Interface modeling algorithms for dispatch control

G V Redreev, V D Chervenchuk, I V Chervenchuk and A I Zabudsky
Omsk State Agrarian University named after P. A. Stolypin, 1, Institutskaya sq.,
Omsk, 644008, Russia
E-mail: gv.redreev@omgau.org

Abstract. The article offers mathematical tools for programming dispatcher tasks on
programmable logic matrices, which allow to obtain closed axiomatic models of the dispatcher's
professional activity in the course of the dispatcher's dialogue with the computer and implement
it in the form of optimal combinational logic schemes with pairwise alternative outputs. These
tools implement a user-friendly interface for modeling a logically complete consistent system of
axioms written in the production language in a tabular form that resembles decision tables with
a limited input. The modeling dialog completely frees the dispatcher from the need to know the
properties of axiomatic models and requires only professional skills of dispatcher management.
Its function in this dialogue is only to specify the desired action in a specific situation that occurs
during the object management process. Such situations are constantly formed during the field
work in crop production. Sudden failure of power units, two or three-day period of adverse
weather increases the risk of violation of agricultural deadlines. This requires prompt decision-
making on re-completing units, changing the plan of mechanized work. The search of all possible
situations and obtaining on their basis an optimal set of consistent products closing the axiomatic
model is provided exclusively by the interface software, which is developed using algorithms
for negating and minimizing disjunctive normal forms of logical functions and includes software
tools for modeling logically complete consistent decision tables and optimizing them in order
to obtain optimal combinational logical devices with their subsequent hardware implementation in
the form of microchips. In addition, these software tools are offered as one of the tools for
software implementation of branched algorithms with high complexity indicators according to
M.V. Arapov in the automated design of control systems for various purposes.

1. Introduction
In control systems of technical objects and technological processes, the dispatcher control subsystems
are often found. Maintaining their effective functioning is especially important in agricultural
production, where the time factor is very important. The functions of such subsystems are described by
the branched structure algorithms. Its programming has some specifics. It is very difficult to describe
such algorithms using algorithmic programming languages (Pascal, C++, etc.). To describe it, it is more
convenient to use specification languages in the form of a product system – a system of logically
complete and consistent decision rules [1]. To automate the process of creating integrated circuits for
dispatching control based on a production system in the form of a decision table, two tasks should be solved:

• an intelligent interface creation for the simulation of the full logically consistent decision tables
[2];
• a creation of software tools for optimization of logically complete consistent decision tables for obtaining optimal combined logical devices (CLD) that implement the dispatcher work [2].

2. Closed axiomatic model simulation interface
In the most general case, any axiomatic model is a four of sets:

\[ M = (A, \Sigma, \Xi, \Pi) \]

where \( A \) — the alphabet, \( \Sigma \) — the language syntax (grammar rules), \( \Xi \) — the system of axioms written in alpha-Sigma (\( A, \Sigma \)), and \( \Pi \) — the rules of inference that allow to obtain new statements \( \Xi \) from the axioms of the set - theorems.

In order for a logical inference machine using axioms \( \Xi \) according to the inference rules \( \Pi \) to be able to prove or disprove statements entered into it without human participation, it is necessary to fulfill the condition of closure of this model, i.e. the axiomatic model must meet two requirements [1]:

a) **logical consistency**, i.e. the system of axioms \( \Xi \) is such that any statement corresponding to the research subject area, in principle, cannot be simultaneously proved and disproved using a system of inference rules \( \Pi \);

b) **logical completeness**, i.e. the system of axioms is such that any statement corresponding to the subject area of research can in principle be proved or disproved using a system of inference rules \( \Pi \).

The decision table is a tabular form of writing a system of axioms \( \Xi \), and these axioms are given by Boolean functions in disjunctive normal form (d.n.f.)

\[ y_i = \bigvee_{j=1}^{r} \bigwedge_{k=1}^{s_i} x_{kj}, \]  

(1)

where \( y_i \) is the Boolean function whose value corresponds to the output \( i \) — digital signal; \( s_i \) — the number of elementary conjunctions in \( i \) — d.n.f.; \( x_{kj} \) — direct or inverse occurrence of \( k \) — input digital signal \( c_k \) (Boolean variable) in \( j \) — elementary conjunction; \( r \) — the number of input digital signals.

Here, the direct occurrence of the signal \( c_k \) in \( j \) — elementary conjunction is determined by the value \( x_{kj} = 1 \), and its inverse occurrence \( \overline{x_k} \) (i.e. \( HE \ c_k \) ) — is determined by the value \( x_{kj} = 0 \). If the signal \( c_k \) does not occur in \( j \) — elementary conjunction, the corresponding element \( x_{kj} \) will correspond to the «gap» symbol.

The hardware implementation of such a Boolean function is the CLD microchip with inputs and one output [7] (Fig. 1).

![Figure 1. CLD with \( r \) inputs \( c_1, c_2, \ldots, c_r \) and one output \( y_i \)](image-url)

The logical function of such the CLD is given by the following truth table
For example, let the CLD be set by a logical function
\[
y_1 = c_1 \overline{c}_3 c_4 \lor c_1 c_2 \overline{c}_4 \lor \overline{c}_1 c_2 \overline{c}_3 c_4 \lor \overline{c}_1 c_2 \overline{c}_3 \overline{c}_4 \lor \overline{c}_1 c_2 c_3 \overline{c}_4. \tag{2}
\]
Here is the number of elementary conjunctions \( s = 5 \), and the number of input signals \( r = 4 \); the conjunction sign (logical multiplication) is omitted, and the inversion sign (logical negation) is indicated by the upper line.

The logical truth table for function (2) has the following form:

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

The decision table structure allows you to create the CLD with multiple exits. For example, let's introduce two more digital outputs into this CLD, the value of which is determined by the d.n.f. \( y_2 = \overline{c}_1 \overline{c}_2 \overline{c}_3 c_4 \lor \overline{c}_2 \overline{c}_3 \overline{c}_4 \) and \( y_3 = c_1 \overline{c}_2 c_3 \), and consider a logical truth table for the CLD with three outputs \( y_1, y_2, y_3 \). We will get:

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\end{array}
\]

This is the decision table, which consists of eight decision rules (eight columns). The decision rule is called a structured proposal
\[
\text{«IF } C, \text{ then } Y \text{ » or } C(c_1, c_2, ..., c_r) \Rightarrow Y(y_1, y_1, ..., y_m). \tag{4}
\]
Here, the system of axioms \( \sum \) is defined by structured sentences of the form (4). This system must be consistent in order to provide the only correct decision – for any combination of signals at the CLD input, no more than one single signal at the output must be formed, i.e.
This condition will be met if
\[ \left( \forall i \neq i' \right) \left[ y_i, y_{i'} = 0 \right]. \]

We can check that for our example \( y_1 y_2 = y_1 y_3 = y_2 y_3 = 0 \), i.e. our system is consistent. If in the second row of the 8th column there would not be 0, i.e. \( y_3 = c_1 c_3 \), then according to (2)
\[ y_1 y_3 = \left( c_1 \bar{c}_3 c_4 \lor c_1 c_2 \bar{c}_4 \lor \bar{c}_1 c_2 \bar{c}_3 c_4 \lor \bar{c}_1 c_2 c_3 \bar{c}_4 \lor c_1 c_2 c_3 \bar{c}_4 \right) c_1 c_3 = c_1 c_2 c_3 \bar{c}_4 \neq 0 \]
and in this case, the computer would highlight the 2nd and 8th columns of the table (3) with the requirement to eliminate the contradiction between this pair of decision rules.

Only if there are no such semantic errors in the description of the decision table, the computer will start analyzing the table for logical completeness. During this analysis, the computer will give the user those situations in the CLD input that the user missed. The detection of situations that are not considered by the user is based on the negation algorithm of d.n.f. For our example this will be the calculation of the d.n.f. function
\[ y_1 \lor y_2 \lor y_3 = \]
\[ c_1 \bar{c}_3 c_4 \lor c_1 c_2 \bar{c}_4 \lor \bar{c}_1 c_2 \bar{c}_3 c_4 \lor \bar{c}_1 c_2 c_3 \bar{c}_4 \lor \bar{c}_1 \bar{c}_2 c_3 \bar{c}_4 \lor \bar{c}_2 \bar{c}_3 \bar{c}_4 \lor \bar{c}_1 \bar{c}_2 c_3 = \]
\[ = \bar{c}_1 c_2 c_3 \lor \bar{c}_1 c_3 c_4 \lor c_2 c_3 c_2. \]

Here three situations that the user has not considered in the CLD input are defined. So, the computer will ask questions about these situations.

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\hline
\overline{c}_1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
c_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
c_3 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
c_4 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
y_1 & y_1 & y_1 & y_1 & y_1 & y_2 & y_2 & y_3 & \text{?} & \text{?} & \text{?} \\
\end{array}
\]

In the darkened zone, insert the ID of the desired output signal in the bottom line instead of the questions. After any user adjustments to the description of this table its analysis for logical completeness and consistency begins again. The output from the simulation cycle will occur only when you get a logically complete consistent decision table.

Next comes the stage of optimizing the description of the decision table.

3. CLD optimization specified by a logically complete consistent decision table

The procedure for optimizing the description of a logically complete consistent decision table is based on the application of the algorithm for minimizing the d.n.f. of each Boolean function that defines the \( y_i \) output signal. As a result of such transformations, the uniform lines may appear in the decision table. A line is called uniform if it does not contain opposite elements, i.e. it contains either only zeros with gaps, or only ones with gaps. The uniform lines correspond to redundant input data. Such input signals do not affect the values of the CLD output signals, so you can delete such lines from the decision table along with the ID input signal.

After crossing out the uniform lines, you should again minimize the d.n.f. of the logical functions, and then identify and delete the newly formed uniform lines. The output from this optimization cycle is the condition of the d.n.f. functions \( y_i \) are no longer minimized and uniform lines no longer occur.
This cyclic optimization algorithm is described in more detail in [2], where its software implementation is also presented.

In our example, the function $y_1$ is minimized. The minimization algorithm (this algorithm is based on the d.n.f minimization algorithm of Quine-McCluskey, which is supplemented by a heuristic algorithm for finding optimal coverages) will simplify the expression (2) and give the result

$$y_1 = c_2ar{c}_4 \lor c_2\bar{c}_3 \lor c_1\bar{c}_3c_4.$$  \hspace{1cm} (6)

After the minimization the function $y_2 = ar{c}_1c_2ar{c}_3c_4 \lor ar{c}_2c_3c_4$ gives

$$y_2 = ar{c}_1c_2\bar{c}_3 \lor \bar{c}_2\bar{c}_3c_4.$$  

The functions $y_3$ and $y_4$ have a minimal d. n. f., since the d. n. f. is just an elementary conjunction, and the d.n.f. of $y_4$ was minimized in the process of forming situations not described by the user (this is how the d. n. f. negation algorithm described in [8] works).

Then the matrix (5) will take the form

$$
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
\end{array}
$$

$$y_1 \quad y_1 \quad y_2 \quad y_2 \quad y_3 \quad y_4 \quad y_4 \quad y_4 \quad y_4$$  \hspace{1cm} (7)

Since there are no uniform lines here, the optimization process will be completed. This optimization reduced the number of table columns by two and increased the number of gaps by 4. If a decision table with $s$ columns, $r$ lines and $n$ outputs is burned on a programmable logic matrix (PLM), it will take $2sr$ diodes and $sn$ transistors in it. We will save money for our table after optimizing it $2 \cdot 2 \cdot 4 = 16$ diodes and $2 \cdot 4 = 8$ transistors. At the same time, the performance of this integrated circuit will also increase; the more gaps in the decision table, the higher the performance of the corresponding CLD.

4. Conclusion

As in any intelligent system, there are three main blocks – a knowledge base represented by axioms – products, an intelligent interface based on the calculus of logical functions, and a solver, which is a hardware implementation of the dispatcher's work. The work of dispatchers in various control systems for complex objects and technological processes requires a lot of effort, quick reaction and complete exclusion of the right to make mistakes. Many accidents and catastrophes are often caused by the human factor. So, it is time to save a person from the work of a dispatcher. The dispatcher should transfer his knowledge about management to the computer in a calm environment during the dialogue, during which a logically complete consistent decision table is formed. The hardware implementation in the form of an integrated circuit must work in automatic control systems. The intelligent interface allows the dispatcher to transmit their knowledge about management without requiring special mathematical knowledge, and forms a closed axiomatic model from this knowledge. This model can not only be a software boot module for a microprocessor, but also be hardware implemented as an integrated circuit.

The scope of practical application of this interface is quite wide. While designing complex technical systems (in particular, in the project of the technical service system of a car and tractor fleet [3]), there are always problems of implementing algorithms of a branched structure, and the proposed tools in this case are quite effective and convenient tools. This tool can also be used in creating software and hardware for implementing supervisory control of multi-core computing systems, including such as [4–11]. In the structure of any, however complex, technical system, there will always be nodes with a large
number of pairwise alternative branches that require logical analysis for completeness and consistency. The proposed interface is intended as one of the tools for designing such systems.

References

[1] Chervenchuk V D and Chervenchuk I V 2013 Intellectual interface for modeling logically complete consistent decision tables Mathematical methods and information technologies in Economics, sociology and education: Materials of the XXXII International scientific and practical conference (Penza: Privolzhsky house of knowledge) pp 88-89

[2] Chervenchuk V D 1984 Methods and means of synthesis of algorithmic and software control systems using decision tables, candidate dissertation (Omsk)

[3] Redreev G V, Luchinovich A A, Ustiyantsev E I and Laskin A S 2018 Information System of Machines and Tractors Fleet Technical Service Journal of Physics: Conference Series 1059 012003 Available at: https://doi.org/10.1088/1742-6596/1059/1/012003.

[4] Ziganurova L and Shchur L Synchronization Aspects of The Optimistic Parallel Discrete Event Simulation Algorithms (Moscow: National Research University Higher School of Economics)

[5] Falcon J S Model Development and Measurements for Real-Time Systems

[6] 2006 Supervisory Control and Data Acquisition (Scada) Systems for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities: Technical Manual (Headquarters, Department of the Army)

[7] Yajuan Suna, Hai Linb and Ben M Chena 2018 Bisimilarity Enforcing Supervisory Control for Deterministic Specifications, Preprint submitted to Automatica

[8] Yudai Liu, Yiqun Pan and Zhizhong Huang 2013 Simulation-Based Receding-Horizon Supervisory Control of HVAC System Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association (Chambéry, France, August 26-28)

[9] Dallal E, Colombo A, Del Vecchio D and Lafortune S, Supervisory Control for Collision Avoidance in Vehicular Networks Using Discrete Event Abstractions: Research supported in part by NSF grant CNS-0930081.

[10] Indriawati K, Musyafa A, Widjiantoro B L and Ummah A M 2018 Study of Supervisory Control Implementation in A Small Scale Variable Speed Wind Turbine E3S Web of Conferences 43 01023 Available at: https://doi.org/10.1051/e3sconf/20184301023

[11] Rybak L, Gaponenko E and Malyshev D 2018 Synthesis of control algorithms for robotic platform with electro-hydraulic drive MATEC Web of Conferences 148 11006 Available at: https://doi.org/10.1051/matecconf/201814811006