B. Steinbach
C. Posthoff

Logic Functions and Equations

Examples and Exercises

Springer

Logic Functions and Equations
Examples and Exercises

Bernd Steinbach · Christian Posthoff

Logic Functions and Equations

Examples and Exercises

Springer

Bernd Steinbach
Institute of Computer Science
Freiberg University of Technology
Bernhard-von-Cotta-Str. 2
09596 Freiberg
Germany
steinb@informatik.tu-freiberg.de

Christian Posthoff
University of the West Indies
Fac. Science & Agriculture
Cpy Augustine
Cpy Augustine Campus
Trinidad and Tobago
Christian.Posthoff@sta.uwi.edu

ISBN 978-1-4020-9594-8 e-ISBN 978-1-4020-9595-5

Library of Congress Control Number: 2008941076

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

This book is dedicated
to all our past, present
and future students

Contents

List of Figures

List of Tables

Preface

Tsutomu Sasao – Kyushu Institute of Technology, Japan

The material covered in this book is quite unique especially for people who are reading English, since such material is quite hard to find in the U.S. literature. German and Russian people have independently developed their theories, but such work is not well known in the U.S. societies. On the other hand, the theories developed in the U.S. are not conveyed to the other places. Thus, the same theory is re-invented or re-discovered in various places. For example, the switching theory was developed independently in the U.S., Europe, and Japan, almost at the same time [4, 18, 19]. Thus, the same notions are represented by different terminologies. For example, the *Shegalkin polynomial* is often called *complement-free ring-sum*, *Reed-Muller expression* [10], or *Positive Polarity Reed-Muller expression* [19]. Anyway, it is quite desirable that such a unique book like this is written in English, and many people can read it without any difficulties.

The authors have developed a logic system called **XBOOLE**. It performs logical operations on the given functions. With XBOOLE, the readers can solve the problems given in the book. Many examples and complete solutions to the problems are shown, so the readers can study at home. I believe that the book containing many exercises and their solutions [9] is quite useful not only for the students, but also the professors.

Here, I would like to show a list of key books [6, 11–15, 17], some of them are already out of print. Especially, [15] shows the logic design system developed in IBM in early 1960's. XBOOLE uses essentially the same logical structure as [7, 15]. However, you can find that this book is more focused on the solutions of Boolean equations [4], rather than the classical minimization of logical expressions.

Finally, I would like to mention that one of the authors, Prof. Dr. Bernd Steinbach, regularly organizes the **International Workshop on Boolean Problems**. This is a quite unique and important workshop to exchange the idea of the Eastern and Western people in Boolean problems. I hope he can continue this important project.

Foreword

This book is considered as an extension, or better, as a complement to the book "Logic Functions and Equations – Binary Models for Computer Science" [18] that has been published in 2004 by Springer. It can be, however, also used with any other course material that covers the required knowledge. Particularly for the area of **Logic Functions**, but also in many other areas, the use of computers and software for the solution of special problems is quite normal, however, the proper computer-based or computer-supported education and the preparation of graduates is not yet fully explored, and there are not only new possibilities, but also new and additional problems and difficulties.

Therefore, many academic institutions all over the world pay increasing attention to "computer-supported" or "computer-assisted" learning. The range of these concepts is very broad, it starts very often with PowerPoint presentations and ends with a fully Internet-based presentation of the teaching material. However, the didactics of such an approach, the advantages and disadvantages, the desired and the actual outcome are often not yet clear and deserve careful and serious consideration.

First of all it has to be emphasized that "computer-supported" not only considers the computer per se, but also the existence of appropriate software or even nowadays an appropriate (digital) multimedia environment which is specialized with regard to the topic to be taught. This combination of hard- and software will be the necessary assumption for the problems to be considered. And this already requires **additional efforts**. Not only is knowledge of the area required, but also the software, the working of the software as well as its proper use must be trained, and this is very often not so easy and can be a real additional burden.

As a second basic assumption we emphasize that it is **not** intended to replace the classroom or conventional teaching. Many people understand the computerization of teaching or learning mainly in replacing

the taking of notes in the classroom by slideshows, by handing over files with the text to the students etc. If this would be the intention, then a file of the respective textbook could be copied to each student, and this would be the solution.

It is our goal to increase the quality of learning considerably through the use of examples and exercises which can be handled on a much higher level when computer support can be or is provided. Sometimes, and in application-oriented areas all the time, computer-based solutions are the **only** possibility of solving such problems. A circuit with 100 gates cannot be designed by hand. We demonstrate this approach by using the book on "Logic Functions and Equations" [18] as the theoretical background (possibly as a textbook for a course "Logic Design" or "Logics for Computer Science" or similarly) and the software package XBOOLE as the learning software for a student. Similar problems arise, however, in many other areas.

Another difficulty that results from the application of this solution procedure is the **impossibility** of verifying the result of such a process. The result can thus only be accepted or rejected. Careful consideration and exploration of the models that are used, the testing of the solution procedure with examples from many different areas, repeated solving of the same problem by means of different algorithms or different software packages – these steps reduce the risk of accepting a result that is not correct – but that is not a guarantee at all.

In this way we want to contribute to the exploration of these difficult problems, we want to invite our colleagues to discussions, we invite our students to give us their feedback (for further considerations and improvements) – we also intend to dedicate a website to the extension of the spectrum of problems, to the discussion of the problems and to the publication of interesting results.

Christian Posthoff
Bernd Steinbach

Introduction

As we have said, there is not yet a fully approved and accepted methodology for this approach. However, we firmly believe that the following assumptions and conditions should be considered.

- *First assumption* The course material has been presented to the student in a proper way (by textbook, in class, by tutorials, assignments, ...) such that the student has a given understanding of the area which has to be extended, deepened and adapted for solving real-world problems (or at least more relevant for the solution of real-world problems). Very often in class only small examples will or can be presented which have more or less the character of toys.

- *Second assumption* The educational software must have a user interface which is sufficiently simple and problem-oriented. Nowadays everyone is using many sophisticated software packages all of which require a good amount of knowledge to be used properly. It is frustrating to sit before the screen and struggle with the peculiarities of the application program, instead of solving relevant problems.

- *Third assumption* A student must be able to transfer a given problem into the respective learning system. He/she must be able to split the problem into a sequence of subproblems which can be solved by using the (XBOOLE) software and the solution of which can be combined into a solution of the overall problem. The best method to learn and to manage this approach is the **learning from examples**, i.e. the presentation and explanation of a sufficiently large number of examples that can and have to be studied carefully and in detail.

Hence, this second book is provided with the same basic structure as the original textbook. Each chapter contains typical problems related to the contents of the respective chapter from the first book. All the

examples that are presented are explained in detail such that the result of a careful investigation of such a problem and independent solution trials result in a better understanding of the field. It is also possible to use the software for the solution of problems from other sources or courses, based on the understanding of the field acquired by studying these two books.

It is quite easy to understand that by using this methodology, at least two outcomes can be achieved:

1 The student will be free of a lot of routine calculations that are error-prone and do not really contribute to the problem-solving process; the solution process goes far beyond problems that can be solved by hand; in this way the educational process will be more relevant for real-life problems, hence, for the later professional life.

2 The student has more time and more possibilities and/or facilities to explore the solutions, to understand the meaning of the solutions, to represent functions graphically etc. which results in a thorough understanding of the problem area, the solution process, the applicability of theories etc.

This approach must be accompanied by (at least) the following steps:

- The educational process must put great(er) emphasis on modeling aspects. Students must be able to formalize real-world problems, to find limitations for the models, to evaluate the correctness of solutions etc. As a logical consequence, courses such as "Modeling and Simulation", "Scientific Computing" etc. must be introduced as soon as possible, compulsory for all Science and Engineering students.
- More specialized areas must be covered by appropriate software packages (which very often have to be created by the teaching academics themselves, see XBOOLE).
- The examination process also must take these developments into consideration. Computer-based examinations are much more appropriate to test the students' knowledge and skills than (trivial) calculations by hand and memorizing facts that can be stored on (tiny) hard disks.

The final solution which we have in mind for the use of XBOOLE comprises

- the textbook "Logic Functions and Equations – Binary Models for Computer Science" [18],
- the book "Logic Functions and Equations – Examples and Exercises",

- the software package XBOOLE that can be downloaded (for free),
- a collection of examples and solutions on a constantly growing web site (by contributions of students, academics, relevant applicants etc.).

Selecting an appropriate software for teaching we have to take two points of view into consideration.

- First of all we have to select the basic data structure for representing the problems, for its use within the algorithms and for doing the relevant calculations. This data structure must be both as closely as possible connected to the problems to be solved for teaching reasons and efficient in terms of memory and computation time because of the exponential growth of the size of Boolean spaces depending on their number of variables. Generally there are two classes of approaches: one of them can be based on sets of Boolean vectors and the other one on trees. The vector representation is more understandable for both the representation of logic functions and the solution sets of logic equations. Using ternary vectors in XBOOLE [2, 21, 18] the comprehensible representation can be combined with the required efficiency. Very popular are Binary Decision Diagrams (BDDs) suggested in [5], among others. This data structure is also efficient for the calculation of logic functions, but has disadvantages for comprehensible representations. For teaching reasons we decide to use XBOOLE. It should be mentioned that all the different approaches of this book can also be implemented using a BDD package such as, for instance, CUDD [20].
- Secondly we have to decide about the level of abstraction. On a high level a problem can be solved using a sequence of operations for logic functions or solution sets of logic equations. More in detail basically the same operations out of a library can be used together with a selected programming language.

  In order to emphasize how a certain problem can be solved, we selected procedures to solve the problems on a high level using the XBOOLE Monitor. The access to the basic operations is easily given in the XBOOLE Monitor by means of menus, toolbars, or commands. In order to solve larger problems, such commands are simply combined into a sequence of a so-called *Problem Program* (PRP). In that way we concentrate on the solution procedure and avoid programming details like preparation of the programming environment, definition and initialization of variables, deletion of data which are no longer needed or fitting pointers to the requirements of the used operations.

> Using the knowledge from this book it is easy to implement complete programs. The XBOOLE library was used, for instance, for the synthesis of combinatorial circuits by bi-decomposition [1], improved for fast circuits and low power consumptions [23] and including implicit test pattern generation [24]. The same basic approach was mapped to BDDs [5] in [22] and implemented using the CUDD package [20] exploiting several extended strategies in [16].

Our research and teaching in this area started nearly 40 years ago, and we considered all our results and lectures over this long period of time. Therefore, we read and used many books, articles, publications, approaches, discussions with colleagues, and we were not always able to trace back each problem or algorithm or theorem to a possible source which we have been using at a given point of time. But we hope that the list of references in [18] as well as here in this book is as correct as possible.

And finally, with these two books we simply want to make a contribution to the best of our abilities and experience, based on the long time of research and teaching in this and other areas to the best of our students. Any discussions, recommendations for improvement, contributions to the web presence will be highly appreciated.

Christian Posthoff
Bernd Steinbach

I

BASIC SOFTWARE

Chapter 1

XBOOLE MONITOR

1. XBOOLE Preliminaries

A logic function is a unique mapping from B^n into B [18]. Thus, 2^n values 0 or 1 define such a function and must be considered for each operation with this function. Logic operations with up to four, five or six variables can be performed by hand, because only $2^4 = 16$, $2^5 = 32$, or $2^6 = 64$ values must be used. In practical applications, however, the number of variables the logic function is depending on is much larger, therefore, thousands or millions of logic values must be manipulated without any error. Obviously a computer and suitable software are required to solve realistic logic problems.

There are several software tools for logic calculations. In this book we use one tool which is called *XBOOLE Monitor*. It provides a wide range of logic operations which can easily be combined to solve logic problems. Chapter 9 in [18] gives a short introduction to this tool that can be considered as a logic pocket calculator. The benefit of such a logic pocket calculator consists in the both fast and correct execution of each operation. Note: there is no restriction for the number of logic variables in the XBOOLE monitor. The remaining, but very important problem for the user of this logic pocket calculator is to know the right operations to be used and the right button to be pressed in order to get the required results.

All logic operations will be executed by functions of the XBOOLE Library [8]. The monitor wraps this library and simplifies the access to the XBOOLE functions for the user. There are several such wrapper programs usable for different operating systems. We use in this book the

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

Table 1.1 Meaning of the Files in the Directory `XBOOLEMonitor`

Name of the File	Meaning of the File
`xbm32.cnt`	content structure of the help file in use
`xbm32.exe`	executable file of the XBOOLE Monitor
`xbm32.hlp`	help file in use
`xbm32_e.cnt`	content structure of the English help file
`xbm32_e.hlp`	help file in English
`xbm32_g.cnt`	content structure of the German help file
`xbm32_g.hlp`	help file in German

XBOOLE Monitor `xbm32.exe` that provides a graphical user interface and runs under several versions of the Windows Operating System.

This XBOOLE Monitor can be downloaded by everybody for free from the following web page:

http://www.informatik.tu-freiberg.de/xboole.

At the left side of this page the link `XBOOLE Monitor` can be seen. A click on this link leads to the download page of the XBOOLE monitor. In order to download the XBOOLE Monitor, the button with the label `XBOOLE Monitor` located at the bottom of this page must be pressed. This opens the dialog `File Download` where the button `save` must be pressed in order to download the file `XBOOLEMonitor.zip` into a directory of your choice. After the download of this file the dialog `Download completed` appears. In this dialog the button `Open Folder` should be pressed. The folder which opens now contains the file `XBOOLEMonitor.zip` which has been loaded before. In the context menu the item `Extract all` can be chosen after a right click to extract all zipped files into a new directory `XBOOLEMonitor`. Of course, other procedures to extract all zipped files can be used as well. The XBOOLE monitor is now ready for use.

The directory `XBOOLEMonitor` includes the unzipped files of Table 1.1. It can be seen that the XBOOLE monitor can be used in English or German. The executable file of the XBOOLE-monitor is `xbm32.exe`. It can start without any further installation, and it selects automatically the language of both all representations and the help environment depending on the language of the Windows Operating System that is used on the respective machine.

All further explanations relate to this XBOOLE Monitor. Thus it is strongly recommended that the reader applies the procedure given above for downloading the XBOOLE Monitor to the respective computer such that it is available for all further logic calculations.

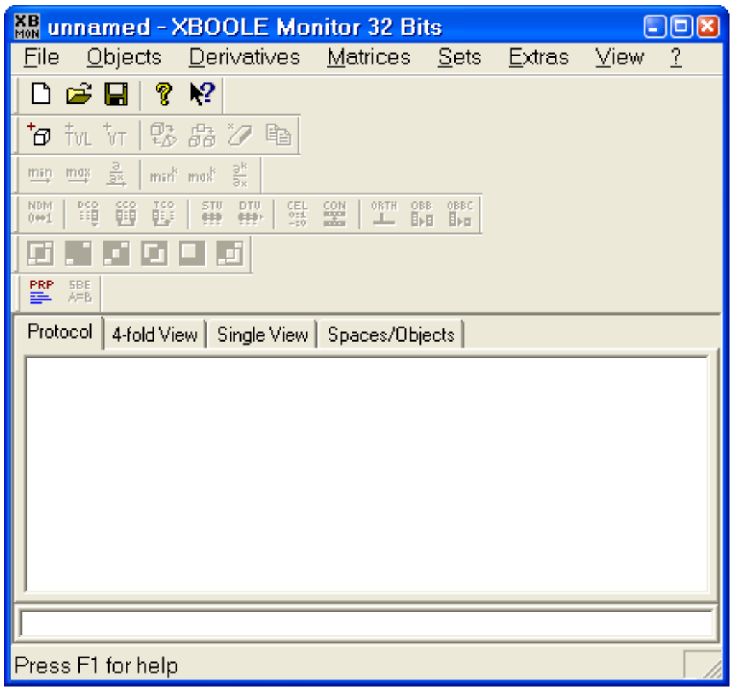


Figure 1.1 Complete window structure of the XBOOLE Monitor

2. The XBOOLE Window Structure

Figure 1.1 shows a screen shot of the XBOOLE monitor the size of which has been strongly minimized in order to meet the size of the pages of this book. The size of this window can be resized in the same way as any other window on the screen.

The window of the XBOOLE Monitor comprises several parts. In the following the main purpose of these parts will be explained.

The headline of the XBOOLE Monitor window shows the icon of this program, the name of the file opened or stored the last time, and the title of the program *XBOOLE Monitor 32 Bits*. The term `unnamed` as name of the file means that no file has been used so far. Otherwise a file name with the extension `sdt` is shown. Such an `sdt`-file allows to interrupt the work with the XBOOLE Monitor. After loading a stored `sdt`-file the work with the XBOOLE Monitor based on previous data can continue.

The menu bar is located below the headline of the XBOOLE Monitor. The menu is structured like a tree and allows the complete control of the behavior of the XBOOLE Monitor. Typically the menu items will be selected using the mouse. Alternatively the ALT-key in connection with

further keys of the keyboard can be used. Additional information for the selected action must be specified in most cases in special dialogs.

Below the menu bar a set of toolbars is located. Each toolbar can be shown or hidden separately. Its position can also be arranged by the user. All these adjustments are automatically stored in the Windows Registry database such that the adjustment of the toolbars does not change after the next start of the XBOOLE Monitor. The toolbars were introduced to shorten the management of the XBOOLE Monitor. Instead of selecting the whole path between the root and the leaf in the menu tree step by step, the action associated to the wanted leaf can be selected by a single click on the associated icon in a toolbar. Note: toolbar icons are defined only for actions which are used frequently. Thus the actions controllable by the toolbars are a subset of all available actions.

The part below the toolbars covers the main space of the XBOOLE Monitor window. The XBOOLE Monitor visualizes different types of information in this area, but also allows partially to interact with the user. This part is structured by a tab control. A click with the mouse brings the page associated to the tab in the foreground.

The first tab is labeled by `Protocol`. A protocol of each action executed by the XBOOLE Monitor is automatically created and shown on this page. The representation of the protocol does not depend on the way the action has been initialized. There are three possibilities to activate an action: first of all by an item of the menu, second of all by an icon of the toolbar, and third of all by a command of the command line. The protocol is written as a sequence of commands. There is a possibility to store the protocol and execute this sequence of commands again at a later point of time. Such a sequence of commands is called *problem program* or shortly `PRP`.

The second tab with the label `4-fold View` and the third tab with the label `Single View` provide basically the same behavior. The difference between these pages is that the 4-fold view consists of a 2×2-matrix, where each separate sheet of this matrix has the same behavior as the whole single view. In such a view a selected list of ternary vectors (TVL), an associated Karnaugh map, or a selected tuple of variables (VT) can be shown. There is an edit mode in such a view which allows to edit the elements of a TVL. If there is not enough space, scroll bars appear automatically and allow the selection of each part of the represented data. It is suggested to use the single view for very large objects. Otherwise the 4-fold view is preferred, because four objects can be seen at the same time.

Figure 1.2 shows an example of the 4-fold view of the monitor of XBOOLE. The top left part shows the first object, a TVL (list of ternary

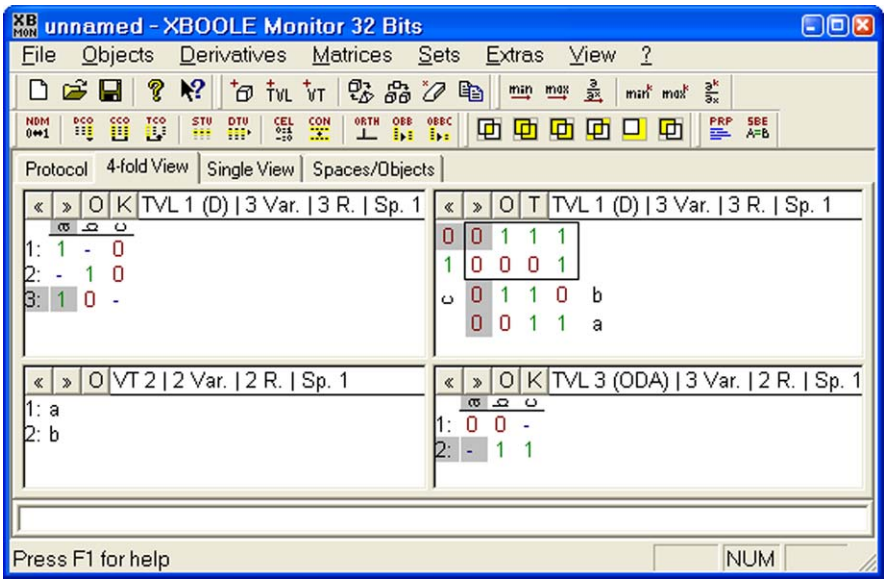


Figure 1.2 Example of the 4-fold view in the XBOOLE Monitor

vectors) in disjunctive form, depending on three variables, consisting of three rows, and defined in the first space. The top right part shows the Karnaugh map of this logic function. The button labeled by `K` in the TVL-representation switches to the Karnaugh map, and the button labeled by `T` in the Karnaugh map representation to the TVL, respectively. The bottom left part shows the second object, a tuple VT of two variables in the first space. The bottom right part shows the third object, a TVL in orthogonal disjunctive/antivalence (ODA) form, depending on three variables, consisting of two rows, and defined in the first space. This TVL was calculated as the complement of the first TVL.

The forth tab is labeled by `Spaces/Objects`. This page is divided vertically into two views. The left view shows a list of all details of the Boolean spaces defined by the user. The right view shows a list of all TVLs and VTs recently stored in the XBOOLE Monitor, where each of these objects is uniquely identified by its object number. In addition to the number of the object and its type information about the form, the space, and numbers of variables, rows and boxes are given.

The command line is located below the main part of the XBOOLE window. While the menu bar and the toolbar allow simple intuitive operations of the XBOOLE Monitor, the application of the command line

requires the knowledge of the commands for the control of the actions. The benefit of the command line is that XBOOLE operations can be activated faster, because interactions in dialog windows are omitted. The details of all commands can be studied in the XBOOLE help system.

The bottom of the XBOOLE window contains the status bar. In this bar help information about the action selected by a menu item or by a toolbar icon will be displayed.

3. XBOOLE Menu

The menu allows to activate each action executable by the XBOOLE Monitor. In this section we give some basic information about the menu. Detailed information will be explained in that section of this book where the actual menu item is used for the first time.

The menu is structured like a tree. The items directly connected to the root are shown in the menu bars visible in Figs. 1.1 and 1.2. If the mouse pointer moves over one of these items, the associated submenu appears. A left click on a high-lighted item activates the associated action.

Some of the actions require a precondition for their execution. It is, for instance, not possible to calculate the complement of a function if no function has been defined before. In such cases the menu item is deactivated by the XBOOLE Monitor in order to protect the XBOOLE user from errors. If a precondition does not hold, then the associated menu item appears in grey, and its activation is not possible.

Three points (...) behind the text of a menu item indicate that a dialog window appears which allows the input of some required information for the selected action. It is, for instance, possible to define the basic settings for the program by selecting the item `Settings...` in the submenu `File`. A click on this menu item opens the dialog window which allows to configure the appearance of the XBOOLE Monitor.

A triangle at the right of the menu item indicates that another submenu is associated. This exists only once in the menu `View` where a next menu opens while selecting the item `Toolbars`.

A tick before an item text in a menu indicates that the associated state is true. Such ticks are used in the `View` menu in order to indicate which parts are visible at present. For instance, one click onto the item `Status Bar` in the menu `View` removes the status bar and the tick associated to this item. The next click on the same item changes the state of the status bar again, that means the status bar and the tick in the item are visible.

All the details of the application of the XBOOLE Monitor can be found in the integrated help environment. There are two methods in order to get help information. First, the complete help information can be studied in a separate dialog window. This window will be opened by the item `Help Topics` in the menu labeled by '`?`'. The access to particular information topics is possible using structured contents, a predefined index, or a supported search across the whole help text. The second help method works context-sensitive. After clicking the item `Context Help` in the menu labeled by '`?`', a special cursor that includes a question mark appears. A click with this cursor on a menu item opens a help window that includes the help information associated to the selected item.

Exercise 1.1 (Help Topics). Activate in the menu labeled by '`?`' the item `Help Topics`. Select in the dialog window that appears now in the tab `Content` the item `Graphical User Interface` and there the item `Menus`. Study the meaning of the menu items.

In the initial state of the XBOOLE Monitor all items in the menus `Derivative`, `Matrices`, and `Sets` are deactivated, indicated by greyed items, because so far no TVLs exist. A TVL can be created using the item `Create TVL...` in the menu `Objects`, but this item is deactivated in the initial state of the XBOOLE Monitor as well, because up to now no Boolean space has been defined. Hence, it is necessary that the user of the XBOOLE Monitor defines at least one Boolean space.

The XBOOLE Monitor allows calculations for logic functions of an unlimited number of variables. These variables must be associated with Boolean spaces where the number of variables in each Boolean space is restricted by the user of the XBOOLE Monitor who defined such spaces. In order to define a new Boolean space, the item `Define Space...` in the menu `Objects` can be used. Figure 1.3 shows the Dialog window that appears after selecting this item. In the upper edit control the number of the Boolean space to be created is suggested and can be changed into a new number which has not been used so far as a space number. In the lower edit control the maximum number of variables in the range of 1 to 1952 can be specified. Note: this value cannot change later on. The suggested number of variables 32 fits to the word length of the CPU and is generally a good choice.

Exercise 1.2 (Define Space). Define two Boolean spaces using the menu. The number of variables in the first space must be 32 and in the second space 512, respectively. Visualize the results in the tab labeled by `Spaces/Objects`. Explain the values `Type` and `Variables`.

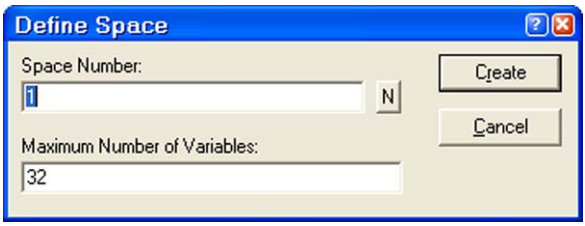


Figure 1.3 Dialog window that allows the definition of a new Boolean space

The definition of a Boolean space defines only the number of variables, but not their names nor the order of their appearance. Very often these names and their order are derived from the input, for instance, of an equation. However, sometimes it might be useful to display the variables in a given order, for instance, first the input variables and thereafter the output variables, or similarly. In order to achieve this, the names and the order of variables in a given space can be defined after the definition of the space, but before the input of a TVL. This can be achieved by using the menu items `Attach Variables...` or `Append Variable(s) to a VT...`

4. Toolbars

The toolbars allow a direct activation of a subset of actions of the menu. This speeds up the activation of XBOOLE actions, but requires the knowledge of the toolbar icons. The behavior of the toolbars is in agreement with the menu. An icon of a toolbar can be activated by a left mouse click only if the precondition of the associated action is satisfied. Otherwise the icon of a toolbar is deactivated which is visible by a grey icon.

There are six toolbars. Using the item `Toolbars` in the menu `View`, it is possible to show or to hide each one of them separately. The position of each toolbar can change by drag and drop. Figure 1.1 shows a vertical arrangement of all six toolbars, while in Fig. 1.2 the toolbars have been arranged in two rows.

The first toolbar `General` covers selected actions of menus labeled by `File` and '`?`'. The toolbar `Objects` covers seven of ten actions of the menu `Objects`. All actions of the menus `Derivatives`, `Matrices` and `Sets` are typically used very often. Therefore all these actions can be activated directly by the icon of the associated toolbars `Derivatives`, `Matrices` and `Sets`, respectively. The last toolbar `Extras` covers two of six actions of menus labeled by `Execute PRP` and `Solve Boolean Equation...`.

The meaning of the toolbar icons can be studied easily. The simplest way is to move the mouse pointer over an icon of a toolbar. Immediately a small yellow window appears which shows a quick info text that explains the meaning of the icon. Additionally, in the status bar a more detailed description is given. A well-structured complete information about the meaning of the icons in all toolbars can be obtained by using the help environment of the XBOOLE Monitor. In order to do this, the item `Help topics` can be used as described above, or simply the `F1` button can be pressed. The item `toolbars` in the first case or the link `Summary of the toolbars` lead to the same information page `Survey of the Toolbars` where information about each toolbar can be selected.

Exercise 1.3 (Toolbars). Use the two possibilities described above to study the meaning of all icons of all six toolbars.

As an example for the application of toolbars TVL 1 of Fig. 1.2 will be created. In order to do this, it is assumed that Exercise 1.2 has been executed; hence, space 1 exists. In order to activate the action that creates a TVL, it is sufficient to click with the left mouse button on the second icon of the toolbar `Objects`. This icon is labeled by `+ TVL`. All the information about the TVL which will be submitted now is transmitted to the XBOOLE Monitor in the same way as it is done for the initialization of a TVL using the item `Create TVL...` of the menu `Objects`.

The action to create a TVL opens first a dialog window shown in Fig. 1.4. Four properties of the TVL to be created must be defined in this dialog window. In a first step the Boolean space which comprises the TVL must be selected from a list. The next step defines an object number for this new TVL; this number can be used in the future to call this TVL. Note: if an object (TVL or VT) with this number already exists, the old object will be deleted. A simple way to select a new number (not yet used) consists in pressing the button labeled by `N` in the dialog window. In a third step the form of the TVL must be selected from a list of possible forms. The selected form `D` means that the TVL will be given in disjunctive form. Note: if the form `ODA` is selected, the input TVL will be taken as a D-form and orthogonalized internally. In a last (fourth) step the variables of the TVL columns must be written into an edit control, where the variable of the first TVL column is defined by the first row, each variable is defined in its own row, and the row order in the dialog window corresponds to the column order of the TVL to be created. The name of a variable may consist of up to 12 characters, the first character cannot be a digit. Finally a TVL

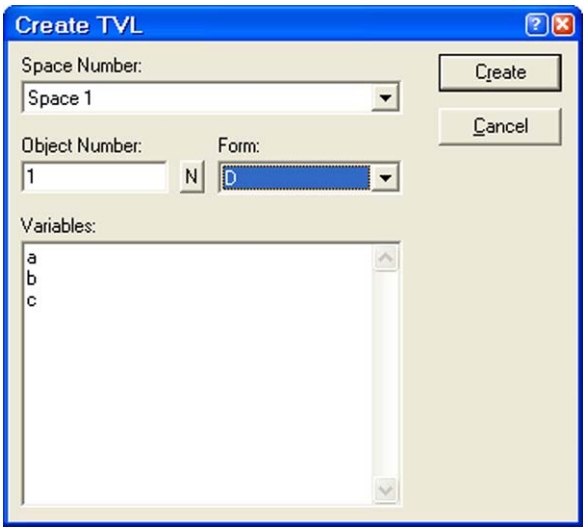


Figure 1.4 Dialog window to specify basic properties of a new TVL

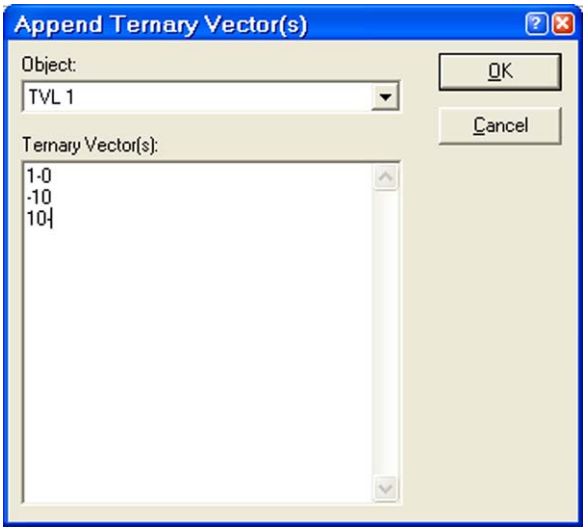


Figure 1.5 Dialog window to append ternary vectors to a selected TVL

which has no row as yet is created after pushing the button labeled `Create`.

There are two methods to define the rows of the TVL. The first one uses the possibility to append ternary vectors to a selected TVL. Appending all the required vectors to an existing empty TVL leads to the complete TVL. In order to do this, the item `Append Ternary`

`Vector(s)...` in the menu `Objects` must be activated, and this opens the dialog window shown in Fig. 1.5. The TVL which is supposed to be extended must be selected in this dialog window, and the respective ternary vectors must be written in an edit control, row by row. It is possible to change these rows. Pushing the OK-button starts the action which appends the defined rows to the selected TVL. This finishes the first method to define a TVL.

The second method to define the rows of a TVL uses the editor which is available in the `Single View` and in each area of the `4-fold View`. If a TVL is shown in such a view, a double click inside the view switches between the TVL mode that displays the TVL and the edit mode that allows to change given ternary values and to add ternary vectors to a TVL. The element to be changed can be selected by a mouse click on the position of the element. After the change or an input operation the cursor moves to the next element, first within the row and at the end of one row to the beginning of the next row. This movement of the cursor is controlled by the XBOOLE Monitor. For the input of a TVL only the keys `0`, `1`, and `-` are required. A double click finishes the edit mode. Alternatively, a right click in the view and the appropriate selection in the opened context menu can be used, a very useful approach which makes all the required input operations very feasible.

Exercise 1.4 (Toolbars). Use the second method described above for the input of the TVL 1 as shown in the top left part of Fig. 1.2. Note: because no other objects have been created in the XBOOLE Monitor, the created TVL will be shown in all four parts of the 4-fold view. Change the top right view such that the Karnaugh map is visible.

5. Command Line

In order to control the actions of the XBOOLE Monitor, a command language has been defined. All the information about a particular action to be executed will be included into the command; therefore no additional dialog windows are required. This speeds up the handling of the XBOOLE Monitor in comparison to the toolbars even more. Of course, the application of commands to control the actions of the XBOOLE Monitor requires the knowledge of the syntax of the command language.

In order to study the details of all commands, it is suggested to use the help environment that can be reached by the item `Help Topics` of the menu labeled '`?`'. In the contents tab there is an item `Command Line` that includes the following items:

- List of the Commands,
- Topics,
- Index of Commands (in alphabetic order),
- Index of Commands (ordered by topics).

A good starting point for the XBOOLE command language is the `List of the Commands`. Similar information is presented in the `Index of Commands (ordered by topics)`. There are four main categories of commands:

- Object Management,
- Derivatives,
- Operations for Matrices,
- Operations for Sets.

Actions that can be activated by items of the menus `File` and `Objects` can be activated alternatively by commands of the category `Object Management`. There is a complete association of the commands of the category `Derivatives` with the items of the menu `Derivatives`. The same statement is true for `Matrices` and `Sets`. The menus of the last three categories help the user to learn the XBOOLE command language, because the keyword of the command is written before the associated item text in the menu.

The XBOOLE command language is used in several implementations of XBOOLE Monitors. The XBOOLE Monitor used in addition to this book does not support all commands. These commands which are not supported are listed in a special section of the `Index of Commands (ordered by topics)`. If such a command is used in the command line, it does not cause an error, but it does not activate any action. An example for such a command is `help` which can simply be activated using the button `F1`.

Vice versa, the XBOOLE Monitor used in addition to this book supports several commands which have been defined in addition to the XBOOLE command language. The keywords of these commands begin with an underline character. These additionally supported commands are listed in the section `Extended Operation` of the index of commands ordered by topics. Most of these commands have a command in the XBOOLE command language which is more or less equivalent. They differ only in the use of VT: the VT is explicitly given in the command, not by an object number.

Exercise 1.5 (Commands). Study the XBOOLE commands in the help environment. Compare the commands for derivatives and matrices with the commands of the `Extended Operation`.

All XBOOLE commands have the same structure. The keyword is followed by parameters which specify the objects to be manipulated by this command. Based on predefined default values, some of the parameters can be omitted. If the command requires larger additional information, it must be given in consecutive lines and finished by a dot (.) on the same or next line.

One example is the command for the input of a VT that has the following syntax:

```
vtin [ sni [ vtno ] ]
var_list.
```

The keyword is `vtin`. The parameter `sni` means 'space number input' and specifies the Boolean space to which the VT must be assigned while the parameter `vtno` means 'VT number output' and specifies the object number for accessing this VT in the future. On the following lines the ordered list of variables (`var_list`) must be given. In the lines the variables are separated by space characters. The end of the list is indicated by a dot. The brackets indicate that the parameters are optional. If `vtno` is not specified the next free object number will be used. The default value for `sni` is equal to 1. If only one parameter is used, it must be `sni`. The command for input of the VT $\langle ab \rangle$ in Boolean space 1 as object number 2 is given as follows:

```
vtin 1 2
a b.
```

Practically you have to type `vtin 1 2` and press ENTER. Then the command line will be empty again, and you type a b. and press ENTER again. Then the object will be created and can be seen in one of the views.

An intelligent help system supports the user during the typing of the commands in the command line. Starting with the first letter, all fitting keywords of valid commands are shown in a small help window. After typing the complete keyword, the required parameter structure of the command is shown in this window. If the command needs further information in consecutive lines, the required syntax will be explained in this help window during the input. In this way it is easy to learn the command language.

Exercise 1.6 (Command Line). Extend the system of XBOOLE objects created in Example 1.4 by a VT $\langle ab \rangle$ with object number 2 in space 1.

Use the command line for the input of this VT and show the VT in the bottom left part of the `4-fold View`. Verify that the result is equal to the information that is shown in the bottom left part of Fig. 1.2.

Exercise 1.7 (Complement). Use the command line to calculate the complement of the logic function 1 defined in Example 1.4 and associate this new function with object number 3. Use the help environment to find the syntax of the command for the complement operation. Show the result in the bottom right part of the `4-fold View`. Verify that your `4-fold View` is the same as in Fig. 1.2. Verify the calculated complement operation. Change for this task the TVL representation of the right bottom part of `4-fold View` into a Karnaugh map representation and compare the Karnaugh maps of function 1 shown at the top right and its complement shown at the bottom right.

6. Problem Program

More complex tasks require a sequence of XBOOLE actions for calculating the solution. In order to achieve this, all three possibilities for controlling XBOOLE can be used conjointly. If in such a sequence an error occurs, certain parts of the sequence must be repeated. In such a situation problem programs (PRP) are a valuable support. Instead of activating the calculation step by step, a sequence of commands is written directly into a text file using any standard editor. Such a PRP can be executed by the XBOOLE Monitor step by step or completely without breaks. Such a PRP is also very helpful when the same sequence of XBOOLE actions must be executed several times for data which have completely or partially changed.

The content of a PRP is a sequence of XBOOLE commands. Any file name is allowed for a PRP, and the extension `.prp` is suggested.

A PRP can be created outside of the XBOOLE Monitor using any text editor or by the XBOOLE Monitor itself. As mentioned above, the XBOOLE commands of all actions executed by the XBOOLE Monitor will be included into a protocol, independent on the way of their specification. This protocol can be stored as a PRP file using the item `Save protocol as PRP...` in the menu `Extras`. In order to prepare several PRP files in this way, the old protocol can be deleted using the item `Delete Protocol` in the menu `Extras`.

The execution of a PRP can be activated by the item `Execute PRP...` in the menu `Extras` or by the icon `PRP` in the toolbar `Extras`. In both cases a dialog window with the name `Execute Problem Program` appears. A click on the button `Open PRP...` opens a file dialog where the PRP file can be selected. The content of the PRP file will be shown in

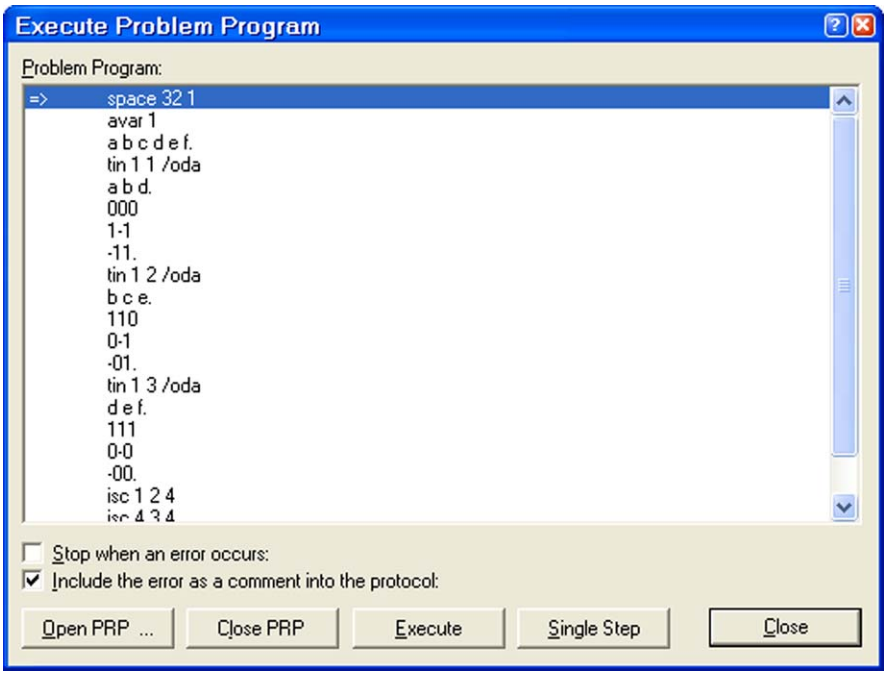
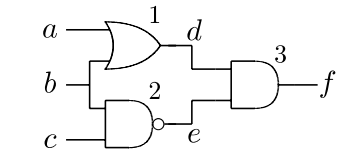


Figure 1.6 Dialog window for the execution of problem programs

the dialog window. Figure 1.6 shows this dialog window that includes a loaded PRP. Using the scroll bar the other commands of the PRP can be visualized. A click on the button `Single Step` executes a single command beginning with the first command and then according to the sequence of the PRP. A click on the button `Execute` executes the sequence of commands completely.

As an example we calculate the behavior of a logic circuit. Figure 1.7 depicts the structure of the circuit a) and lists the PRP that calculates the complete behavior in a minimized form b). The first command in the PRP defines a Boolean space 1 of 32 variables. The second command `avar` associates the variables a, b, c, d, e, and f in this order to the Boolean space 1. In this way a well ordered output is organized. Thereafter the input of the phase lists of the three gates is described by three commands `tin`. Each TVL is associated with the Boolean space 1. The object numbers of these TVLs correspond to the labels of the gates in Fig. 1.7 a). The behavior is calculated by the intersection of all three phase lists and stored as object number 4. Finally, by means of an orthogonal block building using the `obb` command, the resulting TVL is minimized with respect to the number of rows without



a)

```
space 32 1
avar 1
a b c d e f.
tin 1 1 /oda
a b d.
000
1-1
-11.
tin 1 2 /oda
b c e.
110
0-1
-01.
tin 1 3 /oda
d e f.
111
0-0
-00.
isc 1 2 4
isc 4 3 4
obb 4 4
```

b)

Figure 1.7 Structure of a simple circuit **a**) and the associated problem program to calculate the complete behavior **b**)

change of the object number. The loaded PRP is partially shown in Fig. 1.6.

Figure 1.8 shows in the `4-fold View` all basic TVLs and the TVL of the desired complete behavior. A comparison with the PRP of Fig. 1.7 b) reveals that the input TVLs are orthogonalized in the input action because the form `oda` was specified. An orthogonal form is a precondition for the intersection operation of XBOOLE. The result in the right bottom part of the `4-fold View` shows all behavior details of the circuit of Fig. 1.7 a). The output f is equal to 0 if either both inputs a and b are equal to 0 or if both inputs b and c are equal to 1. The output f is equal to 1 if either a is equal to 1 and b is equal to 0 or if b is equal to 1 and c is equal to 0. The values of the intermediate wires are listed in the columns d and e, respectively. The dashes in TVL 4 show that the inputs a and c decide about the value of the output f depending on the value of the input b.

Exercise 1.8 (PRP). Prepare a PRP file as shown in Fig. 1.7 b). Execute this PRP in the XBOOLE-monitor. Note: this PRP can be executed in a new XBOOLE Monitor window, without a precondition, because the space

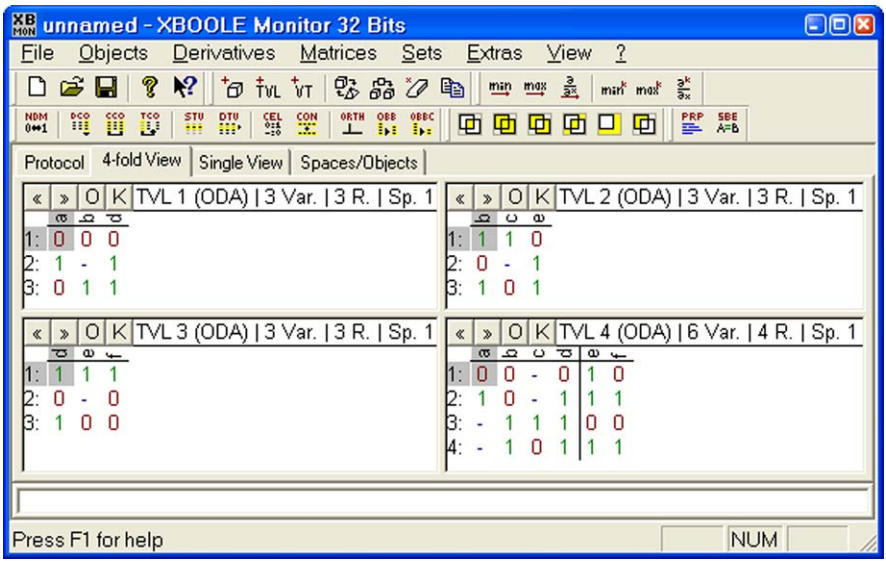


Figure 1.8 The complete behavior calculated by means of a PRP listed in Fig. 1.7

definition is included in the PRP. Show all the created TVLs in the `4-fold View` and verify the result using Fig. 1.8. Compare your PRP file with the created protocol. If there is no error, the only difference is the arrangement of data in lines. Store the protocol as a new PRP file and check that this machine-created PRP leads to the same result as before.

In the following chapters the solution of problems is sometimes given or discussed by some hints or explanations if the problem is not too difficult or the explanations support the material that has been taught. Very often, however, we will show the protocol of the solution. If this is the case, then the student should go carefully through the protocol, line by line, and try to understand each step, i.e. each line of the protocol.

7. XBOOLE Library

The basis of the XBOOLE Monitor is the XBOOLE library. Most of the functions of this library are wrapped by the XBOOLE Monitor for simple application. The XBOOLE library is written in the programming language `C` and can be used in programs written in programming languages `C`, `C++`, `Java`, and other languages on several platforms.

It is necessary to know many programming details in order to use the XBOOLE library. In this book we focus on the modeling of logic problems and not on programming aspects. Therefore the XBOOLE

```
#include "xb_pot.h"

void simplify(uns **ti, uns **res)
/* *ti pointer to the logic function to be simplified */
/* *res pointer to the simplified function */
{
  uns *xi;                        /* selected variable */
  xi = NULL;                      /* initialize xi */

  COPYOBJ(ti, res);               /* copy ti to res */
  while(SV_NEXT(ti, &xi, &xi))    /* select variable */
    if(TE_DERK(res, &xi))         /* if res does not */
                                  /* depend on xi */
      MAXK(res, &xi, res))        /* remove xi from res */
  OBB(res, res);                  /* minimize rows */
}
```

Figure 1.9 C-program to simplify a logic function using the XBOOLE library

library will not be considered in the remaining chapters. The XBOOLE library can be ordered using the following address:

Steinbeis-TZ Logische Systeme
Nelkentor 7
D-09126 Chemnitz
Germany
FAX: +49 371 5381 929
Email: stz158@stw.de.

By using a simple example, the application of the XBOOLE library will be indicated here. Assume there is a TVL in which k variables occur. It may be so that the associated logic function actually depends only on l variables, where $l < k$. It is the task of the function `simplify`, to create a new simplified function the TVL of which only includes such variables on which the logic function really depends.

Figure 1.9 shows the source code of a C-function that removes all dependent variables from a TVL and shortens the number of rows in the result. The prototypes of all functions of the XBOOLE library are defined in the header file `"xb_pot.h"`. In order to use the XBOOLE library, this header file must be included in the source file. All XBOOLE objects are managed by XBOOLE, and the access is possible by pointers to the type `uns` that is defined in the XBOOLE header too.

The function `simplify` gets access to the given TVL by the parameter `ti` and returns the simplified TVL by the parameter `res`. This function requires that an orthogonal TVL is given and ensures that the form of the TVL changes. All XBOOLE functions are indicated by capital letters. The function COPYOBJ copies the TVL `ti` to a new TVL `res` such that the given TVL will not change. In order to facilitate the access to each variable in `ti` separately, a TVL `xi` is used. After its definition and initialization the XBOOLE function `SV_NEXT` assigns in the first loop the first variable of the `ti` to the TVL `xi`, and in the following loops the next variables, one after the other, respectively. The XBOOLE function `SV_NEXT` returns the value `false` if no further variable exists in the TVL `ti`.

In the loop the XBOOLE function `TE_DERK` checks whether the derivative of `ti` with respect to `xi` is equal to 0. If that is true, the function `ti` does not depend on `xi`, and `xi` will be removed from the TVL `res` by means of a k-fold maximum operation realized by the XBOOLE function `MAXK`. The XBOOLE function `OBB` finally reduces the number of rows in the simplified orthogonal TVL `res`.

Chapter 2

BASICS AND LOGIC FUNCTIONS

In order to understand and to solve the problems in this chapter, it is recommended to read Chaps. 1, 2 and 3 of [18] and to follow carefully the introduction into the XBOOLE system. Make sure that you know the respective definitions and concepts and refer to the given examples.

1. Combinatorial Considerations in B and B^n

The use of the definitions for the relation $\leq$ and the operations $\wedge$ and $\vee$ will solve the following problem using the respective tables.

Exercise 2.1 (Relations in B). 1 Show that $x \leq (x \vee y)$ and $x \geq xy$.

2 Show that if $x_1 \leq y_1$ and $x_2 \leq y_2$ then $x_1x_2 \leq y_1y_2$ and $(x_1 \vee x_2) \leq (y_1 \vee y_2)$.

3 Show that $x \leq (y \vee z)$ if $x \leq y$ or $x \leq z$.

4 Show that $x \leq yz$ is equivalent to $(x \leq y)$ and $(x \leq z)$.

We remember that the positions from the right to the left of a binary vector can be considered as the values $x_0, x_1, \ldots$ of a binary number which have to be multiplied by $2^0, 2^1, \ldots$, and the products have to be added. Because of the values 0 and 1, only the values for the positions with the value 1 have to be added.

Exercise 2.2 (Binary Vectors). 1 Find the decimal equivalent $\mathrm{dec}(\mathbf{x})$ for the vectors $(1001) \in B^4$, $(01101) \in B^5$, $(110010) \in B^6$.

2 Find the vector $\mathbf{x} \in B^6$ with $\mathrm{dec}(\mathbf{x}) = 19$. Find the vector $\mathbf{x}$ for $\mathrm{dec}(\mathbf{x}) = 19$ in B^8.

3 Find the binary vectors $\mathbf{x}$ with $2^{n-1} \leq \mathrm{dec}(\mathbf{x}) < 2^n$.

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

4 Let be given the vector $\mathbf{x} = (10010101) \in B^8$.

(a) Find all $\mathbf{y} \in B^8$ with $\mathbf{x} \leq \mathbf{y}$.

(b) Find all $\mathbf{y} \in B^8$ with $\mathbf{y} \leq \mathbf{x}$.

5 Let be given two vectors $\mathbf{x}, \mathbf{y} \in B^n$ with $\mathbf{x} \leq \mathbf{y}$. Find all vectors $\mathbf{z}$ with $\mathbf{x} \leq \mathbf{z} \leq \mathbf{y}$.

This small example can also be used to find a solution based on *Logic Equations*. We remember that the relation $x \leq z$ can be equivalently translated into the equation $x \wedge \overline{z} = 0$. For two vectors $\mathbf{x}$ and $\mathbf{z}$ the equation must hold in each component, hence we get the equivalent system of equations

$$x_1 \wedge \overline{z}_1 = 0, \quad \ldots, \quad x_n \wedge \overline{z}_n = 0,$$

and in the same way

$$z_1 \wedge \overline{y}_1 = 0, \quad \ldots, \quad z_n \wedge \overline{y}_n = 0.$$

Now we use, for instance, $\mathbf{x} = (x_1, \ldots, x_6) = (010101)$ and $\mathbf{y} = (y_1, \ldots, y_6) = (110111)$ and insert the constants which results in

$$\begin{array}{lll} 0 \wedge \overline{z}_1 = 0, & 1 \wedge \overline{z}_2 = 0, & 0 \wedge \overline{z}_3 = 0, \\ 1 \wedge \overline{z}_4 = 0, & 0 \wedge \overline{z}_5 = 0, & 1 \wedge \overline{z}_6 = 0 \end{array}$$

and

$$\begin{array}{lll} z_1 \wedge 0 = 0, & z_2 \wedge 0 = 0, & z_3 \wedge 1 = 0, \\ z_4 \wedge 0 = 0, & z_5 \wedge 0 = 0, & z_6 \wedge 0 = 0. \end{array}$$

These two sets of equations define the solution immediately, and we get

$$z_1 = -, \quad z_2 = 1, \quad z_3 = 0, \quad z_4 = 1, \quad z_5 = -, \quad z_6 = 1,$$

i.e.

$$\mathbf{z} = (-101-1).$$

The two values $z_1 = -$ and $z_5 = -$ are based on the fact that the coefficients of z_1 and z_5 in both equations are equal to 0 which means that the equation is identically satisfied without binding the value of z_1 and z_5. This can also be seen from $x_1 = 0$, $y_1 = 1$ and $x_5 = 0$, $y_5 = 1$.

The function $\|\mathbf{x}\|$ counts the number of values 1 in a given vector.

Exercise 2.3 (Binary Vectors). Let $\mathbf{x}$ and $\mathbf{y}$ be two elements of B^n. Show that

1 $\|\overline{\mathbf{x}}\| = n - \|\mathbf{x}\|$;

2 $\|\mathbf{x} \vee \mathbf{y}\| = \|\mathbf{x}\| + \|\mathbf{y}\| - \|\mathbf{xy}\|$;

3 $\|\mathbf{xy}\| = \|\mathbf{x}\| + \|\mathbf{y}\| - \|\mathbf{x} \vee \mathbf{y}\|$.

Now we consider the *shell* of a sphere which is given by $S_i(\mathbf{x}, \mathbf{c}) = \{\mathbf{x} \mid h(\mathbf{x}, \mathbf{c}) = i\}$, for $i = 0, \ldots, n$, with h as the HAMMING-metric and $\mathbf{c}$ as the center of the sphere. The *open* and the *closed spheres with center* $\mathbf{c}$ themselves are given by $K_i(\mathbf{x}, \mathbf{c}) = \{\mathbf{x} \mid h(\mathbf{x}, \mathbf{c}) < i\}$ and $\overline{K}_i(\mathbf{x}, \mathbf{c}) = \{\mathbf{x} \mid h(\mathbf{x}, \mathbf{c}) \leq i\}$, resp. (see [18], pp. 33, 34). For closed spheres, the radius i will also have the values $0, \ldots, n$, for open spheres we can set $K_0 = \emptyset$ and allow $i = n + 1$ since $\overline{K}_n = K_{n+1}$.

Exercise 2.4 (Shells and Spheres). 1 Use $\mathbf{c} = (0000)$ and find $S_i(\mathbf{x}, \mathbf{c})$ for $i = 0, \ldots, 4$ with regard to this center.

2 Now use $\overline{\mathbf{c}} = (1111)$ and find $S_i(\mathbf{x}, \overline{\mathbf{c}}), i = 0, \ldots, 4$ with regard to this new center.

3 Confirm that $S_i(\mathbf{x}, \mathbf{c}) = S_{n-i}(\mathbf{x}, \overline{\mathbf{c}})$.

4 Let $n = 4, \mathbf{c} = (0000)$. Show that $K_0 = \emptyset$, $K_1 = S_0, \ldots, K_5 = S_0 \cup \cdots \cup S_4$. Generalize this relation to any value of n.

5 Let $n = 4, \mathbf{c} = (0000)$. Show that $\overline{K}_0 = S_0, \ldots, \overline{K}_4 = S_0 \cup \cdots \cup S_4$. Generalize this relation to any value of n.

6 What is the relation between spheres with center $\mathbf{c}$ and spheres with center $\overline{\mathbf{c}}$? Base your considerations on the analogous relation for shells.

2. Logic Functions, Formulas and Expressions

The definition of a logic function is quite easy and fully supported by the XBOOLE Monitor. After starting the XBOOLE Monitor we use the sequence `Objects` – `Define Space` – `Create TVL` – `Append Ternary Vector(s)`.

And now we simply write down all the (ternary) vectors for which the function to be defined has the value 1. When we are using the variables $x1$ and $x2$ and the only ternary vector is selected as (11), then $f = x1 \wedge x2$. By clicking on `K` the representation can change to the Karnaugh map for this function, the letter changes to `T`, and when we click on this letter now, we go back to the representation as TVL with the character `K` on the key.

A second XBOOLE possibility is the use of the concept of a *Logic Expression* or a *Logic Formula*. The menu point `Extras` offers the possibility `Solve Boolean Equation` which is, according to the philosophy of the book, one of the most powerful and sophisticated options of the monitor. After clicking on this topic a new window opens, and there the respective expression (formula) can be typed. Hint: the system is

using as a standard that the right side of an equation is equal to 1. We type, for instance, $x3\,\&\,x4$ and get the solution vector (11) and an object number for this solution set. The view is exactly the same as before for the definition based on TVL. Hence, the two possibilities are completely equivalent. In this way any formula can be used for the input of the respective function.

If we want to emphasize that an equation has a value of 1, then we can type $f = 1$. The system understands the $=$ as the equivalence function $\sim$, and, since we know that $f \sim 1$ is equal to f, we get the correct solution. In the case of an equation $f = 0$ the system solves $(f \sim 0) = 1$, and since $f \sim 0$ is equal to $\overline{f}$, we solve actually $\overline{f} = 1$ which means $f = 0$, again without any problem. The best way to avoid any confusion: type the equation to be solved without the value 0 or 1 on the right side and solve it. If a solution for the value 0 on the right side is required, then use additionally the set operation *complement*.

Exercise 2.5 (Definition of Functions). 1 Define the functions

$$f_0(x) = \overline{x}, \quad f_1(x,y) = x \wedge y, \quad f_2(x,y) = x \vee y,$$
$$f_3(x,y) = x \oplus y, \quad f_4(x,y) = x \sim y, \quad f_5(x,y) = x \to y,$$
$$f_6(x,y) = x|y = \overline{x \wedge y}, \quad f_7(x,y) = x \downarrow y = \overline{x \vee y}$$

using TVLs.

2 Define the same set of functions using `Solve Boolean Equations ...`

3 Which functions are represented by solving the equations $(x \wedge y) = 0$ and $(x \vee y) = 0$, resp. Use the Karnaugh map to define an appropriate expression for these functions.

4 Show that the expressions $(x \oplus y) \oplus z = 1$ and $x \oplus (y \oplus z) = 1$ define the same function?

5 Answer the same question for $\wedge$, $\vee$, $\sim$, $\to$.

Very often the question arises how many functions of a given type can be found. Problems of this kind are dealt with mostly by methods from combinatorics. As an example the following question should be considered.

Exercise 2.6 (Combinatorial Properties). 1 How many functions exist with $f(x_1, \ldots, x_n) = f(\overline{x}_1, \ldots, \overline{x}_n)$?

2 Two binary vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ are called neighbored if $h(\mathbf{x}, \mathbf{y}) = 1$. How many functions exist with $f(\mathbf{x}) = \overline{f(\mathbf{y})}$ for neighbored vectors $\mathbf{x}$ and $\mathbf{y}$?

3 How many functions of n variables exist with less than k values 1, $k \geq 1$?

Sometimes logic functions can be defined in a given context based on verbal descriptions. In a very general understanding this can be understood as "logic modeling" and sometimes be very difficult. We start here with a very simple example.

Exercise 2.7 (Logic Modeling). Find the Karnaugh map, a TVL and a disjunctive form for the following functions!

1 The function $f(x, y, z)$ has the value 1 either for $x = 1$ or if $y \neq z$ and the value of x is less than the value of z, otherwise the value of the function is equal to 0.

2 $f(x_1, x_2, x_3, x_4) = 0$ for such vectors satisfying $x_1 + x_2 > x_3 + 2x_4$. Hint: understand 0 and 1 as integers and $+$ as the addition for integers.

Exercise 2.8 (Definition of Functions by Formulas). Which functions are defined by the following formulas (equations):

1 $(x \to y) \oplus ((y \to z) \oplus (z \to x))$;

2 $\overline{(\overline{x} \vee y) \vee (x\overline{z})} \downarrow (x \sim y)$;

3 $\overline{x} \to (\overline{z} \sim (y \oplus xz))$;

4 $(((x \mid y) \downarrow z) \mid y) \downarrow z$.

Give the disjunctive, conjunctive, antivalence and equivalence normal forms for these functions.

Exercise 2.9 (Normal Forms). Find the disjunctive and the antivalence normal form of the following functions:

1 $f = ((x_1 \vee x_2\overline{x}_3x_4)((\overline{x}_2 \vee x_4) \to x_1\overline{x}_3\overline{x}_4) \vee x_2x_3)(\overline{x}_1 \vee x_4)$;

2 $f = ((x_1 \to x_2x_3)(x_2x_4 \oplus x_3) \to x_1\overline{x}_4) \vee \overline{x}_1$;

3 $f = (x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_9 \vee x_{10})(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee \cdots \vee \overline{x}_9 \vee \overline{x}_{10})$;

4 $f = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}$.

Exercise 2.10 (Normal Forms). Find the conjunctive and the equivalence normal forms of the functions given in the previous question.

Exercise 2.11 (Normal Forms). Generalize the two last items of the previous question to larger values of n:

1 $f = (x_1 \vee x_2 \vee \cdots \vee x_n)(\overline{x}_1 \vee \overline{x}_2 \vee \cdots \vee \overline{x}_n)$;

2 $f = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \oplus x_4 \oplus \cdots \oplus x_n$.

Exercise 2.12 (Antivalence Polynomial). Find the antivalence polynomial and the equivalence polynomial for the following functions:

1 $f = (x_1|x_2) \downarrow x_3$;

2 $f = (x_1 \rightarrow x_2(x_2 \downarrow x_3))$;

3 $f = ((x_1 \rightarrow x_2) \vee \overline{x}_3)|x_1$.

Not very often the transformation of a logic function into a "normal" polynomial is used. This polynomial only uses addition, subtraction and multiplication. A given elementary conjunction (containing all variables) will be translated in the following way: each variable remains unchanged, for negated variables x we write $1-x$, and then all variables are combined by multiplication. The use of elementary conjunctions ensures that at most only one conjunction will be equal to 1 for a given set of values, hence, the expressions for the different conjunctions can be added to form such a polynomial. As an example see, for instance, $C = x_1\overline{x}_2x_3\overline{x}_4 = x_1(1-x_2)x_3(1-x_4) = x_1x_3 - x_1x_3x_4 - x_1x_2x_3 + x_1x_2x_3x_4$. The value of C will be equal to 1 only for the vector (1010), for all the other vectors of B^4 it will be equal to 0.

Exercise 2.13 (Arithmetic Representation). Transform the following functions into the respective arithmetic polynomials:

1 $f = x_1 \oplus x_2 \oplus x_3$;

2 $f = (x_1 \rightarrow x_2) \rightarrow x_3$;

3 $f = x_1x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3$.

Sometimes the question arises whether a given formula represents a *tautology*, or as also can be said whether the function represented by a given formula is always equal to 1. Again the use of the XBOOLE monitor makes this kind of questions very easy. The formula has to be entered, and the solution of the respective equation answers this question immediately. Since, however, many different TVLs can represent a tautology, it is elegant and efficient to use the complement, because this complement must be the empty set, independent on the representation of the tautology.

Exercise 2.14 (Tautologies). Which one of the following formulas defines a tautology?

1 $(x \rightarrow y) \rightarrow ((x \vee z) \rightarrow (y \vee z))$;

2 $((x \oplus y) \sim z)(x \rightarrow yz)$;

3 $((\overline{x} \vee \overline{y}) \downarrow (x \oplus \overline{y})) \oplus (\overline{(x \to \overline{y})} \to (\overline{x} \vee y))$;

4 $((x \vee \overline{y})z \to ((x \sim z) \oplus y))(x(yz))$.

Very often we find simple or complex identities with the meaning that one side of such an identity defines the same function as the other side. In this case the two formulas can be considered as *equivalent*. A simple example is the expression

$$a(b \vee c) = ab \vee ac,$$

the function defined by $a(b \vee c)$ is supposed to be the same function as the function defined by $ab \vee ac$. The solution of these problems can be achieved in two ways:

- We take the XBOOLE Monitor facility for solving equations and solve the two equations *left side* $= 1$ and *right side* $= 1$ and store the solution sets. Then we compare the solution sets using the *symmetric difference*. If the result is the *empty set*, then the identity is correct, otherwise not.
- We type the suspected identity as it is. The system understands the $=$ as the equivalence, and since the equivalence is equal to 1 for $0 = 0$ and $1 = 1$, we also get all the vectors for which the identity holds, and if the set of all these vectors is equal to the respective B^n, the identity is valid. This means that we ask the question whether the given expression represents a tautology.

Exercise 2.15 (Identities). Are the following pairs of formulas equivalent – try to prove this equivalence by building the disjunctive normal form of the two formulas.

1 $f_1 = (x \vee y \vee z) \to (x \vee y)(x \vee z)$; $f_2 = x \sim z$;

2 $f_1 = (x \to y) \to z$; $f_2 = x \to (y \to z)$;

3 $f_1 = [(x \oplus y) \to (x \vee y)][(\overline{x} \to y) \to (x \oplus y)]$; $f_2 = x \mid y$;

4 $f_1 = \overline{(x \to y) \vee (x \to z)y}$; $f_2 = (x\overline{y})(\overline{y} \to x\overline{z})$.

Correct identities (or *equivalent formulas*) can be used as a possibility for the transformation (simplification, normalization) of formulas and should be well known for this purpose. They can be used from the left to the right or from the right to the left (dependent on the respective effect).

Exercise 2.16 (Transformation Rules). Can the following rules be used?

1 $x \vee (y \sim z) = (x \vee y) \sim (x \vee z)$;

2 $x \rightarrow (y \sim z) = (x \rightarrow y) \sim (x \rightarrow z)$;

3 $x \wedge (y \sim z) = (x \wedge y) \sim (x \wedge z)$;

4 $x \rightarrow (y \vee z) = (x \rightarrow y) \vee (x \rightarrow z)$;

5 $x \rightarrow (y \wedge z) = (x \rightarrow y) \wedge (x \rightarrow z)$;

6 $x \oplus (y \rightarrow z) = (x \oplus y) \rightarrow (x \oplus z)$;

7 $x \rightarrow (y \rightarrow z) = (x \rightarrow y) \rightarrow (x \rightarrow z)$.

The composition of functions is a powerful mechanism that can and will be used very often. The basic idea is the *implementation of "smaller" functions* and its combination by other functions. As an illustration answer the following question.

Exercise 2.17 (Composition of Functions). Let be given $f(a, b) = a \vee \overline{b}$ and $g(x_3, x_4) = x_3 \sim x_4$.

1 Find all vectors (x_2, x_3, x_4) with $h(x_2, x_3, x_4) = f(g(x_3, x_4), x_2)$.

2 Find all vectors (x_1, x_2, x_3, x_4) with $h(x_1, x_2, x_3, x_4) = f(x_1, x_2) \vee g(x_3, x_4)$.

3 Find all vectors (x_1, x_2, x_3, x_4) with $h(x_1, x_2, x_3, x_4) = f(x_1, x_2) \wedge g(x_3, x_4)$.

3. Special Functions and Representations

One of the most interesting problems is the question whether a function (given by a formula, a description, a table) is an element of a special class of functions (is *linear* or *monotone* or *linearly degenerated* etc.). This question will be dealt with in Chapter 4 using operations of the Boolean Differential Calculus. Here we will give some examples for these functions using the XBOOLE Monitor.

Exercise 2.18 (Special Formulas). Show the TVL and the Karnaugh map of the following functions:

1 $f_1 = x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}$;

2 $f_2 = x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \vee x_7 \vee x_8 \vee x_9 \vee x_{10}$;

3 $f_3 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}$;

4 $f_4 = x_1 \sim x_2 \sim x_3 \sim x_4 \sim x_5 \sim x_6 \sim x_7 \sim x_8 \sim x_9 \sim x_{10}$.

Exercise 2.19 (Implication). 1 Find the TVLs for the functions $f = (x \to y) \to z$ and $g = x \to (y \to z)$.

2 Which function is the XBOOLE Monitor using for $x \to y \to z$.

3 Discuss the result received for $f_1 = x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x_6 \to x_7 \to x_8 \to x_9 \to x_{10}$.

Exercise 2.20 (NAND and NOR). 1 Compare the two functions $f = (x|y)|z = \overline{(x \wedge y)} \wedge z$ and $g = x|(y|z) = x \wedge \overline{(y \wedge z)}$.

2 Compare $f = (x \downarrow y) \downarrow z = \overline{\overline{(x \vee y)} \vee z}$ with $g = x \downarrow (y \downarrow z) = \overline{x \vee \overline{(y \vee z)}}$.

3 What can be said about $f = x_1|x_2|x_3|x_4$ and $g = x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4$?

The construction of *disjunctive* and *conjunctive forms* and *normal forms* is easy when the XBOOLE Monitor will be used. Again a given overlap with the concept of *logic equations* exists and can be successfully used.

Exercise 2.21 (Conjunctive and Disjunctive Normal Forms). Find a disjunctive and a conjunctive form for the following functions:

1 $f_1 = (x \vee y\overline{z})(x \vee z)$;

2 $f_2 = ((x_1 \vee x_2\overline{x}_3x_4)((\overline{x}_2 \vee x_4) \to x_1\overline{x}_3\overline{x}_4) \vee x_2x_3) \vee (\overline{x}_1 \vee x_4)$;

3 $f_3 = ((x_1 \to x_2x_3)(x_2x_4 \oplus x_3) \to x_1\overline{x}_4) \vee \overline{x}_1$.

In order to find the disjunctive normal form, any ternary vector with the value – at a given position will be replaced by two vectors with the values 0 and 1 at this position, resp. Sometimes the function might be given by means of a vector of the function values, such as (01101100) or (10001110). In this case the assignment of the argument vectors to the positions of the vector of the function values must be known. The given functions can, for instance, be represented by the following table:

x	y	z	f_1	f_2
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

Very often this order of the assignment of the vector components of f to the vectors for the variables (xyz) from the left to the right is assumed, more or less as a standard, any other assignment of the variables to the positions of the vector of the function values must be mentioned appropriately. This arrangement is very popular because the decimal equivalent of the respective vectors corresponds to the integer numbers $0, 1, \ldots, 7$ or to $0, 1, \ldots, 2^n - 1$ for n variables. Therefore we can also write $f_1 = (01101100)$, $f_2 = (10001110)$.

Exercise 2.22 (Function Vectors). 1 Find the disjunctive and the conjunctive normal forms for the two given function vectors.

2 Find shorter disjunctive and conjunctive forms using the Karnaugh map.

3 Use the items `OBB Orthogonal Block Building` and `OBBC Orthogonal Block Building and Change` of the XBOOLE Monitor in order to find shorter versions.

Exercise 2.23 (Special Normal Forms). How many disjunctions (conjunctions) will be used for the conjunctive (disjunctive) normal forms of the following functions:

1 $f = x_1 \oplus x_2 \oplus \cdots \oplus x_n$;

2 $g = (x_1 \vee x_2 \vee \cdots \vee x_n)(\overline{x}_1 \vee \overline{x}_2 \vee \cdots \vee \overline{x}_n)$;

3 $h = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \oplus x_4 \oplus x_5 \oplus \cdots \oplus x_n$. Start your considerations with $n = 4$, $n = 5$, $n = 6$ and try to find a general rule.

Since the property of orthogonality is one of the fundamental principles of the solution process for Boolean equations, it is very easy to find antivalence forms of functions. We simply solve the equation $f = 1$ for any function f. Since the solution is given as a set of conjunctions $K_1, K_2, \ldots$ with the additional property that $K_i \wedge K_j = 0$ for $i \neq j$ (orthogonality), we can simply calculate the disjunctive form of the solution set and replace the $\vee$ by $\oplus$.

Exercise 2.24 (Antivalence Normal Forms). Find an antivalence form for the following functions:

1 $f_1 = (x \vee y\overline{z})(x \vee z)$;

2 $f_2 = ((x_1 \vee x_2\overline{x}_3x_4)((\overline{x}_2 \vee x_4) \to x_1\overline{x}_3\overline{x}_4) \vee x_2x_3) \vee (\overline{x}_1 \vee x_4)$;

3 $f_3 = ((x_1 \to x_2x_3)(x_2x_4 \oplus x_3) \to x_1\overline{x}_4) \vee \overline{x}_1$.

In order to find antivalence forms without complemented variables (Shegalkin polynomials), we remember the two rules $\overline{x} = 1 \oplus x$ and

$x(1 \oplus y) = x \oplus xy$. That means that each value 0 in a conjunction of an antivalence form results in two conjunctions, one without this variable and the other one with the value 1.

Exercise 2.25 (Shegalkin Polynomials). 1 Transform the antivalence forms of the previous problem into Shegalkin polynomials.

2 Find the Shegalkin polynomial for $f = x_1 \vee x_2 \vee x_3$.

The concept of a subfunction can be interesting if only some parts of a function are important in a given context. Any subfunction can be found by specifying the values of some variables. One very clear way is the definition of the function and the intersection with the respective value, the other possibility inserts the values of some variables as a constant into the formula and calculates the respective subfunction. Observe that the subfunctions do not depend on the variables that have now a fixed value.

Exercise 2.26 (Subfunctions). 1 For $f = ((x_1 \to x_2\overline{x}_3) \oplus x_2)\overline{x}_2$, find subfunctions $f_1(x_2, x_3) = f(x_1 = 1, x_2, x_3)$, $f_2(x_1, x_3) = f(x_1, x_2 = 1, x_3)$, $f_3(x_3) = f(x_1 = 1, x_2 = 0, x_3)$ using the intersection and the insertion of constants.

2 Which functions $f(x_1, x_2)$ do not change the value when x_1 and x_2 are exchanging their positions?

When we speak about special logic functions, then very often three questions have to be answered, mostly in a different context:

- how many functions of this special nature can be found (*combinatorial considerations*);
- find some (all) functions with this property (*constructive aspect*);
- check whether a given function is an element of this class (*analytical aspect*).

Exercise 2.27 (Degeneration of Functions). 1 Find all conjunctively degenerated functions of three and four variables. Explain the construction! What is the common property of these functions?

2 Find all disjunctively degenerated functions of three and four variables. Explain the construction! What is the common property of these functions?

3 Find all linearly degenerated functions of three and four variables using the antivalence. Explain the construction! What is the common property of these functions?

4 Find all linearly degenerated functions of three and four variables using the equivalence. Explain the construction! What is the common property of these functions?

Exercise 2.28 (Dual and Self-Dual Functions). Let be given the following functions:

- $f_1 = (11100111)$;
- $f_2 = (01110001)$;
- $f_3 = (11001101)$;
- $f_4 = x_1x_2 \vee \overline{x}_2(x_3 \oplus x_4)$.

Find for each function its dual function! Is one of these functions a self-dual function?

Exercise 2.29 (Symmetric Functions). 1 Find functions $f(x_1, x_2, x_3)$ symmetric in (x_1, x_2).

2 Find functions $f(x_1, x_2, x_3)$ symmetric in (x_2, x_3).

3 What can be said about the intersection of these two sets?

Exercise 2.30 (Symmetric Functions). 1 Find the number of functions $f(x_1, \ldots, x_n)$ which are symmetric in (x_1, x_2), $n \geq 2$.

2 Find all functions that do not change their values for any permutation of the variables.

Exercise 2.31 (Monotone Functions). Which functions of the given set are monotone? Check the increasing as well as the decreasing possibility.

1 $f_1 = (x \to (x \to y)) \to (y \to z)$;

2 $f_2 = x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4$;

3 $f_3 = (0000000010111111)$;

4 $f_4 = (0001010101010111)$.

Monotone functions are in a very strict sense considered as *monotonely increasing* and *monotonely decreasing*. However, this difference is not so important since it is easy to see that a function $f(\mathbf{x})$ is monotonely increasing if and only if $\overline{f(\mathbf{x})}$ is monotonely decreasing and vice versa. Therefore most of the time it is sufficient to deal with one of these properties or to assume the consideration of increasing functions more or less as a standard.

Exercise 2.32 (Monotone Functions). For which value of n are the following functions monotone?

1 $f_1(x_1, \ldots, x_n) = x_1x_2 \vee x_1x_3 \vee \cdots x_{n-1}x_n$ (the disjunctions of all conjunctions consisting of two non-negated variables);

2 $f_2(x_1, \ldots, x_n) = x_1x_2 \ldots x_{n-1}x_n \to (x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1} \oplus x_n)$.

Exercise 2.33 (Monotone Functions). For each monotone function, we have

$$f(\mathbf{x}) = x_i f(x_i = 1) \vee f(x_i = 0),$$

and

$$f(\mathbf{x}) = (x_i \vee f(x_i = 0)) \wedge f(x_i = 1).$$

Prove these identities.

Exercise 2.34 (Monotone Functions). Let be given $f(x_1, x_2, x_3, x_4)$ such that $f(0, 1, 1, 0) = 1$, $f(1, 1, 0, 0) = 1$, $f(1, 0, 1, 0) = 0$, $f(0, 0, 1, 1) = 1$, $f(0, 1, 0, 1) = 0$.

1 Can this definition be used to build monotone functions with these values?

2 How many different monotone functions with these values can be built?

3 Represent these functions by disjunctive forms without negated variables.

Exercise 2.35 (Functional Constraints). Find all functions satisfying the following conditions:

1 $f(1, 0, 0, 0) = 1$, $f(0, 1, 1, 1) = 0$;

2 $f(1, 0, 0, 0) = 1$, f linear in one or more variables;

3 $f(0, 1, 0, 0) \neq f(1, 0, 1, 1)$, f symmetric (consider all possibilities);

4 $f(1, 0, 0, 1) = 0$, f self-dual.

4. Minimization

In Chap. 2, Sect. 4 of [18] the Method of BLAKE and the Algorithm of QUINE-MCCLUSKEY are represented; they give the prime implicants of a function and all irredundant disjunctive forms for a given function f.

We remember that an *implicant* of a function f is a conjunction K with the property $K \leq f$, i.e. $K \vee f = f$ and $K \wedge f = K$. A *prime implicant* is an implicant where this property will be lost if one of the variables in the conjunction is deleted.

Exercise 2.36 (Prime Implicants). Find the prime implicants of the following functions:

1 $f(x, y, z) = (00101111)$;

2 $f(x, y, z) = (01111110)$;

3 $f(x_1, x_2, x_3, x_4) = (1010111001011110)$.

Exercise 2.37 (Minimized Disjunctive Normal Form). What can be said about the minimization of the following functions:

1 $f_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$;

2 $f_2 = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \oplus x_4 \oplus x_5$;

3 $f_3 = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)(x_4 \oplus x_5)$;

4 $f_4 = (x_1 \oplus x_2 \oplus x_3)(x_4 \oplus x_5 \oplus x_6)$;

5 $f_5 = (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)(x_1 \vee x_2 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_5 \vee \overline{x}_6)$?

The XBOOLE Monitor is very flexible with regard to the variables which are used for a function. We can type, for instance, $f = a \vee b$ and $g = b \vee c$ as equations to be solved and consider $f \vee g$ which is solved by the union of the sets of ternary vectors for f and g, respectively. And when we check the solution sets then we see that the result depends on a, b and c. What has been happening in the background, is the extension of functions by variables or the embedding of functions in larger spaces. For the variables (a, b) and (b, c), we would need a space (a, b, c) in order to work with the two functions simultaneously. In the sense of the formulas this can be done in the following way:

$$f = (a \vee b) = (a \vee b)(c \vee \overline{c}) \quad \text{and} \quad g = (a \vee \overline{a})(b \vee c).$$

Now both f and g depend on (a, b, c). It can be seen that $f(a, b, c = 0) = f(a, b, c = 1)$ and $g(a = 0, b, c) = g(a = 1, b, c)$. The variables c or a, respectively, are *not essential*.

Sometimes it is also desirable to find out which variables are not essential, because if this is the case, then it is possible to find a formula without this variable which makes the respective expressions shorter. This will be done most efficiently using the first derivative, after the knowledge of the Boolean Differential Calculus is available (see Chap. 4). Here we will only use the two subfunctions directly. We build, for instance, the subfunction $f(x_i = 1)$ by means of the intersection of f and the ternary vector for $x_i = 1$, the subfunction $f(x_i = 0)$ in the same way and check the equality by using the symmetric difference. The two subfunctions are

equal to each other, if the symmetric difference is equal to the empty set.

Sometimes it might even be possible to get (by using algebraic transformations of formulas or minimization algorithms) expressions without a variable; then this variable is not an essential variable. The equality of the subfunctions can be seen directly.

Exercise 2.38 (Essential Variables). Find the essential or non-essential variables of the following functions and find formulas without non-essential variables:

1 $f(x_1, x_2, x_3) = (x_1 \to (x_1 \vee x_2)) \to x_3$;

2 $f(x_1, x_2) = ((x_1 \vee x_2) \to x_2)$;

3 $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee \overline{x}_2 x_3 \vee \overline{x}_1 \overline{x}_2 \overline{x}_3) x_4$;

4 $f(x_1, x_2) = (x_1 \oplus x_2)(x_1 \downarrow x_2)$;

5 $f(x_1, x_2, x_3) = (((x_3 \to x_2) \vee x_1)(x_2 \to x_1) x_3 \overline{x}_1) \oplus x_3$;

6 $f(x_1, x_2, x_3) = (((x_1 \vee x_2)(x_1 \vee \overline{x}_3) \to \overline{x}_1) \to x_2 \overline{x}_3) x_2$;

7 $f(x_1, x_2, x_3, x_4) = (1011100111001010)$;

8 $f(x_1, x_2, x_3, x_4) = (0011110011000011)$.

5. Complete Systems of Functions

We already know that the systems consisting of *conjunction*, *disjunction* and *negation* are complete systems of functions, i.e. each function can be represented by using these functions only. This can be confirmed when we remember that each function can be represented as a *conjunctive* or *disjunctive normal form*. But it can be seen that there are also other complete systems of logic functions, sometimes consisting only of one single function.

Exercise 2.39. Show that the following systems of functions are complete systems:

1 $\{x \downarrow y\}$;

2 $\{xy \oplus z, (x \sim y) \oplus z\}$;

3 $\{x \to y, \overline{x \oplus y \oplus z}\}$;

4 $\{x \to y, (1100001100111100)\}$;

5 $\{0, xy \vee xz \vee yz, 1 \oplus x \oplus y \oplus z\}$;

6 $\{(1011), (1111110011000000)\}$.

Since the functions of the previous exercise are complete systems, it must be possible that each function can be expressed by the other functions.

Exercise 2.40. Express every function that appears in Exercise 2.39 by all the other complete systems of functions of this exercise.

Exercise 2.41. Represent every function of this exercise using all the complete systems given in Exercise 2.39.

1 $f_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$;

2 $f_2 = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) \oplus x_4 \oplus x_5$;

3 $f_3 = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)(x_4 \oplus x_5)$;

4 $f_4 = (x_1 \oplus x_2 \oplus x_3)(x_4 \oplus x_5 \oplus x_6)$;

5 $f_5 = (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)(x_1 \vee x_2 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_5 \vee \overline{x}_6)$?

6 $f_6 = \overline{x}_2 x_4 x_5 \vee x_2 \overline{x}_4 x_5 \vee (x_4 \oplus x_5)(\overline{x}_1 x_2 x_3 \vee x_1 x_2 \overline{x}_3) \vee x_1 \overline{x}_2 \overline{x}_3 (x_4 \sim x_5)$.

7 $f_7 = (x_1 \oplus x_2) \wedge (x_3 \sim x_4 \sim x_5)$.

8 $f_8 = x_1 \overline{x}_2 \vee x_2 \overline{x}_3 \vee x_3 \overline{x}_4 \vee x_4 \overline{x}_1$.

9 $f_9 = (x_1 | x_2 | x_3) \downarrow (x_4 \downarrow x_5)$.

10 $f_{10} = (x_1 \to x_2) \sim (x_2 \to x_3) \sim (x_3 \to x_4) \sim (x_4 \to x_5) \sim (x_5 \to x_1)$.

Sometimes some functions can be allowed to be used in a given context. They will not be a complete system, but they can be used to build new functions. As an example we consider the system of functions $\{0, \overline{x}\}$. It can be assumed that these two functions are given and can be used several times, according to our convenience. Which functions can be built by using these two functions?

x	$f_1(x)$	$f_2(x)$
0	0	1
1	0	0

By using the function f_2 two times, it is possible to build

$$f_3(x) = f_2(f_2(x)) = x \quad \text{and} \quad f_4(x) = f_2(f_1(x)) = 1.$$

Therefore the whole system of functions that is available consists of the functions $\{0, 1, x, \overline{x}\}$, since $f_1(f_1(x)) = f_1(f_2(x)) = 0$. Here the given

function had only one argument; when several arguments have to be considered, then the space B^n also has to be given. We will simply say that the arguments x_1, x_2, x_3 or x_1, x_2, x_3, x_4 can be used – see the next example. It can be seen later that this approach is also very typical for the design of circuits. A special function is given, and the problem has to be solved what can be done with this circuit or this gate, and how can it be done. The complete systems are particularly interesting because in this case it is known that each function can be implemented by means of this system.

Exercise 2.42. 1 Let be given the set of functions $\{1, x \oplus y\}$. Which functions can be built using these two functions for $M_1 = \{x_1, x_2, x_3\}$ and for $M_2 = \{x_1, x_2, x_3, x_4\}$.

2 Which functions can be built using $\{x \vee y, xyz\}$?

3 Which system of functions can be built by $\{x \vee y, xyz, x \vee yz, (x \vee y)z\}$?

4 Can xy be built using the set $\{0, 1, x, y, \overline{x}, \overline{y}\}$?

5 Which set can be built using the functions of the previous item?

6 Which set can be built using the given set $\{0, 1, x, y, \overline{x}, \overline{y}\}$ together with $f_1(x, y, z) = (11101000)$ or $f_2(x, y, z) = (01111111)$ or $f_3(x, y, z) = (10011001)$.

6. Partially Defined Functions

Very often the values of a function are not completely specified, they are given only for a given set of values of the argument vectors. We will see this property very often when the design of circuits or finite-state machines will be considered.

Such a function always can be seen as a ternary vector where only some of the values are given, the remaining positions are equal to – and can be replaced by values 0 or 1 (as we have seen already very often).

Exercise 2.43. Let be given the function(s) $(10 - - 010 - 11000101)$.

1 Find the set of functions that can be derived from this partial definition.

2 Find for every function the antivalence normal form and compare the different representations.

3 Find for every function the conjunctive normal form and compare the different representations.

The range of values for the variables with specified values can be described by a function $\varphi(\mathbf{x})$ which is equal to 0 for the vectors of variables

with specified values of the function and equal to 1 otherwise, i.e. for vectors with unspecified values. Later two more functions will be used in this context: the function $q(\mathbf{x})$ is equal to 1 for all vectors with $f = 1$ (the ON-set) and 0 otherwise. The function $r(\mathbf{x})$ is equal to 1 for all vectors with $f = 0$ (the OFF-set) and 0 otherwise.

Exercise 2.44. Find the function $\varphi(x_1, x_2, x_3, x_4)$ for the following definition ranges:

1 The function $f(x_1, x_2, x_3, x_4)$ is defined only for vectors with one or three values 1.

2 The function $f(x_1, x_2, x_3, x_4)$ is not defined for all vectors with two values 1.

3 The function $f(x_1, x_2, x_3, x_4)$ is not defined for the vectors (1110) and (1111).

Exercise 2.45. Let be given the function $f(x_1, x_2, x_3, x_4) = x_1\overline{x}_2 \vee x_2\overline{x}_3 \vee x_3\overline{x}_4 \vee x_4\overline{x}_1$. However, it is not defined for (1110) and (1111).

1 Find the set of four functions that meet this partial specification.

2 Find the function $\varphi(x_1, x_2, x_3, x_4)$ for this situation.

3 Represent the four possible functions by means of $f(x_1, x_2, x_3, x_4)$ and $\varphi(x_1, x_2, x_3, x_4)$.

7. Solutions

Exercise 2.1.

1

x	y	$x \vee y$	$x \leq x \vee y$	$x \wedge y$	$x \geq x \wedge y$
0	0	0	yes	0	yes
0	1	1	yes	0	yes
1	0	1	yes	0	yes
1	1	1	yes	1	yes

2 Here the same table will help: use four columns for x_1, x_2, y_1 and y_2. Mark the rows where $x_1 \leq y_1$ and $x_2 \leq y_2$ and check the relations for the conjunctions and disjunctions.

3 Use a table with the same procedure.

4 Here the transformation of $\leq$ into an equation can be used: $x \leq (yz) = x\,\overline{yz} = x(\overline{y} \vee \overline{z}) = x\overline{y} \vee x\overline{z} = 0$. The disjunction is equal to 0 if and only if both conjunctions are equal to 0: $x\,\overline{y} = 0$ and $x\,\overline{z} = 0$ which is equivalent to $x \leq y$ and $x \leq z$.

Exercise 2.2.

1 9, 13 and 50.

2 $(010011) \in B^6, (00010011) \in B^8$.

3 The vector $\mathbf{x} = (10\ldots0) \in B^n$ represents the number 2^{n-1}. The first bit with the value 1 is kept, all the other bits are successively replaced by values 1, possibly in lexicographic order: $(10\ldots00)$, $(10\ldots01)$, $(10\ldots10)$, $(10\ldots11)$ etc. This results in 2^{n-1} different numbers from 2^{n-1} until $2^n - 1$. The number 2^n itself would require one more bit, i.e. it will be represented by $\mathbf{x} = (10\ldots0) \in B^{n+1}$.

An elegant way to describe the set of all these vectors is the use of ternary vectors. We keep the 1 in the first position and replace all the values 0 by $-$: $(1--\ldots-)$ represents the set of all these vectors $\mathbf{x}$ with $2^{n-1} \leq \mathrm{dec}(\mathbf{x}) < 2^n$ since every $-$ can be replaced by 0 or 1. When only the value 0 is used, then we get the original number 2^{n-1}, when only the value 1 is selected, then the resulting number is equal to $2^n - 1$.

4 In the first case the positions with $x_i = 1$ remain unchanged. In order to increase the given vector, the values 0 can change to 1, hence, the ternary vector $(1--1-1-1)$ represents all the desired vectors. Actually we build the interval between the given vector $\mathbf{x} = (10010101)$ and the largest vector $(11\ldots11)$.

In the second case the positions with $x_i = 0$ remain unchanged, the values 1 can change to 0, hence, the ternary vector $(-00-0-0-)$ represents the desired vectors, the interval $(00000000) \leq \mathbf{y} \leq \mathbf{x}$.

5 Let $\mathbf{x} = (010101)$, $\mathbf{y}$ =(110111). Using the considerations of the previous item, all vectors $\mathbf{z}$ with $\mathbf{z} \geq \mathbf{x}$ are given by $(-1-1-1)$, the ternary vector $(--0---)$ describes the set of all vectors $\mathbf{z}$ with $\mathbf{z} \leq \mathbf{y}$. The intersection of these two sets (intervals) will be the desired set which is given by $\{(-101-1)\}$.

(a) We click on `Objects` and select `Define Space`. This is our first space, and 32 variables are sufficient because we need only six.

(b) Now we go back to `Objects` and define two TVLs as objects 1 and 2, and we assign the variables $x1, \ldots, x6$ to both of them. As `Form` predicate we select `ODA`; this is an assumption for the application of set operations.

(c) For the next step we use `Objects` again, followed by the item `Append Ternary Vector(s)`. The vector $(-1-1-1)$ will be appended to `TVL1`, the vector $(--0---)$ to `TVL2`.

(d) The final step is the intersection of these two ternary vectors which can be understood as the intersection of the respective sets of binary vectors. We select `Sets` and `ISC Intersection` and get the desired result after specifying the two operands.

Exercise 2.3.

1 The complement $\overline{\mathbf{x}}$ exchanges the values 0 and 1. Hence, if $\mathbf{x}$ has k values 1 and $n-k$ values 0, then the complement must have $n-k$ values 1 and k values 0.

2 The subtraction is necessary because some of the values 1 are counted twice on the right side.

3 This relation follows from 2 by transformation of the equation.

This norm function which is accessing the components of a (binary or ternary) vector is not available in the XBOOLE monitor. If it is required very often, then you can rely on the XBOOLE library, or you write your own C-routines to be added to the existing system.

Exercise 2.4.

1 $\mathbf{c} = (0000)$

(a) $h = 0$: $S_0 = \{(0000)\}$

(b) $h = 1$: $S_1 = \{(1000), (0100), (0010), (0001)\}$

(c) $h = 2$: $S_2 = \{(1100), (1010), (1001), (0110), (0101), (0011)\}$

(d) $h = 3$: $S_3 = \{(1110), (1101), (1011), (0111)\}$

(e) $h = 4$: $S_4 = \{(1111)\}$.

2 This problem will now start with $S_0 = \{(1111)\}$, followed by $S_1 = \{(0111), (1011), (1101), (1110)\}$ etc.

3 This can be seen immediately by inspecting the previous results.

4 $K_0 = \emptyset$, $K_i = \bigcup_{k=0}^{i-1} S_k$, $K_{n+1} = \bigcup_{k=0}^{n} S_k$.

5 $\overline{K}_0 = S_0$, $\overline{K}_i = \bigcup_{k=0}^{i} S_k$, $\overline{K}_n = \bigcup_{k=0}^{n} S_k$.

6 This follows directly from item 3, 4 and 5.

Exercise 2.5.

1 This item simply repeats the definition of the elementary functions. Ensure that you remember these functions well and use the steps mentioned above.

2 Use the sequence `Objects`, `Define Space...`, `Extras` and `Solve Boolean Equation`. Then you type the respective formulas for f_1 to f_7 and solve these equations which will give the same results.

3 $f(x, y) = \overline{x} \vee \overline{y} = \overline{x \wedge y}$. $f(x, y) = \overline{x} \wedge \overline{y} = \overline{x \vee y}$.

4 We solve the two equations $(x \oplus y) \oplus z$ and $x \oplus (y \oplus z)$ – the right sides are equal to 1 – and compare the solution sets. When we store these two solutions as two different sets, then their symmetric difference would be empty (which means that the two sets are equal).

5 When we explore, for instance, the two expressions $(x \to y) \to z$ and $x \to (y \to z)$, then the symmetric difference of the solution sets will show that the functions are different for the two vectors (000) and (011).

Exercise 2.6.

1 Instead of 2^n argument vectors of the function now only 2^{n-1} pairs of arguments have to be considered. Therefore we get $2^{2^{n-1}}$ functions with this property.

2 The construction of these functions can start at any point. We use $\mathbf{x} = (0, \ldots, 0)$ with $f(0, \ldots, 0) = 0$. Then for all vectors $\mathbf{x}$ with $\|\mathbf{x}\| = 1$ the value of the function must be equal to 1, the value for all vectors $\mathbf{x}$ with $\|\mathbf{x}\| = 2$ must be equal to 0 etc. This means that the value for $\mathbf{x} = (0, \ldots, 0)$ defines the value for all other vectors in a unique way. Since $f(0, \ldots, 0) = 1$ is the second possibility, we only have two such functions.

3 The definition of the function indicates that the value 1 can be assigned to vectors $\mathbf{x}$ with $\|\mathbf{x}\| = 0, \ldots, k-1$. It is known from combinatorics that m vectors with the value 1 can be selected from 2^n possibilities, using the binomial coefficient $\binom{2^n}{m} = \frac{2^n!}{m!(2^n-m)!}$. Therefore, the number of functions with this property is equal to $\binom{2^n}{0} + \binom{2^n}{1} + \cdots + \binom{2^n}{k-1}$.

Exercise 2.7.

1 The condition $x = 1$ can be expressed by the vector $(1--)$, $y \neq z$ defines the two possibilities $y = 0, z = 1$ and $y = 1, z = 0$, but only in the first case $x = 0$ is less than $z = 1$. Therefore we have the two ternary vectors $(1--)$ and (001) as the result. The Karnaugh map follows from the change of the view after the TVL with these two vectors has been created. $f = x \vee \overline{x}\,\overline{y}z$ is the corresponding disjunctive form.

2 The given assumption changes the point of view and introduces or uses different algebraic concepts: the addition and the calculation of $2x_4$ takes the values 0 and 1 simply as numbers, and for those vectors where the values of the variables satisfy the condition the value 0 will be entered. By checking all 16 vectors, we get the following four vectors with $f = 0$: $(0100), (1000), (1100)$ and (1110). In order to find a disjunctive form, a TVL with these four vectors can be created. The complement of this set results in a set that can be used for the disjunctive form, and we get, for instance, $f = x_4 \vee \overline{x}_1\overline{x}_2 \vee \overline{x}_1x_3 \vee \overline{x}_2x_3$.

Exercise 2.8.

1 By using the BOOLE Monitor we get $f = 1$ for the two vectors (000) and (111), hence, $f = \overline{x}\,\overline{y}\,\overline{z} \vee xyz$. It will also be a good exercise to find the disjunctive or conjunctive normal form by algebraic transformations of the given formula. Use, for instance, the following sequence of transformations:
$x \to y = \overline{x} \vee y = \overline{x} \oplus y \oplus \overline{x}y = 1 \oplus x \oplus y \oplus (1 \oplus x)y = 1 \oplus x \oplus y \oplus y \oplus xy = 1 \oplus x \oplus xy$.
We can also use the disjunctive form of this function: $f = \overline{x}\,\overline{y}\,\overline{z} \vee xyz = \overline{x}\,\overline{y}\,\overline{z} \oplus xyz = 1 \oplus x \oplus y \oplus z \oplus xy \oplus xz \oplus yz$. An equivalent conjunctive form can be found when we are using the complement of the set $\{(000), (111)\}$. After using the XBOOLE Monitor for the original formula, the solution set is stored as an object. We can also create a TVL and enter the two vectors directly. The use of the items `Sets` and `CPL Complement` results in the set of ternary vectors $\{(100), (011), (-10), (-01)\}$. The corresponding disjunctions can be used for a conjunctive form of this function: $f = (\overline{x} \vee y \vee z)(x \vee \overline{y} \vee \overline{z})(\overline{y} \vee z)(y \vee \overline{z})$. Since the set operations are always based on orthogonal sets of vectors, we get the equivalence normal form immediately: $f = (\overline{x} \vee y \vee z) \sim (x \vee \overline{y} \vee \overline{z}) \sim (\overline{y} \vee z) \sim (y \vee \overline{z})$. The conjunctive form $f = (\overline{x} \vee y)(\overline{y} \vee z)(x \vee \overline{z})$ would be shorter.

2 Since the operator $\downarrow$ has not been implemented in the XBOOLE Monitor, we must use the definition $x \downarrow y = \overline{x \vee y} = \overline{x} \wedge \overline{y}$ before the input of the equation. This is easy to do: we replace the $\downarrow$ by $\vee$ and the complement:

$$\overline{(\overline{x} \vee y) \vee (x\overline{z})} \downarrow (x \sim y) = \overline{\overline{(\overline{x} \vee y) \vee (x\overline{z})} \vee (x \sim y)}$$
$$= \overline{\overline{(\overline{x} \vee y) \vee (x\overline{z})}} \wedge \overline{(x \sim y)} = (\overline{x} \vee y \vee x\overline{z})(x \oplus y).$$

$f = 1$ for the two ternary vectors $(01-)$ and (100), hence, $f = \overline{x}y \vee x\overline{y}\,\overline{z} = (\overline{x} \vee \overline{z})(x \vee y)(\overline{x} \vee \overline{y})$ etc.

3 $f = x \vee y\overline{z} \vee \overline{y}z = (x \vee \overline{y} \vee \overline{z})(x \vee y \vee z)$.

4 Here $x \mid y = \overline{xy} = \overline{x} \vee \overline{y}$ must be used as well, and we get after some steps the expression $xy\overline{z}$ which already shows that $f = 1$ only for the vector (110).

This type of formulas is also a good possibility to use the XBOOLE Monitor, particularly the possibility of manipulating sets of ternary vectors. $\alpha|\beta$ can be calculated by creating objects for α and β, respectively, followed by the intersection of

these two objects and the complement of the intersection. The NOR-operation will be implemented by the union of the two parts and the complement of the result.

Exercise 2.9.

1 The set of ternary vectors with $f = 1$ is equal to $\{(0110), (0111), (1111)\}$ (given by the XBOOLE Monitor) or, after using the orthogonal block building, equal to $\{(011-), (1111)\}$, and this results in the disjunctive normal form $f = \overline{x}_1 x_2 x_3 \vee x_1 x_2 x_3 x_4$.

2 This expression describes the function $1(x_1, x_2, x_3, x_4)$ which is constant equal to 1 for all vectors (x_1, x_2, x_3, x_4).

3 When the expression is checked carefully, then it can be seen that this is already the conjunctive normal form of the function f. It is equal to 0 only for the two vectors (0000000000) and (1111111111). For the remaining 2^{10}–2 vectors the function would be equal to 1, hence, it would be hardly possible to write down the disjunctive or antivalence normal forms. However, the XBOOLE Monitor needs (only) 18 vectors for the representation of this function.

4 Using the definitions involved in this expression, two parts of the formula can be considered: $f_{\text{left}} = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)$ and $f_{\text{right}} = x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}$, and $f = f_{\text{left}} \oplus f_{\text{right}}$. According to the definition of the antivalence, $f = 1$ if $f_{\text{left}} = 0$ and $f_{\text{right}} = 1$ and vice versa. Therefore, the solution vectors have the vectors (000) and (111) for (x_1, x_2, x_3) together with all vectors $(x_4, x_5, x_6, x_7, x_8, x_9, x_{10})$ having an odd number of components with the value 1. All the other vectors (x_1, x_2, x_3) can be paired with those vectors $(x_4, x_5, x_6, x_7, x_8, x_9, x_{10})$ having an even number of components with the value 1. The XBOOLE Monitor shows 384 ternary vectors for the representation of this function.

Exercise 2.10.
The simplest way is the use of the set of ternary vectors that is already known followed by the correct interpretation of the vectors as disjunctions of a conjunctive normal form. We create, for instance, a list of ternary vectors with the vectors $(0110), (0111), (1111)$. The complement of this set followed by an orthogonal minimization results in three vectors $(-0--), (-10-), (1110)$, and the respective orthogonal conjunctive normal form can be written as $f = x_2(\overline{x}_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee x_4)$. The respective forms for the other functions can be found by using the same approach.

Exercise 2.11.
This more general problem uses the same ideas as the problem before. Since n is not specified, XBOOLE cannot be used immediately. However, the following considerations can be used for any special n after the value has been defined.

1 This conjunctive normal form shows that $f = 0$ for two vectors. For all the other vectors we get $f = 1$. The easiest way to get these vectors is the input of these two vectors followed by the complement.

2 For any $n \geq 4$ the two vectors (000) and (111) will be combined with all the vectors $x_4 \ldots x_n$ with an odd number of values 1, all the other vectors for (x_1, x_2, x_3) will be combined with all the vectors with an even number of values 1. The number

of vectors with $f = 1$ is equal to the number of vectors with $f = 0$, i.e. equal to 2^{n-1}.

Exercise 2.12.

1 $f = x_1x_2\overline{x}_3 = x_1x_2(1 \oplus x_3) = x_1x_2 \oplus x_1x_2x_3$;

2 $f = \overline{x}_1 = 1 \oplus x_1$;

3 $f = 1$ for the two vectors $(0--), (101)$, hence $f = \overline{x}_1 \oplus x_1\overline{x}_2x_3 = 1 \oplus x_1 \oplus x_1x_3 \oplus x_1x_2x_3$.

Exercise 2.13.

1 $f = x_1x_2x_3 + (1 - x_1)(1 - x_2)x_3 + (1 - x_1)x_2(1 - x_3) + x_1(1 - x_2)(1 - x_3) = 4x_1x_2x_3 + x_1 + x_2 + x_3 - 2x_1x_2 - 2x_1x_3 - 2x_2x_3$.
Check the value of f, for instance, for $x_1 = 1, x_2 = 1, x_3 = 1$.

2 We use `Solve Boolean Equation...` and deal with each elementary conjunction as in the previous item. In order to avoid ternary vectors, the Karnaugh map could be used.

3 Here we have two conjunctions that can be transformed as before.

Exercise 2.14.

1 The solution of this equation shows that this is a tautology.

2 We get $f = 1$ only for the vectors (000) and (011).

3 Tautology.

4 This function is always equal to 0, i.e. it is the negation of a tautology.

Exercise 2.15.

1 We use again the option of `Solve Boolean Equation`, for the first expression we get three solution vectors $(1--), (011), (000)$, for the second formula, however, we get two solution vectors $(00), (11)$. When we now check the variables then we can easily see that for the first formula the variables allocated are x, y and z, for the second, however, only x and z are used. It might be useful to know a possibility to adjust the variables in such a way that both formulas depend on the same sets of variables.

The first possibility is a bit tricky. We use the knowledge about the functions: $(x \vee \overline{x}) = 1$ and $(x \sim 1) = x$ and replace the equation $x \sim z$ by $x \sim z \sim (y \vee \overline{y})$ which obviously has the same set of solutions, but depends now on x, y and z. The resulting solution set is now equal to $\{(0 - 0), (1 - 1)\}$.

The second possibility corresponds more to the theoretical understanding of binary functions and equations. We use `Create TVL` and `Append Ternary Vector(s)`, and we append the vector $(---)$ only. This vector naturally describes the whole B^3. The intersection of this object with the solution vectors (00) and (11) considers the correct allocation of variables and results in the correct solutions $(0 - 0)$ and $(1 - 1)$.

By comparing now the two resulting solution sets, it can be seen that these two formulas are not equivalent. This can be done in the most correct way (particularly for large sets depending on many variables) by using the `Symmetric Difference`

which would be empty if the two solution sets are equal. In this example the solution sets are different, hence, the formulas are not equivalent.

2 The two formulas are not equivalent.

3 The two formulas are equivalent.

4 The two formulas are not equivalent. The two solution vectors (10−) and (100) describe different solution sets.

Exercise 2.16.
The solution of the respective equations shows that only the cases 3 and 6 cannot be used as a rule.

Exercise 2.17.

1 In the times of *cut and paste* it is the easiest way to insert the expressions for a and b (first and second argument) into the formula:

$$f = a \vee \overline{b} = (x_3 \sim x_4) \vee \overline{x}_2.$$

The resulting set of orthogonal ternary vectors is equal to $\{(0--), (100), (111)\}$ for (x_2, x_3, x_4). Later on we will deal with the application of logic equations in many different places. It might be useful to see that already for the composition of logic functions the equations can be a very useful tool. The problem shows us that the first argument a has to be equal to $x_3 \sim x_4$, hence, we write $a = (x_3 \sim x_4)$, or, in the language of the XBOOLE Monitor, $a = (x3 = x4)$. We are adding $b = x2$ and also $f = a \vee \overline{b}$, i.e. $a+!b$ as a formula of the XBOOLE Monitor. By using the topics `Extras` and `Solve Boolean Equation` of the XBOOLE Monitor we create the solution sets of these three equations, and the intersection of the three respective objects results in solution vectors for (a, b, x_2, x_3, x_4), and if we are interested only in values for x_2, x_3, x_4, then the components for a and b can be omitted, and only the different vectors for (x_2, x_3, x_4) must be used. In spite of the small example the methodology should be studied carefully, because many much larger applications can be handled in the same way.

2 Use the same approach and get successively $(1-), (00)$ for (x_1, x_2), $(00), (11)$ for (x_3, x_4) and use the union. Watch that the lists of ternary vectors have the predicate `ODA`; this can be achieved by using `Matrices – Orthogonalization`. Even when the vectors remain unchanged, still the predicate changes, and `ODA` is the assumption for the application of the set operations. The orthogonal minimization can be used directly to get simpler representations, such as $\{(1---), (00--), (0100), (0111)\}$ in this case.

3 This problem can be solved in the same way. Here the intersection of the two solution sets has to be used and results in $\{(0011), (0000), (1-00), (1-11)\}$.

Exercise 2.18.

1 Only one vector (out of 1024) satisfies this conjunction, $\mathbf{x} = (1111111111)$.

2 The TVL of the solution set shows very nicely the advantages of the orthogonal representation. Ten ternary vectors are sufficient for the representation of the solution set, each solution vector is represented by precisely one ternary vector.

3 The last two cases show the problem of representing linear functions. The antivalence as well as the equivalence expression represent two sets of 512 vectors. Each vector must be represented by a full conjunction or disjunction with all ten variables. No simplification is possible.

4 Already covered by the previous items.

Exercise 2.19.

1 By solving the respective equations directly, we get $f = z \vee x\overline{y}$ with 5 solution vectors for $f = 1$ and $g = \overline{x} \vee \overline{y} \vee z$ with 7 solution vectors for $g = 1$. By comparing the function values, it can be seen that $f < g$.

2 The XBOOLE Monitor uses the $\rightarrow$ from the left to the right.

3 The solution is given as follows:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & - & - & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & - & - & - & - & 1 & 0 & 0 & 0 & 0 \\ - & - & - & - & - & - & - & 1 & 0 & 0 \\ - & - & - & - & - & - & - & - & - & 1 \end{pmatrix}.$$

Exercise 2.20.

1 It holds that $(x|y)|z = \overline{z} \vee xyz = \overline{z} \vee xy$ whereas $x|(y|z) = \overline{x} \vee xyz = \overline{x} \vee yz$.

The transformation into the respective disjunctive forms can be based on algebraic transformations such as $\overline{x \wedge y} = \overline{x} \vee \overline{y}$, but it is also a good opportunity to get accustomed to the consideration of solution sets. The following steps must be performed, they are more or less trivial at this point of time, but a good possibility to get a better understanding for the transformation of a logic function into (solution) sets of binary vectors.

- We create three objects (TVLs) containing the vectors $(1--), (-1-), (--1)$ representing x, y and z, resp. Each TVL must be orthogonal by using `Matrices – Orthogonalization`. The matrices do not change because they contain only one vector, but they are now marked as `ODA` which is an assumption for the application of set operations.
- The sequence $1 \cap 2 \rightarrow 4$, $\overline{4} \rightarrow 5$, $5 \cap 3 \rightarrow 6$, $\overline{6} \rightarrow 7$ represents $4 = x \wedge y$, $5 = \overline{x \wedge y}$, $6 = \overline{(x \wedge y)} \wedge z$, and $\overline{6} = \overline{\overline{(x \wedge y)} \wedge z}$ is the final solution represented by the two vectors (111) and $(--0)$.

2 This problem uses the same approach for $x \downarrow y = \overline{x \vee y}$.

3 Representations like this should be avoided according to the previous considerations because they would need a definition of the order in which the operations have to be applied. It is better (more readable, easier to understand) when brackets are used for the definition of the sequence of the operations, such as $((x1 \downarrow x2) \downarrow x3) \downarrow x4$ or $(x1 \downarrow (x2 \downarrow x3)) \downarrow x4$ etc.

Exercise 2.21.

1 This function simplifies to $f_1 = x$.

2 For f_2 it is important to consider the structure of the formula in a correct way: $(x_1 \vee x_2\overline{x}_3x_4)$ is one disjunction: the next part comprises $((\overline{x}_2 \vee x_4) \rightarrow x_1\overline{x}_3\overline{x}_4)$, and these two parts must be combined by $\wedge$, or, if we have already the TVLs for the two parts, by intersection. Finally $x_2x_3 \vee \overline{x}_1 \vee x_4$ must be considered. It can also be seen from these considerations that the leftmost bracket and the closing bracket after x_2x_3 are not really necessary. The solution by using set operations or the solution by solving the equation shows that the vector (1010) for $(x_1x_2x_3x_4)$ is the only vector with $f = 1$.

3 The calculations for f_3 follow the same procedure.

Exercise 2.22.

1 The binary argument vectors for $f_1 = 1$ and $f_1 = 0$ can be translated directly into the respective conjunctions and disjunctions as we have seen before:
$f_1 = \overline{x}\,\overline{y}z \vee \overline{x}y\overline{z} \vee x\overline{y}\,\overline{z} \vee x\overline{y}z$; $f_2 = \overline{x}\,\overline{y}\,\overline{z} \vee x\overline{y}\,\overline{z} \vee x\overline{y}z \vee xy\overline{z}$;
$f_1 = (x \vee y \vee z)(x \vee \overline{y} \vee \overline{z})(\overline{x} \vee \overline{y} \vee z)(\overline{x} \vee \overline{y} \vee \overline{z})$; $f_2 = (x \vee y \vee \overline{z})(x \vee \overline{y} \vee z)(x \vee \overline{y} \vee \overline{z})(\overline{x} \vee \overline{y} \vee \overline{z})$.

2 This can be done by creating a TVL, appending the respective binary vectors and using the representation as a Karnaugh map. From here we get, for instance $f_1 = x\overline{y} \vee \overline{y}z \vee \overline{x}y\overline{z}$, $f_2 = x\overline{z} \vee x\overline{y} \vee \overline{y}\,\overline{z}$.

3 The results of the orthogonal block building are not as short as a full minimization, however, the results are already rather good and orthogonal, i.e. no double solutions have to be considered. For f_1 we get, for instance, $f_1 = \overline{x}y\overline{z} \vee x\overline{y}\,\overline{z} \vee \overline{y}z$, the result of `OBBC` is equal to $f_1 = \overline{x}y\overline{z} \vee x\overline{y} \vee \overline{x}\,\overline{y}z$ which has the same size; $f_2 = xy\overline{z} \vee \overline{y}\,\overline{z} \vee x\overline{y}z$, $f_2 = x\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z} \vee x\overline{y}z$.

Exercise 2.23.

1 The value $f = 1$ appears for argument vectors with an odd number of values 1; therefore we get 2^{n-1} vectors with the value 1, the other half has the value $f = 0$ for an even number of values 1 in the argument vector.

2 The expression for g is already in conjunctive normal form. Therefore we get the value $g = 0$ for the two vectors $(0 \ldots 0)$ and $(1 \ldots 1)$, for all the other vectors we have $g = 1$. Therefore the desired number is equal to $2^n - 2$.

3 The two disjunctions $(x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)$ result in two values 0 and six values 1. These values have to be combined with the second part of the formula; this part includes $n-3$ variables, i.e. it describes 2^{n-3} binary vectors, half of them result in 0, the other half results in 1. Therefore we get $6 \times 2^{n-4}$ values 1 and $2 \times 2^{n-4}$ values 0 for the given function.

Exercise 2.24.

1 $f = x$ as a border case contains only one variable. However, the solution set in the XBOOLE monitor is characterized as `ODA` which means *orthogonal – disjunctive or antivalence form*.

2 The result shows originally ten ternary vectors which could be replaced by the respective conjunctions and combined by $\oplus$. The orthogonal block building (minimization) results in four vectors representing $f = \overline{x}_1x_3\overline{x}_4 \oplus x_3x_4 \oplus x_1x_2x_3\overline{x}_4 \oplus \overline{x}_3$.

3 The solution of the given equation (right side equal to 1) shows that this function only has the value 1 which means that $f = 1$ is the respective antivalence form.

Exercise 2.25.

1 We replace the complemented variables $\overline{x}_i$ by $1 \oplus x_i$ and apply the distributive law as often as necessary. Two identical conjunctions can be deleted. For the second function we get, for instance, $f = 1 \oplus x_1x_3 \oplus x_1x_2x_3 \oplus x_1x_3x_4 \oplus x_1x_2x_3x_4$.

2 The solution of $x_1 \vee x_2 \vee x_3 = 1$ results in the orthogonal form $f = x_1 \vee \overline{x}_1x_2 \vee \overline{x}_1\overline{x}_2x_3 = x_1 \oplus \overline{x}_1x_2 \oplus \overline{x}_1\overline{x}_2x_3$, and from here we get $f = x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_1x_2x_3$.

Exercise 2.26.

1 The solution of the equation $f(x_1, x_2, x_3) = 1$ results in $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2$ which does not depend on x_3. Therefore, all three required subfunctions are identically equal to 0. The replacement of the respective variables by the constants and the solution of these equations will naturally show the same result.

2 The solution can easily be found by stating that in this case $f(0, 1)$ must be equal to $f(1, 0)$. The values for $f(0, 0)$ and $f(1, 1)$ are not restricted. When we are using ternary vectors, then the two vectors $(-00-)$ and $(-11-)$ for the function f describe all these functions.

Exercise 2.27.

1 Without loss of generality the representation $f(x_1, x_2, x_3) = x_1 \wedge f'(x_2, x_3)$ will be considered. For $x_1 = 0$ the function f is always equal to 0, independent on x_2 and x_3. For $x_1 = 1$ any function of two variables can be used. Hence, we have 16 functions of three variables which are conjunctively degenerated (in x_1):

x_1	x_2	x_3	f_0	f_1	...	f_{14}	f_{15}
0	0	0	0	0	...	0	0
0	0	1	0	0	...	0	0
0	1	0	0	0	...	0	0
0	1	1	0	0	...	0	0
1	0	0	0	0	...	1	1
1	0	1	0	0	...	1	1
1	1	0	0	0	...	1	1
1	1	1	0	1	...	0	1

Using some of the functions of two variables, we get, for instance, $f_0 = x_1 \wedge 0$, $f_1 = x_1(x_2x_3)$, $f_{14} = x_1(\overline{x}_2 \vee \overline{x}_3)$, $f_{15} = x_1 \wedge 1$. The characteristic property is the value 0 for f if $x_1 = 0$. There are 256 functions of three variables, only 16 are conjunctively degenerated if x_1 is the selected variable.

2 The construction of disjunctively degenerated functions follows the same idea. Only the role of 0 and 1 has to be exchanged.

3 For the construction of linearly dependent functions the representation

$$f(x_1, x_2, x_3) = x_1 \oplus f'(x_2, x_3)$$

will be used. For $x_1 = 0$ any function $f'(x_2, x_3)$ can be used; $x_1 = 1$ results in $f(x_1, x_2, x_3) = 1 \oplus f'(x_2, x_3) = \overline{f'(x_2, x_3)}$. By using subfunctions, this property

can be expressed by the following equation: $f(0, x_2, x_3) = \overline{f(1, x_2, x_3)}$. You can check, for instance, that for $f'(x_2, x_3) = x_2x_3$ you get $\overline{x_2x_3} = \overline{x}_2 \vee \overline{x}_3$, and this results in $f = x_1 \oplus x_2x_3$.

For four variables the same considerations can be applied: $f(x_1, x_2, x_3, x_4) = x_1 \oplus f'(x_2, x_3, x_4)$. It can also be seen that each linear function is linearly degenerated in each variable that appears in the formula. The same considerations can also be applied to the equivalence.

Generally a test of this property can use the relation

$$f(x_1, \ldots, x_i = 0, \ldots, x_n) = \overline{f(x_1, \ldots, x_i = 1, \ldots, x_n)}$$

directly. The easiest way to do this is the checking of the equality:

- create the TVL for f, intersect (ISC) with $x_i = 0 \Rightarrow$ object 1;
- create the TVL for f, intersect (ISC) with $x_i = 1 \Rightarrow$ object 2;
- CPL (complement) of object 2 $\Rightarrow$ object 3;
- symmetric difference SYD of object 1 and object 3 must be equal to $\emptyset$.

4 The solution of this problem uses the same approach. Watch the different meaning of 0, 1 and the equivalence.

Exercise 2.28.
For the given function vectors use the reverse of the vector and the following negation. When we are using $f_4 = x_1x_2 \vee \overline{x}_2(x_3 \oplus x_4)$, then we can transform the formula into $f_4 = x_1x_2 \vee \overline{x}_2\overline{x}_3x_4 \vee \overline{x}_2x_3\overline{x}_4$, and the mutual exchange of $\wedge$ and $\vee$ results in $f_4{}^* = (x_1 \vee x_2)(\overline{x}_2 \vee \overline{x}_3 \vee x_4)(\overline{x}_2 \vee x_3 \vee \overline{x}_4)$.

Exercise 2.29.

1 The definition of a symmetric function results for x_1 and x_2 in $f(0, 1, x_3) = f(1, 0, x_3)$, or by inserting values for x_3, in

$$f(0, 1, 0) = f(1, 0, 0), \quad f(0, 1, 1) = f(1, 0, 1).$$

2 For x_2 and x_3 we get in the same way $f(x_1, 0, 1) = f(x_1, 1, 0)$, or by inserting values for x_1, in

$$f(0, 0, 1) = f(0, 1, 0), \quad f(1, 0, 1) = f(1, 1, 0).$$

3 This means that the function which has both properties is constant for argument vectors with the same weight. Therefore, for three variables we do not have eight degrees of freedom, but only four, as can be seen from the following table:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	α_1
0	0	1	α_2
0	1	0	α_2
0	1	1	α_3
1	0	0	α_2
1	0	1	α_3
1	1	0	α_3
1	1	1	α_4

The parameters α_1, α_2, α_3, α_4, can take any value, therefore we have 16 such functions.

Exercise 2.30.

1 The property of symmetry reduces the degrees of freedom from four to three since $f(0,1) = f(1,0)$ is required. Hence, the number of functions with this property is equal to $2^{\frac{3}{4}2^n} = 2^{3\cdot 2^{n-2}}$.

2 The invariance against all permutations generalizes the property we already have seen for three variables. For each vector with a given weight the function must be constant (either equal to 0 or equal to 1). Since there are weights from 0 to n, their number is equal to 2^{n+1}.

Exercise 2.31.

1 By solving the equation, we get the two vectors (-00) and $(--1)$ as solution vectors. By using the Karnaugh-map it can be seen that $f_1(000) = 1$, but $f_1(010) = 0$, therefore the function is not monotonely increasing. It can also not be monotonely decreasing, because $f_1(111) = 1$ and $f_1(110) = 0$, but $f_1(100) = 1$.

2 We have $f_2(1111) = 0$, but $f_2(x_1, x_2, x_3, x_4) = 1$ for several smaller vectors. Hence, $f_2(x_1, x_2, x_3, x_4)$ is not monotone.

3 $f_3(x_1, x_2, x_3, x_4)$ is not monotone because of $f_3(1000) = 1$ and $f_3(1001) = 0$.

4 $f_4(x_1, x_2, x_3, x_4)$ is monotone. Take, for instance, the Karnaugh-map for this function and draw the graph of the partial order from (0000) to (1111) with the respective values of the function.

Exercise 2.32.

1 We take $n \geq 2$. Then we have the value 0 for f_1 for all vectors with one component equal to 1 and for $(000\ldots000)$. These vectors are the two lowest levels of the graph of the partial order, all the other values of f_2 are equal to 1, therefore f_1 is monotone.

2 We transform the given formula: $f_1 = \overline{x}_1 \vee \overline{x}_2 \vee \cdots \overline{x}_{n-1} \vee \overline{x}_n \vee (x_1 \oplus x_2 \oplus \cdots \oplus x_n)$. The value of the disjunction of the complemented variables is 0 if and only if all the values of $x_1 \ldots x_n$ are equal to 1. Now it depends: if n is odd, then we get $f_2(x_1, \ldots, x_n)$ is constant equal to 1 because then the antivalence is equal to 1. If n is even, then the value of $f(1, 1, \ldots, 1, 1) = 0$, and the function is monotonely decreasing.

Exercise 2.33.

1 We remember the theorem that monotone functions have a disjunctive (normal) form without negated variables. We can split the set of all conjunctions of this normal form into two orthogonal subsets: one subset with conjunctions with the variable x_i, the other subset with all conjunctions without this variable. The inverse use of the distributive law transforms the disjunctive form already into the desired format.

2 This is the equivalent format derived by transformation into the equivalent conjunctive form.

Exercise 2.34.

1 We can use the properties of monotone functions in the following way: if there is the value 1 for a given vector, then all larger vectors must also have the value 1. Therefore, $f(0,1,1,0) = 1$ means that $f = 1$ also for the vectors (1110), (0111) and (1111). It can be derived from $f(1,1,0,0) = 1$ that $f = 1$ also for the vectors (1110), (1101) and (1111). On the other side the value 0 for a given vector implies this value for all smaller vectors. Therefore, $f(1,0,1,0) = 0$ means that $f = 0$ also for the vectors (0010), (1000) and (0000). For the last two values we get $f = 1$ for the vectors (1011), (0111) and (1111) and $f = 0$ for the vectors (0001), (0100) and (0000). When we put all these values together, then it can be seen that only the value $f(1,0,0,1)$ has not been defined. The graph of the function shows that it can be set to 0 or to 1, in both cases we get a monotone function.

2 See the previous item.

3 The creation of the two respective TVLs and minimization show the respective disjunctive forms. When we set, for instance, $f(1,0,0,1) = 0$, then we get $f(x_1,x_2,x_3,x_4) = x_1x_2 \vee x_2x_3 \vee x_1x_3x_4$.

Exercise 2.35.

1 This constraint is not very demanding. Two values of a function f have been set. The remaining 14 positions of the vector of the function values (16 bits) can be set arbitrarily (which results in 2^{14} different possibilities).

2 Here the value 1 is given for the vector (1000). Take, for instance, the nice view that the values of a linear function look like a chessboard, then you will see that this setting defines $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$. the definition $f(1,0,0,0) = 0$ would result in the complement $f = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

3 As an example we explore the symmetry with regard to (x_1,x_2) (other symmetries can be checked in the same way). The problem requires that $f(0,1,0,0) \neq f(1,0,1,1)$. This will be achieved by setting $f(0,1,0,0) = \alpha$, $f(1,0,1,1) = \overline{\alpha}$. Because of the symmetry we get two more values: $f(1,0,0,0) = \alpha$ and $f(0,1,1,1) = \overline{\alpha}$. The symmetry implies two more equalities: $f(0,1,0,1) = f(1,0,0,1) = \gamma$, $f(0,1,1,0) = f(1,0,1,0) = \delta$. Up to now there are three parameters α, β and γ in the solution which results in 2^3 possibilities. These eight possibilities can be combined with the 2^8 possibilities that exist for $(x_1,x_2) = (00)$ and $(x_1,x_2) = (11)$. Therefore we get 2^{11} different solutions of this problem.

4 The condition $f(1,0,0,1) = 0$ implies for a self-dual function that $f(0,1,1,0) = 1$. For the remaining seven pairs of vectors always the setting $f(x_1,x_2,x_3,x_4) = \overline{f(\overline{x}_1,\overline{x}_2,\overline{x}_3,\overline{x}_4)}$ has to be used.

Exercise 2.36.

1 See item 3.

2 See item 3.

3 We will mention here a special possibility that will be based on the properties of the XBOOLE Monitor. Instead of starting with the elementary conjunctions and a comprehensive use of the Method of Blake we create a TVL and enter the respective vectors with the value 1; for the third problem we would enter

$$(0000), (0010), (0100), (0101), (0110), (1001), (1011), (1100), (1101), (1110).$$

Then do not forget to use the topic `Matrices` and there the item `Orthogonalization`. This will not change the TVL itself, but the predicate of the list from D to ODA. And now OBB `Orthogonal Block Building` or `OBBC Orthogonal Block Building and Change` can be used. In this case it will not really make a difference. We get $(00-0), (10-1), (-110), (-10-)$, and the Method of Blake can start here, and the procedure is much shorter and results in $f = \overline{x}_1\overline{x}_4 \vee x_2\overline{x}_3 \vee x_2\overline{x}_4 \vee x_1\overline{x}_2x_4$.

Exercise 2.37.

1 The input of this expression and the finding of the values for $x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 1$ shows 16 Vectors, i.e. 16 conjunctions, and no further simplification is possible. This is a very typical property of *linear functions*.

2 The use of the XBOOLE Monitor shows 12 ternary vectors, the orthogonal minimization does not change the number of ternary vectors. Again the Method of BLAKE will show that there are no further simplifications. Check it out!

Here would also be a possibility to check whether a conjunction C is a prime implicant or not. The conjunction C is a prime implicant if $C \vee f = f$. Now we take, for instance, $C = \overline{x}_1x_3x_4x_5$ and omit x_3 which results in $C = \overline{x}_1x_4x_5$. In order to check the relation, we create a TVL with $(0--11)$ as the only vector, change the predicate to ODA (using orthogonalization for `Matrices`) and perform the union of f and this single vector. The sets for f and $f \vee C$ can be compared by using the symmetric difference. This difference must be equal to $\emptyset$ if the equality holds which is not the case for the situation considered. Therefore $C = \overline{x}_1x_4x_5$ is not a prime implicant.

The general idea can be understood as follows: the omission of one variable creates larger intervals with the value 1 for the function. In order to make this shorter conjunction an implicant and even more a prime implicant f must not be extended (falsified), the relation $C \leq f$ must still hold.

3 Here we get 8 solution vectors, no simplifications.

4 The solution vectors will have an even number of values 1, because $x_1 \oplus x_2 \oplus x_3 = 1$ only holds for an odd number of values 1, and the same holds for $x_4 \oplus x_5 \oplus x_6 = 1$, and the addition of the odd number gives an even number (2, 4 or 6).

5 The homogeneous orthogonal structure of the XBOOLE solution can be simplified and results in $f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 \vee x_2 \vee x_3 \vee x_4\overline{x}_5 \vee x_4\overline{x}_6 \vee \overline{x}_4x_5 \vee \overline{x}_4x_6 \vee \overline{x}_5x_6 \vee x_5\overline{x}_6$.

Exercise 2.38.

The solutions of these problems is not difficult, however, they should be studied carefully to get a good understanding of the underlying concepts.

1 $(x_1 \to (x_1 \vee x_2) \to x_3 = (\overline{x}_1 \vee x_1 \vee x_2) \to x_3 = 1 \to x_3 = 0 \vee x_3 = x_3$.

2 $(x_1 \vee x_2) \to x_2 = \overline{x}_1\overline{x}_2 \vee x_2 = \overline{x}_1 \vee x_2$.

3 This is a good possibility to see the application of BLAKE's rule. After some steps it can be seen that the expression in brackets is identically equal to 1; therefore we have $f = x_4$.

4 f is identically equal to 0.

5 The use of the XBOOLE monitor for this function shows directly the result $f(x_1, x_2, x_3) = x_3$, therefore x_1 and x_2 are non-essential variables. The sequence

of transformations shows the same result: $f(x_1, x_2, x_3) = (((x_3 \to x_2) \vee x_1)(x_2 \to x_1)x_3\overline{x}_1) \oplus x_3 = (x_1 \vee x_2 \vee \overline{x}_3)(x_1 \vee \overline{x}_2)\overline{x}_1 x_3) \oplus x_3 = x_3$. Another possibility is the checking of subfunctions; we insert the values $x_1 = 0$ and $x_1 = 1$ and compare the resulting subfunctions. In the same way we check the variables x_2 and x_3 and get finally the same result.

6 The same considerations result in $f(x_1, x_2, x_3) = x_1x_2 \vee x_2\overline{x}_3$, and the consideration of the respective subfunctions shows that all three variables are essential.

7 In order to find a solution for this problem, variables have to be assigned to the components of the vector with the values of the function. One possible coding is shown by the following representation:

f	1	0	1	1	1	0	0	1	1	1	0	0	1	0	1	0
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Now there are several possibilities. It would be, for instance, possible to enter the argument vectors with $f = 1$ into a list of ternary vectors, and to get a shorter description by using OBB or OBBC. Thereafter, we could again build the subfunctions $f(x_1 = 0)$ and $f(x_1 = 1)$, etc., and compare these subfunctions by means of the *symmetric difference SYD*.

But there is also another possibility for this kind of representation. In the first and second column, we have $x_2 = 0, x_3 = 0, x_4 = 0$, and x_1 changes its value from 0 to 1. The function value of f also changes from 1 to 0 which means that $f(x_1 = 1)$ cannot be equal to $f(x_1 = 0)$, and this means naturally that x_1 is an essential variable. The same can be seen by comparing the columns 2 and 4 (counted from the left) with regard to x_2, by comparing the columns 3 and 7 with regard to x_3, and finally by comparing the columns 3 and 11 with regard to x_4. All the variables are essential variables. This method is successfully working for small numbers of variables only, however, it shows very well the essence of these concepts.

8 This item can be dealt with in the same way as the previous one.

Exercise 2.39.
It is general practice to show that the given new functions are able to build other functions that are already known as contributing to a complete system of functions. Her we assume the knowledge that $\overline{x}$, $x \wedge y$ and $x \vee y$ are sufficient to implement any function (see, for instance, the concepts of *disjunctive* and *conjunctive normal form*).

1 In the first case the function $f(x, y) = x \downarrow y = \overline{x \vee y}$ will be used to implement these three functions:

$$x \downarrow x = \overline{x \vee x} = \overline{x},$$

$$(x \downarrow y) \downarrow (x \downarrow y) = \overline{\overline{x \vee y} \vee \overline{x \vee y}} = (x \vee y) \wedge (x \vee y) = x \vee y,$$

$$(x \downarrow x) \downarrow (y \downarrow y) = \overline{\overline{x \vee x} \vee \overline{y \vee y}} = (x \vee x) \wedge (y \vee y) = x \wedge y.$$

When the expression $h(x,y) = x \downarrow y$ is used, then the three functions can be expressed by using the function $h(x,y)$ alone:

$$\begin{aligned}\overline{x} &= h(x,x),\\ x \vee y &= h(h(x,y),h(x,y)),\\ x \wedge y &= h(h(x,x),h(y,y)).\end{aligned}$$

Only the function $h(x,y)$ has been used.

2 We use the same strategy: $h_1(x,y,z) = xy \oplus z$, $h_1(x,x,z) = x \oplus z$, $h_2(x,y,z) = (x \sim y) \oplus z = x \oplus y \oplus z \oplus 1$, $h_2(x,h_1(x,x,z),z) = 1$, $h_1(x,y,h_2(x,h_1(x,x,z),z)) = xy \oplus 1 = \overline{xy} = x|y$. This function is the NAND-function which itself is a complete system. Please, watch that only applications of h_1 and h_2 have been used.

3 We use $(x \to y) \to y = x \vee y$ and $\overline{x \oplus x \oplus x} = \overline{x}$, and by combination of these two functions $x \downarrow y$ can be implemented which has been explored before.

4 We use again $(x \to y) \to y = x \vee y$ and find a formula for the given vector of a function. When the variables x_1, x_2, x_3, x_4 are used together with the vectors (0000) to (1111) from the left to the right, then we get $f(x_1,x_2,x_3,x_4) = \overline{x}_1\,\overline{x}_2\,\overline{x}_3 \vee \overline{x}_1 x_2 x_3 \vee x_1 x_2 \overline{x}_3 \vee x_1 \overline{x}_2 x_3$. This function is, in fact, independent on x_4, and therefore the function $f(x_1,x_1,x_1) = \overline{x}_1$ can be used. Based on the considerations of the first item, the disjunction and the negation can be used to build the NOR-function which is complete.

5 For $f_1(x,y,z) = 0, f_2(x,y,z) = xy \vee xz \vee yz$ and $f_3(x,y,z) = 1 \oplus x \oplus y \oplus z$ we find $f_2(x,y,0) = xy$ and $f_3(x,x,x) = \overline{x}$ which can be combined to build the NAND-function.

6 In this case we get $f_1(x_1,x_2) = x_1 \vee \overline{x}_2$ which can be used to build the function 1 by using $f_1(x_1,x_1) = 1$. The second function is given by $f_2(x_1,x_2,x_3,x_4) = \overline{x}_1\overline{x}_2 \vee \overline{x}_1\overline{x}_3 \vee \overline{x}_2\overline{x}_3$, and this function produces the NOR-function when the value 1 is inserted: $f_2(x_1,x_2,1) = \overline{x}_1\overline{x}_2 = \overline{x_1 \vee x_2}$.

Exercise 2.40.
As an introduction we show the implemention of $x \oplus y$ by means of $x \downarrow y$.

$$\begin{aligned}\overline{x} &= \overline{x \vee x} = (x \downarrow x),\\ \overline{y} &= \overline{y \vee y} = (y \downarrow y),\\ f_1 &= x \wedge \overline{y} = \{(x \downarrow x) \downarrow [(y \downarrow y) \downarrow (y \downarrow y)]\},\\ f_2 &= \overline{x} \wedge y = \{[(x \downarrow x) \downarrow (x \downarrow x)] \downarrow (y \downarrow y)\},\\ x \oplus y &= f_1 \vee f_2 = (f_1 \downarrow f_2) \downarrow (f_1 \downarrow f_2).\end{aligned}$$

The solution of problems like this is not supported by XBOOLE. If they occur very often, then it would be advisable to write a special program. Sometimes it might be helpful to see that the left side of such a representation is equal to the right side – problems of this kind already have been solved very often.

When we need the constant functions $0(x)$ and $1(x)$, then this can be achieved by using $x \vee \overline{x} = 1(x)$ and $x \wedge \overline{x} = 0(x)$:

$$\begin{aligned}0(x) &= (x \downarrow x) \downarrow ((x \downarrow x) \downarrow (x \downarrow x)),\\ 1(x) &= (x \downarrow (x \downarrow x)) \downarrow (x \downarrow (x \downarrow x)).\end{aligned}$$

Finally we can build the NAND by using NOR alone as follows:

$$x \wedge y = [(x \downarrow x) \downarrow (y \downarrow y)],$$
$$x|y = \overline{x \wedge y} = [(x \downarrow x) \downarrow (y \downarrow y)] \downarrow [(x \downarrow x) \downarrow (y \downarrow y)].$$

Several applications of the given equalities are sufficient to implement all the different functions of the previous exercise.

Exercise 2.41.
The solution of this problem uses the same principles that have been used in the previous item.

Exercise 2.42.

1 We set $h_1(x, y) = x \oplus y$, $h_2(x, y, z) = 1$ and get successively:

$$h_1(x_1, x_1) = 0, \quad h_1(x_1, h_2(x_1, x_2, x_3)) = x_1 \oplus 1 = \overline{x}_1, \quad h_1(x_1, 0) = x_1,$$
$$h_1(x_1, x_2) = x_1 \oplus x_2, \quad h_1(x_1, \overline{x}_2) = x_1 \oplus \overline{x}_2,$$
$$h_1(x_1, (x_2 \oplus x_3)) = x_1 \oplus x_2 \oplus x_3, \quad h_1(\overline{x}_1, (x_2 \oplus x_3)) = \overline{x}_1 \oplus x_2 \oplus x_3.$$

Some functions that can be built by only changing the name of a variable have not been mentioned explicitly, it is assumed and quite understandable that they can be found in the same way (see, for instance $h_1(x_2, x_3) = x_2 \oplus x_3$). In each step we only used the given function(s) and functions that have been constructed in previous steps.

When the set of variables is extended and the set $\{x_1, x_2, x_3, x_4\}$ will be used, then all the functions that have been constructed before can be built again; additionally we get the functions $h(x_1 \oplus x_2, x_3 \oplus x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ and $h(\overline{x}_1 \oplus x_2, x_3 \oplus x_4) = \overline{x}_1 \oplus x_2 \oplus x_3 \oplus x_4$.

2 It can be seen immediately that single variables can be generated by $x_1 \vee x_1 = x_1$ (the same for x_2 and x_3), the disjunction of two variables as well, such as $x_1 \vee x_2$ and also $x_1 \vee x_2 \vee x_3$. The conjunction allows additionally the generation of $x_1 x_2$ (as well as $x_1 x_3$ and $x_2 x_3$) and $x_1 x_2 x_3$. Finally the combination of the conjunctions and disjunctions allows to build $x_1 \vee x_2 x_3$, $x_2 \vee x_1 x_3$, $x_3 \vee x_1 x_2$, $x_1 x_2 \vee x_1 x_3$, $x_1 x_2 \vee x_2 x_3$, $x_1 x_3 \vee x_2 x_3$ and finally $x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$. All these functions are monotone functions. It is not possible to get the constant functions 0 and 1, complemented variables also cannot be achieved. The consideration of four variables follows the same ideas.

3 Here we use the same considerations as in the previous item.

4 By means of the given functions the conjunction cannot be implemented.

5 The application of the given functions will not produce new functions. There is no possibility to extend the given set of functions.

6 We find for the given function vectors $f_1(x, y, z) = \overline{x}\,\overline{y} \vee \overline{x}\,\overline{z} \vee \overline{y}\,\overline{z}$, $f_2(x, y, z) = x \vee y \vee z$ and $f_3(x, y, z) = xz \vee \overline{y}\,\overline{z}$. Now we follow the previous approach: We get, for instance, single negated variables as follows: $f(x_1, 0, 1) = \overline{x}$. Now the negation can be used to get the single variables x_1, x_2, x_3, and from here several disjunctive forms can be built. The two other functions and the extension to four variables follow the same ideas.

As we mentioned before, the XBOOLE Monitor can be used to check the correctness of some of the implemented relations.

Exercise 2.43.
We will use the variables x_1, x_2, x_3, x_4 and start with the vector (0000) from the left. This means that the function values are not defined for the vectors (0010), (0011) and (0111).

1 There are eight functions that can be derived from this incomplete specification. We replace successively the components with a – by all possible combinations of 0 and 1. The XBOOLE Monitor can be used as a tool in the following way: we define a TVL with the vectors (0000), (0101), (1000), (1001), (1101), (1111). This is the basic structure for the following steps and describes already the function generated by using the value 0 for the free components. Thereafter we are adding to this list the vectors describing additional values 1: if, for instance, the value 1 is used for the vector (0011), then this vector is added to the original TVL. After the building of these eight TVL we change the predicate of the list by ORTH and reduce the size of the lists by an orthogonal block building OBB. This finally results in the following TVLs for the eight functions:

$$f_0(x_1,x_2,x_3,x_4) = \begin{pmatrix} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1 \end{pmatrix} \longrightarrow \begin{pmatrix} -&0&0&0\\ 1&0&0&1\\ -&1&0&1\\ 1&1&1&1 \end{pmatrix},$$

$$f_1(x_1,x_2,x_3,x_4) = \begin{pmatrix} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&1&1&1 \end{pmatrix} \longrightarrow \begin{pmatrix} -&0&0&0\\ 1&0&0&1\\ -&1&-&1 \end{pmatrix},$$

$$f_1(x_1,x_2,x_3,x_4) = x_2x_4 \vee \overline{x}_2\overline{x}_3\overline{x}_4 \vee x_1\overline{x}_2\overline{x}_3x_4 = x_2x_4 \oplus \overline{x}_2\overline{x}_3\overline{x}_4 \oplus x_1\overline{x}_2\overline{x}_3x_4.$$

This is the function with the shortest orthogonal representation. This shows clearly that partially defined functions offer many optimization possibilities.

$$f_2(x_1,x_2,x_3,x_4) = \begin{pmatrix} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&0&1&1 \end{pmatrix} \longrightarrow \begin{pmatrix} -&0&0&0\\ 1&0&0&1\\ -&1&0&1\\ 1&1&1&1\\ 0&0&1&1 \end{pmatrix},$$

$$f_3(x_1,x_2,x_3,x_4) = \begin{pmatrix} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&0&1&0 \end{pmatrix} \longrightarrow \begin{pmatrix} -&0&0&0\\ 1&0&0&1\\ -&1&0&1\\ 1&1&1&1\\ 0&0&1&0 \end{pmatrix},$$

$$f_4(x_1,x_2,x_3,x_4) = \left(\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&1&1&1\\ 0&0&1&1 \end{array}\right) \longrightarrow \left(\begin{array}{cccc} -&0&0&0\\ 1&0&0&1\\ 0&0&1&1\\ -&1&-&1 \end{array}\right),$$

$$f_5(x_1,x_2,x_3,x_4) = \left(\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&1&1&1\\ 0&0&1&0 \end{array}\right) \longrightarrow \left(\begin{array}{cccc} -&0&0&0\\ 1&0&0&1\\ 0&0&1&0\\ -&1&-&1 \end{array}\right),$$

$$f_6(x_1,x_2,x_3,x_4) = \left(\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&0&1&1\\ 0&0&1&0 \end{array}\right) \longrightarrow \left(\begin{array}{cccc} -&0&0&0\\ 1&0&0&1\\ -&1&0&1\\ 1&1&1&1\\ 0&0&1&- \end{array}\right),$$

$$f_7(x_1,x_2,x_3,x_4) = \left(\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&1&1&0\\ 0&0&1&1\\ 0&0&1&0 \end{array}\right) \longrightarrow \left(\begin{array}{cccc} -&0&0&0\\ 1&0&0&1\\ -&1&-&1\\ 0&0&1&- \end{array}\right).$$

2 Since the representation consists of orthogonal vectors (i.e. of orthogonal conjunctions), the antivalence and the disjunctive forms can be derived immediately. Further minimization possibilities have to be explored.

3 In order to find the conjunctive normal forms, we can use the vectors with $f = 0$ and build the respective disjunctions. The easiest way is the use of the previous matrix representations together with the complement, such us

$$\left(\begin{array}{cccc} 0&0&0&0\\ 0&1&0&1\\ 1&0&0&0\\ 1&0&0&1\\ 1&1&0&1\\ 1&1&1&1\\ \hline 0&1&1&1 \end{array}\right) \longrightarrow \left(\begin{array}{cccc} -&1&0&0\\ 0&0&0&1\\ -&-&1&0\\ -&0&1&1 \end{array}\right),$$

$$f_1(x_1,x_2,x_3,x_4) = (\overline{x}_2 \vee x_3 \vee x_4)(x_1 \vee x_2 \vee x_3 \vee \overline{x}_4)(\overline{x}_3 \vee x_4)(x_2 \vee \overline{x}_3 \vee \overline{x}_4).$$

Exercise 2.44.
The function $\varphi(x_1,x_2,x_3,x_4)$ will be equal to 1 for the vectors with defined values and equal to 0 for the vectors with undefined values.

1 $\varphi(x_1,x_2,x_3,x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

2 $\varphi(x_1,x_2,x_3,x_4) = 0$ for the vectors (0011), (0101), (1001), (0110), (1010) and (1100). Now there are two possibilities: we use these six vectors to build a conjunctive form; in this way we get a conjunctive form with six disjunctions of four variables. The other possibility would be the use of these vectors as a set, build the complement and try to simplify the result as a disjunctive form.

3 We get $\varphi(x_1,x_2,x_3,x_4) = \overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3$.

Exercise 2.45.

1 The function is equal to 0 only for the vector (0000) and not defined for the two vectors (1110) and (1111). Therefore the conjunctive form is very appropriate to represent the four different functions. When we replace the two empty positions by the four possibilities 00, 01, 10 and 11, then we get successively

$$\begin{aligned}
f_1(x_1,x_2,x_3,x_4) &= (x_1 \vee x_2 \vee x_3 \vee x_4),\\
f_2(x_1,x_2,x_3,x_4) &= (x_1 \vee x_2 \vee x_3 \vee x_4)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee x_4),\\
f_3(x_1,x_2,x_3,x_4) &= (x_1 \vee x_2 \vee x_3 \vee x_4)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4),\\
f_4(x_1,x_2,x_3,x_4) &= (x_1 \vee x_2 \vee x_3 \vee x_4)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3).
\end{aligned}$$

2 $\varphi(x_1,x_2,x_3,x_4) = \overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3$.

3 We get four functions f^* by using the relation

$$\begin{aligned}
&f^*(x_1,x_2,x_3,x_4)\\
&\quad = f(x_1,x_2,x_3,x_4)\varphi(x_1,x_2,x_3,x_4) \vee g(x_1,x_2,x_3,x_4)\overline{\varphi(x_1,x_2,x_3,x_4)};
\end{aligned}$$

$g(x_1,x_2,x_3,x_4)$ can be any function of four variables. Therefore many choices will result in the same function $f^*(x_1,x_2,x_3,x_4)$. Only different values for the two vectors (1110) and (1110) are important.

Chapter 3

LOGIC EQUATIONS

1. Logic Equations

This chapter deals with the most important concept of *Logic Equations* which is a core topic of the present considerations. Many problems can be solved by means of *Logic Equations* in a very elegant and also "very easy" way. XBOOLE and the XBOOLE Monitor are very dedicated to this concept. Therefore it is quite useful to study Chap. 3 in [18] as well as the topics `Sets` and `Matrices` of the XBOOLE Monitor. The knowledge of these two topics makes the solution of Logic Equations more or less very easy. Because the concept of Boolean equations is applied more or less everywhere throughout the book, we will restrict our considerations in this chapter to some basic considerations.

In the item `Extras` we find among others the item `Solve Boolean Equation...`. This item already has been used in the previous chapter very often in order to define the values of logic functions that have been given by logic formulas. The use of this item as an instrument for solving Boolean equations simply is caused by the fact that the values 1 of a logic function $f(\mathbf{x})$ can be described as the solution set of the equation $f(\mathbf{x}) = 1$.

Our main point in this chapter will be the extensive use of set operations for the building of solution sets for larger equations or systems of equations based on the solution sets of smaller parts. The reasoning behind this approach is naturally the *isomorphism* between the Boolean Algebra of logic functions and the Boolean Algebra of sets of vectors related to these functions.

Let us remember the following relations between logic functions and the respective sets of vectors: let f_0 and f_1 be the (solution) sets of

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

(binary or ternary) vectors $\mathbf{x}$ with $f(\mathbf{x}) = 0$ and $f(\mathbf{x}) = 1$, respectively. These elementary solution sets can be found or given by computations based on formulas or the input of ternary vectors.

Then we have immediately

$$f_0 = \overline{f_1}, \quad f_1 = \overline{f_0}.$$

The complement allows to switch between these two sets.

Now we consider two functions $f(\mathbf{x})$ and $g(\mathbf{x})$ with the elementary solution sets f_0, f_1, g_0 and g_1. The two most important relations are dealing with the equations

$$h_0(\mathbf{x}) = f(\mathbf{x}) \vee g(\mathbf{x}) = 0 \quad \text{and} \quad h_1(\mathbf{x}) = f(\mathbf{x}) \wedge g(\mathbf{x}) = 1.$$

The first equation is satisfied by all $\mathbf{x}$ with $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = 0$ which results in the solution set $f_0 \cap g_0$; the solution set of the second equation is equal to $f_1 \cap g_1$.

The equation $h_0(\mathbf{x}) = f(\mathbf{x}) \wedge g(\mathbf{x}) = 0$ has the solution set $f_0 \cup g_0$, because the conjunction is equal to 0 if at least one of the elements is equal to 0. The same set operations must be used for the disjunction of functions in a characteristic equation. Hence, the equation $h_1(\mathbf{x}) = f(\mathbf{x}) \vee g(\mathbf{x}) = 1$ has the solution set $f_1 \cup g_1$, because the disjunction is equal to 1 if at least one of the elements is equal to 1. The *union (UNI)*, the *difference (DIF)*, and the *intersection (ISC)* as well as the *complement (CPL)* have been directly implemented in the XBOOLE Monitor.

Let us now have a look at the equation

$$h(\mathbf{x}) = f(\mathbf{x}) \oplus g(\mathbf{x}) = 1.$$

According to the definition of the antivalence this equation is satisfied by vectors $\mathbf{x}$ with $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = 1$ and $f(\mathbf{x}) = 1$, $g(\mathbf{x}) = 0$. This results in a solution set

$$(f_0 \cap g_1) \cup (f_1 \cap g_0) = (f_0 \cap \overline{g}_0) \cup (\overline{f}_0 \cap g_0) = (f_0 \setminus g_0) \cup (g_0 \setminus f_0) = f_0 \triangle g_0$$

that can be calculated alternatively by

$$(f_0 \cap g_1) \cup (f_1 \cap g_0) = (\overline{f}_1 \cap g_1) \cup (f_1 \cap \overline{g}_1) = (g_1 \setminus f_1) \cup (f_1 \setminus g_1) = f_1 \triangle g_1.$$

The solution set of $f(\mathbf{x}) \oplus g(\mathbf{x}) = 0$ is the complement of this set. Therefore the *symmetric difference (SYD)* indicated by *triangle* can be used to deal with the solution of antivalence equations. If the symmetric difference is equal to the empty set then there are no vectors with different values for f and g, i.e. we have $f(\mathbf{x}) = g(\mathbf{x})$.

The equivalence $h(\mathbf{x}) = f(\mathbf{x}) \sim g(\mathbf{x}) = 1$ is equivalent to the equations $f(\mathbf{x}) \oplus g(\mathbf{x}) = 0$ or $f(\mathbf{x}) = g(\mathbf{x})$ and is solved by all vectors $\mathbf{x}$

with $f(\mathbf{x}) = 0$, $g(\mathbf{x}) = 0$ and $f(\mathbf{x}) = 1$, $g(\mathbf{x}) = 1$, the solution set is equal to $(f_1 \cap g_1) \cup (f_0 \cap g_0) = f_0 \,\overline{\triangle}\, g_0$ which is the *complement of the symmetric difference (CSD)*. This is another item (CSD-Complement of SYD) which is available in the XBOOLE Monitor.

And finally we can consider the implication or the relation $f(\mathbf{x}) \leq g(\mathbf{x})$. Then we have $\mathbf{x} \notin f_1 \cap g_0 = \overline{f}_0 \cap g_0 = g_0 \setminus f_0$, and this is equivalent to $\mathbf{x} \in f_0 \cup \overline{g}_0 = f_0 \cup g_1$. The difference (DIF) of sets is the last set operation that is available in the XBOOLE Monitor.

The SAT-problem is theoretically one of the most important problems; it is NP-complete, and many other problems can be translated into a SAT-problem. For the user of the XBOOLE Monitor, however, the situation is not very difficult. Let us see, for instance, the following problem:

$$f = (x_1 \vee x_3 \vee \overline{x}_5)(x_1 \vee \overline{x}_2 \vee \overline{x}_3)(x_2 \vee \overline{x}_4 \vee \overline{x}_5) = 1.$$

The first disjunction $(x_1 \vee x_3 \vee \overline{x}_5) = 1$ has the ternary vectors $(1 - - - -), (0 - 1 - -), (0 - 0 - 0)$ as solution, and we used these vectors already as orthogonal vectors which is not very difficult to do. We could also enter the three vectors $(1 - - - -), (- - 1 - -), (- - - - 0)$ and use the item `Orthogonalization` for `Matrices`. This matrix of ternary vectors can be stored as an object, and then we create the analogous matrices for the second and the third disjunction. The intersection of these three matrices is the solution set of $f = 1$.

This approach allows to answer the first five items of the problem listed in [18]:

- Does each disjunction contain exactly three literals?
- Does each disjunction contain exactly k literals?
- Is it possible to split the conjunctive form into two parts CF_1 and CF_2, and CF_1 contains only non-negated variables, CF_2 only negated variables?
- Is there at most one vector which satisfies a given conjunctive form?
- How many vectors satisfy a given conjunctive form?

For all these cases we build the respective TVL and use the intersection as the solution procedure. The last two items need an additional counting of the number of (binary) vectors in the solution set.

It is even easier to use the vectors that assign the value 0 to a disjunction, because this is always only one vector: $(x_1 \vee x_3 \vee \overline{x}_5) = 0$ holds for one single vector $(0 - 0 - 1)$. Hence, we create a TVL with this single

vector and use the complement to find all the vectors generating the value 1 for this disjunction. The intersection with other TVLs then finds the solution set.

But we can still do better: we are using the union of all the vectors that assign the value 0 to one of the given conjunctions, because it is sufficient that one disjunction is equal to 0 in order to assign the value 0 to the whole expression. After the creation of the respective TVL the two steps of orthogonalization (not even necessary in the XBOOLE Monitor) and complement give us the solution of the equation which has been defined as a conjunctive form. We should keep this approach in mind when we will consider the SAT-problem: this is already the user side of the solution of the SAT-problem. Some vectors which are easy to find have to be typed, and the software and the computer system(s) have to carry the load.

For the small equation from above we get three vectors and the orthogonalization and the complement result in the following sequence:

$$\begin{pmatrix} 0 & - & 0 & - & 1 \\ 0 & 1 & 1 & - & - \\ - & 0 & - & 1 & 1 \end{pmatrix} \xrightarrow{\text{orthogonalization}} \begin{pmatrix} 0 & - & 0 & - & 1 \\ 0 & 1 & 1 & - & - \\ 1 & 0 & 0 & 1 & 1 \\ - & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\xrightarrow{\text{complement minimization}} \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & - & - \\ - & - & 0 & - & 0 \\ - & 0 & 1 & - & 0 \\ - & 0 & 1 & 0 & 1 \\ 1 & - & 0 & 0 & 1 \end{pmatrix}.$$

Exercise 3.1 (Equation). Let $f(x_1, x_2, x_3, x_4, x_5) = \overline{x}_2 x_3 x_4 \vee x_2 \overline{x}_3 x_5 \vee \overline{x}_1 \overline{x}_3 \overline{x}_5$, $g(x_1, x_2, x_3, x_4, x_5) = (\overline{x}_2 \oplus x_3)$.

1 Find all solution vectors of the equation $f(\mathbf{x}) = g(\mathbf{x})$.

2 Compare this set with the solution set of $f(x_1, x_2, x_3, x_4, x_5) \oplus g(x_1, x_2, x_3, x_4, x_5) = 0$.

3 Compare this set with the solution set of $f(x_1, x_2, x_3, x_4, x_5) \oplus g(x_1, x_2, x_3, x_4, x_5) = 1$.

4 Compare this set with the solution set of $f(x_1, x_2, x_3, x_4, x_5) \sim g(x_1, x_2, x_3, x_4, x_5) = 0$.

5 Compare this set with the solution set of $f(x_1, x_2, x_3, x_4, x_5) \sim g(x_1, x_2, x_3, x_4, x_5) = 1$.

Exercise 3.2 (Partial Solutions). For the given functions $f(\mathbf{x})$ and $g(\mathbf{x})$ of Exercise 3.1 consider the sets f_0, g_0, f_1 und g_1 and build (by set operations) $(f_0 \cap g_0) \cup (f_1 \cap g_1)$. What can be said about this set and its complement?

Exercise 3.3 (Equation). Let

$$f(x_1, x_2, x_3, x_4, x_5) = ((((x_1 \downarrow x_2) \downarrow x_3) \downarrow x_4) \downarrow x_5),$$
$$g(x_1, x_2, x_3, x_4, x_5) = ((((x_1|x_2)|x_3)|x_4)|x_5).$$

1 Find all solution vectors of the equation $f(\mathbf{x}) = g(\mathbf{x})$ by means of set operations.

2 Solve the equation $f(\mathbf{x}) = g(\mathbf{x})$ directly using the procedure implemented in the XBOOLE Monitor.

3 Are the sets calculated in the previous two items identical?

Exercise 3.4 (Equation). Take the functions $f(\mathbf{x})$ and $g(\mathbf{x})$ of Exercise 3.3 and the solution vector $(b_1, b_2, b_3, b_4, b_5) = (10000)$ and compare successively the number of solutions of subequations:

1 $f(x_1 = b_1) = 1$ and $g(x_1 = b_1) = 1$,

2 $f(x_1 = b_1, x_2 = b_2) = 1$ and $g(x_1 = b_1, x_2 = b_2) = 1$,

3 $f(x_1 = b_1, x_2 = b_2, x_3 = b_3) = 1$ and $g(x_1 = b_1, x_2 = b_2, x_3 = b_3) = 1$,

4 $f(x_1 = b_1, x_2 = b_2, x_3 = b_3, x_4 = b_4) = 1$ and $g(x_1 = b_1, x_2 = b_2, x_3 = b_3, x_4 = b_4) = 1$,

5 $f(x_1 = b_1, x_2 = b_2, x_3 = b_3, x_4 = b_4, x_5 = b_5) = 1$ and $g(x_1 = b_1, x_2 = b_2, x_3 = b_3, x_4 = b_4, x_5 = b_5) = 1$.

Exercise 3.5 (Inequality). Take the functions $f(\mathbf{x})$ and $g(\mathbf{x})$ of Exercise 3.3 and find the solutions of $f(\mathbf{x}) \neq g(\mathbf{x})$. Compare this solution set with the solution set of $f(\mathbf{x}) = g(\mathbf{x})$.

Exercise 3.6 (Implication). There are given the functions $f(\mathbf{x}) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$ and $g(\mathbf{x}) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5 \vee x_1\overline{x}_2x_3 \vee x_4\overline{x}_5 \vee x_3x_5$.

1 Check whether $f(\mathbf{x}) \leq g(\mathbf{x})$ holds for the given functions.

2 How the functions $f(\mathbf{x})$ or $g(\mathbf{x})$ must change in order to satisfy the relation $f(\mathbf{x}) \leq g(\mathbf{x})$.

3 Compare the solution sets of $f(\mathbf{x}) \leq g(\mathbf{x})$ and $\overline{f(\mathbf{x})} \vee g(\mathbf{x}) = 1$.

4 Compare the solution sets of $f(\mathbf{x}) \leq g(\mathbf{x})$ and $f(\mathbf{x})\overline{g(\mathbf{x})} = 0$.

Exercise 3.7 (Solution with Regard to Variables). Let be given

$$f(x_1, x_2, x_3x_4) = \overline{x}_1\overline{x}_2\overline{x}_3x_4 \vee x_1\overline{x}_2\overline{x}_3 \vee x_1x_2\overline{x}_3x_4 \vee \overline{x}_1x_2x_3x_4.$$

1 Find functions $x_3(x_1, x_2)$ and $x_4(x_1, x_2)$ which are defined by $f(\mathbf{x}) = 1$.

2 Are the functions $x_3(x_1, x_2)$ and $x_4(x_1, x_2)$ unique? How the function $f(x_1, x_2, x_3, x_4)$ must change in order to make them unique (if they are not unique)?

3 Are there constant functions $x_3(x_1, x_2)$ and $x_4(x_1, x_2)$ which are defined by $f(\mathbf{x}) = 0$?

Exercise 3.8 (Solution with Regard to Variables). Let be given the function $f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n)$.

1 What can be said about the number of solutions of the equation $f(\mathbf{x}) = 1$ if it is required to solve this equation with regard to $x_{k+1}(x_1, \ldots, x_k), \ldots, x_n(x_1, \ldots, x_k)$.

2 Is the condition formulated with regard to the number of solutions a necessary or a sufficient condition?

3 Use this knowledge in order to decide whether the equation

$$(x_1\overline{x}_4 \vee \overline{x}_1x_2x_4 \vee \overline{x}_1x_3x_4 \vee \overline{x}_2\overline{x}_3\overline{x}_4x_5) \wedge (x_2\overline{x}_3 \oplus x_5) = 1$$

can be solved with regard to $x_4(x_1, x_2, x_3)$ and $x_5(x_1, x_2, x_3)$ uniquely. Verify the solution.

Exercise 3.9 (System of Equations). Let be given the system of equations:

$$\begin{aligned} h_1 &= x_1 \oplus x_2, \\ h_2 &= x_1x_2, \\ f_1 &= h_1 \oplus x_3, \\ h_3 &= h_1x_3, \\ f_2 &= h_2 \vee h_3. \end{aligned}$$

1 Solve each of these equations separately and calculate the common solution using the partial solution sets.

2 Solve the system of equations using directly the operation `SBE` of the XBOOLE Monitor. Is the solution set equal to the solution set of item 1?

3 Transform the system of equations into a single equation and solve this equation using the operation SBE of the XBOOLE Monitor. Is the solution set equal to the solution set of item 1?

Exercise 3.10 (Satisfiability). Let be given the following equation in conjunctive form:

$$(x_1 \vee x_3 \vee \overline{x}_5 \vee \overline{x}_6)(x_3 \vee \overline{x}_4 \vee \overline{x}_5 \vee \overline{x}_6)(x_2 \vee x_4 \vee x_6)(\overline{x}_1 \vee \overline{x}_3 \vee \overline{x}_5) = 1.$$

1 Use the orthogonal solution sets for the single disjunctions and the intersection of these sets to find the solution.

2 Use the vectors that are not satisfying the respective single disjunctions and their union and the complement of this set to find the solution. Compare this solution with the result of item 1.

3 Use the SBE operation of XBOOLE to solve this equation and compare this solution with the result of item 1.

4 Show that the 5 questions on page 63 of this book can be answered without any special considerations.

Exercise 3.11 (Maximum Clauses). Let be given the function f by a conjunctive form with $n = 5$ clauses:

$$f = (x_1 \vee x_3 \vee x_4)(\overline{x}_2 \vee x_3 \vee x_4)(\overline{x}_2 \vee x_3 \vee \overline{x}_4)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3).$$

1 Represent the same function by k clauses, $k \leq n$ where the clauses are taken from the existing set of clauses and no further clause can be omitted.

2 Will the solution change when any clauses can be used?

3 Can this problem be solved by minimization?

Exercise 3.12 (MAXSAT). Solve the MAXSAT-problem for the following conjunctive form:

$$f = (a \vee b \vee c)(\overline{b} \vee \overline{c})(\overline{a} \vee \overline{d})(\overline{a} \vee d)(b \vee \overline{c})(c \vee d)(a \vee c).$$

1 Is the equation $f = 1$ for the given function satisfiable?

2 Which disjunctions of the given function must be removed in order to make the equation $f = 1$ satisfiable?

2. Solutions

Exercise 3.1.
By using `Extras` – `Solve Boolean Equation...` and the OBB operation for orthogonal minimization in the XBOOLE Monitor we get the solution vectors as follows:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 0 & 0 & 0 & - & 0 \\ 1 & 1 & 0 & - & 0 \\ - & 0 & 1 & 0 & - \end{pmatrix}.$$

Based on the solution sets of $f(\mathbf{x}) = 1$ and $g(\mathbf{x}) = 1$ and the required operations `SYD`, `CSD`, and `CPL` the solution sets of the items 2, 3, 4, and 5 are calculated. It can be checked by an empty result of `SYD` that the solution sets of item 2 and 5 are the same. Empty TVLs of `CSD` operations confirm that the solution sets of items 3 and 4 are the complement of this set.

Exercise 3.2.
The empty TVL 13 as result of the PRP

```
space 32 1          sbe 1 2          isc 1 2 10          verifies that
tin 1 1             /x2#x3.          isc 3 4 11          TVL 12 includes
x1 x2 x3 x4 x5.     cpl 1 3          uni 10 11 12        all the vectors
-011-               cpl 2 4          syd 12 5 13         (x1, x2, x3, x4, x5)
-10-1               csd 1 2 5                            with
0-0-0.
```

$$f(x_1, x_2, x_3, x_4, x_5) = g(x_1, x_2, x_3, x_4, x_5).$$

Exercise 3.3.

1 In a PRP a Boolean space and TVLs for the five variables are created. The NOR operations of the function f (object 13) are created by `UNI` and `CPL` operations. The NAND operations of the function g (object 23) are created by `ISC` and `CPL` operations.

```
space 32 1          tin 1 3          uni 1 2 10          isc 1 2 20
tin 1 1             x3.              cpl 10 10           cpl 20 20
x1.                 1.               uni 10 3 11         isc 20 3 21
1.                  tin 1 4          cpl 11 11           cpl 21 21
tin 1 2             x4.              uni 11 4 12         isc 21 4 22
x2.                 1.               cpl 12 12           cpl 22 22
1.                  tin 1 5          uni 12 5 13         isc 22 5 23
                    x5.              cpl 13 13           cpl 23 23
                    1.                                   csd 13 23 25
```

The operation `CSD` – `complement of SYD` calculates the solution of the equation $f(\mathbf{x}) = g(\mathbf{x})$ as object 25. The solution sets of $f = 1$ and $g = 1$ are:

$$F_1 = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 0 & 0 & 0 & 0 \\ - & 1 & 0 & 0 & 0 \\ - & - & - & 1 & 0 \end{pmatrix}, \quad G_1 = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 1 & 1 & 1 & 1 \\ - & - & - & - & 0 \\ - & - & 0 & 1 & 1 \end{pmatrix}.$$

Object 25 is the solution set we are searching for.

$$(f = g) = (F_1 \,\overline{\Delta}\, G_1) = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 0 & 0 & 0 & 0 \\ - & 1 & 0 & 0 & 0 \\ - & - & - & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ - & 0 & 1 & 1 & 1 \\ - & - & - & 0 & 1 \end{pmatrix}.$$

2 The result of
sbe 1 30
/(/(/(/(x1+x2)+x3)+x4)+x5)=/(/(/(/(x1&x2)&x3)&x4)&x5).
shows the same vectors as the previous item only in another order.

3 syd 25 30 40
The empty TVL 40 confirms that both solution sets are equal.

Exercise 3.4.

1 The basic sets are prepared by the following PRP:

space 32 1	sbe 1 2	sbe 1 4	sbe 1 8
sbe 1 1	x2=0.	x4=0.	/(/(/(/(x1+x2)+x3)+x4)+x5).
x1=1.	sbe 1 3	sbe 1 5	sbe 1 9
	x3=0.	x5=0.	/(/(/(/(x1&x2)&x3)&x4)&x5).

From isc 8 1 10 and isc 9 1 11 we get $|f(1,\mathtt{x})| = 6$ and $|g(1,\mathtt{x})| = 11$.

2 From isc 10 2 20 and isc 11 2 21 we get $|f(10,\mathtt{x})| = 3$ and $|g(10,\mathtt{x})| = 5$.

3 From isc 20 3 30 and isc 21 3 31 we get $|f(100,\mathtt{x})| = 2$ and $|g(100,\mathtt{x})| = 3$.

4 From isc 30 4 40 and isc 31 4 41 we get $|f(1000,x_5)| = 1$ and $|g(1000,x_5)| = 1$.

5 From isc 40 5 50 and isc 41 5 51 we get $|f(10000)| = 1$ and $|g(10000)| = 1$.

Exercise 3.5.

```
space 32 1
sbe 1 1
/(/(/(/(x1+x2)+x3)+x4)+x5).
sbe 1 2
/(/(/(/(x1&x2)&x3)&x4)&x5).
syd 1 2 3
csd 1 2 4
isc 3 4 5
uni 3 4 6
cpl 6 7
csd 3 4 8
```

Using the PRP on the left object 3 includes the solution of the inequation $f \neq g$ (see below). For the purpose of comparison the object 4 is the solution of the equation $f = g$. The empty TVL 5 confirms that there is no common solution. The empty TVL 7 confirms that the union of both solution sets 3 and 4 covers the Boolean space B^5 completely. Thus the solution sets 3 and 4 are complementary to each other, confirmed by the empty TVL 8.

$$(f \neq g) = (F_1 \Delta G_1) = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ - & - & 0 & 1 & 1 \\ - & - & 1 & 0 & 0 \end{pmatrix}.$$

Exercise 3.6.

```
1 space 32 1
  sbe 1 1
  x1#x2#x3#x4#x5.
  sbe 1 2
  /x1&/x2&/x3&/x4&/x5+
  x1&/x2&x3+x4&/x5+x3&x5.
  dif 1 2 3
```

The solution set of $f(\mathbf{x}) \leq g(\mathbf{x})$ covers all solution vectors of $f(\mathbf{x}) = 1$ that are not solution vectors of $g(\mathbf{x}) = 1$. This set can be calculated by the difference operation (`DIF`) of XBOOLE. The solution set covers 8 vectors (00001), (00100), (01000), (01011), (10000), (10011), (11001), and (11100).

```
2 dif 1 3 10
  dif 10 2 11
  uni 2 3 12
  dif 1 12 13
```

The relation $f(\mathbf{x}) \leq g(\mathbf{x})$ is satisfied if $f(\mathbf{x})$ is restricted to function values 1 that do not appear in the solution set 3. The empty set 11 confirms this property. Vice versa the function $g(\mathbf{x})$ can be extended by the solution set 3 into the new function 12. The empty set 13 confirms that the extended function 12 satisfies the relation too.

```
3 cpl 1 20
  uni 20 2 21
  csd 21 3 22
```

The solution set 21 is calculated by a complement of f and a union with g. The empty set 22 confirms that the solution sets of $f(\mathbf{x}) \leq g(\mathbf{x})$ and $\overline{f(\mathbf{x})} \vee g(\mathbf{x}) = 1$ are complements of each other.

```
4 cpl 2 30
  isc 1 30 31
  cpl 31 32
  csd 32 3 33
```

The solution set 32 covers the solution vectors of $f(\mathbf{x})\overline{g(\mathbf{x})} = 0$. The empty set 33 confirms that this solution set is the complement of the solution set of $f(\mathbf{x}) \leq g(\mathbf{x})$.

Exercise 3.7.

```
1 space 32 1
  tin 1 1
  x1 x2 x3 x4.
  0001
  100-
  1101
  0111.
```

The solution set of the equation $f(\mathbf{x}) = 1$ is given by the TVL 1

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & - \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

```
  sbe 1 2
  x3=/x1&x2.
  sbe 1 3
  x4=1.
  sbe 1 4
  x4=/x1+x2.
  vtin 1 5
  x3 x4.
```

```
  isc 1 2 10
  isc 10 3 11
  dco 11 5 12
  cpl 12 13
  isc 10 4 15
  dco 15 5 16
  cpl 16 17
```

Using x_1 and x_2 as arguments, it can be seen that all the possibilities (00), (01), (10), and (11) for the arguments exist. Taking TVL 1 as function table we get $x_3 = \overline{x}_1 x_2$ as well as $x_4 = 1$ and $x_4 = \overline{x}_1 \vee x_2$. The empty sets 13 and 17 after the substitution of both pairs of functions into the equation $f(\mathbf{x}) = 1$ confirm these solutions.

2 The single function $x_3(x_1, x_2)$ is unique, but there are two functions $x_4(x_1, x_2)$. Hence, $x_4(x_1, x_2)$ is not unique. In order to get a unique solution, the $-$ in the second vector must be replaced by 0 or 1.

3
```
cpl 1 21
sbe 1 22
x3=1.
sbe 1 23
x4=0.
isc 21 22 25
isc 25 23 26
dco 26 5 27
cpl 27 28
```

The solution set of the equation $f(\mathbf{x}) = 0$ is given by TVL 21.

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ - & - & 1 & 0 \\ - & 1 & 0 & 0 \\ - & 0 & 1 & 1 \end{pmatrix}$$

It is the complement of TVL 1. The vector $(- - 10)$ shows that the constant functions $x_3 = 1$, $x_4 = 0$ can be used for all the values of x_1 and x_2. A row of values 1 in the associated Karnaugh map shows this property, too. The empty set 28 after substitutions of this pair of functions into the equation $f(\mathbf{x}) = 0$ confirms these solutions.

Exercise 3.8.

1 The solution functions depend on k variables $x_1, \ldots, x_k$. Hence, the equation $f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n) = 1$ must have at least 2^k solutions such that each vector $(x_1, \ldots, x_k)$ appears in the solution set.

2 It is only a necessary condition, since many vectors $(x_{k+1}, \ldots, x_n)$ can be in the solution set with one vector for $(x_1, \ldots, x_k)$.

3
```
space 32 1
avar 1
x1 x2 x3 x4 x5.
sbe 1 1
(x1&/x4+/x1&x2&x4+/x1&x3&x4+
/x2&/x3&/x4&x5)&(x2&/x3#x5)=1.
vtin 1 2
x4 x5.
sbe 1 10
x4=/x1&(x2+x3).
sbe 1 11
x5=/x2+x3.
isc 1 10 15
isc 15 11 16
dco 16 2 17
cpl 17 18
```

There are 8 solution vectors of the equation 1 such that the necessary condition holds. The Karnaugh map of the solution set shows exactly one solution value 1 for each vector (x_1, x_2, x_3). The empty set 18 confirms that the pair of functions $x_4 = \overline{x}_1 (x_2 \vee x_3)$ and $x_5 = \overline{x}_2 \vee x_3$ is a valid solution.

Exercise 3.9.

1
```
space 32 1
avar 1
x1 x2 x3 h1 h2
h3 f2 f1.
sbe 1 1
h1=x1#x2.
sbe 1 2
h2=x1&x2.
sbe 1 3
f1=h1#x3.
sbe 1 4
h3=h1&x3.
sbe 1 5
f2=h2+h3.
isc 1 2 6
isc 6 3 6
isc 6 4 6
isc 6 5 6
```

The solution set is calculated by intersections of the solution sets of the five equations and includes 8 solution vectors.

2
```
sbe 1 10
h1=x1#x2,
h2=x1&x2,
f1=h1#x3,
h3=h1&x3,
f2=h2+h3.
syd 10 6 11
```

The `SBE` operation of XBOOLE allows to solve systems of equations directly. A comma sign separates two equations of the equation system. The empty set 11 confirms that solution sets 6 and 10 are identical.

```
3 sbe 1 20
  (h1=x1#x2)&
  (h2=x1&x2)&
  (f1=h1#x3)&
  (h3=h1&x3)&
  (f2=h2+h3)=1.
  syd 20 6 21
```

Each equation is transformed into an expression where an equivalence operation connects both sides of the given equations. All these expressions must be equal to one. Hence, AND operations between them form the required single characteristic equation. The empty set 21 confirms that solution sets 6 and 20 are identical.

Exercise 3.10.

```
1 space 32 1
  avar 1
  x1 x2 x3 x4 x5 x6.
  sbe 1 1
  (x1+x3+/x5+/x6)=1.
  sbe 1 2
  (x3+/x4+/x5+/x6)=1.
  sbe 1 3
  (x2+x4+x6)=1.
  sbe 1 4
  (/x1+/x3+/x5)=1.
  isc 1 2 5
  isc 5 3 5
  isc 5 4 5
  obbc 5 5
```

The solution sets of the first and second disjunction are represented by 4 orthogonal ternary vectors each, and solution sets of the third and fourth disjunction need 3 orthogonal ternary vectors each. The intersection of these four sets $M_1 \cap M_2 \cap M_3 \cap M_4$ is the solution S of the SAT-problem. 9 orthogonal ternary vectors represent 43 boolean solution vectors.

$$S = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & - & - & 1 & 0 \\ - & 1 & - & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & - \\ 0 & 0 & - & 1 & 1 & 0 \\ - & - & - & 1 & 0 & - \\ 1 & - & 0 & 1 & 1 & 0 \\ - & - & - & 0 & 0 & 1 \\ 0 & - & 1 & - & 1 & 1 \end{pmatrix}.$$

```
2 cpl 1 11
  cpl 2 12
  cpl 3 13
  cpl 4 14
  uni 11 12 15
  uni 15 13 15
  uni 15 14 15
  cpl 15 16
  obbc 16 16
  syd 5 16 17
```

The TVLs 11, 12, 13, and 14 consist of one vector which does not satisfy the respective single disjunction. Their union and a final complement leads to the same solution, confirmed by the empty set 17.

```
  tin 1 18
  x1 x2 x3 x4 x5 x6.
  0-0-11
  –0111
  -0-0-0
  1-1-1-.
  cpl 18 19
  syd 19 5 20
```

Alternatively the four vectors that do not satisfy the respective single disjunction can be specified in a TVL. The complement of this set is equal to the previous solution set S. This will be again confirmed by the empty set 20.

```
3 sbe 1 30
  (x1+x3+/x5+/x6)&(x3+/x4+/x5+/x6)&
  (x2+x4+x6)&(/x1+/x3+/x5)=1.
  syd 30 5 31
```

The given SAT problem can be solved directly using `SBE`. The empty set 31 confirms that the same solution was calculated.

4 These questions can be answered by a simple inspection of the given conjunctive form and/or the respective solution sets.

Exercise 3.11.

1
```
space 32 1
tin 1 1 /d
x1 x2 x3 x4.
0-00
-100
-101
010-
110-.
orth 1 2
dtv 1 1 10
orth 10 11
syd 11 2 12
dtv 1 2 13
orth 13 14
syd 14 2 15
dtv 13 2 20
orth 20 21
syd 21 2 22
dtv 20 2 30
orth 30 31
syd 31 2 32
dtv 20 3 33
orth 33 34
syd 34 2 35
```

There are several possibilities to solve this problem. We will use the following approach. We combine the ternary vectors corresponding to conjunctions into TVL 1 in such a way that the conjunction is equal to 0 for this vector. For the purpose of comparison the orthogonal set 2 is created. Step by step one vector of TVL 1 is deleted using the XBOOLE operation `DTV` (delete ternary vector). The orthogonalized remaining sets are compared with the original set 2 using the *symmetric difference*. The set 12 is not empty; that means the first row is necessary. The empty set 15 confirms that the second disjunction can be omitted. We continue to remove the third row in addition to the second row. The empty set 22 confirms that these two disjunctions can be omitted. The sets 32 and 35 which are not empty confirm finally that no further disjunction can be omitted. From the TVL 20 we get the expression of the same function expressed with only $k = 3$ disjunctions:

$$f = (x_1 \vee x_3 \vee x_4)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3).$$

2 The last two clauses of the previous result can be expressed by the single clause $(\overline{x}_2 \vee x_3)$. Thus, simpler solutions can exist if any clause can be used.

3
```
obbc 2 40.
```

Even the orthogonal minimization finds expressions with two clauses:

$$f = (x_1 \vee x_2 \vee x_3 \vee x_4)(\overline{x}_2 \vee x_3).$$

Exercise 3.12.

1
```
space 32 1
tin 1 1 /d
a b c d.
000-, -11-, 1–1, 1–0
-01-, –00, 0-0-.
cpl 1 2
```

This problem can be solved in the same way as the previous problem. The `CPL` operation calculates the complement for orthogonal and non-orthogonal D- and K-forms. The empty set 2 confirms that the given equation is not satisfiable.

```
cpl 1 2
dtv 1 1 10
cpl 10 11
dtv 1 2 12
cpl 12 13
dtv 1 3 14
cpl 14 15
dtv 1 4 16
cpl 16 17
dtv 1 5 18
cpl 18 19
dtv 1 6 20
cpl 20 21
dtv 1 7 22
cpl 22 23
```

The sets 13, 15, 19, and 23 are not empty. Hence, the omission of one of the clauses $(\overline{b} \vee \overline{c})$, $(\overline{a} \vee \overline{d})$, $(b \vee \overline{c})$ or $(a \vee c)$ of the given function f results in a satisfiable conjunctive form.

Chapter 4

BOOLEAN DIFFERENTIAL CALCULUS

1. Differentials

Differentials $\mathrm{d}\mathbf{x}$ of Boolean variables $\mathbf{x}$ occur in graph equations. The differential $\mathrm{d}x_i$ of a single variable x_i has a special meaning, the change of the value of the Boolean variable x_i, but it is a Boolean variable too. Thus a graph equation can be solved in the same way as a logic equation.

Exercise 4.1 (Graph Equation). Solve the graph equation $F(a, \mathbf{x}, \mathrm{d}\mathbf{x}) = G(a, \mathbf{x}, \mathrm{d}\mathbf{x})$ where

$$\begin{aligned} F(a,\mathbf{x},\mathrm{d}\mathbf{x}) = {} & \overline{x}_1 x_3\,\mathrm{d}x_1\,\mathrm{d}x_2\,\overline{\mathrm{d}x_3} \vee \overline{x}_1\overline{x}_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_3} \vee x_1\overline{x}_2\,\overline{\mathrm{d}x_1}\,\mathrm{d}x_2 \\ & \vee \overline{a}x_1x_2\overline{x}_3\,\mathrm{d}x_2 \vee x_1\overline{x}_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_2} \\ & \vee x_1x_2x_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \vee \overline{x}_2\,\mathrm{d}x_1\,\mathrm{d}x_3 \\ & \vee ax_1\overline{x}_2\overline{x}_3\,\mathrm{d}x_1\,\mathrm{d}x_2\,\overline{\mathrm{d}x_3} \vee \overline{a}\,\overline{x}_1x_3\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \\ & \vee \overline{a}\,\overline{x}_1x_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_3} \vee \overline{x}_1\overline{x}_2x_3\,\overline{\mathrm{d}x_1}\,\mathrm{d}x_3 \vee a\overline{x}_1x_2\,\overline{\mathrm{d}x_2} \\ & \vee \overline{a}\,\overline{x}_3\,\mathrm{d}x_1\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \vee a\overline{x}_1\overline{x}_3\,\overline{\mathrm{d}x_1}\mathrm{d}x_2\,\mathrm{d}x_3 \\ & \vee x_1\overline{x}_2x_3\,\mathrm{d}x_1\,\mathrm{d}x_2\,\overline{\mathrm{d}x_3} \vee a\overline{x}_1\overline{x}_2\,\mathrm{d}x_1\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \end{aligned}$$

and

$$\begin{aligned} G(a,\mathbf{x},\mathrm{d}\mathbf{x}) = {} & x_2x_3\,\mathrm{d}x_2 \vee \overline{a}x_1x_3\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \vee ax_1x_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\mathrm{d}x_2\,\mathrm{d}x_3 \\ & \vee \overline{a}\,\overline{x}_1\overline{x}_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\mathrm{d}x_2\,\mathrm{d}x_3 \vee \overline{a}\,\overline{x}_1x_2\overline{x}_3\,\mathrm{d}x_2 \\ & \vee ax_1\overline{x}_2x_3\,\mathrm{d}x_1\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \vee ax_1x_2\overline{x}_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_3} \\ & \vee a\overline{x}_2x_3\,\overline{\mathrm{d}x_1}\,\overline{\mathrm{d}x_2}\,\overline{\mathrm{d}x_3} \vee \overline{x}_1\overline{x}_2\overline{x}_3\,\mathrm{d}x_1\,\mathrm{d}x_2\,\overline{\mathrm{d}x_3} \\ & \vee ax_1x_2\,\mathrm{d}x_1\,\overline{\mathrm{d}x_2} \vee ax_2\overline{x}_3\mathrm{d}x_1\,\mathrm{d}x_2 \vee \overline{a}x_2\,\overline{\mathrm{d}x_2}\,\mathrm{d}x_3. \end{aligned}$$

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

Use the XBOOLE-monitor to solve this graph equation. It is suggested to edit the equation in a text file (e.g. e41_graph.txt) using the preferred operation signs (e.g. AND: &, OR: +, NOT: !) and an editor of your choice. Draw the resulting graph. Take the Boolean variables x_1, x_2, and x_3 as code of the vertices of the graph. The differentials $\mathrm{d}x_1$, $\mathrm{d}x_2$, and $\mathrm{d}x_3$ describe the direction of the edges. Take the variable a as label of edges. Answer the following questions:

1 How many edges contains the graph? (Count edges that do not depend on the value of a only once.)

2 How many connected components includes the graph?

3 Cycles of which length can be found in the graph for $a = 1$?

4 Cycles of which length contains the graph for $a = 0$?

Basically there are two representations of graphs. The first describes the edges by the code of a vertex and the direction by values of differentials as used in the exercise above. This graph representation is called differential representation. The second one describes an edge by the codes of the starting vertex $\mathbf{x}$ and the end vertex $\mathbf{x}'$. This graph representation is called sequential representation. Though both representations will describe the same graph, each one of them has some benefits for particular analysis tasks. Therefore we need a transformation between both representations of a graph.

The XBOOLE monitor allows only alphanumerical characters in the names of Boolean variables. Therefore in the following the variable x' will be expressed by xf or the variable x'_i by xf_i, respectively. The character f stands for *following*.

Exercise 4.2 (Graph Transformation). Transform the differential representation $G_d(a, \mathbf{x}, \mathrm{d}\mathbf{x})$ of the graph calculated in Exercise 4.1 into its sequential representation $G_s(a, \mathbf{x}, \mathbf{xf})$.

Use the equation $xf_i = x_i \oplus \mathrm{d}x_i$, $i = 1, 2, 3$, as the theoretical background for the transformation. Restrict the result of the transformation to the variables $(a, \mathbf{x}, \mathbf{xf})$ by means of an m-fold maximum operation. Practical tasks:

1 Write a PRP `e42_gds.prp` that requires the TVL of $G_d(a, \mathbf{x}, \mathrm{d}\mathbf{x})$ as object number 1 and stores the resulting TVL of $G_s(a, \mathbf{x}, \mathbf{xf})$ as object number 2 where intermediate objects may be stored as objects with numbers larger than 10.

2 Prepare the XBOOLE-monitor in such a way that the TVL of $G_d(a, \mathbf{x}, \mathrm{d}\mathbf{x})$ is the object number 1, and execute the PRP `e42_gds.prp`.

Table 4.1 Solution set of thegraph equation of Exercise 4.1

$G_s(a, \mathbf{x}, \mathrm{d}\mathbf{x}) =$

a	x_1	x_2	x_3	$\mathrm{d}x_1$	$\mathrm{d}x_2$	$\mathrm{d}x_3$
–	0	1	1	1	1	0
–	0	0	0	0	0	1
0	1	0	0	1	1	0
–	0	1	0	0	1	0
0	1	1	–	0	0	0
1	1	0	0	1	0	0
–	1	0	1	0	0	1
–	0	0	1	0	1	0
1	1	1	–	0	0	1

3 Draw the graph of $G_s(a, \mathbf{x}, \mathbf{xf})$ stored as object number 2 and compare it with the graph of Exercise 4.1.

4 Write a PRP `e42_gsd.prp` that requires for the inverse transformation the TVL of $G_s(a, \mathbf{x}, \mathbf{xf})$ as object number 2 and stores the resulting TVL of $G_d(a, \mathbf{x}, \mathrm{d}\mathbf{x})$ as object number 3 where intermediate objects may be stored as objects with numbers larger than 20.

5 Execute the PRP `e42_gsd.prp` and compare whether the result of the inverse transformation (TVL 3) is the same function as the given function (TVL 1) by means of a SYD-operation.

There are three total differential operations for a logic function. The difference between them is the internal operation $\oplus, \wedge$, or $\vee$. This operation must be executed between $f(\mathbf{x})$ and $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x})$. Thus the main challenge is the calculation of $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x})$ in order to create the total differential $\mathrm{d}_{\mathbf{x}} f(\mathbf{x})$, the total differential minimum $\mathrm{Min}_{\mathbf{x}} f(\mathbf{x})$, and the total differential maximum $\mathrm{Max}_{\mathbf{x}} f(\mathbf{x})$. One possibility to create the function $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x})$ is the substitution of $(x_i \oplus \mathrm{d}x_i)$ for x_i by hand.

Exercise 4.3 (Total Differential Operation). Calculate all three total differential operations of the function

$$f(x_1, x_2, x_3) = x_1(\overline{x}_2 \vee \overline{x}_3) \oplus \overline{x}_1(\overline{x}_2 \sim x_3) \tag{4.1}$$

and verify relation (4.2) of [18]. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3) = 1$ for function (4.1) and store the result as object number 1.

2 Change the expression of $f(\mathbf{x})$ into $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x})$ using the SBE dialog window of the XBOOLE-monitor and solve the equation $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x}) = 1$ storing the result as object number 2.

3 Calculate the total differential $\mathrm{d}_{\mathbf{x}} f(\mathbf{x})$ by means of a SYD-operation as object number 3. Draw the associated graph and count the edges.

4 Calculate the total differential minimum $\mathrm{Min}_{\mathbf{x}} f(\mathbf{x})$ by means of an ISC-operation as object number 4. Draw the associated graph and count the edges.

5 Calculate the total differential maximum $\mathrm{Max}_{\mathbf{x}} f(\mathbf{x})$ by means of a UNI-operation as object number 5. Draw the associated graph and count the edges.

6 Verify relation (4.2) of [18] by means of SYD-operations between object 3, 4, and 5 and store the result as object number 6.

The substitution method applied in Exercise 4.3 is time-consuming and error-prone. Therefore another method is suggested. Based on formula (4.3) of [18] the function $f(\mathbf{x} \oplus \mathrm{d}\mathbf{x})$ can be replaced by $f(\mathbf{xf})$. The transformation from $f(\mathbf{x})$ into $f(\mathbf{xf})$ is a simple substitution of variables and can easily be done using the XBOOLE operation TCO. Thereafter the total differential operations can be calculated in sequential form. The transformation `gsd.prp` from the sequential form into the differential form created in Exercise 4.2 leads finally to the desired total differential operations.

Exercise 4.4 (Total Differential Operation using Transformation). Calculate all three total differential operations of function (4.1) using the transformation method and verify the solution. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3) = 1$ for function (4.1) and store the result as object number 1.

2 Prepare VT $\langle x1, x2, x3 \rangle$ as object number 2 and VT $\langle xf1, xf2, xf3 \rangle$ as object number 3 as the basis for the change of columns.

3 Create the function $f(xf_1, xf_2, xf_3)$ by means of an appropriate CCO-operation.

4 Write a PRP for the transformation of a differential operation in sequential form given as object number 10 into a differential operation in differential form generated as object number 11.

5 Create the total differential of (4.1) in sequential form as object number 10, apply the PRP for the transformation into the differential form, copy the generated object from number 11 to object number 5, visualize the total differential of number 5 as Karnaugh map, and count the values 1.

6 Create the total differential minimum of (4.1) in sequential form as object number 10, apply the PRP for transformation into the differential

form, copy the generated object from number 11 to object number 6, visualize the total differential minimum of number 6 as Karnaugh map, and count the values 1.

7 Create the total differential maximum of (4.1) in sequential form as object number 10, apply the PRP for the transformation into the differential form, copy the generated object from number 11 to object number 7, visualize the total differential maximum of number 7 as Karnaugh map, and count the values 1.

8 Compare the Karnaugh maps with the results of Exercise 4.3 and verify relation (4.2) of [18] by means of SYD-operations between the objects 5, 6, and 7 and store the result as object number 8.

In oder to verify the relations (4.8), (4.10), and (4.11) of [18], the total differential expansion $F(\mathbf{x}, \mathrm{d}\mathbf{x})$ of $f(\mathbf{x})$ is required. This expansion transforms the logic function from the space B^n into the space B^{2n}. Inside of the XBOOLE system operations between several TVLs are allowed only if these TVLs are defined in the same Boolean space. Thus the relations mentioned above do not require an explicit creation of the total differential expansion $F(\mathbf{x}, \mathrm{d}\mathbf{x})$ of $f(\mathbf{x})$, because the logic function $f(\mathbf{x})$ must be embedded in the same Boolean space as the total differential operations. Note: the visualization of a TVL or Karnaugh map reflects the variables occurring in its data structure. Therefore the explicit creation of the total differential expansion may be necessary. The expansion can be calculated using a full TVL defined over the required differentials.

Exercise 4.5 (Partial Order of Total Differential Operations). Verify the partial order relation (4.11) of [18] using the results of Exercise 4.4, the total differential expansion $F(\mathbf{x}, \mathrm{d}\mathbf{x})$ of (4.1) and the logic function itself. Practical tasks:

1 Write a PRP that takes object number 1 as a logic function of three variables $(x1, x2, x3)$ and generates its total differential expansion as object number 31.

2 Reload the result of Exercise 4.4 and apply the written PRP in order to generate the total differential expansion of (4.1) based on definition (4.6) of [18]. Visualize the Karnaugh map of the generated total differential expansion in object number 31.

3 Verify (4.8) of [18] using the total differential minimum of object number 6 and the differential expansion in object number 31.

4 Verify (4.10) of [18] using the total differential maximum of object number 7 and the differential expansion in object number 31.

5 Verify (4.11) of [18] using the total differential minimum of object number 6, the logic function of object number 1, and the total differential maximum of object number 7.

Partial differential operations restrict the directions for the consideration of changes of the function to selected variables. Consequently, the number of differential variables is reduced in comparison to total differential operations – therefore the calculation effort is smaller for partial differential operations. There are relations for partial differential operations which are analogous to the relations studied before for total differential operations.

Exercise 4.6 (Partial Differential Operation). Calculate all three partial differential operations of the function

$$f(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2)\ x_3 \vee x_1 x_4 \vee \overline{(x_1\ x_2)}\ \overline{(x_3 \vee x_4)} \tag{4.2}$$

with regard to (x_3, x_4), draw their graphs, and verify the relations between these three partial differential operations. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for the function (4.2) and store the result as object number 1.

2 Create the function $f(x_1, x_2, xf_3, xf_4)$ by means of an appropriate PRP and store the orthogonal result as object number 4.

3 Write a PRP that transforms a differential operation in sequential form, given as object number 10 with regard to the variables (x_3, x_4), into a differential operation in differential form generated as object number 11.

4 Create the partial differential $\mathrm{d}_{(x_3,x_4)} f(\mathbf{x})$ of (4.2) in sequential form as object number 10, apply the PRP for transformation into the differential form, copy the generated object from number 11 to object number 5, visualize the partial differential of number 5 as graph, and count the edges.

5 Create the partial differential minimum $\mathrm{Min}_{(x_3,x_4)} f(\mathbf{x})$ of (4.2) in sequential form as object number 10, apply the PRP for transformation into the differential form, copy the generated object from number 11 to object number 6, visualize the partial differential minimum of number 6 as graph, and count the edges.

6 Create the partial differential maximum $\mathrm{Max}_{(x_3,x_4)} f(\mathbf{x})$ of (4.2) in sequential form as object number 10, apply the PRP for transformation into the differential form, copy the generated object from number 11 to object number 7, visualize the partial differential maximum of number 7 as graph, and count the edges.

7 Verify relation (4.21) of [18] by means of SYD-operations between the objects 5, 6, and 7 and store the result as object number 20.

8 Verify the left relation in (4.20) of [18] by means of a DIF-operation between the objects 6 and 1 and store the result as object number 21.

9 Verify the right relation in (4.20) of [18] by means of a DIF-operation between the objects 1 and 7 and store the result as object number 22.

10 For the last two tasks the function of object number 1 could be used instead of the partial differential expansion $F(x_1, x_2, x_3, x_4, \mathrm{d}x_3, \mathrm{d}x_4)$. Explain why!

A partial differential operation with regard to a single variable is again a logic function. Hence, another partial differential operation of the same type with regard to another variable can be calculated. This is the basic procedure for the definition of m-fold differential operations. Both the partial and the m-fold differential operations can depend on a subset of differentials $\mathrm{d}x_i$. The semantics of these classes of differential operations is quite different and should be studied in the following exercise.

Exercise 4.7 (M-fold Differential Operations). Calculate all four m-fold differential operations of the function (4.2) with regard to (x_3) and (x_3, x_4), draw their graphs, compare these graphs with the graphs of the previous Exercise 4.6, and verify the relations between these m-fold differential operations. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for the function (4.2) and store the result as object number 1.

2 Prepare for further calculation the VTs 2: $\langle x3 \rangle$, 3: $\langle xf3 \rangle$, 4: $\langle x4 \rangle$, 5: $\langle xf4 \rangle$, and the TVLs for sequential to differential transformation, 6: for $x3$, and 7: for $x4$.

3 Write a PRP that calculates both $\mathrm{d}_{x_3} f(\mathbf{x})$ as object number 13, and $\mathrm{d}^2_{(x_3,x_4)} f(\mathbf{x})$ as object number 17.

4 Execute this PRP and draw the graphs of the two m-fold differentials.

5 Write a PRP that calculates both $\mathrm{Min}_{x_3} f(\mathbf{x})$ as object number 23, and $\mathrm{Min}^2_{(x_3,x_4)} f(\mathbf{x})$ as object number 27.

6 Execute this PRP and draw the graphs of both m-fold differential minimum operations.

7 Write a PRP that calculates both $\mathrm{Max}_{x_3} f(\mathbf{x})$ as object number 33, and $\mathrm{Max}^2_{(x_3,x_4)} f(\mathbf{x})$ as object number 37.

8 Execute this PRP and draw the graphs of both m-fold differential maximum operations.

9 Calculate both $\vartheta_{x_3} f(\mathbf{x})$ as object number 40, and $\vartheta_{(x_3,x_4)} f(\mathbf{x})$ as object number 41, and draw the associated graphs.

10 Check whether $\mathrm{d}_{x_3} f(\mathbf{x})$ and $\vartheta_{x_3} f(\mathbf{x})$ are the same function.

11 Check whether $\mathrm{d}^2_{(x_3,x_4)} f(\mathbf{x})$ and $\vartheta_{(x_3,x_4)} f(\mathbf{x})$ are the same function.

12 Verify the inequalities

$$\mathrm{Min}^2_{(x_3,x_4)} f(\mathbf{x}) \leq \mathrm{Min}_{x_3} f(\mathbf{x}), \tag{4.3}$$

$$\mathrm{Min}_{x_3} f(\mathbf{x}) \leq f(\mathbf{x}), \tag{4.4}$$

$$f(\mathbf{x}) \leq \mathrm{Max}_{x_3} f(\mathbf{x}), \tag{4.5}$$

$$\mathrm{Max}_{x_3} f(\mathbf{x}) \leq \mathrm{Min}^2_{(x_3,x_4)} f(\mathbf{x}) \tag{4.6}$$

using the objects number 27, 23, 1, 33, and 37, respectively.

2. Derivatives

Derivatives of logic functions are strongly associated with their differential operations. The direction of change is specified in differential operations by the values of the differentials $\mathrm{d}x_i$. If a direction of change is fixed by constant values of the differentials $\mathrm{d}x_i$, the remaining function is the appropriate derivative. The restriction in the direction of change of a derivative is compensated by a reduced number of variables.

Based on three classes of differential operations, there are three classes of derivatives: simple derivatives, vectorial derivatives, and m-fold derivatives. The simple derivatives are a special case of both the vectorial and the m-fold derivatives when the derivatives are calculated with regard to a single variable x_i. Hence, the XBOOLE-monitor includes operations for three vectorial and three m-fold derivatives.

The XBOOLE operations for three vectorial derivatives with regard to a single variable x_i use their definition (4.28), (4.31), and (4.34) of [18] such that the variable x_i appears in the results. The XBOOLE operations for three m-fold derivatives with regard to a single variable x_i use the equation of theorem (4.29), (4.32), and (4.35) of [18] such that the variable x_i does not appear in the result.

Exercise 4.8 (Simple Derivatives). Calculate all three simple derivatives of the function

$$f(x_1, x_2, x_3, x_4) = (x_1 \overline{x}_2 \oplus \overline{x}_2 x_3 x_4) \vee \overline{x}_1 x_2 x_4 \tag{4.7}$$

with regard to x_4 using both types of possible XBOOLE operations, visualize the Karnaugh maps, and verify the relations between these three simple derivatives. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for the function (4.7) and store the result as object number 1.

2 Prepare VT $\langle x4 \rangle$ as object number 2.

3 Calculate $\frac{\partial f(\mathbf{x})}{\partial x_4}$ using the XBOOLE operation `derv`, show the Karnaugh map of the result, and compare it with the Karnaugh map of the given function.

4 Calculate $\frac{\partial f(\mathbf{x})}{\partial x_4}$ using the XBOOLE operation `derk`, show the Karnaugh map of the result, and compare it with the Karnaugh maps of the given function and the result of subtask 3.

5 Calculate $\min_{x_4} f(\mathbf{x})$ using the XBOOLE operation `minv`, show the Karnaugh map of the result, and compare it with the Karnaugh map of the given function.

6 Calculate $\min_{x_4} f(\mathbf{x})$ using the XBOOLE operation `mink`, show the Karnaugh map of the result, and compare it with the Karnaugh maps of the given function and the result of subtask 5.

7 Calculate $\max_{x_4} f(\mathbf{x})$ using the XBOOLE operation `maxv`, show the Karnaugh map of the result, and compare it with the Karnaugh map of the given function.

8 Calculate $\max_{x_4} f(\mathbf{x})$ using the XBOOLE operation `maxk`, show the Karnaugh map of the result, and compare it with the Karnaugh maps of the given function and the result of subtask 7.

9 Verify (4.36) of [18].

10 Verify (4.40) of [18].

The application of the Boolean Differential Calculus leads to expressions that comprise differential and derivative operations. Relations between such operations help to simplify such expressions. The following exercise verifies such relations between simple derivative operations and supports their understanding.

Exercise 4.9 (Relations between Simple Derivatives). Calculate all three simple derivatives of the function (4.7) with regard to x_4 using m-fold XBOOLE derivative operations for $m = 1$, visualize the Karnaugh maps, and verify the six relations (4.43), ..., (4.48) of [18] between these three simple derivatives. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for function (4.7) and store the result as object number 1.

2 Prepare VT $\langle x4 \rangle$ as object number 2.

3 Calculate $\frac{\partial f(\mathbf{x})}{\partial x_4}$ as object number 3, and show the Karnaugh map of the result.

4 Calculate $\min_{x_4} f(\mathbf{x})$ as object number 4, and show the Karnaugh map of the result.

5 Calculate $\max_{x_4} f(\mathbf{x})$ as object number 5, and show the Karnaugh map of the result.

6 Verify (4.43) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 4 and 5.

7 Verify (4.44) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 3 and 5.

8 Verify (4.45) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 3 and 4.

9 Verify (4.46) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 5, 4, and 3.

10 Verify (4.47) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 5, 3, and 4.

11 Verify (4.48) of [18] and emphasize the understanding of this relation by comparing Karnaugh map 4, 3, and 5.

Generally, not only one, but several variables of a logic function may change their values simultaneously. The vectorial derivatives describe the observable properties of this approach. There are particular XBOOLE operations `derv`, `minv`, and `maxv` which compute the vectorial derivative, the vectorial minimum, and the vectorial maximum, respectively.

Exercise 4.10 (Vectorial Derivatives). Calculate all three vectorial derivatives of the function (4.7) with regard to (x_3, x_4) using the XBOOLE operations mentioned above, visualize the Karnaugh maps, observe that the vectorial derivatives depend on the same variables as the given function, and verify basic relations between these three vectorial derivatives. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for the function (4.7) and store the result as object number 1.

2 Prepare VT $\langle x3, x4 \rangle$ as object number 2.

3 Calculate $\frac{\partial f(\mathbf{x})}{\partial (x_3, x_4)}$ as object number 3, and show the Karnaugh map of the result.

4 Calculate $\min_{(x_3,x_4)} f(\mathbf{x})$ as object number 4, and show the Karnaugh map of the result.

5 Calculate $\max_{(x_3,x_4)} f(\mathbf{x})$ as object number 5, and show the Karnaugh map of the result.

6 Verify (4.52) of [18].

7 Verify (4.43) of [18].

The application of the Boolean Differential Calculus leads to expressions that include derivative operations. Using relations between derivative operations allows to simplify such expressions. Because simple derivative operations are special cases of vectorial derivative operations, many relations are valid for both of these classes of derivative operations.

Exercise 4.11 (Relations for Simple and Vectorial Derivatives). Verify all 16 relations (4.52), ..., (4.69) of [18], once for simple derivative operations with regard to x_4, and once for vectorial derivative operations with regard to (x_3, x_4). Use basically the function $f(x_1, x_2, x_3, x_4)$ of (4.7) and additionally the function $g(x_1, x_2, x_3, x_4)$ of (4.8),

$$g(x_1, x_2, x_3, x_4) = x_2 x_4 \vee x_1(\overline{x}_3 \vee \overline{x}_2 \overline{x}_4), \tag{4.8}$$

for the relations (4.64), (4.65), and (4.66) of [18]. Study the TVLs and Karnaugh maps of the intermediate results. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4) = 1$ for function (4.7) and store the result as object number 1.

2 Solve the equation $g(x_1, x_2, x_3, x_4) = 1$ for function (4.8) and store the result as object number 2.

3 Prepare VT $\langle x4 \rangle$ as object number 3.

4 Prepare a PRP that evaluates all 16 relations (4.52), ..., (4.69) of [18] based on the functions given as objects number 1 and 2 where the derivative operations are calculated with regard to the variables of object number 3.

5 Execute the PRP of subtask 4 and evaluate the results for simple derivative operations.

6 Delete object number 3 and prepare VT $\langle x3, x4\rangle$ as a new object number 3.

7 Execute the PRP of subtask 4 and evaluate the results for vectorial derivative operations.

Total and partial differential operations contain simple and vectorial derivative operations. The fixing of the direction of change in a differential operation by constant values for the differentials $\mathrm{d}x_i$ leads to an associated simple or vectorial derivative operation. Vice versa, a total and partial differential operation can be built by using all required simple and vectorial derivative operations and associated conjunctions of differentials $\mathrm{d}x_i$.

Exercise 4.12 (Vectorial Derivative Operations $\leftrightarrow$ Vectorial Differential Operations). Calculate all vectorial derivative operations based on the partial differential operations of Exercise 4.6 and verify the results. Build the three differential operations of Exercise 4.6 using the vectorial derivative operations of XBOOLE and verify the results. Use function (4.2) for all calculations. Practical tasks:

1 Reload the TVL-system of Exercise 4.6 as the basis of all further calculations.

2 Prepare TVLs for conjunctions $\overline{\mathrm{d}x_3}\,\overline{\mathrm{d}x_4}$, $\mathrm{d}x_3\,\overline{\mathrm{d}x_4}$, $\overline{\mathrm{d}x_3}\,\mathrm{d}x_4$, and $\mathrm{d}x_3\,\mathrm{d}x_4$, as objects number 30, ..., 33, and VTs of $\langle x3\rangle$, $\langle x4\rangle$, and $\langle x3, x4\rangle$ as objects number 34, 35, and 36.

3 Write a PRP that calculates all vectorial derivatives using the partial differential, given in object number 5, and verify these results using vectorial derivatives of the function in object number 1.

4 Execute the PRP of the previous subtask.

5 Write a PRP that calculates partial differentials of the function in object number 1 using their vectorial derivatives, calculated in the previous PRP, and verify the result.

6 Execute the PRP of the previous subtask.

7 Repeat the subtasks 3 ... 6 for the differential minimum of object number 6.

8 Repeat the subtasks 3 ... 6 for the differential maximum of object number 7.

Both vectorial and m-fold derivative operations of a logic function $f(\mathbf{x})$ are calculated with regard to a subset of variables $\mathbf{x}_0 \subseteq \mathbf{x}$. The

definition and the semantics of these two kinds of derivative operations are completely different. The m-fold derivative operations describe properties of $f(\mathbf{x})$ in whole subspaces and do not depend on variables of the vector $\mathbf{x}_0$. The XBOOLE Monitor offers operations `derk`, `mink`, and `maxk` which calculate the m-fold derivative, the m-fold minimum, and the m-fold maximum, respectively. The Δ-operation can be calculated using the XBOOLE operation `syd` of the results of `mink` and `maxk`.

Exercise 4.13 (M-fold Derivatives). Calculate all four m-fold derivatives of the function

$$f(x_1, x_2, x_3, x_4, x_5) = x_1(x_2 \vee \overline{x}_4) \vee x_2\overline{x}_3x_5 \vee \overline{x}_2\overline{x}_3x_4\overline{x}_5 \tag{4.9}$$

with regard to (x_4, x_5) using the XBOOLE operations mentioned above, visualize the Karnaugh maps, observe that the m-fold derivative operation depends on less variables than the given function, and verify partial order relations between the m-fold minimum, the given function, and the m-fold maximum. Practical tasks:

1 Solve the equation $f(x_1, x_2, x_3, x_4, x_5) = 1$ for the function (4.9) and store the result as object number 1.

2 Prepare VT $\langle x4, x5\rangle$ as object number 2.

3 Calculate $\frac{\partial^2 f(\mathbf{x})}{\partial x_4 \partial x_5}$ as object number 3, and show the Karnaugh map of the result.

4 Calculate $\min^2_{(x_4,x_5)} f(\mathbf{x})$ as object number 4, and show the Karnaugh map of the result.

5 Calculate $\max^2_{(x_4,x_5)} f(\mathbf{x})$ as object number 5, and show the Karnaugh map of the result.

6 Calculate $\Delta_{(x_4,x_5)} f(\mathbf{x})$ as object number 6, and show the Karnaugh map of the result.

7 Calculate the simple minimum and the simple maximum of $f(\mathbf{x})$ with regard to x_4 and verify (4.88) and (4.90) of [18].

Similar to simple and vectorial derivative operations, there are also many relations between m-fold derivative operations. The knowledge of these relations helps to simplify expressions which include m-fold derivative operations. The verification of these relations in the following exercise supports the understanding of m-fold derivative operations.

Exercise 4.14 (Relations for m-fold Derivatives). Calculate all four m-fold derivative operations of the functions $f(\mathbf{x})$ (4.9) and $g(\mathbf{x})$,

$$g(x_1, x_2, x_3, x_4, x_5) = x_2(x_3 \vee \overline{x}_4x_5) \vee x_1x_4(\overline{x}_3 \vee x_5) \tag{4.10}$$

with regard to (x_4, x_5), visualize the Karnaugh maps, and verify the 16 relations (4.98), ..., (4.113) of [18] for these m-fold derivatives. Practical tasks:

1 Write a PRP that solves this task completely.

2 Execute this PRP and verify the 16 relations mentioned above.

The m-fold differential contains only the associated m-fold derivative. All the other m-fold differential operations are superpositions of several associated m-fold derivative operations. Thus, all four m-fold differential operations of a logic function can be extracted from their m-fold differential operation, see (4.118), ..., (4.121) of [18].

Exercise 4.15 (M-fold Differential Operations $\rightarrow$ m-fold Derivative Operations). Calculate all four twofold derivative operations based on the 2-fold differential operations of Exercise 4.7 and verify the results. Practical tasks:

1 Reload the TVL-system of Exercise 4.7 as the basis of all further calculations.

2 Write a PRP that uses $f(\mathbf{x})$ of object number 1, $\mathrm{d}^2_{(x_3,x_4)} f(\mathbf{x})$ of object number 17, $\mathrm{Min}^2_{(x_3,x_4)} f(\mathbf{x})$ of object number 27, $\mathrm{Max}^2_{(x_3,x_4)} f(\mathbf{x})$ of object number 37, and $\vartheta_{(x_3,x_4)} f(\mathbf{x})$ of object number 41 for the calculation of $\frac{\partial^2 f(\mathbf{x})}{\partial x_3 \partial x_4}$, $\min^2_{(x_4,x_5)} f(\mathbf{x})$, $\max^2_{(x_4,x_5)} f(\mathbf{x})$, and $\Delta_{(x_4,x_5)} f(\mathbf{x})$, and verify these results by means of the direct m-fold derivative operations of XBOOLE.

3 Execute this PRP and evaluate the results.

For extended studies it is suggested to execute the transformation in the reverse direction of Exercise 4.15 based on (4.114), ..., (4.117) of [18].

3. Applications

Differentials of variables are the fundament of the Boolean Differential Calculus. These differentials $\mathrm{d}x_i$ allow to solve several problems of analysis.

We assume that $F(\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathrm{d}\mathbf{s}) = 1$ describes the behavior of an asynchronous finite-state machine, where $\mathbf{x}$ is the input vector, $\mathbf{y}$ the output vector, and $\mathbf{s}$ codes the states. New values of $\mathbf{x}$ cause a sequence of changes of the states $\mathbf{s}$ until a stable state will be reached. Hence, designer of asynchronous finite-state machines are interested in these stable states. The stable states are visible by loops in the behavioral

graph. The input and output values have no influence on the existing stable states; hence, the behavioral function can be simplified using an m-fold maximum.

Exercise 4.16 (Stable States). Calculate the stable states of the asynchronous finite-state machine given by the following TVL in ODA-form.

$$F(x, y, \mathbf{s}, \mathrm{d}\mathbf{s}) =$$

x	y	s_1	s_2	s_3	$\mathrm{d}s_1$	$\mathrm{d}s_2$	$\mathrm{d}s_3$
0	0	–	0	0	1	1	0
0	0	1	1	–	0	0	0
1	0	1	1	0	0	1	1
–	1	–	0	1	0	0	1
0	1	0	1	0	0	1	1
1	1	0	1	0	1	1	1
1	0	1	1	1	0	1	0
1	1	–	0	0	0	0	0
0	0	0	1	1	0	1	0
1	0	0	1	1	0	0	0

Practical tasks:

1 Store the given TVL as object number 1.

2 Prepare a VT $\langle x, y \rangle$ as object number 2 and solve the equation $\overline{\mathrm{d}s_1}\,\overline{\mathrm{d}s_2}\,\overline{\mathrm{d}s_3} = 1$ and store the result as object number 3.

3 Calculate $F(\mathbf{s}, \mathrm{d}\mathbf{s}) = \max^2_{(x,y)} F(x, y, \mathbf{s}, \mathrm{d}\mathbf{s})$ as object number 4.

4 Calculate the stable states using an intersection and an m-fold maximum.

The high speed of asynchronous finite-state machines must be based on the explicit analysis of critical transitions between different states. A transition can be critical when more than one variable must change its value at the same time. These edges can be determined in the behavioral graph similarly to the selection of the loops for stable states.

Exercise 4.17 (Critical Transition). Calculate all possible critical transitions of the asynchronous finite-state machine given in Exercise 4.16. Practical tasks:

1 Reload the TVL-system of Exercise 4.16 as the basis of all further calculations.

2 Solve the equation $\mathrm{d}s_1\,\mathrm{d}s_2 \vee \mathrm{d}s_1\,\mathrm{d}s_3 \vee \mathrm{d}s_2\,\mathrm{d}s_3 = 1$ as object number 10.

3 Find the potentially critical transitions of the given asynchronous finite-state machine 1.

Functional hazards of combinational circuits may lead to malfunctions of controlled sequential circuits. A static functional hazard appears if the

input of the function $f(\mathbf{x})$ changes from $\mathbf{x} = \mathbf{c}_0$ to $\mathbf{x} = \mathbf{c}_1$, the function is constant for these vectors, $f(\mathbf{c}_0) = f(\mathbf{c}_1)$, and intermediately the inverse value $\overline{f(\mathbf{c}_0)}$ occurs as the value of the function. [3] proved that all static functional hazards of a logic function are given by the solution of the equation

$$\vartheta f(\mathbf{x}) \cdot \overline{\mathrm{d}f(\mathbf{x})} = 1. \tag{4.11}$$

Exercise 4.18 (Static Functional Hazard). Calculate all static functional hazards of function (4.2) restricted to all possible changes of values of x_3 and x_4. Practical tasks:

1 Reload the TVL-system of Exercise 4.7, minimize object number 41 that represents $\vartheta_{(x_3,x_4)} f(\mathbf{x})$ and write down this TVL.

2 Reload the TVL-system of Exercise 4.6 where $\mathrm{d}_{(x_3,x_4)} f(\mathbf{x})$ is stored as object number 5.

3 Recreate the TVL of $\vartheta_{(x_3,x_4)} f(\mathbf{x})$ as object number 30.

4 Calculate the required static functional hazard based on (4.11).

5 Are there static functional hazards for the change of either only x_3 or only x_4? Substantiate this observation.

In order to broaden the understanding of both the differential operations and the hazards of logic functions, we suggest to solve Exercise 4.18 without the restriction to variables x_3 and x_4. All static functional hazards of function (4.2) can be calculated by means of a PRP. More convenient and therefore suggested is the use of an XBOOLE library C-program as introduced in Chap. 1, Sect. 7.

Now we come back to some concepts that have been explored in Chap. 2 by elementary means. We will see that the Boolean Differential Calculus will allow us to deal with these concepts in a very understandable and efficient way.

The differential operations are very powerful for compact theoretical explorations. However, the additional variables $\mathrm{d}x_i$ complicate practical calculations. In contrast to the differential operations, derivative operations require not more and in many cases even less variables. Therefore we focus in the rest of this section on derivative operations.

Expressions of logic functions $f(\mathbf{x})$ may include a variable x_i although the function does not depend on x_i. Whether a function really depends on x_i may be found out using the simple derivative with regard to x_i. The function $f(\mathbf{x})$ is independent on x_i if (4.12) holds:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0. \tag{4.12}$$

If a function does not depend on the variable x_i, the two subfunctions $f(\mathbf{x}_0, x_i = 0)$ and $f(\mathbf{x}_0, x_i = 1)$ are equal to each other, hence, the simplified function $f(\mathbf{x}_0)$ can be created using the simple maximum of $f(\mathbf{x})$ with regard to x_i.

Exercise 4.19 (Dependency on Variables). Check on which variables the given functions $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, and $f_3(\mathbf{x})$ depend:

$$f_1(\mathbf{x}) = (x_1x_2 \oplus x_3x_4) \vee x_3\overline{x}_4 \vee x_1\overline{x}_2 \vee x_1x_3, \tag{4.13}$$

$$f_2(\mathbf{x}) = x_2\overline{x}_4 \vee \overline{x}_1 \cdot ((x_3 \oplus \overline{x}_2) \vee \overline{(x_2 \cdot x_3)}), \tag{4.14}$$

$$f_3(\mathbf{x}) = x_1x_2 \vee x_3x_4 \vee (x_1 \oplus x_4) \vee (x_2 \oplus x_3). \tag{4.15}$$

Simplify the functions as much as possible and verify the results. Practical tasks:

1. Prepare the functions $f_1(\mathbf{x})$ (4.13), $f_2(\mathbf{x})$ (4.14), and $f_3(\mathbf{x})$ (4.15) as objects number 1, 2, and 3, respectively.
2. Prepare VTs $\langle x_1 \rangle$, $\langle x_2 \rangle$, $\langle x_3 \rangle$, and $\langle x_4 \rangle$ as objects number 4, 5, 6, and 7.
3. Calculate the simple derivatives of each function (4.13), (4.14), and (4.15) with regard to each variable. Are there independences?
4. Simplify the three functions as much as possible and verify the result.

Logic functions can possess certain properties which have an influence on their possibilities for applications. Derivative operations allow to check whether a given function possesses such a special property.

A logic function $g(\mathbf{x})$ is called *dual* to the function $f(\mathbf{x})$ if $g(\mathbf{x}) = \overline{f(\overline{\mathbf{x}})}$. Hence, there is a dual function $g(\mathbf{x})$ for each function $f(\mathbf{x})$. In some case the dual function is equal to the given function itself. Logic functions $f(\mathbf{x})$ that satisfy (4.16) are called *self-dual*:

$$f(\mathbf{x}) = \overline{f(\overline{\mathbf{x}})}. \tag{4.16}$$

Exercise 4.20 (Self-dual Function). A logic function is self-dual if and only if

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 1. \tag{4.17}$$

Prove this theorem, identify the number of self-dual functions of n variables, and verify whether given functions $f_1(\mathbf{x})$ (4.18) and $f_2(\mathbf{x})$ (4.19) are self-dual functions:

$$f_1(\mathbf{x}) = x_1\overline{x}_2 \vee x_2\overline{x}_3 \vee x_3\overline{x}_4 \vee x_4\overline{x}_1, \tag{4.18}$$

$$f_2(\mathbf{x}) = x_1(x_3 \oplus x_4) \vee x_3x_4\overline{(x_1x_2)} \vee \overline{x}_1\overline{x}_2x_3x_4. \tag{4.19}$$

Practical tasks:

1 Prove the theorem.

2 How much self-dual functions of n variables exist?

3 Prepare the functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ as objects number 1 and 2, and the VT $\langle x_1, x_2, x_3, x_4 \rangle$ as object number 3.

4 Apply vectorial derivatives in order to verify whether given functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are self-dual functions.

A logic function $f(x_i, x_j, \mathbf{x}_0)$ is called *symmetric* with regard to (x_i, x_j) if the exchange of the input values between x_i and x_j does not change the function value. A function possesses this property if (4.20) holds for each $\mathbf{x}$:

$$(x_i \oplus x_j) \wedge \frac{\partial f(x_i, x_j, \mathbf{x}_0)}{\partial (x_i, x_j)} = 0. \tag{4.20}$$

Exercise 4.21 (Symmetric Function). Check whether there are pairs of variables, for which the function

$$f(\mathbf{x}) = (x_1 \oplus x_2 \oplus \overline{x}_3\overline{x}_4 \oplus x_2x_3\overline{x}_4) \vee x_1x_2 \tag{4.21}$$

is symmetric. If there are such pairs, express the function by an expression that emphasizes this property. Remember: derivative operations will be executed by the XBOOLE Monitor with regard to the variables an object is depending on. Hence, any derivative operation can be executed for a TVL of the function with regard to a TVL that defines the set of variables for the derivative operation. Practical tasks:

1 Write a PRP that finds pairs of variables in which function (4.21) is symmetric.

2 Execute this PRP and check for pairs of symmetric variables.

3 Express the function by an expression that emphasizes the symmetry property for each pair of symmetric variables that has been found.

In [18] the interesting class of *monotone functions* has been defined globally with regard to all its variables. For practical applications it is helpful to define several more restricted classes of monotone functions. A logic function $f(x_i, \mathbf{x}_0)$ is *monotonously increasing with regard to* x_i if

$$f(x_i = 0, \mathbf{x}_0) \leq f(x_i = 1, \mathbf{x}_0), \tag{4.22}$$

or *monotonously decreasing with regard to* x_i if

$$f(x_i = 0, \mathbf{x}_0) \geq f(x_i = 1, \mathbf{x}_0). \tag{4.23}$$

For a function monotonously increasing with regard to x_i exists an expression that does not include negated variables $\overline{x}_i$. Vice versa a function monotonously decreasing with regard to x_i can be expressed without using variables x_i. If only one type of a variable (negated or not negated) appears in an expression, the realization of an associated circuit can be simplified. Using simple derivatives, the monotone properties of a function $f(x_i, \mathbf{x}_0)$ with regard to x_i can be checked. A logic function $f(x_i, \mathbf{x}_0)$ is *monotonously increasing with regard to* x_i if

$$\overline{x}_i f(x_i, \mathbf{x}_0) \frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i,} = 0, \tag{4.24}$$

or *monotonously decreasing with regard to* x_i if

$$x_i f(x_i, \mathbf{x}_0) \frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i,} = 0. \tag{4.25}$$

Exercise 4.22 (Monotone Function). In the expression (4.26) of $f(\mathbf{x})$ each variable appears both negated and not negated:

$$f(\mathbf{x}) = \overline{x}_2 x_3 \overline{x}_4 \vee x_4(x_1 \vee \overline{x}_2) \vee \overline{x}_1 \overline{x}_3 (x_2 \oplus \overline{x}_4). \tag{4.26}$$

Check by means of (4.24) and (4.25) whether function (4.26) is monotonously increasing or monotonously decreasing, separately with regard to each variable of the set $\{x_1, x_2, x_3, x_4\}$. Practical tasks:

1 Write a PRP that detects all monotone properties of the function (4.26) with regard to each of its variables.

2 Execute this PRP and check for monotone properties.

3 Express the function in such a way that only one type of monotone variables occurs.

4 Verify the simplified expression of the function.

Still stronger than monotony, linearity can be used to simplify a function. As defined in [18], a logic function $f(\mathbf{x})$ can be linear with regard to all of its variables. Unfortunately very few logic function are completely linear. In order to simplify a function it is already helpful, if the function is linear with regard to a single variable. A logic function $f(x_i, \mathbf{x}_0)$ is *linear* with regard to x_i if

$$f(x_i = 0, \mathbf{x}_0) = x_i \oplus f(\mathbf{x}_0). \tag{4.27}$$

This property can be checked using a simple derivative. A logic function $f(x_i, \mathbf{x}_0)$ is *linear* with regard to x_i if

$$\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i} = 1. \tag{4.28}$$

Exercise 4.23 (Linear Function). In expression (4.29) of $f(\mathbf{x})$ it is not directly visible whether the function is linear with regard to certain variables:

$$f(\mathbf{x}) = \overline{x}_1(\overline{x}_3 \oplus \overline{x}_2 x_4) \vee x_1(x_2 \oplus \overline{x}_3)x_4 \vee x_1\, x_3\, \overline{x}_4. \tag{4.29}$$

Check by means of (4.28) whether function (4.29) is linear, separately for each variable of the set $\{(x_1, x_2, x_3, x_4)\}$. Practical tasks:

1 Write a PRP that detects all linearity properties of the function (4.29) with regard to each of its variables.

2 Execute this PRP and check for linearity of $f(x_i, \mathbf{x}_0)$ with regard to each of its variables. Use the view of Karnaugh map to check whether a calculated function is equal to 1.

3 If $f(x_i, \mathbf{x}_0)$ is linear with regard to x_i then use definition (4.27) in order to calculate the function $f(\mathbf{x}_0)$ independent on x_i.

4 Create the simplified expression of the function and verify it.

Many applications use the solution of an equation with regard to variables in combination with the uniqueness of such variables. The basic equation is mostly given or can easily be transformed into a homogeneous form having a constant right-hand side $c \in \{0, 1\}$. An equation

$$f(\mathbf{x}, y) = c \tag{4.30}$$

can be solved with regard to y if there is a function $y = g(\mathbf{x})$ with $f(\mathbf{x}, g(\mathbf{x})) = c$. Equation (4.30) is unique with regard to y if there is no $\mathbf{x} = \mathbf{c}_x$ having more than one $y = c_y$ with $f(\mathbf{c}_x, c_y) = c$.

By using simple derivative operations, it can be checked which property holds. A characteristic equation $f(\mathbf{x}, y) = 1$ can be solved with regard to y if

$$\max_y f(\mathbf{x}, y) = 1, \tag{4.31}$$

and it is unique with regard to y if

$$\min_y f(\mathbf{x}, y) = 0. \tag{4.32}$$

Both of these properties hold for $f(\mathbf{x}, y) = 1$ with regard to y if

$$\frac{\partial f(\mathbf{x}, y)}{\partial y} = 1. \tag{4.33}$$

A restrictive equation $f(\mathbf{x}, y) = 0$ can be solved with regard to y if

$$\min_y f(\mathbf{x}, y) = 0, \tag{4.34}$$

and it is unique with regard to y if

$$\max_y f(\mathbf{x}, y) = 1. \tag{4.35}$$

Both of these properties hold for $f(\mathbf{x}, y) = 0$ with regard to y if

$$\frac{\partial f(\mathbf{x}, y)}{\partial y} = 1. \tag{4.36}$$

Notice: the conditions (4.33) and (4.36) are the same for both the characteristic equation and the restrictive equation while the conditions for solvability and uniqueness with regard to y are different.

We assume that an equation is solvable with regard to y in a unique way. Then the solution functions can be calculated as

$$g(\mathbf{x}) = \max_y(y \sim f(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 1, \tag{4.37}$$

$$g(\mathbf{x}) = \max_y(y \oplus f(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 0. \tag{4.38}$$

If the equation $f(\mathbf{x}, g(\mathbf{x})) = c$ is solvable with regard to y, but not in a unique way, then a set of solution functions exists (see [18], pages 277 and 278). All functions of such a set must be equal to 1 for the ON-set function $q(\mathbf{x})$, and must be equal to 0 for the OFF-set function $r(\mathbf{x})$, respectively. These mark functions can be calculated by

$$q(\mathbf{x}) = \max_y(yf(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 1, \tag{4.39}$$

$$r(\mathbf{x}) = \max_y(\overline{y}f(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 1, \tag{4.40}$$

$$q(\mathbf{x}) = \max_y(\overline{y}f(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 0, \tag{4.41}$$

$$r(\mathbf{x}) = \max_y(yf(\mathbf{x}, y)) \quad \text{for } f(\mathbf{x}, y) = 0. \tag{4.42}$$

Exercise 4.24 (Uniqueness and Solvability with Regard to a Variable). Check the following equations for uniqueness and solvability with regard to y. Calculate all solution functions $y = g(\mathbf{x})$ and verify them:

$$\overline{x}_1 x_3 \overline{y} \vee \overline{x}_2(x_3 \oplus y) \vee x_1 \overline{x}_3 y = 1, \tag{4.43}$$

$$\overline{x}_1 x_3 \overline{y} \vee \overline{x}_2(x_3 \oplus y) \vee x_2 y \cdot (x_1 \vee \overline{x}_3) = 1, \tag{4.44}$$

$$x_2 \overline{x}_3 \overline{y} \vee \overline{x}_2(x_3 \oplus y) \vee \overline{x}_1 x_2 \overline{x}_3 = 0, \tag{4.45}$$

$$x_1 x_2 \overline{y} \vee \overline{x}_2(x_3 \oplus y) = 0. \tag{4.46}$$

Practical tasks:

1 Prepare the functions of the left-hand sides of (4.43), (4.44), (4.45), (4.46) as objects number 1, 2, 3, and 4, respectively. Assign the variable y to object number 4.

2 Which equation is unique with regard to y? Evaluate the calculated Karnaugh maps for this equation.

3 Which equation is unique with regard to y? Evaluate the calculated Karnaugh maps for this equation.

4 Which equation is solvable with regard to y? Evaluate the calculated Karnaugh maps for this equation.

5 Calculate the solution functions for all equations that are unique with regard to y.

6 Verify the solution of the equations which are unique with regard to y.

7 Calculate the mark functions $q(\mathbf{x})$ and $r(\mathbf{x})$ of the solution sets for all equations that are solvable with regard to y, but not uniquely.

8 Calculate all functions of the solution sets based on the created mark functions $q(\mathbf{x})$ and $r(\mathbf{x})$.

9 Verify whether these function solve the associated equation.

4. Solutions

Exercise 4.1.

1 11 edges.
3 cycle lengths 2 and 5.

2 2 components (4 edges and 7 edges).
4 cycle lengths 0, 0, and 6.

Exercise 4.2.

1
```
tin 1 10        tin 1 11        tin 1 12        vtin 1 13       isc 14 12 14
x1 dx1 xf1.     x2 dx2 xf2.     x3 dx3 xf3.     dx1 dx2 dx3.    maxk 14 13 15
000, 110,       000, 110,       000, 110,       isc 1 10 14     obbc 15 2
101, 011.       101, 011.       101, 011.       isc 14 11 14
```

2 TVL 1 as shown in Table 4.1.

3 Both graphs are identical.

4
```
tin 1 20        tin 1 21        tin 1 22        vtin 1 23       isc 24 22 24
x1 dx1 xf1.     x2 dx2 xf2.     x3 dx3 xf3.     xf1 xf2 xf3.    maxk 24 23 25
000, 110,       000, 110,       000, 110,       isc 2 20 24     obbc 25 3
101, 011.       101, 011.       101, 011.       isc 24 21 24
```

5 The result of SYD 1 3 4 is an empty TVL that indicates the correctness of both transformations.

Exercise 4.3.

1 x1&(/x2+/x3)#/x1&(/x2=x3)
2 (x1#dx1)&(/(x2#dx2)+ /(x3#dx3))#/(x1#dx1)& (/(x2#dx2)=(x3#dx3))
3 30 edges.
4 25 edges.
5 55 edges.
6 The empty TVL in object number 6 confirms the correctness.

Exercise 4.4.

1 x1&(/x2+/x3)# /x1&(/x2=x3)
2 2: ⟨x1,x2,x3⟩ 3: ⟨xf1,xf2,xf3⟩
3 CCO 1 2 3 4

4 tin 1 20 x1 dx1 xf1. 000, 110, 101, 011.
tin 1 21 x2 dx2 xf2. 000, 110, 101, 011.
tin 1 22 x3 dx3 xf3. 000, 110, 101, 011.
vtin 1 23 xf1 xf2 xf3. isc 10 20 24 isc 24 21 24
isc 24 22 24 maxk 24 23 25 obbc 25 11

5 30 values 1.
6 25 values 1.
7 55 values 1.
8 The empty TVL in object number 8 confirms the correctness.

Exercise 4.5.

1 tin 1 30 /ODA dx1 dx2 dx3. − − −. isc 1 30 31
2 Karnaugh map of object 31 shows 40 values 1 and depends on 6 variables ($x1, x2, x3, \mathrm{d}x1, \mathrm{d}x2, \mathrm{d}x3$).
3 dif 6 31 32 – the empty TVL in object number 32 confirms the correctness of (4.8).

4 dif 31 7 33 – the empty TVL in object number 33 confirms the correctness of (4.10).

5 dif 6 1 34 – dif 1 7 35 – the empty TVLs in objects number 34, and 35 confirm the correctness of (4.11).

Exercise 4.6.

1 (x1#x2)&x3+ x1&x4+ /(x1&x2)& /(x3+x4)
2 vtin 1 2 x3 x4. vtin 1 3 xf3 xf4. cco 1 2 3 4 orth 4 4
3 tin 1 12 /ODA x3 xf3 dx3. 000, 011, 101, 110. tin 1 13 /ODA x4 xf4 dx4. 000, 011, 101, 110.
vtin 1 14 xf3 xf4. isc 10 12 11 isc 11 13 11 maxk 11 14 11 obbc 11 11

4 Four isolated subgraphs possess 6, 6, 8, and 0 edges, totally 20.

5 Four isolated subgraphs possess 1, 9, 4, and 16 edges, totally 30.

6 Four isolated subgraphs possess 7, 15, 12, and 16 edges, totally 50.

7 The empty TVL in object number 20 confirms the correctness.

8 The empty TVL in object number 21 confirms the correctness.

9 The empty TVL in object number 22 confirms the correctness.

10 The logic function of object number 1 is defined in the XBOOLE-monitor in the same Boolean space as the partial differential operations, thus the given function depends implicitly on the same variables.

Exercise 4.7.

1 (x1#x2)&x3+x1&x4+/(x1&x2)&/(x3+x4) 2 4 vtin and 2 tin commands.

3 cco 1 2 3 10 syd 1 10 11 maxk 12 3 13 orth 14 14 isc 15 7 16
orth 10 10 isc 11 6 12 cco 13 4 5 14 syd 13 14 15 maxk 16 5 17.

4 (x3): 4 edges in 8 subgraphs; (x3, x4): 8 edges in 4 subgraphs.

5 PRP analogously to subtask 3 using `isc` and objects 20 to 27.

6 (x3): 18 edges in 8 subgraphs; (x3, x4): 28 edges in 4 subgraphs.

7 PRP analogously to subtask 3 using `uni` and objects 30 to 37.

8 syd 23 33 40 – (x3): 22 edges in 8 subgraphs;
syd 27 37 41 – (x3, x4): 52 edges in 4 subgraphs.

9 (x3): 4 edges in 8 subgraphs; (x3, x4): 24 edges in 4 subgraphs.

10 The functions are equal to each other.

11 The functions are different.

12 dif 27 23 44 dif 1 33 46 All four inequations are satisfied.
dif 23 1 45 dif 33 37 47

Exercise 4.8.

1 (x1&/x2#/x2&x3&x4)+/x1&x2&x4 2 2: $\langle x4 \rangle$

3 3: 8 of 16 values 1 4 4: 4 of 8 values 1. 5 5: 2 of 16 values 1.

6 6: 1 of 8 values 1. 7 7: 10 of 16 values 1. 8 8: 5 of 8 values 1.

9 syd 4 6 10 – syd 10 8 10 – the empty TVL in object number 10 confirms the correctness of (4.36).

10 dif 6 1 11 – dif 1 8 12 – the empty TVLs in objects number 11 and 12 confirm the correctness of (4.40).

Exercise 4.9.

1 (x1&/x2#/x2&x3&x4)+/x1&x2&x4 2 2: $\langle x4 \rangle$

3 derk 1 2 3 4 mink 1 2 4 5 maxk 1 2 5
4 of 8 values 1. 3 of 8 values 1. 7 of 8 values 1.

6 dif 4 5 6 – the empty TVL 6 confirms the correctness of (4.43).

7 dif 3 5 7 – the empty TVL 7 confirms the correctness of (4.44).

8 isc 4 3 8 – the empty TVL 8 confirms the correctness of (4.45).

9 dif 5 4 9 – syd 9 3 10 – TVL 9 is equal to TVL 3 – the empty TVL 10 confirms the correctness of (4.46).

10 dif 5 3 11 – syd 11 4 12 – TVL 11 is equal to TVL 4 – the empty TVL 12 confirms the correctness of (4.47).

11 uni 4 3 13 – syd 13 5 14 – TVL 13 is equal to TVL 5 – the empty TVL 14 confirms the correctness of (4.48).

Exercise 4.10.

1 (x1&/x2#/x2&x3&x4)+/x1&x2&x4 2 2: $\langle x3, x4\rangle$

3 derv 1 2 3 4 minv 1 2 4 5 maxv 1 2 5
8 of 16 values 1. 2 of 16 values 1. 10 of 16 values 1.

6 dif 4 1 6 – dif 1 5 7 – the empty TVLs 6 and 7 confirm the correctness of (4.52).

7 syd 3 4 8 – syd 8 5 8 – the empty TVL 8 confirms the correctness of (4.53).

Exercise 4.11.

1 (x1&/x2#/x2&x3&x4)+ 2 x2&x4+ 3 3: $\langle x4\rangle$
/x1&x2&x4 x1&(/x3+/x2&/x4)

4
derv 1 3 4	syd 12 4 12	derv 17 3 17	syd 1 2 20	uni 6 9 27
minv 1 3 5	syd 5 4 13	syd 17 4 17	derv 20 3 20	syd 26 27 28
maxv 1 3 6	syd 13 6 13	cpl 1 18	syd 4 7 21	derv 4 3 29
derv 2 3 7	dif 6 4 14	maxv 18 3 18	syd 20 21 22	minv 5 3 30
minv 2 3 8	syd 14 5 14	cpl 18 18	isc 1 2 23	syd 30 5 30
maxv 2 3 9	dif 6 5 15	syd 18 5 18	minv 23 3 23	maxv 6 3 31
isc 5 4 10	syd 15 4 15	cpl 1 19	isc 5 8 24	syd 31 6 31
syd 4 6 11	uni 5 4 16	minv 19 3 19	syd 23 24 25	
syd 11 5 11	syd 16 6 16	cpl 19 19	uni 1 2 26	
syd 5 6 12	cpl 1 17	syd 19 6 19	maxv 26 3 26	

5 The empty TVLs 10 to 19, 22, 25, and 28 to 31 confirm the correctness of all 16 simple relations with regard to x_4. 6 3: $\langle x3, x4\rangle$

7 The empty TVLs 10 to 19, 22, 25, and 28 to 31 confirm the correctness of all 16 vectorial relations with regard to (x_3, x_4).

Exercise 4.12.

1 lds e406_res.sdt.

2
tin 1 30 /oda	tin 1 32 /oda	vtin 1 34	3 isc 5 31 40
dx3 dx4.	dx3 dx4.	x3.	maxk 40 30 40
00.	01.	vtin 1 35	derv 1 34 41
tin 1 31 /oda	tin 1 33 /oda	x4.	syd 40 41 42
dx3 dx4.	dx3 dx4.	vtin 1 36	isc 5 32 43
10.	11.	x3 x4.	maxk 43 30 43

derv 1 35 44
syd 43 44 45
isc 5 33 46
maxk 46 30 46
derv 1 36 47
syd 46 47 48

4 The empty TVLs 42, 45, and 48 confirm the correctness of the extracted vectorial derivatives.

5 isc 41 31 50 isc 47 33 52 syd 53 52 53
isc 44 32 51 syd 50 51 53 syd 53 5 54

6 The empty TVL 54 confirms the correctness of the composed partial differential.

7 The execution results of analogous PRPs confirm the correctness for the partial differential minimum.

8 The execution results of analogous PRPs confirm the correctness for the partial differential maximum.

Exercise 4.13.

1 x1&(x2+/x4)+x2&/x3&x5+ /x2&/x3&x4&/x5

2 2: $\langle x4, x5\rangle$

3 derk 1 2 3
2 of 8 values 1.

4 mink 1 2 4
2 of 8 values 1.

5 maxk 1 2 5
6 of 8 values 1.

6 syd 4 5 6
4 of 8 values 1.

7 vtin 1 10
x4.
mink 1 10 11

maxk 1 10 12
dif 4 11 13
dif 11 1 14

dif 1 12 15
dif 12 5 16

The empty TVLs 13, 14, 15, and 16 confirm the correctness.

Exercise 4.14.

1
space 32 1
avar 1
x1 x2 x3 x4 x5.
sbe 1 1
x1&(x2+/x4)+
x2&/x3&x5+
/x2&/x3&x4&/x5.
sbe 1 2
x2&(x3+/x4&x5)+
x1&x4&(/x3+x5).
vtin 1 3
x4 x5.

derk 1 3 4
mink 1 3 5
maxk 1 3 6
syd 5 6 7
derk 2 3 8
mink 2 3 9
maxk 2 3 10
syd 9 10 11
dif 5 6 21
dif 7 6 22
isc 5 7 23
dif 6 5 24

syd 24 7 25
dif 6 7 26
syd 26 5 27
uni 5 7 28
syd 28 6 29
cpl 1 30
derk 30 3 31
mink 30 3 32
maxk 30 3 33
syd 32 33 34
syd 31 4 35
syd 34 7 36

cpl 33 37
syd 37 5 38
cpl 32 39
syd 39 6 40
syd 1 2 41
derk 41 3 42
syd 4 8 43
syd 42 43 44
isc 1 2 45
mink 45 3 46
isc 5 9 47
syd 46 47 48

uni 1 2 49
maxk 49 3 50
uni 6 10 51
syd 50 51 52
mink 7 3 53
maxk 7 3 54
syd 53 54 55
mink 4 3 56
syd 56 4 57
maxk 5 3 58
syd 58 5 59

2 The empty TVLs 21, 22, 23, 25, 27, 29, 35, 36, 38, 40, 44, 48, 52, 55, 57, and 59 confirm the correctness of all 16 relations between m-fold derivative operations with regard to (x_4, x_5).

Exercise 4.15.

1 lds e407_res.sdt.

2
tin 1 50
dx3 dx4.
11.
vtin 1 51
x3 x4.

isc 17 50 60
maxk 60 50 61
derk 1 51 62
syd 61 62 63

isc 27 50 70
maxk 70 50 71
mink 1 51 72
syd 71 72 73

isc 37 50 80
maxk 80 50 81
maxk 1 51 82
syd 81 82 83

isc 41 50 90
maxk 90 50 91
syd 72 82 92
syd 91 92 93

3 The empty TVLs 63, 73, 83 and 93 confirm the correctness of the extracted twofold derivative operations.

Exercise 4.16.

1 1: given TVL of 10 rows.

2 2: $\langle x, y\rangle$
3: /ds1&/ds2&/ds3=1

3 maxk 1 2 4

4 isc 4 3 5
maxk 5 3 6

There are 5 stable states.

Exercise 4.17.

```
1 lds e416_res.sdt
2 10: ds1&ds2+ds1&ds3+ds2&ds3=1
3 isc 1 10 11
```

There are five potentially critical transitions, four of them change two state variables, and the last even all three state variables.

Exercise 4.18.

```
1 lds e407_res.sdt
  obbc 41 50
  50:
  x1 x2 x3 x4 dx3 dx4
  01-110
  00-010
  0-1-11
  0-0--1
  11---1.
```

```
2 lds e406_res.sdt.
3 tin 1 30
    x1 x2 x3 x4 dx3 dx4.
    01-110
    00-010
    0-1-11
    0-0--1
    11---1.
```

```
4 cpl 5 31
  isc 30 31 32
  There are 4 hazards.
```

5 No hazard for $dx_3\overline{dx}_4$ or $\overline{dx}_3dx_4$ was calculated. In a subspace of size 2 no hazard can appear.

Exercise 4.19.

```
1 space 32 1
  avar 1
  x1 x2 x3 x4.
```

```
sbe 1 1
(x1&x2#x3&x4)+
x3&/x4+
x1&/x2+x1&x3.
```

```
sbe 1 2
x2&/x4+/x1&
x((x3#/x2)+
x/(x2&x3)).
```

```
sbe 1 3
x1&x2+
x3&x4+
(x1#x4)+
(x2#x3).
```

```
2 4: ⟨x1⟩
  5: ⟨x2⟩
  6: ⟨x3⟩
  7: ⟨x4⟩
```

```
3 derk 1 5 12
  derk 1 7 14
  derk 2 6 23
```

Empty TVLs 12, 14, and 23 confirm that f_1 does not depend on x_2 and x_4, and f_2 does not depend on x_3. Function f_3 depends on all variables.

```
4 maxk 1 5 15
  maxk 1 7 15
  syd 15 1 16
```

```
maxk 2 6 25
syd 25 2 26
```

The empty TVLs 16 and 26 confirm the executed simplification for f_1 on 15 and f_2 on 25.

Exercise 4.20.

1 $f(\mathbf{x}) = \overline{f(\overline{\mathbf{x}})}$
$f(\mathbf{x}) \oplus f(\overline{\mathbf{x}}) = \overline{f(\overline{\mathbf{x}})} \oplus f(\overline{\mathbf{x}})$
$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 1$

2 The vectorial derivative defines relations between pairs of function values such that 2^{n-1} self-dual functions of n variables exist.

```
3 1: x1&/x2+x2&/x3+x3&/x4
  x4&/x1
  2: x1&(x3#x4)+/x3&/x4&
  /(x1&x2)+/x1&/x2&x3&x4
  3: ⟨x1, x2, x3, x4⟩
```

```
4 derv 1 3 4
  derv 2 3 5
  The function f2(x) is self-dual.
```

Exercise 4.21.

```
1 space 32 1
  sbe 1 1
  (x1#x2#
  /x3&/x4#
  x2&x3&/x4)+
  x1&x2.
```

```
sbe 1 2
x1#x2.
sbe 1 3
x1#x3.
sbe 1 4
x1#x4.
```

```
sbe 1 5
x2#x3.
sbe 1 6
x2#x4.
sbe 1 7
x3#x4.
```

```
derv 1 2 10
isc 10 2 11
derv 1 3 12
isc 12 3 13
derv 1 4 14
isc 14 4 15
```

```
derv 1 5 16
isc 16 5 17
derv 1 6 18
isc 18 6 19
derv 1 7 20
isc 20 7 21
```

2 The empty TVL 19 confirms that function (4.21) is symmetric with regard to (x_2, x_4).

3 $f(\mathbf{x}) = x_2\, x_4 \vee (x_2 \oplus x_4)x_1\, \overline{x}_3 \vee \overline{(x_2 \vee x_4)}\overline{x}_1\overline{x}_3 \vee x_1x_3$.

Exercise 4.22.

1
```
space 32 1
avar 1
x1 x2 x3 x4.
sbe 1 1
/x2&x3&/x4+
x4&(x1+/x2)+
/x1&/x3&
(x2#/x4).
sbe 1 2
x1=0.
sbe 1 3
x2=0.
sbe 1 4
x3=0.
sbe 1 5
x4=0.
derk 1 2 10
isc 1 10 11
isc 11 2 12
cpl 2 13
isc 13 11 14
derk 1 3 20
isc 1 20 21
isc 21 3 22
cpl 3 23
isc 23 21 24
derk 1 4 30
isc 1 30 31
isc 31 4 32
cpl 4 33
isc 33 31 34
derk 1 5 40
isc 1 40 41
isc 41 5 42
cpl 5 43
isc 43 41 44
```

2 The empty TVLs 24 and 42 confirm that function (4.26) is monotonously decreasing with regard to x_2 and monotonously increasing with regard to x_4.

3 50: /x2&(/x1+x3)+x4&(x1+/x3)

4 syd 1 50 51 The empty TVL 51 confirms the correct simplification.

Exercise 4.23.

1
```
space 32 1
avar 1
x1 x2 x3 x4.
sbe 1 1
/x1&(/x3#/x2&x4)+
x1&(x2#/x3)&x4+
x1&x3&/x4.
sbe 1 2
x1.
sbe 1 3
x2.
sbe 1 4
x3.
sbe 1 5
x4.
derk 1 2 11
derk 1 3 12
derk 1 4 13
derk 1 5 14
```

2 The derivatives 11 and 13 are equal to 1 such that the function is linear with regard to x_1 and x_3.

3
```
syd 1 2 20
maxk 20 2 21
syd 21 4 22
maxk 22 4 23
```
The remaining function is $x_2 \vee \overline{x}_4$.

4
```
50: x1#x3#(x2+/x4)
syd 1 50 51
```
The empty TVL 51 confirms the correct simplification.

Exercise 4.24.

1
```
space 32 1
avar 1
x1 x2 x3 y.
sbe 1 1
/x1&x3&/y+
/x2&(x3#y)+
x1&/x3&y.
sbe 1 2
/x1&x3&/y+
/x2&(x3#y)+
x2&y&(x1+/x3).
sbe 1 3
x2&/x3&/y+
/x2&(x3#y)+
/x1&x2&/x3.
sbe 1 4
x1&x2&/y+
/x2&(x3#y).
sbe 1 5
y.
```

2
```
derk 1 5 11
derk 2 5 12
derk 3 5 13
derk 4 5 14
```
The function 12 is constant equal to 1 such that (4.44) is uniquely solvable with regard to y.

3
```
mink 1 5 21
mink 2 5 22
maxk 3 5 23
maxk 4 5 24
```
The functions 21 and 22 are constant equal to 0 such that (4.43) and (4.44) are unique with regard to y.

4
```
maxk 1 5 31
maxk 2 5 32
mink 3 5 33
mink 4 5 34
```
The function 32 is constant equal to 1, and the function 34 is constant equal to 0 such that (4.44) and (4.46) are solvable with regard to y.

```
5 csd 2 5 40
  maxk 40 5 41
```
41: $g(\mathbf{x}) = x_1x_2 \vee \overline{x}_3$.

6
```
csd 5 41 42
isc 2 42 43
maxk 43 5 44
cpl 44 45
```
The function 44 is constant equal to 1, and the function 45 is constant equal to 0, and this confirms the solution function 41.

7
```
cpl 5 50
isc 50 4 51
maxk 51 5 52
isc 5 4 53
maxk 53 5 54
```
The solution set of (4.46) is defined by:
52: $q(\mathbf{x}) = x_1x_2 \vee \overline{x}_2x_3$
54: $r(\mathbf{x}) = \overline{x}_2\overline{x}_3$

8
```
uni 52 54 55
cpl 55 56
sbe 1 57
/x1&x2&/x3.
sbe 1 58
/x1&x2&x3.
```
```
sbe 1 59
/x1&x2.
_copy 52 61
uni 52 57 62
uni 52 58 63
uni 52 59 64
```
The solution functions of (4.46) are:
61: $g1(\mathbf{x}) = x_1x_2 \vee \overline{x}_2x_3$
62: $g2(\mathbf{x}) = x_1x_2 \vee \overline{x}_2x_3 \vee \overline{x}1x2\overline{x}3$
63: $g3(\mathbf{x}) = x_1x_2 \vee \overline{x}_2x_3 \vee \overline{x}1x2x3$
64: $g4(\mathbf{x}) = x_1x_2 \vee \overline{x}_2x_3 \vee \overline{x}1x2$

9
```
csd 5 61 70
isc 4 70 71
maxk 71 5 72
csd 5 62 73
isc 4 73 74
maxk 74 5 75
```
```
csd 5 63 76
isc 4 76 77
maxk 77 5 78
csd 5 64 79
isc 4 79 80
maxk 80 5 81
```
The empty TVLs 72, 75, 78, and 81 confirm that the four functions $g1(\mathbf{x}), \ldots, g4(\mathbf{x})$ solve (4.46) with regard to y.

Chapter 5

THE SOLUTION OF LOGIC EQUATIONS

1. Tasks

Exercise 5.1 (Equation 1). Let be given $f(\mathbf{x}) = (((x_1 \rightarrow x_2) \downarrow x_3)|x_4) \oplus x_5$.

1 Solve the equations $f(\mathbf{x}) = 0$ and $f(\mathbf{x}) = 1$ by using the XBOOLE Monitor in one step.

Hint: transfer | (NAND) and $\downarrow$ (NOR) into formulas that can be understood by the XBOOLE Monitor.

2 Solve these equations by operations with sets.

3 Verify the results by comparison of the previous two subtasks.

Exercise 5.2 (Equation 2). Let be given partial solution sets of $f(\mathbf{x}) = 1$, $g(\mathbf{x}) = 1$, $f(\mathbf{x}) = 0$, and $g(\mathbf{x}) = 0$, respectively. Consider the two equations $f(\mathbf{x}) \wedge g(\mathbf{x}) = 1$ and $f(\mathbf{x}) \vee g(\mathbf{x}) = 0$.

1 Why in both cases the intersection of partial solution sets with the same right side has to be used?

2 What is the required set operation for the equations $f(\mathrm{x}) \wedge g(\mathbf{x}) = 0$ and $f(\mathbf{x}) \vee g(\mathbf{x}) = 1$?

Exercise 5.3 (Equation 3). Let be given the two equations

$$x_1 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_4x_5 = 0$$

and

$$x_1 \sim x_1x_2 \sim x_1x_2x_3 \sim x_1x_2x_3x_4 \sim x_1x_2x_3x_4x_5 = 0.$$

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

1 Compare the solution sets of these two equations by means of transforming the given expressions.

2 Verify your result using the XBOOLE Monitor.

3 Explain the use of the symmetric difference and the complement of the symmetric difference.

Exercise 5.4 (Linear Disjunctive Equations). Let be given the following system of equations:

$$x_1 \vee x_3 \vee x_6 = 0, \quad x_2 \vee x_4 \vee x_6 = 0, \quad x_1 \vee x_2 \vee x_4 \vee x_5 = 1.$$

1 Combine these three equations into one equation and use the XBOOLE Monitor for a solution in one step.

2 Solve this system of equations by using and combining the solution sets of the three single equations.

3 Combine considerations of the right sides and the values of variables together with required (partial) solution sets.

Exercise 5.5 (Linear Equations). Let be given the following system of equations:

$$x_1 \oplus x_3 \oplus x_6 = 0, \quad x_2 \oplus x_4 \oplus x_6 = 0, \quad x_1 \oplus x_2 \oplus x_4 \oplus x_5 = 1.$$

1 Combine these three equations into one equation and use the XBOOLE Monitor for a solution in one step, after the possible simplification of the equation.

2 Solve this system of equations by using and combining the solution sets of the three single equations.

We could continue with many different questions to the remaining areas of Chap. 5 in [18]. We felt, however, that this would be more an exercise in typing and using the different possibilities for solving logic equations. The main purpose to have some problems here was the synchronization with the chapter structure of [18]. But now it will be more useful to study all the methods that have been made available in [18] together with the application process. Therefore it might be useful to read again Chap. 5 in the previous book and check whether the required knowledge is available or must be improved. After doing so (if necessary) the solution of applied problems can start.

2. Solutions

Exercise 5.1.

1 When the XBOOLE Monitor is used, then the given equations has to be typed as follows: $/(/((x1 > x2) + x3)\&x4)\#x5 = 0$. and $/(/((x1 > x2) + x3)\&x4)\#x5 = 1$. The item "Extras – Solve Boolean Equation" results in the following orthogonal sets of solutions:

$$f(\mathbf{x}) = 0\text{:} \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ - & - & - & 0 & 1 \\ - & - & 1 & 1 & 1 \\ - & 1 & 0 & 1 & 1 \end{pmatrix}, \quad f(\mathbf{x}) = 1\text{:} \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ - & 1 & 0 & 1 & 0 \\ - & - & 1 & 1 & 0 \\ - & - & - & 0 & 0 \end{pmatrix}.$$

Some other representations (for example, the replacement of $x_1 \to x_2$ by $\overline{x}_1 \vee x_2$) of the equation could lead to another representation of the solution set, however, the solution set will always be the same; this can be tested by using the *symmetric difference*, for instance.

2 One way to solve this problem is the creation of five different objects (TVLs) containing the ternary vectors $x_1 = (1----)$, $x_2 = (-1---)$, $x_3 = (--1--)$, $x_4 = (---1-)$ and $x_5 = (----1)$. Then we use the sequence of set operations as follows: $x_1 \to$ complement $\to$ union with $x_2 \to$ union with $x_3 \to$ complement $\to$ intersection with $x_4 \to$ complement $\to$ symmetric difference with x_5. This results in the solution of $f(\mathbf{x}) = 1$. The solution of $f(\mathbf{x}) = 0$ requires a final complement.

3 Empty results of *symmetric differences* between the associated solutions of the previous two subtasks confirm the equivalence of the found results.

Exercise 5.2.

1 Here it is possible to go back to the definition of $\wedge$ and $\vee$: the conjunction is equal to 1 if and only if both operands are equal to 1, the disjunction is equal to 0 if and only if both operands are equal to 0. Therefore the intersection of the appropriate partial solution sets has to be used.

2 The conjunction is equal to 0 if at least one of the operands is equal to 0. The disjunction is equal to 1 if at least one of the operands is equal to 1, therefore in both cases the union of the appropriate partial solution sets can be used to find the solutions of the given equations.

Exercise 5.3.

1 We use the relation between $\sim$ and $\oplus$: $x \sim y = \overline{x \oplus y} = x \oplus y \oplus 1$. Taking into consideration that $1 \oplus 1 = 0$ we get $x_1 \oplus x_1x_2 \oplus x_1x_2x_3 \oplus x_1x_2x_3x_4 \oplus x_1x_2x_3x_4x_5 = x_1 \sim x_1x_2 \sim x_1x_2x_3 \sim x_1x_2x_3x_4 \sim x_1x_2x_3x_4x_5$. Therefore the two equations have the same solution set.

2 Solve both equations. Both solutions sets have the same 21 values 1 in their Karnaugh maps. An empty TVL as result of *symmetric difference* between the solution sets confirm that the solution sets of both equations are equal.

3 $x \oplus y = 1$ is equivalent to $x\overline{y} \vee \overline{x}y = 1$, and therefore the solution set of the equation $f(\mathbf{x}) \oplus g(\mathbf{x}) = 1$ can be based on the symmetric difference. $f(\mathbf{x})\overline{g(\mathbf{x})} \vee \overline{f(\mathbf{x})}g(\mathbf{x}) = 1$ has the solution set $F_1 \cap \overline{G}_1 \cup \overline{F}_1 \cap G_1 = F_1 \triangle G_1$ (see also Chap. 3). The equation $f(\mathbf{x}) \sim g(\mathbf{x}) = 1$ has the complement of the solution set of the equation $f(\mathbf{x}) \oplus g(\mathbf{x}) = 1$, hence, the solution set of the equation $f(\mathbf{x}) \sim g(\mathbf{x}) = 1$ can be based on the complement of the symmetric difference.

Exercise 5.4.

1 We use $/(x1 + x3 + x6)\&/(x2 + x4 + x6)\&(x1 + x2 + x4 + x5) = 1$. Typing this equation we get the single solution vector (000010).

2 The first two equations have a single solution vector $(0 - 0 - -0)$ and $(-0 - 0 - 0)$. The intersection of these two vectors leaves only one component undefined: $(0000 - 0)$. x_1, x_2 and x_4 are already equal to 0, therefore x_5 must take the value 1.

3 Such a disjunction is equal to 0 if and only if all the operands are equal to 0 which defines the values in a unique way. When the right side is equal to 1, at least one operand has to be equal to 1. Therefore the last equation has in principle, four ternary solution vectors $(1 - - - - -)$, $(01 - - - -)$, $(00 - 1 - -)$ and $(00 - 01 -)$, however, three of them will not contribute to the final solution.

Exercise 5.5.

1 We apply the first part of a PRP

```
space 32 1
avar 1
x1 x2 x3 x4 x5 x6.
sbe 1 1
/(x1#x3#x6)&/(x2#x4#x6)&
(x1#x2#x4#x5)=1.
```

and get as result of the XBOOLE Monitor the solution shown on the right.

$$S = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

2
```
sbe 1 2
(x1#x3#x6)=0.
sbe 1 3
(x2#x4#x6)=0.
sbe 1 4
(x1#x2#x4#x5)=1.
isc 2 3 5
isc 5 4 5
syd 5 1 6
```

The intersection of the partial solution sets 2, 3, and 4 finds the same solution in another order. The empty result of the last SYD operation confirms that both solution sets contain the same solution vectors. That can be verified by comparing the Karnaugh maps, too.

II

APPLICATIONS

Chapter 6

LOGICS AND ARITHMETICS

The concepts of exact "logical reasoning" and "thinking" are fully based on propositional logics. This development already started in ancient times and continues until now. And the "computerization of the human thinking" is still a vibrant field of discussions. Hence, some examples will be given with regard to these problems, particularly to the application of the implication and the equivalence. Keep in mind that the implication $x \to y$ can be expressed by **if** x **then** y. All the programming languages that are used now have constructions like this or the extension **if** x **then** y **else** z. Therefore it is very necessary to understand very well the truth of the assumption (or condition) x and its negation (to be used for the else-branch). The equivalence, expressed by $x \leftrightarrow y$, $x \equiv y$, $x \odot y$ or $x \sim y$, will be translated by x **if and only if** y. Make sure that you understand to translate the formulas into "natural language" and real-world problems into formulas.

1. Propositional Logics

Exercise 6.1 (Inference Rules). Show that the following rules are always valid! Translate these rules into "natural language"!

1 $(x \to y) \sim (\overline{x} \vee y)$;

2 $(x \to y) \sim \overline{(x \wedge \overline{y})}$;

3 $(x \to y) \sim (\overline{x} \vee (x \wedge y))$.

Exercise 6.2 (Propositional Rule). The argument $(x_1 \to x_2) \wedge (x_3 \to x_4) \wedge (x_1 \vee x_3) \to (x_2 \vee x_4)$ is given.

1 Show that this argument form is valid.

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

2 Translate this expression into a set of assumptions and a set of rules. Express the meaning of this form in natural language.

Exercise 6.3 (Propositional Model). Remember the definition of the exclusive *or* operation $\oplus$.

1 Express "p *or* q *but not both*" in terms of p, q, $\overline{p}$, $\overline{q}$, $\vee$ and $\wedge$.

2 Verify that the formula found in the previous item expresses $p \oplus q$.

Exercise 6.4 (Paradoxes). Show the paradoxes of the *implication* by confirming that the following rules are tautologies.

1 $x \to (y \to x)$. *A true proposition* x *is implied by any proposition* y.

2 $\overline{x} \to (x \to y)$. *A false proposition* x *implies any proposition* y.

3 $(x \to y) \vee (y \to x)$. *Of any two propositions, one implies the other.*

Exercise 6.5 (Rules). Explain the following rules in natural language:

1 $\overline{x} \vee x$ is a tautology – *Law of the excluded middle*;

2 $x \vee (y \vee z) \to (x \vee y) \vee z$ – *Associative rule*;

3 $x \to (x \vee y)$ – *Expansion rule*;

4 $(x \vee x) \to x$ – *Contraction rule*;

5 $(x \vee y) \wedge (\overline{x} \vee z) \to (y \vee z)$.

Exercise 6.6 (Implications). Verify the following rules and explain these rules in natural language:

1 $x \wedge (x \to y) \to y$ *modus ponens*;

2 $\overline{y} \wedge (x \to y) \to \overline{x}$ *modus tollens*;

3 $(x \to y) \to (\overline{y} \to \overline{x})$ *contrapositive*;

4 $(x \to y) \wedge (y \to z) \to (x \to z)$ *hypothetical syllogism*;

5 $(x \vee y) \wedge \overline{x} \to y$ *disjunctive syllogism*;

6 $(y \to z) \to ((x \wedge y) \to z)$ *strengthening the antecedent*;

7 $(x \to y) \to (x \to (y \vee z))$ *weakening the consequent*;

8 $(x \to y) \wedge (\overline{x} \to y) \to y$;

9 $(x \to y) \wedge (x \to \overline{y}) \to \overline{x}$.

An important concept is the concept of a *system of axioms*. These axioms are considered to be true and can be applied without further consideration. Based on these axioms, so-called derivation rules can be applied. The *modus ponens* is the most famous example. It states that if x is true and x implies y i.e. the implication $x \to y$ is true, then y will be true as well. Therefore, a mathematical theory can start with a set of axioms, and by using these axioms, the *modus ponens* and possibly the definition of new concepts, a theory can be built. Typical requirements for sets of axioms are **completeness**, it also must be **free of contradictions** and **easy to use**.

A famous example is the system that has been proposed by the German mathematician *David Hilbert*.

Exercise 6.7 (Hilbert's Axiomatic System). Show that the following rules are always valid. Translate these rules into "natural language".

1 $(x \to x)$;

2 $x \to (y \to x)$;

3 $(x \to y) \to ((y \to z) \to (x \to z))$;

4 $(x \to (y \to z)) \to ((x \to y) \to (x \to z))$;

5 $x \to (x \vee y)$, $y \to (x \vee y)$;

6 $(x \to z) \to ((y \to z) \to ((x \vee y) \to z))$;

7 $(x \wedge y) \to x$, $(x \wedge y) \to y$;

8 $(z \to x) \to ((z \to y) \to (z \to (x \wedge y)))$;

9 $((x \wedge y) \vee z) \to ((x \vee z) \wedge (y \vee z))$,
$((x \vee z) \wedge (y \vee z)) \to ((x \wedge y) \vee z)$;

10 $((x \vee y) \wedge z) \to ((x \wedge z) \vee (y \wedge z))$,
$((x \wedge z) \vee (y \wedge z)) \to ((x \vee y) \wedge z)$;

11 $(x \to y) \to (\overline{y} \to \overline{x})$;

12 $x \wedge \overline{x} \to y$;

13 $y \to (x \vee \overline{x})$.

According to our understanding of logic formulas and logic equations, it would be possible to replace the two axioms $((x \wedge y) \vee z) \to ((x \vee z) \wedge (y \vee z))$, $((x \vee z) \wedge (y \vee z)) \to ((x \wedge y) \vee z)$ by one formula $((x \wedge y) \vee z) \sim ((x \vee z) \wedge (y \vee z))$ or by $((x \wedge y) \vee z) = ((x \vee z) \wedge (y \vee z))$. It was, however, one intention of Hilbert's System to use the implication as much as possible.

As we can see from these examples, the calculations can be completely transferred to the XBOOLE Monitor (or a similar software package). It is no longer so important to do these calculations by hand. However, the **logical modeling** of a real-world situation will still be of high importance, even higher than before, since more complex situations can be modeled. As a very complex example, let us have a look at a combinatorial problem that was interesting for the famous German mathematician C. F. Gauß.

Exercise 6.8 (The Queens' Problem). Find a binary model for the problem and answer the following questions.

1 Prepare a PRP that describes all possibilities to place eight queens on a chessboard 8×8 in such a way that no queen attacks another one.

2 How many solutions exist for this problem?

3 How many solutions exist for a board 7×7 or a board 9×9?

4 For which value of the board size no solution can be found?

The modeling of this problem by means of logical equations is rather challenging and requires some new ideas and skills. We use binary variables x_{ik} with $i = 1, \ldots, 8$ and $k = 1, \ldots, 8$ for a detailed description of the position of the queens on the board and its effect. We assume that everybody understands that in each horizontal row, in each vertical column and in each diagonal only one queen can be placed. The values of the variables x_{ik} will be defined as follows:

$$x_{ik} = \begin{cases} 0 & \text{if no queen is on the field } (i,k) \\ 1 & \text{if there is a queen on the field } (i,k). \end{cases}$$

Therefore, the position of a queen on the field $(1,1)$, for instance, and the consequences of this position can be expressed by the following conjunction being equal to 1:

$$x_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{21}\overline{x}_{31}\overline{x}_{41}\overline{x}_{51}\overline{x}_{61}\overline{x}_{71}\overline{x}_{81}\overline{x}_{22}\overline{x}_{33}\overline{x}_{44}\overline{x}_{55}\overline{x}_{66}\overline{x}_{77}\overline{x}_{88}.$$

This term ensures that no other queen is in the first column, in the first row and in the respective diagonal if there is a queen on the field $(1,1)$. Now we express the requirement that one queen must be in the first column by a sequence of similar terms for the fields

$$(1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8).$$

These terms have to be combined by $\vee$ since the queen can be on this field **or** on this field **or** .... In this way we get for the first column

$$x_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{21}\overline{x}_{31}\overline{x}_{41}\overline{x}_{51}\overline{x}_{61}\overline{x}_{71}\overline{x}_{81}\overline{x}_{22}\overline{x}_{33}\overline{x}_{44}\overline{x}_{55}\overline{x}_{66}\overline{x}_{77}\overline{x}_{88}$$
$$\vee\overline{x}_{11}x_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{22}\overline{x}_{32}\overline{x}_{42}\overline{x}_{52}\overline{x}_{62}\overline{x}_{72}\overline{x}_{82}\overline{x}_{21}\overline{x}_{23}\overline{x}_{34}\overline{x}_{45}\overline{x}_{56}\overline{x}_{67}\overline{x}_{78}$$
$$\vee\overline{x}_{11}\overline{x}_{12}x_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{23}\overline{x}_{33}\overline{x}_{43}\overline{x}_{53}\overline{x}_{63}\overline{x}_{73}\overline{x}_{83}\overline{x}_{22}\overline{x}_{31}\overline{x}_{24}\overline{x}_{35}\overline{x}_{46}\overline{x}_{57}\overline{x}_{68}$$
$$\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}x_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{24}\overline{x}_{34}\overline{x}_{44}\overline{x}_{54}\overline{x}_{64}\overline{x}_{74}\overline{x}_{84}\overline{x}_{23}\overline{x}_{32}\overline{x}_{41}\overline{x}_{25}\overline{x}_{36}\overline{x}_{47}\overline{x}_{58}$$
$$\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}x_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{25}\overline{x}_{35}\overline{x}_{45}\overline{x}_{55}\overline{x}_{65}\overline{x}_{75}\overline{x}_{85}\overline{x}_{24}\overline{x}_{33}\overline{x}_{42}\overline{x}_{51}\overline{x}_{26}\overline{x}_{37}\overline{x}_{48}$$
$$\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}x_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{26}\overline{x}_{36}\overline{x}_{46}\overline{x}_{56}\overline{x}_{66}\overline{x}_{76}\overline{x}_{86}\overline{x}_{25}\overline{x}_{34}\overline{x}_{43}\overline{x}_{52}\overline{x}_{61}\overline{x}_{27}\overline{x}_{38}$$
$$\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}x_{17}\overline{x}_{18}\overline{x}_{27}\overline{x}_{37}\overline{x}_{47}\overline{x}_{57}\overline{x}_{67}\overline{x}_{77}\overline{x}_{87}\overline{x}_{26}\overline{x}_{35}\overline{x}_{44}\overline{x}_{53}\overline{x}_{62}\overline{x}_{71}\overline{x}_{28}$$
$$\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}x_{18}\overline{x}_{28}\overline{x}_{38}\overline{x}_{48}\overline{x}_{58}\overline{x}_{68}\overline{x}_{78}\overline{x}_{88}\overline{x}_{27}\overline{x}_{36}\overline{x}_{45}\overline{x}_{54}\overline{x}_{63}\overline{x}_{72}\overline{x}_{81}$$
$$= 1.$$

Such an equation does not look very nice, however, it is easy to construct and easy to understand. By using a TVL instead of the expression, it is easy to write down a PRP for the XBOOLE Monitor that solves this task. For several sizes of the chess board it may be faster to generate such a special PRP by means of a piece of software. Of cause, it is more elegant to use the XBOOLE library directly in such a program. The solution could, as we have done before, be represented by eight ternary vectors. The terms for the second, third, ..., eighth column follow the same example, and the intersection of the respective ternary matrices will give the final solution. This can be done for any value of n.

The special branch of *fairy chess* uses additional pieces that are not existing for the "normal chess". One of them combines the movement of a rook and a knight – in the same way as the queen combines the movement of rook and bishop; it is called *Chancellor*. The *Cardinal* combines the movement of a bishop and a knight.

Exercise 6.9 (Fairy Chess). Find all combinations of a maximum number of Cardinals on a very small board 3×3 and for larger boards as well.

Many more pieces from the area of fairy chess could be explored, however, the same approach will be used for all of them.

Another problem of this kind is the so-called $(n + k)$-problem. In order to start we use $k = 1$. One pawn will be placed onto the normal chessboard, say at the position $(4, 4)$. In this case, we have $n = 8$ and $k = 1$. Generally the n defines the size of the board, the k the number of pawns on the board. This pawn interrupts the attacking line of queens, in the fourth horizontal line, in the fourth vertical row, in the diagonal from the field $(1, 1)$ to $(8, 8)$ and from $(7, 1)$ to $(1, 7)$, and this allows the placement of two queens (sometimes) in these rows and columns. Again generally, it is expected that k additional pawns on the board allow the placement of k additional queens.

Exercise 6.10 (The $n+k$-problem). $n+1$ queens and one pawn should be placed on an $n \times n$ chess board such that no queen attacks any other queen.

1 How many such positions for 9 queens with the pawn on the field $(4,4)$ of an (8×8)-board are possible?

2 How many different positions of the queens are possible altogether with one pawn on any field of an (8×8)-board?

3 How many such positions for 10 queens with two pawns on the fields $(4,4)$ and $(5,5)$ of an (8×8)-board are possible?

4 How many different positions of the queens are possible altogether with two pawns on any field of an (8×8)-board?

As additional exercise the previous task can be solved also for other sizes of boards, e.g. other values of n ($n = 6$ or 7 or 9).

There is not yet a comprehensive summary and exploration of these results for reasonable values of n and k, this is still a problem for exploration and publication. This applies even more to the following area of problems – the results achieved so far and the methodology have not yet been published somewhere else, they have been achieved only recently. The authors, however, wanted to include these results into the book because it shows extremely well how strong and efficient the mechanism of Boolean equations can be used to solve discrete problems which are normally dealt with in combinatorics.

Let us consider the difficult area of coloring problems. Formally the problem can be described in the following way: Can the nodes of a planar graph be colored by means of at most four colors in such a way that two nodes connected by an edge have different colors? The general answer is "yes", many controversial discussions took place because the proof of this theorem used computer assistance. However, we want to deal with this problem in a more constructive way, because even when it is known that four colors are sufficient to color a given graph, then this will not yet give a solution for one special given graph.

Exercise 6.11 (The coloring problem). A graph G is given by means of the adjacency matrix M:

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

This adjacency matrix describes a graph with 10 nodes. The element $M_{1,2} = 1$ describes an edge from node 1 to node 2 etc. By the way, this graph is supposed to be a difficult example for coloring questions, it is called Birkhoff's Diamond.

1 Draw a sketch of this graph.

2 Try to find a coloring of this graph by hand.

3 Design a Boolean model of this problem.

4 Find a solution.

5 Is the solution a unique solution?

The next question will also indicate a very difficult and interesting area of Graph Theory and Discrete Mathematics. For this class of problems a graph is given, and for this graph (at least) one sequence of edges should be found that crosses each node precisely once and ends in the node where it started. We are setting here a problem where the edges are undirected, the problem can also be considered (and sometimes solved) when the edges are directed. Furthermore we extend the problem such that more than one disjoint sequence of edges can exist which commonly crosses each node precisely once.

Exercise 6.12 (Hamiltonian Graphs). Let be given the following graph by its adjacency matrix:

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

1 Draw a sketch of this graph.

2 Design a Boolean model of the problem of extended Hamiltonian graphs and fit it into a PRP.

3 Execute the PRP in the XBOOLE Monitor in order to find all Hamiltonian paths for the given graph. How many Hamiltonian paths exist?

Another example for Hamiltonian paths in a graph are movements of a knight on the chessboard.

Exercise 6.13 (Knight on the Chess Board). Place a knight on any selected field of the chessboard 6×6 and find a path of the knight that returns to the selected field and uses other fields only once. If not all fields are used then repeat the same procedure starting form a new unused field. Finally all fields must be used.

1 Is there a set of paths of the knight where each path uses all the other fields of the board exactly once and returns to the selected start field?

2 How many Boolean variables are necessary to model this problem?

3 Find and explain a Boolean model for this problem.

4 Generate a PRP in order to solve the problem. Due to the marginal overlapping of the generated TVLs, the order of the required intersections is essential. Find an appropriate order using the XBOOLE Monitor. How many solutions exist?

A next very interesting area of problems is the existence of *Eulerian paths*. Such a path has the property that each edge must be used once and only once, from a starting point through all edges to any end point. A subproblem is the existence of *Eulerian circuits*, where the start point and the end point are identical. We will start with a very simple problem in order to study the methodology.

Exercise 6.14 (Simple Eulerian Path). Let be given a graph with four nodes `a`, `b`, `c` and `d` and edges between `a` and `b`, `b` and `c`, `c` and `d`, `d` and `a` as well as `b` and `d`.

1 Draw a sketch of this graph.

2 Find a Boolean model for this problem.

3 Find all Eulerian paths.

4 How the graph must change by adding or removing one edge such that the existing Eulerian paths are Eulerian circuits, too.

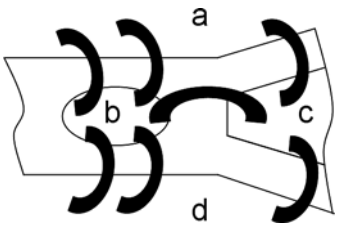


Figure 6.1 The seven brigdes of Königsberg

The original problem of *Eulerian Paths* originated in the town of *Königsberg*. In the next task we study the original problem and a slightly modified one.

Exercise 6.15 (Bridges in Königsberg). As can be seen in Fig. 6.1 there were seven bridges, two from point a (the north side of a river) to point b (an island in the middle of the river), two from point d (on the south side of the river) to point b (on the island). After the island the river branches to the north and the south, and there are three more bridges, one from point c (on the east of the branch) to point a, one to point b and one to point d. And the problem is quite simple.

1 Draw a sketch of this graph.

2 Find a Boolean model for this problem.

3 Find all Eulerian paths.

4 Modify the problem such that the bridge from point b to point c is removed. Find a Boolean model for the modified problem.

5 Find all Eulerian paths for the modified problem.

A last demanding example for the power of modeling will be taken from the area of game playing, however, with a considerable mathematical background. It relates to the game of *Sudoku*, a game which is originating from Japan. A board of 9×9 fields will be split in 9 subboards with a size of 3×3. In each column, each row and each subboard of the board the numbers from 1 to 9 must appear once and only once. Some fields have a value which has been set before in order to narrow down the search space.

Exercise 6.16 (Sudoku). On a board of 9×9 fields the following values are given: $(1,5)$:5, $(1,9)$:2, $(2,4)$:6, $(2,9)$:9, $(3,2)$:1, $(3,3)$:8, $(3,4)$:4, $(4,2)$:2, $(4,3)$:3, $(4,5)$:7, $(5,3)$:5, $(5,5)$:6, $(5,7)$:2, $(6,7)$:6, $(6,8)$:4, $(7,6)$:2, $(7,7)$:8, $(7,8)$:5, $(8,1)$:9, $(8,6)$:1, $(9,1)$:7, and $(9,5)$:3.

1 Draw a sketch of this board with the given values.

2 Find a Boolean model for these problems. How many Boolean variables are needed?

3 Solve this example using XBOOLE.

2. Solutions

Exercise 6.1.
Solve 3 equations having each one of the given expression on one side and the value 0 on the other side. Empty solution sets confirm that all these expressions represent tautologies, i.e. they represent logical identities and can be used everywhere in logics.

1 The crucial point is a true assumption x. In order to get a true implication $x \to y$, the conclusion must be true as well, or in other words, if the assumption x is true and the implication is true, then y must be true as well, a true assumption and a true implication allow to derive the truth of the conclusion.

2 A true implication does not allow a true assumption and a false conclusion.

3 In order to get a true implication, x and y must be true at the same time, or x must be false. It is important to understand this theorem: it says that "everything follows from a false assumption".

Exercise 6.2.

1 The empty solution set of `((x1>x2)&(x3>x4)&(x1+x3)>(x2+x4))=0` confirms that there is no contradiction. Hence, the given expression is a tautology.

2 The conclusion $(x_2 \vee x_4)$ has an assumption consisting of three parts: $x_1 \to x_2$, $x_3 \to x_4$ and $x_1 \vee x_3$. This theorem means that if there are two true implications and at least one true assumption, then there is also at least one true conclusion.

Exercise 6.3.

1 The *exclusive or* expresses the alternative truth of two statements which cannot be true at the same time, therefore it must be expressed that *one proposition is true, but not the other* and vice versa. This can be done by $p \wedge \overline{q} \vee \overline{p} \wedge q$. The application of *or* in the language of the daily life very often neglects this difference.

2 The empty solution set of `((p&/q+/p&q)=(p#q))=0` confirms the equivalence of both expressions.

Exercise 6.4.
It is necessary (and not so easy) to get accustomed to these statements, they "look a bit strange".

1 The empty solution set of `(x>(y>x))=0` confirms tautology.

2 The empty solution set of `(/x>(x>y))=0` confirms tautology.

3 The empty solution set of `((x>y)+(y>x))=0` confirms tautology.

Exercise 6.5.
Empty solution sets of restrictive equations having one of the given rules on one side confirm that all these rules represent tautologies.

1 Either a proposition x or its negation is true. They cannot be true at the same time, and they cannot be false at the same time.

2 If there are three propositions in a disjunctive assumption of an implication, then their order is not important. They can be used in any order.

3 A true proposition x can be extended by another proposition y using the disjunction.

4 The same statement must not be repeated, it can be replaced by the statement alone.

5 When the assumption is transformed, then we get the following formula:

$$(x \vee y)(\overline{x} \vee z) = (xz \vee \overline{x}y \vee yz).$$

Now we can consider the three parts of the assumption:

- If xz is true, then particularly z is true, and therefore $(y \vee z)$ is true.
- If $\overline{x}y$ is true, then particularly y is true, and therefore $(y \vee z)$ is true.
- If yz is true, then both y and z are true, and therefore $(y \vee z)$ is true.

Since we see x and $\overline{x}$ in the assumption which cannot be true at the same time, one of the two propositions y or z must be true, and therefore $(y \vee z)$ is always true.

Exercise 6.6.
Empty solution sets of restrictive equations having one of the given rules on one side confirm again that all these rules represent tautologies.

1 The use of the modus ponens guarantees that the right side of an implication (the consequent) is true when the left side is true. From a true assumption and a true rule we get a true conclusion. This is one of the most basic approaches in Mathematics. Let us see, for instance, the following more or less trivial example:

Theorem. If a and b are natural numbers and $a < b$, then $a^2 < b^2$.

Proof. If $a < b$ then $b = a + c,\ c > 0$ and $b^2 = (a+c)^2 = a^2 + 2ac + c^2 > a^2$ since $2ac \geq 2a$ and $c^2 \geq 1$.

This is the general statement, or the general rule. Now the assumption can be equal to true by, for instance, $a = 10$ and $b = 15$, and without any calculations we know that $a^2 < b^2$. And this applies to any natural numbers, even numbers like 123456789 and 234567890. Here you must only be careful to determine which number is the smaller one and which one is the larger one.

2 For a true implication a false consequent has to be based on a false assumption.

3 Any implication (any rule) can be reversed when the negated propositions are used.

4 A two-step use of implications can be replaced by a one-step direct implication. This can be transferred to a finite number of steps: if there is a chain of implications, then this chain can be abbreviated by one direct step.

5 This item describes a property of the disjunction: If a disjunction of x and y is true, and one proposition x is false, then the other one must be true.

6 If there is any true implication, then the assumption can be strengthened by the conjunction with other propositions.

7 If there is any true implication, then the conclusion can be extended by disjunctions with other propositions.

8 If a proposition is implied by a given assumption and its negation, then the proposition is independent on these assumptions.

9 When an assumption implies a given conclusion and its negation, then this assumption must be false.

Exercise 6.7.
We believe **David Hilbert** that these rules are useful or necessary for something. The discussion whether an axiomatic system is appropriate, understandable, minimal etc. can fill another book. In order to verify that these rules are tautologies, the following PRP can be executed in the XBBOLE Monitor. Alternatively the solution sets for the characteristic equations of the left side and the right side of the most global implication can be calculated. In this approach empty sets of the difference (`DIF`) operation between the solution sets of the left side minus the right side confirm the tautology.

```
space 32 1
sbe 1 1
(x>x)=0.
sbe 1 2
(x>(y>x))=0.
sbe 1 3
((x>y)>((y>z)>(x>z)))=0.
sbe 1 4
((x>(y>z))>((x>y)>(x>z)))=0.
sbe 1 5
((x>(x+y))+(y>(x+y)))=0.
sbe 1 6
((x>z)>((y>z)>((x+y)>z)))=0.
sbe 1 7
(((x&y)>x)+((x&y)>y))=0.
sbe 1 8
((z>x)>((z>y)>(z>(x&y))))=0.
sbe 1 9
((((x&y)+z)>((x+z)&(y+z)))+
(((x+z)&(y+z))>((x&y)+z)))=0.
sbe 1 10
((((x+y)&z)>((x&z)+(y&z)))+
(((x&z)+(y&z))>((x+y)&z)))=0.
sbe 1 11
((x>y)>(/y>/x))=0.
sbe 1 12
(x&/x>y)=0.
sbe 1 13
(y>(x+/x))=0.
```

Empty solution sets of all these equations confirm that all these rule are always valid. The meaning of the rules in "natural language" is as follows.

1 The meaning is that each proposition implies itself. That might be rather trivial, however, in many occasions such a rule can be used as a starting point for a sequence of actions.

2 If a proposition x is true, then it is implied by any proposition y.

3 This axiom is a possibility to shorten a sequence of conclusions: if x implies y and y implies z, then z is implied by x directly.

4 If x implies $y \to z$ and also y, then it implies z directly.

5 A true proposition x implies the *or* of this proposition with any other proposition.

6 If z is implied by x and also by y, then it is implied by $x \vee y$.

7 The conjunction of two propositions implies the two propositions themselves.

8 If z implies x and also y, then it implies $x \wedge y$.

9 It is very interesting to see that the two distributive laws are included into the set of axioms. This axiom shows the distributivity of $\vee$ with regard to $\wedge$.

10 This axiom shows the distributivity of $\wedge$ with regard to $\vee$.

11 This axiom is the base for proofs by contradiction. In order to show that x implies y, we assume $\overline{y}$ and find out that then $\overline{x}$ must hold which is a contradiction to the original assumption that x is true.

12 $x \wedge \overline{x}$ is always false, and a false assumption implies everything; an implication with the left argument equal to 0 is always equal to 1.

13 A true statement ($x \vee \overline{x}$ is always true) is implied by any assumption.

Exercise 6.8.
We will at least mention that the term in the text is already an abbreviation of a sequence of rules. For a queen on the field $(1, 1)$ and the consideration of the first row we get:

$$x_{11} \rightarrow \overline{x}_{12}, x_{11} \rightarrow \overline{x}_{13}, x_{11} \rightarrow \overline{x}_{14}, x_{11} \rightarrow \overline{x}_{15}, x_{11} \rightarrow \overline{x}_{16}, x_{11} \rightarrow \overline{x}_{17}, x_{11} \rightarrow \overline{x}_{18}.$$

The transformation of the implications and their conjunction results in

$$(\overline{x}_{11} \vee \overline{x}_{12})(\overline{x}_{11} \vee \overline{x}_{13})(\overline{x}_{11} \vee \overline{x}_{14})(\overline{x}_{11} \vee \overline{x}_{15})(\overline{x}_{11} \vee \overline{x}_{16})(\overline{x}_{11} \vee \overline{x}_{17})(\overline{x}_{11} \vee \overline{x}_{18}).$$

The calculation of all these intersections together with the conjunction with x_{11} results in the formula given above. This is a very detailed modeling process, however, it is also very understandable and correct.

1 The term given in the text can be simply represented by a TVL of 8 rows in a PRP. Similar TVLs are added to the PRP for the other columns as well. Thereafter, the intersection of these eight TVLs shows all the solutions.

2 For the board 8×8, there are 92 solutions.

3 For a board 7×7 we have 40 solutions, and 352 solutions for a board 9×9 can be found.

4 It is very easy to see that a board of 3×3 has no solutions, solutions for $n = 7$ exist, therefore, the values $n = 4$, $n = 5$ and $n = 6$ have to be checked with the same method. If this is done carefully, by using the XBOOLE Monitor or even XBOOLE itself, then you will get the following results:

n	number of solutions	n	number of solutions
3	0	9	352
4	2	10	724
5	10	11	2680
6	4	12	14200
7	40	13	73712
8	92	14	365596

Exercise 6.9.
This problem will be solved basically in the same way. Only the equations have to change. Because the board is so small, we can enumerate the fields from 1 to 9, as follows:

7	8	9
4	5	6
1	2	3

The respective Boolean variables can be defined as usual:

$$x_i = \begin{cases} 0 & \text{if no cardinal is on the field } i \\ 1 & \text{if there is a cardinal on the field } i \end{cases}$$

The next step is the definition of the ternary vectors for the nine fields:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1:	1	–	–	–	0	0	–	0	0
2:	–	1	–	0	–	0	0	–	0
3:	–	–	1	0	0	–	0	0	-
4:	–	0	0	1	–	–	–	0	0
5:	0	–	0	–	1	–	0	–	0
6:	0	0	–	–	–	1	0	0	-
7:	–	0	0	–	0	0	1	–	-
8:	0	–	0	0	–	0	–	1	-
9:	0	0	–	0	0	–	–	–	1

These ternary vectors describe the position of such a Cardinal on one field and its consequences. In the PRP for the XBOOLE Monitor each of these vectors should be defined as a TVL assigned to the object number as the label given on the left.

How to proceed now? The requirements for two Cardinals on the board are satisfied, when two positions can be combined without contradiction, and this can be achieved when the intersection of two vectors is used. Each intersection which is not empty characterizes an allowed position of two Cardinals on the board. Since the intersection is a commutative operation, only one intersection must be calculated for each pair of ternary vectors: $1 \cap 2 = 2 \cap 1$.

The following ternary vectors describe the found allowed positions of two Cardinals together with their common restricted fields on the board.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1–2:	1	1	–	0	0	0	0	0	0
1–3:	1	–	1	0	0	0	0	0	0
1–4:	1	0	0	1	0	0	–	0	0
1–7:	1	0	0	–	0	0	1	0	0
2–3:	–	1	1	0	0	0	0	0	0
2–5:	0	1	0	0	1	0	0	–	0
2–8:	0	1	0	0	–	0	0	1	0
3–6:	0	0	1	0	0	1	0	0	–
3–9:	0	0	1	0	0	–	0	0	1
4–5:	0	0	0	1	1	–	0	0	0
4–6:	0	0	0	1	–	1	0	0	0
4–7:	–	0	0	1	0	0	1	0	0
5–6:	0	0	0	–	1	1	0	0	0
5–8:	0	–	0	0	1	0	0	1	0
6–9:	0	0	–	0	0	1	0	0	1
7–8:	0	0	0	0	0	0	1	1	–
7–9:	0	0	0	0	0	0	1	–	1
8–9:	0	0	0	0	0	0	–	1	1

Now we have 18 ternary vectors describing all the possible positions with two Cardinals on the board. For the next sequence of intersections the number of the third vector can be selected larger than the number of the second vector (such as 1–2–3) in order to avoid duplications. The intersection with the nine original vectors already produces the final solutions. It is quite easy to see and to understand the properties of the solutions such as symmetry etc.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1–2–3:	1	1	1	0	0	0	0	0	0
1–4–7:	1	0	0	1	0	0	1	0	0
2–5–8:	0	1	0	0	1	0	0	1	0
3–6–9:	0	0	1	0	0	1	0	0	1
4–5–6:	0	0	0	1	1	1	0	0	0
7–8–9:	0	0	0	0	0	0	1	1	1

Exercise 6.10.
The given methodology allows easily to solve this problem as well. Only the equations for the queens have to change.

1 A pawn on $(4,4)$ changes the values for the fourth column, the fourth row and the diagonals. We go back to the queen on the field $(1,1)$ and show these changes:

$$
\begin{aligned}
&x_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{21}\overline{x}_{31}\overline{x}_{41}\overline{x}_{51}\overline{x}_{61}\overline{x}_{71}\overline{x}_{81}\overline{x}_{22}\overline{x}_{33}\overline{x}_{44}\\
&\vee\overline{x}_{11}x_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{22}\overline{x}_{32}\overline{x}_{42}\overline{x}_{52}\overline{x}_{62}\overline{x}_{72}\overline{x}_{82}\overline{x}_{21}\overline{x}_{23}\overline{x}_{34}\overline{x}_{45}\overline{x}_{56}\overline{x}_{67}\overline{x}_{78}\\
&\vee\overline{x}_{11}\overline{x}_{12}x_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{23}\overline{x}_{33}\overline{x}_{43}\overline{x}_{53}\overline{x}_{63}\overline{x}_{73}\overline{x}_{83}\overline{x}_{22}\overline{x}_{31}\overline{x}_{24}\overline{x}_{35}\overline{x}_{46}\overline{x}_{57}\overline{x}_{68}\\
&\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}x_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{24}\overline{x}_{34}\overline{x}_{44}\overline{x}_{23}\overline{x}_{32}\overline{x}_{41}\overline{x}_{25}\overline{x}_{36}\overline{x}_{47}\overline{x}_{58}\\
&\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}x_{15}\overline{x}_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{25}\overline{x}_{35}\overline{x}_{45}\overline{x}_{55}\overline{x}_{65}\overline{x}_{75}\overline{x}_{85}\overline{x}_{24}\overline{x}_{33}\overline{x}_{42}\overline{x}_{51}\overline{x}_{26}\overline{x}_{37}\overline{x}_{48}\\
&\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}x_{16}\overline{x}_{17}\overline{x}_{18}\overline{x}_{26}\overline{x}_{36}\overline{x}_{46}\overline{x}_{56}\overline{x}_{66}\overline{x}_{76}\overline{x}_{86}\overline{x}_{25}\overline{x}_{34}\overline{x}_{43}\overline{x}_{52}\overline{x}_{61}\overline{x}_{27}\overline{x}_{38}\\
&\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}x_{17}\overline{x}_{18}\overline{x}_{27}\overline{x}_{37}\overline{x}_{47}\overline{x}_{57}\overline{x}_{67}\overline{x}_{77}\overline{x}_{87}\overline{x}_{26}\overline{x}_{35}\overline{x}_{44}\\
&\vee\overline{x}_{11}\overline{x}_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{15}\overline{x}_{16}\overline{x}_{17}x_{18}\overline{x}_{28}\overline{x}_{38}\overline{x}_{48}\overline{x}_{58}\overline{x}_{68}\overline{x}_{78}\overline{x}_{88}\overline{x}_{27}\overline{x}_{36}\overline{x}_{45}\overline{x}_{54}\overline{x}_{63}\overline{x}_{72}\overline{x}_{81}\\
&\quad = 1.
\end{aligned}
$$

The same modification has to be implemented for the other columns (except column 4). Here we get two disjunctions $x_{41} \vee x_{42} \vee x_{43}$ and $x_{45} \vee x_{46} \vee x_{47} \vee x_{48}$ with the respective consequences which have to be included into the final equation. The intersection of the prepared 9 partial solution sets results in 10 solutions for 9 queens and one pawn on the field $(4,4)$.

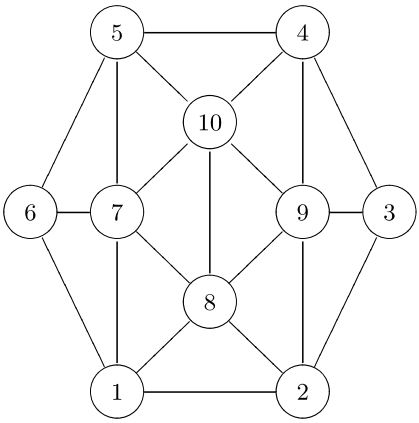
2 There are 128 positions without attacks of 9 queens and one pawn on an 8×8-chess board.

3 There are 2 positions without attacks of 10 queens and two pawns on the fields $(4,4)$ and $(5,5)$ on an 8×8-chess board.

4 There are 44 positions without attacks of 10 queens and two pawns on an 8×8-chess board.

Exercise 6.11.

1



2 Assuming the colors *red* (r), *green* (g), *blue* (b) and *yellow* (y) one coloring of the graph is: r: 1, 3, 10; g: 2, 4, 6; b: 5, 8; y: 7, 9.

3 The methodology might not be very difficult anymore when the previous ideas have been studied carefully. The biggest advantage of the approach is the fact that only structural properties of the graph have to be described and then all solutions will be found without any special considerations. It is suggested to generate a PRP by a piece of software and solve the task using the XBOOLE Monitor. Assuming the colors introduced above, we introduce Boolean variables in the following way:

$$x_{ir} = \begin{cases} 1 & \text{if the colour of node } i \text{ is red} \\ 0 & \text{otherwise.} \end{cases}$$

In the same way the following definitions will be used:

$$x_{ig} = \begin{cases} 1 & \text{if the color of node } i \text{ is green} \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ib} = \begin{cases} 1 & \text{if the color of node } i \text{ is blue} \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{iy} = \begin{cases} 1 & \text{if the color of node } i \text{ is yellow} \\ 0 & \text{otherwise.} \end{cases}$$

Now the structure of the graph can be taken into consideration. For node 1 it can be written:

$$(x_{1r} \vee x_{1g} \vee x_{1b} \vee x_{1y}).$$

The consequences can be split into two parts:

- When one node has a given color then this node cannot have the three other colors.
- When a node has a given color then the nodes connected to this node must have another color.

In this way we get for the first node

$$\begin{aligned} & x_{1r}\overline{x}_{1g}\overline{x}_{1b}\overline{x}_{1y}\overline{x}_{2r}\overline{x}_{6r}\overline{x}_{7r}\overline{x}_{8r} \\ & \vee x_{1g}\overline{x}_{1r}\overline{x}_{1b}\overline{x}_{1y}\overline{x}_{2g}\overline{x}_{6g}\overline{x}_{7g}\overline{x}_{8g} \\ & \vee x_{1b}\overline{x}_{1r}\overline{x}_{1g}\overline{x}_{1y}\overline{x}_{2b}\overline{x}_{6b}\overline{x}_{7b}\overline{x}_{8b} \\ & \vee x_{1y}\overline{x}_{1r}\overline{x}_{1g}\overline{x}_{1b}\overline{x}_{2y}\overline{x}_{6y}\overline{x}_{7y}\overline{x}_{8y}. \end{aligned}$$

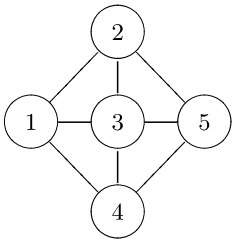
Such expressions can be created for each node of the graph. It is helpful to specify these expressions by TVLs, one for each node. Since there are 10 nodes and 4 colors, the problem requires 40 Boolean variables for a complete description of the problem. The intersection of the 10 TVLs for the 10 nodes shows all possible solutions of the problem.

4 There are 576 valid colorings of the graph.

5 The solution is not unique because permutations of the colors allow 576 colorings of the graph.

Exercise 6.12.

1



2 Boolean variables are introduced for each edge and each direction as follows:

$$x_{ik} = \begin{cases} 1 & \text{if the edge is used from node } i \text{ to node } k \\ 0 & \text{otherwise.} \end{cases}$$

16 Boolean variables are needed for both directions of the 8 existing edges. In node 1 there are three edges available, one to node 2, one to node 3 and one to node 4. Because each node can be used only once, the use of one edge forbids the use of other edges starting or ending on the same node on the same node as well as the use of this edge in the reverse direction. For the edge from node 1 to node 2 the following conjunction describes these restrictions:

$$x_{12}\overline{x}_{13}\overline{x}_{14}\overline{x}_{21}\overline{x}_{32}\overline{x}_{52}.$$

For the starting node, incoming edges are not forbidden: x_{12} will still allow x_{31} or x_{41}. Associated to the node 1 we get for x_{13} and x_{14}:

$$\begin{aligned} & \overline{x}_{12}x_{13}\overline{x}_{14}\overline{x}_{23}\overline{x}_{31}\overline{x}_{43}\overline{x}_{53}, \\ & \overline{x}_{12}\overline{x}_{13}x_{14}\overline{x}_{34}\overline{x}_{41}\overline{x}_{54}. \end{aligned}$$

These edges are all edges related to the node 1, and one of them must be used which can be expressed by a common TVL in disjunctive form. In this way the requirements for the other nodes are described.

Since these five situations must be considered simultaneously, the result of intersection of these five TVLs is the solution of all Hamiltonian paths we are searching for.

```
space 32 1
tin 1 1
x12 x13 x14 x21 x23 x25 x31 x32 x34 x35 x41 x43 x45 x52 x53 x54.
1000---0-----0--
010-0-0----0--0-
001-----0-0----0.
tin 1 2
x12 x13 x14 x21 x23 x25 x31 x32 x34 x35 x41 x43 x45 x52 x53 x54.
0--1000---0-----
---010-0---0--0-
---001---0--00--.
tin 1 3
x12 x13 x14 x21 x23 x25 x31 x32 x34 x35 x41 x43 x45 x52 x53 x54.
-0-0--10000-----
0---0-0100---0--
--0---0010-0---0
-----00001--0-0-.
tin 1 4
x12 x13 x14 x21 x23 x25 x31 x32 x34 x35 x41 x43 x45 x52 x53 x54.
--00--0---100---
-0--0---0-010-0-
-----0---0001--0.
tin 1 5
x12 x13 x14 x21 x23 x25 x31 x32 x34 x35 x41 x43 x45 x52 x53 x54.
0----0-0-----100
-0--0----0-0-010
--0-----0---0001.
isc 1 2 6
isc 6 3 6
isc 6 4 6
isc 6 5 6
```

3 There are eight Hamiltonian paths for the given graph. It is not possible to split the graph into subgraphs which have at least one Hamiltonian path each.

« » O TVL 6 (ODA) | 16 Var. | 8 R. | Sp. 1

	x12	x13	x14	x21	x23	x25	x31	x32	x34	x35	x41	x43	x45	x52	x53	x54
1:	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1
2:	1	0	0	0	0	1	1	0	0	0	0	1	0	0	0	1
3:	1	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0
4:	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0
5:	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1
6:	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	0
7:	0	0	1	1	0	0	0	0	0	1	0	1	0	1	0	0
8:	0	0	1	0	1	0	1	0	0	0	0	0	1	1	0	0

Exercise 6.13.

1 Hamilton path 1: a1—c2—d4—e2—c1—a2—c3—b1—a3—b5—d6—f5—e3—f1—d2—b3—a1
Hamilton path 2: a6—b4—d5—f6—e4—f2—d1—b2—a4—b6—c4—a5—c6—e5—f3—e1—d3—f4—e6—c5—a6

These Hamilton paths cover all fields of the chess board exactly once. As additional task you should combine this set of two Hamilton paths to a single one.

2 The number of Boolean variables is equal to the sum of the number of edges that exist to move from one node to another one. Modeling the path of the knights on a chess board, these numbers are equal to the allowed moves of the knight depending on the field. The matrix on the right enumerates these values. Hence, the number of required Boolean variables is 160.

$$\begin{bmatrix} 2 & 3 & 4 & 4 & 3 & 2 \\ 3 & 4 & 6 & 6 & 4 & 3 \\ 4 & 6 & 8 & 8 & 6 & 4 \\ 4 & 6 & 8 & 8 & 6 & 4 \\ 3 & 4 & 6 & 6 & 4 & 3 \\ 2 & 3 & 4 & 4 & 3 & 2 \end{bmatrix}$$

3 When the knight is on the field `a1`, then there are two possibilities for the first move: $a1 - c2$ or $a1 - b3$. The following definition of Boolean variables can be used:

$$x_{a1-c2} = \begin{cases} 1 & \text{if the knight moves from a1 to c2} \\ 0 & \text{otherwise.} \end{cases}$$

When the move `a1-c2` is used, then other moves starting from `a1` are forbidden. In this case only `a1-b3` must be excluded. The reverse move `c2-a1` is forbidden as well. Because the field `c2` is taken and cannot be used anymore, all basically possible moves to this field must be excluded finally.

Therefore we get altogether the following conjunctions for the field `a1` which can be used alternatively:

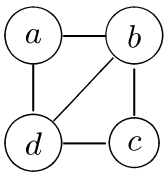
$$x_{a1-c2}\overline{x}_{a1-b3}\overline{x}_{c2-a1}\overline{x}_{a3-c2}\overline{x}_{b4-c2}\overline{x}_{d4-c2}\overline{x}_{e3-c2}\overline{x}_{e1-c2},$$
$$x_{a1-b3}\overline{x}_{a1-c2}\overline{x}_{b3-a1}\overline{x}_{a5-b3}\overline{x}_{c5-b3}\overline{x}_{d4-b3}\overline{x}_{d2-b3}\overline{x}_{c1-b3}.$$

The same principle must be applied to define the condition for the other fields. The intersection of these 36 TVLs will show all the possible sets of Hamiltonian paths of the knight where each set covers each field exactly once.

4 A program should be written that generates the required PRP from a 36×36 adjacency matrix of the graph that describes the allowed moves of the knight. The intersections of the 36 TVLs must be executed in an order that restricts the number of vectors of the intermediate results. There are 185,868 sets of disjoint Hamiltonian cycles that cover all fields of the 6×6 chess board exactly once.

Exercise 6.14.

1



2 The sketch of this graph shows that this is simply a rectangle with an additional diagonal. Eulerian paths can go clockwise around the rectangle followed by the diagonal: `b-c-d-a-b-d`, counterclockwise around the rectangle followed by the diagonal: `b-a-d-c-b-d`, starting with the diagonal, e.g.: `d-b-c-d-a-b` or even taking the diagonal in the middle of the path, e.g.: `b-c-d-b-a-d`.

The introduction of Boolean variables follows the same idea that has been used already very often:

$$ab = \begin{cases} 1 & \text{if the edge is used from a to b} \\ 0 & \text{otherwise.} \end{cases}$$

This simple example shows nodes with an even (`a, c`) or an odd (`b, d`) number of adjacent edges. All edges can be used only in the case when the number of edges used to move to the node is equal the number of edges used to leave the node. Due to odd numbers of edges additional variables for the start node and the end node of Eulerian path can solve this requirement.

$$as = \begin{cases} 1 & \text{if the Eulerian path starts from node a} \\ 0 & \text{otherwise,} \end{cases}$$
$$ae = \begin{cases} 1 & \text{if the Eulerian path ends on node a} \\ 0 & \text{otherwise.} \end{cases}$$

In case of an odd number of adjacent edges to a node the Eulerian path must start *or* end on that node in order to cover all these edges. In case of an even number of adjacent edges to a node the Eulerian path must start *and* end on that node or uses simply one edge to move to the node and one edge to move from the node in order to cover all these edges. An additional requirement for Eulerian paths is exactly one start node and exactly one end node of the path. The solution vectors describe the combination of the directions to use the edges. One solution can cover several Eulerian paths.

```
3 space 32 1
  avar 1
  ab ba bc cb cd dc da ad bd db as ae bs be cs ce ds de.

  tin 1 1          tin 1 2          tin 1 3          tin 1 4
  ab ba ad da      ba ab bc cb      cb bc cd dc      da ad dc cd
  as ae.           bd db bs be.     cs ce.           db bd ds de.
  100100           10010101         100100           10010101
  011000           01100101         011000           01100101
  100111           01011001         100111           01011001
  011011.          01101010         011011.          01101010
                   10011010                          10011010
                   10100110.                         10100110.


  tin 1 5               tin 1 6               isc 1 2 7
  as bs cs ds.          ae be ce de.          isc 7 3 7
  1000                  1000                  isc 7 4 7
  0100                  0100                  isc 7 5 7
  0010                  0010                  isc 7 6 7
  0001.                 0001.
```

There are six Eulerian paths for the given graph. All of them start on node `b` and end on node `d` or vice versa.

« » 0 TVL 7 (ODA) | 18 Var. | 6 R. | Sp. 1

	ab	ba	bc	cb	cd	dc	da	ad	bd	db	as	ae	bs	be	cs	ce	ds	de
1:	1	0	1	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0
2:	1	0	0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0
3:	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	0	0	1
4:	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0	1	0
5:	0	1	0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	1
6:	0	1	1	0	1	0	0	1	0	1	0	0	1	0	0	0	0	1

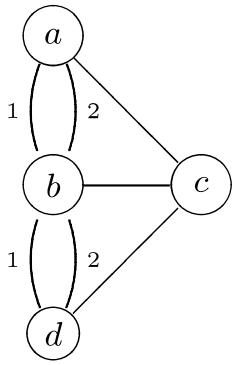
4 If the diagonal edge is removed, an Eulerian circuit around the rectangle in both directions exists. Adding a second edge in parallel to the given diagonal edge leads to Eulerian circuits too. The calculation of the Eulerian circuits can be done in the same way as shown for Eulerian paths. Of couse, the restriction TVLs for the nodes must be adapted to the number of adjacent edges. Results of the intersection of the Eulerian paths with the TVL

as	*ae*	*bs*	*be*	*cs*	*ce*	*ds*	*de*
1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1

which are not empty confirm that Eulerian circuits were calculated.

Exercise 6.15.

1



2 Basically the Boolean model is an extension of the Boolean model of the previous task. Separate variables must be introduced for each of the parallel bridges. There are three edges at the nodes `a`, `c`, and `d`, therefore the restriction for the nodes `b` or `d` of the previous task can be reduced. An extended restriction must be created for the node `b` of the "island". Two edges used for moves to this node imply three edges for moves from this node. Consequently this node must be an ending node for the Eulerian path. The only possible alternative for the node `b` are three edges used for moves to this node and two edges for moves from it which requires that this node is a starting node for the Eulerian path.

3
```
space 32 1
avar 1
ab1 ba1 ab2 ba2 ac ca bc cb cd dc bd1 db1 bd2 db2
as ae bs be cs ce ds de.
```

```
tin 1 1
ab1 ba1 ab2 ba2 ac
ca as ae.
10010101
01100101
01011001
01101010
10011010
10100110.
tin 1 3
ca ac cb bc cd dc
cs ce.
10010101
01100101
01011001
01101010
10011010
10100110.
tin 1 4
dc cd db1 bd1 db2
bd2 ds de.
10010101
01100101
01011001
01101010
10011010
10100110.
```

```
tin 1 2
ba1 ab1 ba2 ab2
bc cb bd1 db1
bd2 db2 bs be.
101010010110
101001100110
101001011010
100110100110
100110011010
100101101010
011010100110
011010011010
011001101010
010110101010
011010010101
011001100101
011001011001
010110100101
010110011001
010101101001
101010100101
101010011001
101001101001
100110101001.
```

```
tin 1 5
as bs cs ds.
1000
0100
0010
0001.
tin 1 6
ae be ce de.
1000
0100
0010
0001.
isc 1 2 7
isc 7 3 7
isc 7 4 7
isc 7 5 7
isc 7 6 7
```

The empty TVL 7 confirms that there is no Eulerian path over the bridges of Königsberg.

4 Removing the edge between the node `b` and `c` does not influence the restrictions for the nodes `a` and `d`. The changed node `c` is connected by two edges and can be modeled like node `c` of Exercise 6.14.

5 In a copy of the previous PRP where all object numbers are increased by 100 the following TVLs are substituted.

```
tin 1 102
ba1 ab1 ba2 ab2
bd1 db1 bd2 db2
bs be.
1010010100
1001100100
1001011000
0110100100
0110011000
0101101000
1010010111
1001100111
1001011011
0110100111
0110011011
0101101011.
```

```
tin 1 103
ca ac cd dc cs ce.
100100
011000
100111
011011.
```

There are ten solution vectors for the modified problem.

« » 0 TVL 107 (ODA) | 20 Var. | 10 R. | Sp. 1

	ab1	ba1	ab2	ba2	ac	ca	cd	dc	bd1	db1	bd2	db2	as	ae	bs	be	cs	ce	ds	de
1:	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	1	0
2:	1	0	0	1	0	1	0	1	0	1	1	0	0	1	0	0	0	0	1	0
3:	0	1	1	0	0	1	0	1	1	0	0	1	0	1	0	0	0	0	1	0
4:	0	1	1	0	0	1	0	1	0	1	1	0	0	1	0	0	0	0	1	0
5:	0	1	0	1	1	0	1	0	0	1	0	1	0	1	0	0	0	0	1	0
6:	0	1	1	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	1
7:	0	1	1	0	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1
8:	1	0	0	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	1
9:	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1
10:	1	0	1	0	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	1

Exercise 6.16.

1

				5				2
			6					9
	1	8	4					
	2	3		7				
		5		6		2		
						6	4	
					2	8	5	
9					1			
7				3				

2 The game can be modeled by the variables:

$$x_{i,j,k} = \begin{cases} 1 & \text{if } k \text{ is on the field } (i,j) \\ 0 & \text{otherwise} \end{cases}$$

where $1 \leq i \leq 9$, $1 \leq j \leq 9$, and $1 \leq k \leq 9$ as well. Hence, $9^3 = 729$ Boolean variables are needed. In order to restrict such a gigantic search space, the given values and the resulting restrictions should be used first in an intersection of the requirements and restrictions of the game. For a given value

- all the other values on the same field,
- the given value on the same row,
- the given value on the same column, and
- the given value on the same subsquare

are forbidden. Therefore ternary vectors having a single value 1 and 28 associated values 0 can be created. It is suggested to generate a 729×729 ternary matrix that includes all these vectors by a piece of software. From this TVL the fitting ternary vectors of the given values can be selected using the STV operation of the XBOOLE Monitor. The result of the intersection of these ternary vectors is a single ternary vector, if the given values do not include any contradiction.

Now the requirements of the game must be considered. The rule *there must be the value k in one of the columns j of the row i* can be expressed by the disjunction of the appropriate 9 ternary vectors of the prepared TVL. 81 intersections for all values and all rows are sufficient to solve the Sudoku game completely. The other rules are used implicitly due to the strong restrictions in the generated large TVL. In spite of the large number of Boolean variables it needs much less the one second to solve the game using the 729×729 ternary matrix which must be generated only once.

3 The given Sudoku has the following solution:

4	9	6	1	**5**	3	7	8	**2**
5	3	7	**6**	2	8	4	1	**9**
2	**1**	**8**	**4**	9	7	5	3	6
6	**2**	**3**	8	**7**	4	1	9	5
1	4	**5**	3	**6**	9	**2**	7	8
8	7	9	2	1	5	**6**	**4**	3
3	6	1	9	4	**2**	**8**	**5**	7
9	5	2	7	8	**1**	3	6	4
7	8	4	5	**3**	6	9	2	1

Chapter 7

COMBINATORIAL CIRCUITS

1. The Circuit Model

A combinatorial circuit is realized by a structure of basic elements called gates. The gates of the circuit are visualized in a schematic diagram by symbols listed in Table 7.1 of [18] together with their logic function. The connection wires between the gates of a circuit are expressed by lines in the schematic diagram.

A combinatorial circuit possesses a unique behavior, but the same behavior may be realized by different circuit structures. Therefore both structural and behavioral models are necessary in order to describe a combinatorial circuit.

The main tasks for combinatorial circuits preform transformations between these classes of models. The basic task of analysis requires a structural model of a combinatorial circuit and creates the associated behavioral model. Vice versa, the basic task of synthesis takes a behavioral model of a combinatorial circuit and leads to structural models of one or several associated circuits.

This chapter includes both exercises for analysis and synthesis of combinatorial circuits. The required models are prepared in this section. It is strongly recommended to store all the solution of each exercise because succeeding exercises may use results of previous exercises. For comfortable work and sometimes required corrections it is suggested to store for each exercise

- your own prepared problem programs (PRPs),
- complete TVL systems created in the XBOOLE Monitor, and

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

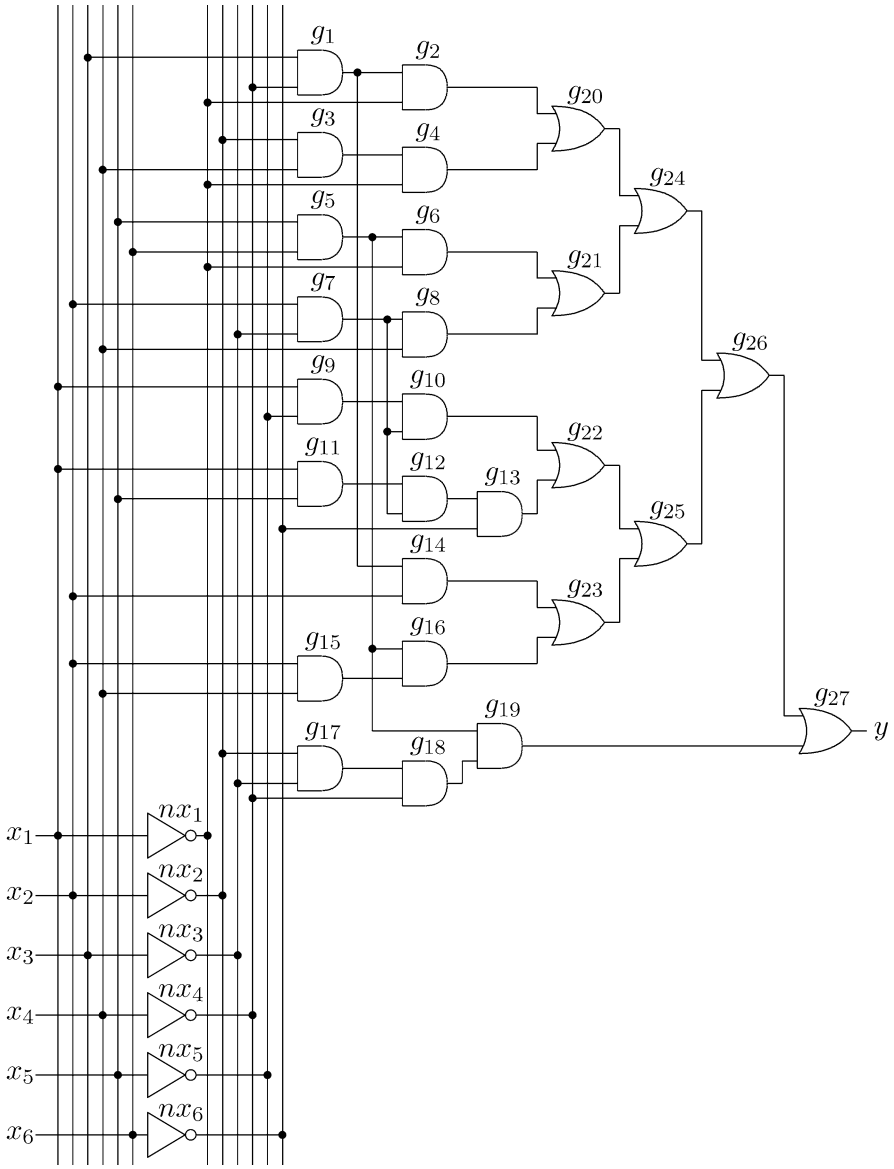


Figure 7.1 Structure of a circuit using AND- and OR-gates restricted to two inputs

- the PRP that documents all executed steps to solve all tasks of an Exercise which is generated by the menu item `Extras` – `Save Protocol as PRP`.

One structural model is a system of logic equations, where each equation describes one gate and the connections are expressed by using the same names of variables in different equations. The behavior of each gate is specified by the operations applied in the associated equation.

Exercise 7.1 (Structural Model – System of Logic Equations). Create a system of logic equations as structural model of the combinatorial circuit shown as schematic diagram in Fig. 7.1. Prepare this system of equations as PRP so that it can be used later on for an analysis task. It is helpful to use the names of the gates as names of their output signals.

Alternatively a set of local lists of phases can be used as a structural model of a combinatorial circuit. In this case the phases describe the behavior of the gates, and the same names of column variables express connections.

Exercise 7.2 (Structural Model – Set of Local Lists of Phases). Create a set of local lists of phases as structural model of the combinatorial circuit shown as schematic diagram in Fig. 7.1. Prepare this set of local lists of phases as PRP such that it can be used later on for an analysis task. It is helpful to use the names of the gates as names of their output signals. The required phases can be taken from Table 7.1 of [18].

In the special case of two level circuits a TVL in d-form expresses the structure of an AND-OR or a NOR-NOR circuit where the conjunctions of the associated disjunctive form are enumerated in the rows and the OR-gate is described implicitly by the form of the TVL. As shown in Fig. 7.5 of [18], other circuit structures correspond directly to TVLs in c-form, a-form or e-form.

Exercise 7.3 (Structural Model – TVL in a Certain Form). Assume the circuit structure in Fig. 7.1 may by transformed into a two level AND-OR structure such that the conjunctions of the disjunctive form are built by the AND-gates connected directly with an OR-gate. Describe the associated structural model by a TVL in d-form.

The behavior of a combinatorial circuit can be described by a system of explicit logic equations in which the output values y_i appear on the left-hand side only while on the right-hand sides expressions of the input variables are given. Such a specification defines a set of completely specified functions but is not easy to understand. More comprehensible is the solution of such an equation system expressed by a TVL in ODA-form. This TVL can be interpreted as system function $F(\mathbf{x}, \mathbf{y})$ where the equation $F(\mathbf{x}, \mathbf{y}) = 1$ possesses the same solution.

Exercise 7.4 (Behavioral Model – Explicit Equation System – System Function of a Completely Specified Circuit). The behavior of a circuit is

given by the equation system (7.1),

$$\begin{aligned} y_1 = &((x_1(x_2 \oplus (\overline{x}_3 \vee x_4)) \vee (x_5 \oplus x_6) \vee x_2\overline{x}_3\overline{x}_4) \\ &\oplus (x_1x_3x_4(\overline{x}_5 \oplus x_6))) \vee x_2\overline{x}_3x_4, \\ y_2 = &(\overline{x}_1x_5\overline{x}_6 \vee x_1x_2\overline{x}_3 \vee x_1\overline{x}_3(\overline{x}_2 \vee x_4) \vee \overline{x}_5x_6(\overline{x}_1\overline{x}_3 \vee x_4)) \\ &\oplus \overline{x}_4\overline{x}_5x_6. \end{aligned} \tag{7.1}$$

Calculate a minimized system function $F(\mathbf{x}, \mathbf{y})$. Practical tasks:

1 Prepare a Boolean space and attach the input and output variables in a convenient order.

2 Solve the equation system (7.1).

3 Minimize the solution and show this list of phases that can be interpreted as system function $F(\mathbf{x}, \mathbf{y})$.

The system function in Fig. 7.2 a) describes in 17 rows for all 64 input patterns which values are associated to the outputs y_1 and y_2. Alternatively the more compact TVL for the functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ can be used as behavioral description.

Exercise 7.5 (Behavioral Model – System of Function TVLs of a Completely Specified Circuit). Separate the system function of Exercise 7.4 into a system of function TVLs using formula (7.56) of [18]. Practical tasks:

1 Load the TVL system of Exercise 7.4.

2 Prepare solutions of the simple equations $y_1 = 1$, $y_2 = 1$, and a VT $\langle y_1, y_2 \rangle$.

3 Calculate the function TVLs $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$.

4 Show the function TVLs $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ and evaluate the results.

The system function $F(\mathbf{x}, \mathbf{y})$ of Exercise 7.4 describes the completely specified behavior of a combinatorial circuit. Generally a system function $F(\mathbf{x}, \mathbf{y})$ can also be described as incompletely specified behavior or even a behavior not realizable by a combinatorial circuit. An incomplete behavior gives some freedom that can be exploited in the synthesis of the circuit. Alternatively to the system function $F(\mathbf{x}, \mathbf{y})$ of an incomplete function each pair of the *mark functions* $f_q(\mathbf{x})$ for the ON-set, $f_r(\mathbf{x})$ for the OFF-set, and $f_\varphi(\mathbf{x})$ for the don't-care-set can be used.

Exercise 7.6 (Behavioral Model – Incompletely Specified Circuit). Verify that the system function $F(\mathbf{x}, y)$ (7.2) describes an incompletely specified function.

$$\begin{aligned} F(\mathbf{x}, y) = ((&x_1\overline{x}_2(x_3 \oplus x_4)y \vee x_3\overline{x}_4(x_5 \oplus x_6)\overline{y} \vee \overline{x}_1x_7y \\ &\vee x_1x_2x_6\overline{x}_7) \oplus \overline{x}_6) \vee (x_3 \oplus \overline{x}_4)x_6 \vee x_3\overline{x}_4x_5 \\ &\vee \overline{x}_1\overline{x}_3x_6\overline{x}_7 \vee x_1x_2(x_3 \oplus x_4)x_6x_7. \end{aligned} \tag{7.2}$$

Calculate all three associated mark functions and verify that these functions are pairwisely disjoint. Reconstruct the function $F(\mathbf{x}, y)$ based on each pair of mark functions and verify the result. Practical tasks:

1 Prepare a Boolean space and attach the input and output variables in a convenient order.

2 Solve (7.2).

3 Prepare solutions of the simple equations $y = 1$.

4 Calculate the mark function $f_\varphi(\mathbf{x})$ of the don't-care set based on formula (7.24) of [18].

5 Calculate the mark function $f_q(\mathbf{x})$ of the ON-set based on formula (7.25) of [18].

6 Calculate the mark function $f_r(\mathbf{x})$ of the OFF-set based on the formula (7.25) of [18].

7 Verify whether the three mark functions are pairwisely disjoint and cover the Boolean space completely.

8 How many functions includes the characteristic function set $F^C(\mathbf{x})$ which is specified by the calculated mark functions?

9 Calculate the system function using the mark functions $f_q(\mathbf{x})$ and $f_r(\mathbf{x})$ based on formula (7.21) of [18] and verify the result.

10 Calculate the system function using the mark functions $f_\varphi(\mathbf{x})$ and $f_r(\mathbf{x})$ based on formula (7.22) of [18] and verify the result.

11 Calculate the system function using the mark functions $f_q(\mathbf{x})$ and $f_\varphi(\mathbf{x})$ based on formula (7.23) of [18] and verify the result.

2. Analysis

The basic analysis task for combinatorial circuits is the calculation of the behavior realized by the circuit. The set of solution vectors of the system of logic equations given as structural model describes the global list of phases of the associated combinatorial circuit. Hence, it is a behavioral model that can be interpreted as system function $F(\mathbf{x}, y, \mathbf{nx}, \mathbf{g})$. Notice, it supports the clearness when the variables are well ordered in the phases.

Exercise 7.7 (Behavior of a Combinatorial Circuit Based on a System of Logic Equations). Calculate the behavior of the circuit given in Fig. 7.1. Use the system of logic equations prepared in Exercise 7.1 as structural model. Practical tasks:

1 The output y depends on 6 inputs. How many phases exist?

2 Define a Boolean space, large enough for this task, and attach the variables for well-ordered phases.

3 Solve the system of equations prepared in Exercise 7.1 and show the global list of phases.

Each structural model of a combinatorial circuit can be used in order to calculate its global behavior.

Exercise 7.8 (Behavior of a Combinatorial Circuit Based on a Set of Local Lists of Phases). Calculate the behavior of the circuit given in Fig. 7.1. Use the set of local lists of phases prepared in Exercise 7.2 as structural model and verify whether the calculated behavior coincides with the result of Exercise 7.7. Practical tasks:

1 Load the final TVL system of Exercise 7.7.

2 Create the set of local lists of phases executing the PRP prepared in Exercise 7.2.

3 Calculate the intersection of all 33 local lists of phases of the gates.

4 Verify whether the behavior calculated on the basis of local lists of phases coincides with the behavior of the circuit calculated by solving a system of equations in Exercise 7.7 which is available as object number 1.

A more compact representation of the behavior takes the black box view of the combinatorial circuit into account. Such a restricted global list of phases includes the values of the input and output variables only, and can be interpreted as input-output system function $F(\mathbf{x}, y)$.

Exercise 7.9 (Input-Output Behavior of a Combinatorial Circuit). Calculate the input-output behavior of the circuit given in Fig. 7.1. Minimize the result to an orthogonal TVL. Practical tasks:

1 Which operation of the Boolean differential calculus calculates the list of phases of the input-output-behavior from a given global list of phases that includes the values of all gates? Motivate your answer!

2 Load the final TVL system of Exercise 7.7.

3 Prepare a VT of the internal variables.

4 Calculate the input-output-behavior.

5 Minimize the input-output-behavior to a short orthogonal TVL.

The global list of phases is a complete model of the behavior that is or has to be realized in a combinatorial circuit. The global list of phases calculated in Exercise 7.7 depends on only 6 input variables, but together with all signals of the circuit the 64 rows and 39 columns require a whole page to print this TVL (see Fig. 7.3 on page 170). For larger circuits it is more convenient to use the global list of phases as basis of several types of simulation. In fact, there is no restriction answering questions about relationships between selected values of the behavior.

Exercise 7.10 (Simulation Based on a Global List of Phases). Execute several simulations of a modified circuit created from the circuit structure in Fig. 7.1. The circuit is extended such that additional outputs y_2, y_3, and y_4 are introduced which are connected with the outputs of the gates g_{20}, g_{11}, and g_{23} in the given order. Practical tasks:

1 Load the TVL system of Exercise 7.7.

2 Extend the behavior by the outputs y_2, y_3, and y_4 as specified in this Exercise.

3 Prepare VTs for the inputs, the internal signals, and the outputs.

4 Execute a forward simulation based on formula (7.52) of [18] that finds the values of the four outputs for the input pattern $(x_1, x_2, x_3, x_4, x_5, x_6)$ that are equal to either (100010) or (011010). Calculate both the simulation pattern $(\mathbf{x}, \mathbf{y})$ and the restricted simulation result $(\mathbf{y})$. Evaluate the simulation result.

5 Execute a backward simulation based on formula (7.53) of [18] that finds the values of the six inputs for the output pattern (y, y_2, y_3, y_4) that are equal to either (1010) or (0101). Calculate both the simulation pattern

$(\mathbf{x}, \mathbf{y})$ and the restricted simulation result $(\mathbf{x})$. Evaluate the simulation result.

6 Execute a general simulation based on formula (7.54) of [18] that finds the values of the six inputs and four outputs for the pattern (g_1, g_5, g_7) of the reused internal signals that are equal to either (010) or (101). Calculate the restricted simulation result $(\mathbf{x}, \mathbf{y})$ and evaluate it.

The simulations executed in Exercise 7.10 were controlled by single vectors of values. The applied method is usable without any changes for sets of vectors, too. The analysis of a global list of phases allows furthermore the identification of important information for a subsequent synthesis task. In some applications the outputs of one combinatorial circuit control the inputs of another such circuit. If not all output patterns appear on the output of the predecessor circuit, a don't-care function can simplify the successor circuit.

Exercise 7.11 (Don't-Care Function Defined by the Outputs of a Global List of Phases). Calculate both the output pattern of the circuit modified in Exercise 7.10 and the don't-care function caused by this circuit for a successor circuit. Practical tasks:

1 Load the TVL system of Exercise 7.10.

2 What is the maximal number of output patterns that can be observed on a circuit having four outputs.

3 Calculate the output pattern of the circuit modified in Exercise 7.10. How many patterns can be seen for the outputs (y, y_2, y_3, y_4). Minimize this TVL and show the result.

4 Calculate the don't-care function $f_\varphi(y, y_2, y_3, y_4)$ caused by the circuit modified in Exercise 7.10 using formula (7.55) of [18]. Show the calculated don't-care function.

Further analysis tasks investigate the behavioral description with regard to certain properties. In the equation system (7.1) appear the variables $x_i, i = 1, \ldots, 6$, for both functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$. Figure 7.2 c) on page 168 shows that the function $y_2(\mathbf{x})$ calculated in Task 3 of Exercise 7.5 does not depend on x_2. The condition that only dashes appear in a column of a function TVL is sufficient, but not necessary for the independence of a function regarding a variable.

Exercise 7.12 (Independence of a Function Regarding Variables). Analyze whether the functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ calculated in Task 3 of Exercise 7.5 really depend on the variables $(x_1, x_2, x_3, x_4, x_5, x_6)$ and simplify these functions if possible. Practical tasks:

1 Load the TVL system of Exercise 7.5. This TVL system includes the function $y_1(\mathbf{x})$ as object number 11 and the function $y_2(\mathbf{x})$ as object number 12.

2 Prepare a PRP that checks whether a function depends on the variables $(x_1, x_2, x_3, x_4, x_5, x_6)$.

3 Check for the function $y_1(\mathbf{x})$ the variables it is depending on.

4 Check for the function $y_2(\mathbf{x})$ the variables it is depending on.

5 Simplify the function $y_1(\mathbf{x})$ as much as possible and verify the result.

6 Simplify the function $y_2(\mathbf{x})$ as much as possible and verify the result.

A system function $F(\mathbf{x}, \mathbf{y})$ specifies all allowed phases of a circuit. A combinatorial circuit possesses for each input pattern an associated output pattern. Consequently only such system functions describe realizable circuits which allow for each input pattern at least one output pattern. The analysis whether a system function is realizable is a necessary task before starting a design.

Exercise 7.13 (Realizability of a System Function). Check whether the system functions of Exercises 7.5 and 7.6 are realizable. Practical tasks:

1 Load the TVL system of Exercise 7.5. This TVL system includes the system function $F(\mathbf{x}, y_1, y_2)$ as object number 1 and the VT $\langle y_1, y_2 \rangle$ as object number 5.

2 Check whether the loaded system function is realizable using formula (7.64) of [18].

3 Load the TVL system of Exercise 7.6. This TVL system includes the system function $F(\mathbf{x}, y)$ as object number 1 and a solution of $y = 1$ as object number 2.

4 Check whether the loaded system function is realizable using formula (7.64) of [18].

Assume the system function is realizable. Then $F(\mathbf{x}, \mathbf{y}) = 1$ can be solved with regard to $\mathbf{y}$ whereas one or more functions for each selected output of the circuit may exist. It can be analyzed whether the system function specifies a selected output function uniquely.

Exercise 7.14 (Unique Solution of a System Function with Regard to an Output). From Exercise 7.13 it is known that the system functions of Exercises 7.5 and 7.6 are realizable. Check whether these system functions have unique solutions with regard to their outputs. Practical tasks:

1 Load the TVL system of Exercise 7.5. This TVL system includes the system function $F(\mathbf{x}, y_1, y_2)$ as object number 1.

2 Check whether the loaded system function is uniquely realizable using formulas (7.66) and (7.67) of [18].

3 Load the TVL system of Exercise 7.6. This TVL system includes the system function $F(\mathbf{x}, y)$ as object number 1 and a solution of $y = 1$ as object number 2.

4 Check whether the loaded system function is uniquely realizable using formula (7.67) of [18].

It was analyzed in Exercise 7.12 whether a completely specified function depends on all variables used in its specification. This analysis can be generalized for incompletely specified functions (ISF). Basically each completely specified function of the characteristic function set associated to an incompletely specified function can be explored separately. The incompletely specified function specified in Exercise 7.6 can be realized by 2^{98} functions. If each of these functions is checked within 100 nanoseconds it takes approximately 10^{12} years to solve this task completely using the mentioned method. Hence, another analysis approach is required.

Using formula (7.65) of [18] it can be explored whether the characteristic function set associated to the incompletely specified function $y = f(\mathbf{x})$ described by the system function $F(\mathbf{x}, y)$ includes a function that does not depend on the variable x_i. The more functions are covered by the function set, the larger is the probability to find such variables. If several such variables for a system function $F(\mathbf{x}, y)$ were found, it cannot be concluded that there is one function that does not depend on all of them. Thus a detailed analysis of subsets of the found variables is required. This analysis can be organized constructively starting with pairs of variables, or restrictively starting with the set of all possible variables.

Exercise 7.15 (Independence of an Incompletely Specified Function Regarding Variables). Analyze whether all functions represented by the system function $F(\mathbf{x}, y)$ specified in Exercise 7.6 really depend on the variables $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Find all functions described by $F(\mathbf{x}, y)$ that depend on the smallest number of variables. Practical tasks:

1 Load the TVL system of Exercise 7.6. This TVL system includes the system function $F(\mathbf{x}, y)$ as object number 1.

2 Check whether the system function $F(\mathbf{x}, y)$ includes functions that do not depend on one of the variables $(x_1, x_2, x_3, x_4, x_5, x_7)$.

3 Formula (7.65) of [18] allows to analyze the independence of $F(\mathbf{x}, y)$ with regard to a single variable. Generalize this formula for a set of variables. Substantiate this generalization.

4 Check whether the system function $F(\mathbf{x}, y)$ includes functions that do not depend on all variables detected in Task 2 of Exercise 7.15.

5 In Task 2 of Exercise 7.15 n variables were found for which the system function $F(\mathbf{x}, y)$ includes at least one function that does not depend on one of these variables. Check whether the system function $F(\mathbf{x}, y)$ includes functions that do not depend on $n-1$ variables detected in task 2 of Exercise 7.15.

6 Develop a formula that restricts the system function $F(\mathbf{x}_1, \mathbf{x}_2, y)$ such that functions depending on more variables than $\mathbf{x}_1$ only are excluded. Substantiate this formula.

7 Calculate all minimal functions described by $F(\mathbf{x}, y)$ that do not depend on the sets of variables found in Task 5 of Exercise 7.15 and show the results.

3. Design

The basic design task for combinatorial circuits is the calculation of a circuit structure that realizes a given behavior. The most common behavioral description is the system function $F(\mathbf{x}, \mathbf{y})$ that describes the allowed input-output phases.

A combinatorial circuit yields for each input pattern exactly one output pattern. Thus, the system function must allow at least one output pattern for each input pattern. Consequently, only for such system functions $F(\mathbf{x}, \mathbf{y})$ for which the equation $F(\mathbf{x}, \mathbf{y}) = 1$ can be solved with regard to $\mathbf{y}$, a combinatorial circuit exists. If this equation can be solved with regard to $\mathbf{y}$ there may be a unique solution or a set of solutions.

A system equation $F(\mathbf{x}, y) = 1$ can be solved with regard to y if

$$\max_y F(\mathbf{x}, y) = 1, \tag{7.3}$$

and can be solved with regard to y uniquely if

$$\frac{\partial F(\mathbf{x}, y)}{\partial y} = 1. \tag{7.4}$$

If a system equation $F(\mathbf{x}, y) = 1$ is uniquely solvable with regard to y, the logic function $y = f(\mathbf{x})$ can be calculated by (7.5).

$$y = f(\mathbf{x}) = F(\mathbf{x}, y = 1) = \max_y[(y \wedge F(\mathbf{x}, y))]. \tag{7.5}$$

Exercise 7.16 (Verify Whether a Combinatorial Circuit Exists for a System Function). Verify whether the system function calculated in Exercise 7.9 can be realized as combinatorial circuit. If that is possible, then calculate the logic function $y = f(\mathbf{x})$. Practical tasks:

1 Load the final TVL system of Exercise 7.9. This TVL system includes the system function $F(\mathbf{x}, y)$ as object 4.

2 Prepare a TVL for $y = 1$ as object number 5.

3 Verify whether $F(\mathbf{x}, y)$ given in object 4 can be solved with regard to y.

4 Verify whether $F(\mathbf{x}, y)$ given in object 4 can be solved with regard to y uniquely.

5 Calculate the logic function $y = f(\mathbf{x})$ that must be realized in the combinatorial circuit and show the result.

Basically it is possible to realize a given behavior by several structures of combinatorial circuits. The most explored basic architecture is the disjunctive form of a logic function which can be mapped on a two-level combinatorial circuit. A minimal circuit structure requires prime conjunctions in the disjunctive form. The calculation of all prime conjunctions is a NP-complete task. This task can be simplified if a sufficient number of prime conjunctions, but not all of them are calculated.

A simple method to find prime conjunctions was introduced in [18], page 295. A variable can be removed from a conjunction if this changed conjunction is covered completely by the function. Starting with a minimized TVL in ODA form instead of a BVL simplifies this task because several variables of the conjunctions have been removed in a ternary vector already.

The given conjunction of a function can be restricted to prime conjunctions using a PRP that must be applied repeatedly. The results of such a PRP are completely visible in the window `Spaces/Objects` in the XBOOLE Monitor and a value 0 in the column `Rows` indicates quickly which omissions of variables are allowed.

Exercise 7.17 (Prime Conjunctions of a Logic Function). Create all prime conjunctions of the function of Fig. 7.4 b) on page 172 calculated in Exercise 7.16 that can be built by restriction of the given ternary vectors. Collect each found prime conjunction only once. Practical tasks:

1 Load the final TVL system of Exercise 7.16.

2 Prepare a PRP that allows to check whether one of the variables can be removed from a conjunction.

3 Create prime conjunctions of the conjunction 1 in Fig. 7.4 b).

4 Create prime conjunctions of the conjunction 2 in Fig. 7.4 b).

5 Create prime conjunctions of the conjunction 3 in Fig. 7.4 b).

6 Create prime conjunctions of the conjunction 4 in Fig. 7.4 b).

7 Create prime conjunctions of the conjunction 5 in Fig. 7.4 b).

8 Create prime conjunctions of the conjunction 6 in Fig. 7.4 b).

9 Create prime conjunctions of the conjunction 7 in Fig. 7.4 b).

10 Create prime conjunctions of the conjunction 8 in Fig. 7.4 b).

11 Create prime conjunctions of the conjunction 9 in Fig. 7.4 b).

12 Create prime conjunctions of the conjunction 10 in Fig. 7.4 b).

13 Show all found prime conjunctions.

The simple method applied in Exercise 7.17 finds several, but in general not all prime conjunction of a function. Therefore their completeness should be studied.

Exercise 7.18 (Completeness of the Simple Set of Prime Conjunctions). Verify whether the 8 prime conjunctions calculated in Exercise 7.17, see Fig. 7.4 c), cover the function of Fig. 7.4 b) and whether all these prime conjunctions are necessary to cover this function. Practical tasks:

1 Substantiate why a disjunctive form of the 8 prime conjunctions of Fig. 7.4 c) must cover the function of Fig. 7.4 b) completely.

2 Load the final TVL system of Exercise 7.17.

3 Verify that the disjunctive form of the 8 prime conjunctions of Fig. 7.4 c) (object 15) covers the function of Fig. 7.4 b) (object 11).

4 Prepare a PRP that allows to check whether one of these 8 prime conjunctions can be removed without loss of the cover of the function of Fig. 7.4 b).

5 Verify that the 8 prime conjunctions of Fig. 7.4 c) describe a minimal disjunctive form of the function of Fig. 7.4 b).

The minimal disjunctive form of Fig. 7.4 c) on page 172 requires 8 prime conjunctions. That is one prime conjunction less than the given circuit in Fig. 7.1 on page 136 that realizes the same function. In Exercise 7.17 only a subset of all prime conjunctions was calculated. The

question arises whether there is a shorter disjunctive form of the function that uses any other prime conjunction. All prime conjunctions can be found using the consensus law, see [18] page 17. If exactly one variable appears in one conjunction negated and in a second conjunction not negated, a new conjunction that does not include this variable but all other variables of both conjunctions, can be added without change of the function. Such new conjunctions can be taken in order to create new prime conjunctions.

Exercise 7.19 (Additional Prime Conjunctions Based on the Consensus Law). Extend the set of prime conjunctions of Fig. 7.4 c) based on the consensus law. Take for this search all combinations of the given 8 prime conjunctions into account. Only such consensus conjunctions that cannot be absorbed by one of the known conjunctions must be taken as basis to check for new prime conjunctions. Such a check can be done using the PRP of Exercise 7.17 Task 2.

1 Load the final TVL system of Exercise 7.17.

2 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_1.

3 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_2.

4 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_3.

5 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_4.

6 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_5.

7 Extend the set of prime conjunctions of Fig. 7.4 c) by prime conjunctions based one such consensus conjunctions created by removing the variable x_6.

8 Show all found prime conjunctions.

The six new prime conjunctions can be the source of new consensus conjunctions as the basis for further prime conjunctions. The check for

additional consensus conjunctions must be executed between the six new prime conjunctions and all the other prime conjunctions. It is not necessary to repeat this procedure for pairs of conjunctions that were already analyzed.

Exercise 7.20 (Complete Check for Additional Prime Conjunctions Based on the Consensus Law). Check based on the result of Exercise 7.19 whether there are additional prime conjunctions and add them, if found, to the set of prime conjunctions. This check can be restricted such that only the 6 prime conjunctions created in Exercise 7.19 must be taken as first conjunctions in pairs of conjunctions in the consensus law. Practical tasks:

1 Load the final TVL system of Exercise 7.19.

2 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_1.

3 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_2.

4 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_3.

5 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_4.

6 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_5.

7 Execute this task for the prime conjunctions in rows $9 \ldots 14$ of Fig. 7.7 a) and consensus conjunctions with regard to the variable x_6.

8 Show all found prime conjunctions.

Knowing all prime conjunctions it is required to find minimal subsets of prime conjunctions that cover the associated function completely. This search can be simplified be means of the essential prime conjunctions which must be inherent parts of each of these subsets. Using formula (7.79) of [18] on page 299 it can be checked whether a prime conjunction is an essential prime conjunction.

Exercise 7.21 (Essential Prime Conjunctions). Detect all essential prime conjunctions based on the set of all prime conjunctions given in Fig. 7.7 a). Practical tasks:

1 Load the final TVL system of Exercise 7.20.

2 Prepare a PRP that finds all essential prime conjunctions out of the set of all 14 prime conjunctions given as object number 15.

3 Apply the PRP of Task 2 and enumerate the found essential prime conjunctions.

4 Which cubes of the function $f(\mathbf{x})$ given in Fig. 7.4 b) on page 172 are covered by the found essential prime conjunctions only.

Generally there are several minimal disjunctive forms where no prime conjunction can be removed from the expression without changing the function. All minimal disjunctive forms can be found using the cover function $\mathrm{cov}_f(\mathbf{p})$, see [18], page 297. The cover function is created in a conjunctive form which must be transformed into a simplified disjunctive form. The function of Fig. 7.4 b) covers 36 cubes, therefore the cover function consists of 36 disjunctions. Their transformation into disjunctive form is a time-consuming and error-prone task.

The solution of this task can be supported by the XBOOLE monitor. The cover function $\mathrm{cov}_f(\mathbf{p})$ can be created quickly and concisely using a TVL in disjunctive form where 1 and – are the only elements. Using negation according to DE MORGAN (NDM) and a complement (CPL), an orthogonal disjunctive form of $\mathrm{cov}_f(\mathbf{p})$ can be created by XBOOLE. A following OBBC operation minimizes the number of disjunctions. For orthogonality reasons 0 values appear in the result. Such a 0 values means that the associated prime conjunction is not part of the minimal disjunctive form. Because a – has the same meaning in the cover function, all 0 values can be substituted by – values using the change elements operation (CEL). Remaining absorptions can be eliminated by the operation 'delete ternary vector' (DTV).

Exercise 7.22 (All Minimal Disjunctive Forms). Create the sets of all minimal disjunctive forms based on the set of all prime conjunctions given in Fig. 7.7 a) for the function given in Fig. 7.4 b). Evaluate these sets with regard to the number of required prime conjunctions. Enumerate the shortest minimal disjunctive forms. Practical tasks:

1 Create a Boolean space for the cover function $\mathrm{cov}_f(\mathbf{p})$.

2 Create a TVL in K-form having the columns of the variables $p1 \ldots p14$. Add rows to this TVL such that the 1 marks usable prime conjunctions for a function cube and fill the remaining positions with dashes $-$. In order to do this compare systematically the ternary vectors of the function given in Fig. 7.4 b) on page 172 with the prime conjunctions given in Fig. 7.7 a) on page 177 for allowed covers. Equal rows can be removed immediately.

3 Transform the TVL created in K-form into a minimized TVL in D-form as described above. Remove all such rows that can be absorbed by any other row.

4 Each row of the minimized cover function in D-form represents a minimal disjunctive form of the basic function. Evaluate the found sets with regard to the number of required prime conjunctions.

5 Create and show TVLs of the shortest minimal disjunctive forms.

The final design step is the mapping of the found expression to available gates. Generally it is possible to map the minimal disjunctive forms of the function directly to a PLA. Other basic structures are more restrictive. Look-up tables (LUT) inside of configurable logic blocks (CLB) of an FPGA allow typically four inputs. The fastest gates even restrict to two inputs. Applying the distributive and other laws of logic functions, the minimal disjunctive forms of a function can change such that the technological restrictions hold and certain gates are reused several times. In the following we assume that only NOT-gates and AND-, OR- and EXOR-gates of two inputs are available in a multiple level circuit structure.

Exercise 7.23 (Technology Mapping). All minimal disjunctive forms were calculated in Exercise 7.22 based on the function that is given by the circuit of Fig. 7.1. Does this circuit realize a minimal disjunctive form, or is it possible to simplify this circuit? Create a circuit structure for one of the shortest minimal disjunctive forms calculated in Exercise 7.22. Compare both the required numbers of gates and the depths of these circuits. Practical tasks:

1 Compare the circuit of Fig. 7.1 with the set of all prime conjunctions in Fig. 7.7 a) in order to verify whether the circuit realizes one of the minimal disjunctive forms enumerated in Fig. 7.8 b).

2 How the circuit of Fig. 7.1 must change in order to realize one of the minimal disjunctive forms?

3 Apply the distributive law to one of the shortest minimal disjunctive forms calculated in Exercise 7.22 such that it can be realized by using NOT gates, and AND-, OR- and EXOR-gates of two inputs only. Draw the found circuit.

4 Compare the required number of gates and the depths of the three different circuits of the same function used in this exercise.

The results of Exercise 7.23 have shown that the effort for the calculation of an exact minimal disjunctive form leads after technology mapping to a multiple-level circuit to a very small improvement only. Alternatively the decomposition approach can be used. The bi-decomposition can be checked with regard to the OR-, AND- and EXOR-operation for

logic functions that are completely or incompletely specified. In the next exercises the complete design of the same logic function will be executed step by step.

Exercise 7.24 (Strong Elementary Bi-Decomposition of the Completely Specified Function $y = f(\mathbf{x})$). Check for each pair of variables whether a bi-decomposition with regard to the OR-, AND- and EXOR-operation exists for the function of Fig. 7.1. Practical tasks:

1 Load the final TVL system of Exercise 7.9. This TVL system includes the list of phases $F(\mathbf{x}, y)$ as object number 4. Calculate the function $f(\mathbf{x})$. Store this TVL system as `e73dec1.sdt` for later use.

2 Prepare a PRP that checks for each pair of the six variables based on (7.86) in [18] whether an OR-bi-decomposition exists.

3 Prepare a PRP that checks for each pair of the six variables based on (7.90) in [18] whether an AND-bi-decomposition exists.

4 Prepare a PRP that checks for each pair of the six variables based on (7.93) in [18] whether an EXOR-bi-decomposition exists.

5 Are there any OR-bi-decompositions?

6 Are there any AND-bi-decompositions?

7 Are there any EXOR-bi-decompositions?

If more than one pair for the same type of bi-decomposition exists, it is possible that such a pair of variables can be extended to a pair of sets of variables. A complete bi-decomposition is found when no further variables can be added to the decomposition sets of variables without loss of the decomposition property. Complete bi-decompositions simplify result functions of the decomposition most strongly.

Exercise 7.25 (Complete EXOR-Bi-Decomposition). Check whether an extended complete EXOR-bi-decomposition based on the two pairs of variables found in Exercise 7.24 exists. Calculate the decomposition functions of the complete EXOR-bi-decomposition. Practical tasks:

1 Load the TVL system `e73dec1.sdt` of Exercise 7.24.

2 In Exercise 7.24 the allowed pairs of variables (x_1, x_5) and (x_1, x_6) for an EXOR-bi-decomposition were found. Check based on (7.94) of [18] whether there is an EXOR-bi-decomposition of the function in object 7 with regard to $a = x_1$ and $x_b = (x_5, x_6)$.

3 Calculate the decomposition function g_1 based on (7.95) of [18] and h_1 based on (7.97) of [18] and show Karnaugh-maps of these decomposition functions.

4 Verify the calculated EXOR-bi-decomposition.

5 Remove the intermediate results and store the TVL system as `e73dec2.sdt` for later use.

As next the simpler function g_1 of Fig. 7.10 a) on page 182 have to be decomposed. The check for all three types of strong bi-decomposition can be executed similarly to Exercise 7.24. Due to the reduced number of variables the PRP for the check can be simplified.

Exercise 7.26 (Strong Elementary Bi-Decomposition of the Completely Specified Function $g_1(\mathbf{x})$). Check for each pair of variables whether for the function $g_1(\mathbf{x})$ of Fig. 7.10 a) on page 182 exists a bi-decomposition with regard to the OR-, AND- and EXOR-operation. Practical tasks:

1 Load the TVL system `e73dec2.sdt` of Exercise 7.25. This TVL system includes the function $g_1(\mathbf{x})$ as object number 10. Copy for universal use in the PRPs the function $g_1(\mathbf{x})$ as object number 30 and create their complement as object number 31.

2 Prepare a PRP that checks for each pair of the four variables based on (7.86) in [18] whether an OR-bi-decomposition exists.

3 Prepare a PRP that checks for each pair of the four variables based on (7.90) in [18] whether an AND-bi-decomposition exists.

4 Prepare a PRP that checks for each pair of the four variables based on (7.93) in [18] whether an EXOR-bi-decomposition exists.

5 Are there any strong OR-bi-decompositions?

6 Are there any strong AND-bi-decompositions?

7 Are there any strong EXOR-bi-decompositions?

Exercise 7.26 confirms that a bi-decomposition with regard to a selected pair of variables is a property that characterizes the given function. There are logic functions which do not have any strong bi-decompositions. Theorem 7.12 of [18] describes the completeness of the bi-decomposition such that either an EXOR-bi-decomposition or at least one of the weak bi-decompositions with regard to an OR- or an AND-operation exists. If a weak bi-decomposition exists, additional don't-cares extend the function to an incompletely specified function (ISF) or, if given, an

ISF itself. The characteristic function set of an extended ISF includes a set with more functions which can be used in the following decomposition.

Generally it is possible to find and use a weak bi-decomposition with respect of a set of variables. The larger the number of variables in such a set of variables, the larger is typically the depth of the circuit. Due to this observation in the next Exercise weak bi-decompositions with respect to single variables are explored.

Exercise 7.27 (Weak OR-Bi-Decomposition of the Completely Specified Function $g_1(\mathbf{x})$). Check for each variable whether for the function $g_1(\mathbf{x})$ of Fig. 7.10 a) on page 182 exists a weak bi-decomposition with regard to the OR- and AND-operation. If possible, create a ISF of g_2. Practical tasks:

1 Load the TVL system `e73dec2.sdt` of Exercise 7.25. This TVL system includes the function $g_1(\mathbf{x})$ as object number 10.

2 Prepare a PRP that checks for each of the 4 variables based on (7.122) in [18] whether a weak OR-bi-decomposition exists.

3 Prepare a PRP that checks for each of the 4 variables based on (7.127) in [18] whether a weak AND-bi-decomposition exists.

4 Are there any weak OR-bi-decompositions?

5 Are there any weak AND-bi-decompositions?

6 Calculate the mark function g_{2q} as object 30 and g_{2r} as object number 31 using formulas (7.123) and (7.124) in [18] for the existing weak OR-bi-decomposition with regard to x_1. Show the Karnaugh-maps of the calculated mark functions.

7 Remove the intermediate results and store the TVL system as `e73dec3.sdt` for later use.

The extension of the function $g_1(\mathbf{x})$ to an incompletely specified function with the mark functions g_{2q} and g_{2r} as the result of a weak OR-bi-decomposition may allow strong bi-decompositions again. All functions of the function set specified by g_{2q} and g_{2r} can be checked for bi-decompositions using the formulas for incompletely specified functions simultaneously. The comparison of the formulas (7.86) and (7.99) of [18] which allow to check completely or incompletely specified functions for OR-bi-decompositions shows that f is replaced by f_q and $\overline{f}$ by f_r. The same observation will be found comparing the formulas (7.90) and

(7.106) of [18] which allow to check completely or incompletely specified functions for AND-bi-decompositions, respectively. For that reason the PRPs of Exercise 7.26 that check the AND- or OR-bi-decomposition with respect to 4 variables can be reused for an incompletely specified function.

Exercise 7.28 (Elementary Bi-Decomposition of the Incompletely Specified Function $\langle g_{2q}(\mathbf{x}), g_{2r}(\mathbf{x})\rangle$). Check for each pair of variables whether for the incompletely specified function with the mark function $g_{2q}(\mathbf{x})$ and $g_{2r}(\mathbf{x})$ calculated in Task 6 of Exercise 7.27 a bi-decomposition with regard to the OR-, AND- and EXOR-operation exists. Practical tasks:

1 Load the TVL system `e73dec3.sdt` of Exercise 7.27. This TVL system includes the function $g_{2q}(\mathbf{x})$ as object number 30 and $g_{2r}(\mathbf{x})$ as object number 31.

2 Prepare a PRP that checks for each pair of the 4 variables based on (7.112), (7.113), and (7.114) in [18] whether an EXOR-bi-decomposition of the incompletely specified function exists.

3 Are there any strong OR-bi-decompositions? Use the PRP of Task 2 of Exercise 7.26.

4 Are there any strong AND-bi-decompositions? Use the PRP of Task 3 of Exercise 7.26.

5 Are there any strong EXOR-bi-decompositions?

In the case that more than one pair for the same type of bi-decomposition for an incompletely specified function exists, it is possible that such a pair of variables can be extended to a pair of sets of variables. If an elementary bi-decomposition with regard to a pair of variables does not exist, the extension of this pair of variables by other variables does not lead to an allowed bi-decomposition of the same type. This condition helps to restrict the search space for complete bi-decompositions.

Exercise 7.29 (Complete Bi-Decomposition of the Incompletely Specified Function $\langle g_{2q}(\mathbf{x}), g_{2r}(\mathbf{x})\rangle$). Check, based on the allowed pairs of variables found for all types of bi-decompositions in Exercise 7.24, whether extended complete bi-decompositions exist for the incompletely specified function with the mark function $g_{2q}(\mathbf{x})$ and $g_{2r}(\mathbf{x})$. Calculate the decomposition functions of a complete bi-decomposition having the most variables in their decomposition sets. Practical tasks:

1 Load the TVL system `e73dec3.sdt` of Exercise 7.27. This TVL system includes the function $g_{2q}(\mathbf{x})$ as object number 30 and $g_{2r}(\mathbf{x})$ as object number 31.

2 Prepare a PRP that checks whether a one-to-two OR-bi-decomposition of the given ISF based on the known elementary OR-bi-decompositions exist.

3 Prepare a PRP that checks whether a one-to-two AND-bi-decomposition of the given ISF based on the known elementary AND-bi-decompositions exists.

4 Prepare a PRP that checks whether a one-to-two EXOR-bi-decomposition of the given ISF based on the known elementary EXOR-bi-decompositions exists.

5 Are there one-to-two OR-bi-decompositions?

6 Are there one-to-two AND-bi-decompositions?

7 Are there one-to-two EXOR-bi-decompositions?

8 Prepare a PRP that checks whether a two-to-two AND-bi-decomposition of the given ISF based on the known elementary AND-bi-decompositions exists.

9 Are there two-to-two AND-bi-decompositions?

10 Prepare a PRP that checks whether a one-to-three AND-bi-decomposition of the given ISF based on the known elementary AND-bi-decompositions exists.

11 Are there one-to-three AND-bi-decompositions?

12 Load the TVL system `e73dec3.sdt` of Exercise 7.27 in order to remove the intermediate results.

13 Calculate the mark function $g_{3q}(\mathbf{x})$ as object 32 and $g_{3r}(\mathbf{x})$ as object number 33 using the formulas (7.107) and (7.108) in [18] for the existing AND-bi-decomposition with regard to $(x_1, [x_3, x_4])$. Select the function $g_3(x_1, x_2)$ to be realized in the multilevel circuit and store it as object 13. Show the Karnaugh-maps of the calculated mark functions and the selected function.

14 Calculate the mark functions $h_{3q}(\mathbf{x})$ as object 34 and $h_{3r}(\mathbf{x})$ as object number 35 using the formulas (7.109) and (7.110) in [18] for the existing AND-bi-decomposition with regard to $(x_1, [x_3, x_4])$. Show the Karnaugh-maps of the calculated mark functions.

15 Store TVL system as `e73dec4.sdt` for later use.

The characteristic function set of $h_3(\mathbf{x})$ includes four functions. All of them depend on three variables. If any strong bi-decomposition for one of these functions exists, both decomposition functions depend on two variables only. Such functions can be realized by two input gates such that the decomposition terminates. The required PRPs to check for bi-decomposition are similar and simpler than the PRPs created in the previous Exercises. Thus, the change of stored PRPs in an editor can save some time to solve the following Exercise.

Exercise 7.30 (Complete Bi-Decomposition of the Incompletely Specified Function $\langle h_{3q}(\mathbf{x}), h_{3r}(\mathbf{x})\rangle$). Check first for each pair of variables whether for the incompletely specified function with the mark functions $h_{3q}(\mathbf{x})$ and $h_{3r}(\mathbf{x})$ calculated in Task 14 of Exercise 7.29 exists a bi-decomposition with regard to the OR-, AND- and EXOR-operation. Extend the found elementary bi-decompositions to complete bi-decompositions. Select the best existing bi-decomposition and calculate the decomposition functions. Practical tasks:

1 Load the TVL system `e73dec4.sdt` of Exercise 7.29. This TVL system includes the function $h_{3q}(\mathbf{x})$ as object number 34 and $h_{3r}(\mathbf{x})$ as object number 35.

2 Prepare a PRP that checks for each pair of the three variables based on (7.99) in [18] whether an OR-bi-decomposition exists.

3 Prepare a PRP that checks for each pair of the three variables based on (7.106) in [18] whether an AND-bi-decomposition exists.

4 Prepare a PRP that checks for each pair of the three variables based on (7.112), (7.113), and (7.114) in [18] whether an EXOR-bi-decomposition exists.

5 Are there strong one-to-one OR-bi-decompositions?

6 Are there strong one-to-one AND-bi-decompositions?

7 Are there strong one-to-one EXOR-bi-decompositions?

8 Are there strong one-to-two OR-bi-decompositions?

9 Are there strong one-to-two AND-bi-decompositions?

10 Are there strong one-to-two EXOR-bi-decompositions?

11 Load the TVL system `e73dec4.sdt` of Exercise 7.29 in order to remove the intermediate results.

12 Calculate the function $g_4(\mathbf{x})$ of the existing EXOR-bi-decomposition based on (7.115) in [18] as object 15.

13 Calculate the mark function $h_{4q}(\mathbf{x})$ as object 36 and $h_{4r}(\mathbf{x})$ as object number 37 using the formulas (7.116) and (7.117) in [18] for the existing EXOR-bi-decomposition with regard to $(x_2, [x_3, x_4])$. Show the Karnaugh-maps of the calculated mark functions.

14 Select the function $h_4(x_3, x_4)$ to be realized in the multilevel circuit and store it as object 15. Show the Karnaugh-map of the selected function $h_4(x_3, x_4)$.

15 Remove the intermediated TVL and store for later use the TVL system as `e73dec5.sdt`.

The design of a multilevel circuit by means of the bi-decomposition is organized as a recursive procedure. When both decomposition functions of a bi-decomposition are chosen from the allowed sets, the function realized by the decomposition gate can be calculated. In several cases such a g-function is needed to calculate the required incompletely specified h-function. Found gates which are controlled by input variables directly initialize the calculation process of the realized functions.

Exercise 7.31 (Calculate the Realized Functions $h_3(\mathbf{x})$, $g_2(\mathbf{x})$, and the Incompletely Specified Function $\langle h_{2q}(\mathbf{x}), h_{2r}(\mathbf{x})\rangle$ That Must Be Decomposed). Calculate the mark functions $h_{2q}(\mathbf{x}), h_{2r}(\mathbf{x})$ of the weak OR-bi-decomposition selected in Exercise 7.27. This calculation requires first the output function of the EXOR-bi-decomposition selected in Exercise 7.30 and secondly the output function of the AND-bi-decomposition selected in Exercise 7.29. Show the Karnaugh-maps of the realized functions $h_3(\mathbf{x})$, $g_2(\mathbf{x})$ and the mark functions $h_{2q}(\mathbf{x}), h_{2r}(\mathbf{x})$ of the next decomposition task. Practical tasks:

1 Load the TVL system `e73dec5.sdt` of Exercise 7.30. This TVL system includes the function $g_4(\mathbf{x})$ as object number 15, $h_4(\mathbf{x})$ as object number 16, $g_3(\mathbf{x})$ as object number 13, and $g_1(\mathbf{x})$ as object number 10.

2 Calculate $h_3(\mathbf{x})$ as object number 14. This function is realized by an EXOR-gate controlled by $g_4(\mathbf{x})$ and $h_4(\mathbf{x})$. Show the Karnaugh-map of $h_3(\mathbf{x})$.

3 Calculate $g_2(\mathbf{x})$ as object number 12. This function is realized by an AND-gate controlled by $g_3(\mathbf{x})$ and $h_3(\mathbf{x})$. Show the Karnaugh-map of $g_2(\mathbf{x})$.

4 Calculate the mark functions $h_{2q}(\mathbf{x}), h_{2r}(\mathbf{x})$ of the weak OR-bi-decomposition using the formulas (7.125) and (7.126) of [18] as objects 38

and 39. Remember that the weak OR-bi-decomposition was created for the completely specified function $g_1(\mathbf{x})$. Show the Karnaugh-maps of the calculated mark functions.

5 Remove the intermediated TVL and store for later use the TVL system as `e73dec6.sdt`.

The comparison of the mark functions $h_{2q}(\mathbf{x})$ and $h_{2r}(\mathbf{x})$ in Fig. 7.14 a) and b) on page 187 reveals that only the function $h_2(\mathbf{x}) = h_{2q}(\mathbf{x})$ is allowed. Thus the formulas of a completely specified function can be applied for the bi-decomposition, and the known function $h_{2r}(\mathbf{x})$ can be taken as complement of the function to be decomposed.

Exercise 7.32 (Complete Bi-Decomposition of $h_2(\mathbf{x})$). Execute the bi-decomposition of the completely specified $h_2(\mathbf{x}) = h_{2q}(\mathbf{x})$ which depends on the variables (x_2, x_3, x_4) and verify the result. Practical tasks:

1 Load the TVL system `e73dec6.sdt` of Exercise 7.31. This TVL system includes the function $h_{2q}(\mathbf{x})$ as object number 38, $h_{2r}(\mathbf{x}) = \overline{h_{2q}(\mathbf{x})}$ as object number 39, $g_3(\mathbf{x})$ as object number 13, and $g_1(\mathbf{x})$ as object number 10.

2 Prepare a PRP that checks for each pair of the three variables based on (7.86) in [18] whether an OR-bi-decomposition exists.

3 Prepare a PRP that checks for each pair of the three variables based on (7.90) in [18] whether an AND-bi-decomposition exists.

4 Prepare a PRP that checks for each pair of the three variables based on (7.93) in [18] whether an EXOR-bi-decomposition exists.

5 Are there strong one-to-one OR-bi-decompositions?

6 Are there strong one-to-one AND-bi-decompositions?

7 Are there strong one-to-one EXOR-bi-decompositions?

8 Are there strong one-to-two bi-decompositions?

9 Calculate the function g_5 of the existing disjoint AND-bi-decomposition with regard to $(x_2, [x_3, x_4])$ based on (7.91) in [18] as object 18.

10 Calculate the function h_5 of the existing disjoint AND-bi-decomposition with regard to $(x_2, [x_3, x_4])$ based on (7.92) in [18] as object 19.

11 Calculate the function $h_2(x_2, x_3, x_4)$ to be realized in the multilevel circuit and store it as object 17. Show the Karnaugh-map of the selected function $h_2(x_2, x_3, x_4)$.

12 Remove the intermediate TVL and store for later use the TVL system as `e73dec7.sdt`.

In order to design the circuit structure of the function $y = f(\mathbf{x})$ of Fig. 7.4 completely, the decomposition of the function $h_1(\mathbf{x})$ shown in Fig. 7.10 b) on page 182 still remains. A programmed procedure solves this task as executed in the previous exercises by checking for one-to-one bi-decompositions followed by possible extensions of found bi-decompositions. The Karnaugh-map of the function $h_1(\mathbf{x})$ in Fig. 7.10 b) shows that the three function values appear in the line selected by $x_5 = 1$ and $x_6 = 1$. This observation indicates the existence of a disjoint AND-bi-decomposition with regard to $([x_2, x_3, x_4], [x_5, x_6])$ which simplifies the next calculations.

Exercise 7.33 (Complete Bi-Decomposition of $h_1(\mathbf{x})$). Verify whether an AND-bi-decomposition $h_1(\mathbf{x})$ shown in Fig. 7.10 b) with regard to $([x_2, x_3, x_4], [x_5, x_6])$ exists and calculate allowed decomposition functions. xPractical tasks:

1 Load the TVL system `e73dec7.sdt` of Exercise 7.32. This TVL system includes the function $h_1(\mathbf{x})$ as object number 11.

2 Check by means of (7.90) in [18] whether an AND-bi-decomposition of $h_1(\mathbf{x})$ with regard to $([x_2, x_3, x_4], [x_5, x_6])$ exists.

3 Calculate the function g_6 of the existing disjoint AND-bi-decomposition with regard to $([x_2, x_3, x_4], [x_5, x_6])$ based on (7.91) in [18] as object 20.

4 Calculate the function h_6 of the existing disjoint AND-bi-decomposition with regard to $([x_2, x_3, x_4], [x_5, x_6])$ based on (7.92) in [18] as object 21.

5 Verify the calculated decomposition.

6 Remove the intermediated TVLs and store for later use the TVL system as `e73dec8.sdt`.

If a bi-decomposition for the function $h_6(\mathbf{x})$ exists, the whole design procedure terminates.

Exercise 7.34 (Complete Bi-Decomposition of $h_6(\mathbf{x})$). Check first for each pair of variables whether a bi-decomposition exists with regard to the OR-, AND- and EXOR-operation for the function $h_6(\mathbf{x})$. Extend the found elementary bi-decompositions to complete bi-decompositions. Select the best existing bi-decomposition and calculate the decomposition functions. Practical tasks:

1 Load the TVL system `e73dec8.sdt` of Exercise 7.33. This TVL system includes the function $h_6(\mathbf{x})$ as object number 21.

2 Prepare a PRP that checks for each pair of the three variables based on (7.86) in [18] whether an OR-bi-decomposition exists.

3 Prepare a PRP that checks for each pair of the three variables based on (7.90) in [18] whether an AND-bi-decomposition exists.

4 Prepare a PRP that checks for each pair of the three variables based on (7.93) in [18] whether an EXOR-bi-decomposition exists.

5 Are there strong one-to-one OR-bi-decompositions?

6 Are there strong one-to-one AND-bi-decompositions?

7 Are there strong one-to-one EXOR-bi-decompositions?

8 Are there strong one-to-two bi-decompositions?

9 Calculate the function g_7 of the existing AND-bi-decomposition with regard to (x_2, x_3) based on (7.91) in [18] as object 22. Show the Karnaugh-map of the calculated function $g_7(x_2, x_4)$.

10 Calculate the function h_7 of the existing AND-bi-decomposition with regard to (x_2, x_3) based on (7.92) in [18] as object 23. Show the Karnaugh-map of the calculated function $h_7(x_3, x_4)$.

11 Verify the calculated decomposition.

12 Remove the intermediate TVL and store for later use the TVL system as `e73dec9.sdt`.

The function of the circuit of Fig. 7.1 was taken to design a multiple-level circuit based on the bi-decomposition applied in a recursive procedure. The detailed results of the bi-decompositions were calculated in Exercises 7.24 ... 7.34.

Exercise 7.35 (Technology Mapping and Verification). Draw the circuit structure calculated by bi-decompositions in Exercises 7.24 ... 7.34. Verify whether the decomposition structure realizes the given basic function of Fig. 7.1. Compare both the required numbers of gates and the depth of the designed structure with the results of minimal disjunctive forms. Practical tasks:

1 Load the TVL system `e73dec9.sdt` of Exercise 7.34. This TVL system includes the system function $F(\mathbf{x}, y)$ as object number 4.

2 Draw the circuit structure calculated by bi-decompositions in Exercises 7.24 ... 7.34.

3 Verify whether the decomposition structure realizes the given basic function of Fig. 7.1.

4 Extend the comparison of Task 4 in Exercise 7.23 by the values of the design using bi-decomposition. Summarize conclusions from this comparison.

4. Test

For technical reasons the controllability and observability of combinatorial circuits is restricted. Typically it is only possible to control the inputs and to observe the outputs of the circuit. One strategy to test the circuit is to verify whether a stuck-at-0 or a stuck-at-1 error exists on any gate connection. There are several methods to calculate the required test pattern. One of them is the method of the sensible path. The advantage of this method is that all stuck-at errors along the selected sensible path can be excluded if the change of an input value causes the change of the output value.

Exercise 7.36 (Test Pattern Calculated for a Sensible Path). Calculate all test patterns for the selected sensible path $x_2 - g_7 - h_6 - h_1 - y$ in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Practical tasks:

1 Load the TVL system `e73dec9.sdt` of Exercise 7.34. This TVL system includes the output function $y = f(\mathbf{x})$ as object 7, and the controlling functions $g_1(\mathbf{x})$ as object 10, $g_6(\mathbf{x})$ as object 20, and $h_7(\mathbf{x})$ as object 23. Prepare a variable tuple for the independent variable x_2 and a TVL for the function $nx_4 = \overline{x}_4$.

2 Calculate all test patterns for the sensible path $x_2 - g_7 - h_6 - h_1 - y$ using formula (7.144) of [18].

3 Enumerate all stuck-at errors which can be excluded by the calculated test pattern if correct values appear at the circuit output y.

The disadvantage of the method of the sensible path is that in many cases no test pattern can be found, although for each gate connection on the path both stuck-at-0 and stuck-at-1 test patterns exist.

Exercise 7.37 (Restriction of the Sensible Path Method). Calculate all test patterns for the selected sensible path $x_2 - g_4 - h_3 - g_2 - g_1 - y$ in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Practical tasks:

1 Load the TVL system `e73dec9.sdt` of Exercise 7.34. This TVL system includes the output function $y = f(\mathbf{x})$ as object 7, and the controlling

functions $h_1(\mathbf{x})$ as object 11, $h_2(\mathbf{x})$ as object 17, $g_3(\mathbf{x})$ as object 13, and $h_4(\mathbf{x})$ as object 16. Prepare a variable tuple for the independent variable x_2.

2 Calculate all test patterns for the sensible path $x_2 - g_4 - h_3 - g_2 - g_1 - y$ using formula (7.144) of [18].

3 Enumerate all stuck-at errors which can be excluded by the calculated test patterns if correct values appear at the circuit output y.

Each test pattern can be found by the method of the sensible point. First the detailed method is applied in order to calculate all test pattern for the connection h_3 of the circuit in Fig. 7.16. Remember, the method of sensible path does not find any test pattern for this connection.

Exercise 7.38 (Test Pattern Calculated for a Sensible Point Using the Detailed Method). Calculate all test patterns for the sensible point h_3 in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Use (7.146) ... (7.149) of [18] and the equation system of Exercise 7.35, Task 3 as basis. Practical tasks:

1 Define a Boolean space of 32 variables and assign the variables appearing in the circuit of Fig. 7.16.

2 Solve the equation system of Exercise 7.35, Task 3 that describes the required behavior.

3 Modify the equation system of Exercise 7.35, Task 3 in order to get the observing part of the circuit such that h_3 is substituted by s and the controlling equations are removed. Solve this equation system.

4 Prepare an equation system for the controlling part of the circuit. This equation system includes the equation with the variables h_3, g_4 and h_4 on the left-hand side. Solve this equation system.

5 Solve the simple equations $y = 1$, $h_2 = 1$, $t = 1$ and prepare a VT of the internal variables of the circuit. These objects are required to calculate the test pattern.

6 Solve (7.146) of [18] for the sensible point h_3 that describes the error controlling condition.

7 Solve (7.147) of [18] for the sensible point h_3 that describes the error observability condition.

8 Solve (7.148) of [18] for the circuit that allows the error evaluation.

9 Solve (7.149) of [18] for the sensible point h_3. The result of this equation are all existing test patterns for the sensible point h_3. How many test patterns exist?

10 Substantiate why the method of the sensible path does not find any test pattern for h_3.

It is not necessary to handle the controllability and observability separately. Using the list of error phases $F^E(\mathbf{x}, f, s, t)$ it is possible to calculate all existing test patterns of a sensible point in a much simpler way.

Exercise 7.39 (Test Patterns Calculated for a Sensible Point Using the List of Error Phases). Calculate all test patterns for the sensible point h_3 in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Use (7.150) of [18] and the equation system of Exercise 7.35, Task 3 as basis. Practical tasks:

1 Define a Boolean space of 32 variables and assign the variables appearing in the circuit of Fig. 7.16.

2 Solve the equation system of Exercise 7.35, Task 3 that describes the required behavior.

3 Modify the equation system of Exercise 7.35, Task 3 such that the list of error phases can be calculated. In order to do this, substitute the variable h_3 appearing on the left-hand side of an equation by $\overline{t}$ and on the right-hand side by s, respectively. Solve this equation system.

4 Prepare a VT of the internal variables of the circuit.

5 Calculate the function $F^E(\mathbf{x}, f, s, t)$ for the sensible point h_3 that describes the list of error phases.

6 Calculate the function $F^R(\mathbf{x}, f)$ for the sensible point h_3 that describes the list of required phases.

7 Solve (7.150) of [18] for the sensible point h_3. The results of this equation are all existing test patterns for the sensible point h_3. Compare these test patterns with the test patterns calculated in Exercise 7.38.

Assume that the output of a gate controls exactly one input of a following gate. If an error occurs at this connection, it is not possible to distinguish between an error at the gate output and the controlled gate input. In the case that several inputs are controlled by one gate output, different test patterns may exist for the connected pins of the

gates. There is a method that allows the calculation of all existing test patterns for connected pins of a local branch separately.

Exercise 7.40 (Test Pattern Calculated for a Sensible Point of a Local Branch). Calculate all test patterns for the sensible point nx_4 in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Use (7.152) ... (7.156) of [18] and the equation system of Exercise 7.35, Task 3 as basis. Practical tasks:

1 Define a Boolean space of 32 variables and assign the variables appearing in the circuit of Fig. 7.16.

2 Modify the equation system of Exercise 7.35, Task 3 such that the NOT-gate nx_4 is described explicitly, and the controlled inputs of nx_4 are labeled by the model variables s_1 and s_2, respectively. Solve this system of logic equations.

3 Prepare for the sensible point nx_4 both the branch function $F_B(t, s_1, s_2)$ and the function $F_T(nx_4, t)$ that describe the controllability condition.

4 Prepare VTs of the internal variables of the circuit, first of all model variables of the local branch $\langle s_1, s_2, t\rangle$, secondly of each of these variables separately, and finally of $\langle s_1, s_2\rangle$.

5 Calculate the possible behavior for the local branch of nx_4 based on (7.152) of [18].

6 Calculate the required behavior for the local branch of nx_4 based on (7.154) of [18].

7 Calculate all existing test patterns for the signal source of the local branch of nx_4 based on (7.156) of [18].

8 Calculate all existing test patterns for the signal target s_1 on the OR-gate of the local branch of nx_4 based on (7.156) of [18].

9 Calculate all existing test patterns for the signal target s_2 on the EXOR-gate of the local branch of nx_4 based on (7.156) of [18].

10 Calculate all these test patterns which can detect errors on the signal source of the branch nx_4, but not on the signal targets s_1 and s_2, respectively.

11 Calculate all these test patterns which can detect errors on the target s_1 of the branch nx_4, but not on the associated signal source.

12 Calculate all these test patterns which can detect errors on the target s_2 of the branch nx_4, but not on the associated signal source.

13 Verify whether there are test patterns which detect errors on both signal targets of the branch nx_4 but not on the signal source.

The result of the bi-decomposition based on the formulas given in Chap. 7 of [18] is a completely testable circuit. The required test patterns can be calculated as a byproduct during the design using intermediate design results.

Exercise 7.41 (Test Pattern for the Path $x_2 - g_4 - h_3 - g_2 - g_1 - y$ Based on Bi-Decomposition Results). Calculate all test patterns for the path $x_2 - g_4 - h_3 - g_2 - g_1 - y$ in the circuit of Fig. 7.16 on page 190 designed in Section 3. using the bi-decomposition. Use the equations (7.157) and (7.158) of [18] and the intermediate design results stored in `e7dec9.sdt` created in Exercise 7.34 as basis. Practical tasks:

1 Load the TVL system `e73dec9.sdt` of Exercise 7.34. This TVL system includes the solution of $y = f(\mathbf{x})$ as object 4, the function $h_{2r}(\mathbf{x}) = \max^k_{x_a} g_{1r}$ as object 39, and the function $g_3(\mathbf{x})$ as object 13. Prepare the solution of $x_2 \oplus t = 1$.

2 Calculate the requirements for the test pattern on the given path using (7.157) and (7.158) of [18].

3 Extend test patterns calculated in the previous task by information about the expected behavior.

There is a direct relationship between the testability and the realization of a minimal disjunctive form. It was found in Exercise 7.23, Task 2 that the input of x_5 of the gate g_{11} is not part of a prime conjunction. Let us finally calculate all test patterns for this point using the method of the sensible point that finds all existing test patterns.

Exercise 7.42 (Test Pattern for the Sensible Point of a Redundant Variable). Calculate all test patterns for the sensible point x_5 of the gate g_{11} in the circuit of Fig. 7.1 on page 136 and evaluate the result. Use (7.150) and (7.152) ... (7.154) of [18] and the equation system created in Exercise 7.1 as basis. Practical tasks:

1 Modify the equation system created in Exercise 7.1 such that test patterns for the sensible point x_5 of the gate g_{11} can be calculated. Solve the equation system in an appropriate Boolean space.

2 Prepare the solution of $x_5 \oplus t = 1$, the degenerate branch function $F_B(t, s)$ and a VT of the internal variables of the circuit.

3 Calculate the possible behavior for the sensible point x_5 of the gate g_{11} based on (7.152) of [18].

4 Calculate the required behavior for the sensible point x_5 of the gate g_{11} based on (7.154) of [18].

5 Calculate all existing test patterns for the sensible point x_5 of the gate g_{11} based on (7.150) of [18].

6 Evaluate the calculated test patterns with regard to the circuit structure.

5. Solutions

Exercise 7.1.

```
sbe 1 1
nx1=/x1,
nx2=/x2,
nx3=/x3,
nx4=/x4,
nx5=/x5,
nx6=/x6,
g1=x3&nx4,
g2=nx1&g1,
g3=nx2&x4,
g4=nx1&g3,
g5=x5&x6,
g6=nx1&g5,
g7=x2&nx3,
g8=x4&g7,
g9=x1&nx5,
g10=g7&g9,
g11=x1&x5,
g12=g7&g11,
g13=nx6&g12,
g14=x2&g1,
g15=x2&x4,
g16=g5&g15,
g17=nx2&nx3,
g18=nx4&g17,
g19=g5&g18,
g20=g2+g4,
g21=g6+g8,
g22=g10+g13,
g23=g14+g16,
g24=g20+g21,
g25=g22+g23,
g26=g24+g25,
y=g19+g26.
```

Exercise 7.2.

```
tin 1 21
x1 nx1.
01,10.
tin 1 22
x2 nx2.
01,10.
tin 1 23
x3 nx3.
01,10.
tin 1 24
x4 nx4.
01,10.
tin 1 25
x5 nx5.
01,10.
tin 1 26
x6 nx6.
01,10.
tin 1 31
x3 nx4 g1.
111,0-0,100.
tin 1 32
nx1 g1 g2.
111,0-0,100.
tin 1 33
nx2 x4 g3.
111,0-0,100.
tin 1 34
nx1 g3 g4.
111,0-0,100.
tin 1 35
x5 x6 g5.
111,0-0,100.
tin 1 36
nx1 g5 g6.
111,0-0,100.
tin 1 37
x2 nx3 g7.
111,0-0,100.
tin 1 38
x4 g7 g8.
111,0-0,100.
tin 1 39
x1 nx5 g9.
111,0-0,100.
tin 1 40
g7 g9 g10.
111,0-0,100.
tin 1 41
x1 x5 g11.
111,0-0,100.
tin 1 42
g7 g11 g12.
111,0-0,100.
tin 1 43
nx6 g12 g13.
111,0-0,100.
tin 1 44
x2 g1 g14.
111,0-0,100.
tin 1 45
x2 x4 g15.
111,0-0,100.
tin 1 46
g5 g15 g16.
111,0-0,100.
tin 1 47
nx2 nx3 g17.
111,0-0,100.
tin 1 48
nx4 g17 g18.
111,0-0,100.
tin 1 49
g5 g18 g19.
111,0-0,100.
tin 1 50
g2 g4 g20.
000,1-1,011.
tin 1 51
g6 g8 g21.
000,1-1,011.
tin 1 52
g10 g13 g22.
000,1-1,011.
tin 1 53
g14 g16 g23.
000,1-1,011.
tin 1 54
g20 g21 g24.
000,1-1,011.
tin 1 55
g22 g23 g25.
000,1-1,011.
tin 1 56
g24 g25 g26.
000,1-1,011.
tin 1 57
g19 g26 y.
000,1-1,011.
```

a)

«	»	O	K	TVL 2 (ODA) | 8 Var.							
		x1	x2	x3	x4	x5	x6	y1	y2
1:		0	-	-	-	1	0	1	1
2:		-	-	-	1	0	1	1	1
3:		0	1	0	-	1	1	1	0
4:		0	0	-	-	1	1	0	0
5:		0	0	-	-	0	0	0	0
6:		1	-	1	-	1	0	1	0
7:		1	1	1	-	0	0	1	0
8:		1	0	1	-	0	0	0	0
9:		0	1	1	-	0	0	0	0
10:		0	1	0	-	0	0	1	0
11:		1	1	1	-	1	1	1	0
12:		1	0	1	-	1	1	0	0
13:		0	1	1	-	1	1	0	0
14:		-	-	1	0	0	1	1	1
15:		-	-	0	0	0	1	1	0
16:		1	-	0	-	1	1	1	1
17:		1	-	0	-	-	0	1	1

b)

«	»	O	K	TVL 11 (ODA)					
		x1	x2	x3	x4	x5	x6		
1:		0	-	-	-	1	0		
2:		-	-	-	1	0	1		
3:		0	1	0	-	1	1		
4:		1	-	1	-	1	0		
5:		1	1	1	-	0	0		
6:		0	1	0	-	0	0		
7:		1	1	1	-	1	1		
8:		-	-	1	0	0	1		
9:		-	-	0	0	0	1		
10:		1	-	0	-	1	1		
11:		1	-	0	-	-	0		

c)

«	»	O	K	TVL 12 (ODA)					
		x1	x2	x3	x4	x5	x6		
1:		0	-	-	-	1	0		
2:		-	-	-	1	0	1		
3:		-	-	1	0	0	1		
4:		1	-	0	-	1	1		
5:		1	-	0	-	-	0		

Figure 7.2 Behavior of a completely specified circuit: **a**) system function $F(x_1, x_2, x_3, x_4, x_5, x_7, y_1, y_2)$ that is identical with the list of phases as object 2, **b**) function $y_1(\mathbf{x})$ as object 11, **c**) function $y_2(\mathbf{x})$ as object 12

Exercise 7.3.

```
tin 1 1 /d
x1 x2 x3 x4 x5 x6.
0-10--,00-1--,0---11,-101--,110-0-,110-11,-110--,-1-111,-00011.
```

Exercise 7.4.

1 space 32 1
avar 1
x1 x2 x3 x4 x5 x6 y1 y2.

2 sbe 1 1
y1=((x1&(x2#(/x3+x4))+(x5#x6)+x2&/x3&/x4) #(x1&x3&x4&(/x5#x6)))
+x2&/x3&x4,
y2=(/x1&x5&/x6+x1&x2&/x3+x1&/x3 &(/x2+x4)+/x5&x6&(/x1&/x3+x4))
#/x4&/x5&x6.

3 obbc 1 2
Figure 7.2 a) shows the system function $F(x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2)$.

Exercise 7.5.

1 lds e7104.sdt

2 sbe 1 3
y1=1.

sbe 1 4
y2=1.

vtin 1 5
y1 y2.

3 isc 2 3 6
maxk 6 5 11

isc 2 4 8
maxk 8 5 12

4 Figure 7.2 b) and c) show the functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ as TVLs in ODA-form. The functions are equal to 1 for the shown input pattern. Because only dashes

appear in the column x_2 of the TVL 12, the function $y_2(\mathbf{x})$ does not depend on the variable x_2.

Exercise 7.6.

1 space 32 1
avar 1
x1 x2 x3 x4 x5 x6 x7 y.

2 sbe 1 1
((x1&/x2&(x3#x4)&y+x3&/x4&(x5#x6)&/y+/x1&x7&y
+x1&x2&x6&/x7)#/x6)
+(x3#/x4)&x6+x3&/x4&x5+/x1&/x3&x6&/x7+x1&x2&(x3#x4)&x6&x7.

3 sbe 1 2
y=1.

4 mink 1 2 3

5 isc 1 2 4 dif 4 3 4 maxk 4 2 4

6 cpl 2 5 isc 5 1 5 dif 5 3 5 maxk 5 2 5

7 isc 3 4 6 isc 3 5 7 isc 4 5 8
The TVLs 6, 7, and 8 are empty. Thus the mark functions are pairwisely disjoint.
uni 3 4 9 uni 9 5 9 cpl 9 9
The TVL 9 is empty. Hence, the mark functions cover the whole Boolean space.

8 The mark function $f_\varphi(\mathbf{x})$ possesses 98 function values 1. Hence, it can be chosen out of the huge amount of 2^{98} functions in the design process.

9 csd 2 4 10 syd 2 5 11 uni 10 11 12 syd 1 12 13
The TVL 13 is empty. Thus the TVL 12 is an identical system function $F(\mathbf{x}, y)$ reconstructed using the mark functions $f_q(\mathbf{x})$ and $f_r(\mathbf{x})$.

10 syd 2 5 20 uni 20 3 21 syd 1 21 22
The TVL 22 is empty. Thus the TVL 21 is an identical system function $F(\mathbf{x}, y)$ reconstructed using the mark functions $f_\varphi(\mathbf{x})$ and $f_r(\mathbf{x})$.

11 csd 2 4 30 uni 30 3 31 syd 1 31 32
The TVL 32 is empty. Thus the TVL 31 is an identical system function $F(\mathbf{x}, y)$ reconstructed using the mark functions $f_q(\mathbf{x})$ and $f_\varphi(\mathbf{x})$.

Exercise 7.7.

1 $2^6 = 64$ phases exist.

2 space 64 1
avar 1
x1 x2 x3 x4 x5 x6 y nx1 nx2 nx3 nx4 nx5 nx6 g1 g2 g3 g4 g5 g6 g7 g8 g9
g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20 g21 g22 g23 g24 g25 g26.

3 Execute the PRP prepared in Exercise 7.1. Figure 7.3 shows the calculated global list of phases.

Exercise 7.8.

1 lds e7201.sdt

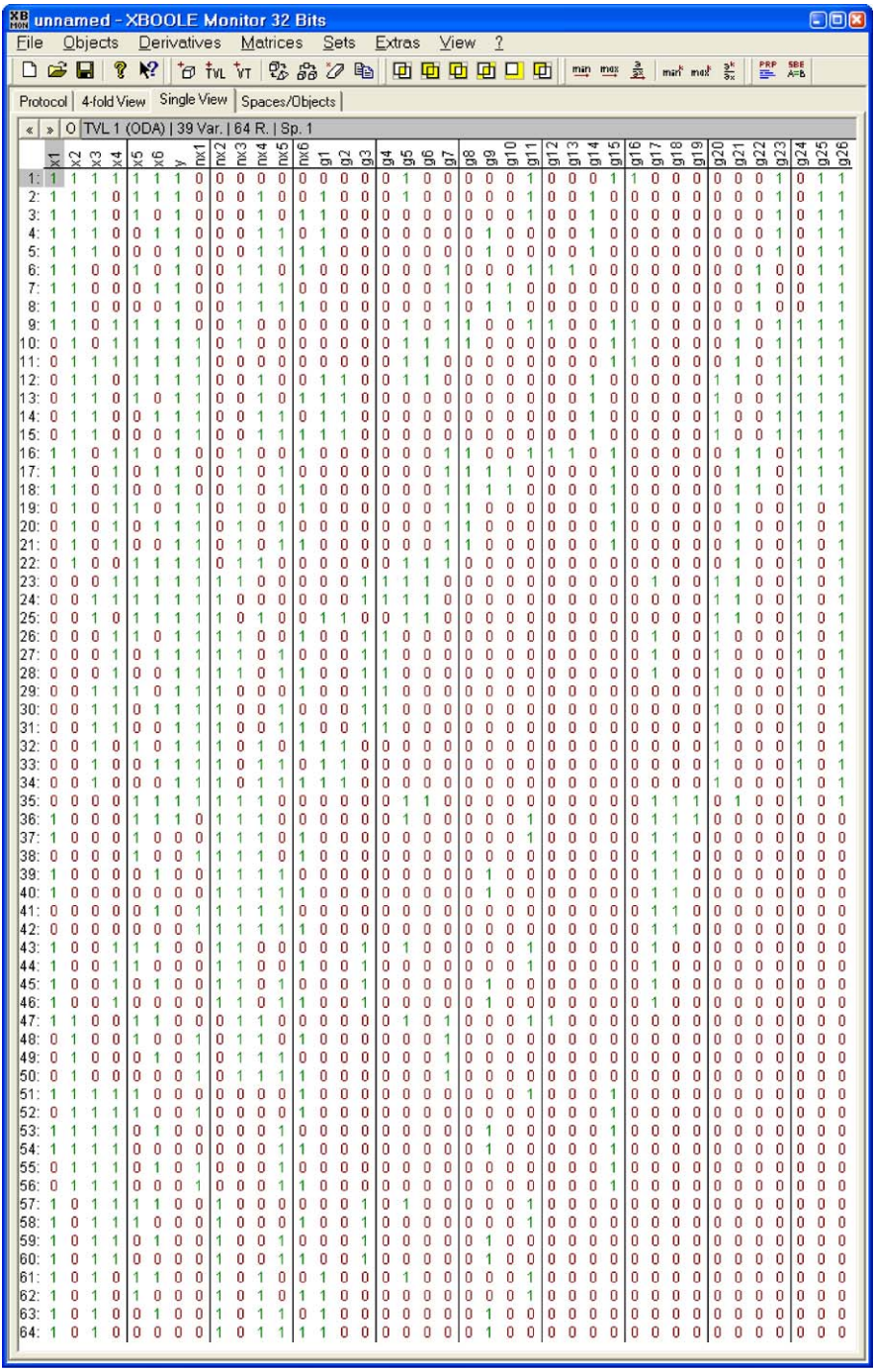


Figure 7.3 Global list of phases that describes the behavior of the combinatorial circuit given in Fig. 7.1

2 Local list of phases in objects 21 . . . 26 for NOT-gates and 31 . . . 57 for the remaining gates.

3
isc 21 22 60	isc 60 34 60	isc 60 42 60	isc 60 50 60
isc 60 23 60	isc 60 35 60	isc 60 43 60	isc 60 51 60
isc 60 24 60	isc 60 36 60	isc 60 44 60	isc 60 52 60
isc 60 25 60	isc 60 37 60	isc 60 45 60	isc 60 53 60
isc 60 26 60	isc 60 38 60	isc 60 46 60	isc 60 54 60
isc 60 31 60	isc 60 39 60	isc 60 47 60	isc 60 55 60
isc 60 32 60	isc 60 40 60	isc 60 48 60	isc 60 56 60
isc 60 33 60	isc 60 41 60	isc 60 49 60	isc 60 57 60

4 The order of phases in the objects 1 and 60 is different. The empty TVL 61 as result of – syd 1 60 61 – confirms that both TVL include the same phases.

Exercise 7.9.

1 The m-fold maximum with regard to all internal variables solves this task. The function $F(\mathbf{x}, y)$ is equal to 1 if there exists at least one pattern ($\mathbf{nx} = const$, $\mathbf{g} = const$) such that $F(\mathbf{x}, y, \mathbf{nx} = const, \mathbf{g} = const)$ is equal to 1.

2 lds e7201.sdt

3 vtin 1 2
nx1 nx2 nx3 nx4 nx5 nx6 g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20 g21 g22 g23 g24 g25 g26.

4 There are 64 rows in the TVL calculated by `maxk` 1 2 3.

5 Figure 7.4 a) shows 64 phases in 19 rows and 7 columns

Exercise 7.10.

1 lds e7201.sdt

2 sbe 1 2
y2=g20, y3=g11, y4=g23.
isc 1 2 3

3 vtin 1 4
x1 x2 x3 x4 x5 x6.
vtin 1 5
nx1 nx2 nx3 nx4 nx5 nx6 g1 g2 g3 g4 g5 g6 g7 g8 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19 g20 g21 g22 g23 g24 g25 g26.
vtin 1 6
y y2 y3 y4.

4
tin 1 10	tin 1 11	maxk 3 5 7	isc 7 11 14
x1 x2 x3 x4 x5 x6.	x1 x2 x3 x4 x5 x6.	isc 7 10 12	maxk 14 4 15
100010.	011010.	maxk 12 4 13	

The output signals are uniquely defined by the input signals.
The simulation patterns are:
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (100010; 0010)$ and
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (011010; 1101)$.

5
tin 1 20	tin 1 21	isc 7 20 22	isc 7 21 24
y y2 y3 y4.	y y2 y3 y4.	maxk 22 6 23	maxk 24 6 25
1010.	0101.		

a)

«	»	O	K	TVL 4 (ODA)					
	x1	x2	x3	x4	x5	x6	y
1:	0	-	1	0	1	0	1
2:	1	1	-	0	0	1	1
3:	0	0	-	0	1	1	1
4:	0	-	1	0	0	-	1
5:	0	1	0	0	1	1	1
6:	1	0	0	0	1	1	1
7:	-	0	0	0	1	0	0
8:	1	1	0	0	1	1	0
9:	0	1	0	0	0	1	0
10:	0	1	0	0	-	0	0
11:	-	1	1	1	1	0	0
12:	1	1	-	0	-	0	1
13:	-	1	1	-	1	1	1
14:	-	1	0	1	-	-	1
15:	0	0	-	1	-	-	1
16:	-	0	0	0	0	-	0
17:	-	1	1	1	0	-	0
18:	1	0	-	1	-	-	0
19:	1	0	1	0	-	-	0

b)

«	»	O	K	TVL 11 (ODA)					
	x1	x2	x3	x4	x5	x6
1:	0	-	1	0	1	0
2:	1	1	-	0	0	1
3:	0	0	-	0	1	1
4:	0	-	1	0	0	-
5:	0	1	0	0	1	1
6:	1	0	0	0	1	1
7:	1	1	-	0	-	0
8:	-	1	1	-	1	1
9:	-	1	0	1	-	-
10:	0	0	-	1	-	-

c)

«	»	O	K	TVL 15 (D)					
	x1	x2	x3	x4	x5	x6
1:	0	-	1	0	-	-
2:	1	1	-	0	0	-
3:	0	-	-	-	1	1
4:	-	0	0	0	1	1
5:	1	1	-	0	-	0
6:	-	1	1	-	1	1
7:	-	1	0	1	-	-
8:	0	0	-	1	-	-

Figure 7.4 Behavioral descriptions of the combinatorial circuit given in Fig. 7.1: **a**) List of phases of the input-output-behavior, **b**) logic function $f(\mathbf{x})$, and **c**) prime conjunctions

The input signals are not uniquely defined by the output signals. There are three simulation patterns for the first given output pattern:
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (110010; 1010)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (110110; 1010)$, and
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (100011; 1010)$,
but there is no input pattern that creates the second given output pattern.

6
```
tin 1 30                tin 1 31                isc 3 30 32             isc 3 31 34
g1 g5 g7.               g1 g5 g7.               maxk 32 5 33            maxk 34 5 35
010.                    101.
```

The input and output signals are not uniquely defined by the internal signals. There are 8 input-output patterns for $(g_1, g_5, y_7) = (010)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (111111; 1011)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (011111; 1001)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (000111; 1100)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (001111; 1100)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (000011; 1000)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (100011; 1010)$,
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (100111; 0010)$, and
$(x_1, x_2, x_3, x_4, x_5, x_6; y, y_2, y_3, y_4) = (101111; 0010)$,
but no such pattern for $(g_1, g_5, y_7) = (101)$.

Exercise 7.11.

1 lds e7204.sdt

2 $2^4 = 16$ output patterns can exist for a circuit with four outputs.

3 maxk 7 4 40
Eight patterns can be seen at the outputs of the given circuit.

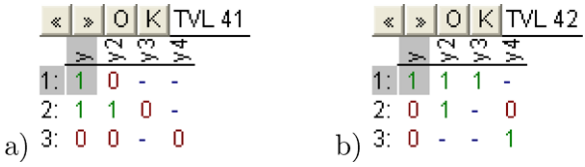


Figure 7.5 Output behavior of the combinatorial circuit given in Fig. 7.1 and modified such that additional outputs are defined by $y_2 = g_{20}, y_3 = g_{11}, y_4 = g_{23}$: **a**) all output patterns of the circuit as object 41, **b**) don't-care function $f_\varphi(y, y_2, y_3, y_4)$ for a successor circuit as object 42

obbc 40 41
Figure 7.5 a) shows the output patterns (y, y_2, y_3, y_4) of the given circuit.

4 cpl 41 42
Figure 7.5 b) shows the don't-care function $f_\varphi(y, y_2, y_3, y_4)$.

Exercise 7.12.

1 lds e7105.sdt

2 _derk 20 <x1> 21
_derk 20 <x2> 22
_derk 20 <x3> 23
_derk 20 <x4> 24
_derk 20 <x5> 25
_derk 20 <x6> 26

3 _copy 11 20 — Execute the PRP of Task 2. TVL 24 is empty. Hence, the function $y_1(\mathbf{x})$ does not depend on x_4.

4 _copy 12 20 — Execute the PRP of Task 2. TVL 22 is empty. Hence, the function $y_2(\mathbf{x})$ does not depend on x_2.

5 _maxk 11 <x4> 13
syd 11 13 14 — The TVL 14 is empty. Hence, object 13 is an allowed short representation of the function $y_1(\mathbf{x})$.

6 _maxk 12 <x2> 15
syd 12 15 16 — The TVL 16 is empty. Hence, object 15 is an allowed short representation of the function $y_2(\mathbf{x})$.

Exercise 7.13.

1 lds e7105.sdt

2 maxk 1 5 20
cpl 20 21 — The TVL 21 is empty. Thus both functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ can be realized as combinatorial circuit.

3 lds e7106.sdt

4 maxk 1 2 40
cpl 40 41 — TVL 41 is empty. Thus this system function includes at least one realizable function $y(\mathbf{x})$.

Exercise 7.14.

1 lds e7105.sdt

2 _maxk 1 <y2> 20
_mink 20 <y1> 21
_maxk 1 <y1> 22
_mink 22 <y2> 23 — The TVLs 21 and 23 are empty. Thus both functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$ are uniquely defined.

3 lds e7106.sdt

4 mink 1 2 40 — TVL 40 is not empty. Thus a function $y(\mathbf{x})$ can be chosen out of a lattice of functions.

Exercise 7.15.

1 lds e7106.sdt

2
```
_mink 1 <x1> 41        _mink 1 <x3> 43        _mink 1 <x5> 45        _mink 1 <x7> 47
maxk 41 2 41           maxk 43 2 43           maxk 45 2 45           maxk 47 2 47
cpl 41 41              cpl 43 43              cpl 45 45              cpl 47 47
_mink 1 <x2> 42        _maxk 1 <x4> 44        _mink 1 <x6> 46
maxk 42 2 42           maxk 44 2 44           maxk 46 2 46
cpl 42 42              cpl 44 44              cpl 46 46
```

TVL 46 is not empty. Hence all functions described by $F(\mathbf{x}, y)$ depend on the variable x_6. The TVLs 41 ... 45 and 47 are empty. Consequently the system function $F(\mathbf{x}, y)$ includes functions which do not depend on at least one of the variables $(x_1, x_2, x_3, x_4, x_5, x_7)$.

3

$$\max_{y}[\min_{\mathbf{x}_1}{}^{m} F(\mathbf{x}_1, \mathbf{x}_2, y)] = 1. \tag{7.6}$$

Function values 1 of $F(\mathbf{x}_1, \mathbf{x}_2, y)$ for all patterns of $\mathbf{x}_1$ fix either the associated value of y or allow as part of a don't-care description the free choice of the y value. Thus, if formula (7.6) holds, there must be at least one function covered by $F(\mathbf{x}_1, \mathbf{x}_2, y)$ that does not depend on the variables $\mathbf{x}_1$.

4
```
_mink 1 <x1 x2 x3 x4 x5 x7> 50        maxk 50 2 50        cpl 50 50
```
TVL 50 is not empty. Thus there is no function described by $F(\mathbf{x}_1, \mathbf{x}_2, y)$ that depends on x_6 only.

5
```
_mink 1 <x1 x2 x3 x4 x5> 51            _mink 1 <x1 x2 x4 x5 x7> 54
maxk 51 2 51                           maxk 54 2 54
cpl 51 51                              cpl 54 54
_mink 1 <x1 x2 x3 x4 x7> 52            _mink 1 <x1 x3 x4 x5 x7> 55
maxk 52 2 52                           maxk 55 2 55
cpl 52 52                              cpl 55 55
_mink 1 <x1 x2 x3 x5 x7> 53            _mink 1 <x2 x3 x4 x5 x7> 56
maxk 53 2 53                           maxk 56 2 56
cpl 53 53                              cpl 56 56
```

The TVLs 53 and 54 are empty. Hence, the system function $F(\mathbf{x}_1, \mathbf{x}_2, y)$ includes simple functions that depend on either (x_4, x_6) or (x_3, x_6) only.

6

$$F(\mathbf{x}_2, y) = \min_{\mathbf{x}_1}{}^{m} F(\mathbf{x}_1, \mathbf{x}_2, y) \tag{7.7}$$

Function values 1 must appear for all patterns of $\mathbf{x}_1$ in the system function $F(\mathbf{x}_1, \mathbf{x}_2, y)$ in order to restrict the system function $F(\mathbf{x}_2, y)$ such that it includes only functions not depending on the variables $\mathbf{x}_1$.

7
```
_mink 1 <x1 x2 x3 x5 x7> 60        _mink 1 <x1 x2 x4 x5 x7> 61
```
The completely specified functions $f_1(x_4, x_6) = x_4 x_6$ and $f_2(x_3, x_6) = \overline{x}_3 x_6$ are the simplest functions covered by the system function $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, y)$ each depending on two variables only. Figure 7.6 shows the associated lists of phases.

Exercise 7.16.

1 lds e7203.sdt

a)

« » O K TVL 60

	x4	x6	y
1:	0	1	0
2:	0	0	0
3:	1	1	1
4:	1	0	0

b)

« » O K TVL 61

	x3	x6	y
1:	1	1	0
2:	1	0	0
3:	0	1	1
4:	0	0	0

Figure 7.6 Simplest system functions of the characteristic function set represented by $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7, y)$ (7.2) on page 139: **a)** $F(x_4, x_6, y)$ as object 60, **b)** $F(x_3, x_6, y)$ as object 61

2	tin 1 5 /oda	y. 1.
3	maxk 4 5 6 cpl 6 7	The empty TVL 7 confirms that at least one combinatorial circuit exists.
4	derk 4 5 8 cpl 8 9	The empty TVL 9 confirms that exactly one combinatorial circuit exists.
5	isc 4 5 10 maxk 10 5 11	Figure 7.4 b) on page 172 shows the logic function of the combinatorial circuit given in Fig. 7.1.

Exercise 7.17.

1	lds e7301.sdt		
2	_maxk 12 <x1> 20 dif 20 11 21 _maxk 12 <x2> 20 dif 20 11 22	_maxk 12 <x3> 20 dif 20 11 23 _maxk 12 <x4> 20 dif 20 11 24	_maxk 12 <x5> 20 dif 20 11 25 _maxk 12 <x6> 20 dif 20 11 26
3	stv 11 1 12 execute PRP of task 2 removable var.: x5, x6	_maxk 12 <x5> 12 execute PRP of task 2	_maxk 12 <x6> 15 pc: $\overline{x}_1 x_3 \overline{x}_4$
4	stv 11 2 12 execute PRP of task 2 removable var.: x6	_maxk 12 <x6> 12 con 15 12 15 pc: $x_1 x_2 \overline{x}_4 \overline{x}_5$	
5	stv 11 3 12 execute PRP of task 2 removable var.: x2, x4	_maxk 12 <x2> 12 execute PRP of task 2	_maxk 12 <x4> 12 con 15 12 15 pc: $\overline{x}_1 x_5 x_6$
6	stv 11 4 12 execute PRP of task 2 removable var.: x5	_maxk 12 <x5> 12 pc: $\overline{x}_1 x_3 \overline{x}_4$ repeated found	
7	stv 11 5 12 execute PRP of task 2 removable var.: x2, x3, x4	_maxk 12 <x2> 12 execute PRP of task 2	_maxk 12 <x3> 12 execute PRP of task 2 pc: $\overline{x}_1 x_5 x_6$ repeated found
8	stv 11 6 12 execute PRP of task 2 removable var.: x1	_maxk 12 <x1> 12 con 15 12 15 pc: $\overline{x}_2 \overline{x}_3 \overline{x}_4 x_5 x_6$	
9	stv 11 7 12 execute PRP of task 2	removable var.: not any con 15 12 15	pc: $x_1 x_2 \overline{x}_4 \overline{x}_6$

10	stv 11 8 12 execute PRP of task 2	removable var.: not any con 15 12 15	pc: $x_2x_3x_5x_6$
11	stv 11 9 12 execute PRP of task 2	removable var.: not any con 15 12 15	pc: $x_2\overline{x}_3x_4$
12	stv 11 10 12 execute PRP of task 2	removable var.: not any con 15 12 15	pc: $\overline{x}_1\overline{x}_2x_4$

13 The found eight prime conjunctions are shown in Figure 7.4 c) on page 172.

Exercise 7.18.

1 All conjunctions of the function of Fig. 7.4 b) were extended or directly used.

2 lds e7302.sdt

3 orth 15 27 The empty TVL 28 confirms the complete cover.
syd 27 11 28

4	dtv 15 1 29 dif 11 29 31	dtv 15 3 29 dif 11 29 33	dtv 15 5 29 dif 11 29 35	dtv 15 7 29 dif 11 29 37
	dtv 15 2 29 dif 11 29 32	dtv 15 4 29 dif 11 29 34	dtv 15 6 29 dif 11 29 36	dtv 15 7 29 dif 11 29 38

5 Not any of the TVLs 31 . . . 38 is empty after the execution of the PRP. Hence, the eight conjunctions of Fig. 7.4 c) on page 172 describe a minimal disjunctive form.

Exercise 7.19.

1 lds e7302.sdt

2	_dco 15 <x1> 30 consensus: p1-p2 stv 30 1 12 stv 30 2 31	isc 12 31 12 execute PRP of Ex. 7.17 task 2 _maxk 12 <x5> 12	con 15 12 15 new p9: $x_2x_3\overline{x}_4$ consensus: p1-p5 new pc is equal to p9
3	_dco 15 <x2> 30 consensus: p6-p8	absorbed by p3 consensus: p7-p8	new p10: $\overline{x}_1\overline{x}_3x_4$
4	_dco 15 <x3> 30 consensus: p1-p4	absorbed by p3 consensus: p6-p7	new p11: $x_2x_4x_5x_6$
5	_dco 15 <x4> 30 consensus: p1-p8 new p12: $\overline{x}_1\overline{x}_2x_3$	consensus: p2-p7 new p13: $x_1x_2\overline{x}_3\overline{x}_5$ consensus: p4-p8	absorbed by p3 consensus: p5-p7 new p14: $x_1x_2\overline{x}_3\overline{x}_6$
6	_dco 15 <x5> 30	consensus: p2-p6	absorbed by p9
7	_dco 15 <x6> 30	consensus: p5-p6	absorbed by p9

8 The found 6 prime conjunctions are added to the given eight prime conjunctions. The six new prime conjunctions are shown in rows 9 . . . 14 of Fig. 7.7 a).

Exercise 7.20.

1 lds e7304.sdt

2	_dco 15 <x1> 30 consensus: p10-p13 stv 30 10 12 stv 30 13 31	isc 12 31 12 absorbed by p7 consensus: p10-p14	stv 30 10 12 stv 30 14 31 isc 12 31 12 absorbed by p7

a)

«	»	0	K	TVL 15 (D)			
	x1	x2	x3	x4	x5	x6
1:	0	-	1	0	-	-
2:	1	1	-	0	0	-
3:	0	-	-	-	1	1
4:	-	0	0	0	1	1
5:	1	1	-	0	-	0
6:	-	1	1	-	1	1
7:	-	1	0	1	-	-
8:	0	0	-	1	-	-
9:	-	1	1	0	-	-
10:	0	-	0	1	-	-
11:	-	1	-	1	1	1
12:	0	0	1	-	-	-
13:	1	1	0	-	0	-
14:	1	1	0	-	-	0

b)

«	»	0	K	TVL 15 (D)			
	x1	x2	x3	x4	x5	x6
1:	0	-	1	0	-	-
2:	1	1	-	0	0	-
3:	0	-	-	-	1	1
4:	-	0	0	0	1	1
5:	1	1	-	0	-	0
6:	-	1	1	-	1	1
7:	-	1	0	1	-	-
8:	0	0	-	1	-	-

c)

«	»	0	K	TVL 60 (D)			
	x1	x2	x3	x4	x5	x6
1:	0	-	-	-	1	1
2:	-	0	0	0	1	1
3:	-	1	1	0	-	-
4:	0	-	0	1	-	-
5:	-	1	-	1	1	1
6:	0	0	1	-	-	-
7:	1	1	0	-	0	-
8:	1	1	0	-	-	0

Figure 7.7 All prime conjunctions and minimal disjunctive forms of the combinatorial circuit given in Fig. 7.1: **a**) all prime conjunctions, **b**) first minimal disjunctive form having eight conjunctions, and **c**) second minimal disjunctive form having eight conjunctions

3	_dco 15 <x2> 30	equal to p1	absorbed by p3
	consensus: p9-p12	consensus: p11-p12	
4	_dco 15 <x3> 30	consensus: p9-p14	consensus: p10-p12
	consensus: p9-p13	equal to p5	equal to p8
	equal to p2	consensus: p10-p6	consensus: p12-p4
		absorbed by p3	absorbed by p3
5	_dco 15 <x4> 30	consensus: p10-p4	consensus: p11-p1
	consensus: p9-p11	absorbed by p3	absorbed by p3
	equal to p6		
6	_dco 15 <x5> 30	consensus: p11-p13	absorbed by p7
7	_dco 15 <x6> 30	consensus: p11-p14	absorbed by p7

8 There are no additional prime conjunctions. Hence, Fig. 7.7 a) shows the set of all prime conjunctions of the function shown in Fig. 7.4 b) on page 172.

Exercise 7.21.

1 lds e7305.sdt

2	dtv 15 1 32	dif 11 32 44	dtv 15 8 32	dif 11 32 51
	dif 11 32 41	dtv 15 5 32	dif 11 32 48	dtv 15 12 32
	dtv 15 2 32	dif 11 32 45	dtv 15 9 32	dif 11 32 52
	dif 11 32 42	dtv 15 6 32	dif 11 32 49	dtv 15 13 32
	dtv 15 3 32	dif 11 32 46	dtv 15 10 32	dif 11 32 53
	dif 11 32 43	dtv 15 7 32	dif 11 32 50	dtv 15 14 32
	dtv 15 4 32	dif 11 32 47	dtv 15 11 32	dif 11 32 54

3 There are only two TVLs in the set of solution TVLs 41 . . . 54 which are not empty. Associated to these TVLs 43 and 44 are the prime conjunctions p3 (row 3) and p4 (row 4) which are essential prime conjunctions.

4 The TVLs 43 and 44 show the cubes covered by essential prime conjunctions only. The cube `010011` is covered by p3 only. The cube `100011` is covered by p4 only.

Exercise 7.22.

1 space 32 1

2 The rows in Fig. 7.8 a) are associated with the rows in Fig. 7.4 b) as follows: 1: 1(0), 2: 1(1), 3: 2(0), 4: 2(1), 5: 3(0), 6: 3(1), 7: 4(0-), 8: 4(1-), 9: 5(), 10: 6(), 11: 7(00), 12: 7(01), 13: 7(10), 14: 7(11), 15: 8(00), 16: 8(01), 17: 8(10), 18: 8(11), 19: 9(00-,010), 20: 9(011), 21: 9(100), 22: 9(101), 23: 9(110), 24: 9(111), 25: 10(00-,010), 26: 10(011), 27: 10(10-,110), 28: 10(111), where the elements in an expression [a: b(c)] have the following meaning: a – row number in this TVL, b – row in the function, and c – values assigned to dashes in the function.

3 ndm 1 2 cpl 2 3 obbc 3 4 cel 4 4 5 /0 – /11 / – –
row 22 can be absorbed by row 29: dtv 5 22 6
Fig. 7.8 b) shows the cover function in disjunctive form.

4 Figure 7.8 b) shows that there are 28 different minimal disjunctive forms for the analyzed function. 24 of these forms require 9 prime conjunctions. Two minimal disjunctive forms, see rows 5 and 25, require even 11 prime conjunctions. There are two minimal disjunctive forms indicated in rows 7 and 28 that require 8 prime conjunctions only. The columns of p3 and p4 of the TVL shown in Fig. 7.8 b) include values 1 only, because the associated conjunctions are essential prime conjunctions.

5 Fig. 7.7 b) and c) show the shortest minimal disjunctive forms. The expression of Fig. 7.7 b) was already found by the simple procedure of Exercise 7.17.

_copy 15 60	dtv 60 5 60	dtv 60 5 60	dtv 60 1 60
dtv 60 5 60	dtv 60 5 60	dtv 60 1 60	

The second shortest minimal disjunctive form of Fig. 7.7 c) includes the essential prime conjunctions and all six prime conjunctions found additionally using the consensus law in Exercise 7.19.

Exercise 7.23.

1 The AND-gates that control the tree of OR-gates in Fig. 7.1 directly create the conjunctions of the realized disjunctive form. The following prime conjunctions of Fig. 7.7 a) are realized by AND-gates of Fig. 7.1: g2 $\Leftrightarrow$ p1, g4 $\Leftrightarrow$ p8, g6 $\Leftrightarrow$ p3, g8 $\Leftrightarrow$ p7, g10 $\Leftrightarrow$ p13, g14 $\Leftrightarrow$ p9, g16 $\Leftrightarrow$ p11, and g18 $\Leftrightarrow$ p4. The AND-gate g13 realizes the conjunction $x_1x_2\overline{x}_3x_5, \overline{x}_6$ that includes (in comparison to the prime conjunction p14) the variable x_5 additionally. Figure 7.8 b) shows in row 10 that there is a minimal disjunctive form of p1, p3, p4, p7, p8, p9, p11, p13, and p14. Because of the additional variable x_5 in the conjunction realized by g13 the circuit of Fig. 7.1 does not realize a minimal disjunctive form.

2 The prime conjunction p14 can be realized by three gates: $g_7 = x_2\overline{x}_3$ as given and again reused, $g_{11} = x_1\overline{x}_6$ changed for one input, and $g_{12} = g_7g_{11}$ as given. The gate g13 can be removed which requires the change $g_{22} = g_{10} \vee g_{12}$. This changed circuit realizes the minimal disjunctive form of row 10 in Fig. 7.8 b), requires one gate less than the basic circuit of Fig. 7.1 and reduces the depth of the circuit by one level.

a)

« » O TVL 1 (K) | 14 Var. | 28 R. | Sp. 1

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14
1:	1	-	-	-	-	-	-	-	-	-	1	-	-	
2:	1	-	-	-	-	-	-	-	1	-	-	-	-	-
3:	-	1	-	-	-	-	-	-	-	-	-	-	1	-
4:	-	1	-	-	-	-	-	-	1	-	-	-	-	-
5:	-	-	1	1	-	-	-	-	-	-	-	-	-	-
6:	1	-	1	-	-	-	-	-	-	-	-	1	-	-
7:	1	-	-	-	-	-	-	-	-	-	-	1	-	-
8:	1	-	-	-	-	-	-	-	1	-	-	-	-	-
9:	-	-	1	-	-	-	-	-	-	-	-	-	-	-
10:	-	-	-	1	-	-	-	-	-	-	-	-	-	-
11:	-	1	-	-	1	-	-	-	-	-	-	-	1	1
12:	-	-	-	-	1	-	-	-	-	-	-	-	-	1
13:	-	1	-	-	1	-	-	-	1	-	-	-	-	-
14:	-	-	-	-	1	-	-	-	1	-	-	-	-	-
15:	1	-	1	-	-	1	-	-	1	-	-	-	-	-
16:	-	-	1	-	-	1	-	-	-	-	1	-	-	-
17:	-	-	-	-	-	1	-	-	1	-	-	-	-	-
18:	-	-	-	-	-	1	-	-	-	-	1	-	-	-
19:	-	-	-	-	-	-	1	-	-	1	-	-	-	-
20:	-	-	1	-	-	-	1	-	-	1	1	-	-	-
21:	-	-	-	-	-	-	1	-	-	-	-	-	1	1
22:	-	-	-	-	-	-	1	-	-	-	-	-	1	-
23:	-	-	-	-	-	-	1	-	-	-	-	-	-	1
24:	-	-	-	-	-	-	1	-	-	-	1	-	-	-
25:	-	-	-	-	-	-	-	1	-	1	-	-	-	-
26:	-	-	1	-	-	-	-	1	-	1	-	-	-	-
27:	-	-	-	-	-	-	-	1	-	-	-	1	-	-
28:	-	-	1	-	-	-	-	1	-	-	-	1	-	-

b)

« » O TVL 6 (D) | 14 Var. | 28 R. | Sp. 1

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14
1:	-	-	1	1	1	1	1	1	1	-	-	1	1	-
2:	-	1	1	1	1	1	1	1	1	-	-	1	-	-
3:	-	-	1	1	-	1	1	1	1	-	-	1	1	1
4:	-	1	1	1	-	1	1	1	1	-	-	1	-	1
5:	1	1	1	1	1	1	-	1	-	1	1	-	1	1
6:	-	1	1	1	-	-	1	-	1	1	1	1	-	1
7:	-	-	1	1	-	-	-	-	1	1	1	1	1	1
8:	-	-	1	1	1	-	1	1	1	-	1	1	1	-
9:	-	1	1	1	1	-	1	1	1	-	1	1	-	-
10:	1	-	1	1	-	-	1	1	1	-	1	-	1	1
11:	-	-	1	1	-	-	1	1	1	-	1	1	1	1
12:	-	1	1	1	-	-	1	1	1	-	1	1	-	1
13:	1	1	1	1	-	-	1	1	1	-	1	-	-	1
14:	1	1	1	1	1	-	1	1	1	-	1	-	-	-
15:	1	-	1	1	1	1	1	1	1	-	-	-	1	-
16:	-	-	1	1	1	1	1	-	1	1	-	1	1	-
17:	-	1	1	1	1	1	1	-	1	1	-	1	-	-
18:	1	1	1	1	1	1	1	-	-	1	-	1	-	-
19:	1	-	1	1	-	1	1	1	1	-	-	-	1	1
20:	-	-	1	1	-	1	1	-	1	1	-	1	1	1
21:	-	1	1	1	-	1	1	-	1	1	-	1	-	1
22:	-	-	1	1	1	-	1	-	1	1	1	1	1	-
23:	-	1	1	1	1	-	1	-	1	1	1	1	-	-
24:	1	-	1	1	-	-	-	1	1	1	1	-	1	1
25:	1	1	1	1	1	1	-	-	-	1	1	1	1	1
26:	1	1	1	1	-	1	1	1	1	-	-	-	-	1
27:	1	-	1	1	1	-	1	1	1	-	1	-	1	-
28:	1	1	1	1	1	1	1	1	-	-	-	-	-	-

Figure 7.8 Cover function $\mathrm{cov}_f(\mathbf{p})$ based on the set of all prime conjunctions given in Fig. 7.7 a): **a**) conjunctive form, and **b**) minimized disjunctive form which indicates all minimal disjunctive forms

3 Using the distributive and associative laws for the shortest minimal disjunctive forms of Fig. 7.7 b) such that only operations for two variables occur leads to formula (7.8),

$$\begin{aligned} f(\mathbf{x}) = {} & (\overline{x}_1 x_3)\overline{x}_4 \\ & \vee (x_1 x_2) \wedge [\overline{x}_4 \overline{x}_5 \vee \overline{x}_4 \overline{x}_6] \\ & \vee (x_5 x_6) \wedge [\overline{x}_1 \vee ((\overline{x}_2 \overline{x}_3)\overline{x}_4) \vee x_2 x_3] \\ & \vee (x_2 \overline{x}_3) x_4 \\ & \vee (\overline{x}_1 \overline{x}_2) x_4 . \end{aligned} \tag{7.8}$$

The OR-operation can be realized by a tree of OR-gates. An EXOR-gate does not simplify the circuit structure. The created circuit structure is shown in Fig. 7.9.

4 Table 7.1 shows the requested values. It follows that the number of gates could be reduced by two gates only and the circuits of both explored minimal disjunctive forms need seven instead of eight levels.

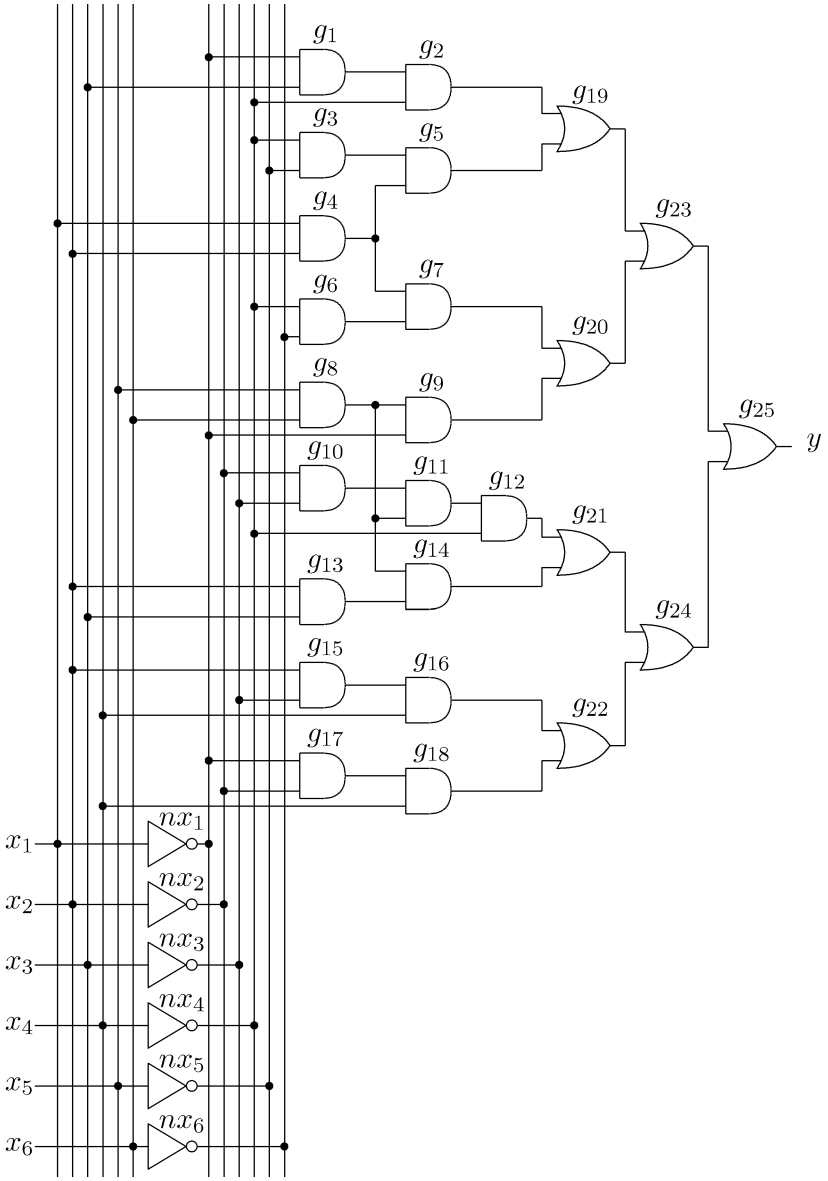


Figure 7.9 Structure of a circuit using AND- and OR-gates restricted to two inputs that realizes the shortest minimal disjunctive form of Fig. 7.7 b)

Table 7.1 Gates and levels of different circuits for the same function

Circuit	# NOT	# AND	# OR	all gates	levels
Figure 7.1	6	19	8	33	8
Exercise 7.23 Task 2	6	18	8	32	7
Figure 7.9	6	18	7	31	7

Exercise 7.24.

1
```
lds e7203.sdt
sbe 1 5
y=1.
isc 4 5 6
maxk 6 5 7
sts e73dec1.sdt
```

2
```
cpl 7 20
_maxk 20 <x1> 21
_maxk 20 <x2> 22
isc 7 21 23
isc 23 22 40
_maxk 7 <x3> 22
isc 23 22 41
_maxk 20 <x4> 22
isc 23 22 42
_maxk 20 <x5> 22
isc 23 22 43
```
```
_maxk 20 <x6> 22
isc 23 22 44
_maxk 20 <x2> 21
_maxk 20 <x3> 22
isc 7 21 23
isc 23 22 45
_maxk 20 <x4> 22
isc 23 22 46
_maxk 20 <x5> 22
isc 23 22 47
```
```
_maxk 20 <x6> 22
isc 23 22 48
_maxk 20 <x3> 21
_maxk 20 <x4> 22
isc 7 21 23
isc 23 22 49
_maxk 20 <x5> 22
isc 23 22 50
_maxk 20 <x6> 22
isc 23 22 51
```
```
_maxk 20 <x4> 21
_maxk 20 <x5> 22
isc 7 21 23
isc 23 22 52
_maxk 20 <x6> 22
isc 23 22 53
_maxk 20 <x5> 21
_maxk 20 <x6> 22
isc 7 21 23
isc 23 22 54
```

3
```
cpl 7 20
_maxk 7 <x1> 21
_maxk 7 <x2> 22
isc 20 21 23
isc 23 22 60
_maxk 7 <x3> 22
isc 23 22 61
_maxk 7 <x4> 22
isc 23 22 62
_maxk 7 <x5> 22
isc 23 22 63
```
```
_maxk 7 <x6> 22
isc 23 22 64
_maxk 7 <x2> 21
_maxk 7 <x3> 22
isc 20 21 23
isc 23 22 65
_maxk 7 <x4> 22
isc 23 22 66
_maxk 7 <x5> 22
isc 23 22 67
```
```
_maxk 7 <x6> 22
isc 23 22 68
_maxk 7 <x3> 21
_maxk 7 <x4> 22
isc 20 21 23
isc 23 22 69
_maxk 7 <x5> 22
isc 23 22 70
_maxk 7 <x6> 22
isc 23 22 71
```
```
_maxk 7 <x4> 21
_maxk 7 <x5> 22
isc 20 21 23
isc 23 22 72
_maxk 7 <x6> 22
isc 23 22 73
_maxk 7 <x5> 21
_maxk 7 <x6> 22
isc 20 21 23
isc 23 22 74
```

4
```
_derk 7 <x1 x2> 80
_derk 7 <x1 x3> 81
_derk 7 <x1 x4> 82
_derk 7 <x1 x5> 83
_derk 7 <x1 x6> 84
```
```
_derk 7 <x2 x3> 85
_derk 7 <x2 x4> 86
_derk 7 <x2 x5> 87
_derk 7 <x2 x6> 88
_derk 7 <x3 x4> 89
```
```
_derk 7 <x3 x5> 90
_derk 7 <x3 x6> 91
_derk 7 <x4 x5> 92
_derk 7 <x4 x6> 93
_derk 7 <x5 x6> 94
```

5 There is no empty object 40 . . . 54. Hence, there is no OR-bi-decomposition for the function f.

6 There is no empty object 60 . . . 74. Hence, there is no AND-bi-decomposition for the function f.

7 There are the empty objects 83 and 84. Hence, there are EXOR-bi-decompositions for the function f for the pairs of variables (x_1, x_5) and (x_1, x_6).

a)

«	»	O	T	TVL 10 (ODA) | 4 Var.			
0	0	0	0	1	0		
0	1	1	1	1	0		
1	1	1	0	0	0		
1	0	1	1	1	0		
		0	1	1	0	x2	
x3	x4	0	0	1	1	x1	

b)

«	»	O	T	TVL 11 (ODA) | 5 Var. |							
0	0	0	0	0	0	0	0	0	0		
0	1	0	0	0	0	0	0	0	0		
1	1	1	0	0	0	0	1	0	1		
1	0	0	0	0	0	0	0	0	0		
x5	x6	0	1	1	0	0	1	1	0	x4	
		0	0	1	1	1	1	0	0	x3	
		0	0	0	0	1	1	1	1	x2	

Figure 7.10 Results of the EXOR-bi-decomposition of the function in Fig. 7.4 b): **a**) $g_1(x_1, x_2, x_3, x_4)$ as TVL 10, and **b**) $h_1(x_2, x_3, x_4, x_5, x_6)$ as TVL 11

Exercise 7.25.

1 lds e73dec1.sdt

2 _derk 7 <x1> 30 _maxk 30 <x5 x6> 31 _mink 30 <x5 x6> 32 syd 31 32 33
The empty TVL 33 confirms that an EXOR-bi-decomposition with regard to $(x_1, [x_5, x_6])$ exists.

3 tin 1 40 /oda
x5 x6.
00.
isc 7 40 41
maxk 41 40 10
syd 7 10 42
_maxk 42 <x1> 11
Figure 7.10 shows both decomposition functions g_1 (object 10) and h_1 (object 11).

4 syd 10 11 43 syd 43 7 44
The empty TVL 44 confirms the correctness of the decomposition.

5 del 30 del 31 del 32 del 33 del 40 del 41 del 42 del 43 del 44 sts e73dec2.sdt

Exercise 7.26.

1 lds e73dec2.sdt _copy 10 30 cpl 10 31

2
_maxk 31 <x1> 40
isc 30 40 41
_maxk 31 <x2> 42
isc 41 42 43
_maxk 31 <x3> 42
isc 41 42 44

_maxk 31 <x4> 42
isc 41 42 45
_maxk 31 <x2> 40
isc 30 40 41
_maxk 31 <x3> 42
isc 41 42 46

_maxk 31 <x4> 42
isc 41 42 47
_maxk 31 <x3> 40
isc 30 40 41
_maxk 31 <x4> 42
isc 41 42 48

3
_maxk 30 <x1> 50
isc 31 50 51
_maxk 30 <x2> 52
isc 51 52 53
_maxk 30 <x3> 52
isc 51 52 54

_maxk 30 <x4> 52
isc 51 52 55
_maxk 30 <x2> 50
isc 31 50 51
_maxk 30 <x3> 52
isc 51 52 56

_maxk 30 <x4> 52
isc 51 52 57
_maxk 30 <x3> 50
isc 31 50 51
_maxk 30 <x4> 52
isc 51 52 58

4
_derk 30 <x1 x2> 60
_derk 30 <x1 x3> 61

_derk 30 <x1 x4> 62
_derk 30 <x2 x3> 63

_derk 30 <x2 x4> 64
_derk 30 <x3 x4> 65

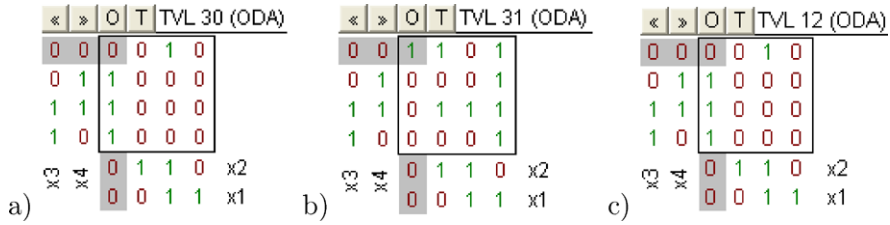


Figure 7.11 Results of the weak OR-bi-decomposition of the function in Fig. 7.10 a): **a**) $g_{2q}(x_1, x_2, x_3, x_4)$ as object 30, **b**) $g_{2r}(x_1, x_2, x_3, x_4)$ as object 31, and **c**) the selected function $g_2(x_1, x_2, x_3, x_4)$ as object 12

5 The TVLs 43 ... 48 are not empty. Hence, there is no strong OR-bi-decomposition for the function g_1.

6 The TVLs 53 ... 58 are not empty. Hence, there is no strong AND-bi-decomposition for the function g_1.

7 The TVLs 60 ... 65 are not empty. Hence, there is no strong EXOR-bi-decomposition for the function g_1.

Exercise 7.27.

1 lds e73dec2.sdt

2
```
cpl 10 70
_maxk 70 <x1> 71
dif 10 71 72
```
```
_maxk 70 <x2> 71
dif 10 71 73
_maxk 70 <x3> 71
```
```
dif 10 71 74
_maxk 70 <x4> 71
dif 10 71 75
```

3
```
cpl 10 80
_maxk 10 <x1> 81
dif 80 81 82
```
```
_maxk 10 <x2> 81
dif 80 81 83
_maxk 10 <x3> 81
```
```
dif 80 81 84
_maxk 10 <x4> 81
dif 80 81 85
```

4 The TVLs 72 ... 75 are not empty. Hence, the function g_1 is weakly OR-bi-decomposable with regard to each of the four variables.

5 The TVLs 82 ... 85 are not empty. Hence, the function g_1 is weakly AND-bi-decomposable with regard to each of the four variables.

6
```
cpl 10 31
```
```
_maxk 31 <x1> 30
```
```
isc 10 30 30
```

Figure 7.11 shows the mark functions g_{2q} and g_{2r}.

7 Delete the TVLs 70 ... 85 using the `del` command. sts e73dec3.sdt

Exercise 7.28.

1 lds e73dec3.sdt

2
```
_maxk 30 <x1> 60
_maxk 31 <x1> 61
isc 60 61 62
_mink 30 <x1> 60
_mink 31 <x1> 61
uni 60 61 63
_maxk 62 <x2> 64
isc 64 63 64
_maxk 62 <x3> 65
isc 65 63 65
```
```
_maxk 62 <x4> 66
isc 66 63 66
_maxk 30 <x2> 60
_maxk 31 <x2> 61
isc 60 61 62
_mink 30 <x2> 60
_mink 31 <x2> 61
uni 60 61 63
_maxk 62 <x3> 67
isc 67 63 67
```
```
_maxk 62 <x4> 68
isc 68 63 68
_maxk 30 <x3> 60
_maxk 31 <x3> 61
isc 60 61 62
_mink 30 <x3> 60
_mink 31 <x3> 61
uni 60 61 63
_maxk 62 <x4> 69
isc 69 63 69
```

3 There are the empty objects 46, 47 and 48. Hence, there are OR-bi-decompositions for the pairs of variables (x_2, x_3), (x_2, x_4), and (x_3, x_4).

4 There are the empty objects 53 ... 57. Hence, there are OR-bi-decompositions for all pairs of variables except the pair (x_3, x_4).

5 There are the empty objects 67 and 68. Hence, there are EXOR-bi-decompositions for the pairs of variables (x_2, x_3), and (x_2, x_4).

Exercise 7.29.

1 lds e73dec3.sdt

2
```
_maxk 31 <x2> 80
isc 30 80 81
_maxk 31 <x3 x4> 82
isc 81 82 83
```
```
_maxk 31 <x3> 80
isc 30 80 81
_maxk 31 <x2 x4> 82
isc 81 82 84
```
```
_maxk 31 <x4> 80
isc 30 80 81
_maxk 31 <x2 x3> 82
isc 81 82 85
```

3
```
_maxk 30 <x1> 90
isc 31 90 91
_maxk 30 <x2 x3> 92
isc 91 92 93
_maxk 30 <x2 x4> 92
isc 91 92 94
_maxk 30 <x3 x4> 92
isc 91 92 95
```
```
_maxk 30 <x2> 90
isc 31 90 91
_maxk 30 <x1 x3> 92
isc 91 92 96
_maxk 30 <x1 x4> 92
isc 91 92 97
_maxk 30 <x3 x4> 92
isc 91 92 98
```
```
_maxk 30 <x3> 90
isc 31 90 91
_maxk 30 <x1 x2> 92
isc 91 92 99
_maxk 30 <x4> 90
isc 31 90 91
_maxk 30 <x1 x2> 92
isc 91 92 100
```

4
```
_maxk 30 <x2> 110
_maxk 31 <x2> 111
isc 110 111 112
```
```
_mink 30 <x2> 110
_mink 31 <x2> 111
uni 110 111 113
```
```
_maxk 112 <x3 x4> 114
isc 114 113 114
```

5 There is no empty object 83 ... 85. Hence, there is no one-to-two OR-bi-decomposition for the ISF of g_2.

6 The empty TVLs 95 and 98 confirm that AND-bi-decompositions for the ISF of g_2 with regard to $(x_1, [x_3, x_4])$ and $(x_2, [x_3, x_4])$ exist.

7 The TVL 114 is not empty. Hence, there is no one-to-two EXOR-bi-decomposition for the ISF of g_2.

8
```
_maxk 30 <x1 x2> 90
isc 31 90 91
```
```
_maxk 30 <x3 x4> 92
isc 91 92 101
```

9 The TVL 101 is not empty. Hence, there is no two-to-two AND-bi-decomposition for the ISF of g_2.

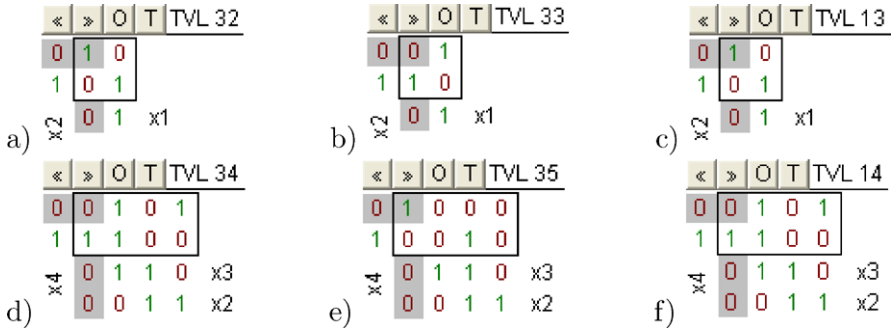


Figure 7.12 Results of the AND-bi-decomposition of the ISF in Fig. 7.11 a) and b): **a)** $g_{3q}(x_1, x_2)$ as object 32, **b)** $g_{3r}(x_1, x_2)$ as object 33, **c)** the selected function $g_3(x_1, x_2)$ as object 13, **d)** $h_{3q}(x_2, x_3, x_4)$ as object 34, **e)** $h_{3r}(x_2, x_3, x_4)$ as object 35, and **f)** the selected function $h_3(x_2, x_3, x_4)$ as object 14

10 _maxk 30 <x1> 90
isc 31 90 91
_maxk 30 <x2 x3 x4> 92
isc 91 92 102

11 The TVL 102 is not empty. Hence, there is no one-to-three OR-bi-decomposition for the ISF of g_2.

12 lds e73dec3.sdt

13 _maxk 30 <x3 x4> 32
_maxk 30 <x1> 33
isc 33 31 33
_maxk 33 <x3 x4> 33

Figure 7.12 a) and b) shows the mark functions g_{3q} and g_{3r}. The calculated ISF includes only the single function $g_3(x_1, x_2) = \overline{x}_1 \oplus x_2$ that can be realized by an EXOR-gate. Hence, `_copy 32 13`. Figure 7.12 c) shows the selected functions g_3.

14 _maxk 30 <x1> 34 isc 13 31 35 _maxk 35 <x1> 35
Figure 7.12 d) and e) show the mark functions h_{3q} and h_{3r}.

15 sts e73dec4.sdt

Exercise 7.30.

1 lds e73dec4.sdt

2 _maxk 35 <x2> 40
isc 34 40 41
_maxk 35 <x3> 42
isc 41 42 43

_maxk 35 <x4> 42
isc 41 42 44
_maxk 35 <x3> 40
isc 34 40 41

_maxk 35 <x4> 42
isc 41 42 45

3 _maxk 34 <x2> 50
isc 35 50 51
_maxk 34 <x3> 52
isc 51 52 53

_maxk 34 <x4> 52
isc 51 52 54
_maxk 34 <x3> 50
isc 35 50 51

_maxk 34 <x4> 52
isc 51 52 55

4
```
_maxk 34 <x2> 60
_maxk 35 <x2> 61
isc 60 61 62
_mink 34 <x2> 60
_mink 35 <x2> 61
uni 60 61 63
```
```
_maxk 62 <x3> 64
isc 64 63 64
_maxk 62 <x4> 65
isc 65 63 65
_maxk 34 <x3> 60
_maxk 35 <x3> 61
```
```
isc 60 61 62
_mink 34 <x3> 60
_mink 35 <x3> 61
uni 60 61 63
_maxk 62 <x4> 66
isc 66 63 66
```

5 The TVLs 43, 44 and 45 are empty. Hence, all possible one-to-one OR-bi-decompositions for the ISF of h_3 exist.

6 The TVLs 53, 54 and 55 are not empty. Hence, no AND-bi-decomposition for the ISF of h_3 exists.

7 The TVLs 64 and 65 are empty. Hence, EXOR-bi-decompositions for the ISF of h_3 with regard to (x_2, x_3) and (x_2, x_4) exist.

8
```
_maxk 35 <x2> 80
isc 34 80 81
_maxk 35 <x3 x4> 82
isc 81 82 83
```
```
_maxk 35 <x3> 80
isc 34 80 81
_maxk 35 <x2 x4> 82
isc 82 82 84
```
```
_maxk 35 <x4> 80
isc 34 80 81
_maxk 35 <x2 x3> 82
isc 81 82 85
```

The TVLs 83, 84 and 85 are not empty. Hence, there are no one-to-two OR-bi-decompositions for the ISF of h_3.

9 Because no one-to-one AND-bi-decomposition for the ISF of h_3 exists, no one-to-two AND-bi-decomposition can exist.

10
```
_maxk 34 <x2> 90
_maxk 35 <x2> 91
isc 90 91 92
```
```
_mink 34 <x2> 90
_mink 35 <x2> 91
uni 90 91 93
```
```
_maxk 92 <x3 x4> 94
isc 94 93 94
```

The empty TVL 94 confirms that the EXOR-bi-decompositions for the ISF of h_3 with regard to $(x_2, [x_3, x_4])$ exist.

11 lds e73dec4.sdt

12
```
_maxk 34 <x2> 95
_maxk 35 <x2> 96
isc 95 96 97
```
```
tin 1 98
x2.
1.
```
```
_maxk 97 <x3 x4> 99
isc 98 99 15
```

The decomposition function is equal to $g_4(x_2) = x_2$ stored as object 15.

13
```
cpl 15 100
isc 100 34 101
isc 15 35 102
```
```
uni 101 102 103
_maxk 103 <x2> 36
isc 100 35 104
```
```
isc 15 34 105
uni 104 105 106
_maxk 106 <x2> 37
```

Figure 7.13 a) and b) show the mark functions h_{4q} and h_{4r}.

14 The calculated ISF includes only the single function $h_4(x_3, x_4) = x_3 \vee x_4$ that can be realized by an OR-gate. Hence, `_copy 36 16` stores the function $h_{4_q}(x_3, x_4)$ into the function $h_4(x_3, x_4)$ as object 16. Figure 7.13 c) shows the selected functions h_4.

15 Delete TVLs 95,..., 106 using the `del` command.
sts e73dec5.sdt

Exercise 7.31.

1 lds e73dec5.sdt

2 syd 15 16 14 Figure 7.12 f) on page 185 shows $h_3(\mathbf{x})$.

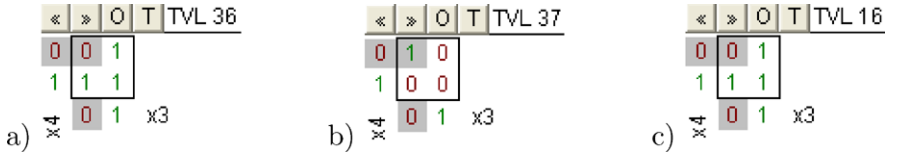


Figure 7.13 Results of the EXOR-bi-decomposition of the incompletely specified function h_3 with the mark functions of Fig. 7.12 d) and e): **a**) $h_{4q}(x_3, x_4)$ as object 30, **b**) $h_{4r}(x_3, x_4)$as object 31, and **c**) the selected function $h_4(x_3, x_4)$

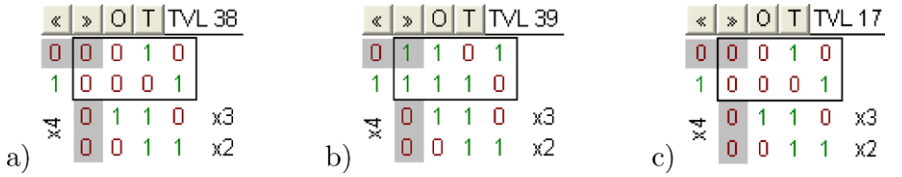


Figure 7.14 Functions on the branch h_2 of the weak OR-bi-decomposition of the function g_1 of Fig. 7.10 a): **a**) $h_{2q}(x_2, x_3, x_4)$ as object 38, **b**) $h_{2r}(x_2, x_4)$as object 39, and **c**) the realized function $h_2(x_2, x_3, x_4)$ as object 17

3 isc 13 14 12 Figure 7.11 c) on page 183 shows $g_2(\mathbf{x})$.

4 cpl 12 40 _maxk 41 <x1> 38 _maxk 42 <x1> 39
isc 40 10 41 cpl 10 42
Figure 7.14 a) and b) show the mark functions $h_{2q}(\mathbf{x})$ and $h_{2r}(\mathbf{x})$.

5 del 40 del 41 del 42 sts e73dec6.sdt

Exercise 7.32.

1 lds e73dec6.sdt

2 _maxk 39 <x2> 50 _maxk 39 <x4> 52 _maxk 39 <x4> 52
isc 38 50 51 isc 51 52 54 isc 51 52 55
_maxk 39 <x3> 52 _maxk 39 <x3> 50
isc 51 52 53 isc 38 50 51

3 _maxk 38 <x2> 60 _maxk 38 <x4> 62 _maxk 38 <x4> 62
isc 39 60 61 isc 61 62 64 isc 61 62 65
_maxk 38 <x3> 62 _maxk 38 <x3> 60
isc 61 62 63 isc 39 60 61

4 _derk 38 <x2 x3> 70 _derk 38 <x2 x4> 71 _derk 38 <x3 x4> 73

5 The TVLs 53, 54 and 55 are not empty. Hence, no OR-bi-decomposition for the function h_2 exist.

6 There are the empty objects 63 and 64. Hence, there are AND-bi-decompositions for the function h_2 with regard to the pairs of variables (x_2, x_3), and (x_2, x_4).

7 There is only the empty object 72. Hence, an EXOR-bi-decomposition for the function h_2 with regard to the pair of variables (x_2, x_4) exists.

8 As result of the analyzed one-to-one bi-decompositions only a one-to-two AND-bi-decomposition can exist.

Figure 7.15 OR-bi-decomposition of the function $h_6(\mathbf{x})$: **a**) the function $h_6(x_2, x_3, x_4)$ as object 21, **b**) the decomposition function $g_7(x_2, x_4)$ as object 22, and **c**) the decomposition function $h_7(x_3, x_4)$ as object 23

_maxk 38 <x2> 60
isc 39 60 61
_maxk 38 <x3 x4> 62
isc 61 62 66

The empty object 66 confirms that the disjoint AND-bi-decomposition of the function h_2 with regard to $(x_2, [x_3, x_4])$ exists.

9 _maxk 38 <x3 x4> 18
The decomposition function is equal to $g_5(x_2) = x_2$.

10 _maxk 38 <x2> 19
The decomposition function is equal to $h_5(x_3, x_4) = x_3 \oplus x_4$.

11 isc 18 19 17
Figure 7.14 c) shows the realized function h_2 as object 17 that is equal to the function h_{2q} of Fig. 7.14 a) which confirms the correct execution of the decomposition.

12 Delete TVLs 50 ... 72 using the `del` command.
sts e73dec7.sdt

Exercise 7.33.

1 lds e73dec7.sdt

2 _maxk 11 <x2 x3 x4> 50
_maxk 11 <x5 x6> 51
cpl 11 52
isc 52 50 53
isc 53 51 53

The empty object 53 confirms that the disjoint AND-bi-decomposition of the function h_1 with regard to $([x_2, x_3, x_4], [x_5, x_6])$ exists.

3 _maxk 11 <x2 x3 x4> 20
The decomposition function is $g_6(x_5, x_6) = x_5 \wedge x_6$. This function can be realized by an AND-gate of two inputs directly such that no further decomposition is required.

4 _maxk 11 <x5 x6> 21
The decomposition function $h_6(x_2, x_3, x_4)$ depends on three variables such that a further decomposition is necessary. Figure 7.15 a) shows the function $h_6(x_2, x_3, x_4)$.

5 isc 20 21 54
syd 54 11 55

The empty object 55 confirms that the specified AND-gate realizes the correct function.

6 Delete TVLs 50,..., 55 using the `del` command.
sts e73dec8.sdt

Exercise 7.34.

1 lds e73dec8.sdt

2
cpl 21 50	isc 52 53 54	isc 21 51 52
_maxk 50 <x2> 51	_maxk 50 <x4> 53	_maxk 50 <x4> 53
isc 21 51 52	isc 52 53 55	isc 52 53 56
_maxk 50 <x3> 53	_maxk 50 <x3> 51	

3
cpl 21 60	isc 62 63 64	isc 60 61 62
_maxk 21 <x2> 61	_maxk 21 <x4> 63	_maxk 21 <x4> 63
isc 60 61 62	isc 62 63 65	isc 62 63 66
_maxk 21 <x3> 63	_maxk 21 <x3> 61	

4 _derk 21 <x2 x3> 70 _derk 21 <x2 x4> 71 _derk 21 <x3 x4> 72

5 The objects 54, 55, and 56 are not empty. Hence, no OR-bi-decomposition for the function h_6 exists.

6 There are the empty object 64 and 65. Hence, there are AND-bi-decompositions for the function h_6 with regard to the pairs of variables (x_2, x_3), and (x_2, x_4).

7 The objects 70, 71 and 72 are not empty. Hence, no EXOR-bi-decomposition for the function h_6 exists.

8 As a result of the analyzed one-to-one bi-decompositions, only a one-to-two AND-bi-decomposition can exist.
_maxk 21 <x2> 61 isc 60 61 62 _maxk 21 <x3 x4> 63 isc 62 63 67
The object 67 is not empty. Hence, the OR-bi-decomposition for the function h_6 with regard to $(x_2, [x_3, x_4])$ does not exist.

9 _maxk 21 <x3> 22
The decomposition function is $g_7(x_2, x_4) = x_2 \vee \overline{x}_4$. This function can be realized by an OR-gate of two inputs directly such that no further decomposition is required.

10 _maxk 21 <x2> 23
The decomposition function is $h_7(x_3, x_4) = x_3 \oplus \overline{x}_4$. This function can be realized by an EXOR-gate of two inputs directly such that no further decomposition is required. Alternatively the complement of the function h_5 can be reused: $h_7(x_3, x_4) = \overline{(x_3 \oplus x_4)} = \overline{h_5(x_3, x_4)}$ which requires a NOT-gate instead of an EXOR-gate.

11
isc 22 23 80	The empty object 81 confirms that the specified
syd 21 80 81	AND-gate realizes the right function.

12 Delete TVLs 50 ... 81 using the `del` command.
sts e73dec9.sdt

Exercise 7.35.

1 lds e73dec9.sdt

2 Figure 7.16 shows the circuit calculated by bi-decompositions in Exercises 7.24, ..., 7.34. The labels on the connection lines correspond to the names of the designed functions.

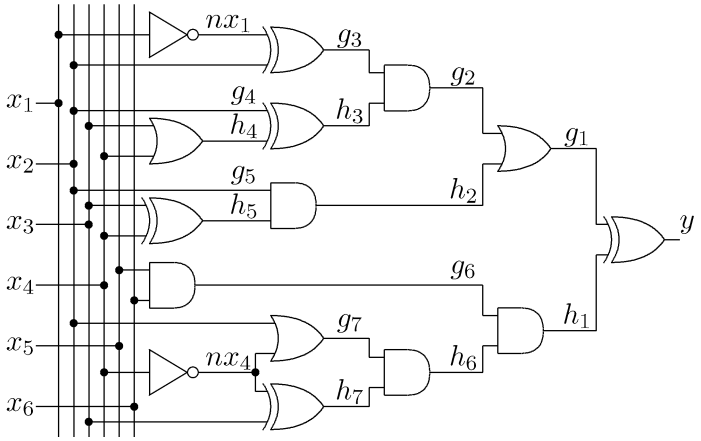


Figure 7.16 Structure of a circuit using AND-, OR- and EXOR-gates restricted to two inputs for the function of Fig. 7.1 designed by bi-decomposition

Table 7.2 Gates and levels of different circuits for the same function

Circuit	# NOT	# AND	# OR	EXOR	all gates	levels
Figure 7.1	6	19	8	0	33	8
Exercise 7.23 Task 2	6	18	8	0	32	8
Figure 7.9	6	18	7	0	31	7
Figure 7.16	2	5	3	5	15	5

3 sbe 1 50
y=g1#h1, g1=g2+h2, g2=g3&h3,
g3=/x1#x2, g4=x2, h4=x3+x4,
h2=g5&h5, g5=x2, h5=x3#x4, h1=g6&h6,
g6=x5&x6, h6=g7&h7, g7=x2+/x4, h7=x3#/x4.

vtin 1 51
g1 g2 g3 g4 g5 g6 g7 h1 h2 h3 h4 h5 h6 h7.
maxk 50 51 52
syd 52 4 53

The empty object 53 confirms that the circuit designed by bi-decomposition realizes the given function.

4 Table 7.2 shows the requested values. It follows that the number of gates in the decomposed circuit is less than half of the shortest minimal disjunctive form. The number of levels is reduced by 2 which speeds up the circuit.

Exercise 7.36.

1 lds e73dec9.sdt
vtin 1 50
x2.
tin 1 51
x4.
0.

2	derk 51 50 52 cpl 52 53 cpl 51 54 isc 54 53 55	derk 23 50 56 cpl 56 57 isc 57 23 58	derk 20 50 59 cpl 59 60 isc 60 20 61	derk 10 50 62 cpl 62 63 derk 7 50 64	isc 64 55 65 isc 65 58 66 isc 66 61 67 isc 67 63 68 isc 68 64 69

The variable x_2 must change for the selected sensible path. There is only one test pattern of the remaining variables $(x_1, x_3, x_4, x_5, x_6) = (11111)$ calculated as object 69.

3 This path is not affected by complement operations. Hence, the pattern $(x_1, x_2, x_3, x_4, x_5, x_6) = (101111)$ detects stuck-at-1 errors at the x_2-input of g_7 and on the connections g_7, h_6, h_1, and y, respectively, and the pattern $(x_1, x_2, x_3, x_4, x_5, x_6) = (111111)$ detects stuck-at-0 errors at the same gate connections.

Exercise 7.37.

1	lds e73dec9.sdt		vtin 1 50 x2.	
2	derk 16 50 51 cpl 51 52 derk 13 50 53 cpl 53 54	isc 54 13 55 derk 17 50 56 cpl 56 57 cpl 17 58	isc 57 58 59 derk 11 50 60 cpl 60 61 derk 7 50 62	isc 62 52 63 isc 63 55 64 isc 64 59 65 isc 65 61 66

The object 66 is empty. Thus, no test pattern could be found for the selected path using the method of the sensible path.

3 Because no test pattern was found using the method of the sensible path, other methods must be utilized in order to calculate the required test pattern.

Exercise 7.38.

1 space 32 1
avar 1
x1 x2 x3 x4 x5 x6 y g1 g2 g3 g4 g5 g6 g7 h1 h2 h3 h4 h5 h6 h7.

2	sbe 1 1 y=g1#h1, g1=g2+h2, g2=g3&h3,	g3=/x1#x2, h3=g4#h4, g4=x2, h4=x3+x4,	h2=g5&h5, g5=x2, h5=x3#x4, h1=g6&h6,	g6=x5&x6, h6=g7&h7, g7=x2+/x4, h7=x3#/x4.
3	sbe 1 2 y=g1#h1, g1=g2+h2, g2=g3&s,	g3=/x1#x2, h2=g5&h5, g5=x2,	h5=x3#x4, h1=g6&h6, g6=x5&x6,	h6=g7&h7, g7=x2+/x4, h7=x3#/x4.
4	sbe 1 3	h3=g4#h4,	g4=x2,	h4=x3+x4.

5	sbe 1 4 y=1. sbe 1 5 h3=1.	sbe 1 6 t=1.	vtin 1 7 g1 g2 g3 g4 g5 g6 g7 h1 h2 h3 h4 h5 h6 h7.

6	isc 3 5 10	maxk 10 5 11	maxk 11 7 12	syd 12 6 13
7	isc 2 4 20	maxk 20 4 21	maxk 21 7 22	_derk 22 <s> 23

a)

« » O K TVL 42 (ODA) | 8 Var.

	x1	x2	x3	x4	x5	x6	y	t
1:	1	1	0	0	1	1	0	0
2:	1	1	0	0	0	1	1	0
3:	1	1	0	0	-	0	1	0
4:	1	1	1	1	1	1	1	1
5:	1	1	1	1	0	1	0	1
6:	1	1	1	1	-	0	0	1
7:	0	0	0	0	1	1	1	1
8:	0	0	0	0	0	1	0	1
9:	0	0	0	0	-	0	0	1
10:	0	0	1	-	-	-	1	0
11:	0	0	0	1	-	-	1	0

b)

« » O K TVL 31 (ODA) | 8 Var.

	x1	x2	x3	x4	x5	x6	y	t
1:	0	0	0	0	1	1	1	1
2:	0	0	0	0	0	1	0	1
3:	0	0	0	0	-	0	0	1
4:	1	1	1	1	-	0	0	1
5:	1	1	0	0	1	1	0	0
6:	1	1	0	0	0	1	1	0
7:	1	1	0	0	-	0	1	0
8:	0	0	1	-	-	-	1	0
9:	1	1	1	1	1	1	1	1
10:	1	1	1	1	0	1	0	1
11:	0	0	0	1	-	-	1	0

Figure 7.17 All test patterns of the selected sensible point h_3 of the circuit shown in Fig. 7.16: **a**) calculated by the detailed formulas (7.146), . . . , (7.149) of [18], **b**) calculated by the formula (7.150) of [18]

8 maxk 1 7 30

9 isc 13 23 40 isc 40 30 41 obbc 41 42
Figure 7.17 a) shows the 24 test pattern of the sensible point h_3 in TVL 42.

10 The value of the model variable t indicates the type as SAt test pattern. In all test patterns for the sensible point h_3 the variables x_1 and x_2 possess the same values. Hence, there is no test pattern where only the change of the x_2 value changes between an SA0 and an SA1 test pattern, how has been explored in the method of the sensible path.

Exercise 7.39.

1 Use the same solution as the first task of Exercise 7.38.

2 Use the same solution as the second task of Exercise 7.38.

3 sbe 1 2
y=g1#h1,
g1=g2+h2,
g2=g3&s,
g3=/x1#x2,
/t=g4#h4,
g4=x2,
h4=x3+x4,
h2=g5&h5,
g5=x2,
h5=x3#x4,
h1=g6&h6,
g6=x5&x6,
h6=g7&h7,
g7=x2+/x4,
h7=x3#/x4.

4 vtin 1 7
g1 g2 g3 g4 g5 g6 g7 h1 h2 h3 h4 h5 h6 h7.

5 maxk 2 7 10

6 maxk 1 7 20

7 _derk 10 <s> 11 isc 11 20 30 obbc 30 31
Figure 7.17 b) shows the 24 test patterns of the sensible point h_3 in TVL 31. Figure 7.17 a) shows the same test patterns in another order.

Exercise 7.40.

1 Use the same solution as the first task of Exercise 7.38.

a)

« | » | O | K | TVL 24 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	1	1	0	0	1	1	0	0
2:	1	0	0	1	1	1	0	1
3:	0	-	0	0	1	1	1	0
4:	0	0	0	1	1	1	1	1
5:	1	0	0	0	1	1	1	0
6:	-	1	-	1	1	1	1	1
7:	-	1	1	0	1	1	1	0

b)

« | » | O | K | TVL 34 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	-	0	0	0	1	1	1	0
2:	0	0	1	1	1	1	1	1
3:	1	0	1	1	1	1	0	1

c)

« | » | O | K | TVL 44 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	1	1	1	0	1	1	1	0
2:	1	0	1	0	1	1	0	0
3:	1	1	0	0	1	1	0	0
4:	0	0	1	0	1	1	1	0
5:	0	1	-	0	1	1	1	0
6:	-	0	0	0	1	1	1	0
7:	-	1	-	1	1	1	1	1

d)

« | » | O | K | TVL 50 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	0	0	0	1	1	1	1	1
2:	1	0	0	1	1	1	0	1

e)

« | » | O | K | TVL 51 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	1	0	1	1	1	1	0	1
2:	0	0	1	1	1	1	1	1

f)

« | » | O | K | TVL 52 (ODA) |

	x1	x2	x3	x4	x5	x6	y	t
1:	0	0	1	0	1	1	1	0
2:	1	0	1	0	1	1	0	0

Figure 7.18 Test pattern of the sensible point on the local branch nx_4 of the circuit shown in Fig. 7.16: **a**) all test patterns of the signal source, **b**) all test patterns of the signal target s_1, **c**) all test patterns of the signal target s_2, **d**) test patterns that detect errors of the signal source only, **e**) test patterns that detect errors of the signal target s_1 only, **f**) test patterns that detect errors of the signal target s_2 only

```
2  sbe 1 1
   y=g1#h1,        h3=g4#h4,       g5=x2,          h6=g7&h7,
   g1=g2+h2,       g4=x2,          h5=x3#x4,       g7=x2+s1,
   g2=g3&h3,       h4=x3+x4,       h1=g6&h6,       h7=x3#s2,
   g3=/x1#x2,      h2=g5&h5,       g6=x5&x6,       nx4=/x4.
3  sbe 1 2         tin 1 3         001
   /t=nx4.         s1 s2 t.        110.
4  vtin 1 4                        vtin 1 6        vtin 1 8
   g1 g2 g3 g4 g5 g6 g7            t.              s2.
   h1 h2 h3 h4 h5 h6 h7 nx4.       vtin 1 7        vtin 1 9
   vtin 1 5                        s1.             s1 s2.
   s1 s2 t.
5  isc 1 2 10                      maxk 10 4 11
6  isc 11 3 12                     maxk 12 5 13
7  maxk 3 6 20     derk 21 9 22    obbc 23 24      shows the test
   isc 20 11 21    isc 22 13 23    Figure 7.18 a)  pattern as object
                                                   24.
8  maxk 3 7 30     derk 31 9 32    obbc 33 34      shows the test
   isc 30 11 31    isc 32 13 33    Figure 7.18 b)  pattern as object
                                                   34.
9  maxk 3 8 40     derk 41 9 42    obbc 43 44      shows the test
   isc 40 11 41    isc 42 13 43    Figure 7.18 c)  pattern as object
                                                   44.
10 dif 24 34 50    Figure 7.18 d) shows the wanted test pattern as object 50.
   dif 50 44 50
11 dif 34 24 51    Figure 7.18 e) shows the wanted test pattern as object 51.
12 dif 44 24 52    Figure 7.18 f) shows the wanted test pattern as object 52.
13 isc 51 52 53    The TVL 53 is empty. Thus, no such test pattern exists.
```

Exercise 7.41.

1 lds e73dec9.sdt sbe 1 50 x2#t.

2 isc 50 39 51
isc 51 13 52
obbc 52 53
SA1 test patterns require $x_1 = 0, x_2 = 0$, and SA0 test patterns requires $x_1 = 1, x_2 = 1$ in combination with either $x_3 = 0, x_4 = 0$ or $x_3 = 1, x_4 = 1$.

3 isc 53 4 54 There are 16 SA1 test patterns and 8 SA0 test patterns.

Exercise 7.42.

1
```
space 64 1      g1=x3&nx4,   g10=g7&g9,    g19=g5&g18,
sbe 1 1         g2=nx1&g1,   g11=x1&s,     g20=g2+g4,
nx1=/x1,        g3=nx2&x4,   g12=g7&g11,   g21=g6+g8,
nx2=/x2,        g4=nx1&g3,   g13=nx6&g12,  g22=g10+g13,
nx3=/x3,        g5=x5&x6,    g14=x2&g1,    g23=g14+g16,
nx4=/x4,        g6=nx1&g5,   g15=x2&x4,    g24=g20+g21,
nx5=/x5,        g7=x2&nx3,   g16=g5&g15,   g25=g22+g23,
nx6=/x6,        g8=x4&g7,    g17=nx2&nx3,  g26=g24+g25,
                g9=x1&nx5,   g18=nx4&g17,  y=g19+g26.
```

2
```
sbe 1 2         vtin 1 4
x5=/t.          g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13
sbe 1 3         g14 g15 g16 g17 g18 g19 g20 g21 g22 g23 g24 g25 g26
t#s=1.          nx1 nx2 nx3 nx4 nx5 nx6.
```

3 isc 1 2 10 maxk 10 4 11

4 isc 11 3 12 maxk 12 3 13

5 _derk 11 <s> 14 isc 14 13 15

There exists the SA0 test pattern $(x_1, x_2, x_3, x_4, x_5, x_6, y, t) = (11001010)$ only.

6 Because there is no SA1 test pattern for the input x_5 of gate g_{11}, this SA1 fault does not affect the circuit behavior. For that reason the input can be substituted by a constant value 1 such that finally the gate g_{11} can be replaced by a wire from x_1 to g_{12}. An alternative reconfiguration was given in Task 2 of Exercise 7.23.

Chapter 8

FINITE-STATE MACHINES

1. The Circuit Model

A sequential circuit is realized by a structure of basic elements. These basic elements are visualized in a schematic diagram by symbols. The connection wires between the basic elements of a circuit are expressed by lines in the schematic diagram. The basic elements of a sequential circuit can be dtaken from logic gates and flip-flops. Table 7.1 of [18] gives a list which includes for each gate the associated symbol together with its logic function, and its list of phases. In contradiction to all types of gates a flip-flop can store one logic value. Table 8.3 of [18] summarizes for several flip-flops the associated symbols together with their graphs, their logic functions and their lists of phases.

A sequential circuit possesses a unique behavior, but the same behavior may be realized by different circuit structures. Therefore both structural and behavioral models are necessary in order to describe a sequential circuit.

The main tasks for sequential circuits perform transformations between these classes of models. The basic task of analysis requires a structural model of a sequential circuit and creates the associated behavioral model. Vice versa, the basic task of synthesis takes a behavioral model of a sequential circuit and leads to structural models of one or several associated circuits.

This chapter includes both exercises for analysis and synthesis of sequential circuits. The required models are prepared in this section. It is strongly recommended to store all the solutions of each exercise because succeeding exercises may use results of previous exercises. For comfort-

Bernd Steinbach, Christian Posthoff, *Logic Functions and Equations, Examples and Exercises,*

able work and sometimes required corrections it is suggested to store for each exercise

- your own prepared problem programs (PRPs),
- complete TVL systems created in the XBOOLE Monitor, and
- the PRP that documents all executed steps to solve all tasks of an Exercise which is generated by the menu item `Extras` – `Save Protocol as PRP`.

The behavior of a finite-state machine can be described by a graph. The vertices of the graph represent the states of the finite-state machine and are labeled by names. The edges of the graph indicate the following state to be reached and can be labeled by an input condition. Depending on the type (Moore or Mealy) of the finite-state machine, information about the outputs is associated to the vertices or the edges of the graph. In order to realize a finite-state machine as sequential circuit, each state must be coded by logic values.

Alternatively the behavior of a finite-state machine can be expressed by its system function $F(\mathbf{x}, \mathbf{s}, \mathbf{s}', \mathbf{y})$, where $\mathbf{x}$ indicates the inputs, $\mathbf{s}$ the states, $\mathbf{s}'$ the states in the following time step, and $\mathbf{y}$ the outputs, respectively. For practical reasons we use in this chapter $\mathbf{sf}$ instead of $\mathbf{s}'$. The solution of the system equation $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}) = 1$ is the list of phases of the finite-state machine.

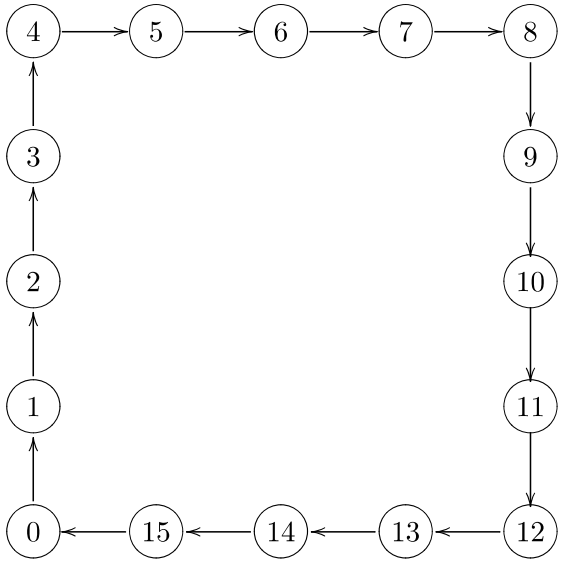
Exercise 8.1 (Behavioral Model – Simple Traffic Light – List of Phases). Figure 8.1 describes the behavior of a simple traffic light that can be used to control a road work, where three periods of time are required to pass the distance to be in work and yellow phases are included for security reasons. Create a list of phases as behavioral model for both traffic lights. Use a binary code for the states that indicates the state number directly, and a 1-out-of-3 code for the outputs that controls the color of both traffic lights. Prepare this list of phases as PRP such that it can be used later on for an analysis and synthesis tasks. Practical tasks:

1 How many variables are required to code the states.

2 Prepare a Boolean space large enough for the list of phases.

3 Define the list of phases as object number 1.

4 Add variable tuples for state variables as object number 3, state variables for the following time step as object number 4, and outputs as object number 5. Object number 2 is not used because there are no input variables in this finite-state machine. Store TVL system as `e8101.sdt` for later use.



state	tl1	tl2
0	r/y	r
1, 2, 3	g	r
4	y	r
5, 6, 7	r	r
8	r	r/y
9, 10, 11	r	g
12	r	y
13, 14, 15	r	r

abbreviations:

tl1:	traffic light 1
tl2:	traffic light 2
r:	red
y:	yellow
g:	green

Figure 8.1 Behavior of a synchronous finite-state machine of a simple control for a road work traffic light: *left hand side* – graph, *right hand side* – assignment of the light colors

The traffic control of Exercise 8.1 allows only in three periods of time to drive in one direction. Assume this traffic control is used on a main street at the border of a town: a strong traffic jam will occur in the morning in direction to the town and in the evening in the reverse direction, but not so strong. For that reason a controllable finite-state machine for the traffic light is required.

Exercise 8.2 (Behavioral Model – Extended Traffic Light – Graph – List of Phases). Create a behavioral model of a finite-state machine for the traffic light that extends the model of Fig. 8.1. Two input variables $(x1, x2)$ allow to control the extended traffic light. If $x1 = 0$, the traffic light behaves as shown in Fig. 8.1 which is the preferred mode for traffic in both directions. If $x1 = 1$ and $x2 = 0$, three additional green states are included before the state 1 for the strong traffic in direction to the town. If $x1 = 1$ and $x2 = 1$, two additional green states are included before the state 9 for the strong traffic in direction out of the town. Practical tasks:

1 Draw the graph of the extended finite-state machine and extend the table of the required outputs.

2 How many variables are required to code the states.

3 Prepare a Boolean space which is large enough for the list of phases.

4 Define the list of phases as object number 1.

5 Add variable tuples for inputs as object number 2, state variables as object number 3, state variables for the following time step as object number 4, and outputs as object number 5. Store the TVL system as `e8102.sdt` for later use.

In addition to the behavioral model of a finite-state machine a model of its structure is required as the result of the synthesis or as a source of the analysis. The structure of a finite-state machine is given by its sequential circuit. The structure model must represent the switching elements. In the case of a clocked sequential circuit these elements are gates and flip-flops. Both types may be expressed in a structural model either by a logic equation or by local lists of phases. The connection wires between these elements are shown in the schema by lines. In the structural model the same variable in descriptions of different elements expresses such a connection. Variable tuples of the inputs, state variables, state variables for the following time step, and outputs complete the structural model of a finite state machine.

Exercise 8.3 (Structural Model – System of Equations). Figure 8.2 describes the structure of a finite-state machine using three types of flip-flops and AND-, OR-, and EXOR-gates. Create a system of logic equations as structural model for this sequential circuit. Add information about the meaning of the variables using variable tuples. Prepare this system of equations together with the variable tuples as PRP such that it can be used later on for an analysis task. Practical tasks:

1 How many states are coded by the flip-flops of the sequential circuit?

2 Prepare a Boolean space large enough for the used variables and associate the variables of the inputs, states, states for the following time step, and outputs in an appropriate order.

3 Define the system of equations of the circuit elements as object number 1. Use the names of the logic gates as signal names of their outputs and the signal names given in the circuit structure for the flip-flops. Remember that each flip-flop creates both the state signal and its negated signal on the outputs.

4 Add variable tuples for inputs as object number 2, state variables as object number 3, state variables for the following time step as object number 4, outputs as object number 5, and all other internal variables as object number 6. Store the TVL system as `e8103.sdt` for later use.

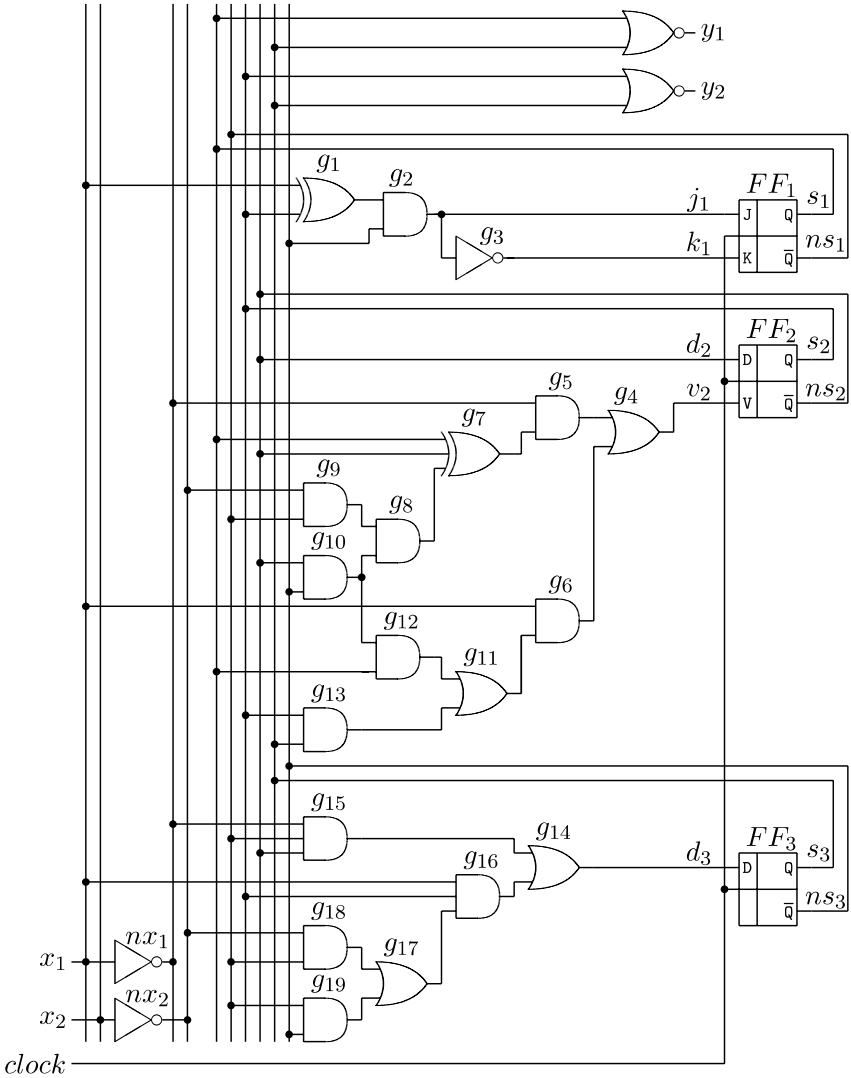


Figure 8.2 Structure of a finite-state machine using three types of flip-flops and AND-, OR-, and EXOR-gates restricted to three inputs

Alternatively to the equations of the switching elements in Fig. 8.2 local lists of phases can be used.

Exercise 8.4 (Structural Model – Set of Local Lists of Phases). Create a set of local lists of phases as structural model for the sequential circuit given in Fig. 8.2. This set of local lists of phases will be used later on in combination with the PRP of Exercise 8.3. Practical tasks:

1 Prepare a PRP that creates a set of local lists of phases for all switching elements of the sequential circuit given in Fig. 8.2. Use the object numbers larger than 10 so that this PRP can used in combination with the PRP of Exercise 8.3.

2 Store the TVL system as `e8104.sdt` for later use.

2. Analysis

The basic analysis task for sequential circuits is the calculation of the behavior realized by the circuit. The set of solution vectors of the system of logic equations given as structural model describes the global list of phases of the associated sequential circuit. Hence, it is a behavioral model that can be interpreted as system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$. In addition to these variables the variables of negated inputs $\mathbf{nx}$ the negated state variables $\mathbf{ns}$, the variables of internal gate functions $\mathbf{g}$, and the variables of the flip-flop inputs can be included in the system function. Equivalently to the global list of phases, the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ depicts the graph of the finite-state machine and its behavior.

Exercise 8.5 (Behavior of a Sequential Circuit Based on a System of Logic Equations). Calculate the behavior of the sequential circuit given in Fig. 8.2. Use the system of logic equations prepared in Exercise 8.3 as structural model. Practical tasks:

1 What type of finite-state machine is given by the sequential circuit in Fig. 8.2?

2 The sequential circuit depends on 2 inputs and includes 3 flip-flops, How many phases exist?

3 Solve the system of equations prepared in Exercise 8.3 and show the global list of phases.

4 Simplify the global list of phases that depends on all variables to a global list of phases that depends on the necessary variables to describe the behavior of the finite-state machine. Show the associated system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$.

5 Store TVL system as `e8201.sdt` for later use.

6 Draw the graph of the analyzed finite-state machine and describe the visualized behavior.

Alternatively to the structural model by a system of logic equations the set of local lists of phases can be used in order to calculate the global

behavior of the sequential circuit. Independent structural models allow the verification of the calculated behavior.

Exercise 8.6 (Behavior of a Sequential Circuit Based on a Set of Local Lists of Phases). Calculate the behavior of the circuit given in Fig. 8.2. Use the set of local lists of phases prepared in Exercise 8.4 as structural model and verify whether the calculated behavior coincides with the result of Exercise 8.5. Practical tasks:

1 Load the final TVL system of Exercise 8.5.

2 Create the set of local lists of phases executing the PRP prepared in Exercise 8.4.

3 Calculate the intersection of all 34 local lists of phases of the gates, flip-flops, and connections.

4 Verify whether the behavior calculated on the basis of local lists of phases coincides with the behavior of the circuit calculated by solving a system of equations in Exercise 8.5 which is available as object number 1.

A global list of phases represents for a finite-state machine the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ and describes its allowed behavior completely. The selection of certain partial behaviors can be helpful to understand the global behavior. Such analysis steps can be combined to simulations of the behavior, both in forward and in backward direction of the final state machine.

Exercise 8.7 (Partial Behavior of a Sequential Circuit). Calculate several partial behaviors of the sequential circuit of Fig. 8.2. The global list of phases of this finite-state machine is available as object 7 in the solution of Exercise 8.5. This solution includes additionally variable tuples for inputs as object 2, variables of the states as object 3, variables of the states at the following time step as object 4, and outputs as object 5. Minimize the number of rows in the solution only, but not the fixed demand information. Practical tasks:

1 Load the final TVL system of Exercise 8.5.

2 Calculate the partial behaviors for all four possible input patterns.

3 How many and which states can be reached in the following time step starting form the states (s_1, s_2, s_3) equal to (000), (100), and (101).

4 From how many and which states it is possible to reach the states (sf_1, sf_2, sf_3) equal to (000), (100), and (101) in the following time step.

5 Are there states which can not be reached?

An important analysis task as precondition for the design is to verify whether the finite-state machine is realizable by a sequential circuit. This precondition requires that the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ allows for each state and each input pattern at least one phase, e.g. one state for the following time step and one output pattern. This task is equivalent to the check whether the system equation $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}) = 1$ is solvable with regard to the dependent variables $(\mathbf{sf}, \mathbf{y})$ in the sequential circuit. Remember, the system equation is uniquely solvable with regard to the dependent variables $(\mathbf{sf}, \mathbf{y})$ for any sequential circuit.

Exercise 8.8 (Verification of the Realizability of a Finite-State Machine Specified by a Sequential Circuit). Analyze the realizability of several finite-state machines. Detailed information about the finite-state machines to be analyzed are given in Exercises 8.7. Practical tasks:

1 Load the final TVL system of Exercise 8.5.

2 Verify whether the system function of object number 7 is realizable.

3 Verify that all memory functions $sf_i(x_1, x_2, s_1.s_2, s_3), i = 1, 2, 3$ are uniquely specified by the system equation $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}) = 1$.

4 Verify that all result functions $y_j(x_1, x_2, s_1.s_2, s_3), j = 1, 2$ are uniquely specified by the system equation $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}) = 1$.

Exercise 8.9 (Check of the Realizability of a Finite-State Machine – Simple Traffic Light). Analyze the realizability of the finite-state machine for the simple road work traffic lights defined in Exercises 8.1. Detailed information about this finite-state machine is given in the mentioned exercise. Practical tasks:

1 Load the final TVL system of Exercise 8.1.

2 What type of finite-state machine possesses the global list of phases of the simple traffic light control specified in Exercises 8.1 and shown in Fig. 8.3 a).

3 Is the finite-state machine of the simple traffic light control given as object number 1 realizable? Are there missing phases, and, if yes, how many such phases exist?

Exercise 8.10 (Check of the Realizability of a Finite-state Machine – Extended Traffic Light). Analyze the realizability of the finite-state machine for the extended road work traffic lights defined in Exercises 8.2. Detailed information about this finite-state machine is given in the mentioned Exercise. Practical tasks:

1 Load the final TVL system of Exercise 8.2.

2 What type of finite-state machines possesses the global list of phases of the extended traffic light control specified in Exercises 8.2 and shown in Fig. 8.3 b).

3 Is the finite-state machine of the extended traffic light control given an object number 1 realizable? Are there missing phases, and, if yes, how many such phases exist?

3. Design

The basic design task for finite-state machines is the calculation of the structure of a sequential circuit that realizes a given behavior. The most common behavioral description is the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ that describes the allowed input-output phases.

The realizability of a given finite-state machine is a precondition of the design of a sequential circuit. In Exercise 8.8 this precondition was verified for the finite-state machine of Fig. 8.7. In the practical Task 5 of Exercise 8.7 it was found that the states (111) and (101) can not be reached from any other state. Such states outside of the main behavior can be used to simplify the circuit structure.

We assume for the next exercises that the states (111) and (101) are not necessary for the behavior of the finite-state machine of Fig. 8.7. Often such states fill up the required number of states to the next higher number a power of 2. In order to get some freedom for optimization we allow that any other state coded by the used 3 state variables can be the reached state of the following time step for these two inessential states.

Exercise 8.11 (Definition of a Non-deterministic Finite-state Machine). Define the allowed behavior of a non-deterministic finite-state machine by means of a global list of phases. In the 6 states (s_1, s_2, s_3) equal to (000), (100), (110), (010), (011), and (001) the deterministic behavior must be identical to the global list of phases given in Fig. 8.6. In the additional states (s_1, s_2, s_3) equal to (111) and (101) any of the necessary other states may be reached in the following time step. In these two states any output pattern is allowed. Practical tasks:

1 Why is the finite-state machine specified in this exercise a non-deterministic machine?

2 Prepare a PRP that defines a space large enough to allow the design of this non-deterministic finite-state machine completely.

3 Define the global list of phases for the non-deterministic finite-state machine, described in this Exercise, as object number 1.

4 Add variable tuples for inputs as object number 2, state variables as object number 3, state variables for the following time step as object number 4, and outputs as object number 5.

5 Execute the PRP and store the TVL system as `e8301.sdt` for later use.

In the next exercises the controlling functions of the flip-flop inputs will be calculated. In order to get more routine several types of flip-flops are used. For reasons of comparability the same types of flip-flops as used in Fig. 8.2 were selected.

Subtasks of the next exercises require the design of the combinatorial circuits based on the mark functions of ON-sets $q(\mathbf{x}, \mathbf{s})$ and OFF-sets $r(\mathbf{x}, \mathbf{s})$. These subtasks can be solved by bi-decomposition as studied in Sect. 3.. Alternatively the function can be selected from the Karnaugh-maps and simplified using algebraic transformation rules.

Exercise 8.12 (Controlling Functions j_1 and k_1). The memory function sf_1 of the non-deterministic finite-state machine defined in Exercise 8.11 shall be realized by a JK-flip-flop. Calculate the controlling functions j_1 and k_1 taking into consideration the freedom of the specified non-deterministic behavior. Restrict the behavior of the finite-state machine regarding the selected controlling functions j_1 and k_1. Practical tasks:

1 Load the final TVL system of Exercise 8.11. Object number 1 includes the non-deterministic system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ for which the circuit structure of the memory function sf_1 must be calculated.

2 Map the behavior of the memory function sf_1 to the behavior of controlling functions j_1 and k_1 using an appropriate list of phases of a JK-flip-flop taken from Table 8.3 in [18]. The system function $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ will be created.

3 Restrict $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ to the allowed behavior of the controlling functions j_1 and k_1 which results in $F_{1jk}(\mathbf{x}, \mathbf{s}, j_1, k_1)$.

4 Calculate the list of phases $F_{1j}(\mathbf{x}, \mathbf{s}, j_1)$ for the controlling function j_1 of the JK-flip-flop.

5 Calculate the don't-care function $j_{1\varphi}(\mathbf{x}, \mathbf{s})$.

6 Calculate the function $j_{1q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function j_1 and show the associated Karnaugh-map.

7 Calculate the function $j_{1r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function j_1 and show the associated Karnaugh-map.

8 Select a simple function for the controlling functions j_1.

9 Restrict $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ to the allowed behavior with regard the selected controlling functions j_1.

10 Restrict the new system function $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ into the allowed behavior $F_{1jk}(\mathbf{x}, \mathbf{s}, j_1, k_1)$ of the controlling functions j_1 and k_1.

11 Calculate the list of phases $F_{1k}(\mathbf{x}, \mathbf{s}, k_1)$ for the controlling function k_1 of the JK-flip-flop.

12 Calculate the don't-care function $k_{1\varphi}(\mathbf{x}, \mathbf{s})$.

13 Calculate the function $k_{1q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function k_1 and show the associated Karnaugh-map.

14 Calculate the function $k_{1r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function k_1 and show the associated Karnaugh-map.

15 Select a simple function for the controlling function k_1.

16 Restrict $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ to the allowed behavior with regard the selected controlling function k_1.

17 Remove all information about j_1 and k_1 from the final system function $F_1(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, j_1, k_1)$ and store the restricted result $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ as object number 6.

18 Which effect has the selection of the controlling functions j_1 and k_1 to the remaining system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$?

19 Store the TVL system as `e8302.sdt` for later use.

Exercise 8.13 (Controlling Functions d_2 and v_2). The memory function sf_2 of the non-deterministic finite-state machine restricted in Exercise 8.12 shall be realized by a DV-flip-flop. Calculate the controlling functions d_2 and v_2 taking into consideration the freedom of the non-deterministic specified behavior. Restrict the behavior of the finite-state machine regarding the selected controlling functions d_2 and v_2. Practical tasks:

1 Load the final TVL system of Exercise 8.12. Object number 6 includes the non-deterministic system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ for which a circuit structure of the memory function sf_2 must be calculated.

2 Map the behavior of the memory function sf_2 to the behavior of controlling functions d_2 and v_2 using an appropriate list of phases of a DV-flip-flop taken from Table 8.3 in [18]. The system function $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ will be created.

3 Restrict $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ to the allowed behavior of the controlling functions d_2 and v_2 which results in $F_{2dv}(\mathbf{x}, \mathbf{s}, d_2, v_2)$.

4 Calculate the list of phases $F_{2d}(\mathbf{x}, \mathbf{s}, d_2)$ for the controlling function d_2 of the DV-flip-flop.

5 Calculate the don't-care function $d_{2\varphi}(\mathbf{x}, \mathbf{s})$.

6 Calculate the function $d_{2q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function d_2 and show the associated Karnaugh-map.

7 Calculate the function $d_{2r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function d_2 and show the associated Karnaugh-map.

8 Select a simple function for the controlling functions d_2.

9 Restrict $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ to the allowed behavior with regard the selected controlling function d_2.

10 Restrict the new system function $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ to the allowed behavior $F_{2dv}(\mathbf{x}, \mathbf{s}, d_2, v_2)$ of the controlling functions d_2 and v_2.

11 Calculate the list of phases $F_{2v}(\mathbf{x}, \mathbf{s}, v_2)$ for the controlling function v_2 of the DV-flip-flop.

12 Calculate the don't-care function $v_{2\varphi}(\mathbf{x}, \mathbf{s})$.

13 Calculate the function $v_{2q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function v_2 and show the associated Karnaugh-map.

14 Calculate the function $v_{2r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function v_2 and show the associated Karnaugh-map.

15 Select a simple function for the controlling function v_2.

16 Restrict $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ to the allowed behavior with regard to the selected controlling function v_2.

17 Remove all information about d_2 and v_2 from the final system function $F_2(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_2, v_2)$ and store the restricted result $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ as object number 7.

18 Which effect has the selection of the controlling functions d_2 and v_2 to the remaining system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$?

19 Store the TVL system as `e8303.sdt` for later use.

Exercise 8.14 (Controlling Function d_3). The memory function sf_3 of the non-deterministic finite-state machine restricted in Exercise 8.13 shall

be realized by a D-flip-flop. Calculate the controlling function d_3 taking into consideration the freedom of the non-deterministic specified behavior. Restrict the behavior of the finite-state machine regarding the selected controlling function d_3. Practical tasks:

1 Load the final TVL system of Exercise 8.13. Object number 7 includes the non-deterministic system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ for which a circuit structure of the memory function sf_3 must be calculated.

2 Map the behavior of the memory function sf_3 to the behavior of the controlling function d_3 using an appropriate list of phases of a D-flip-flop taken from Table 8.3 in [18]. The system function $F_3(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_3)$ will be created.

3 Restrict $F_3(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_3)$ to the allowed behavior of the controlling functions d_3 which results in $F_{3d}(\mathbf{x}, \mathbf{s}, d_3)$.

4 Calculate the don't-care function $d_{3\varphi}(\mathbf{x}, \mathbf{s})$.

5 Calculate the function $d_{3q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function d_3 and show the associated Karnaugh-map.

6 Calculate the function $d_{3r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function d_3 and show the associated Karnaugh-map.

7 Select a simple function for the controlling functions d_3.

8 Restrict $F_3(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_3)$ to the allowed behavior with regard the selected controlling function d_3.

9 Remove the information about d_3 from $F_3(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y}, d_3)$ and store the restricted system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ as object number 8.

10 Which effect has the selection of the controlling functions d_3? to the remaining system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$.

11 Store the TVL system as `e8304.sdt` for later use.

In order to complete the design of the finite-state machine the circuit structure for the output functions must be calculated.

Exercise 8.15 (Output Functions y_1 and y_2). Calculate output functions y_1 and y_2 of the non-deterministic finite-state machine restricted in Exercise 8.14. Restrict the behavior of the finite-state machine regarding the selected output functions y_1 and y_2, respectively. Practical tasks:

1 Load the final TVL system of Exercise 8.14. Object number 8 includes the non-deterministic system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ for which the a circuit structure of the output functions y_1 and y_2 must be calculated.

2 Restrict $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ to the allowed behavior of the output functions y_1 and y_2 which results in $F_y(\mathbf{x}, \mathbf{s}, \mathbf{y})$.

3 Calculate the list of phases $F_{y1}(\mathbf{x}, \mathbf{s}, y_1)$ for the first output function y_1.

4 Calculate the don't-care function $y_{1\varphi}(\mathbf{x}, \mathbf{s})$.

5 Calculate the function $y_{1q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function y_1 and show the associated Karnaugh-map.

6 Calculate the function $y_{1r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function y_1 and show the associated Karnaugh-map.

7 Select a simple function for the output y_1.

8 Restrict $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ to the allowed behavior with regard to the selected output function y_1.

9 Restrict the new system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ to the allowed behavior $F_y(\mathbf{x}, \mathbf{s}, \mathbf{y})$ of the output functions y_1 and y_2.

10 Calculate the list of phases $F_{y2}(\mathbf{x}, \mathbf{s}, y_2)$ for the output function y_2.

11 Calculate the don't-care function $y_{2\varphi}(\mathbf{x}, \mathbf{s})$.

12 Calculate the function $y_{2q}(\mathbf{x}, \mathbf{s})$ that describes the ON-set of the incompletely specified function y_2 and show the associated Karnaugh-map.

13 Calculate the function $y_{2r}(\mathbf{x}, \mathbf{s})$ that describes the OFF-set of the incompletely specified function y_2 and show the associated Karnaugh-map.

14 Select a simple function for the output functions y_2.

15 Restrict $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ to the allowed behavior with regard to the selected output function y_2 and store the restricted result $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ as object number 9.

16 Which effect has the selection of the controlling functions y_1 and y_2 on the remaining system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$?

17 Store the TVL system as `e8305.sdt` for later use.

The function of the circuit of Fig. 8.2 was analyzed. The associated behavior was extended to a non-deterministic finite-state machine in Exercise 8.11 and re-designed using the same three types of flip-flops. In order to do this, the method of merging behaviors of finite-state machine and the used flip-flops was applied, followed by the subtask of solving equations with regard to variables. In order to optimize the circuit the

degrees of freedom of both the non-deterministic behavior of the finite-state machine and the control functions for the flip-flop were exploited. The detailed results were calculated in Exercises 8.12 ... 8.15.

Exercise 8.16 (Technology Mapping and Verification). Draw both the graph that describes the behavior of the sequential circuit and the circuit structure calculated in Exercises 8.12 ... 8.15. Verify whether the behavior of the designed structure is covered by the allowed behavior defined in Exercise 8.11. Compare both the required numbers of gates and the depth of the designed structure with the given sequential circuit of Fig. 8.2. Practical tasks:

1 Load the TVL system `e8305.sdt` of Exercise 8.15. This TVL system includes the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ of the given non-deterministic finite-state machine as object number 1 and the respective system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ of the designed sequential circuit as object number 9.

2 Show the global lists of phases of both the given non-deterministic finite-state machine of object number 1 and the designed deterministic finite-state machine of object number 9.

3 Draw the graph of the designed finite-state machine and describe the visualized behavior in comparison to the graph of Fig. 8.7.

4 Draw the structure of the sequential circuit calculated by solving equations with regard to variables in Exercises 8.12... 8.15. As in Fig. 8.2 the number of inputs of the gates is restricted to three.

5 Verify whether the behavior of the structure of the sequential circuit is covered by the allowed behavior defined in Exercise 8.11.

6 Compare both the required numbers of gates and the depth of the designed circuit structure with the given sequential circuit of Fig. 8.2.

In the next Exercise the circuit structure for a deterministic finite-state machine will be calculated. The used JK-flip-flops lead to incompletely specified controlling functions for the memory inputs.

Exercise 8.17 (Complete Circuit Design of the Simple Control for a Road Work Traffic Light). Design the sequential circuit of a simple control for a road work traffic light and verify the result of the synthesis. Use the same method as applied in the previous exercises. The behavioral model is given in Fig. 8.1 and the result of Exercise 8.1. Practical tasks:

1 Load the TVL system `e8101.sdt` of Exercise 8.1. This TVL system includes the system function $F(\mathbf{s}, \mathbf{sf}, \mathbf{y})$ of the given deterministic finite-state machine as object number 1.

2 Calculate all required controlling functions using JK-flip-flops. Calculate additionally the output functions and the resulting global list of phases. Check the correctness of each calculated memory and output function immediately.

3 Verify whether the designed sequential circuit realizes the required behavior.

In the final exercise the circuit structure for a non-deterministic finite-state machine will be calculated. The behavior description of this finite-state machine is not realizable and must be extended before the design can start. Both the non-deterministic behavior and the used DV-flip-flops contribute to incompletely specified controlling functions for the memory inputs.

Exercise 8.18 (Complete Circuit Design of the Extended Control for a Road Work Traffic Light). Design the sequential circuit of an extended control for a road work traffic light and verify the result of the synthesis. Use the same method as applied in the previous exercises. The behavioral model is given in Fig. 8.3 b), and the result of Exercise 8.2, and the associated graph in Fig. 8.4. Practical tasks:

1 Load the TVL system `e8102.sdt` of Exercise 8.2. This TVL system includes the system function $F(\mathbf{x}, \mathbf{s}, \mathbf{sf}, \mathbf{y})$ of the given non-deterministic finite-state machine as object number 1.

2 It is known from Exercise 8.10 that this finite-state machine is not realizable. Extend the finite-state machine such that from each state not used so far any of the used states is allowed to be reached and for security reasons of the traffic light all output functions must be 0 which means the controlled lights are off. Solve this task by appropriate XBOOLE operations.

3 Verify whether the completed finite-state machine of the extended control for a road work traffic light is realizable.

4 Calculate all required controlling functions using DV-flip-flops. Calculate additionally the output functions and the resulting global list of phases. Check the correctness of each calculated memory and output function immediately.

5 Verify whether the designed sequential circuit realizes the required behavior.

6 Draw the graph of the realized finite-state machine of an extended control for a road work traffic light and compare it with the given behavior of Fig. 8.4.

a)

« » O TVL 1 (ODA) | 14 Var. | 16 R. | Sp. 1

	s3	s2	s1	s0	sn3	sn2	sn1	sn0	r1	y1	g1	r2	y2	g2
1:	0	0	0	0	0	0	0	1	1	1	0	1	0	0
2:	0	0	0	1	0	0	1	0	0	0	1	1	0	0
3:	0	0	1	0	0	0	1	1	0	0	1	1	0	0
4:	0	0	1	1	0	1	0	0	0	0	1	1	0	0
5:	0	1	0	0	0	1	0	1	0	1	0	1	0	0
6:	0	1	0	1	0	1	1	0	1	0	0	1	0	0
7:	0	1	1	0	0	1	1	1	1	0	0	1	0	0
8:	0	1	1	1	1	0	0	0	1	0	0	1	0	0
9:	1	0	0	0	1	0	0	1	1	0	0	1	1	0
10:	1	0	0	1	1	0	1	0	1	0	0	0	0	1
11:	1	0	1	0	1	0	1	1	1	0	0	0	0	1
12:	1	0	1	1	1	1	0	0	1	0	0	0	0	1
13:	1	1	0	0	1	1	0	1	1	0	0	0	1	0
14:	1	1	0	1	1	1	1	0	1	0	0	1	0	0
15:	1	1	1	0	1	1	1	1	1	0	0	1	0	0
16:	1	1	1	1	0	0	0	0	1	0	0	1	0	0

b)

« » O TVL 1 (ODA) | 18 Var. | 25 R. | Sp. 1

	x1	x2	s4	s3	s2	s1	s0	sn4	sn3	sn2	sn1	sn0	r1	y1	g1	r2	y2	g2
1:	0	-	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
2:	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0
3:	-	-	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0
4:	-	-	0	0	0	1	0	0	0	0	1	1	0	0	1	1	0	0
5:	-	-	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0
6:	-	-	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0
7:	-	-	0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	0
8:	-	-	0	0	1	1	0	0	0	1	1	1	1	0	0	1	0	0
9:	-	-	0	0	1	1	1	0	1	0	0	0	1	0	0	1	0	0
10:	0	-	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0
11:	1	0	0	1	0	0	0	0	1	0	0	1	1	0	0	1	1	0
12:	-	-	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	1
13:	-	-	0	1	0	1	0	0	1	0	1	1	1	0	0	0	0	1
14:	-	-	0	1	0	1	1	0	1	1	0	0	1	0	0	0	0	1
15:	-	-	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	0
16:	-	-	0	1	1	0	1	0	1	1	1	0	1	0	0	1	0	0
17:	-	-	0	1	1	1	0	0	1	1	1	1	1	0	0	1	0	0
18:	-	-	0	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0
19:	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0
20:	-	-	1	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0
21:	-	-	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0
22:	-	-	1	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0
23:	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	1	1	0
24:	-	-	1	1	0	0	0	1	0	0	0	1	1	0	0	0	0	1
25:	-	-	1	1	0	0	1	0	1	0	0	1	1	0	0	0	0	1

Figure 8.3 Behavior of the finite state machine of a traffic light to control the a road work: **a**) simple version with green phases, **b**) extended version with controllable green phases

4. Solutions

Exercise 8.1.

1 There are 16 states and $2^4 = 16$. Hence 4 variables are required to code the states.

2 space 32 1 3 Figure 8.3 a) shows the 16 phases of the finite state machine.

4 vtin 1 3 vtin 1 4 vtin 1 5 sts e8101.sdt
s3 s2 s1 s0. sf3 sf2 sf1 sf0. r1 y1 g1 r2 y2 g2.

Exercise 8.2.

1 The graph of the extended finite state machine is shown in Fig. 8.4. The required outputs for the added states are shown in the table on the right.

state	tl1	tl2
16, 17, 18	g	r
24, 25	r	g

2 There are 21 states and $(2^4 = 16) < 21 < (2^5 = 32)$. Hence 5 variables are required to code the states.

3 space 32 1 4 Figure 8.3 b) shows the 84 phases of the finite state machine.

5 vtin 1 2 vtin 1 3 vtin 1 4 vtin 1 5 sts e8102.sdt
x1 x2. s4 s3 s2 s1 s0. sf4 sf3 sf2 sf1 sf0. r1 y1 g1 r2 y2 g2.

Exercise 8.3.

1 The three flip-flops of the sequential circuit code $2^3 = 8$ states.

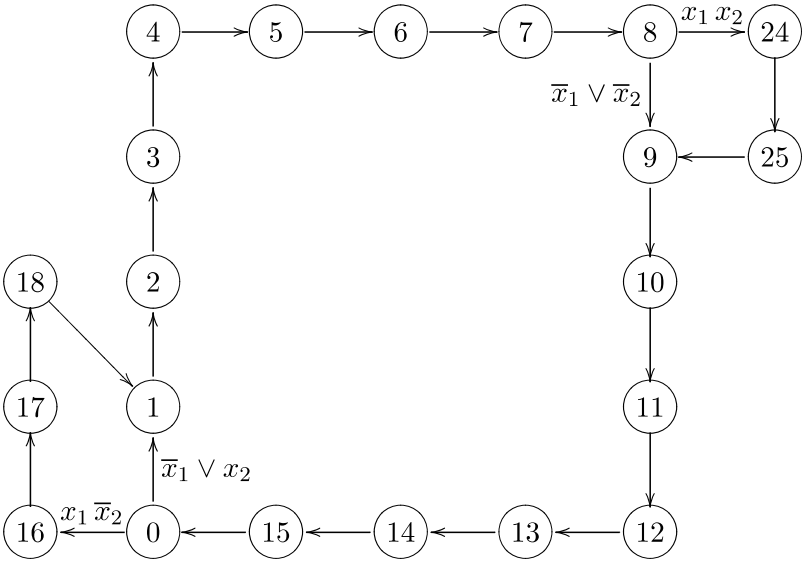


Figure 8.4 Behavior of a synchronous finite-state machine of an extended control for a road work traffic light

```
2  space 64 1
   avar
   x1 x2 s1 s2 s3 sf1 sf2 sf3 y1 y2 nx1 nx2 ns1 ns2 ns3 j1 k1 d2 v2 d3
   g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 g19.

3  sbe 1 1
   nx1=/x1,
   nx2=/x2,
   ns1=/s1,
   ns2=/s2,
   ns3=/s3,
   j1=g2,
   g2=g1&ns3,
   g1=x1#s2,
   k1=g3,
   g3=/g2,
   d2=ns2,
   v2=g4,
   g4=g5+g6,
   g5=nx1&g7,
   g6=x1&g11,
   g7=s1#ns2#g8,
   g8=g9&g10,
   g9=nx2&ns1,
   g10=ns2&ns3,
   g11=g12+g13,
   g12=g10&s1,
   g13=s2&s3,
   d3=g14,
   g14=g15+g16,
   g15=nx1&ns1&ns2,
   g16=x1&s2&g17,
   g17=g18+g19,
   g18=nx2&ns1,
   g19=ns1&ns3,
   sf1=j1&ns1+/k1&s1,
   sf2=/v2&s2+d2&v2,
   sf3=d3,
   y1=/(s1+s3),
   y2=/(s2+s3).

4  vtin 1 2
   x1 x2.
   vtin 1 3
   s1 s2 s3.
   vtin 1 4
   sf1 sf2 sf3.
   vtin 1 5
   y1 y2.
   vtin 1 6
   nx1 nx2 ns1 ns2 ns3 j1 k1 d2 v2 d3
   g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11
   g12 g13 g14 g15 g16 g17 g18 g19.
   sts e8103
```

Exercise 8.4.

1
```
tin 1 10
x1 nx1.
01
10.
tin 1 11
x2 nx2.
01
10.
tin 1 12
s1 ns1.
01
10.
tin 1 13
s2 ns2.
01
10.
tin 1 14
s3 ns3.
01
10.
tin 1 15
g2 j1.
00
11.
tin 1 16
g1 ns3 g2.
0-0, 100
111.
tin 1 17
x1 s2 g1.
000, 011
101, 110.
tin 1 18
g3 k1.
00
11.
tin 1 19
g2 g3.
01
10.
tin 1 20
ns2 d2.
00
11.
tin 1 21
g4 v2.
00
11.
tin 1 22
g5 g6 g4.
000, 011
1-1.
tin 1 23
nx1 g7 g5.
0-0, 100
111.
tin 1 24
x1 g11 g6.
0-0, 100
111.
tin 1 25
s1 ns2 g8 g7.
0000
0011
0101
0110
1001
1010
1100
1111.
tin 1 26
g9 g10 g8.
0-0, 100
111.
tin 1 27
nx2 ns1 g9.
0-0, 100
111.
tin 1 28
ns2 ns3 g10.
0-0, 100
111.
tin 1 29
g12 g13 g11.
000, 011
1-1.
tin 1 33
g10 s1 g12.
0-0, 100
111.
tin 1 31
s2 s3 g13.
0-0, 100
111.
tin 1 32
g14 d3.
00, 11.
tin 1 33
g15 g16 g14.
000, 011
1-1.
tin 1 34
nx1 ns1 ns2
g15.
0–0, 10-0
1100, 1111.
tin 1 35
x1 s2,
g17 g16.
0--0, 10-0,
1100, 1111.
tin 1 36
g18 g19 g17.
000, 011
1-1.
tin 1 37
nx2 ns1 g18.
0-0, 100
111.
tin 1 38
ns1 ns3 g19.
0-0, 100
111.
tin 1 39
j1 k1 s1 sf1.
1-01, -011
0-00, -110.
tin 1 40
d2 v2 s2 sf2.
-000, -011
01-0, 11-1.
tin 1 41
d3 sf3.
00, 11.
tin 1 42
s1 s3 y1.
001, 010
1-0.
tin 1 43
s2 s3 y2.
001, 010
1-0.
```

2 sts e8104

Exercise 8.5.

1 The output of the sequential circuit in Fig. 8.2 depends on the state variables only. Hence, it is a finite-state machine of the Moore-type.

2 $2^{2+3} = 2^5 = 32$ phases exist.

3 Execute the PRP prepared in Exercise 7.1. Fig. 8.5 shows the calculated global list of phases.

4
```
lds e8103
maxk 1 6 7
obbc 7 7
```
Using the variable tuple 6 of the unessential internal variable defined in Exercise 8.3, the 32 phases of the finite-state machine could be expressed by 16 orthogonal ternary vectors as shown in Fig. 8.6.

5 sts e8201

6 The graph of the finite-state machine is shown in Fig. 8.7. There is a cycle of the length 6. If $x_1 = 1$ and $x_2 = 0$, the finite-state machine follows the states (000) → (100) → (110) → (010) → (011) → (001) and returns to (000). In case of $x_1 = 0$ and $x_2 = 0$ the reverse direction of the same cycle is used. If $x_2 = 1$, the state (001) is excluded from the cycle such that the remaining length is equal to 5. The states (111) and (101) cannot be reached. Starting from these states the state (000) is reached.

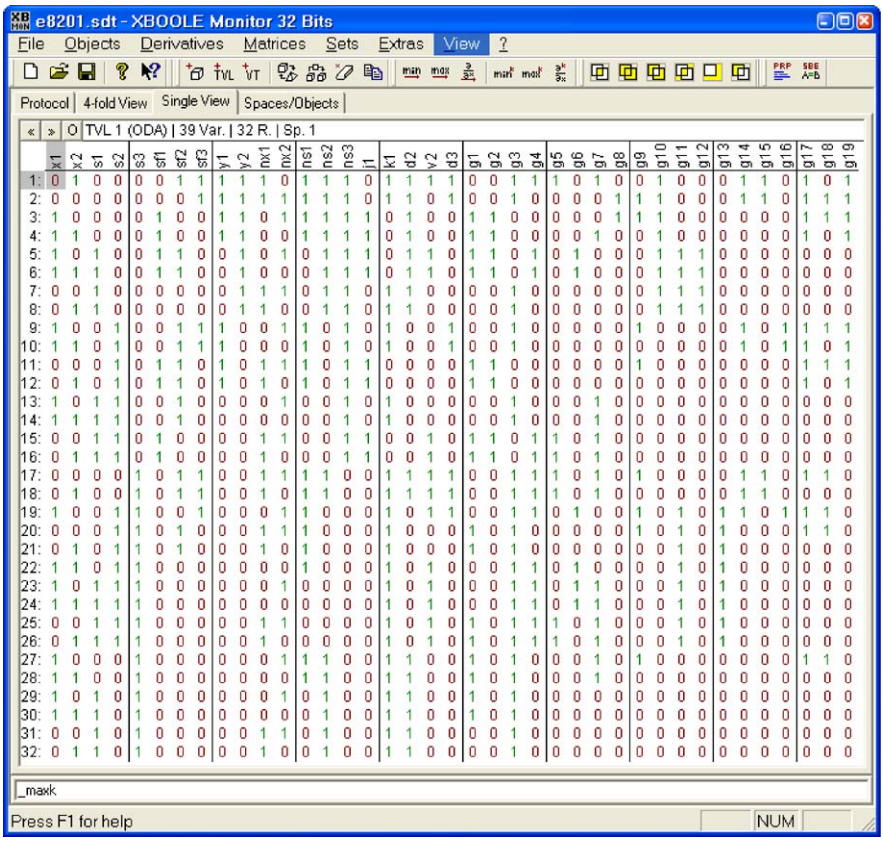


Figure 8.5 Global list of phases that describes the behavior of the sequential circuit given in Fig. 8.2

Exercise 8.6.

1 lds e8201

2 Execute the PRP prepared in Exercise 8.4.

3
```
isc 10 11 50    isc 50 18 50    isc 50 25 50    isc 50 32 50    isc 50 39 50
isc 50 12 50    isc 50 19 50    isc 50 26 50    isc 50 33 50    isc 50 40 50
isc 50 13 50    isc 50 20 50    isc 50 27 50    isc 50 34 50    isc 50 41 50
isc 50 14 50    isc 50 21 50    isc 50 28 50    isc 50 35 50    isc 50 42 50
isc 50 15 50    isc 50 22 50    isc 50 29 50    isc 50 36 50    isc 50 43 50
isc 50 16 50    isc 50 23 50    isc 50 30 50    isc 50 37 50    syd 50 1 51
isc 50 17 50    isc 50 24 50    isc 50 31 50    isc 50 38 50
```

4 syd 50 1 51 — The empty TVL 51 verifies that both structural models describe the same behavior of a finite-state machine.

Exercise 8.7.

1 lds e8201.sdt

« » O K TVL 7 (ODA) | 10 Var. | 16 R.

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	0	1	0	0	0	0	1	1	1	1
2:	0	0	0	0	0	0	0	1	1	1
3:	1	-	0	0	0	1	0	0	1	1
4:	1	-	1	0	0	1	1	0	0	1
5:	0	-	1	0	0	0	0	0	0	1
6:	1	-	0	1	0	0	1	1	1	0
7:	0	-	0	1	0	1	1	0	1	0
8:	1	-	1	1	0	0	1	0	0	0
9:	0	-	1	1	0	1	0	0	0	0
10:	0	-	0	0	1	0	1	1	0	0
11:	1	0	0	1	1	0	0	1	0	0
12:	0	-	0	1	1	0	1	0	0	0
13:	1	1	-	1	1	0	0	0	0	0
14:	1	0	1	1	1	0	0	0	0	0
15:	0	-	1	-	1	0	0	0	0	0
16:	1	-	-	0	1	0	0	0	0	0

Figure 8.6 Global list of phases that describes the behavior of the sequential circuit given in Fig. 8.2 restricted to the essential variables and minimized

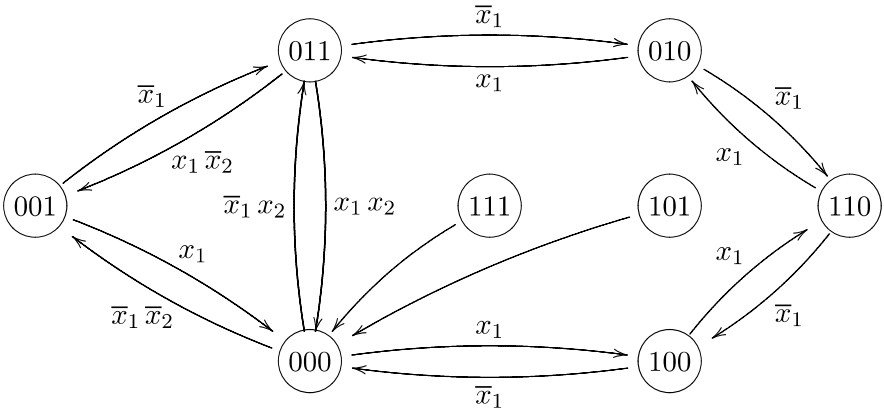


Figure 8.7 Behavior of a finite-state machine of the sequential circuit given in Fig. 8.2 – the states are labeled by $(s_1, s_2, s_3) - y_1 = 1$ in the states (000) and (010), $y_2 = 1$ in the states (000) and (100)

2	tin 1 10	tin 1 12	tin 1 14	tin 1 16
	x1 x2.	x1 x2.	x1 x2.	x1 x2.
	00.	01.	10.	11.
	isc 7 10 11	isc 7 12 13	isc 7 14 15	isc 7 16 17
	obb 11 11	obb 13 13	obb 15 15	obb 17 17

The partial behaviors for all four possible input pattern are shown in Fig. 8.8. More clearly the different cycle lengths of 6 (5) for $x_2 = 0$ ($x_2 = 1$) and the different directions in the cycles for $x_1 = 0$ ($x_1 = 1$) are visible.

« » O K TVL 11 (ODA) | 10 Var. | 7 R. |

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	0	0	0	0	0	0	0	1	1	1
2:	0	0	1	0	0	0	0	0	0	1
3:	0	0	0	1	0	1	1	0	1	0
4:	0	0	1	1	0	1	0	0	0	0
5:	0	0	0	0	1	0	1	1	0	0
6:	0	0	0	1	1	0	1	0	0	0
7:	0	0	1	-	1	0	0	0	0	0

« » O K TVL 13 (ODA) | 10 Var. | 7 R

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	0	1	0	0	0	0	1	1	1	1
2:	0	1	1	0	0	0	0	0	0	1
3:	0	1	0	1	0	1	1	0	1	0
4:	0	1	1	1	0	1	0	0	0	0
5:	0	1	0	0	1	0	1	1	0	0
6:	0	1	0	1	1	0	1	0	0	0
7:	0	1	1	-	1	0	0	0	0	0

« » O K TVL 15 (ODA) | 10 Var. | 7 R. |

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	1	0	0	0	0	1	0	0	1	1
2:	1	0	1	0	0	1	1	0	0	1
3:	1	0	0	1	0	0	1	1	1	0
4:	1	0	1	1	0	0	1	0	0	0
5:	1	0	0	1	1	0	0	1	0	0
6:	1	0	1	1	1	0	0	0	0	0
7:	1	0	-	0	1	0	0	0	0	0

« » O K TVL 17 (ODA) | 10 Var. | 5 R

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	1	1	0	0	0	1	0	0	1	1
2:	1	1	1	0	0	1	1	0	0	1
3:	1	1	0	1	0	0	1	1	1	0
4:	1	1	1	1	0	0	1	0	0	0
5:	1	1	-	-	1	0	0	0	0	0

Figure 8.8 Global list of phases that describes the behavior of the sequential circuit given in Fig. 8.2 for each fixed input pattern

3	tin 1 20 s1 s2 s3. 000. isc 7 20 21 obb 21 21	tin 1 22 s1 s2 s3. 100. isc 7 22 23 obb 23 23	tin 1 24 s1 s2 s3. 101. isc 7 24 25 obb 25 25	TVL 21 indicates 3 reachable states from (000), in detail (011), (001), and (100). TVL 23 indicates 2 reachable states from (100), in detail (110), and (000). TVL 25 indicates that only the state (000) is reachable from (101).
4	tin 1 30 sf1 sf2 sf3. 000. isc 7 30 31 obb 31 31	tin 1 32 sf1 sf2 sf3. 100. isc 7 32 33 obb 33 33	tin 1 34 sf1 sf2 sf3. 101. isc 7 34 35 obb 35 35	TVL 31 indicates 5 states from which the state (000) is reachable, in detail (100), (011), (111), (101) and (001). TVL 33 indicates two states from which the state (100) is reachable, in detail (000), and (110). TVL 35 indicates that there is a state from which the state (101) is reachable.
5	isc 7 34 35 obb 35 35 maxk 7 2 40 maxk 40 3 40	maxk 40 5 40 cpl 40 41 obb 41 41 sts 8203		TVL 41 indicates that 2 states are not reachable from any state controlled by any input pattern. These two states are (111), and (101).

Exercise 8.8.

1	lds e8201.sdt	
2	maxk 7 4 10 maxk 10 5 10 cpl 10 11	The empty TVL 11 confirms that the analyzed finite-state machine is realizable.

3 _maxk 7 5 20
_maxk 20 ⟨$sf2$ $sf3$⟩ 21
_derk 21 ⟨$sf1$⟩ 22
cpl 22 22

_maxk 20 ⟨$sf1$ $sf3$⟩ 23
_derk 23 ⟨$sf2$⟩ 24
cpl 24 24

_maxk 20 ⟨$sf1$ $sf2$⟩ 25
_derk 25 ⟨$sf3$⟩ 26
cpl 26 26

The empty TVLs 22, 24, and 26 confirm that all three memory functions of the analyzed finite-state machine are uniquely defined.

4 maxk 7 4 30

_maxk 30 ⟨$y2$⟩ 31
_derk 31 ⟨$y1$⟩ 32
cpl 32 32

_maxk 30 ⟨$y1$⟩ 33
_derk 33 ⟨$y2$⟩ 34
cpl 34 34

The empty TVLs 32, 34 confirm that both result functions of the analyzed finite-state machine are uniquely defined.

Exercise 8.9.

1 lds e8101.sdt

2 The finite-state machine of the simple traffic light does not depend on any input variables. Hence, it is a finite-state machine of the Moore-type, more in detail an autonomous one.

3 maxk 1 4 10
maxk 10 5 10
cpl 10 11

The empty TVL 11 confirms that the analyzed finite-state machine is realizable. There are no missing phases.

Exercise 8.10.

1 lds e8102.sdt

2 The finite-state machine of the extended traffic light depends on 2 input variables. Hence, it can be a finite-state machine of the Mealy-type or of the Moore-type. It is necessary to check whether at least one of the outputs depends for any state on the inputs. This can be analyzed using the Δ-operator of the Boolean Differential Calculus.

maxk 1 4 10

mink 10 2 11
maxk 10 2 12
syd 11 12 13

The empty TVL 13 confirms that no output depends on any input. Hence, it is a finite-state machine of the Moore-type.

3 maxk 1 4 20
maxk 20 5 20
cpl 20 21

The TVL 21 is not empty. Hence the given finite-state machine of the extended traffic light is not realizable. There are 44 missing phases. The reason is the extension by one state variable that allows 16 additional states. Only 5 of them are used. The behavior of the remaining states must be defined in order to realize this finite-state machine by a sequential circuit.

Exercise 8.11.

1 More than one phase is allowed for each input pattern in the states (s_1, s_2, s_3) equal to (111) and (101).

2 space 32 1

a)

«	»	O	T	TVL 16 (ODA) | 5 Var. |								
0	0	0	0	0	0	1	0	0	1			
0	1	0	0	0	0	0	0	0	0			
1	1	0	0	0	0	0	0	0	0			
1	0	1	0	0	1	0	0	0	0			
s2	s3	0	1	1	0	0	1	1	0	s1		
		0	0	1	1	1	1	0	0	x2		
		0	0	0	0	1	1	1	1	x1		

b)

«	»	O	T	TVL 17 (ODA) | 5 Var. |								
0	0	1	0	0	1	0	0	0	0			
0	1	1	0	0	1	1	0	0	1			
1	1	1	0	0	1	1	0	0	1			
1	0	0	0	0	0	1	0	0	1			
s2	s3	0	1	1	0	0	1	1	0	s1		
		0	0	1	1	1	1	0	0	x2		
		0	0	0	0	1	1	1	1	x1		

c) $j_1 = (x_1 \oplus s_2) \wedge \overline{s}_3$

d)

«	»	O	T	TVL 26 (ODA) | 5 Var. |								
0	0	0	1	1	0	0	0	0	0			
0	1	0	0	0	0	0	0	0	0			
1	1	0	0	0	0	0	0	0	0			
1	0	0	0	0	0	0	1	1	0			
s2	s3	0	1	1	0	0	1	1	0	s1		
		0	0	1	1	1	1	0	0	x2		
		0	0	0	0	1	1	1	1	x1		

e)

«	»	O	T	TVL 27 (ODA) | 5 Var. |								
0	0	0	0	0	0	0	1	1	0			
0	1	0	0	0	0	0	0	0	0			
1	1	0	0	0	0	0	0	0	0			
1	0	0	1	1	0	0	0	0	0			
s2	s3	0	1	1	0	0	1	1	0	s1		
		0	0	1	1	1	1	0	0	x2		
		0	0	0	0	1	1	1	1	x1		

f) $k_1 = \overline{(x_1 \oplus s_2) \wedge \overline{s}_3} = \overline{j}_1$

Figure 8.9 Karnaugh-maps of the mark functions to control the JK-flip-flop: **a**) j_{1q} as object 16, **b**) j_{1r} as object 17, **c**) the selected function j_1, **d**) k_{1q} as object 26, **e**) k_{1r} as object 27, **f**) the selected function k_1

```
3 tin 1 1
  x1 x2 s1 s2 s3
  sf1 sf2 sf3 y1 y2.
  1-00010011
  1-10011001
  1-11001000
  1-01001110
  1101100000
  1001100100
  1-00100000
  0000000111
  0100001111
  0-00101100
  0-01101000
  0-01011010
  0-11010000
  0-10000001
  --1-1--0--
  --1-10-1--.
```

```
4 vtin 1 2
  x1 x2.
  vtin 1 3
  s1 s2 s3.
  vtin 1 4
  sf1 sf2 sf3.
  vtin 1 5
  y1 y2.
```

```
5 sts e8301
  After
  execution of
  the PRP the
  TVL system
  is stored as
  file "e8301.sdt".
```

Exercise 8.12.

```
1 lds e8301.sdt
```

```
2 tin 1 10
  j1 k1 s1 sf1.
  1-01, -011
  0-00, -110.
  isc 1 10 11
```

```
3 maxk 11 4 12
  maxk 12 5 12

4 _maxk 12 ⟨k1⟩ 13
  obbc 13 13
```

```
5 sbe 1 14
  j1 = 1.
  mink 13 14 15
  obb 15 15
  don't-care
  function j1φ
```

```
6 isc 13 14 16
  maxk 16 14 16
  dif 16 15 16
  obb 16 16
  j1q see
  Fig. 8.9 a)
```

a)

«	»	O	T	TVL 36 (ODA) | 5 Var. |							
0	0	0	0	0	1	0	1	1	0		
0	1	1	0	0	1	0	0	0	0		
1	1	0	0	0	0	0	0	0	0		
1	0	0	0	0	0	0	0	0	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

b)

«	»	O	T	TVL 37 (ODA) | 5 Var. |							
0	0	0	0	0	0	0	0	0	0		
0	1	0	0	0	0	0	0	0	0		
1	1	0	0	0	0	1	0	0	1		
1	0	0	1	1	0	0	0	0	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

c) $d_2 = \overline{s}_2$

d)

«	»	O	T	TVL 46 (ODA) | 5 Var. |							
0	0	0	0	0	1	0	1	1	0		
0	1	1	0	0	1	0	0	0	0		
1	1	0	0	0	0	1	0	0	1		
1	0	0	1	1	0	0	0	0	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

e)

«	»	O	T	TVL 47 (ODA) | 5 Var. |							
0	0	1	1	1	0	1	0	0	1		
0	1	0	0	0	0	1	0	0	1		
1	1	1	0	0	1	0	0	0	0		
1	0	1	0	0	1	1	1	1	1		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

f) $v_2 = s_3 \cdot (\overline{x}_1 \oplus s_2) \vee s_1 \cdot (x_1 \oplus s_2) \vee \overline{x}_1 \cdot x_2 \cdot \overline{s}_1 \cdot \overline{s}_2$

Figure 8.10 Karnaugh-maps of the mark functions to control the *DV*-flip-flop: **a**) d_{2q} as object 36, **b**) d_{2r} as object 37, **c**) the selected function d_2, **d**) v_{2q} as object 46, **e**) v_{2r} as object 47, **f**) the selected function v_2

```
7  cpl 14 14
   isc 13 14 17
   maxk 17 14 17
   dif 17 15 17
   obb 17 17
   j1r see
   Fig. 8.9 b)

8  selected j1 see
     Fig. 8.9 c)
     sbe 1 18
     j1 = (x1#s2)&/s3.

9  isc 11 18 21

10 maxk 21 4 22
     maxk 22 5 22

11 _maxk 22 ⟨j1⟩ 23
     obbc 23 23

12 sbe 1 24
     k1 = 1.
     mink 23 24 25
     obb 25 25
     don't-care
     function k1φ

13 isc 23 24 26
   maxk 26 24 26
   dif 26 25 26
   obb 26 26
   k1q see
   Fig. 8.9 d)

14 cpl 24 24
     isc 23 24 27
     maxk 27 24 27
     dif 27 25 27
     obb 27 27
     k1r see
     Fig. 8.9 e)

15 selected k1 see
     Fig. 8.9 f)
     sbe 1 28
     k1 = /j1.

16 isc 21 28 29

17 _maxk 29 ⟨j1 k1⟩ 6
     obbc 6 6
18 syd 1 6 100
     The TVL 100 shows
     that 64 of the 192
     non-deterministic
     phases are removed.

19 sts e8302.sdt
```

Exercise 8.13.

```
1  e8302.sdt
```

a)

«	»	O	T	TVL 56 (ODA) | 5 Var. |							
0	0	1	0	0	1	0	0	0	0		
0	1	1	0	0	1	0	0	0	0		
1	1	0	0	0	0	0	0	0	1		
1	0	0	0	0	0	1	0	0	1		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

b)

«	»	O	T	TVL 57 (ODA) | 5 Var. |							
0	0	0	1	1	0	1	1	1	1		
0	1	0	0	0	0	1	0	0	1		
1	1	1	0	0	1	1	0	0	0		
1	0	1	1	1	1	0	1	1	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

c) $d_3 = \overline{x}_1\,\overline{s}_1\,\overline{s}_2 \vee x_1\,\overline{s}_1\,s_2 \cdot (\overline{x}_2 \vee \overline{s}_3)$

Figure 8.11 Karnaugh-maps of the mark functions to control the D-flip-flop: **a**) d_{3q} as object 56, **b**) d_{3r} as object 57, **c**) the selected function d_3

```
2  tin 1 30
   d2 v2 s2 sf2.
   -000, -011
   01-0, 11-1.
   isc 6 30 31

3  maxk 31 4 32
     maxk 32 5 32

4  _maxk 32 ⟨v2⟩ 33
     obbc 33 33

5  sbe 1 34
     d2 = 1.
     mink 33 34 35
     obb 35 35
     don't-care
     function d2φ

6  isc 33 34 36
     maxk 36 34 36
     dif 36 35 36
     obb 36 36
     d2q see
     Fig. 8.10 a)

7  cpl 34 34
   isc 33 34 37
   maxk 37 34 37
   dif 37 35 37
   obb 37 37
   d2r see
   Fig. 8.10 b)

8  selected d2 see
     Fig. 8.10 c)
     sbe 1 38
     d2 = /s2.

9  isc 31 38 41

10 maxk 41 4 42
     maxk 42 5 42

11 _maxk 42 ⟨d2⟩ 43
     obbc 43 43

12 sbe 1 44
     v2 = 1.
     mink 43 44 45
     obb 45 45
     don't-care
     function v2φ

13 isc 43 44 46
   maxk 46 44 46
   dif 46 45 46
   obb 46 46
   v2q see
   Fig. 8.10 d)

14 cpl 44 44
     isc 43 44 47
     maxk 47 44 47
     dif 47 45 47
     obb 47 47
     d2r see
     Fig. 8.10 e)

15 selected v2 see
     Fig. 8.10 f)
     sbe 1 48
     v2=
     s3&(/x1#s2)
     +s1&(x1#s2)
     +/x1&x2&/s1&/s2.

16 isc 41 48 49

17 _maxk 49 ⟨d2 v2⟩ 7
     obbc 7 7

18 syd 1 7 101

   The TVL 101 shows that 128 of the 192
   non-deterministic phases are removed.

19 sts e8303.sdt
```

Exercise 8.14.

```
1  e8303.sdt
```

a)

«	»	O	T	TVL 66 (ODA) | 5 Var. |							
0	0	1	0	0	1	1	0	0	1		
0	1	0	0	0	0	0	0	0	0		
1	1	0	0	0	0	0	0	0	0		
1	0	1	0	0	1	1	0	0	1		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

b)

«	»	O	T	TVL 67 (ODA) | 5 Var. |							
0	0	0	1	1	0	0	1	1	0		
0	1	1	0	0	1	1	0	0	1		
1	1	1	0	0	1	1	0	0	1		
1	0	0	1	1	0	0	1	1	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

c) $y_1 = \overline{s}_1 \wedge \overline{s}_3$

d)

«	»	O	T	TVL 76 (ODA) | 5 Var. |							
0	0	1	1	1	1	1	1	1	1		
0	1	0	0	0	0	0	0	0	0		
1	1	0	0	0	0	0	0	0	0		
1	0	0	0	0	0	0	0	0	0		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

e)

«	»	O	T	TVL 77 (ODA) | 5 Var. |							
0	0	0	0	0	0	0	0	0	0		
0	1	1	0	0	1	1	0	0	1		
1	1	1	0	0	1	1	0	0	1		
1	0	1	1	1	1	1	1	1	1		
s2	s3	0	1	1	0	0	1	1	0	s1	
		0	0	1	1	1	1	0	0	x2	
		0	0	0	0	1	1	1	1	x1	

f) $y_2 = \overline{s}_2 \wedge \overline{s}_3$

Figure 8.12 Karnaugh-maps of the mark functions of the outputs y_1 and y_2: **a**) y_{1q} as object 66, **b**) y_{1r} as object 67, **c**) the selected function y_1, **d**) y_{2q} as object 76, **e**) y_{2r} as object 77, **f**) the selected function y_2

2 tin 1 50
d3 sf3.
00
11.
isc 7 50 51

3 maxk 51 4 52
maxk 52 5 52
obbc 52 53

4 sbe 1 54
d3 = 1.
mink 53 54 55
obb 55 55
don't-care function $d_{3\varphi}$

5 isc 53 54 56
maxk 56 54 56
dif 56 55 56
obb 56 56
d_{3q} see Fig. 8.11 a)

6 cpl 54 54
isc 53 54 57
maxk 57 54 57
dif 57 55 57
obb 57 57
d_{3r} see Fig. 8.11 b)

7 selected d_3 see Fig. 8.11 c)
sbe 1 58
d3=
/x1&/s1&/s2
+x1&/s1&s2
&(/x2+/s3).

8 isc 51 58 59

9 _maxk 59 ⟨d3⟩ 8
obbc 8 8

10 syd 1 8 102
The TVL 102 shows that 160 of the 192 non-deterministic phases are removed.

11 sts e8304.sdt

Exercise 8.15.

1 e8304.sdt

a)

« » O K TVL 1 (ODA) | 10 Var. | 16 R.

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	1	-	0	0	0	1	0	0	1	1
2:	1	-	1	0	0	1	1	0	0	1
3:	1	-	1	1	0	0	1	0	0	0
4:	1	-	0	1	0	0	1	1	1	0
5:	1	1	0	1	1	0	0	0	0	0
6:	1	0	0	1	1	0	0	1	0	0
7:	1	-	0	0	1	0	0	0	0	0
8:	0	0	0	0	0	0	0	1	1	1
9:	0	1	0	0	0	0	1	1	1	1
10:	0	-	0	0	1	0	1	1	0	0
11:	0	-	0	1	1	0	1	0	0	0
12:	0	-	0	1	0	1	1	0	1	0
13:	0	-	1	1	0	1	0	0	0	0
14:	0	-	1	0	0	0	0	0	0	1
15:	-	-	1	-	1	-	-	0	-	-
16:	-	-	1	-	1	0	-	1	-	-

b)

« » O K TVL 9 (ODA) | 10 Var. | 16 R.

	x1	x2	s1	s2	s3	sf1	sf2	sf3	y1	y2
1:	1	-	0	0	0	1	0	0	1	1
2:	1	-	1	0	0	1	1	0	0	1
3:	1	-	1	1	0	0	1	0	0	0
4:	1	-	0	1	0	0	1	1	1	0
5:	1	1	0	-	1	0	0	0	0	0
6:	1	0	0	1	1	0	0	1	0	0
7:	1	0	0	0	1	0	0	0	0	0
8:	0	0	0	0	0	0	0	1	1	1
9:	0	1	0	0	0	0	1	1	1	1
10:	0	-	0	0	1	0	1	1	0	0
11:	0	-	0	1	1	0	1	0	0	0
12:	0	-	0	1	0	1	1	0	1	0
13:	0	-	1	1	0	1	0	0	0	0
14:	0	-	1	0	0	0	0	0	0	1
15:	-	-	1	1	1	0	0	0	0	0
16:	-	-	1	0	1	0	1	0	0	0

Figure 8.13 Behavior of a finite-state machine: **a**) given non-deterministic global list of phases as object number 1, **b**) deterministic global list of phases as object number 9 of the designed sequential circuit

2 maxk 8 4 62

3 _maxk 62 ⟨$y2$⟩ 63

4 sbe 1 64
y1=1.

mink 63 64 65
obb 65 65
don't-care
function $y_{1\varphi}$

5 isc 63 64 66
maxk 66 64 66

dif 66 65 66
obb 66 66
y_{1q} see
Fig. 8.12 a)

6 cpl 64 64
isc 63 64 67

maxk 67 64 67
dif 67 65 67
obb 67 67
y_{1r} see
Fig. 8.12 b)

7 selected y_1 see
Fig. 8.12 c)
sbe 1 68
y1=/s1&/s3.

8 isc 8 68 71

9 maxk 71 4 72

10 _maxk 72 ⟨$y1$⟩ 73
obbc 73 73

11 sbe 1 74
y2=1.
mink 73 74 75
obb 75 75
don't-care
function $y_{2\varphi}$

12 isc 73 74 76
maxk 76 74 76
dif 76 75 76
obb 76 76
y_{2q} see
Fig. 8.12 d)

13 cpl 74 74
isc 73 74 77
maxk 77 74 77
dif 77 75 77
obb 77 77
y_{2r} see
Fig. 8.12 e)

14 selected y_2 see
Fig. 8.12 f)
sbe 1 78
y2 = /s2&/s3.

15 isc 71 78 9
obbc 9 9

16 syd 1 9 103
The TVL 103 shows that 184 of the 192 non-deterministic phases are removed. The remaining 8 phases describe the deterministic behavior in the states (111) and (101) for each of the four input patterns. Hence, a deterministic finite-state machine is designed.

17 sts e8305.sdt

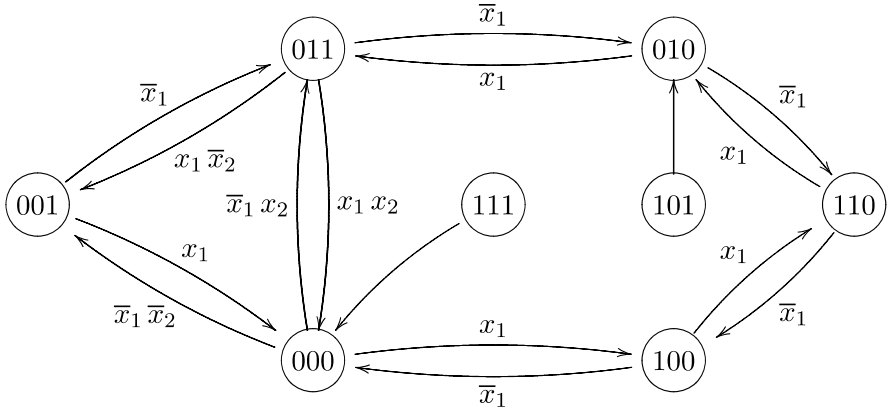


Figure 8.14 Behavior of a finite-state machine of the sequential circuit designed in Exercises 8.12 ... 8.15 – the states are labeled by (s_1, s_2, s_3) – $y_1 = 1$ in the states (000) and (010), $y_2 = 1$ in the states (000) and (100)

Exercise 8.16.

1 e8305.sdt

2 Figure 8.13 a) shows the global lists of phases of the given non-deterministic finite-state machine of object number 1, and Fig. 8.13 b) shows the global lists of phases the designed deterministic finite-state machine of object number 9, respectively. The rows 15 and 16 show how the non-deterministic behavior in the states (101) and (111) was specified during the design process into a fixed deterministic behavior.

3 Figure 8.14 depicts the graph of the designed finite-state machine. In comparison with the graph of Fig. 8.7 on page 215 of the basic finite machine only the edge starting from the state (101) was redirected from the state (000) to the state (010) such that the working cycle is reached from this inessential state in one time step.

4 Figure 8.15 shows the structure of the sequential circuit calculated in Exercises 8.12 ... 8.15. The number of inputs of the gates is restricted to three.

5 dif 9 1 80
The empty TVL 80 confirms that the behavior of the structure of the sequential circuit given in object number 9 is covered by the allowed behavior defined in Exercise 8.11 and given in object number 1.

_maxk 9 ⟨$sf2$ $sf3$ $y1$ $y2$⟩ 81
_derk 81 ⟨$sf1$⟩ 82
cpl 82 82

_maxk 9 ⟨$sf1$ $sf3$ $y1$ $y2$⟩ 83
_derk 83 ⟨$sf2$⟩ 84
cpl 84 84

_maxk 9 ⟨$sf1$ $sf2$ $y1$ $y2$⟩ 85
_derk 85 ⟨$sf3$⟩ 86
cpl 86 86

The empty TVLs 82, 84, and 86 confirm that the memory functions sf_1, sf_2, and sf_3 are uniquely defined.

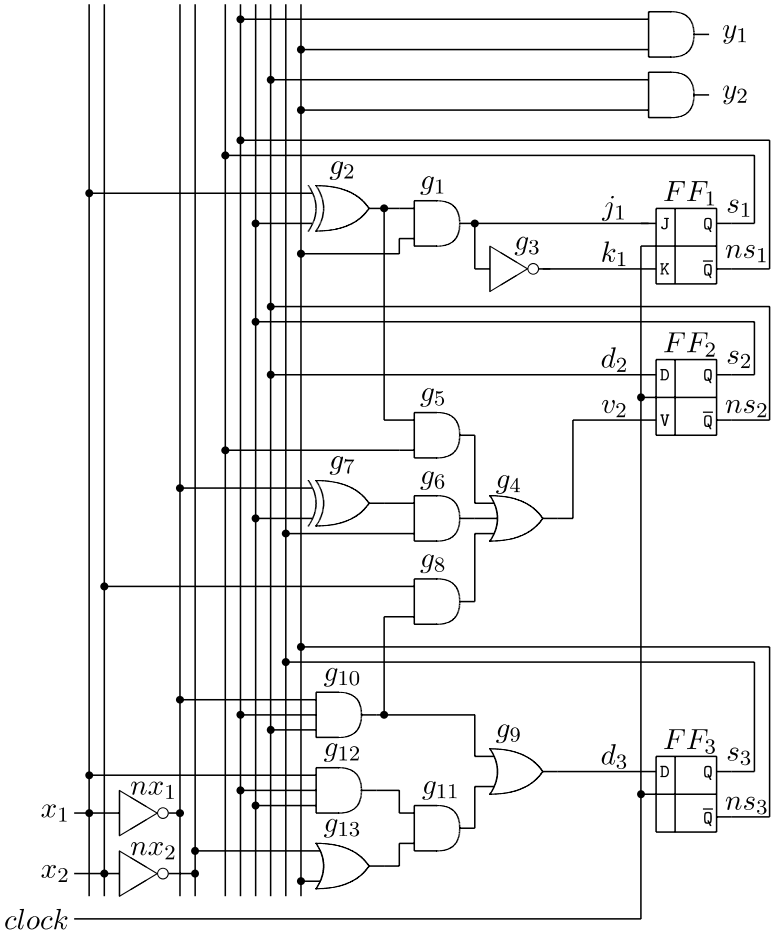


Figure 8.15 Structure of the designed finite-state machine using three types of flip-flops and AND-, OR-, and EXOR-gates restricted to three inputs

_maxk 9 ⟨$sf1$ $sf2$ $sf3$ $y2$⟩ 87
_derk 87 ⟨$y1$⟩ 88
cpl 88 88

_maxk 9 ⟨$sf1$ $sf2$ $sf3$ $y1$⟩ 89
_derk 89 ⟨$y2$⟩ 90
cpl 90 90

The empty TVLs 88 and 90 confirm that the output functions y_1 and y_2 are uniquely defined.

6 Figure 8.15 depicts the circuit structure of the designed finite-state machine which differs in one inessential phase from the circuit structure given in Fig. 8.2. The circuit structures to control the JK-flip-flop $FF1$ are identical. This indicates the high quality of both designs. Nevertheless the number of gates in the designed circuit structure was reduced by 6 gates from 23 gates to 17 gates. The longest path between the inputs of the circuit and the inputs of the flip-flops is reduced

by two gates from 6 gates to 4 gate. Hence, the clock frequency of the designed circuit can be 150 percent of the given sequential circuit.

Exercise 8.17.

1 lds e8101.sdt

2
```
sbe 1 6
j0=1,
k0=1,
j1=s0,
k1=s0,
j2=s0&s1,
k2=s0&s1,
j3=s0&s1&s2,
k3=s0&s1&s2,
r1=(s2+s3)#(/s0&/s1&/s3),
y1=/s0&/s1&/s3,
g1=(s0+s1)&/s2&/s3,
r2=(s2+/s3)#(/s0&/s1&s3),
y2=/s0&/s1&s3,
g2=(s0+s1)&/s2&s3,
sf0=j0&/s0+/k0&s0,
sf1=j1&/s1+/k1&s1,
sf2=j2&/s2+/k2&s2,
sf3=j3&/s3+/k3&s3.
```

3 _maxk 6 ⟨*j*0 *k*0 *j*1 *k*1 *j*2 *k*2 *j*3 *k*3⟩ 7
syd 1 7 8
The empty TVL 8 confirms that the designed sequential circuit realizes the behavior of the simple control for a road work traffic light given in Fig. 8.1.

Exercise 8.18.

1 lds e8102.sdt

2
```
maxk 1 2 6
maxk 6 4 6
maxk 6 5 6
cco 6 3 4 7
cpl 6 8
isc 7 8 9
obbc 9 9
tin 1 10
r1 y1 g1 r2 y2 g2.
000000.
isc 9 10 11
uni 1 11 12
obbc 12 12
```
TVL 6 includes all states used so far. TVL 9 includes the phase (**s**, **sf**) which completes the finite-state machine. TVL 11 extends these phases by the allowed output values. TVL 12 is the realizable nondeterministic finite-state machine to design.

3
```
maxk 12 4 13
maxk 13 5 13
cpl 13 13
```
The empty TVL 13 confirms that the completed finite-state machine is realizable.

4
```
sbe 1 20
d0=/s0,
v0=/s1&/s2&/s3&s4
+((/s0+/s4)
#(x1&/s0&/s1&/s2&/s4&(/x2#s3))),
d1=s0&/s1,
v1=s0&/s4+/s2&/s3&s4&(s0#s1),
d2=/s2,
v2=s0&s1,
d3=/s3,
v3=s0&s1&s2,
d4=/s0&/s1&/s2,
v4=s1+s2+s3&s4
+x1&/s2&/s4&(x2#/s3),
r1=s3&/s4+/s3&/s4
&(s2#(/s0&/s1))+/s1&/s2&s3,
y1=/s0&/s1&/s3&/s4,
g1=/s2&/s3&(/s4&(s0+s1)
+s4&(/s0+/s1)),
r2=/s3&/s4+/s4&
(s2#(/s0&/s1))+/s2&/s3&(/s0+/s1),
y2=/s0&/s1&s3&/s4,
g2=/s2&s3&(/s1&s4+/s4&(s0+s1)),
sf0=/v0&s0+d0&v0,
sf1=/v1&s1+d1&v1,
sf2=/v2&s2+d2&v2,
sf3=/v3&s3+d3&v3,
sf4=/v4&s4+d4&v4.
```

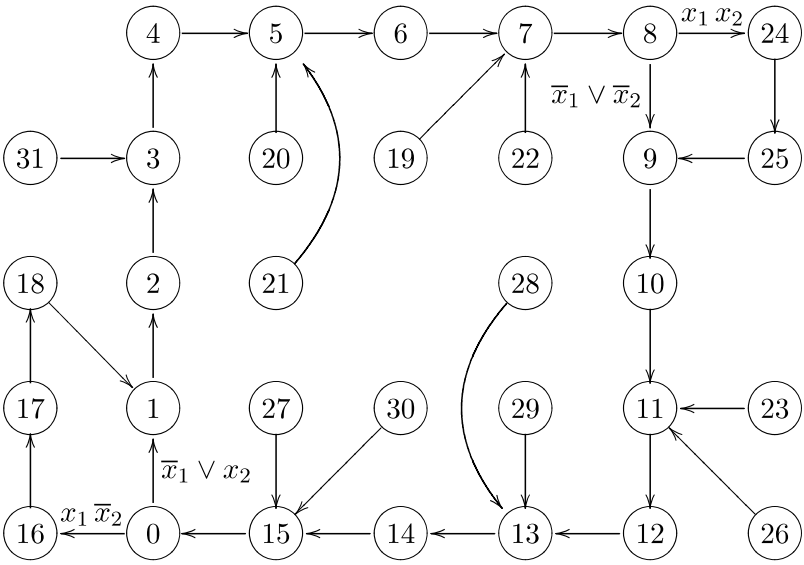


Figure 8.16 Behavior of the realized synchronous finite-state machine of an extended control for a road work traffic light

5 _maxk 20 ⟨*d*0 *v*0 *d*1 *v*1 *d*2 *v*2 *d*3 *v*3 *d*4 *v*4⟩ 21 maxk 21 4 23
obbc 21 21 maxk 23 5 23
dif 21 12 22 cpl 23 23

The empty TVL 22 confirms that the circuit described by the system of logic equations in object number 20 behaves as allowed in the behavioral description 12. The empty TVL 23 confirms that this finite-state machine is realizable.

6 Figure 8.16 shows the graph of the realized finite-state machine of an extended control for a road work traffic light. It can be seen that the given behavior of Fig. 8.4 is embedded and a unique deterministic behavior was selected by the designed sequential circuit.

References

[1] D. Bochmann, F. Dresig, and B. Steinbach. A new decomposition method for multilevel circuit design. In *European Conference on Design Automation*, pages 374–377, Amsterdam, Holland, 1991.

[2] D. Bochmann and B. Steinbach. *Logikentwurf mit XBOOLE*. Verlag Technik, Berlin, 1991.

[3] E. Böhl. *Ein Beitrag zur Theorie dynamischer Fehlererscheinungen in kombinatorischen und sequentiellen Schlatnetzwerken*. PhD thesis, University of Technology, Karl-Marx-Stadt (Chemnitz), Germany, 1975

[4] F.M. Brown. *Boolean Reasoning: The Logic of Boolean Equations*. Kluwer Academic Publishers, Boston, 1990

[5] R.E. Bryant. Graph-based algorithms for boolean function manipulation. C-35, 1986

[6] M. Davio, J.-P. Deschamps, and A. Thayse. *Discrete and Switching Functions*. McGraw-Hill International, New York, 1978

[7] D.L. Dietmeyer. *Logic Design of Digital Systems*. Allyn and Bacon Inc., Boston, 2 edition, 1978

[8] F. Dresig, N. Kümmling, B. Steinbach, and J. Wazel. *Programmmieren mit XBOOLE*. Number 5 in Wissenschaftliche Schriftenreihe. Technische Universität, Chemnitz, Germany, 1992

[9] G.P. Gavrilov and A.A. Sapozhenko. *Problems and Exercises in Discrete Mathematics*. Springer, Berlin, June 30, 1996

[10] D.H. Green. *Modern Logic Design*. Addison-Wesley Publishing Company, New York, 1986

[11] G.D. Hachtel and F. Somenzi. *Logic Synthesis and Verification Algorithms*. Kluwer Academic Publishers, Boston, 1996

[12] M.A. Harrison. *Introduction to Switching and Automata Theory*. McGraw-Hill, New York, 1965

[13] Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, 1970

[14] G.D. Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, Inc., New York, 1994

[15] R.E. Miller. *Switching Theory*. Wiley, New York, 1965

[16] A. Mishchenko, B. Steinbach, and M. Perkowski. An algorithm for bi-decomposition of logic functions. In *Proceedings of the 38th Design Automation Conference*, pages 103–108, Las Vegas (Nevada) USA, 2001

[17] S. Muroga. *Logic Design and Switching Theory*. Wiley, New York, 1979

[18] C. Posthoff and B. Steinbach. *Logic Functions and Equations – Binary Models for Computer Science*. Springer, Dordrecht, The Netherlands, 2004

[19] T. Sasao. *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, Norwell, MA, 1999

[20] F. Somenzi. *CUDD: CU Decision Diagram Package Release 2.4.1*. Department of Electrical and Computer Engineering – University of Colorado at Boulder. http://vlsi.colorado.edu/~fabio/CUDD/

[21] B. Steinbach. XBOOLE – A toolbox for modelling, simulation, and analysis of large digital systems. *System Analysis and Modelling Simulation*, 9:297–312, 1992

[22] B. Steinbach. *Decision Diagram Technique for Micro- and Nanoelectronic Design*. Chapter: Decomposition Using Decision Diagrams, pages 509–544. CRC PRESS, Boca Raton, London, New York, 2006

[23] B. Steinbach, F. Schumann, and M. Stöckert. *Power and Timing Modelling for Performance of Integrated Circuits*. Chapter: Functional Decomposition of Speed Optimized Circuits, pages 65–77. IT Press, Bruchsal, 1993

[24] B. Steinbach and M. Stöckert. Design of fully testable circuits by functional decomposition and implicit. In *Test Pattern Generation. Proceedings of the 12th IEEE VLSI Test Symposium*, pages 22–27, Cherry Hill (New Jersey) USA, 1994

Index