$ and therefore there is a positive function $f_\alpha : \mathbb{Z}_3^n \mapsto \mathbb{Z}_3$ which is a consistent extension of $\tilde{U}^\alpha$, i.e. $F(a^{\alpha_O}) = F(a^\alpha) = f_\alpha(a^\alpha) \leq f_\alpha(b^\alpha) = F(b^\alpha = F(b^{\alpha_O})$.

Now let us assume, that condition (7) is satisfied and let $\alpha \in \mathcal{A}(F)$ be arbitrary, we shall find a positive function $f : \mathbb{Z}_3^n \mapsto \mathbb{Z}_3$, which is consistent with F and α. Let

$$\begin{aligned} O &= \{I(a^\alpha) \mid F(a^\alpha) = 0\} \\ L &= \{I(a^\alpha) \mid F(a^\alpha) = 1\} \\ H &= \{I(a^\alpha) \mid F(a^\alpha) = 2\} \end{aligned}$$

Since according to Lemma 2.8 is $\alpha_O = \alpha \restriction_{ASO(F)} \in \mathcal{A}_O(F)$, condition (7) implies for given α the following:

$$(\forall a, b)\ [I(a^\alpha) \leq I(b^\alpha) \Rightarrow F(a^\alpha) \leq F(b^\alpha)]$$

This is exactly condition, which according to Lemma 4.1 implies existence of positive extension f_α of a pd3f (O, L, H), which is clearly consistent with F and α, as well. ■

As in general case, problem REP3 can be solved in a very similar way as problem FCP3.

Theorem 4.4 *Problem REP3($\mathcal{C}_3^+$) can be solved in polynomial time.*

Proof : The proof is similar to the case of FCP3, and also to the case of pBmbs, as it was done in Theorem 7 in [7], we only have to use robust potential inequality instead of potential inequality. We claim, that a p-p3mbuv $F = (\tilde{U}, \pi)$ has a positive robust extension if and only if the following condition is satisfied:

$$(\forall a, b \in \tilde{U})\ (\forall \alpha_O, \beta_O \in \mathcal{A}_O(F))\ [I(a) \lesssim I(b) \Rightarrow F(a^{\alpha_O}) \leq F(b^{\beta_O})]. \tag{8}$$

This condition can easily be checked.

Let us at first suppose, that F has a robust positive extension f and let $a, b \in \tilde{U}$ be arbitrary, such that $I(a) \lessapprox I(b)$, let $\alpha_I, \beta_I \in \mathcal{A}_I(F)$ be such, that $I(a^{\alpha_I}) \leq I(b^{\beta_I})$. Let $\alpha_O, \beta_O \in \mathcal{A}_O(F)$ be arbitrary and let $\alpha = \alpha_I \circ \alpha_O$, $\beta = \beta_I \circ \beta_O$, according to Lemma 2.8 is $\alpha, \beta \in \mathcal{A}(F)$ and $I(a^\alpha) \leq I(b^\beta)$. Since f is arobust extension of F, $F(a^{\alpha_O}) = F(a^\alpha) = f(a^\alpha) \leq f(b^\beta) = F(b^\alpha = F(b^{\alpha_O})$.

Now let us assume, that condition (8) is satisfied and let $\alpha \in \mathcal{A}(F)$ be arbitrary, we shall find a positive function $f : \mathbb{Z}_3^n \mapsto \mathbb{Z}_3$, which is robustly consistent with F. Let

$$O^\alpha = \{I(a^\alpha) \mid F(a^\alpha) = 0\}$$

$$\begin{aligned}
L^\alpha &= \{I(a^\alpha) \mid F(a^\alpha) = 1\}\\
H^\alpha &= \{I(a^\alpha) \mid F(a^\alpha) = 2\}\\
O &= \bigcup_{\alpha\in\mathcal{A}(F)} O^\alpha\\
L &= \bigcup_{\alpha\in\mathcal{A}(F)} L^\alpha\\
H &= \bigcup_{\alpha\in\mathcal{A}(F)} H^\alpha
\end{aligned}$$

Let $F' = (O, L, H)$ be corresponding pd3f. Since according to Lemma 2.8 is $\alpha_O = \alpha \restriction_{ASO(F)}$ $, \beta_O = \beta \restriction_{ASO(F)} \in \mathcal{A}_O(F)$, condition (7) implies for every $\alpha, \beta \in \mathcal{A}(F)$ the following:

$$(\forall a, b)\ [I(a^\alpha) \leq I(b^\beta) \Rightarrow F(a^\alpha) \leq F(b^\beta)]$$

Therefore given $a, b \in O \cup L \cup H$ such that $a \leq b$, $a = u^\alpha$ for some $u \in \tilde{U}$ and $\alpha \in \mathcal{A}(F)$ and $b = v^\beta$ for some $v \in \tilde{U}$ and $\beta \in \mathcal{A}(F)$. Hence $F'(a) = F(u^\alpha) \leq F(v^\beta) = F'(b)$. According to Lemma 4.1, F' has a positive extension f, which is clearly also a robust extension of F. ■

4.3 Problems FCOP3 and RCOP3

In this subsection we consider problems FCOP3 and RCOP3 for the class of positive three valued functions. As in case of general functions, we first solve problem FCOP3, solution of RCOP3 can be then found in the same way with considering robust potential inequality instead of simple potential inequality. We shall at first show, how to decide, whether problem FCOP3 has a solution for a given p-p3mbuv and if so, how to find one. Then we shall concentrate on output of all solutions of FCOP3 for positive three valued functions, i.e. all assignments to outputs for which given p-p3mbuv has a fully consistent extension, with polynomial delay.

The idea behind our algorithm is similar to the one behind Algorithm 3.4. We first associate a graph with given p-p3mbuv, which now will be directed, then we consider an acyclic condensation of this graph, i.e. graph, in which each strongly connected component becomes a single vertex. Then we shall reformulate problem FCOP3 as a problem of assigning values 0, 1, and 2 to vertices in this graph, which we then finally solve. We start with associating a graph with each p-p3mbuv.

Lemma 4.5 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables. Let $G = (\tilde{U}, E)$ be a directed graph with $\tilde{U}$ as the set of vertices and the set of arcs E defined as*

$$\begin{aligned}
E &= E_1 \cup E_2 \cup E_3, \text{ where}\\
E_1 &= \{(u, v) \mid u, v \in \tilde{U} \wedge I(u) \lessapprox I(v)\}\\
E_2 &= \{(u, v) \mid u, v \in \tilde{U} \wedge u = \pi(v) \wedge F(u) = *_- \wedge F(v) = *_=\}\\
E_3 &= \{(u, v) \mid u, v \in \tilde{U} \wedge u = \pi(v) \wedge F(u) = F(v) = *_=\}
\end{aligned}$$

Let C be an arbitrary strongly connected component of G and let $\alpha_O \in \mathcal{A}_O(F)$ be an arbitrary assignment. If $(\tilde{U}^{\alpha_O}, \pi)$ has a positive fully consistent extension, then the following hold:

1. *For every $a, b \in \tilde{U}$, such that there is a directed path from a to b in G is $F(a^{\alpha_O}) \leq F(b^{\alpha_O})$ and if there is a directed path from a to b, which contains an arc from E_2, then $F(a^{\alpha_O}) < F(b^{\alpha_O})$.*
2. *For every $a, b \in C$, $F(a^{\alpha_O}) = F(b^{\alpha_O})$, and*
3. *there is no arc in E_3 with both endpoints in C.*

Proof : Let $a, b \in \tilde{U}$ be arbitrary, such that there is a directed path from a to b, we shall observe, that in this case $F(a^{\alpha_O}) \leq F(b^{\alpha_O})$ and if this path contains an arc from E_2, then this inequality is strict. We can proceed by a simple induction on the length of the path from a to b. If $(a, b) \in E$, then the proposition clearly holds, in case of $(a, b) \in E_1$ according to Theorem 4.3 and in case of $(a, b) \in E_2 \cup E_3$ according to definition. General step follow by the same arguments and transitivity of inequality. By this we have shown the first proposition.

The second proposition follows from the first one and the fact, that in case of $a, b \in C$ there are directed path from a to b and from b to a. The third proposition follows easily from the second one. ■

Now we shall define a graph problem, the solution of which will serve as the basic subroutine in all subsequent algorithms of this section.

Definition 4.6 Let $D = (V, A)$ be an acyclic directed graph where A is a disjoint union of A_1 and A_2.

- The *acyclic three valued inequality solving problem (AIS3)* on D is the following: Assign a value $c(v) \in \{0, 1, 2\}$ to every vertex of V such that for every $(u, v) \in A_1$ is $c(u) \leq c(v)$ and for every $(u, v) \in A_2$ is $c(v) = c(u) + 1$.
- The *restricted acyclic three valued inequality solving problem (AIS3R)* on D furthermore requires that for each $v \in V$ the assigned value $c(v)$ belongs to a preselected set $S(v) \subseteq \{0, 1, 2\}$, where $S(v)$ can be any subset of $\{0, 1, 2\}$ except $\{0, 2\}$.

We shall at first observe, that the problem AIS3R is equivalent to the problem we are solving, then we shall show, how to transform AIS3R to AIS3 and how to solve it in polynomial time. Then we shall describe all possible assignments solving problems AIS3 and AIS3R respectively. We shall use this description to find all these assignments with polynomial delay.

Lemma 4.7 *Let $F = (\tilde{U}, \pi)$ be a p-p3mbuv on n variables and let $G = (\tilde{U}, E)$ be a directed graph defined as in Lemma 4.5. Let graph $D = (V, A)$ be a directed graph, which* *is an acyclic condensation of G. I.e. $V = \{C_1, \ldots, C_k\}$, where $C_1, \ldots, C_k$ are all strongly connected components of G and $A = A_1 \cup A_3$, where*

$$A_1 = \{(C_i, C_j) \mid (\exists u \in C_i)(\exists v \in C_j)\ (u, v) \in E_1\}$$
$$A_2 = \{(C_i, C_j) \mid (\exists u \in C_i)(\exists v \in C_j)\ (u, v) \in E_3\}$$

If G satisfies the properties of Lemma 4.5, then there exists an assignment $\alpha_O \in \mathcal{A}_O(F)$, for which $(\tilde{U}^{\alpha_O}, \pi)$ has a positive fully consistent extension if and only if problem AIS3R has solution for graph D. Moreover, there is one to one correspondence between feasible assignments $\alpha_O \in \mathcal{A}_O(F)$ for which $(\tilde{U}^{\alpha_O}, \pi)$ has a positive fully consistent extension and solutions to the problem AIS3R for graph D.

Proof : This propostion is a mere observation, since problem AIS3R is a simple reformulation of the problem of finding the assignment to outputs, which would respect arcs of G. Note, that all vectors in a strongly connected component C of G have to receive same output value according to Lemma 4.5 and that their output values have to belong to $S(C) = \bigcap_{u \in C} \Delta(u, n+1)$. These sets put restriction on the values of vertices of D in problem AIS3R. According to this, it is not possible, that $S(C) = \{0, 2\}$, which we forbid in AIS3R. ■

Now we shall show, how to transform problem AIS3R to problem AIS3.

Lemma 4.8 *Let $D = (V, A = A_1 \cup A_2)$ be an acyclic directed graph with $A_1 \cap A_2 = \emptyset$ and for every $v \in V$ let $S(v) \subseteq \{0, 1, 2\}$ be a restriction put on v such that $S(v) \neq \{0, 2\}$. Then there exists $D' = (V', A' = A'_1 \cup A'_2)$ an instance of AIS3 problem, such that AIS3 has solution for D' if and only if AIS3R has solution for D and restrictions S and there is one to one to one correspondence between the solutions of AIS3 for D' and solutions of AIS3R for D with restrictions S.*

Proof : We shall proceed by induction on the number of vertices $v \in V$, for which $S(v) \neq \{0, 1, 2\}$. If there is no such vertex, then we may clearly set $D' = D$. Now, let $v \in V$ be an arbitrary vertex with $S(v) \neq \{0, 1, 2\}$. We shall at first construct a directed graph D'' and restrictions S'', such that $S''(v) = \{0, 1, 1\}$ and $S''(v') = S(v')$ for $v' \in V \setminus \{v\}$. At the begining we set $D'' = (V'', A''_1 \cup A''_2) = D$, with $V'' = V$, $A''_1 = A_2$ and $A''_2 = A_2$ and we add some new arcs and vertices. We shall distinguish several cases according to $S(v)$.

- If $S(v) = \{0\}$, then we add two new vertices v' and v'' into V'' and two new arcs (v, v'), (v', v'') into A''_2.
- If $S(v) = \{1\}$, then we add two new vertices v' and v'' into V'' and two new arcs (v', v), (v, v'') into A''_2.
- If $S(v) = \{2\}$, then we add two new vertices v' and v'' into V'' and two new arcs (v', v''), (v'', v) into A''_2.

- If $S(v) = \{0, 1\}$, then we add one new vertex v' into V'' and one new arc (v, v') into A''_2.
- If $S(v) = \{1, 2\}$, then we add one new vertex v' into V'' and one new arc (v', v) into A''_2.

We claim, that D'' has solution to AIS3R with restrictions S'' if and only if D has solution to AIS3R with restrictions S. We shall at first show, that if we have solution c'' of AIS3R for D'' and S'', then $c = c'' \restriction_V$ is a solution for D and S. This can be observed from the fact, that added arcs ensure that $c''(v) \in S(v)$. For example if $S(v) = \{0\}$, then clearly in every solution to D'' and S'' there is only one possible assignment $c''(v'') = c''(v')+1 = c''(v)+2$ and hence $c''(v) = 0$. The cases when $S(v) = \{1\}$ or $S(v) = \{2\}$ are analogous. If $S(v) = \{0, 1\}$, then one added arc ensures, that $c''(v) = c''(v) + 1 \leq 2$ and hence $c''(v) \in \{0, 1\}$ as required. The case when $S(v) = \{1, 2\}$ is analogous. Now let us have solution c of AIS3R for D and S and let us extend it to the solution c'' of AIS3R for D'' and S. Given that the added arcs belong to A''_2, there is only one possible extension, which is possible since these arcs respect $S(v)$. We leave the details to the reader. By this we can also see, there there is one to one correspondence between the solutions of AIS3R for D'' with restrictions S'' and D with restrictions S.

Since in D'' with restrictions S'' there is one vertex with $S''(v) \neq \{0, 1, 2\}$ less, we can use induction hypothesis to directly achieve the proposition of the lemma. ■

Now, we shall concentrate on solving the problem AIS3. We shall at first describe how to find one solution for given graph D, then we shall show, how to generate all solutions with polynomial delay.

We shall start with observing, that we can assume, that D has only one source vertex (vertex with no incoming arc) and one sink vertex (with no outgoing arc). If this is not the case, then we can add vertices s, t to V, arcs from s to all source vertices of D to A_1, and arcs from every sink of D to t to A_1. It should be clear, that we in fact changed nothing on solvability of AIS3 on D, and if we denote the solution on the new graph c'', then c'' restricted to the original vertices is clearly solution of AIS3 on original graph. From now on we shall often use the restriction that D has only one source and one sink.

Algorithm 4.9 solves problem AIS3 for a given graph D.

Algorithm 4.9 AIS3

Input: *An acyclic directed graph* $D = (V, A = A_1 \cup A_2)$ *with* $A_1 \cap A_2 = \emptyset$ *and with one source* s *and one sink* t.
Output: *A function* $c : V \mapsto \{0, 1, 2\}$ *solving AIS3 if it exists, or* **no**.

```
1: if exists a vertex $v$ and a directed path from $s$ to $v$ containing more than two arcs
   from $A_2$
2: then
3:    return no
4: endif
```

```
 5: Let $O$ be a set of vertices, from which there is a directed path to $t$ containing two
    arcs from $A_2$.
 6: Let $H$ be a set of vertices, to which there is a directed path from $s$ containing two
    arcs from $A_2$.
 7: $L := \emptyset$
 8: $O', L', H' := \emptyset$
 9: **for each** $u \in O$
10: **do**
11:   $c[u] := 0$
12: **enddo**
13: **for each** $u \in H$
14: **do**
15:   $c[u] := 1$
16: **enddo**
17: **while** $O' \neq O$ **or** $L' \neq L$ **or** $H' \neq H$
18: **do**
19:   $O'' := O \setminus O', L'' := L \setminus L', H'' := H \setminus H'$
20:   $O' := O, L' := L, H' := H$
21:   Let $W$ be a set of vertices, which can be reached by undirected path from some
      vertex in $O \cup L \cup H$ in $D$ using only arcs from $A_2$.
22:   Find unique assignment of $c$ to vertices in $W$ using BFS (as in Algorithm 3.4).
23:   **if** such assignment does not exist
24:   **then**
25:     **return no**
26:   **endif**
27:   $O := \{u \in W \mid c[u] = 0\}$
28:   $L := \{u \in W \mid c[u] = 1\}$
29:   $H := \{u \in W \mid c[u] = 2\}$
30:   **if** exists vertices $u$ and $v$, such that $v \in O$ and there is a directed path from $u$ to
      $v$ which contains an arc from $A_2$
31:   **then**
32:     **return no**
33:   **endif**
34:   **if** exists vertices $u$ and $v$, such that $v \in H$ and there is a directed path from $v$ to
      $u$ which contains an arc from $A_2$
35:   **then**
36:     **return no**
37:   **endif**
38:   Add to $O$ all vertices $v \in V$, for which there is a vertex $u \in O$ and directed path
      from $v$ to $u$, or there is a vertex $u \in L$ and directed path from $v$ to $u$ which
      contains an arc from $A_2$, and assign $c[v] := 0$ for all these vertices.
```

```
39:     Add to $H$ all vertices $v \in V$, for which there is a vertex $u \in H$ and directed
        path from $u$ to $v$, or there is a vertex $u \in L$ and directed path from $u$ to $v$ which
        contains an arc from $A_2$, and assign $c[v] := 1$ for all these vertices.
40: enddo
41: Add to $L$ all vertices in $V \setminus (O \cup L \cup H)$, which can be reached from $s$ by arcs
    belonging only to $A_1$ and assign $c[v] := 1$ for all these vertices.
42: Add to $H$ all remaining vertices in $V \setminus (O \cup L \cup H)$ and assign $c[v] := 2$ for all these
    vertices.
43: return assignment $c$
```

Correctness of Algorithm 4.9 is justified by the following lemma.

Lemma 4.10 *Let $D = (V, A = A_1 \cup A_2)$ be an acyclic directed graph with $A_1 \cap A_2 = \emptyset$ and with one source s and one sink t. Then AIS3 is solvable for D if and only if Algorithm 4.9 finds a solution.*

Proof : We shall start with the following proposition. Let O, L, H are the sets in the Algorithm 4.9 after performing the while cycle on lines 17 – 40. Let c be any solution to AIS3 for D, then for every $v \in O$ is $c[v] = 0$, for every $v \in L$ is $c[v] = 1$, and for every $v \in H$. We shall show this proposition by induction on the number of cycles.

At the begining, O contains those vertices, from which t can be reached by a path containing two arcs from A_2. Obviously, these vertices have to have the value 0 in every solution to AIS3 for D. Similarly all vertices in initial H, i.e. those, which can be reached from s by a path containing two arcs from A_2, have to have value 1 in every solution to AIS3 for D. Now let us suppose, that the proposition is satisfied for O', L', H' at the begining of the while cycle on lines 17 – 40 and let us show, that the proposition is satisfied at the and of this cycle as well. It is easy to observe, that as in Algorithm 3.4, the values of vertices in W are uniquely defined, given the values of vertices in O, L, H. Therefore the proposition is satisfied after performing the line 29 as well. Moreover, if this unique assignment is not correct, then no solution to AIS3 may exist provided induction hypothesis is satisfied. Value of each vertex, added to O in step 38 is then forced to be 0, because this value has to be simply smaller or equal to 0. For the same reasons, if we return in step 32, then no assignment can exist, since there is vertex, which would have to get negative value. Similarly, steps 39 and 36 are correct as well.

Now let us inspect, what happens after the while cycle. Firstly, let us show, that when we add a vertex $v \in (O \cup L \cup H)$ into L in step 41 and we assign $c[v] = 1$ to these vertices, we do not break feasibility of function c. Let $(u, v) \in A$ be arbitrary, we shall inspect several cases.

- If $(u, v) \in A_1$, and both u and v are among newly added vertices, then clearly on this arc the condition is satisfied.
- If $(u, v) \in A_1$, and u is among newly added vertices, while v is not, then the only wrong case would be, if $v \in O$. However, in this case we would have added u into O in step 38, which was not the case.

- If $(u, v) \in A_1$, and v is among newly added vertices, while u is not, then the only wrong case would be, if $u \in H$. However, in this case, we would have added v into H in step 39, which was not the case.
- If none of u and v is among newly added vertices, then clearly the condition of AIS3 solution is satisfied, provided it has been satisfied until now.
- If $(u, v) \in A_2$, then it is not possible, that both u and v are among newly added vertices.
- If $(u, v) \in A_2$, and u is among newly added vertices, while v is not, then the bad case would be, if $v \in O \cup L$. If $v \in O$, then we would have rejected in step 32. If $v \in L$, we would have added u into O in step 38.
- If $(u, v) \in A_2$, and v is among newly added vertices, while u is not, then the bad case would be, if $u \in L \cup H$. If $u \in H$, then we would have rejected in step 36. if $u \in L$, we would have added v into H in step 39.

Therefore, c is still feasible on every arc of D.

Now let us consider the last step on line 42. Let $(u, v) \in A$ be arbitrary, and let us inspect several cases.

- If $(u, v) \in A_1$, and both u and v are among newly added vertices, then clearly, on this arc the condition is satisfied.
- If $(u, v) \in A_1$, and u is among newly added vertices, while v is not, then the bad case would be, if $v \in O \cup L$. If $v \in O$, we would have added u into O in step 38. If $v \in L$, but we have not added u into L in step 41, then there is a path from s to u containing an arc from A_2. Therefore, there is also a path from s to v containing an arc from A_2, which means we have not added v into L in step 41. Thus we must have added v to L during the while cycle on step 28, in which case, we would have added u into O in step 38, which we did not, and hence $v \notin L$.
- If $(u, v) \in A_1$, and v is among newly added vertices, while u is not, then clearly condition of AIS3 solution is satisfied on (u, v).
- If none of u and v is among newly added vertices, then clearly the condition of AIS3 solution is satisfied, provided it has been satisfied until now.
- If $(u, v) \in A_2$, then it is not possible, that both u and v are among newly added vertices, because otherwise there would be a path from s to v containing two arcs from A_2 and we would have put v into H at the beginning.
- If $(u, v) \in A_2$, and u is among newly added vertices, while v is not, then let us distinguish three cases. If $v \in O$, then we would have rejected in step 32. If $v \in L$, then we have not put v into L in step 41, but we must have done this during the while

cycle in step 28, in which case we would have put u into O in step 38. If $v \in H$ but we have not added u into L in step 41, then there must be a path from s to u containing an arc from A_2. Therefore, we would have added u into O during the initialization in step 5. Hence this case is not possible.

- If $(u, v) \in A_2$, and v is among newly added vertices, while u is not, then the bad case would be, if $u \in O \cup H$. If $u \in H$, then we would have rejected in step 36. if $u \in O$, we would have added v into L in step 28.

The only rejection step, which we have not mentioned yet is on line 3, which is obviously correct. Thus assignment c is returned if it exists and the returned assignment is correct, by which the proof is completed. ■

Now, we are ready to present Algorithm 4.11, which solves problem FCOP3 for the class of positive three valued functions.

Algorithm 4.11 FCOP3($\mathcal{C}_3^+$)

Input: *A p-p3mbuv* $F = (\tilde{U}, \pi)$.
Output: *An assignment* $\alpha_O \in \mathcal{A}_O(F)$, *for which* $(\tilde{U}^{\alpha_O}, \pi)$ *has a positive fully consistent extension or* **no**, *if no such assignment exists.*

```
 1: Construct directed graph $G = (\tilde{U}, E)$, where
        $$\begin{aligned} E &= E_1 \cup E_2 \cup E_3 \\ E_1 &= \{(u,v) \mid u,v \in \tilde{U} \wedge I(u) \lessapprox I(v)\} \\ E_2 &= \{(u,v) \mid u,v \in \tilde{U} \wedge u = \pi(v) \wedge F(u) = *_- \wedge F(v) = *_=\} \\ E_3 &= \{(u,v) \mid u,v \in \tilde{U} \wedge u = \pi(v) \wedge F(u) = F(v) = *_=\} \end{aligned}$$
 2: Find all strongly connected components $C_1, \dots, C_k$ of graph $G$.
 3: for $i$:=1 to $k$
 4: do
 5:    $S(C_i) := \bigcap_{u \in C_i} \Delta(u, n+1)$
 6:    if $S(C_i) = \emptyset$ or $(\exists u, v = \pi(u) \in C_i)\ [F(u) = *_- \wedge F(v) = *_+]$
 7:    then
 8:       return no
 9:    endif
10: enddo
11: Let $D = (V, A)$ be a directed graph, which is an acyclic condensation of $G$. I.e.
    $V = \{C_1, \dots, C_k\}$ and $A = A_1 \cup A_3$, where
        $$\begin{aligned} A_1 = \{(C_i, C_j) \mid (\exists u \in C_i)(\exists v \in C_j)\ (u,v) \in E_1\} \\ A_2 = \{(C_i, C_j) \mid (\exists u \in C_i)(\exists v \in C_j)\ (u,v) \in E_2\} \end{aligned}$$
```

```
12: for each $C \in V$
13: do
14:    if $S(C) = \{0\}$
15:    then
16:       Add two new vertices $C'$ and $C''$ into $V$ and two arcs $(C, C')$ and $(C', C'')$ into $A_2$.
17:    else if $S(C) = \{1\}$
18:    then
19:       Add two new vertices $C'$ and $C''$ into $V$ and two arcs $(C', C)$ and $(C, C'')$ into $A_2$.
20:    else if $S(C) = \{2\}$
21:    then
22:       Add two new vertices $C'$ and $C''$ into $V$ and two arcs $(C', C'')$ and $(C'', C)$ into $A_2$.
23:    else if $S(C) = \{0, 1\}$
24:    then
25:       Add one new vertex $C'$ into $V$ and a new arc $(C, C')$ into $A_2$.
26:    else if $S(C) = \{1, 2\}$
27:    then
28:       Add one new vertex $C'$ into $V$ and a new arc $(C', C)$ into $A_2$.
29:    endif
30: enddo
31: Add a new source vertex $s$ to $V$ and add arcs from $s$ to every vertex with indegree 0 into $A_1$.
32: Add a new sink vertex $t$ to $V$ and add arcs to $t$ from every vertex with outdegree 0 into $A_1$.
33: Using Algorithm 4.9 find solution of problem AIS3 for graph $D$.
34: if Algorithm 4.9 returned no
35: then
36:    return no
37: else
38:    return assignment returned by Algorithm 4.9
39: endif
```

We shall at first show, that Algorithm 4.11 works correctly.

Lemma 4.12 *Algorithm 4.11 works correctly, i.e. given p-p3mbuv $F = (\tilde{U}, \pi)$, it outputs an assignment $\alpha_0 \in \mathcal{A}_O(f)$, for which F has a positive fully consistent extension or rejects, if no such assignment exists.*

Proof : Correctness of the algorithm is justified by Lemma 4.5, 4.7, 4.8, and 4.10. Let us describe particular steps of the algorithm. It at first construct the same directed graph G as in Lemma 4.5 in step 1 and then it checks its strongly connected components in cycle on lines 3 – 10, correctness of possible rejecting on line 8 is justified by Lemma 4.5.

In step 11, we construct graph D, which is an input to the problem AIS3R as it was shown in Lemma 4.7. Then we construct an equivalent instance of problem AIS3 in steps $12 - 32$, as it was described in Lemma 4.8. Finally, we solve this instance of AIS3 using Algorithm 4.9 correctness of which was shown in Lemma 4.10. ■

Now we shall inspect running time of our algorithms. We start with Algorithm 4.9.

Lemma 4.13 *Algorithm 4.9 can be implemented in such a way, that its running time is bounded by $O(|V| + |A|)$.*

Proof : We shall at first describe, how to evaluate the maximum number of arcs on any path from s to all other vertices from V. We can do that simply using an algorithm for finding shortest path in a directed acyclic graphs, which works in time $O(|V| + |A|)$, since if we set length of every arc in A_1 to 0 and length of every arc in A_2 to -1, then shortest path with these lengths corresponds to longest path with respect to the number of arcs from A_2. Algorithm for finding shortest path in a directed acyclic graphs, which we would use, works also with negative weights.

By this it is clear, that condition on line 1 can be tested in linear time. Similarly construction of set O on line 5 and construction of set H on line 6 can be done in linear time as well. All steps until the start of *while* cycle on line 17 can be easily done in linear time, we shall therefore concentrate on how to implement the *while* cycle between lines 17 and 40.

If we represent sets $O, L, H, O', L', H', O'', L'', H''$ properly, e.g. as one list with several pointers to it, steps 19 and 20 can be done in constant time, i.e. in linear time over all cycles. On line 21 we have to be more careful. We can use for instance BFS for finding set W, we start from vertices in $O'' \cup L'' \cup H''$ and we do not use vertices in $O' \cup L' \cup H'$ in this search, since we have used them already in previous cycle. In this way, since all vertices in W will become members of future $O' \cup L' \cup H'$ in next cycle, we use every vertex only once in this BFS, therefore over all cycles, step 21 requires linear time. Note, that during this BFS, we can simultaneously find unique assignment to the vertices in W and we can update sets O, L, H accordingly, hence also steps $23 - 29$ can be done in linear time over all cycles. When we add a vertex v into O, we can also check, if there is a directed path from s to v which contains an arc from A_2, since this we can have precomputed from the begining of algorithm for every vertex. Similarly, we can check if there is a directed path from u to t containing an arc from A_2 for every u which we have added into H. Clearly, by this we can check conditions in steps 30 and 34 can be checked in linear time over all cycles.

The first part of line 38 is simple, since we only have to consider new vertices which we have added into O in this cycle, we can for each such vertex traverse back to source with ignoring vertices being already in O', in this way, we visit each vertex only once over all cycles. A bit mor tricky is second part, i.e. checking, whether there is a vertex $u \in L$ and a directed path from v to u containing an arc from A_2. We have not time for traversing back to source, since we could spend too much time on visiting vertices which we would not add into O. Let us denote by $d_s(u)$ maximum number of arcs from A_2 on a path from s to u and let us suppose, that we have this value precomputed during initialization phase.

Clearly, we are interested only to vertices $u \in L \setminus L'$ for which $d_s(u) = 1$. We shall traverse back to source with visiting only predecessors $v \notin O' \cup L' \cup H'$, such that $(v, u) \in A_2$ or $(v, u) \in A_1$ but $d_s(v) = 1$, in the latter case, we also add v into L ensuring, that we will not consider this vertex in later cycles. This clearly does not change anything on correctness of Algorithm 4.9, since such vertex v is forced to have value $c[v] = 1$ anyway. Similarly, we proceed with predecessors, since in this way each time, if we visit a vertex, we add it into O or L, we use every vertex only once and therefore we spend linear time over all cycles on line 38. Similarly we can proceed with line 39.

The remaining two steps on lines 41 and 42 can be clearly done in linear time. ■

Now, we can estimate time requirements of Algorithm 4.11.

Lemma 4.14 *Given a p-p3mbuv $F = (\tilde{U}, \pi)$ on nvariables on input, Algorithm 4.11 requires at most $O((n|\tilde{U}|)^2)$ time.*

Proof : Step 1 may take as much as quadratic time, since graph G has vectors from $\tilde{U}$ as vertices and therefore E may have size quadratic in $|\tilde{U}|$. To check, whether $(u, v) \in E$, we need time $\Theta(n)$ and hence constructing graph G may take time $(n|\tilde{U}|)^2$. The remaining steps are linear in the size of G, which is bounded by $|\tilde{U}|^2$. In this time we can find strongly connected components and according to Lemma 4.13, we can also solve problem AIS3 for graph D, size of which is linear in the size of G. ■

Now, we shall describe, how for a given p-p3mbuv $F = (\tilde{U}, \pi)$ we can output all assignments $\alpha_O \in \mathcal{A}_O(F)$ for which F^{α_O} has a fully consistent extension with polynomial delay. According to Lemma 4.7 and Lemma 4.8 it is sufficient to describe, how to enumerate all solutions of an AIS3 problem with polynomial delay.

Algorithm 4.15 Find All Solutions of AIS3

Input: *An acyclic directed graph $D = (V, A = A_1 \cup A_2)$ with $A_1 \cap A_2 = \emptyset$ and with one source s and one sink t. Initial sets O, L, H*
Output: *All functions solving AIS3 for D, if any exists, or* **no**.

```
1:  Perform the while cycle on lines 17–40 of Algorithm 4.9 on graph $D$ and sets $O, L, H$
    and modify these sets accordingly.
2:  if the while cycle rejected
3:  then
4:     return no
5:  endif
6:  if $(\exists v \in V \setminus (O \cup L \cup H))$
7:  then
8:     Recursively find all solutions of AIS3 on graph $D$ and sets $O \cup \{v\}, L, H$.
9:     Recursively find all solutions of AIS3 on graph $D$ and sets $O, L \cup \{v\}, H$.
10:    Recursively find all solutions of AIS3 on graph $D$ and sets $O, L, H \cup \{v\}$.
11: else
```

```
12:     output function $c$ which assigns values to vertices according to sets $O, L, H$.
13: endif
```

Lemma 4.16 *Let $D = (V, A = A_1 \cup A_2)$ be an acyclic directed graph with $A_1 \cap A_2 = \emptyset$ and let O, L, H be sets as initialized in Algorithm 4.9 during steps 1–16. Then Algorithm 4.15 outputs all solutions of AIS3 for graph D with delay $O(|V|(|V| + |A|))$ if called with parameters D, L, O, H.*

Proof : We shall at first show, that Algorithm 4.15 works correctly. We shall proceed by induction on height of an instance of Algorithm 4.15 in recursion tree. If we are at the bottom of recursion tree, i.e. we proceed with line 12, then there is clearly only one solution to be output as we do. If after making necessary unique assignments in step 1 we reject on line 4, then there is no assignment to be output, assuming correctness of step 1, which has been proved in Lemma 4.10. If there remains a vertex $v \in (O \cup L \cup H)$ with undecided value after performing step 1, then we simply try all three possibilities. If we assume by induction hypothesis, that recursive calls are correct, then we obviously output all solutions.

Now, let us show, that Algorithm 4.15 outputs solutions with delay $O(|V|(|V| + |A|))$. We shall start by showing, that Algorithm 4.15 outputs the first solution in time $O(|V|(|V| + |A|))$. This follows from the fact, that each instance of Algorithm 4.15 either rejects in linear time, or it outputs at least one solution. This is because, if it does not reject, then a solution exists and a vertex v which we choose has a value 0, 1, or 3 in this solution. If we use implementation described in the proof of Lemma 4.13, we can even observe, that the time required for rejecting is linear in $|V \setminus (O \cup L \cup H)|$. So the worst case scenario is, that we do not reject in step 1, and we receive rejection from the first two recursive calls. Hence, we have to account time $O(|V| + |A|)$ for this situation. Since there is at most $|V|$ vertices, recursive depth is at most $|V|$ and hence we get together, that before we output first solution, we spend at most $O(|V|(|V| + |A|))$ time. Clearly, the same can be told about time between two subsequent calls and we therefore leave details to the reader. ■

It is easy to modify all techniques for problem RCOP3, in fact the only change is in definition of graph G in Lemma 4.5, namely of arcs in E_1, where we would use robust potential inequality instead of potential inequality. We can conclude this section with the following theorem, proof which follows from other lemmas of this section.

Theorem 4.17 *Given a p-p3mbuv $F = (\tilde{U}, \pi)$ on n variables, we can find all solutions of FCOP3 and RCOP3 of F in time $O(n|\tilde{U}|^3)$. We can test, whether problems FCOP3 and RCOP3 have solutions and output one solution in time $O(n|\tilde{U}|^2)$.*

5 Conclusions

In this paper we have studied various extension problems for paired partially defined 3-valued functions with missing bits and uncertain value. In Section 3 we have concentrated on the unrestricted case when the extension can be any three valued function. We have shown that deciding about the existence of a consistent extension (CEP3) is NP-complete while constructing a fully consistent or robust extension (FCP3, RCP3) can be done in polynomial time (if one exists). These results are more or less straightforward extensions of similar results for the binary case for partially defined Boolean functions. Then we have extended the polynomial time results to problems of checking the existence of fully consistent outputs (FCOP3) and robustly consistent outputs (RCOP3).

Section 4 deals with the same extension problems as Section 3 except that the extending functions are required to be positive in all variables, which is a necessary condition in the analysis of DNA microarray data (see the discussion of this issue in the Introduction). The main results of the paper are the following five statements: the CEP3 problem is NP-complete (unlike in the binary case, where it is solvable in polynomial time), while FCP3 and REP3 are polynomially solvable. Finally, the problems FCOP3 and RCOP3 are also shown to be solvable in polynomial time, with the additional property that not just one but in fact all solutions to these problems can be generated with a polynomial delay.

References

[1] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pac. Symp. Biocomput.*, pages 17–28, 1999.

[2] Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano. Identification of genetic networks by strategic gene disruptions and gene over expressions under a boolean model. *Theor. Comput. Sci.*, 298(1):235–251, 2003.

[3] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. In *RECOMB '00: Proceedings of the fourth annual international conference on Computational molecular biology*, pages 8–14, New York, NY, USA, 2000. ACM.

[4] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara. Inferring qualitative relations in genetic networks and metabolic pathways. *Bioinformatics*, 16(8):727–734, 2000.

[5] Gabriela Alexe, Sorin Alexe, Tibérius O. Bonates, and Alexander Kogan. Logical analysis of data — the vision of peter l. hammer. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):265–312, 2007.

[6] E. Boros, T. Ibaraki, and K. Makino. Error-free and best-fit extensions of partially defined boolean functions. *Information and Computation*, 140:254 – 283, 1998.

[7] E. Boros, T.Ibaraki, and K. Makino. Extensions of partially defined boolean functions with missing data. Technical Report 6-96, RUTCOR Research Report RRR, Rutgers University, New Brunswick, NJ, 1996.

[8] Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Fully consistent extensions of partially defined boolean functions with missing bits. Technical Report 99-30, RUTCOR Research Report RRR, 2, 1999.

[9] H.J. Bussemaker, H. Li, and E.D. Siggia. Regulatory element detection using correlation with expression. *Nature Genetics*, 2(27):167–171, 2001.

[10] Ting Chen, Vladimir Filkov, and Steven S. Skiena. Identifying gene regulatory networks from experimental data. In *RECOMB '99: Proceedings of the third annual international conference on Computational molecular biology*, pages 94–103, New York, NY, USA, 1999. ACM.

[11] Y. Crama, P. L. Hammer, and T. Ibaraki. Cause-effect relationships and partially defined boolean functions. *Ann. Oper. Res.*, 16(1-4):299–325, 1988.

[12] J.L. DeRisi, V.R. Lyer, and P.O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278, 1997.

[13] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during cns development and injury. *Pac. Symp. Biocomput.*, pages 41–52, 1999.

[14] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[15] H. Imade, R. Morishita, I. Ono, and M. Okamoto. A grid-oriented genetic algorithm for estimating genetic networks by s-systems. *Proc. SICE Annual Conference*, 3:2750–2755, 2003.

[16] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, May 1993.

[17] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, volume 3, pages 18–29, 1998.

[18] Kevin Murphy and Saira Mian. Modelling gene expression data using dynamic bayesian networks. Technical report, University of California, Berkeley, 1999.

[19] M. Schena, D. Shalon, R. W. Davis, and P.O. Brown. Quantitative monitoring of gene expression pattern with a complementing dna microarray. *Science*, 270:467–470, 1995.

[20] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–74, 2002.

[21] Hiroyuki Toh and Katsuhisa Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical gaussian modeling. *Bioinformatics*, 18(2):287–297, 2002.

[22] Eugene P. van Someren, L. F. A. Wessels, and Marcel J. T. Reinders. Linear modeling of genetic networks from experimental data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 355–366. AAAI Press, 2000.

[23] D. C. Weaver, C. T. Workman, and G. D. Stromo. Modeling regulatory networks with weight matrices. *Pacific Symposium on Biocomputing*, 1999.