DUMKA_E

An Event Based Control (EBC) Traffic Signal Controller

DUMKA_E Programming Manual

January 2024

Denis Sarazhinsky Department of Transport Systems and Technologies, Belarusian National Technical University, Minsk, Belarus, 220013 (*sarazhinsky@mail.ru*)

Taraneh Ardalan Department of Civil & Environmental Engineering, University of Pittsburgh, 3700 O'Hara Street Pittsburgh, Pittsburgh, PA 15261, USA (*taa93@pitt.edu*)

Aleksandar Stevanovic Department of Civil & Environmental Engineering, University of Pittsburgh, 3700 O'Hara Street Pittsburgh, Pittsburgh, PA 15261, USA (*stevanovic@pitt.edu*)

Development of the Event-Based Traffic Control Concept: **Dr. Denis Sarazhinsky**
Development of the DUMKA_E Software: **Dr. Denis Sarazhinsky**
Prototyping DUMKA_E Manual: **Dr. Denis Sarazhinsky**
Crafting DUMKA_E Manual: **Taraneh Ardalan, Dr. Denis Sarazhinsky, Dr. Aleksandar Stevanovic**

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose of this Manual

The purpose of the manual for the signal Event-Based Control (EBC) traffic controller, known as DUMKA_E, is to offer clear, concise, and comprehensive instructions for experienced traffic engineers and traffic controller operators interested in integrating flexible traffic signal control strategies into their projects. The manual provides detailed information on the basic concepts and principles of the signal Event-Based Control (EBC) traffic controller framework, along with instructions on programming the traffic signal controller DUMKA_E using this framework. It offers step-by-step guidance for performing common tasks and serves as a reference for troubleshooting, problem-solving, and debugging. In summary, the manual is designed to be a valuable resource for users during the implementation and operation of DUMKA_E in their system.

1.2. Event Based Control (EBC) Signal Traffic Controller

The Event-Based Control (EBC) Signal Traffic Controller DUMKA_E is a flexible traffic signal controller that is designed to handle conflicting requests arising from various road users using an event-driven paradigm. Its logic is based on signal events, which are defined as the change in signal display determined by the user-defined algorithm in the controller and the responses from the controller. The DUMKA_E virtual controller uses a precedence-based approach to process signal events and adjusts signal timings, accordingly, enabling it to respond to real-time traffic conditions. It is intended to provide a more adaptive and efficient solution for traffic signal control compared to traditional controllers' frameworks. The EBC controller framework provides a flexible environment and an efficient programmable system for traffic signal control based on desired rules, allowing for programable actions applicable to common traffic control systems and providing a foundation to develop logics that requires advanced flexibilities. The Graphical User Interface of DUMKA_E, comprising its tabs, is illustrated in **Figure 1**.

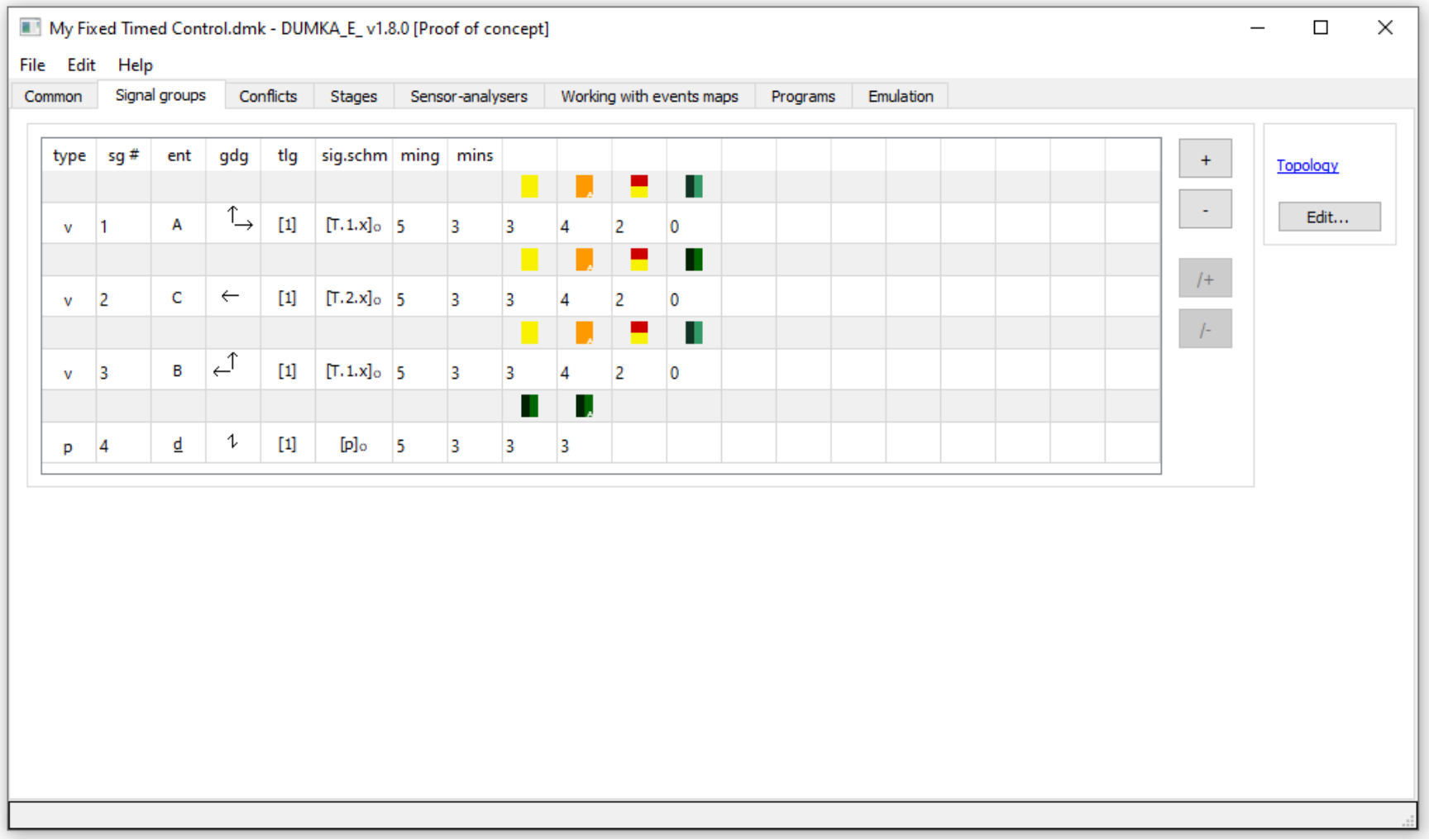


Figure 1. Event Based Controller Graphical Interface

1.3. Contents of this Manual

This manual is divided into *12 chapters*, followed by appendices.

Table 1. Chapters' Descriptions

Chapters	Description
1- Introduction	Outlines the purpose of the DUMKA_E Manual
2- Basic Concepts	Provides an overview of fundamental terminology
3- Moving Around Inside DUMKA_E	Introduces the user interface of the DUMKA_E
4- Signal Groups	Covers the essential steps in defining signal groups
5- Conflicts	Covers the essential steps in defining conflicts
6- Stages	Provides a summary of stages in DUMKA_E
7- Sensors-Analyzers (Detectors and Processors)	Covers the essential steps in defining sensors (primary and logical sensors)
8- Events Map	Covers the essential steps in building events map
9- Operative Map Editing Algorithms	Covers the essential steps in building and editing algorithms
10- Programs	Covers the essential steps in developing programs
11- Emulation	Provides essential information about debugging and emulation in DUMKA_E
12- Examples	Illustrates common control examples using DUMKA_E

2. Basic Concepts

This section provides an overview of the fundamental concepts and terminology used in the software. This section aims to clearly and concisely define concepts and key terms used throughout the manual. It is important for users to have a good understanding of the terms and their meanings to use and program the EBC effectively and smoothly to meet their traffic control goals. This section will serve as a reference for users to clarify terms and concepts that may be unfamiliar.

Logical Control Signal (Meta-Signal)

Meta-signal is a logical signal message, based on the standardized meaning from the Vienna Convention on Road Signs and Signals, Manual on Uniform Traffic Control Devices (MUTCD), or the Australian Road Rules, that can be displayed to a road user through different signal indications. It should be noted that the meta-signal could visually vary in different countries or states, but the meaning of it is the same. As an example, **Figure *2*** shows the different indications with the same message/meaning: the pedestrians who have not started crossing should wait, while those already crossing can continue and finish crossing.

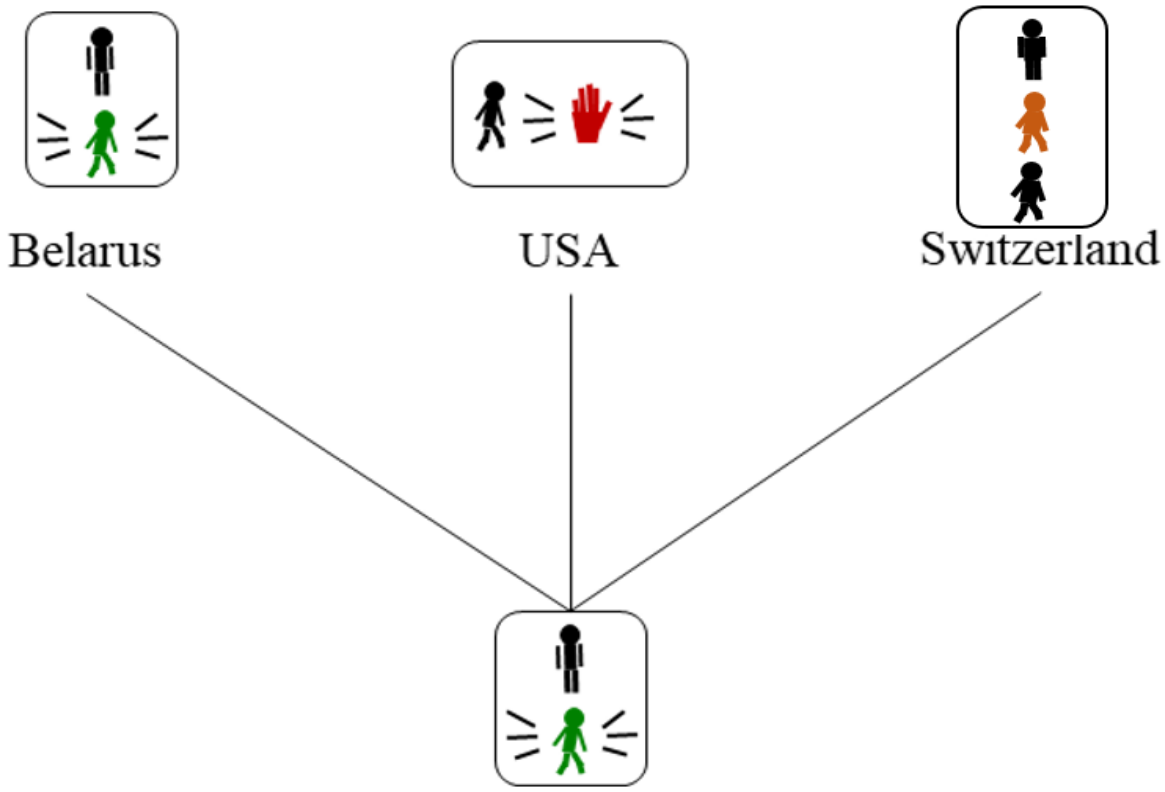


Figure 2. Meta-signal Example

The meta-signals can be divided into two categories:

Primary Logical Control Signal (Primary Meta-Signal)

A **Primary Meta-signal** is a type of meta-signal that conveys only primary control information:

- Type of controlled road user and the direction to which the signal applies.
- Right of way status (allowed/prohibited).
- Priority of passing in situations of conflict.

Transitional Logical Control Signal (Transitional Meta-Signal)

A Transitional Meta-signal is a type of meta-signal which follows the primary control information, by conveying supplemental information about the transitional process. The types of transitional processes are the beginning transition (B-transition) and/or the ending transition (E-transition), each of which is associated with the relevant state of the primary control information.

General Control Signal (General Meta-signal or Effective Meta-signal)

A General Meta-signal or Effective Meta-signal represents a group of meta-signals that share the same primary control information (loosely speaking, same right-of-way information). Every meta-signal in the group can be considered then as general meta-signal having some specific form – eighter primary or E-/B-transitional form.

For example, the group "red meta-signal", "yellow meta-signal", and "red and yellow meta-signal" is an example of an Effective Meta-signal since they share the same primary control information of stopping or slowing down the traffic.

Figure *3* illustrates three general meta-signals consisting of three primary meta-signals and two transitional meta-signals.

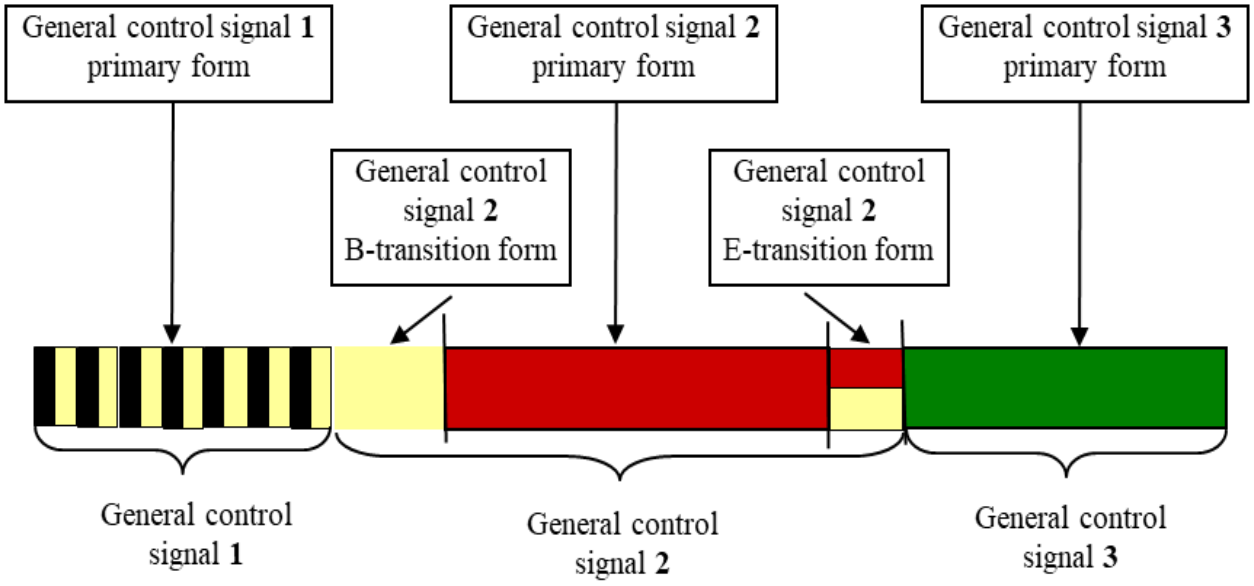


Figure 3. Primary, Transitional, and General Meta-signal

Different types of Primary and Transitional Meta-Signals based on the Vienna Convention are presented in the Appendix A.

Traffic Lights System (Traffic Signal Head Arrangement)

A traffic signal head arrangement is a class of signal heads that have the same layout of signal sections of given shapes and colors. This arrangement (or its equivalent) must be allowed by the Vienna Convention or other national standards (e.g., Manual on Uniform Traffic Control Devices). **Figure 4** shows some common traffic signal head arrangements.

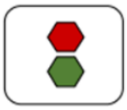
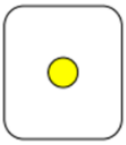
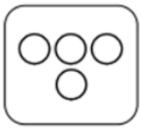
Name	Canonic Layout of Lights
Vehicles	
T	
t	
W	
Y	
R	
Pedestrian	
p	

Figure 4. Common Traffic Signal Head Arrangement

Traffic Light (Signal Head)

A Signal Head is any representative of traffic signal head arrangement.

Traffic Lights Group (Signal Heads Group)

The signal heads group is a collection of identical (or equivalent) signal heads of the same traffic signal head arrangement that are aimed to have identical signal indications at any moment of time.

Controlled Entry

The area of an intersection/crosswalk approach where signal indications of a specific signal head can only have effect (as defined by the Vienna Convention) on traffic entering the intersection/crosswalk.

Controlled Directions Group

Group of traffic directions (one or many), on a single controlled entry, controlled by the same signal head which can never be controlled independently, due to displaying constraints of the signal head.

Signal Scheme

Signal Scheme is an abstract scheme that clearly specifies how the given meta-signals can be translated into signal indications for a particular traffic signal head arrangement.

More details of signal schemes examples in DUMKA_E is presented in Appendix B.

Signal Group

A group of paths through an intersection/crosswalk, whose movements are controlled by the same meta-signal with the same signal scheme at any given time, as a choice of a traffic signal engineer.

Figure 5 indicates the difference between Controlled Entry, Controlled Directions Group, and Signal Group in an exemplary intersection.

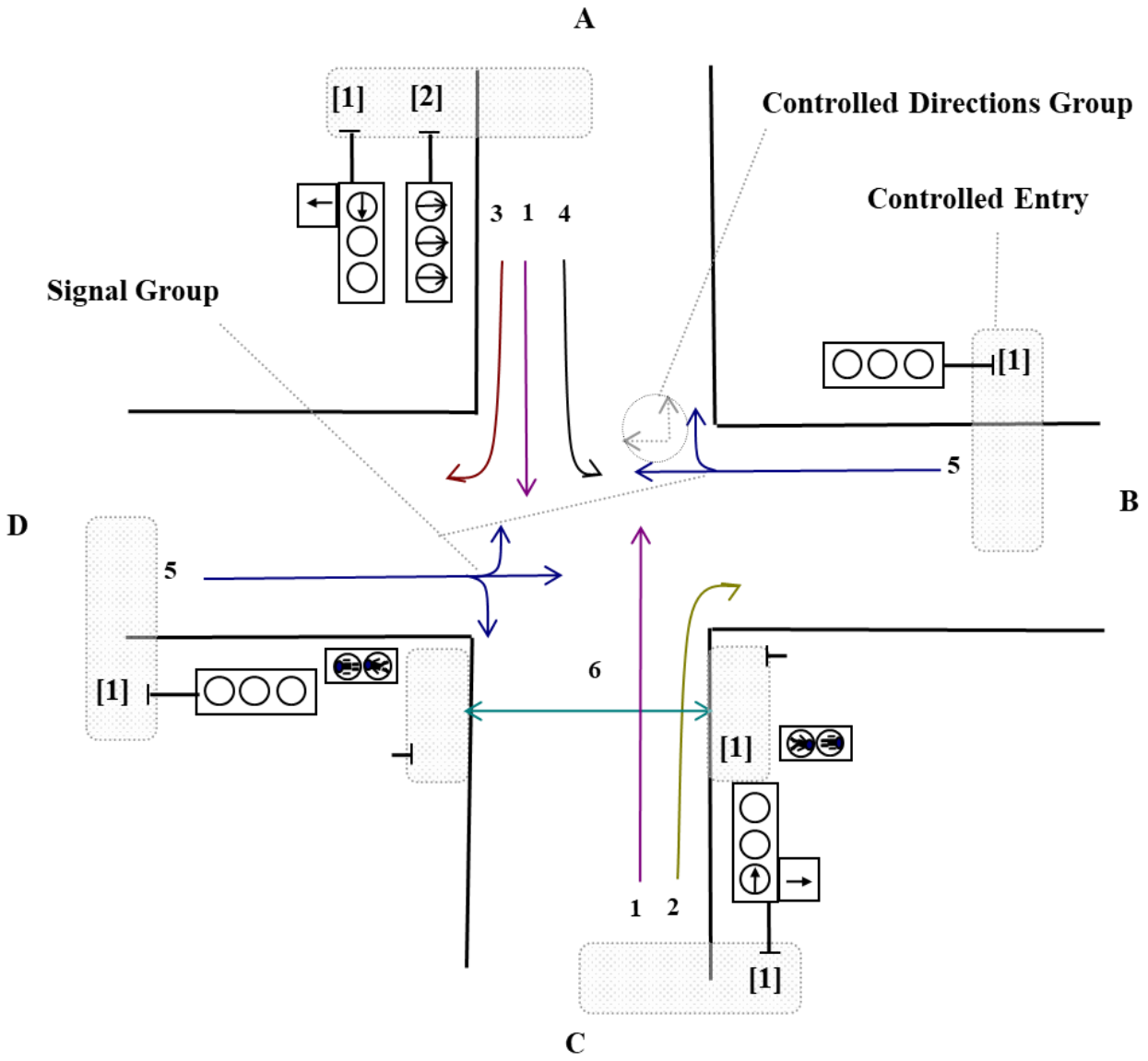


Figure 5. Controlled Entry, Controlled Directions Group, and Signal Group

As the reader begins to explore the new concepts of EBC DUMKA_E, it is important to note that this section only provides a brief introduction. For more in-depth information, readers are encouraged to consult the glossary at the beginning of each chapter, where the concepts are explained in detail. Furthermore, some of the concepts are given their own dedicated chapters, which provide a step-by-step explanation of all the details.

3. Moving Around Inside DUMKA_E

The chapter focuses on the different parts of the graphical user interface (GUI) of the EBC DUMKA_E software. The chapter will describe the various menus and display panels that are available to users. It will provide an overview of the purpose and function of each of these components and explain how they can be used to navigate and interact with the software. Overall, the chapter will provide a comprehensive introduction to the user interface of DUMKA_E, enabling users to easily access and utilize the software's full range of features and capabilities.

3.1. Window Interface Descriptions

The user interface is a crucial component of any software application. In the case of the EBC DUMKA_E software, the user interface provides a graphical representation of the software's functionalities and features. Users interact with the software via buttons, menus, and display panels, which are all part of the user interface. The user interface allows users to perform different tasks and operations, such as defining signal groups, defining conflicts, and developing logic. The subsequent section provides a detailed breakdown of the overall architecture of EBC DUMKA_E. **Figure 6** illustrates a screenshot of the DUMKA_E software user interface consisting of several modules where users' input is required to develop logic for a traffic signal controller.

Program a Signal Controller in DUMKA_E: Essential Information Guide

The following text provides the minimum information needed to program the desired signal control in DUMKA_E.

Step 1: Defining Signal Groups: To establish well-defined signal groups, it is necessary to provide qualitative data (e.g., road user type, geometrical directions/approaches, etc.) and quantitative data (e.g., signal group changing times, minimum greens, etc.).

Step 2: Defining Signal Groups Movements Conflicts: To prevent potential conflicts, it is crucial to meticulously define signal groups with conflicting rights of way. Additionally, careful consideration should be given to the "(safe) intergreen times", the minimum duration between the end of the right of way for one signal group, and the commencement of the right of way for the conflicting group.

Step 3: Defining Signal Events Map: Constructing a signal events map hinges on the creation of a map detailing every signal event. A signal event in EBC is defined as the change in signal display determined by a user-defined algorithm in EBC. This map is a chronological compilation of signal events linked by precedence relations based on their occurrences in a timely manner.

Step 4: Defining Logical Sensor (if needed): To simplify the handling of traffic data, logical sensors could be used. Defining the logical sensor(s) is a step that involves the post-processing of

detection data and the pre-processing of controller data. This step takes place only if there are physical detectors installed in the field. Based on the acquired data from physical detectors, logical sensors generate high-level logical quantity values. Examples of such logical sensors include:

- "Right-of-Way-Demand-Detector": This logical Sensor outputs zero if, at the moment, there are no unserved road users in the detection zone; otherwise, it outputs 1.
- "Right-of-Way-Usage-Gap-Out-Detector": This logical Sensor reports a 0 if, during the green phase, there is no gap exceeding the given threshold value for traffic related to the specified signal group; otherwise, it outputs 1.

Step 5: Defining Modification Algorithm for Operational Signal Events Map: This section offers information to design an algorithm that, when activated, can modify the specified operational signal diagram. To define it, the following information should be specified:

- "What" should be modified:
    - The portion of the signal event map under consideration for flexible changes (e.g., part of the current cycle, the next cycle, or both).
    - Signal event map "changing actions" that are intended for application (e.g., shorten green, skip green etc.)
- "When" it should be modified:
    - Flow-chart-like control logic defining moments when the "changing actions" should be applied to the signal event map.

An Operational Signal Diagram is a real-time representation of the signal diagram (i.e. time-series of signals for each signal group) indicating activated signals (realized part of the diagram) or scheduled activations (planned part of the signal diagram) for each signal group at any moment. These diagrams are continuously updated in real-time according to the operational signal event map.

Step 6: Defining Program: Defining the Program marks the final phase of the controller programming process, offering signal professionals the chance to establish a self-contained source of traffic signal control. This entails creating a cyclic fixed-time control program based on a signal events map and, if necessary, integrating adaptive algorithms to transform the fixed-time control into a traffic-dependent one.

Step 7: Emulating the Controller's Performance: EBC DUMKA_E programming tools can also emulate the operation of the programmed controller (with the aid of a built-in EBC DUMKA_E virtual controller). The emulation is helpful for the following purposes: verifying operating, fine-tuning operations of the controller, and debugging the adaptive algorithm (if needed) before implementing it.

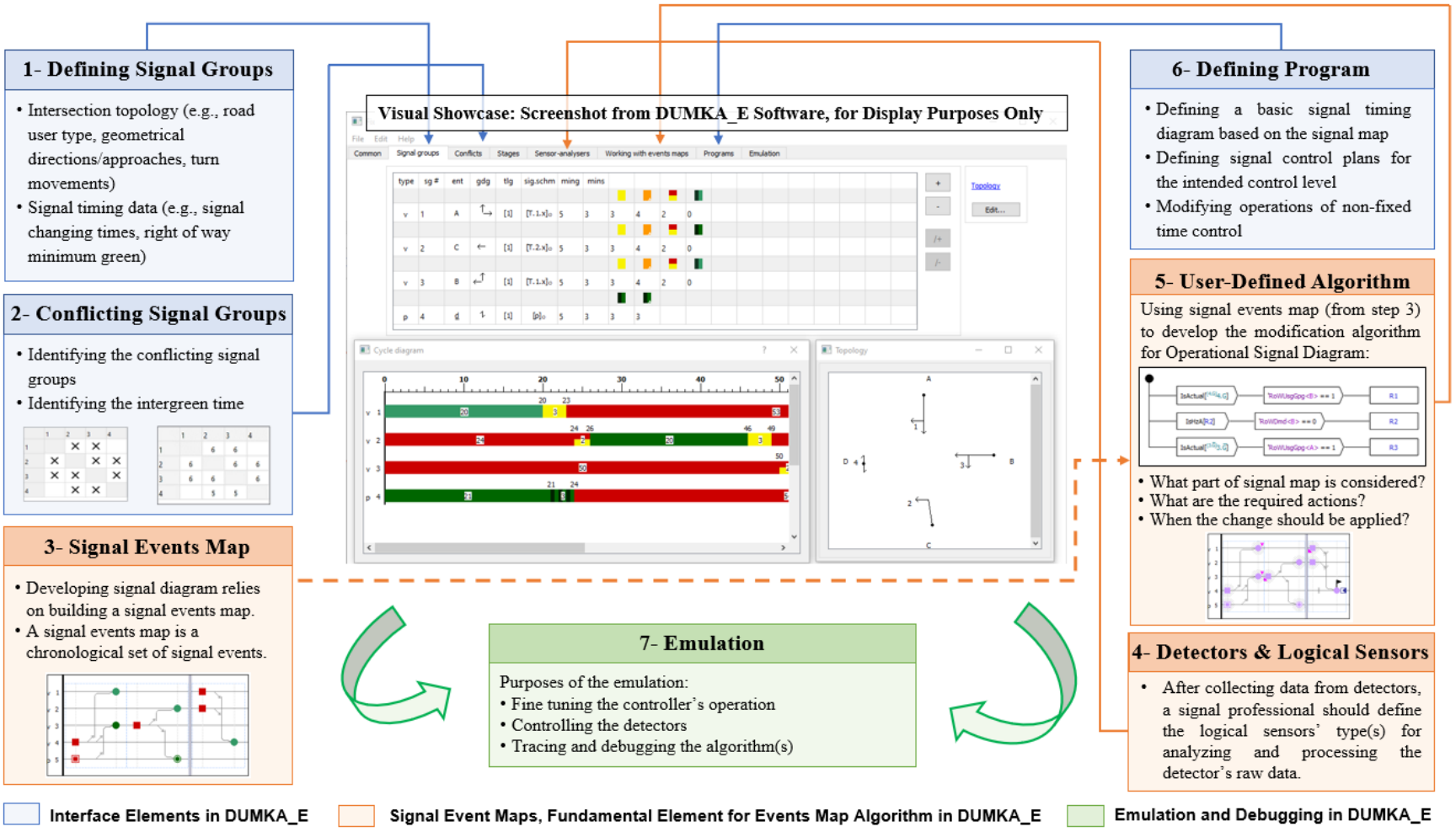


Figure 6. DUMKA_E Interface and Section

3.2. Main Window

The main window has 8 tabs, as shown in the illustration above and described in more detail in the **Table *2***. Sub-tabs are described later in each chapter.

Table 2. Main Tabs Summaries

Main Menu	Summary
Signal groups	Access to a submenu to define manageable signal groups providing information on road user type, signal group number, entrance approach, geometrical directions group, traffic lights group, signal scheme, and base timing. Refer to Chapter 4 for instructions.
Conflicts	Access to two submenus; conflict matrix and "intergreen" times table to define any movement conflicts. Refer to Chapter 5 for instructions.
Stages	Not available in this version of the software. General information in Chapter 6.
Detectors and Logical Sensors	Access to a submenu to provide information on detectors' kinds, locations, data sources, and other parameters that cover all aspects of detectors. Refer to Chapter 7 for instructions.
Events Maps	Access to two submenus; Events Map and Operative map editing algorithm to unlock the full potential of the signal events-based control methodology framework. Refer to Chapter 8 for instructions.
Operative Map Editing Algorithms	Access to two submenus; events map and operative map editing algorithms which enables the flexibility to develop the signal control logics within "signal event map editor", "algorithm flow-chart editor", and "edition editor". Refer to Chapter 9 for instructions.
Programs	Access to four submenus; Basic programs bank which provides information on the specifications required for the program's basic diagram, start-up/finish logic, and adaptation algorithm, if needed, and Hourly metaprograms bank, Daily metaprograms bank, and Yearly metaprograms bank. Refer to Chapter 10 for instructions.
Emulation	Access to various functions to provide fine-tuning operations of the controller, observing detection status, and tracing and debugging the algorithm(s). Refer to Chapter 11 for instructions.

4. Signal Groups

A user must provide both **qualitative and quantitative data** (e.g. topology of the intersection and basic signal timing data) to define accurate signal groups in EBC DUMKA_E.

Qualitative data includes the type of road user for the signal group (vehicle or pedestrian), the controlled directions groups (such as geometrical directions/approaches, turn movements, etc.), and the signal scheme (refer to Appendix B for more information on this concept) to be used for the desired control. **Figure *7*** shows the signal groups overview in EBC DUMKA_E.

Figure 7. Signal Groups Overview

Qualitative Data

- Signal group road user type (vehicle or pedestrian)
- Signal group-controlled directions
  - Controlled entry
  - Corresponding geometrical directions group
  - Traffic lights group (unique within the controlled entry)
- Signal scheme

Quantitative data includes the transitional control signals' duration times, giving the right-of-way primary control signals' minimum time, and motion forbidding primary control signals' minimum time (minimum amount of time that a traffic signal must remain red (or display a "motion forbidding" signal, such as a solid red light or a red arrow) before it can switch to a green or yellow signal to allow traffic to proceed. This minimum time is typically to prevent drivers and pedestrians from perceiving the traffic signal control as malfunctioning, thereby avoiding the likelihood of them ignoring it. These values are necessary for determining the timing of the signals within the signal group.

Quantitative Data

- Transitional control signals' duration times.
- Right-of-way primary control signals' minimum time.
- Motion forbidding primary control signals' minimum time.

Table *3* provides an example of how to define qualitative and quantitative data for a situation similar to that shown in **Figure *8***. By providing both qualitative and quantitative data, a user can specify a signal group that meets the necessary criteria for proper traffic control within EBC.

Table 3. Corresponding Qualitative Data for Figure 8

Road User Type	Signal Group #	Controlled Directions Group		Traffic Lights Group	Signal Scheme
		Controlled Entry	Geometrical Directions Group		
v	1	A	↑	[1]	$[T.1.x]_o$
		C	↑	[1]	
v	2	C	→	[1]	$[T.1.e]_A$
v	3	A	→	[1]	$[T.1.e]_A$
v	4	A	←	[2]	$[T.2.x]_o$
v	5	B	↑→	[1]	$[T.1.x]_o$
		D	←↑→	[1]	
p	6	c	↕	[1]	$[p]_o$

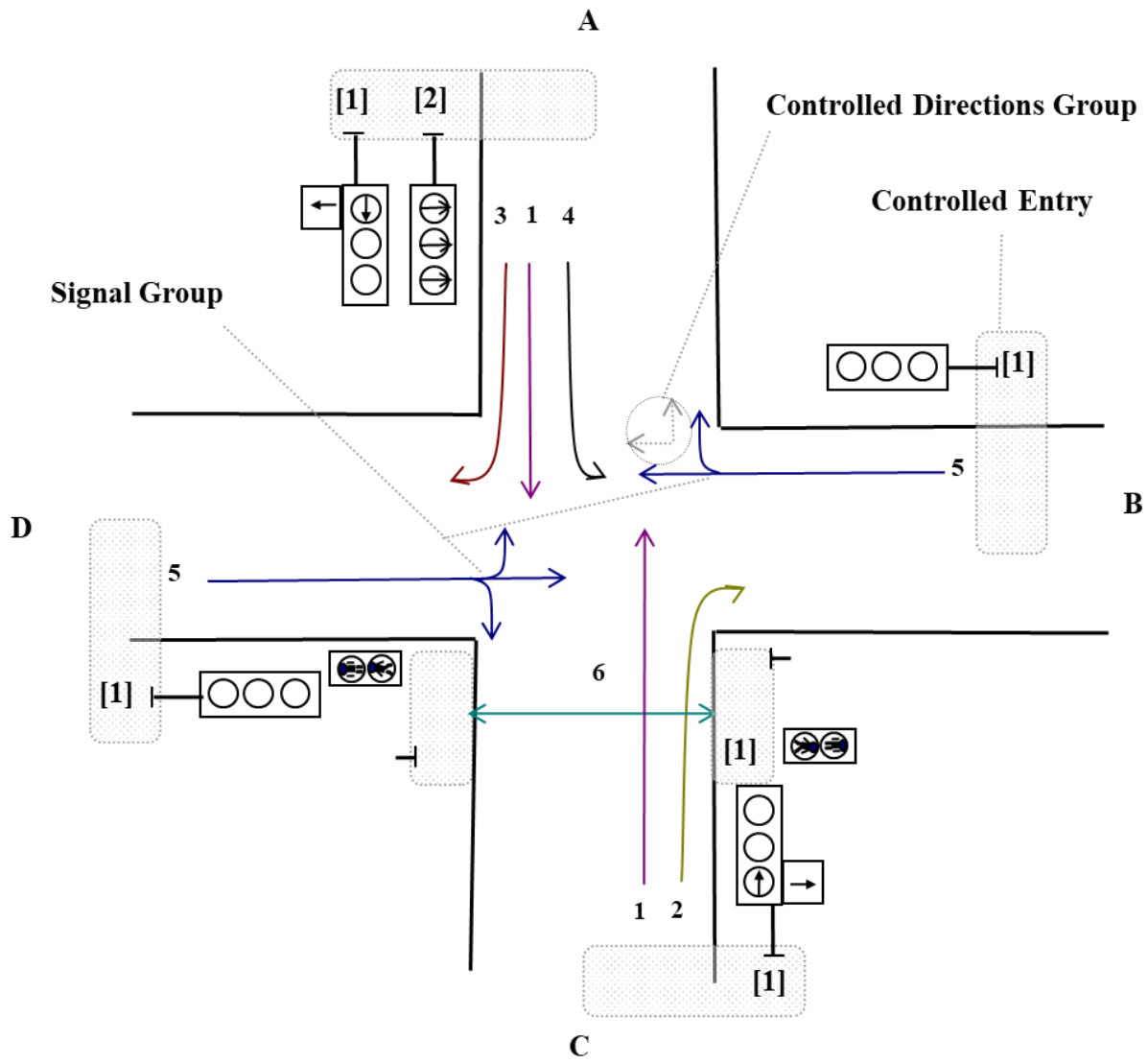


Figure 8. Exemplary Intersection to Define Signal Groups

Graphical User Interface – Signal Groups

This section provides instructions on how to specify Signal Groups for a traffic signal control system in EBC DUMKA_E. The signal groups tab is used to create and manage signal groups with detailed specifications, including road user type, signal group number, controlled entry, geometrical directions group, traffic lights group, signal scheme, and more. Users can add, edit, or remove signal groups and controlled direction groups, and perform undo/redo operations. The section also includes “Remarks” which contains helpful notes and recommendations for identification and graphical representation of intersection signal group topology.

STEP 1: Go to the "Signal groups" tab. **Figure *9*** illustrates step 1 of defining signal groups.

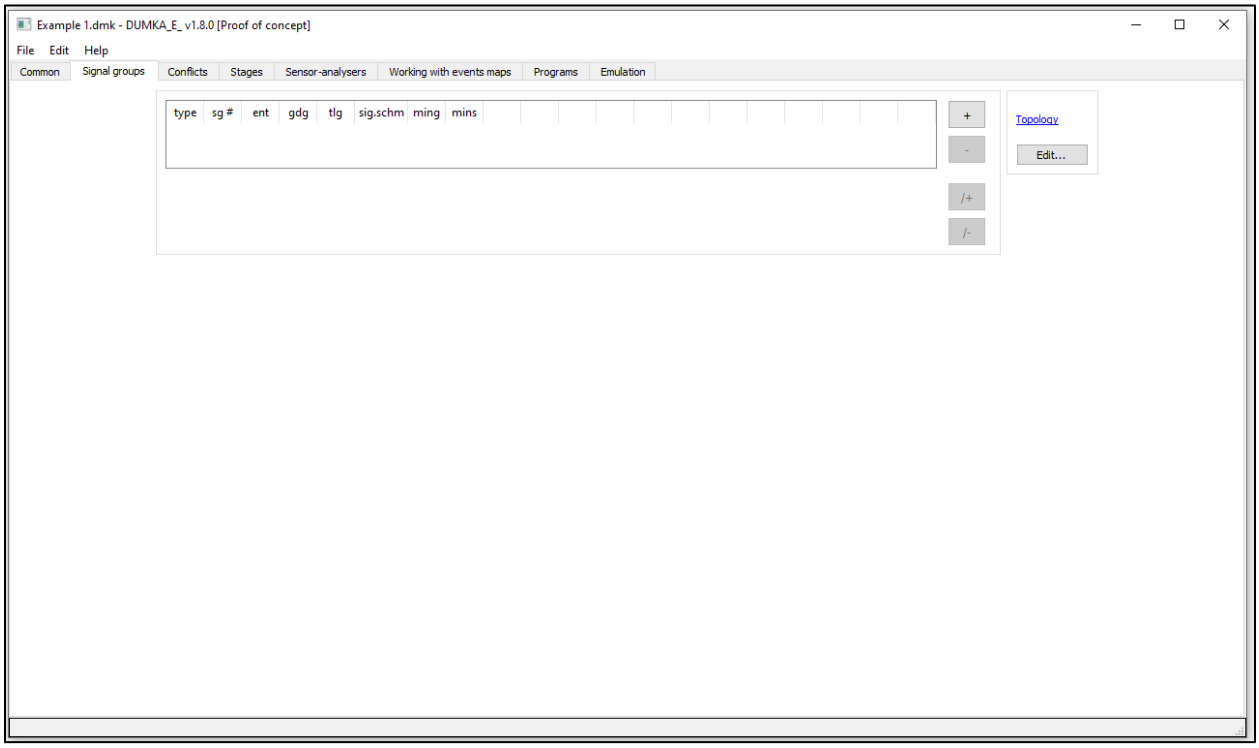


Figure 9. Defining Signal Groups - STEP 1

The "Signal groups" tab contains a table for specifying signal groups. The table has columns for the type of road user (type), signal group number (sg #), controlled entry (ent), geometrical directions group (gdg), traffic lights group (tlg), signal scheme (sig.schm), minimum time for the primary signal to give right of way (ming), minimum time for the primary signal to forbid motion (mins), and the remaining space in the table for defining the duration times for transitional signals.

STEP 2: To add a new signal group to the table, click on the "+" button (highlighted in **Figure *10***). This will create an empty row that can be filled with the corresponding data for the signal group.

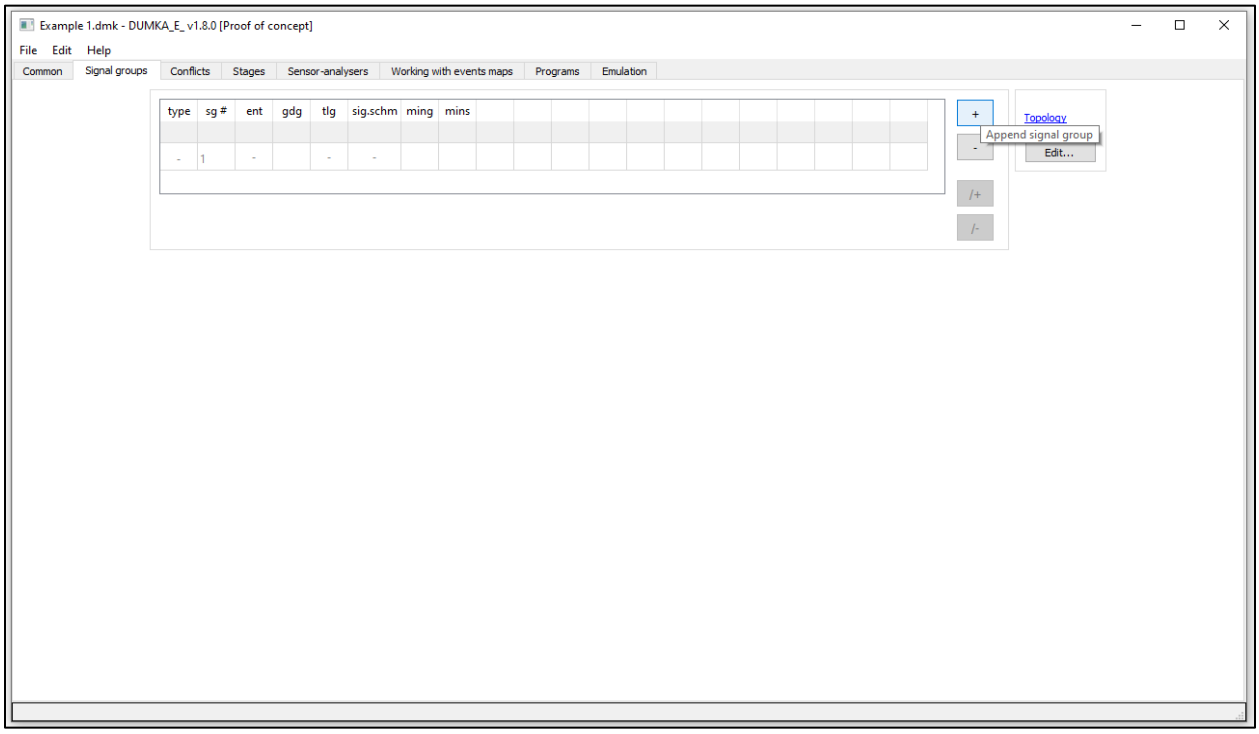


Figure 10. Defining Signal Groups - STEP 2

STEP 3: To define the signal group data, specify the type of user (v for vehicles and p for pedestrians) in the field “type” of the relevant table (shown in **Figure *11***). To edit an entry, double-click on a cell . When finished, click outside of the cell(s) you were editing while remaining inside the table area. To cancel an edit, press the “Esc” key.

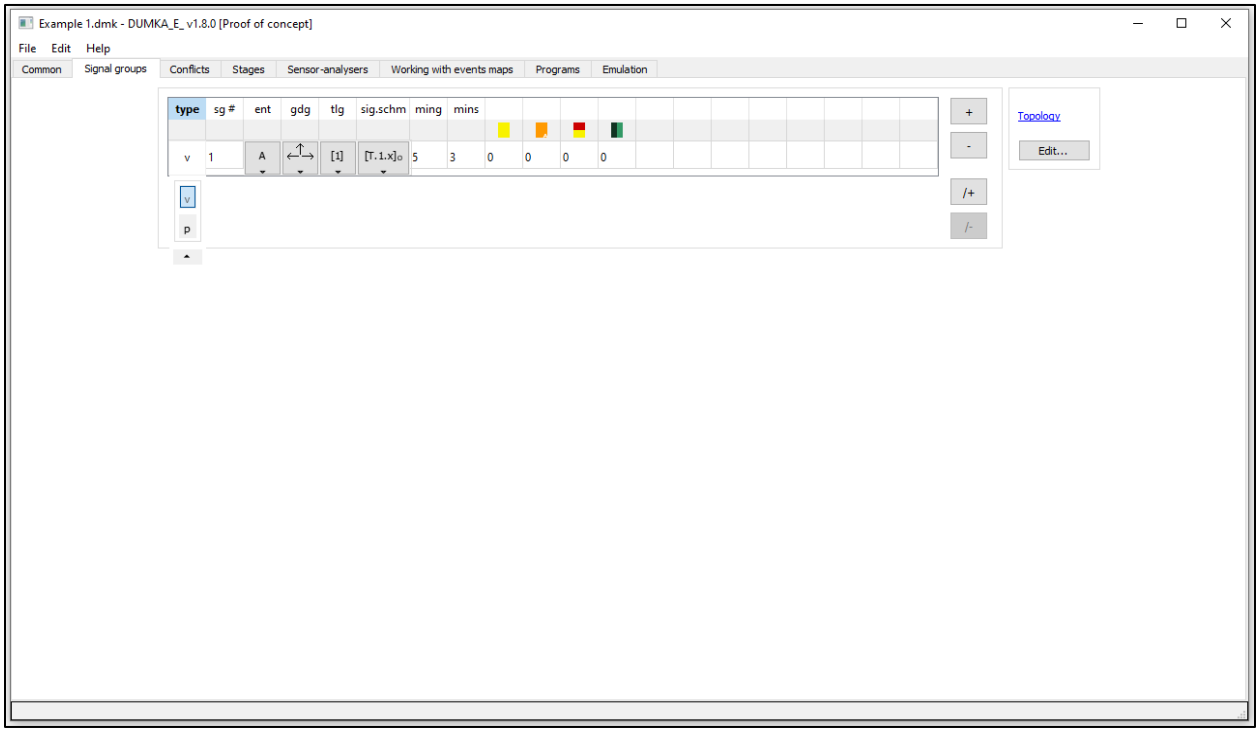


Figure 11. Defining Signal Groups - STEP 3

If multiple signal groups share a controlled entry (intersection approach zone), they should not have the same geometrical directions to control.

Remark 1: It is recommended to use the following coding (shown in **Figure 12**), based on location description, to identify controlled entry (intersection approach zone):

- For the road approach entry, use a Latin letter (capital for vehicle and small for pedestrian approach) in the code.
- For the road approach forward carriageway entry, use an upper index (carriageway index) in the code.
- For the road approach backward carriageway entry, use a lower index (carriageway index) in the code.
- For the sub intersection, use a left upper index as a Latin number in the code. This enables automatic graphical representation of the intersection signal group topology (accessible via the “Topology” hyperlink).

Figure 12. Signal Groups – Defining Intersection Approach Zone

If the entered data is consistent, the signal group will be treated as “implementable” and shown in usual colors. Otherwise, it will be treated as “non-implementable” and represented in gray colors.

The automatic graphical representation of the intersection shown in **Figure *8*** is presented in **Figure *13*** below retrieve from the Topology hyperlink in the Signal groups tab.

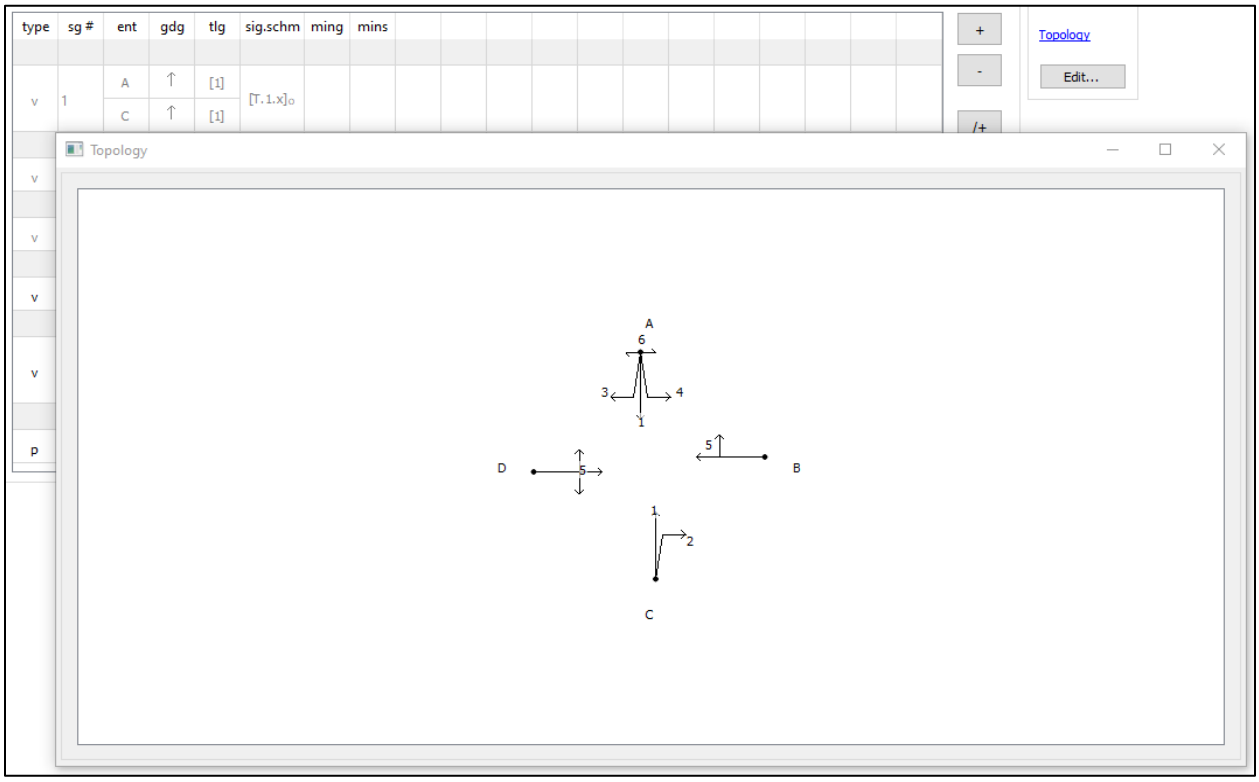


Figure 13. Graphical Representation of Intersection in Topology

Remark 2: The unwanted transitional signals, such as “flashing green” (shown in **Figure 14**) can be silenced by setting their time to 0. Some transitional signals may not be useful or necessary for every intersection or in every country as their use varies by rules and locations.

type	sg #	ent	gdg	tlg	sig.schm	ming	mins				
v	1	A	↑	[1]	[T.1.x]o	5	3	4	0	2	0
		C	↑	[1]							

Figure 14. Defining the Unwanted Transitional Signals

STEP 4: To add a second controlled direction group to the signal group, simply select any field of the signal group and click the "/+" button (highlighted in **Figure *15***).

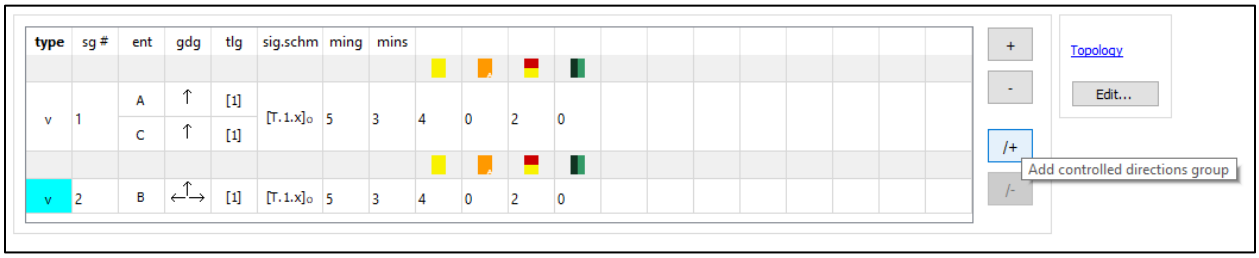


Figure 15. Defining Signal Groups - Adding Another Controlled Direction Group

To remove the last signal group in the table, click the "-" button.

To remove the last controlled direction of a signal group, select any field in the corresponding signal group and press the "/-" button.

Remark 3: It is important to note that once a signal group is added to the table, it cannot be removed from the beginning or the middle of the table. The only way to remove a signal group is by removing the last one in the table using the "-" button or by clearing the entire table. Therefore, it is recommended to plan the order of the signal groups carefully before adding them to the table.

To undo or redo an operation, navigate to the main menu, click on "Edit", and select "Undo" or "Redo".

This section presented a comprehensive guide on defining signal groups in the EBC DUMKA_E software. The step-by-step process accompanied noteworthy points to help users effectively create and organize signal groups.

5. Conflicts

This chapter in the EBC DUMKA_E Manual discusses defining conflicts. In the framework, two signal groups are considered to have no movement conflicts if all combinations of their signals are consistent from a safety perspective. If the combinations of signals are inconsistent, they are considered to have movement conflicts.

In the EBC DUMKA_E manual, conflicts between signal groups are specified by providing two key pieces of information. The first is identifying which combination of signals giving right of way is inconsistent, meaning that the signals could potentially conflict with each other from a safety perspective. The second is determining the "intergreen" times, which is the minimum allowable period of time between the end of the right of way for one signal group and the start of the right of way for the conflicting signal group. By specifying this information, the system can ensure that traffic flow is properly managed and safety risks are minimized.

Graphical User Interface – Conflicts

This section provides information on how to define conflicts through a graphical interface in EBC DUMKA_E. The Conflicts tab consists of two sub-tabs, "Conflict Matrix" and "Intergreen times table." This section outlines the step-by-step process for defining conflicts and provides important notes and remarks for users' consideration.

To define the Conflicts, follow the steps below:

STEP 1: Go to the "Conflicts" tab (see **Figure *16***).

The Conflict Matrix is located on the first sub-tab on the left in the Conflicts tab in EBC DUMKA_E. It is a graphical representation of conflicts between different signal groups. The Conflict Matrix provides a visual representation of the conflicts between different signal groups, making it easier to identify potential conflicts and plan appropriate intergreen times.

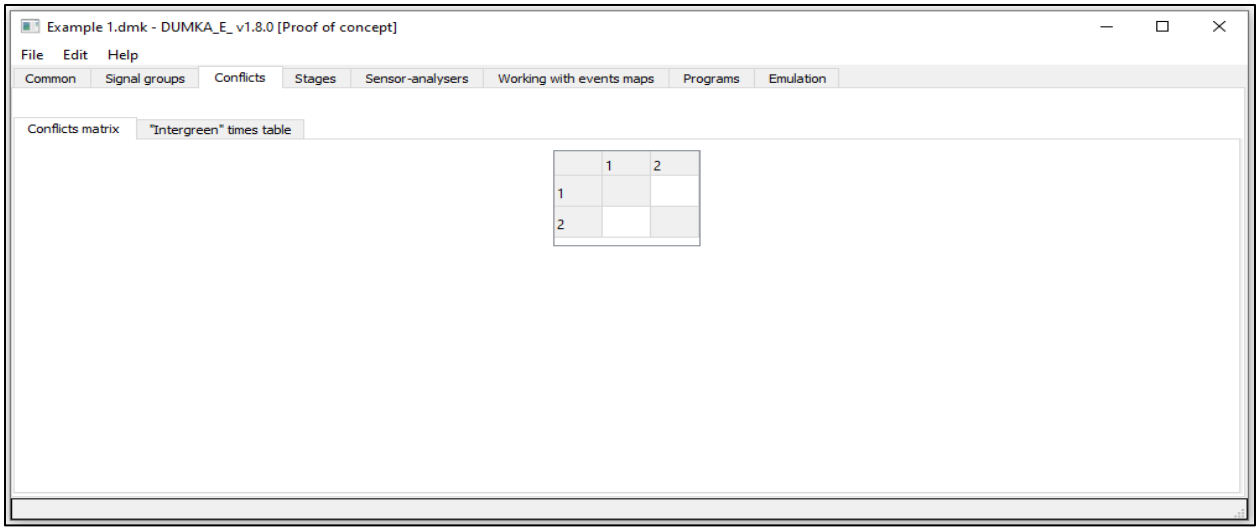


Figure 16. Defining Conflicts - Overview

STEP 2: To fill in the conflict data in the matrix cells, use the special conflict selector provided in the graphical user interface. Conflict matrix is shown in **Figure *17***.

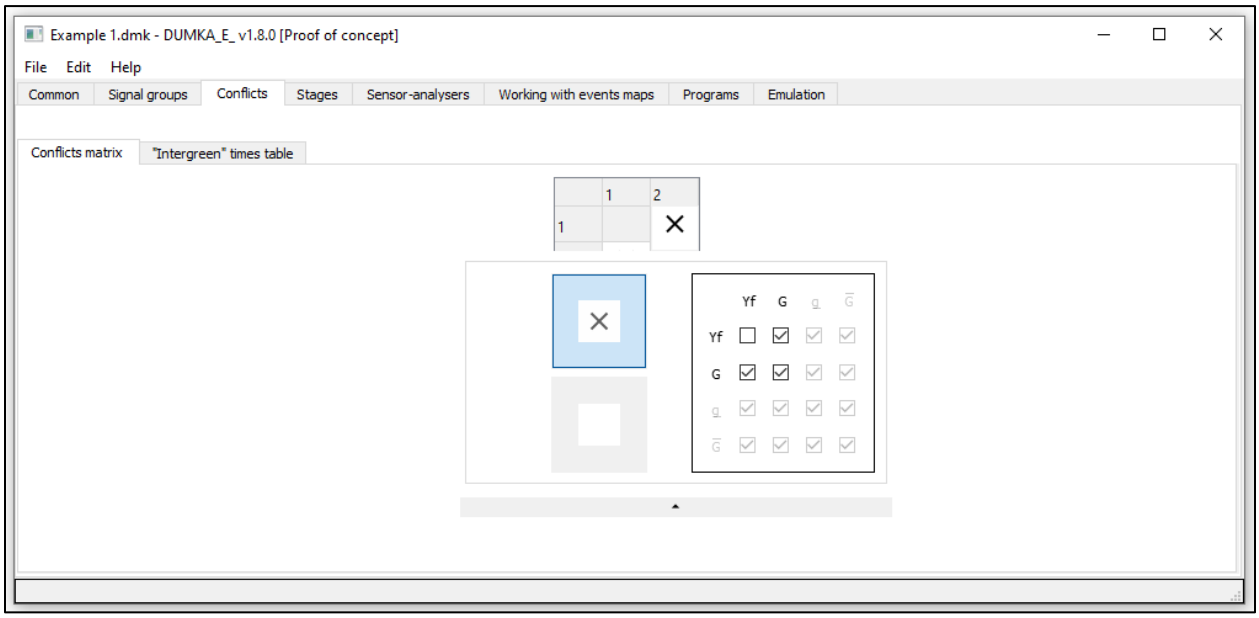


Figure 17. Defining Conflicts – Conflicts Matrix

This conflict matrix corresponds to the signal groups in the system. To indicate a conflict, simply select the corresponding cell in the matrix and enter the necessary conflict data. It is important to accurately identify the conflicting signal groups and provide the appropriate intergreen times to ensure the safe and efficient operation of the system.

The signal combinations inconsistency matrix is presented on the right, with each row representing a signal group with a right-of-way ending and each column representing the one

with a right-of-way starting. To specify conflicts, the corresponding matrix cells need to be filled with the conflict data, which can be done using the special conflict selector.

For the most common conflict where all signals giving right-of-way are inconsistent except for a combination like “yellow flashing” - “yellow flashing,” a particular button is provided with an “X” mark. This button can be pressed to specify that type of conflict promptly.

STEP 3: Proceed to the “Intergreen Times Table” tab, where you will need to input the intergreen time values in the corresponding table cells. The intergreen time table is shown in **Figure *18***.

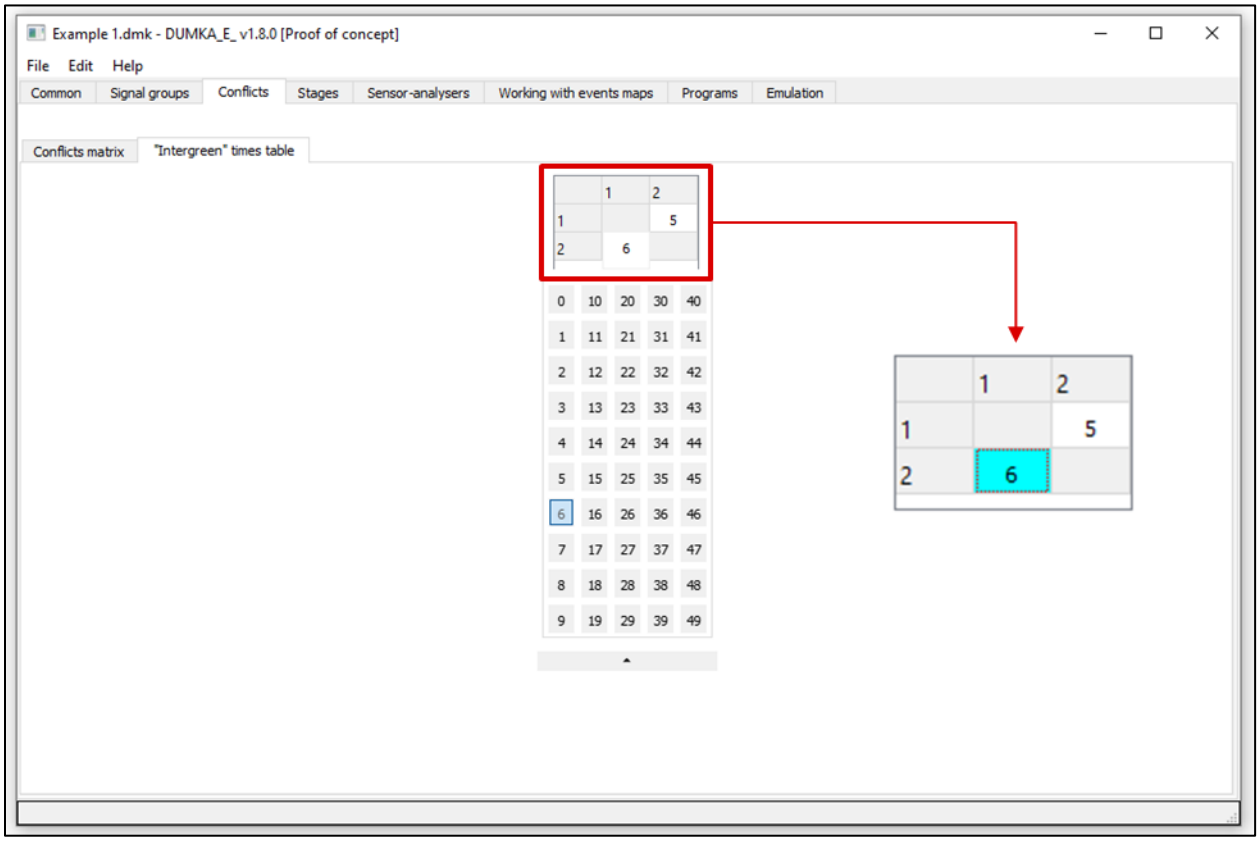


Figure 18. Defining Conflicts – “Intergreen” Times Table

STEP 4: To undo or redo an operation, navigate to the main menu, click on “Edit”, and select “Undo” or “Redo”.

In conclusion, the Conflicts chapter in the EBC DUMKA_E manual provided a step-by-step guide on how to define conflicts through a graphical interface using the Conflict Matrix and Intergreen Times Table sub-tabs. By specifying the inconsistent signal combinations and intergreen times, users can ensure safe and efficient signal control for intersections.

6. Stages

The “Stage” tab is currently unavailable in this version of the DUMKA software. The following information serves as a general overview and outlines what users can anticipate in future updates.

A stage represents a specific control state where each signal group is governed by a Primary Meta-Signal. A stage is defined as a collection of primary meta-signals, where each signal in the collection corresponds to a specific signal group (see **Figure *19*** for visual representation).

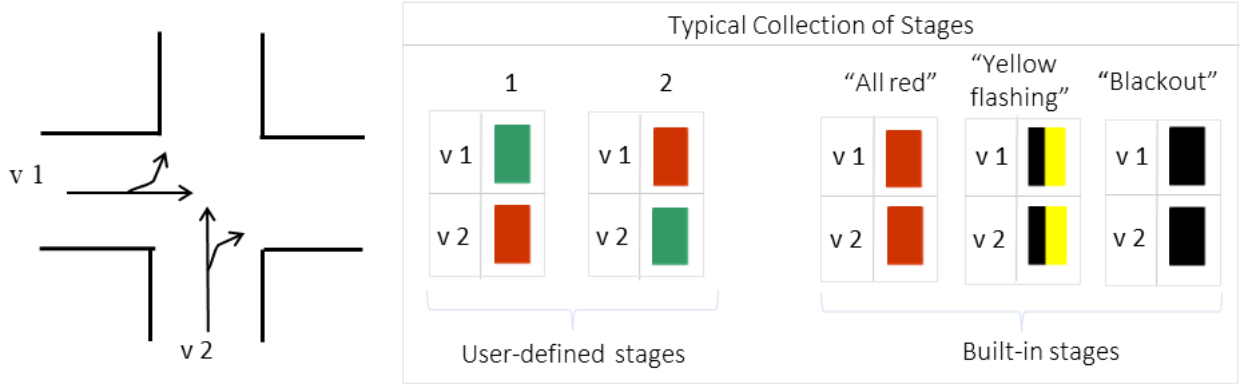


Figure 19. Explanation of Stage Concept

The concept of a stage is valuable for establishing a steady (with no transition), long-term condition for all signal groups simultaneously. For instance, the “all red” stage ensures a consistent state where all movements are stopped. These types of stages are also referred to as *main stages*.

In DUMKA_E, stages are utilized in various scenarios:

1. To deactivate all traffic signals, the predefined “blackout” stage can be activated.
2. To transition from the “blackout” or “yellow flashing” stage, a startup procedure is initiated before activating a control element (e.g., a program). This procedure involves sequentially transitioning through the following stages:
   “blackout” -> “yellow flashing” -> “all red” -> launching the control element. These intermediary stages are referred to as *meso-stages*.
3. To operate within the “blackout” or “yellow flashing” stage, a similar process occurs as in scenario 2, with the sequence:
   “Operational control element” -> “all red” -> “yellow flashing” -> “blackout”.
4. Stage-based traffic control requires constructing a signal diagram by sequentially activating different main stages interconnected by transitions through the *inter-stages.* **Figure *20*** illustrates the stage-based signal control diagram. This option is not available for users in this version of DUMKA_E.

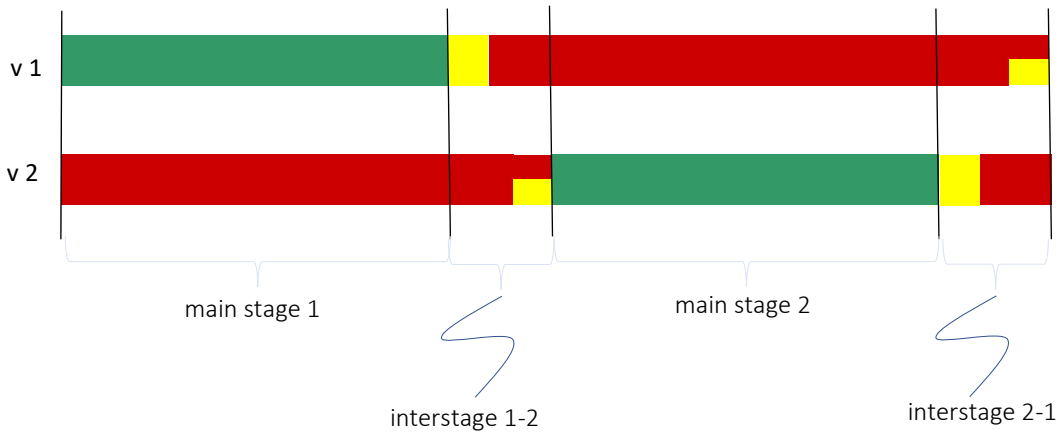


Figure 20. Example of Stage-based Signal Control Diagram (Cyclic)

7. Primary Sensors (Detectors) and Logical Sensors

Primary sensors (detectors) and logical sensors are essential components for creating the traffic-dependent control logic in the operation of Event-Based Control (EBC) systems.

Primary Sensor (Detector)

A Primary Sensor is a physical detector that interacts with an object of interest (such as a vehicle or pedestrian in a predefined zone) to obtain and deliver standard primary data about the object's characteristics. This data typically includes the presence or absence of the object in the zone, as well as the duration of the object's presence in the zone.

- **Logical Sensor-Analyzer**

A Logical Sensor-Analyzer (or simply logical sensor) is a logical entity that processes and occasionally analyzes data from detectors to generate new post-processed measured values for use in control logic.

Table 4 shows two important kinds of logical sensors in EBC DUMKA_E that are commonly useful in building signal control logic.

Table 4. Most Common logical sensors in EBC DUMKA_E

Logical Sensor Kind		Version	Measured Quantities		Required Parameters
Symbol	Meaning		ID	Meaning	
RtwDmd	R.O.W Demand Sensor	v0	Demand	Service Demand Presence (1: Present, 0: Absent)	1- Signal Group serviced; 2- Continuous demand flag
			Waiting time	Service Awaiting Time (seconds)	
RtwUsgGpg	R.O.W Usage Gapping Sensor	v0	Gapping	R.O.W Usage Time Gap Presence (1: Gap Presence Detected, 0: No Gap Detected)	1-Signal Group serviced; 2- Initial Period when gapping is not detected; 3- Gap 0 (initial gap value to detect); 4- Gap 1 (final gap value to detect); 5- Time interval where gap value changes; 6- Gap detection ratification delay.
…					

Logical sensors are, in general, stateful units that process the primary sensor values to produce high-level data. Here, “stateful” refers to a system or component that has a memory of prior events or user interactions and can maintain and track its internal state based on that memory

(that is why they have also “-analyzer” suffix in the full name). This contrasts with a “stateless” system or component, which does not maintain any memory of prior interactions and operates solely on the information it receives in the present moment. The logical sensors have several quantities that have been processed from the primary sensor data. They also have several parameters that can be fine-tuned for optimal performance. In simple terms, logical sensors take in the raw detector data and transform it into useful information that can be utilized for decision-making.

Graphical user interface – Sensors -Analyzers

This section provides information on how to define logical sensor-analyzers through a graphical interface in EBC DUMKA_E. The Sensors-Analyzers tab offers an empty box that can be populated with a table containing all the necessary information to define the primary sensors (detectors) and logical sensor-analyzers. This section outlines the step-by-step process for defining detectors and logical sensor-analyzers and provides important notes and remarks for user's consideration.

To define the logical sensors, follow the steps below:

STEP 1: Go to the “Sensors-Analyzers” tab (See **Figure *21***).

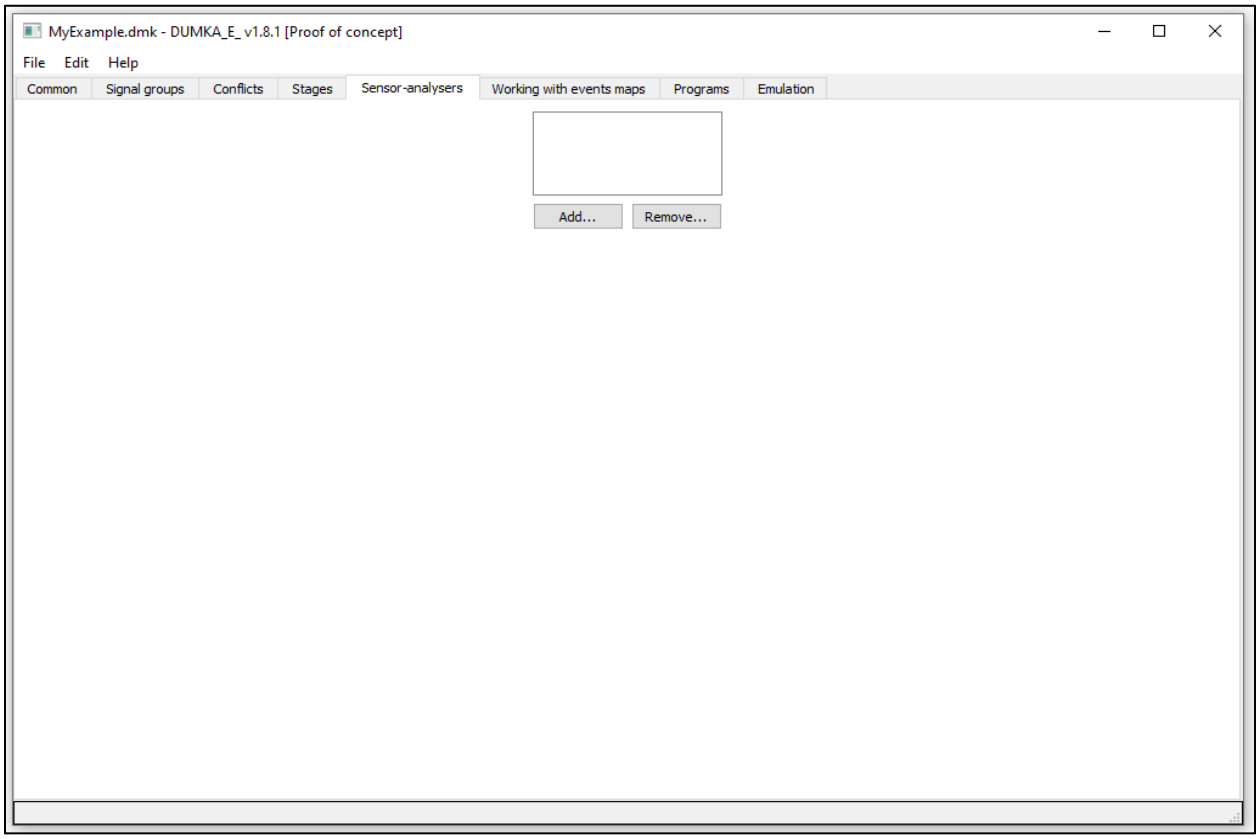


Figure 21. Defining Sensors and Analyzers - STEP 1

STEP 2: Click on the "Add..." button shown in **Figure *22*** to open the dialog box for creating a new *Sensors-Analyzers*.

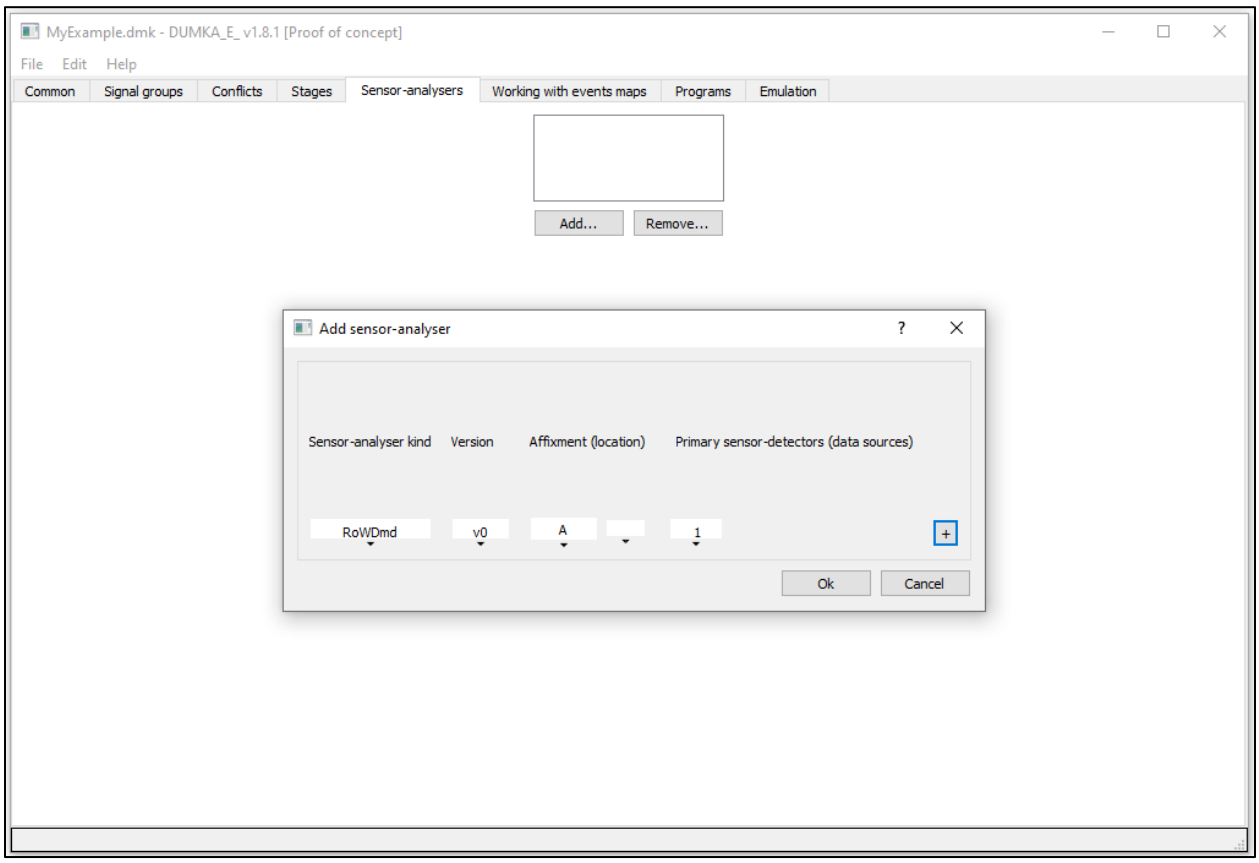


Figure 22. Defining Sensors and Analyzers - STEP 2

In the EBC DUMKA_E, the *Sensors-Analyzers* are identified using a specific format.

As it is shown in Figure 18, to define a *Sensors-Analyzer,* user needs to provide the following information:

- The kind of *Sensors-Analyzer* (i.e. what the sensor-analyzer is measuring)
- The version of the Sensors-Analyzer
- The location of the Sensors-Analyzer within the intersection (i.e. which entry and lane the detector is on)
- The primary sensors (detector) that the processor is using as a source of data
- The required parameters that are needed for the logical sensor-analyzer (i.e. parameters of algorithm of detectors data handling)

STEP 3: Select the appropriate configurations for each category for the logical sensor-analyzer and click on the “Ok” button to confirm.

The newly created logical sensor-analyzer will be added to the logical sensor-analyzer’s table with the chosen configurations and the default required parameters. The information in the parameters can be adjusted as shown in **Figure *23***.

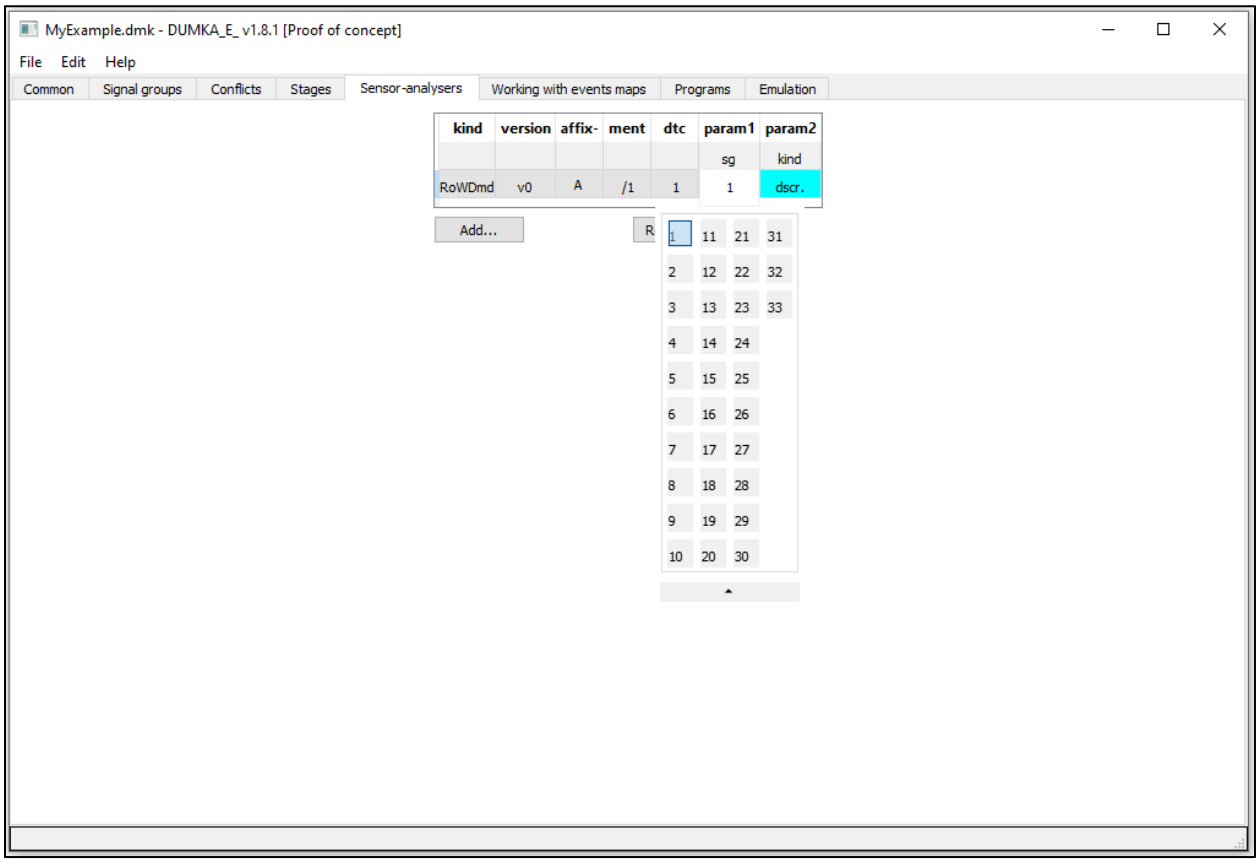


Figure 23. Defining Sensors and Analyzers - STEP 3

To remove a logical sensor-analyzer, choose the desired one and click on the “Remove...” button.

To undo or redo an operation, navigate to the main menu, click on “Edit”, and select “Undo” or “Redo”.

This section provides a comprehensive guide on how to define Sensors-Analyzers using the EBC DUMKA_E software. The step-by-step process is accompanied by noteworthy points to help users effectively create and organize the logical sensor-analyzers (essential elements of traffic-dependent control).

8. Events Maps

This chapter will explore how to work with the events maps in the signal events-based control methodology framework. This methodology builds a signal diagram on a signal events map. The events map serves as a specification of the system's core behavior via signal events and their relationships. By analyzing the events map, a user can gain insight into how the system functions and how it can be controlled through event-based mechanisms.

This section will focus on specifying an abstract signal events map, an essential step in building a signal diagram. This will cover the basics of creating an events map, including defining signal events and their attributes, specifying event relationships, and organizing the order of the events. Defining events involves identifying the events that can occur in the system, while specifying event relationships involves describing how events are related to each other, such as the conditions under which events can occur or the sequence in which they occur. **Figure *24*** illustrates the overview of Events Maps.

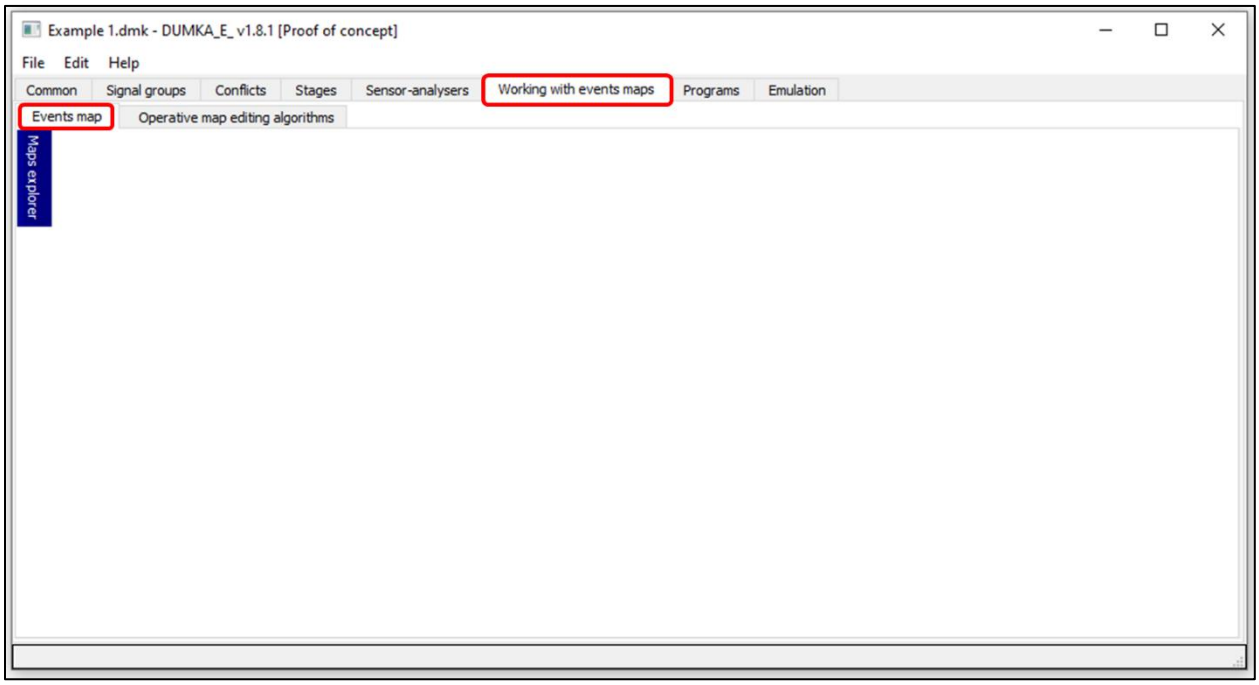


Figure 24. Events Map Overview

The process of creating a signal events map begins with the specification of an abstract signal events map. This involves defining the events and their attributes, such as event urgency and event signal transparency. After defining the abstract signal events map, the next step is to gather quantitative data specific to the program user is working with (the information about programs is discussed more in Chapter 10). This data will help to concretize the abstract map and create a more detailed signal events map.

After gathering quantitative data specific to the program, the next critical step is to refine the abstract signal events map to create a more detailed and concrete signal events map. This process is called “specifying a concrete signal events map”.

Finally, the concrete signal events map is processed and resulted in a signal diagram (**Figure *25***).

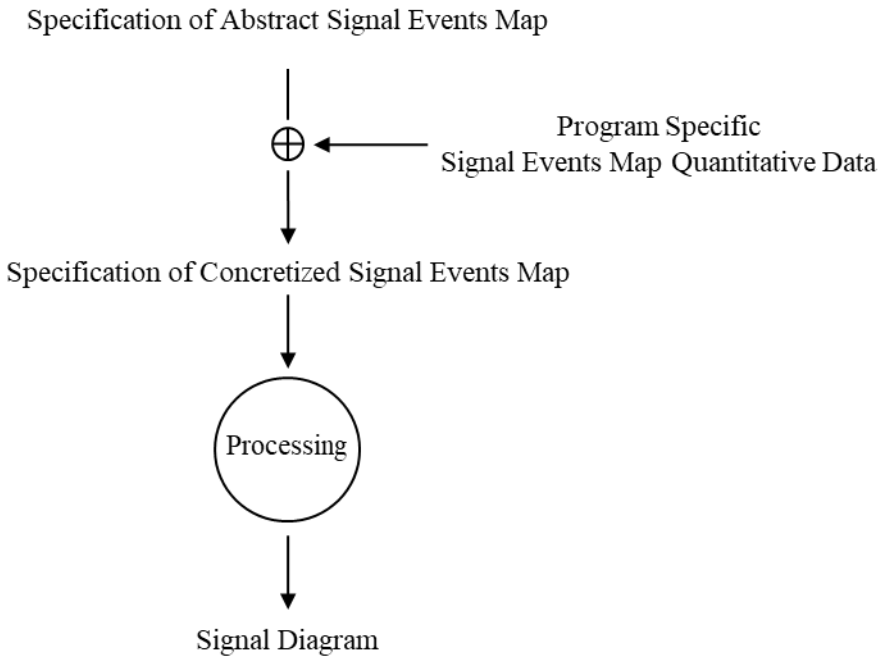


Figure 25. Signal Event-Based Control Methodology

Within the signal events-based control methodology framework, only regular signal events maps are permitted. Regular signal events maps can be presented as repetitions of the same infinite segment.

In the following, the essential terms and concepts related to events maps are introduced and defined in detail.

Signal Event: A General Meta-Signal Event (GMSE) is the activation of a general meta-signal. The activation of a GMSE is considered as an “event occurrence” that causes a traffic signal to change. In other words, a signal event is an occurrence of a specific signal state defined by the activation or deactivation of a signal.

Signal Events Graph (Basic General Meta-Signal Events Precedence Graph): Basic General Meta-Signal Events Precedence Graph is an infinite precedence graph of general meta signal events that define precedence relations between these events, where the precedence relations specify the order in which events occur. **Figure *26*** shows an example of a signal event graph for a T-intersection.

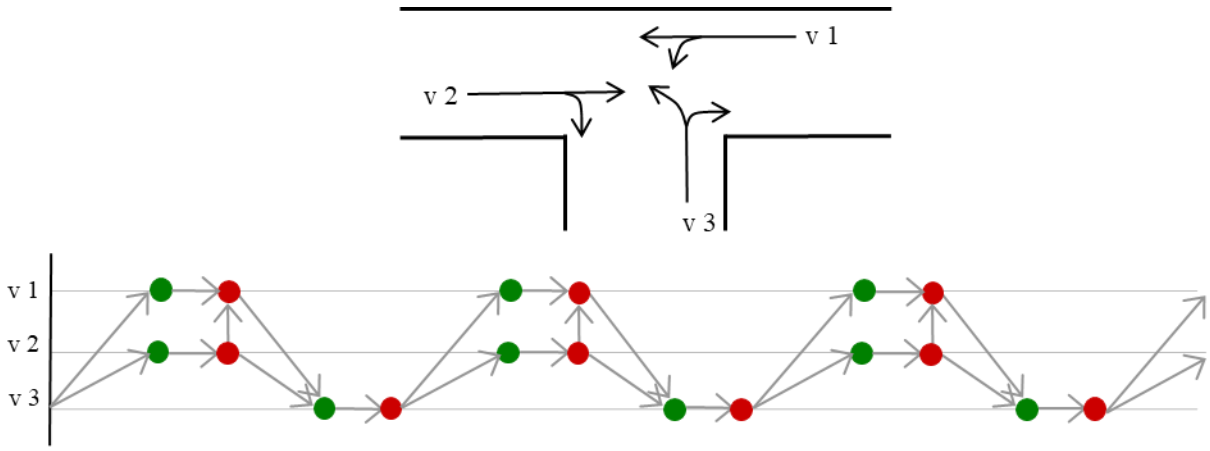


Figure 26. Signal Event Graph, T-Intersection Example

Signal Urgency, (General Meta-Signal Event) GMSE Urgency: Urgency is an attribute of GMSE that specifies how close or far that GMSE should occur relative to the present moment. In the context of the event-based controller system, urgency refers to the degree to which a signal event needs to be addressed in a timely manner based on its level of importance.

- *Positive Signal Urgency:* Positive urgency is an attribute of the GMSE which means that the GMSE should occur as close to the present moment as possible.
- *Negative Signal Urgency:* Negative urgency is an attribute of the GMSE which means that the GMSE should occur as far from the present moment as possible.

GMSE Urgency Weight: GMSE Urgency Weight represents an important factor (e.g., 1-4.) applied to positive and negative urgencies. **Table *5*** shows the graphical representation of different urgency weights.

In the following, the graphical convention is used to represent the *GMSE Urgency Weight* associated with a signal event in the event-based controller system. In this convention, positive urgency is represented graphically by a circular shape, while negative urgency is represented by a rectangular shape. The size of these shapes is used to convey the urgency weight.

As indicated in the table below, *GMSE Urgency Weight* is a number assigned to each signal event, and it represents the degree of urgency for that event. The higher the urgency weight, the greater the urgency of the event. The urgency weight is used to determine the order (priority) in which events are arranged on a timeline.

Overall, this convention helps visually differentiate between different types of signal events and their degree of urgency, which can be useful for users of the event-based controller system.

Table 5. Urgency Graphical Representation

Urgency Weight	(+)4	(+)2	(-)1	(-)3
Graphical Representation	●	◉	▣	■

Signal Transparency: Transparency is an attribute of GMSE that, being set to "transparent signal", enables the traffic signal engineer to omit the given signal event indication (e.g., omit green on a signal indication when traffic is not detected). "Transparent signal" doesn't change the signals in the signal diagram when activated. This attribute is particularly important in adaptive control systems, where the signal diagram may need to be adjusted based on changing traffic conditions.

Graphically, transparent signals are represented by a special texture filling, different from the standard filling used for other signal events. The special texture filling helps visually differentiate transparent signals from other signal events. Transparent signal representations are shown in **Figure 27**.

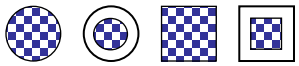
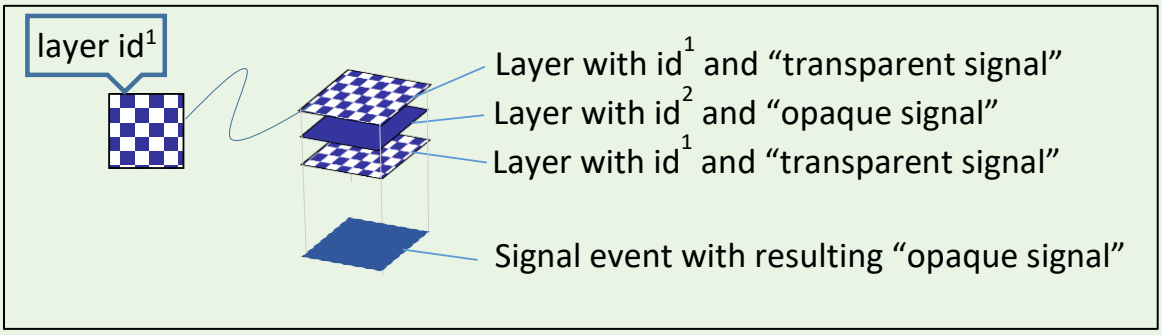


Figure 27. Transparent Signal Representations

By default, the transparency attribute is set to "opaque" for all signal events, which means that they will change the signals in the signal diagram when activated. However, when a signal event is designated as transparent, it will not change the signals in the signal diagram. A special texture filling will be used to indicate this attribute. As can be noticed from the representations above, the transparent signal follows the same logic regarding the signal urgency concept.

Remark 1: In essence, a signal event is conceptualized as comprising multiple layers, each potentially possessing its distinct signal transparency attribute. To denote these individual layers, an icon with the layer identifier is appended alongside the event icon itself.



Conceptual Framework for Layered Representation of Event Signal Transparency

The transparency attribute of the entire event is determined based on the following rule: if all layers are set to "transparent," the resulting value will also be "transparent." However, if at least one layer is set to "opaque," the resulting value will be "opaque." This mechanism facilitates a "voting procedure" among multiple signal groups to decide whether to make the event signal transparent.

Abstract Signal Events Map (Enhanced General Meta-Signal Events Precedence Graph): Enhanced General Meta-Signal Events Precedence Graph is a Basic General Meta-Signal Events Precedence Graph enhanced with qualitative attributes of GMSEs, specifically urgency and transparency.
Figure *28* is the Abstract Signal Events Map based on **Figure *26***. The circular signal events correspond to positive urgency, while rectangular signal events correspond to negative urgency.

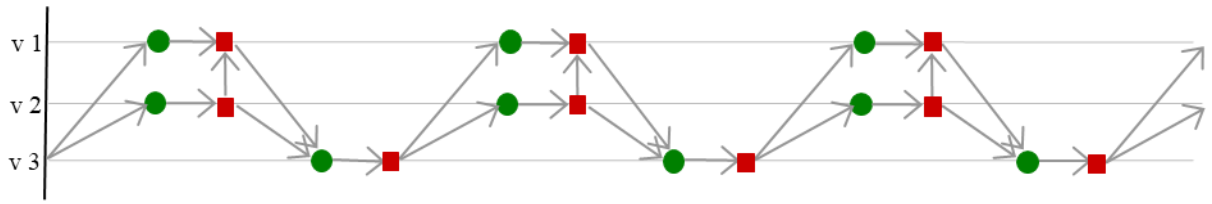


Figure 28. Abstract Signal Events Map, T-Intersection Example Figure 26

Event Aftereffect: Event aftereffect is a time duration following a general meta-signal event, which restricts other general meta-signal events (of the same signal group) from occurring. Two parameters define the event aftereffect:

- The minimal aftereffect duration: this is the minimum amount of time that must pass before another signal event of the same signal group can occur.
- The main aftereffect duration: this is the typical (used in typical situations) amount of time that should pass before another signal event of the same signal group can occur.

Event Temporal Separation: Event Temporal Separation is an interval that represents a minimum time that needs to elapse between two GMSEs defined by a precedence relation. In other words, if there is a relationship between two events in the graph, this characteristic defines the minimum amount of time that must pass before the next occurrence of either event so that the relationship can be maintained.

Concretized Signal Events Map (Comprehensive General Meta-Signal Events Precedence Graph): Comprehensive General Meta-Signal Events Precedence Graph is an Enhanced General Meta-Signal Events Precedence Graph enhanced additionally with quantitative attributes of GMSEs, specifically Event Aftereffect and Event Temporal Separation.

Figure 29 is an example of a Comprehensive General Meta-Signal Events Precedence Graph based on Figure 25. The numbers shown above, in brackets, are the event aftereffects durations and the number "2" shown in a blue bubble is the event temporal separation.

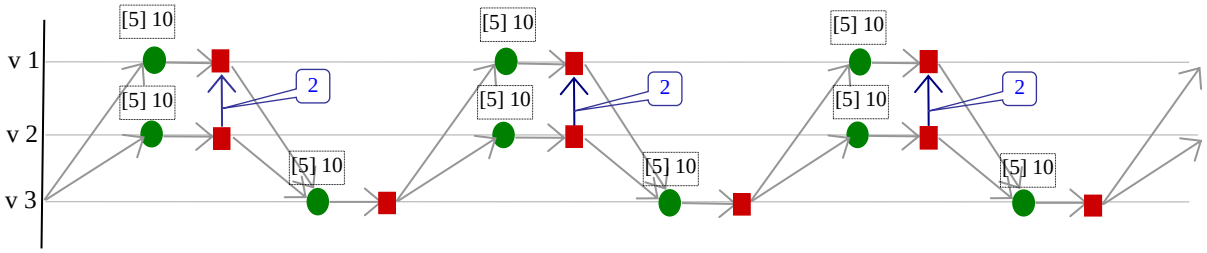


Figure 29. Concretized Signal Event Map, T-Intersection Example Figure 26

The following tables show the information regarding the example in Figure 26. **Table 6** indicates the transitional signal timings, and **Table 7** provides the intergreen times table for this example.

Table 6. Transitional Signals

v 1	0	3	2
v 2	0	3	2
v 3	0	3	2

Table 7. Intergreen Times Table

		R.O.W Given		
		v 1	v 2	v 3
R.O.W Lost	v 1			5
	v 2			5
	v 3	5	5	

Figure 30 illustrates the resulted signal diagram built by EBC framework (a visual representation of the signal events).

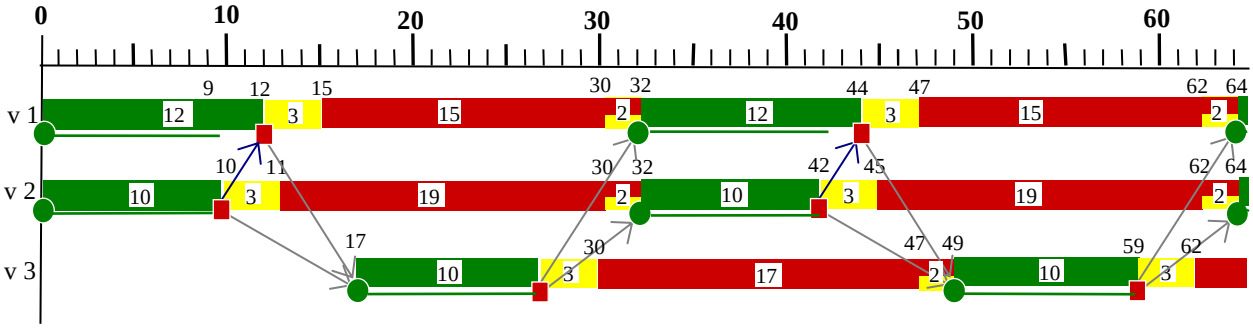


Figure 30. Visual Representation of the Signal Events in Figures 28 & 29

It is worth noting that, as can be seen in **Figure 30**, the duration of the main aftereffect of an event is not necessarily equal to the signal length. For example, the signal group "v 1" has 12 seconds of signal lengths whereas, the main aftereffect in this signal group lasts for 9 seconds. The difference between the duration of the signal and its main aftereffect is referred to as the "*Additional Signal Duration*".

Cyclical Comprehensive General Meta-Signal Events Precedence Graph: Cyclical Comprehensive General Meta-Signal Events Precedence Graph is a Comprehensive General Meta-Signal Events Precedence Graph that consists of a repeating segment, which is called a cyclic pattern.

Cyclic Pattern: Cyclic pattern is a repeating segment of Cyclical Comprehensive General Meta-Signal Events Precedence Graph. The prime cyclic pattern is a cyclic pattern that cannot be decomposed into a smaller cyclic pattern; otherwise, it is called a compound cyclic pattern.

It is important to highlight that all signal events must be connected to each other through precedence relations.

Figure 31 illustrates different variations of prime cyclic pattern of the same signal map shown in Figure . More information relating to the cyclic patterns is available in Appendix C.

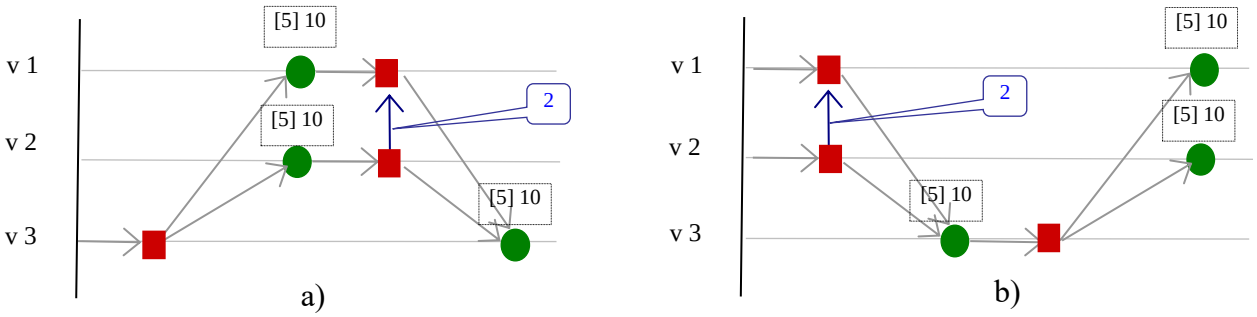


Figure 31. Variants of Prime Cyclic Pattern of the same Signal Map in Figure 26

Graphical User Interface – Events Map

To effectively utilize the signal events-based control methodology framework, it is crucial to understand how to define and create the events maps. Events maps serve as the backbone of the methodology, providing a representation of the signal events that occur in a system. This section provides a step-by-step guide on how to define events maps by using a graphical interface. By following this guide, users will be able to create regular events maps that can be utilized for effective events-based control.

To define the Abstract Events Maps, follow the steps below:

STEP 1: Go to the "Working with events maps" tab (see **Figure *32***).

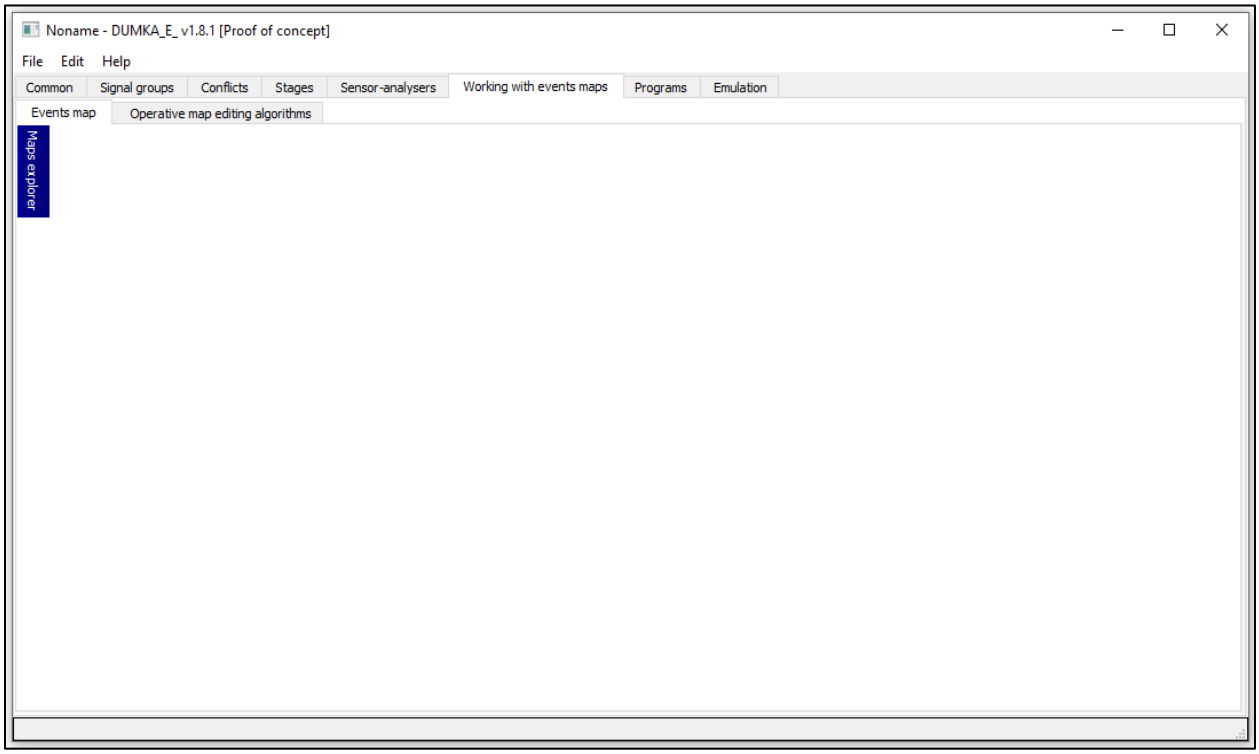


Figure 32. Working with Events Maps Overview

The tab is divided into two sub-tabs: "Events Map" and "Operative Map Editing Algorithms." The "Events Map" tab is used for developing desired abstract events maps, while the "Operative Map Editing Algorithms" tab is designed for working with operative signal maps and editing algorithms. Chapter Eight only focuses on the first sub-tab, the Events map.

STEP 2: click on the "Maps Explorer" button located on the left side of the tab to access the signal events map bank content browser. **Figure *33*** illustrates the events map bank browser.

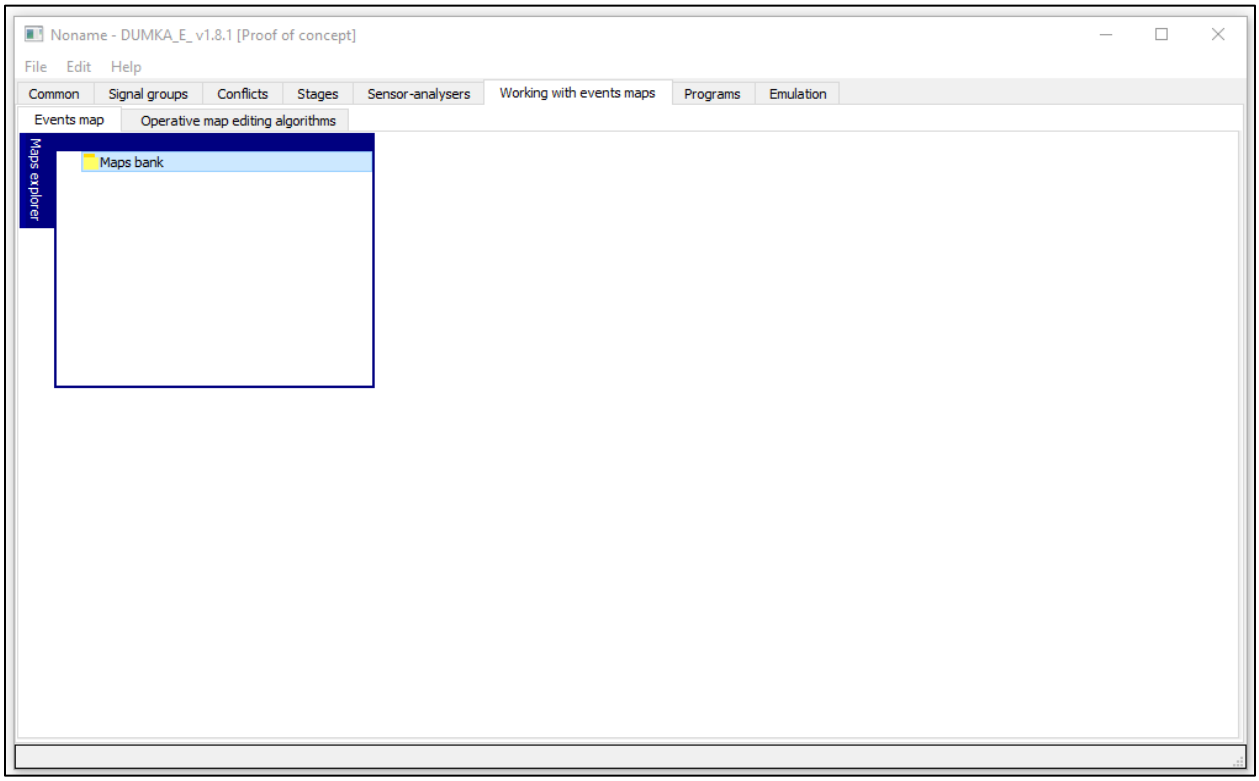


Figure 33. Events Map Bank Content Browser – Maps Explorer Button Location

STEP 3: To create a Map, after selecting the Maps bank folder, right-click on the Maps bank and open the context menu. **Figure *34*** illustrates how to create a map from the scratch.

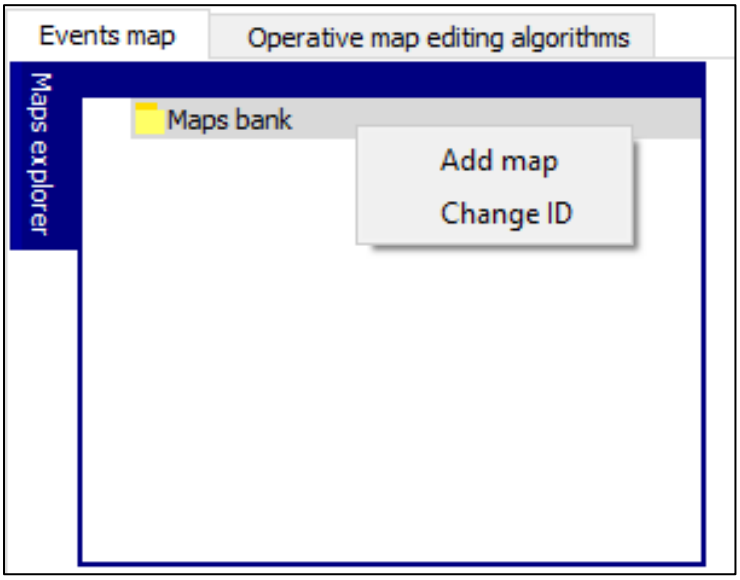


Figure 34. Events Map - Create a Map

STEP 3: To add a new signal events map to the project events maps bank, click on the “Add map” action with the left mouse button. The new signal events map will be created and automatically identified as “M1”. Its graphical representation will be displayed in the central area of the page as the start point. **Figure *35*** shows the location of the newly created map.

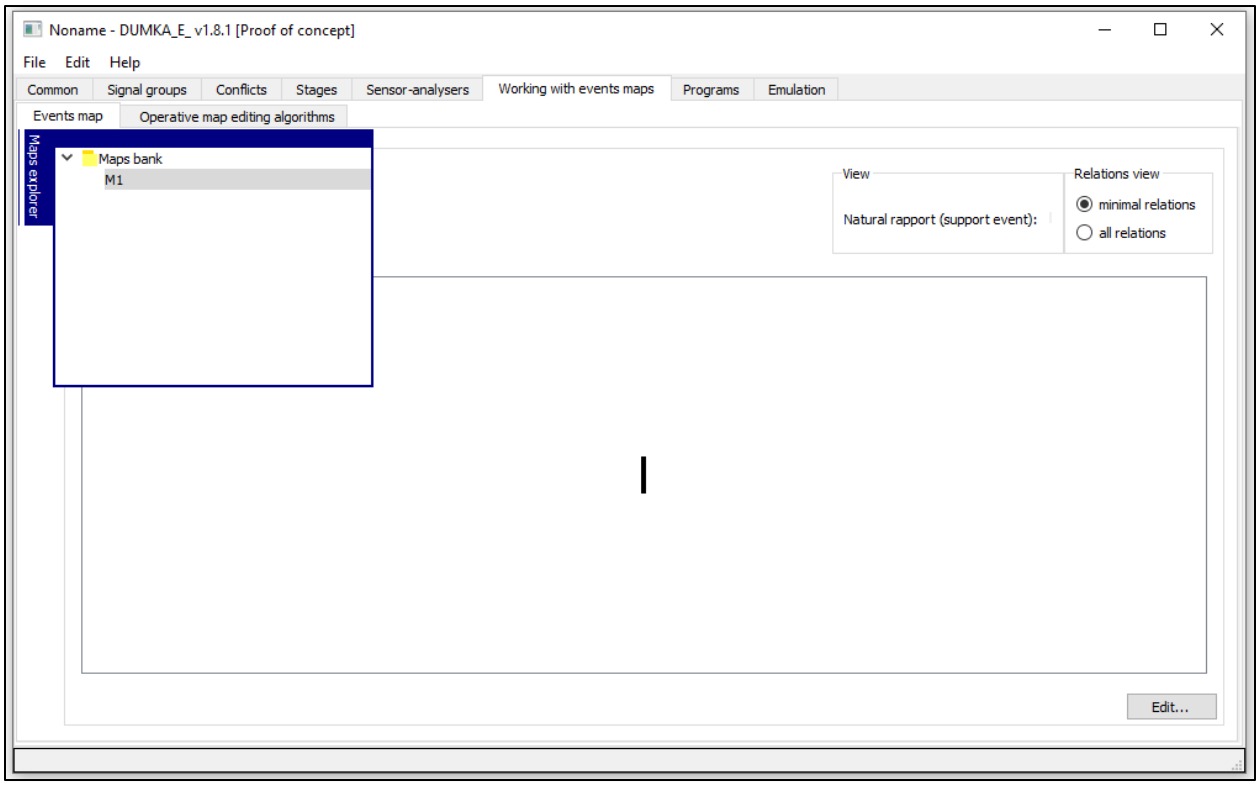
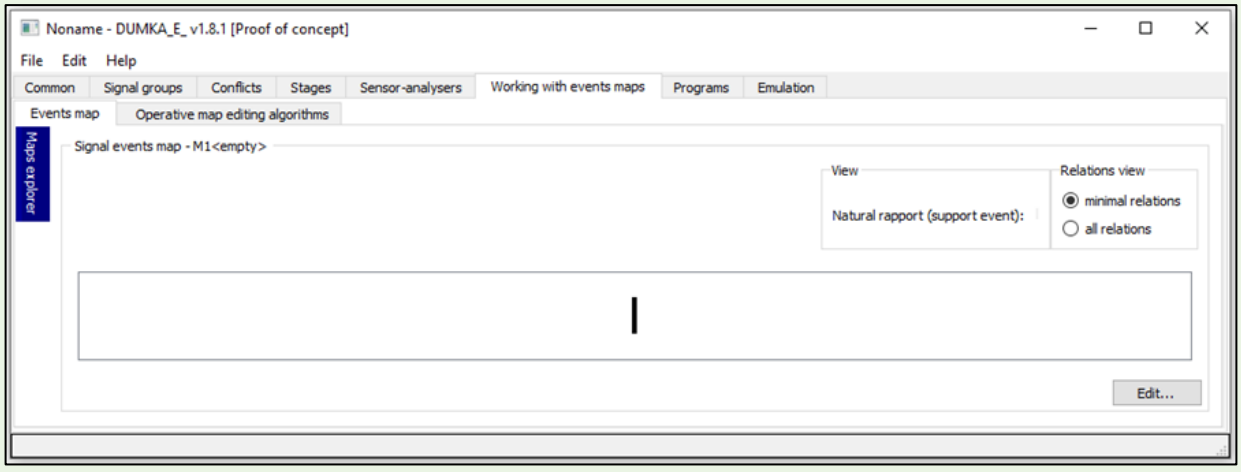
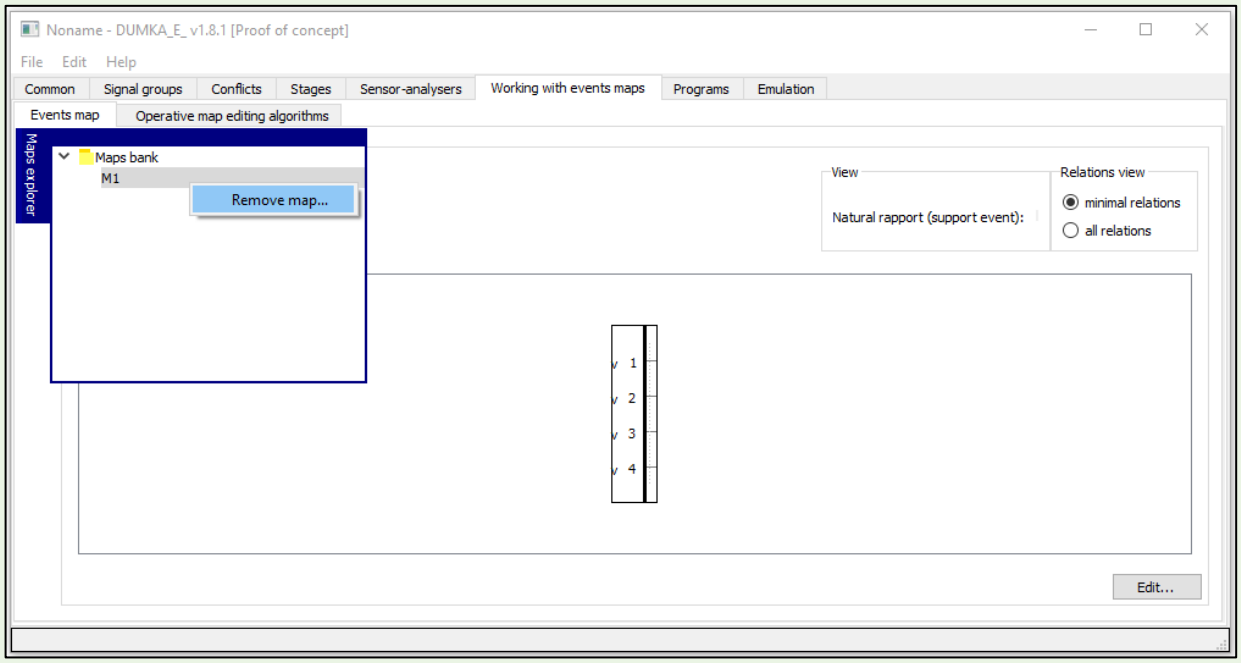


Figure 35. Graphical Representation of the Newly Created Signal Events Map

Remark 2: The graphical representation of the Events Map has been generated automatically upon creation, as shown below. The Signal events map is identified as "M1<empty>" initially, and as the first map is created in subsequent steps, the <empty> will disappear.



Remark 3: To remove a generated Events Map, first, click on the Map explorer on the left side of the screen. Then, within the Event maps section of the Events map bank, select the one that needs to be removed with a left click. Finally, right-click on the selected map (M1 in this example) and choose "remove map..." from the dropdown menu.



STEP 4: To open the signal events map editor, click on the “Edit...” button (see **Figure *36***).

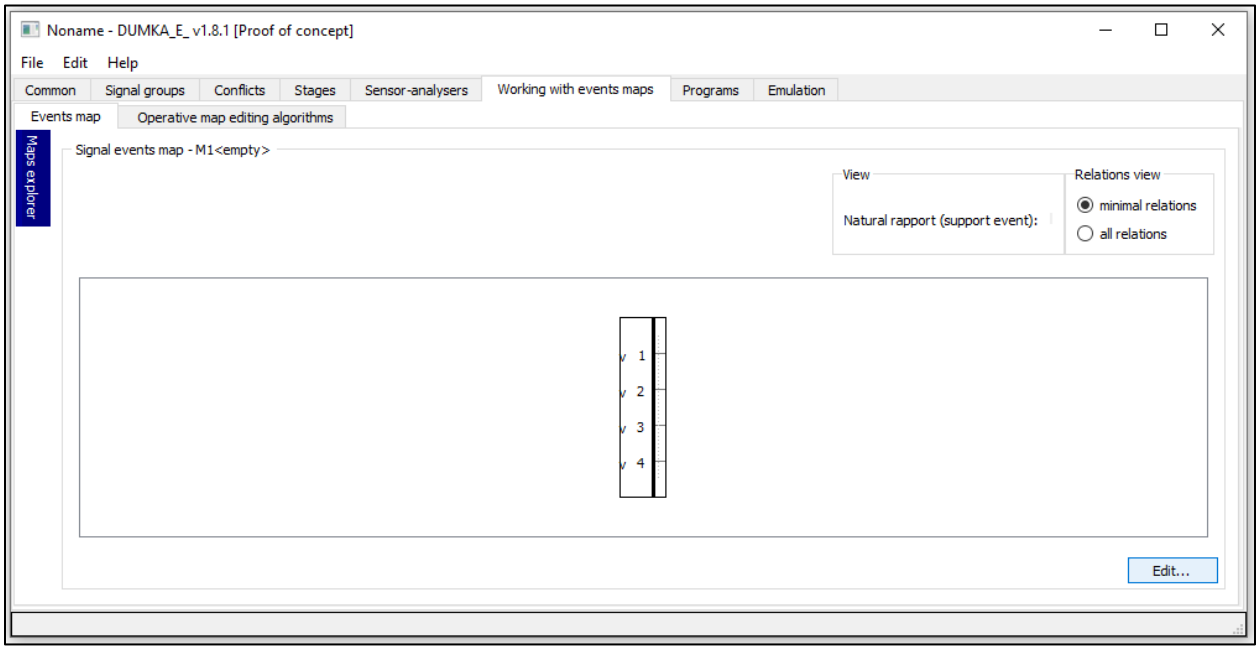


Figure 36. Access to Editor Section of Event Map

Figure *37* shows the “map editor” interface which consists of several options:

- The option to complete the relations automatically.
- The button to remove all relations simultaneously.
- The wellness status indicator was represented by a smiley face. The wellness status indicates whether the event map provided is correct or not.
- The potential empty map for creating the desired signal events.

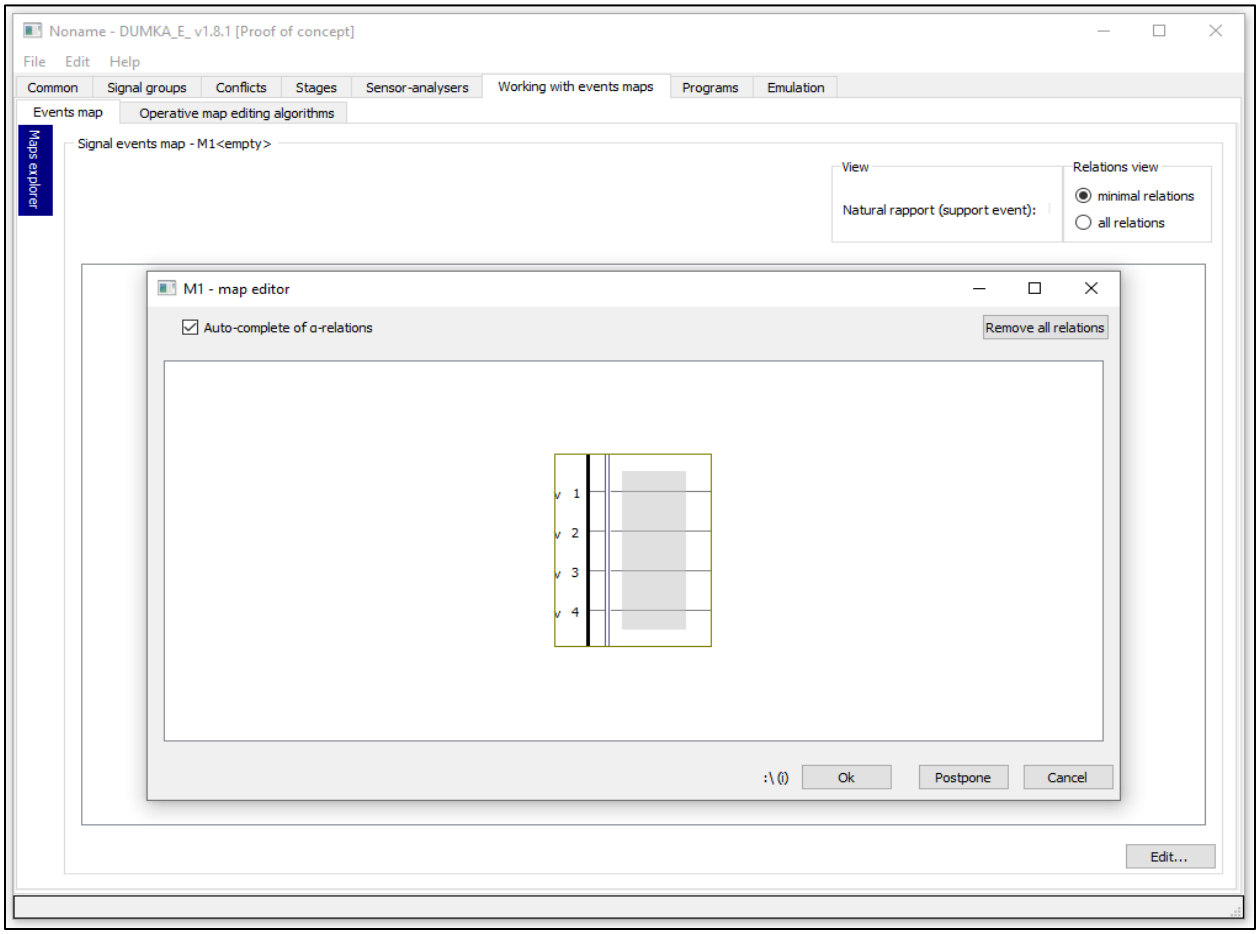


Figure 37. Map Editor Interface

The signal events map editor is based on the concept of “Events t-disposition”. The term “Events t-disposition” refers to the arrangement of signal events on a timeline based on their relative occurrence order (i.e., which event happened before another event). This arrangement does not consider the time interval between events.

For an event's t-disposition to be compatible with a given signal event map, the order of event occurrences in the t-disposition layout must match the corresponding order of event precedence in the map. In other words, if event A is supposed to happen before event B in the signal event map, then the events t-disposition must also show that event A occurs before event B on the

timeline. It should be noted that the t-disposition representation is composed of a set of event columns that are colored in gray (see the gray column in **Figure *37***).

STEP 5.1: To add a new event to the t-disposition column, double-click the left mouse button on the location where the new event should appear. The new event will then be added to that location. **Figure 38** shows the two consecutive steps of adding a new event to the map.

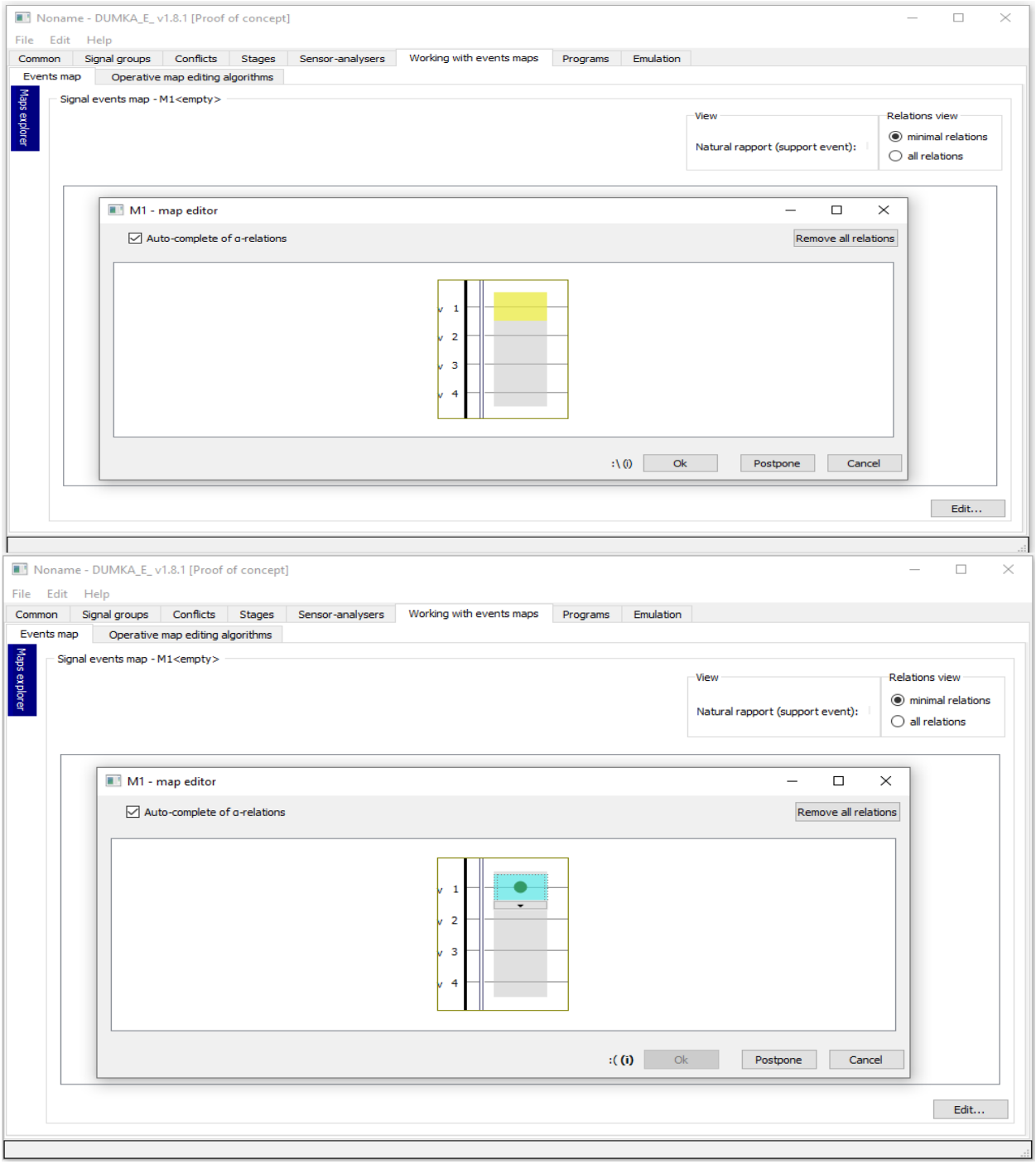


Figure 38. Adding a New Event to the T-Disposition Column

STEP 5.2: To modify event properties, position the mouse cursor over the event icon that needs to be edited. This will cause the event toolbar to be displayed directly beneath the event icon. **Figure *39*** illustrates the available options for the events. A user can choose the desired event from the available options.

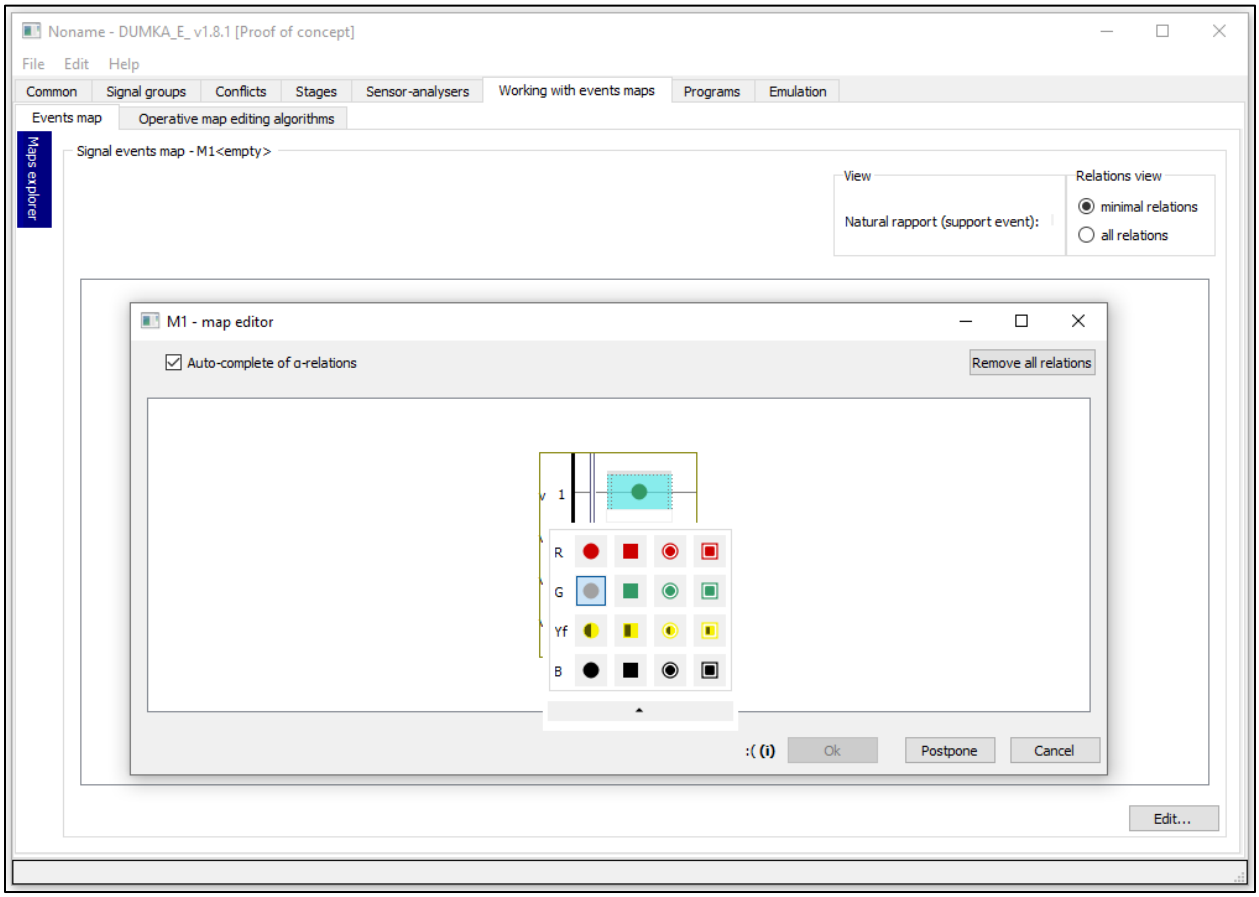
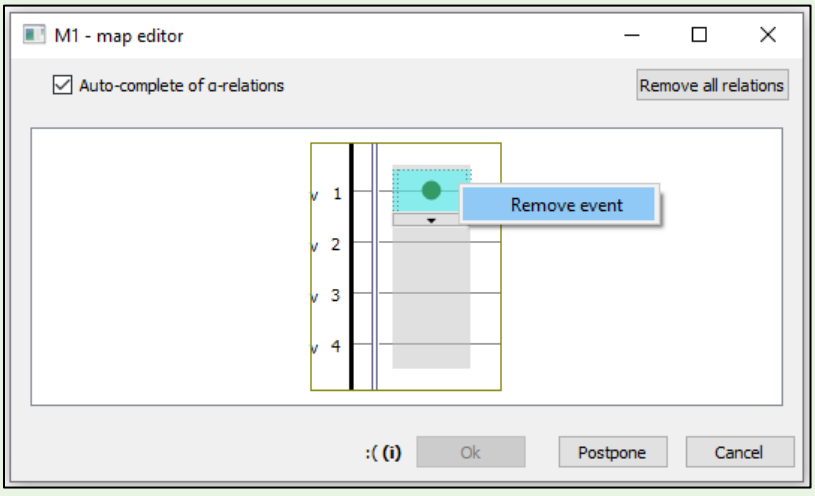


Figure 39. Editing Event Features

Remark 4: To delete an event, left-click on the event to select it, then right-click to open the context menu and choose the "Remove event" action.



STEP 5.3: To insert a new column into the t-disposition, place the mouse cursor where you want the new column to be inserted, either before or after an existing column. The space where the new column will be inserted will be highlighted (**Figure *40***).

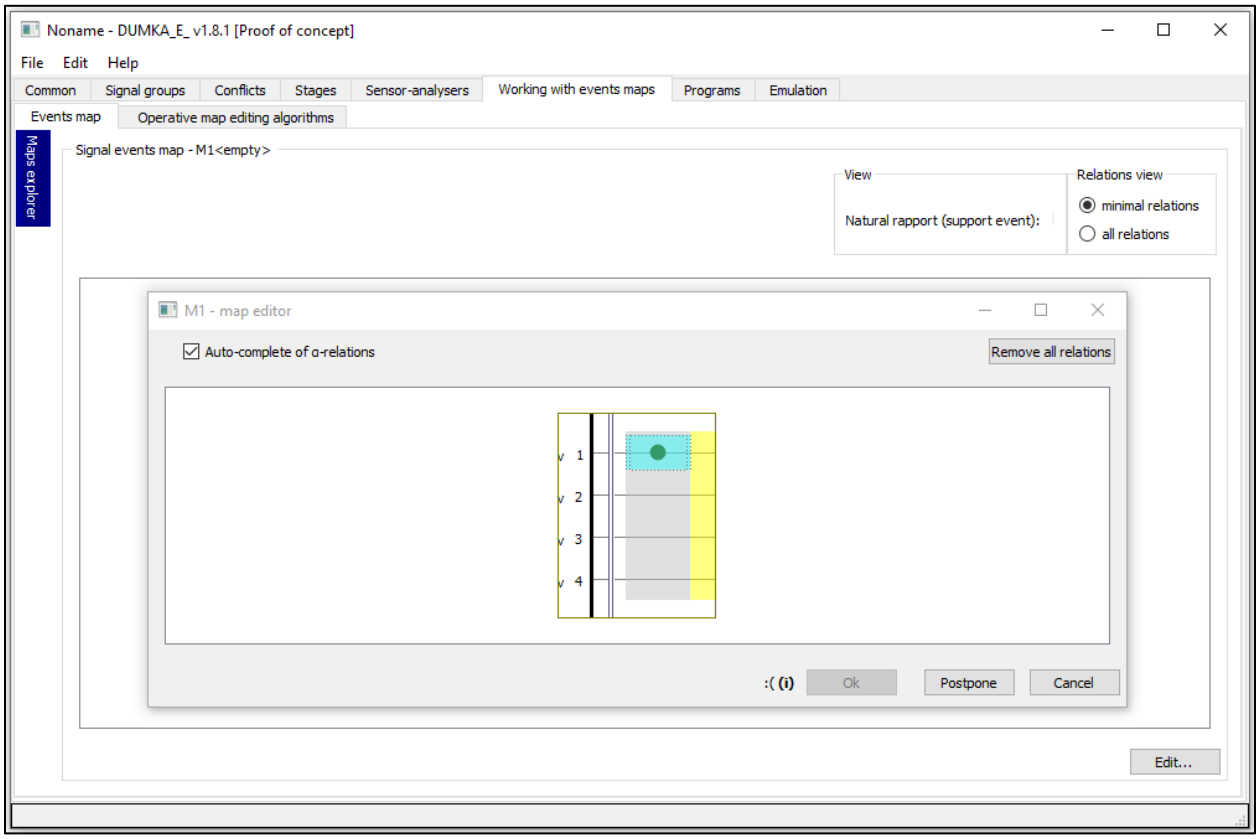


Figure 40. Adding a New Column to the T-Disposition

To insert a new t-disposition column, double-click the left mouse button on the highlighted area where you want the new column to appear, either before or after an existing column. This action will insert a new empty column at the selected location (**Figure *41***).

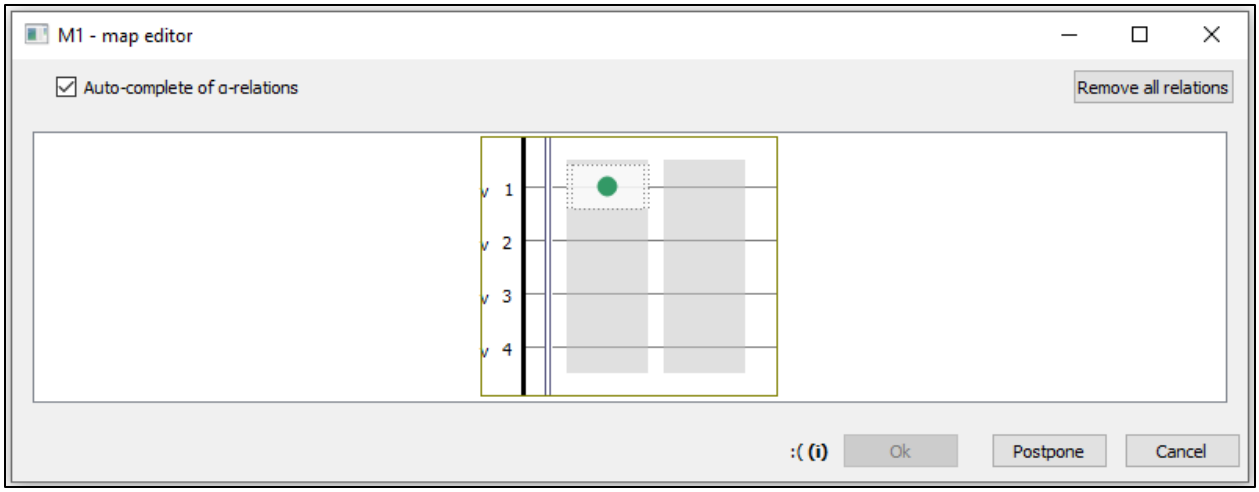
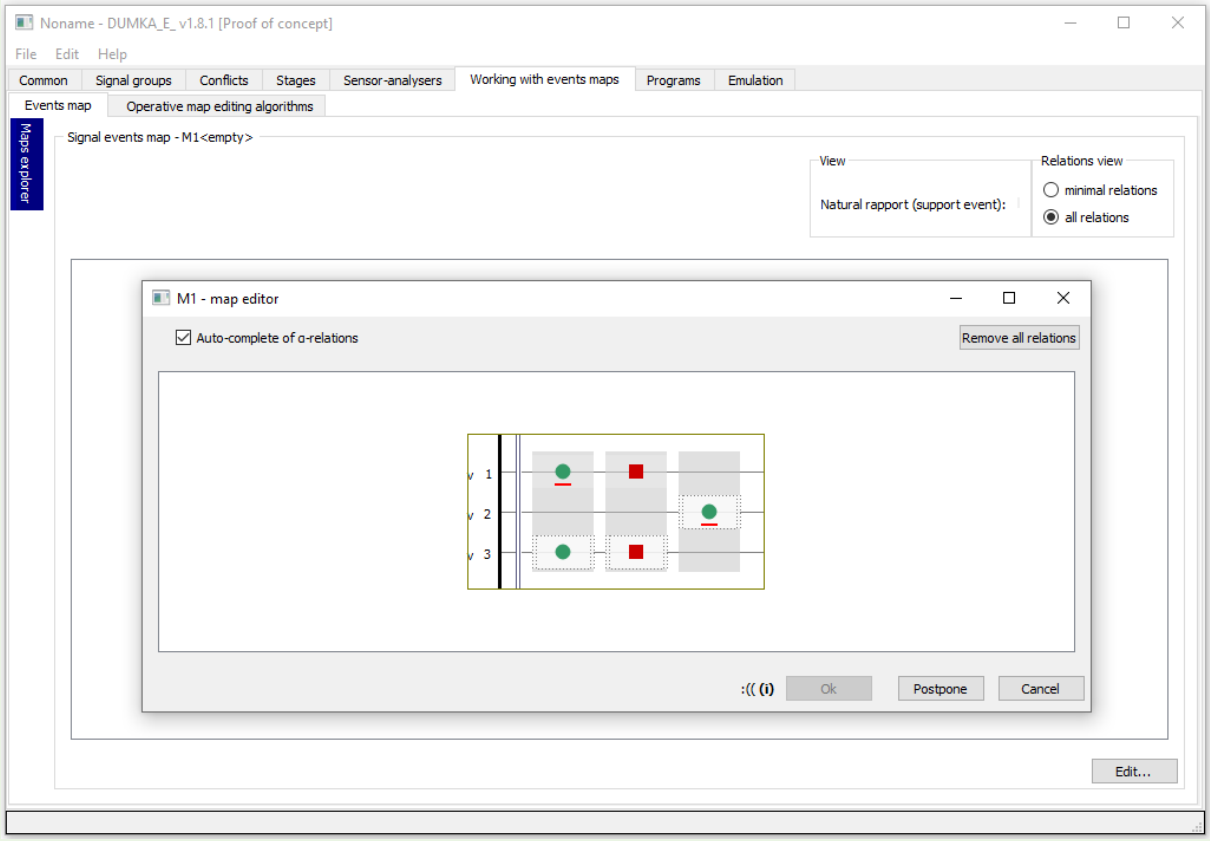


Figure 41. Adding a New Column to the T-Disposition

Remark 5: Events may conflict with each other in an events map if placed in a certain order, which can cause errors in the signal events map. For example, if the event map allows for the green signal to occur before the red signal in a conflicting signal group, it can cause conflicts. The editor will highlight these potentially conflicting events by using red lines and underlines while constructing the events map. This will alert users about potential conflicts and allow them to make appropriate adjustments to prevent errors in the signal.



To avoid possible conflicts, it is necessary to add the appropriate precedence relations.

Remark 6: To enable the automatic construction of precedence relations, simply select the "Auto-complete of α-relation" checkbox above the "map editor" window. When this feature is enabled, the editor will analyze potential conflicts and attempt to add the minimum necessary precedence relations to the map to eliminate these conflicts based on the existing t-disposition and relations automatically.

STEP 5.4: To view the current precedence relations, left click on the event for which you want to see the relations. The relations will be displayed as arrows near the events (see **Figure *42***).

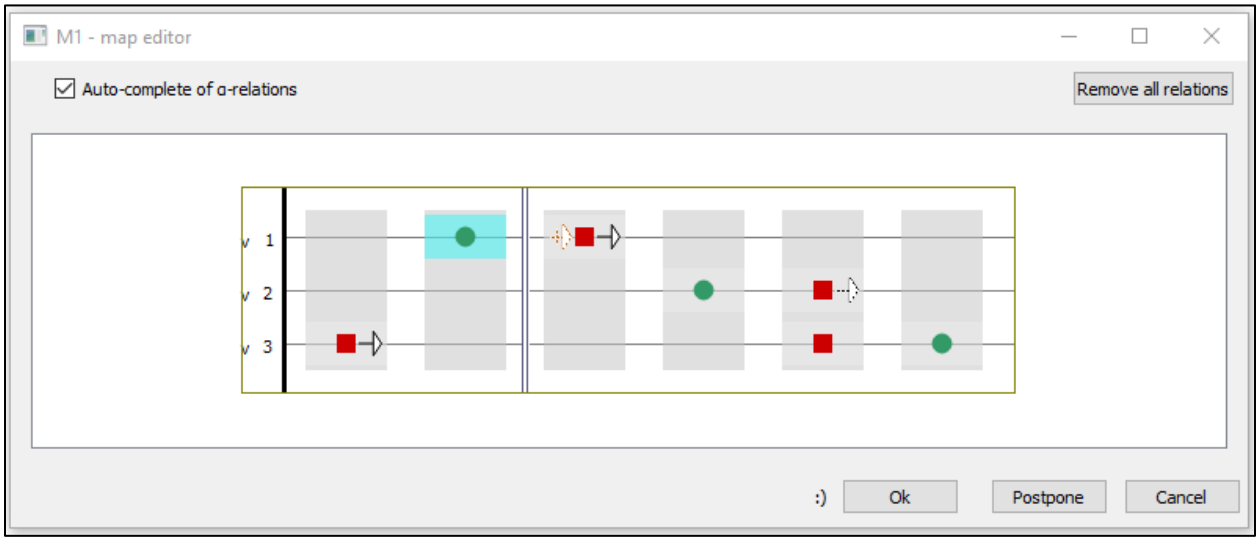
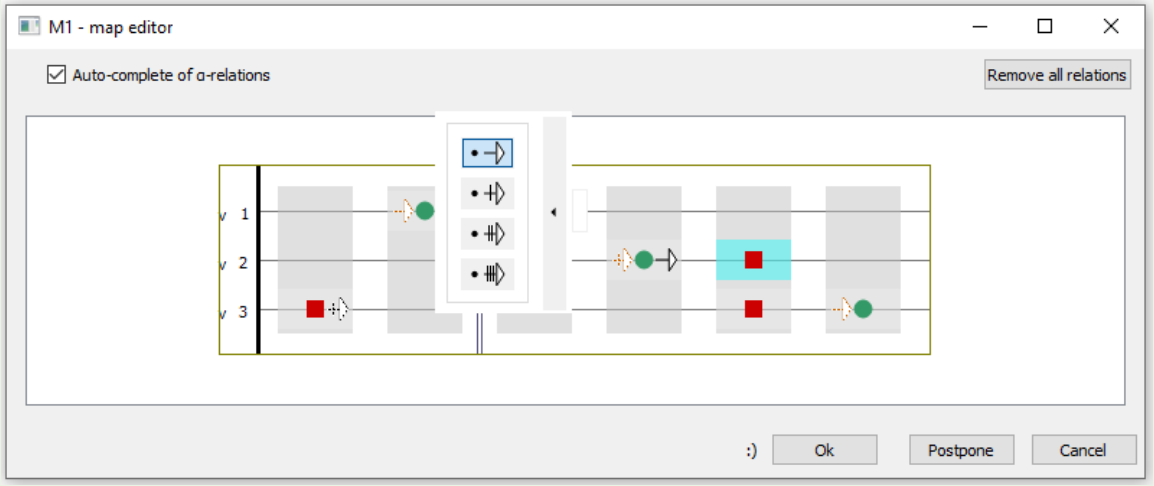


Figure 42. Precedence Relations Between Events

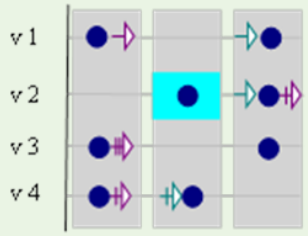
Remark 7: To add a precedence relation manually, follow these steps:

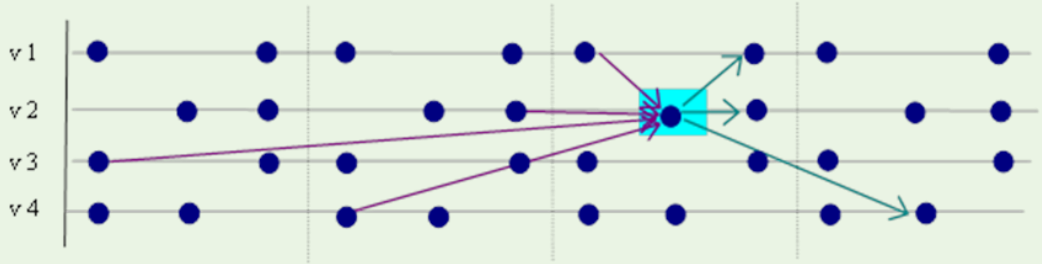
1. Select the event that should be the starting point of the relation by clicking it with the left mouse button.
2. Move the mouse over the event that should be the endpoint of the relation. Near the icon, a button with "Relation" stock will appear.
3. Click this button and choose the type of desired relation to add from the available options. The relation will then be added to the map.

Figure below shows the available options.



Remark 8: The meaning of the relation arrows is explained below:





The following outlines the criteria that must be satisfied for a constructed map to be considered valid. Here is a brief explanation of each of the four conditions:

- All events must be associated with signal groups that can be implemented (i.e., the map must be feasible).
- The map must be 3g-strongly connected, meaning, loosely speaking, there must be a path from any event in the first cyclic pattern to itself in the third cyclic pattern that includes any other event from the cyclic pattern.
- The maximal weight urgency of all events on the map must be positive.
- The map must have a universal reference event, which serves as a common reference point for all other events on the map.

The following outlines the criteria that must be satisfied for a constructed map to be considered as well-defined. Here is a brief explanation of each of the three conditions:

- The map must be valid, meaning that it must satisfy the criteria outlined in the previous passage.
- The map must have at least one event associated with each signal group.
- The map must not have any potentially conflicting realization events. In other words, there should be no events on the map that, when implemented, could potentially conflict with each other.

It is important to note that only valid maps can be included in the project. Additionally, only well-defined maps can be used for controller programming.

The EBC DUMKA_E can show the current state of wellness of the signal events map by an “emoji face” icon next to the “Ok” button. If the map is well-defined, a smile will appear next to the “Ok” button, and the “Ok” button will be activated. **Figure *43*** shows that situation.

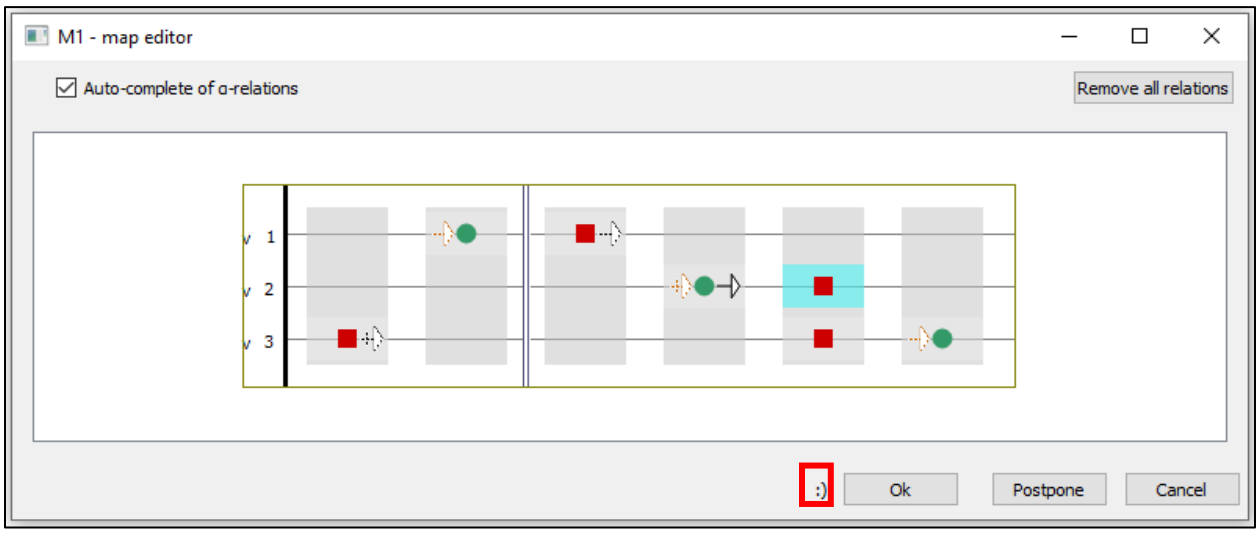
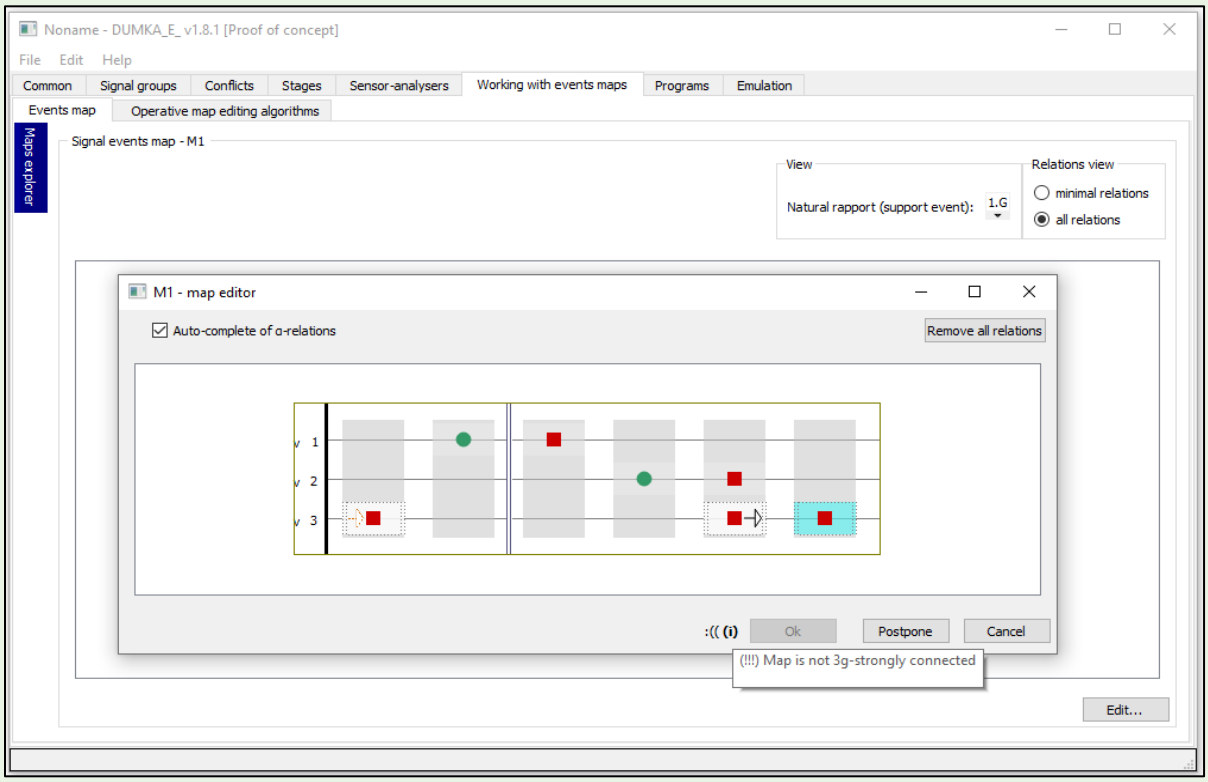


Figure 43. Wellness Status

Remark 9: In a case that the events map is not valid or well-defined, the wellness status changes according to the situation. Figure below shows a situation where the map is not 3g-strongly connected. More details about the error are available by hovering over the attached tooltip.

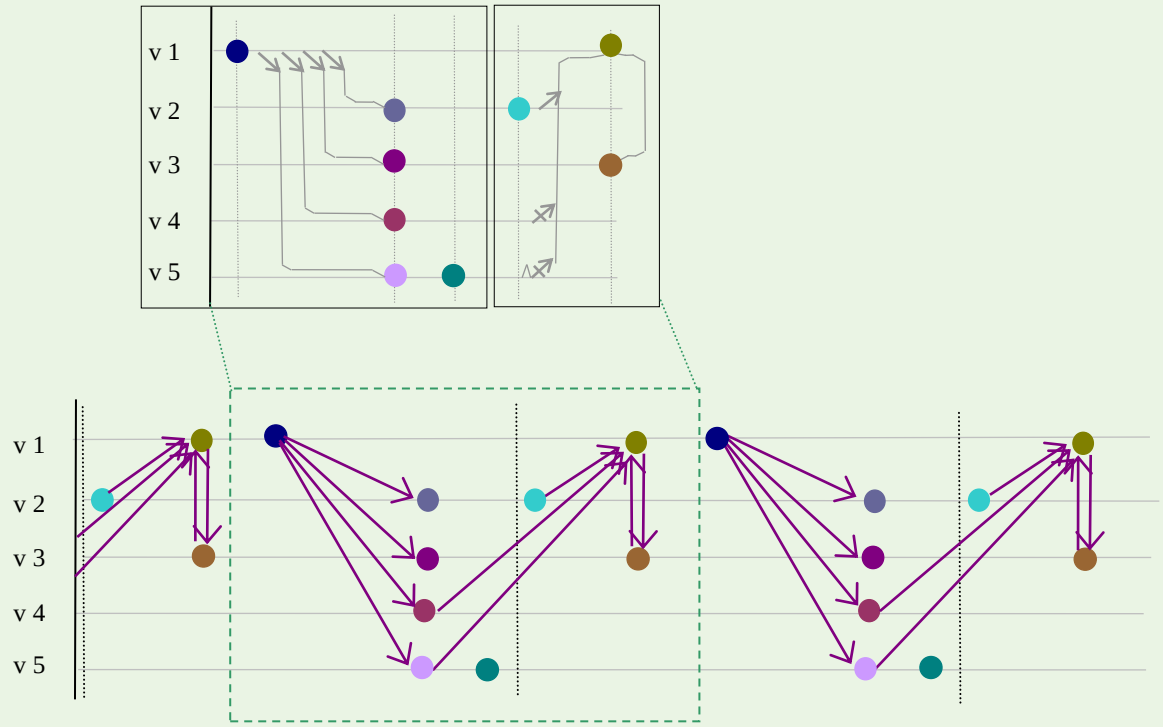


Remark 10: In the situation where none of the events are 3g-connected, the corresponding events can be found by a dashed box around them.

Remark 11: Once the constructed map is at least valid, click on the "Ok" button to finish editing and include the map edition in the project.

Remark 12: The EBC DUMKA_E provides the option to postpone map editing and exit the editor. Click the "Postpone" button, between "Ok" and "Cancel" for that purpose.

Remark 13: The meaning of the denotations is explained below:



To perform undo or redo on the latest operation, navigate to the main menu, click on "Edit", and select "Undo" or "Redo".

To summarize, this chapter has covered how to work with events maps in the signal events-based control methodology framework. By analyzing the events map, users can gain insight into how the system functions and how it can be controlled through event-based mechanisms. The section focused on specifying an abstract signal events map, an essential step in building a signal diagram. This included creating an events map by defining events and their attributes, specifying event relationships, and organizing events into hierarchies.

9. Operative Map Editing Algorithms

This chapter will explain the operative signal diagram, the signal diagram for adaptive signal control, and how to build them in the EBC DUMKA_E methodology framework. In the context of signal events-based control methodology, adaptive control is regarded as the process of building the signal diagram (the operative signal diagram) in real time for a given transport situation during the operation of the road traffic controller. This process is referred to as operative adaptive control. **Figure 44** illustrates the steps involved in building the adaptive control strategy in the framework of the signal events-based control (EBC) concept.

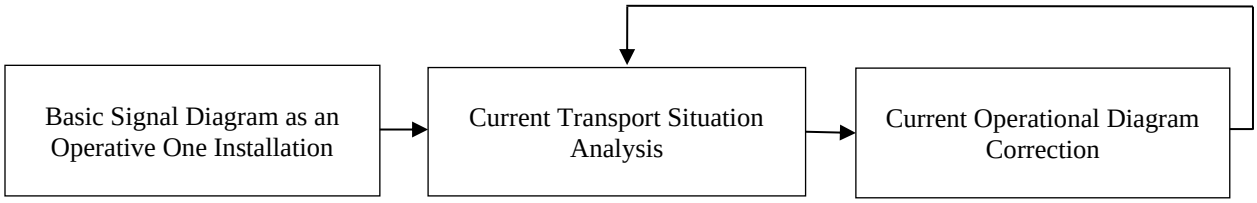


Figure 44. Scheme of Adaptive Control Process

1- Basic Signal Diagram as an Operative One Installation

This is the first step in the process of adaptive control. The basic signal diagram is a diagram that shows the basic layout of the traffic signals in each area. It is used as a starting point for the adaptation process.

2- Current Transport Situation Analysis

The next step is to analyze the current traffic situation. This is done by using sensors (detectors) data. The goal of this step is to identify the current situation with the traffic flow.

3- Current Operational Diagram Correction

The final step is to make the necessary changes to the basic signal diagram. This is done by using the information from the current transport situation analysis. This step aims to create a new operative signal diagram that is optimal for the current traffic situation.

As is shown in Figure 44, steps two and three are parts of a continuous cycle of analyzing the current situation and reflecting on the changes to make corrections.

The operative signal diagram is built from the operative editable events map. The operative signal diagram consists of two parts: i) the “past” part and ii) the “planned” part. The past part consists of the events that have already occurred. The planned part consists of the events that are planned to occur in the future. In the following, some of the important concepts are explained in detail:

Occurred Event: The occurred event is an event that has already occurred.

Actual Event: The most recent signal event that has occurred (It should be noted that every signal group has its own actual signal event).

Planned Event: A planned signal event is a signal event whose execution time is planned but may change (e.g. based on relevant logic).

Pre-Occurred Event: The pre-occurred event is a planned event that is to occur in a second.

Signal Horizon: Signal horizon is an operative signal diagram one-second fragment specifying the control signals to be exhibited to the road users for the coming second.

Figure 45 depicts the past part, the signal horizon, and the planned part in an operative signal events map.

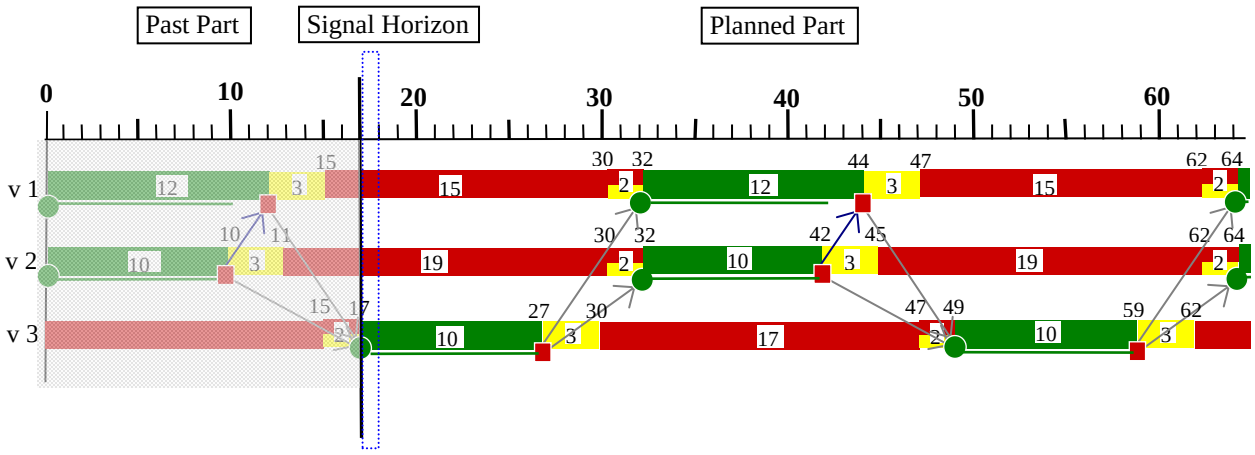


Figure 45. Operative Signal Events Map

This section of the EBC DUMKA_E operative signal events map provides an editable signal events map that allows for modifying events, such as moving events around or adding or removing time from events. This type of map is useful for defining modifications of the operative signal diagram.

Event Left Displaceability: The left displaceability is responsible for whether, at signal diagram rebuilding, the occurrence time of the event may become less than it was before rebuilding.

Additional Signal Time Absorbability: The event additional signal time absorbability says whether additional signal time (signal time exceeded the event main aftereffect duration) will be included in the event main aftereffect duration after “reconfiguration”.

The attributes of an event are represented by icons that accompany the event icon. The icons are as follows:

A triangle placed on the left of the event icon indicates that the displaceability attribute has the value “not displaceable to the left.” This means the event cannot be moved to the left on the timeline.

A “v”-shaped line placed on the right of the event icon indicates that the absorbability attribute has the value “absorbable.” This means that the event can absorb additional signal time after it has been reconfigured.

Remark 1: The absence of an icon indicates that the attribute has the default value. For example, the absence of a triangle on the left of the event icon indicates that the displaceability attribute has the default value of “displaceable to the left.”

Remark 2: The icons are used to provide a visual representation of the attributes of an event. This can be helpful when planning and managing events, as it allows users to quickly see the attributes of an event and how they might affect the timing of subsequent events.

The following refers to the process of editing the event map. It explains that changing the primary and secondary attributes of events is considered a type of “**editing action**”. A set of these editing actions are called an “**edition**,” and the process of performing these actions is called “**editing the event map**.”

The main editing actions are provided in **Table 8** below:

Table 8. Main Editing Actions

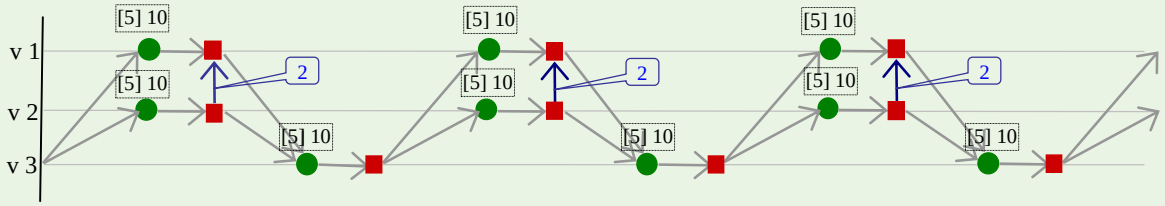
	Action	Graphical representation
Changing Qualitative Attributes	Setting the qualitative attribute's value	Same as the representation of attribute's value to be set
Changing Quantitative Attributes	1) Decreasing the main aftereffect duration to minimum aftereffect duration.	A triangle placed on the right side of the event icon: 
	2) Decreasing the main and minimum aftereffect durations to minimum feasible value.	A double-triangle placed on the right side of the event icon: 
	3) Decrementing (decreasing by 1) the main aftereffect duration while considering the minimum aftereffect duration.	A circle attached to a triangle placed on the right side of the event icon: 
	4) Decrementing the absorbed portion of the main aftereffect duration.	A circle attached to a triangle placed on the rightest side of the event icon: 
	Combination of 3 and 4.	Double circles are attached to the double triangles inside the icon: 
Others	Completing the finalization of primary event attributes (urgency and transparency) by marking them as "protected from further modification."	Positioning the letter «F» on the bottom right of the event icon.

Remark 3: To represent a partial combination of qualitative attributes editing actions for a single event, an "asterisk", ✱ , is used in place of the respective aspect of the event's graphical representation if the action is not included in the combination. Such as:

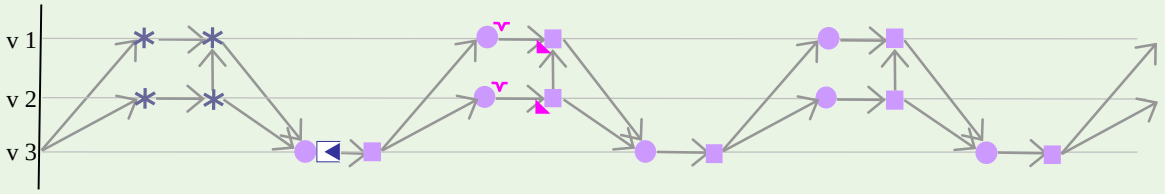
- The geometrical form of the event icon as an "asterisk" indicates the absence of the urgency attribute setting action in the combination.
- An "asterisk" within the event icon signifies the absence of the transparency attribute setting action in the combination.
- An "asterisk" at the secondary qualitative attributes icons places denotes the absence of these attributes setting actions in the combination.
- An "asterisk" at the event icon place implies the absence of the qualitative attributes editing actions.

To differentiate between the graphical representation of an events map and its edition, the latter utilizes a lilac-colored filling for the event iconic form. For instance:

In the following, there is an example of a comprehensive meta-signal event precedence graph with its corresponding events map edition representation.



a) Comprehensive meta-signal event precedence graph



b) Edition representation of above graph

In the framework, map editions are constructed based on a cyclic pattern, specifically referred to as the “edition cyclic pattern.” This cyclic pattern is known as the t-cyclic pattern, identifying the feasible event as the starting event. From this starting event, all events within the cyclic pattern can be encountered solely on directed paths, either in the forward direction (t+ -cyclic pattern) or in the inverse direction (t- -cyclic pattern). The distinction lies in whether the event holds for the forward or inverse direction, as illustrated in the figure below. The example below illustrates

the t-cyclic patterns with possible starting events indicated by the corresponding directed triangles include (**Figure 46**):

a) Scenario where both t+ -cyclic and t- -cyclic patterns exist.

b) Scenario where only a t+ -cyclic exists.

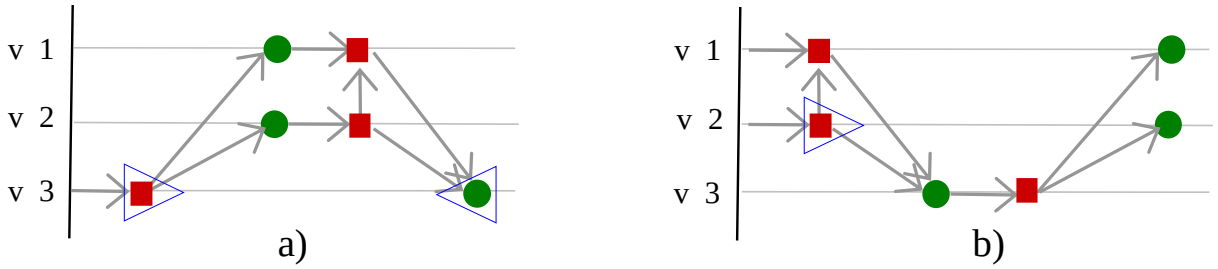


Figure 46. T-cyclic Pattern Types

Remark 4: A t-cyclic pattern from which the starting event is excluded and substituted with its closest similar event in the direction of the cyclic pattern orientation is a punctured t-cyclic pattern.

To identify the edition cyclic pattern, the following information must be provided:

1. Details about the tie-pole event, which is the starting event of the cyclic pattern in the operative events map.
2. Information about the orientation of the cyclic pattern (t+ - or t- - cyclic pattern).
3. The count of prime components, particularly if the cyclic pattern is compound.

This information, especially points 1 and 2, is visually represented on the signal map edition through the *tie-pole indicator*.

Table 9. Typical Representation of Editions for a Regular Map

	common t-cyclic pattern	punctured t-cyclic pattern	tie-pole event
t^+ -cyclic pattern			last occurred event
			first planned event
			second planned event
t^- -cyclic pattern			last occurred event
			first planned event
			second planned event

It should be noted that the operative map is guaranteed to include only two similar events in the part of *occurred events* available for editing.

Additionally, the execution of the edition is discarded if any of the following conditions are met:

- The tie-pole event is not yet in the operative events map (for example, it is at the beginning of the program).
- It leads to an operative signal events map with an undrivable signal diagram (for example, due to predicted movement conflicts).
- It contains an editing action for an event that is no longer present in the operational signal diagram.
- It attempts to modify finalized attributes.
- It attempts to change the event signal transparency attribute from "*transparent*" to "*opaque*."
- It is manifolded and contains an event signal transparency resetting action.
- It is manifolded and contains actions that decrease main and minimum aftereffect durations to the minimum feasible value.

Graphical User Interface – Algorithms and Editions

To effectively utilize the signal events-based control methodology framework, it's essential to understand how to define and create signal events map editing algorithms. This section provides a step-by-step guide on defining events map algorithms using a graphical and editing algorithms interface. Following this guide, users can create events maps algorithms suitable for effective events-based control. The "Operative Map Editing Algorithms" consists of two sub-tabs; Algorithm flow-chart and Operational map editions. **Figure 47** shows the overview of the "Operative Map Editing Algorithms".

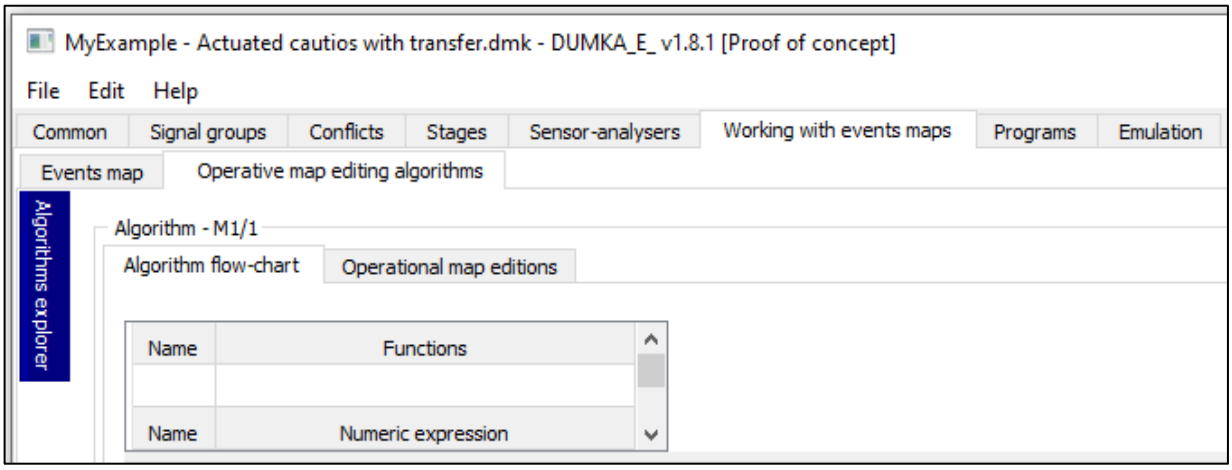


Figure 47. Overview Working with Events Maps Overview

To define the Events Maps Editing Algorithms, follow the steps below:

STEP 1: Navigate to the "Working with events maps" tab, then click on the "Operative map editing algorithms" tab (**Figure *48***).

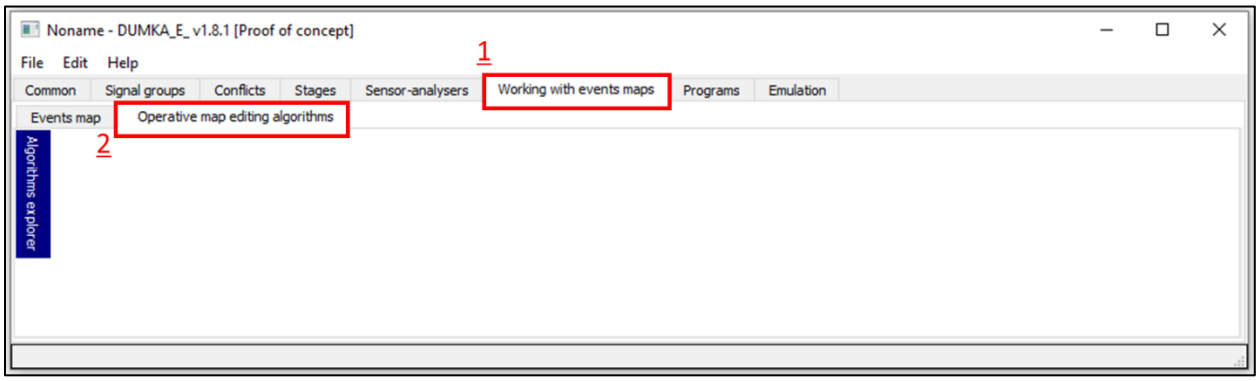


Figure 48. Access to Operative Map Editing Algorithms

STEP 2: Click on the "Algorithms Explorer" button located on the left side of the page for the maps editing algorithms banks. This will bring up a list of algorithm banks for the existing signal events map. **Figure *49*** illustrates that the signal events map does not have any algorithms related to it.

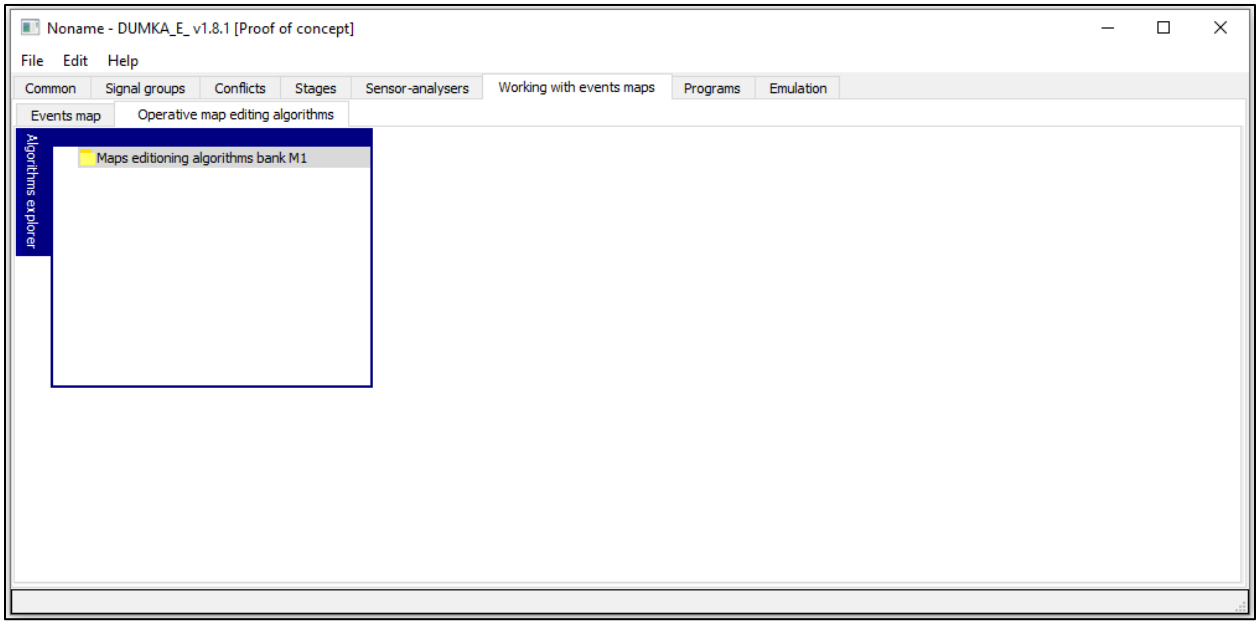


Figure 49. Accessing Algorithm Banks for Signal Event Map Editing

To access the desired algorithm, select the corresponding signal events map folder in the opened explorer window by left clicking with the mouse. Then, right-click the folder to call up the context menu.

STEP 3: To add a new algorithm, simply right-click the folder and select “Add algorithm” (**Figure *50***). This action will create a new empty algorithm in the events map algorithms bank, automatically identified as '/1'.

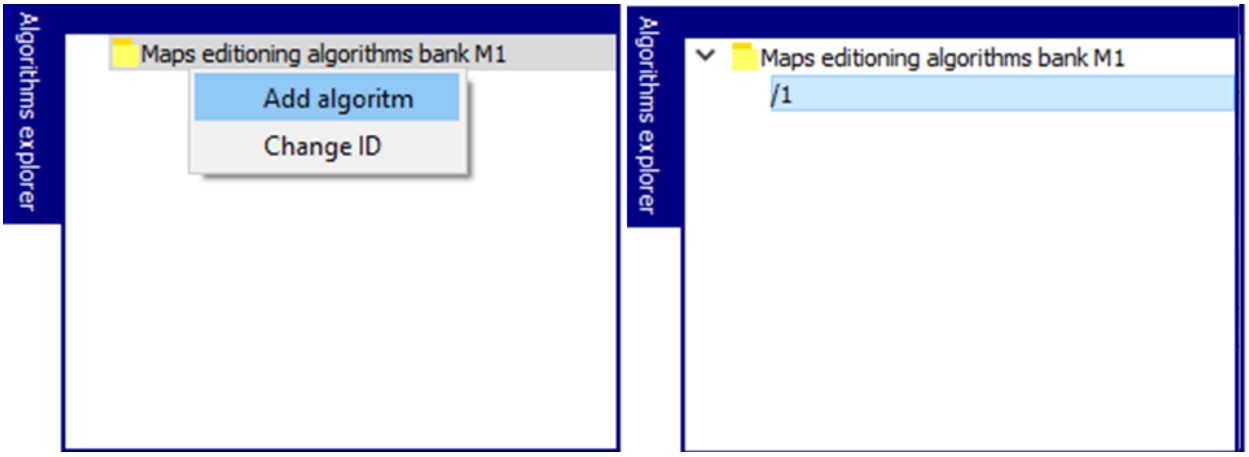


Figure 50. Add a New Algorithm

The graphical representation of the new algorithm will then appear in the central area of the page (see **Figure *51***).

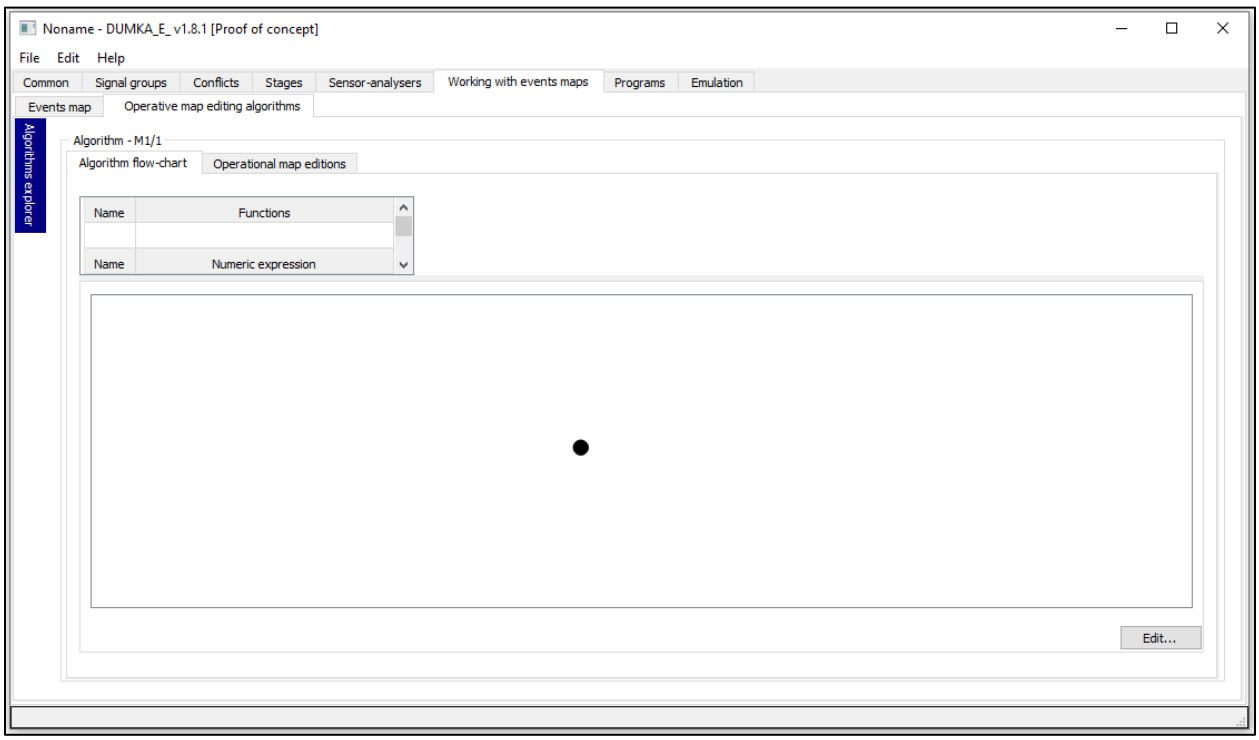


Figure 51. New Algorithm Representation

STEP 4: To begin, it is recommended to first create editions of the signal map that will be used in the algorithms. To do so, a user should switch to the other sub-tab “Operational map editions”.

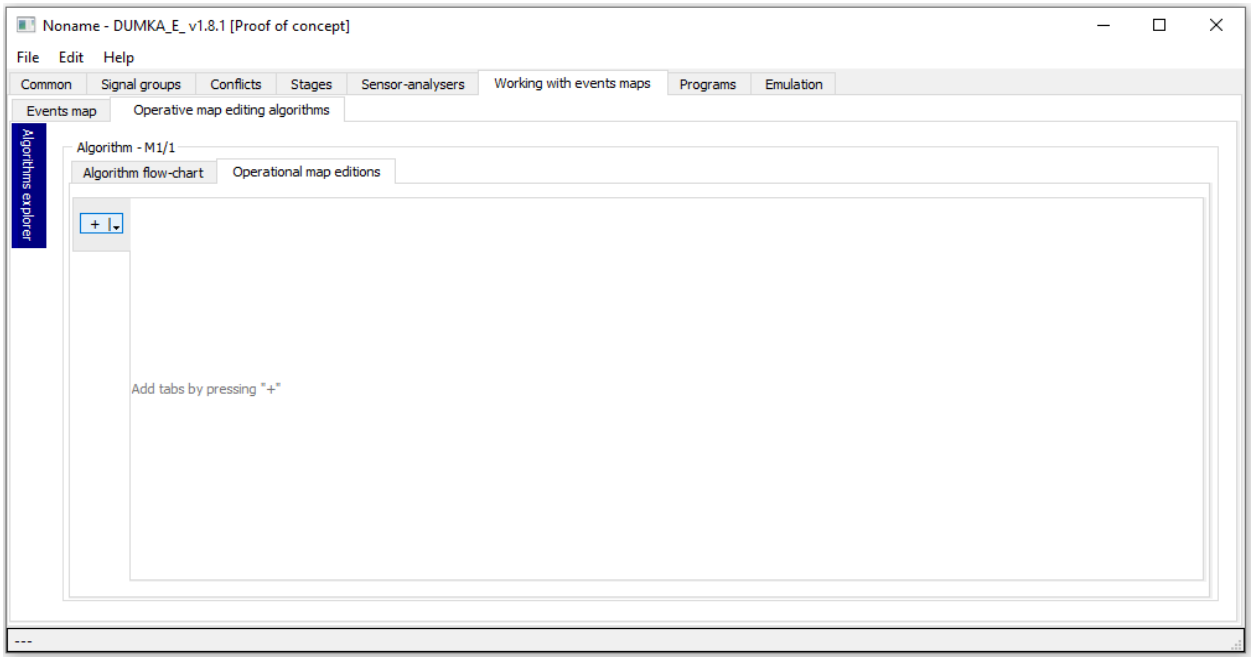


Figure 52. Creating Operational Map Editions for Algorithm Development

STEP 5: Click the “+” button to create a new edition. This will open a dialog for creating a new edition (shown in **Figure *53***).

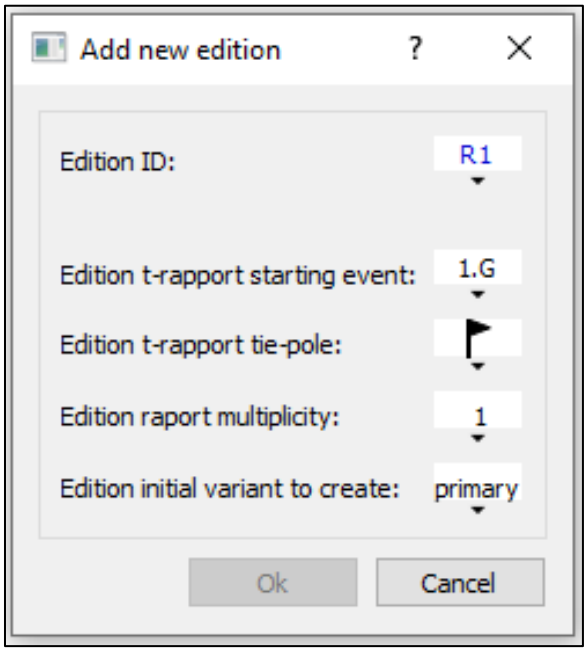


Figure 53. Dialog Box for Adding a New Edition

From the dialog box, select the desired settings and click “Ok”. The newly created edition will be added to the set of algorithm editions, and a reduced graphical representation of it will open in the central area of the page (**Figure *54***).

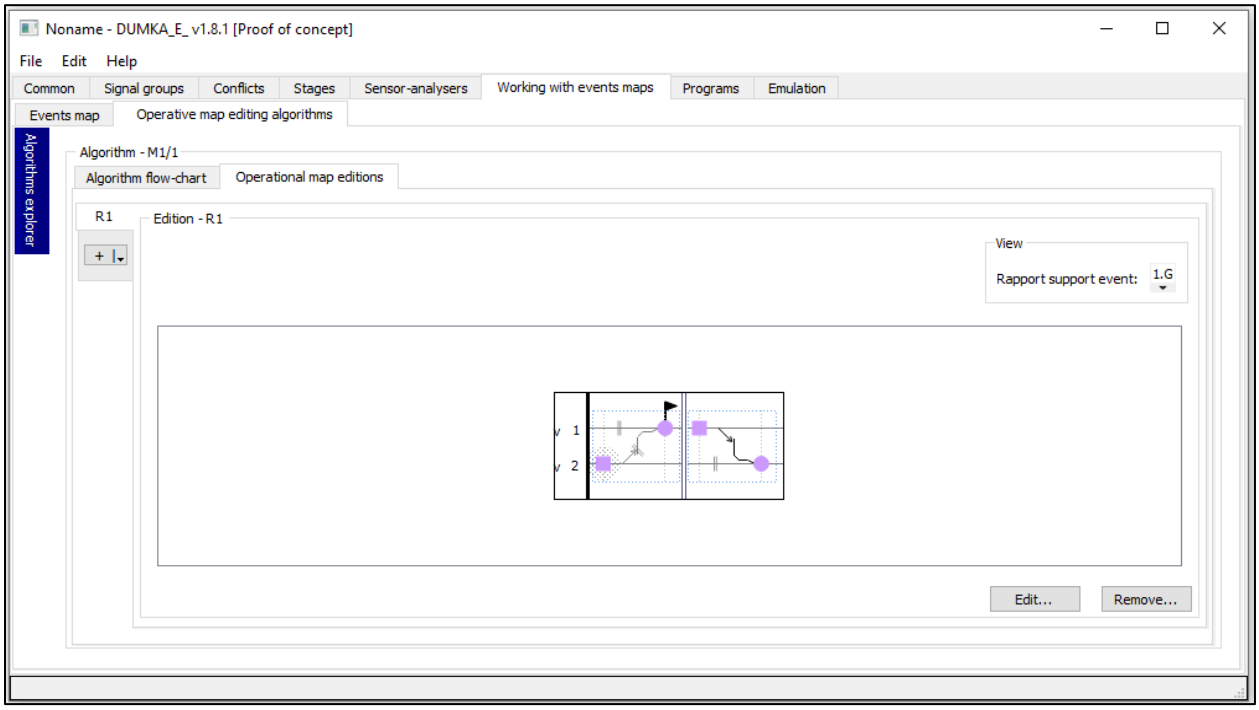
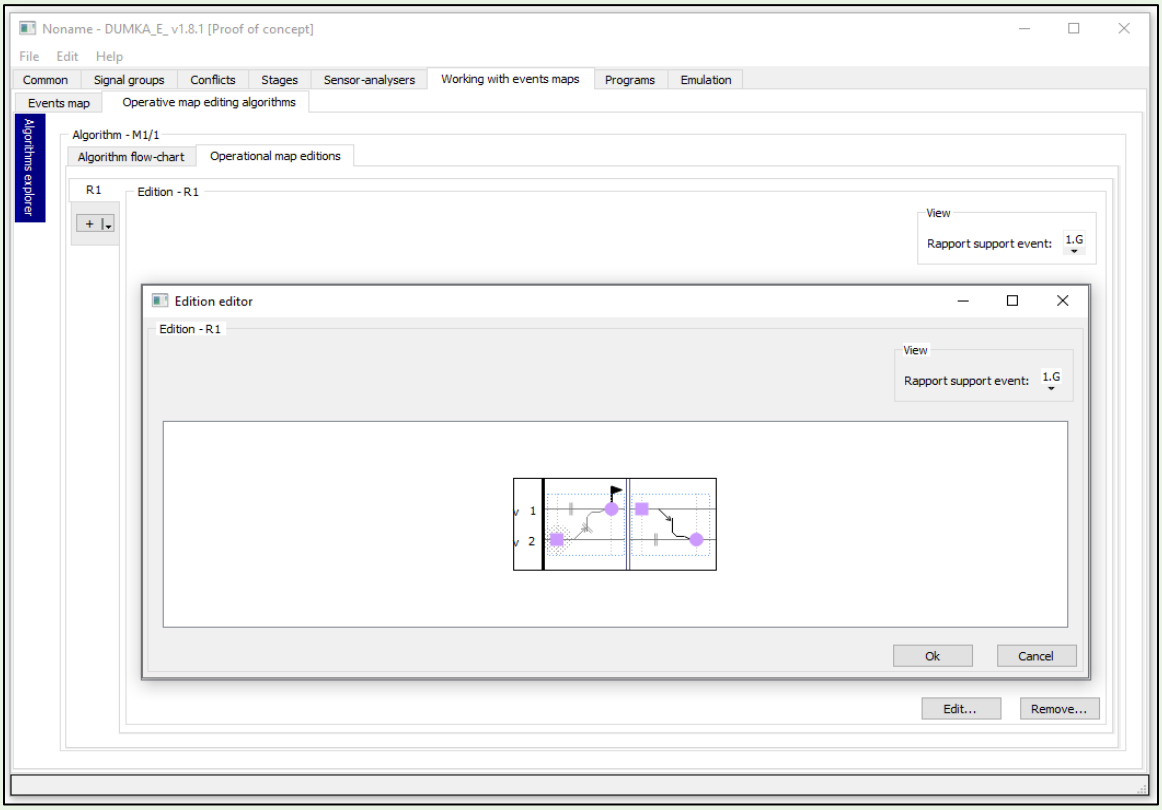


Figure 54. New Edition for Algorithm Integration

Remark 3: Click the "Remove..." button in the bottom right corner to remove a displayed edition.

Remark 4: To edit the currently displayed edition, click the "Edit..." button to launch the edition editor. The following window will appear which provides the capability to modify any action in the map.



STEP 6: To modify event editing actions, simply hover over the bottom area of the desired event icon. This will cause the editing actions selector button to appear (**Figure *55***).

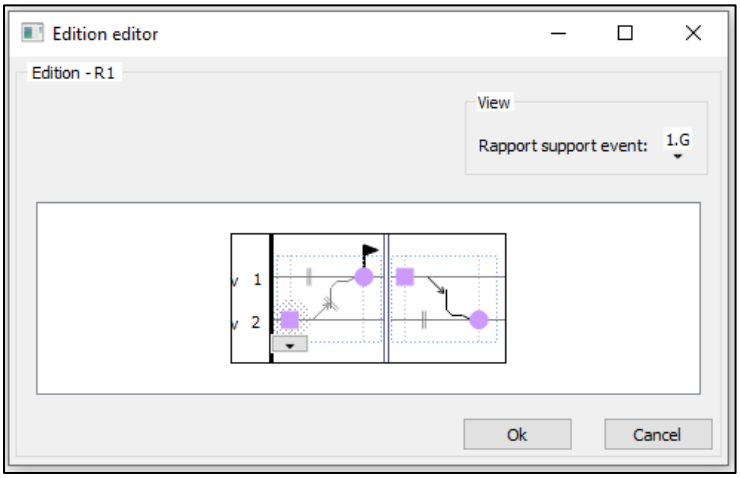


Figure 55. Edition Editor Interface

After hovering over the bottom area of the event icon, click the editing actions selector button to choose the desired editing action for the event. **Figure *56*** shows all the available options that could be selected for editing the actions.

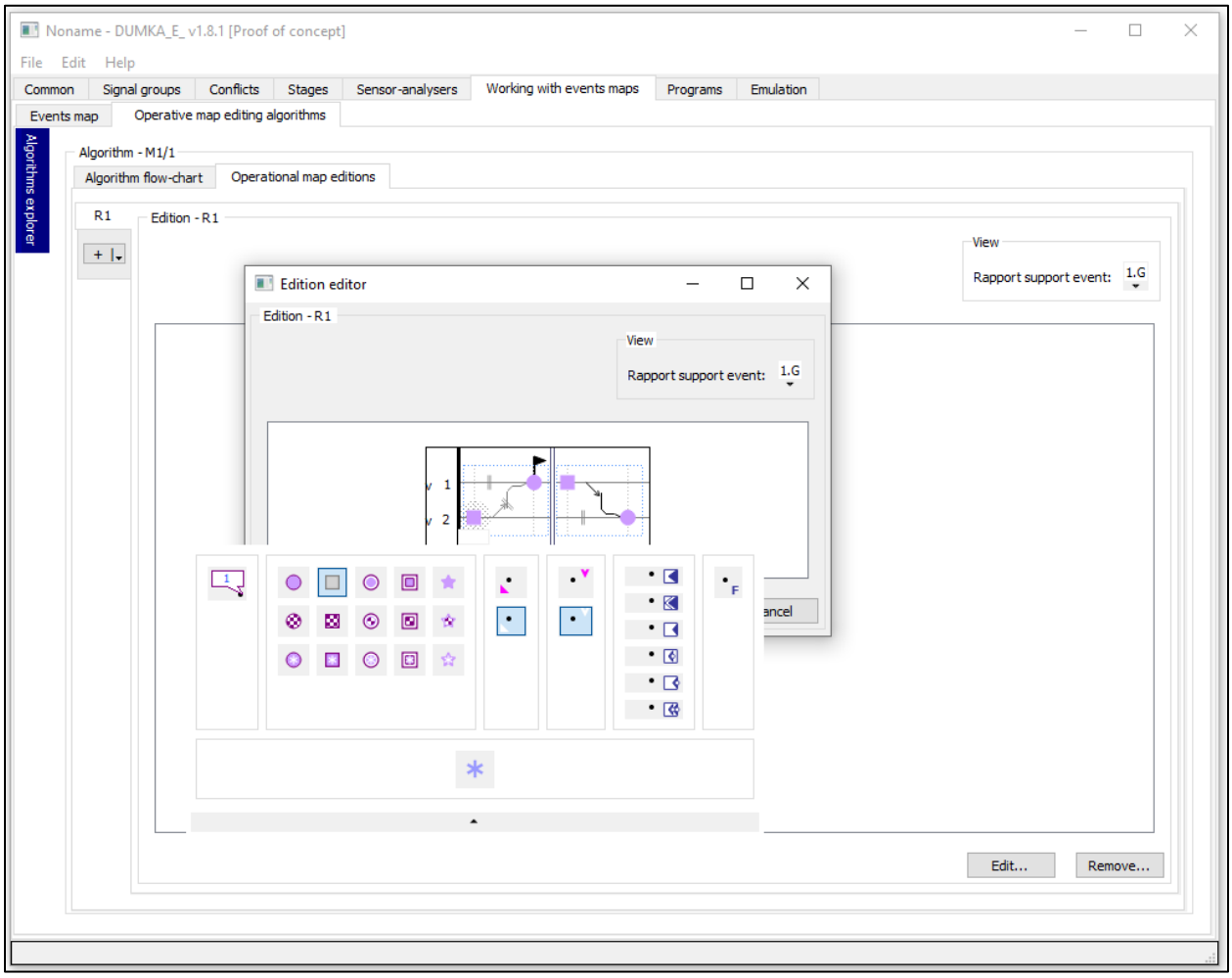


Figure 56. Selecting an Editing Action for an Event: Using the Editing Actions Button

STEP 7: To save any changes made and exit from the editor, simply click on the "Ok" button.

Once all required editions have been created, navigate to the "Algorithm Flow-chart" tab to continue with the map editing algorithm.

Figure *57* illustrates the Algorithm Flow-chart tab which consists of Declaration Table (1) and a window representing the algorithm flowchart (2).

The algorithm flowchart is equipped with the "Edit…" option which provides access to Algorithm flow-chart editor window. The "Edit…" button is in the bottom right corner of this page.

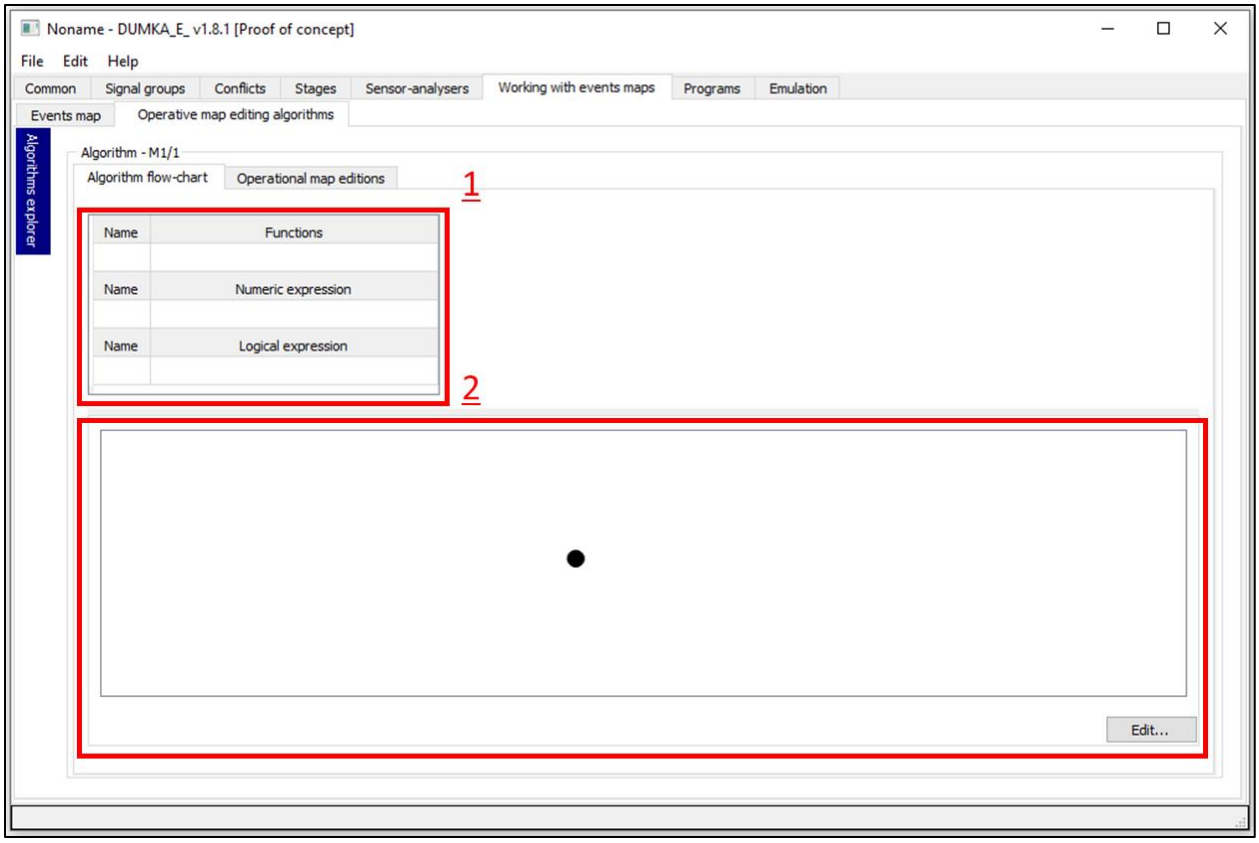


Figure 57. Algorithm Flow-Chart Tab Overview

The table of declarations provides the option to declare names and expressions, including functional, numerical, and logical expressions, which can be referenced later by name. This feature is optional but useful for using named expressions in the algorithm flowchart specification. Follow the steps outlined below to utilize a named expression in the algorithm flowchart.

STEP 8: To create a new named expression, hover the mouse cursor over the empty record in the table of declarations (within the relevant section, e.g., Functions, Numeric expression, or Logical expression, **Figure *58***) and double-click the left mouse button. This action will trigger the named expression creator (**Figure *59***).

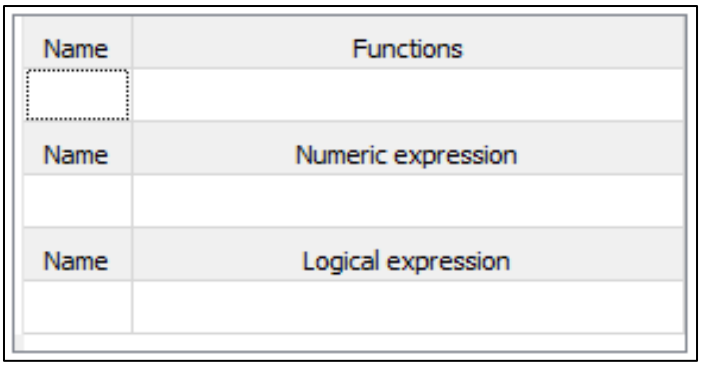


Figure 58. Declaration Table Overview

Figure 59. Expression Creator

STEP 9: Choose a suitable name for the expression from the available options in the dropdown menu.

STEP 10: To select a sub-expression for editing, click on the corresponding sub-expression interactive strip (specified by red in **Figure *60***).

Figure 60. Access to Editing the Sub-Expression through Interactive Strip

Once a user clicks on the sub-expression interactive strip (specified by red in **Figure 61**), the available sub-expression patterns for the given kind will appear. The available options in each expression (Functions, Numerical, and Logical) differ from each other. The available sub-expression patterns are shown in Figure 58 for the Functions (F) and Logical expression (L).

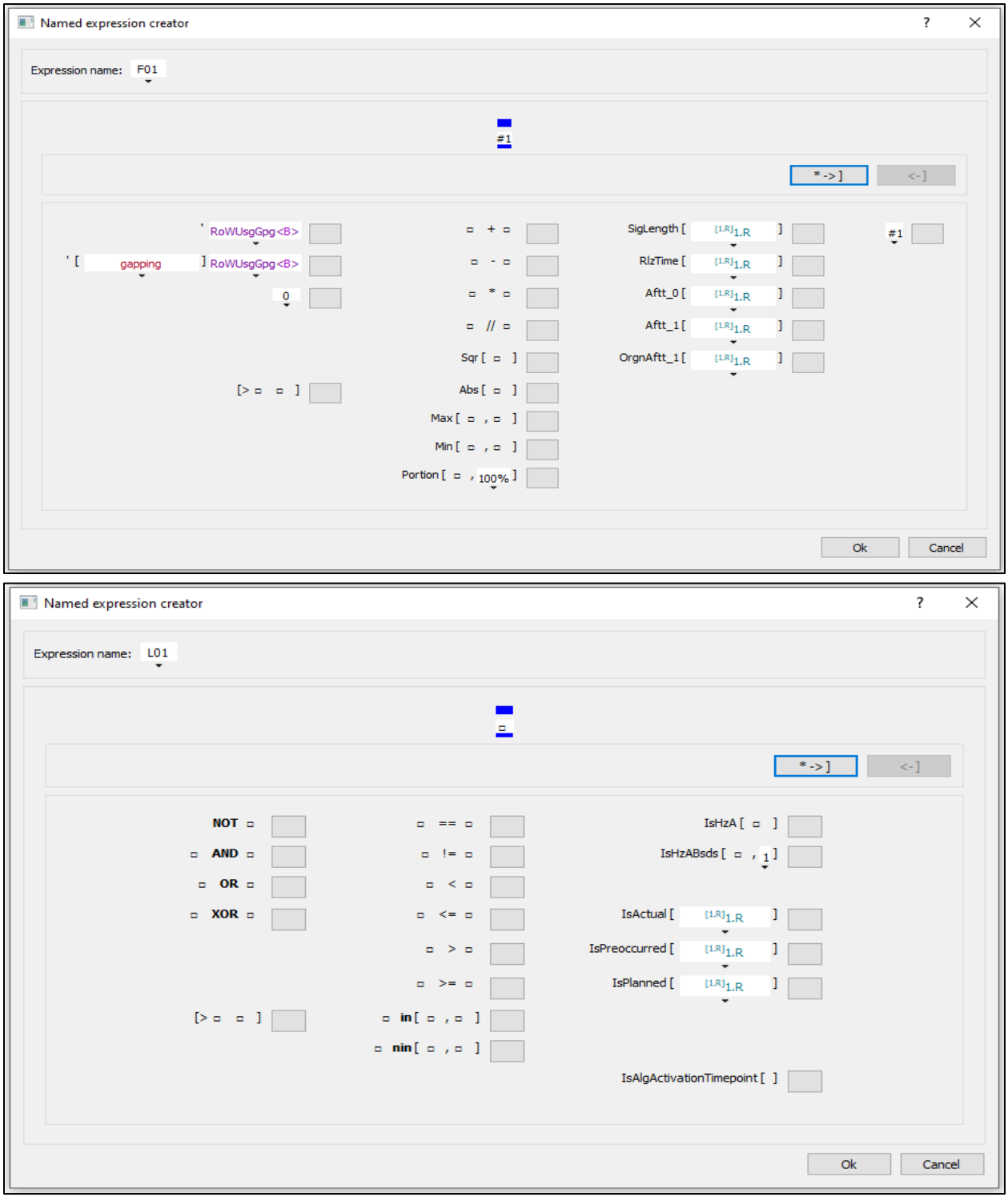


Figure 61. Available Sub-Expression Patterns

STEP 11: Click on the button of the desired pattern to replace the selected sub-expression. For instance, click on the button of the "□ + □" pattern (**Figure *62***).

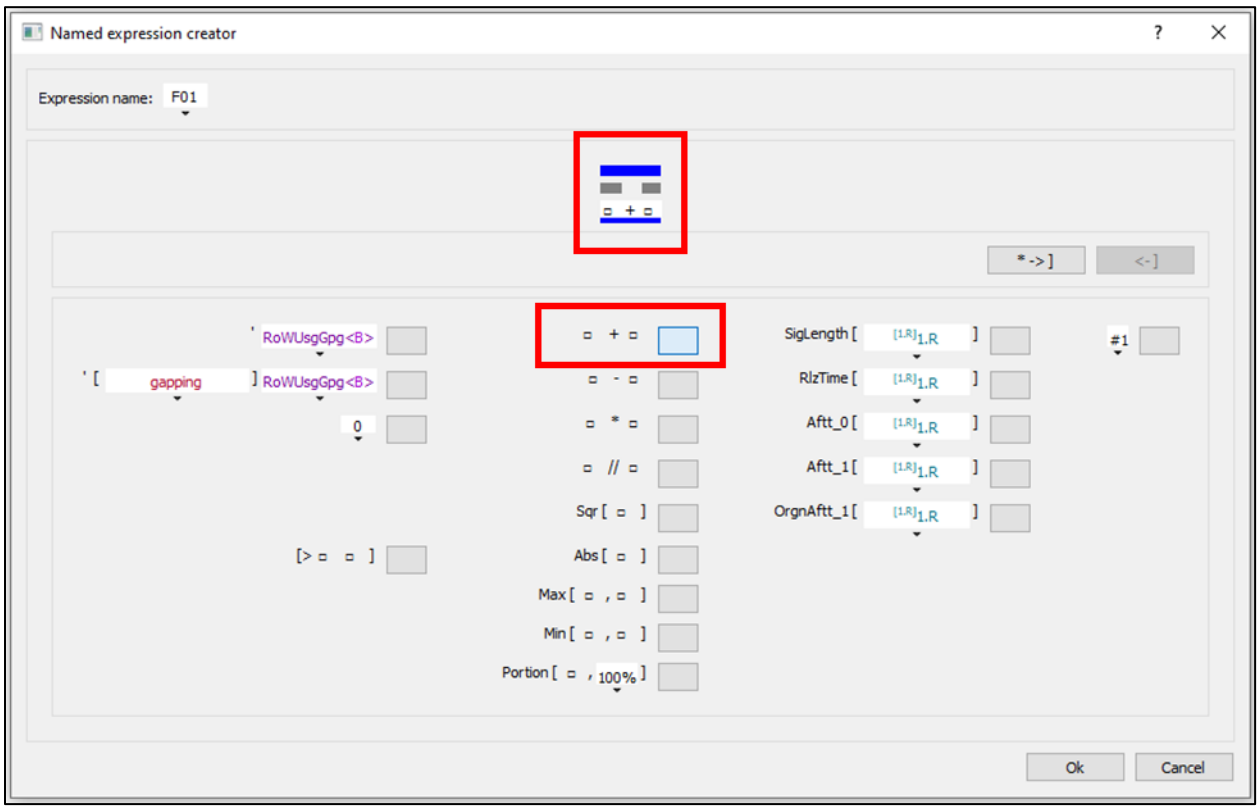
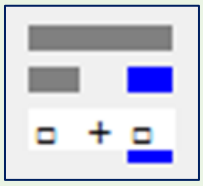


Figure 62. Creating Desired Pattern

Remark 5: To edit the next sub-expression, click on its interactive strip using the left mouse button, and repeat the previous actions. For example, select the expression of the right summand.



Remark 6: With a similar approach, users can create the desired sub-expression using the available patterns.

The list of the patterns available in three different expressions are provided in the following corresponding with their meanings within each expression.

Create Functions

The available options within the "Functions" are provided in **Figure *63***.

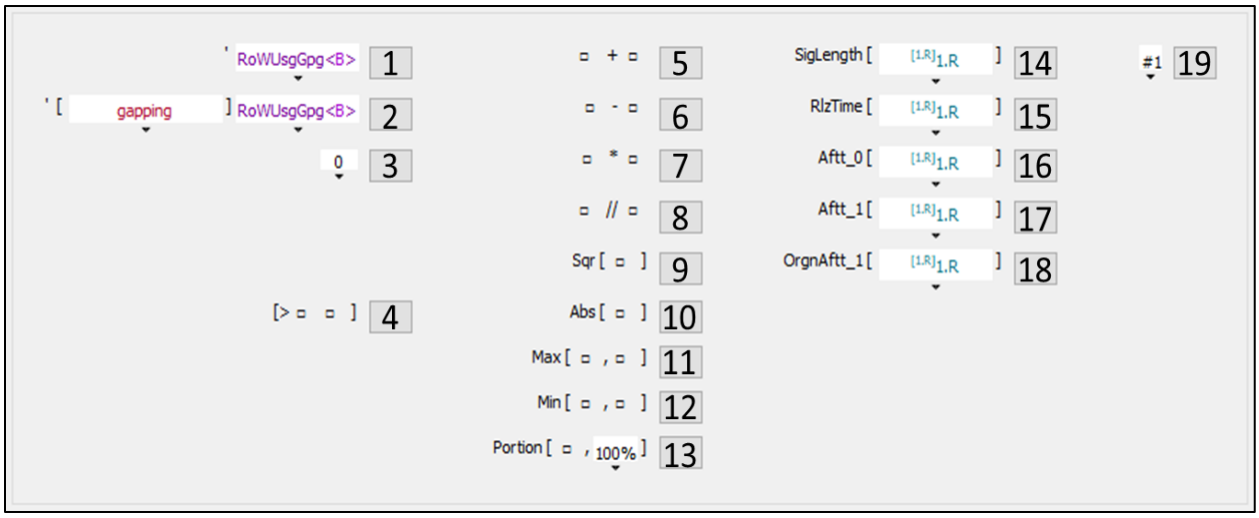


Figure 63. Functions and Numerical Expressions

(1) Expression for choosing the Logical Sensor (Processor) type through its identifier.
(2) Expression for choosing the secondary quantity of the Logical Sensor (Processor).
(3) Expression for accessing a desired number selection.
(4) Contextual (considered in an edition executed context) numeric expression [shouldn't be used in a named expression]

(5-13) Expression for mathematical operations and functions can be formed by using the corresponding symbols and functions.

(14) Signal length
(15) Event occurrence time
(16) Minimum Aftereffect
(17) Main Aftereffect
(18) Original Aftereffect (original/initial aftereffect from program table)
(19) Number of parameters of user defined function

Create Numeric Expression(s)

The options within the "Numeric Expressions" are similar to the options in "Functions", except that option number 19 (number of parameters) does not exist.

Create Logical Expression(s)

The available options within the "Logical Expressions" are provided in the following figure (**Figure *64***).

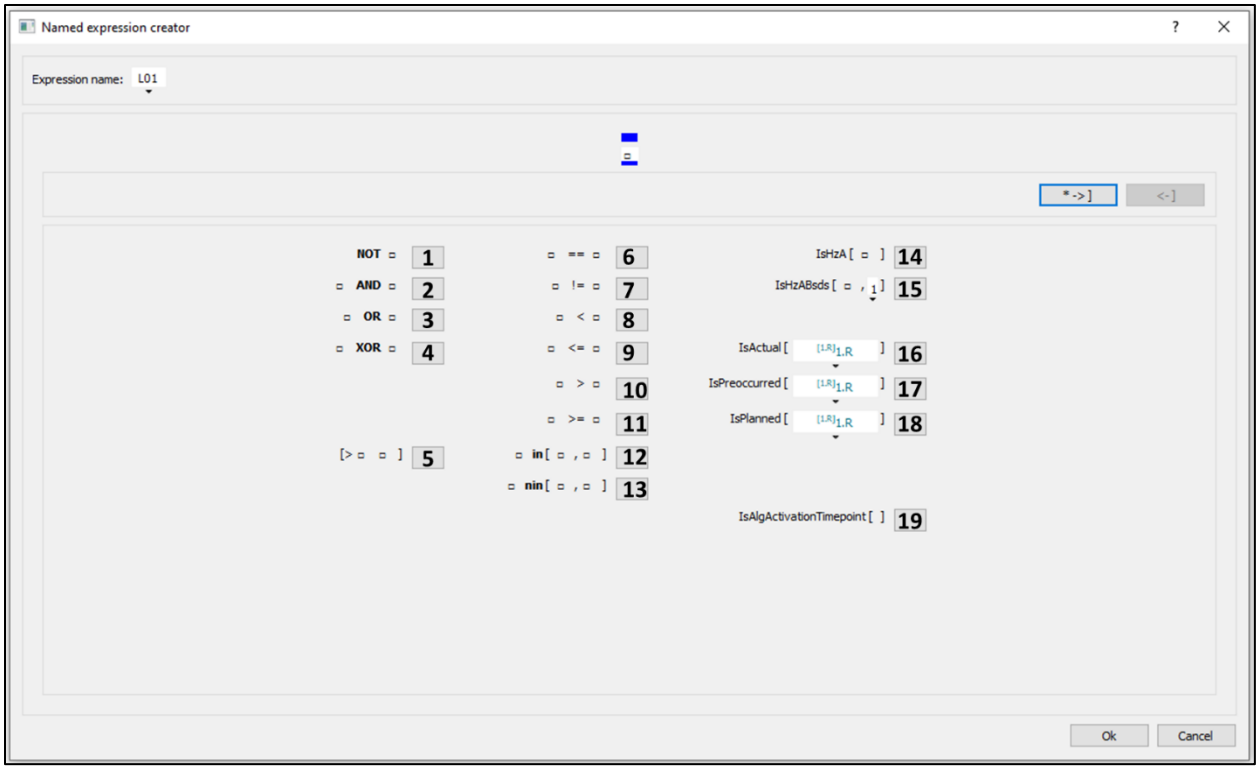


Figure 64. Logical Expressions

(1-4) Logical operators or logical connectives.

(5) Contextual (considered in an edition executed context) numeric expression [shouldn't be used in a named expression].

(6-11) Comparison operators

(12-13) Conditional expression for inclusion/non-inclusion in the closed interval [a, b] determines if a given value falls within the inclusive range defined by the closed interval from a to b.

(14) A predicate with an edition as an argument. Returns True if executing the edition would alter the operative signal horizon.

(15) Similar to **(14)**, but changes in the signal group provided as the second argument will not be considered.

(16-19) Check if the event belongs to the specified class.

STEP 12: After adding any of the desired expressions, click the "Ok" button to add the current named expression to the table and close the expression creator.

Remark 7: To delete a declared named expression, click on the corresponding expression record in the table, right-click to open the context menu, and choose the "Remove..." action.

STEP 13: Click the “Edit...” button to open the Algorithm Flow-chart editor (see **Figure *65***).

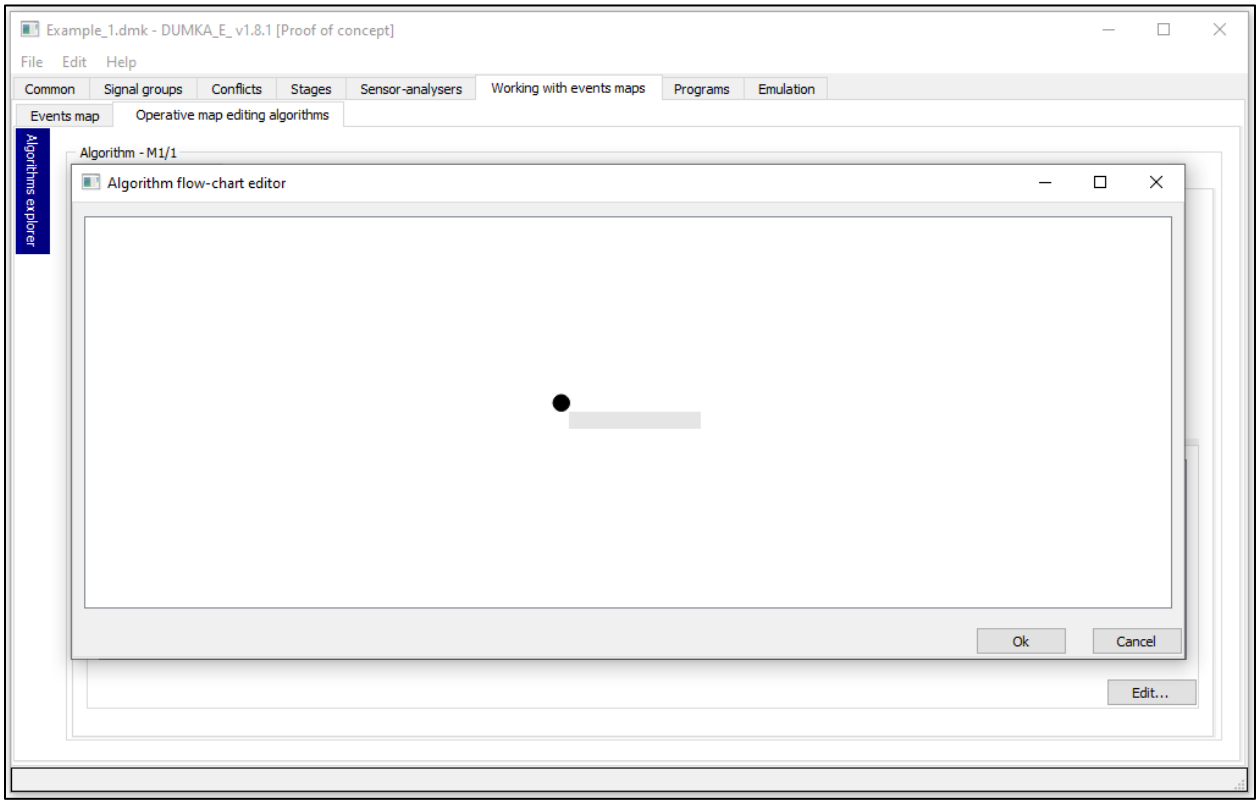


Figure 65. Algorithm Flow-Chart Editor

In the central area of the editor, the control flow source is represented by a black circle.

STEP 14: To add a new flow-chart branch, position the mouse cursor over an interactive "block-progenitor" strip (**Figure *66*-1**).

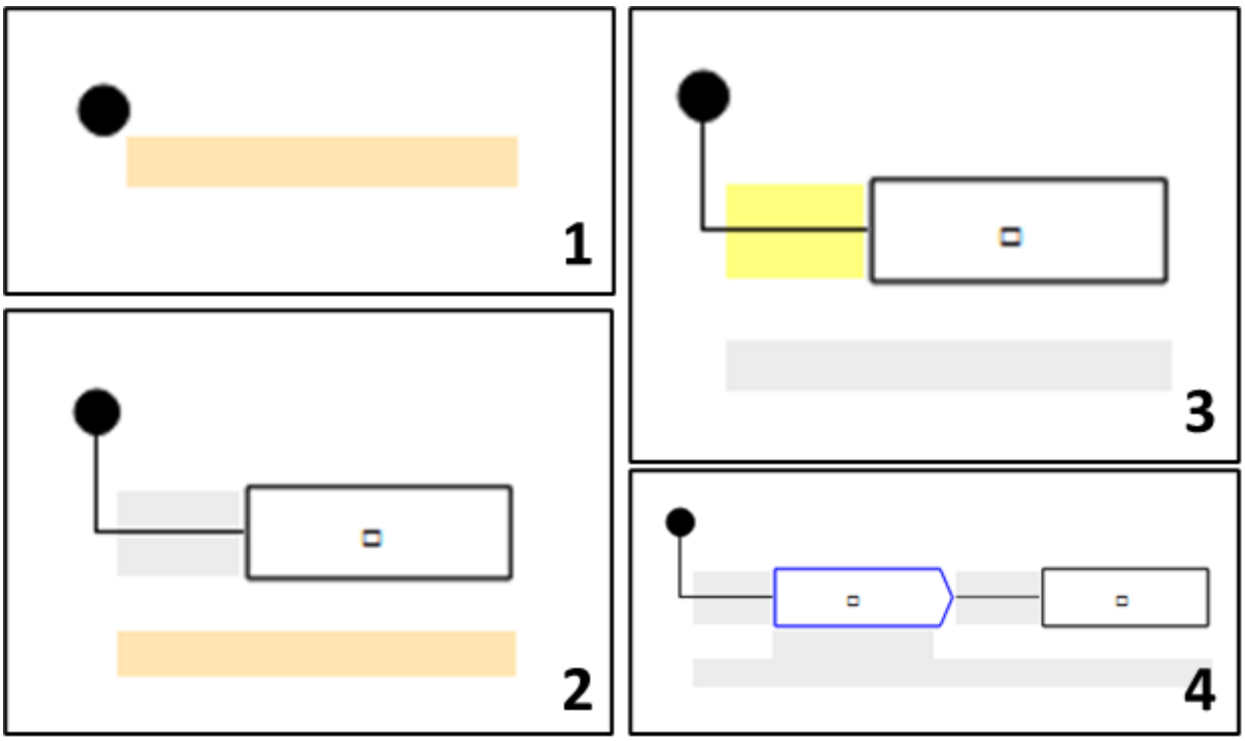
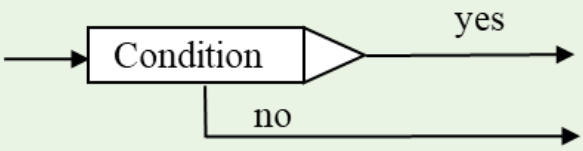


Figure 66. Flow Chart Creating Process

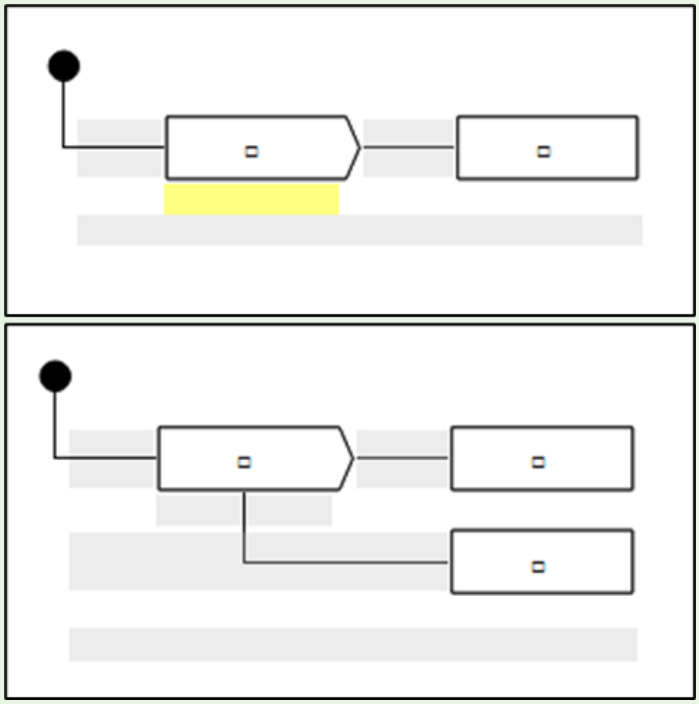
Double-click the left mouse button over the interactive "block-progenitor" strip to add a new minimal branch (**Figure *66*-2**). Here, the rectangle represents the edition execution block (with an undefined edition by default).

To insert a condition block into the branch, hover over an interactive "block-progenitor" strip located on the link and double-click the left mouse button. This will insert a new condition block with an undefined condition by default (**Figure *66*-3 and 4**).

Remark 8: To create a condition block that operates as follows:



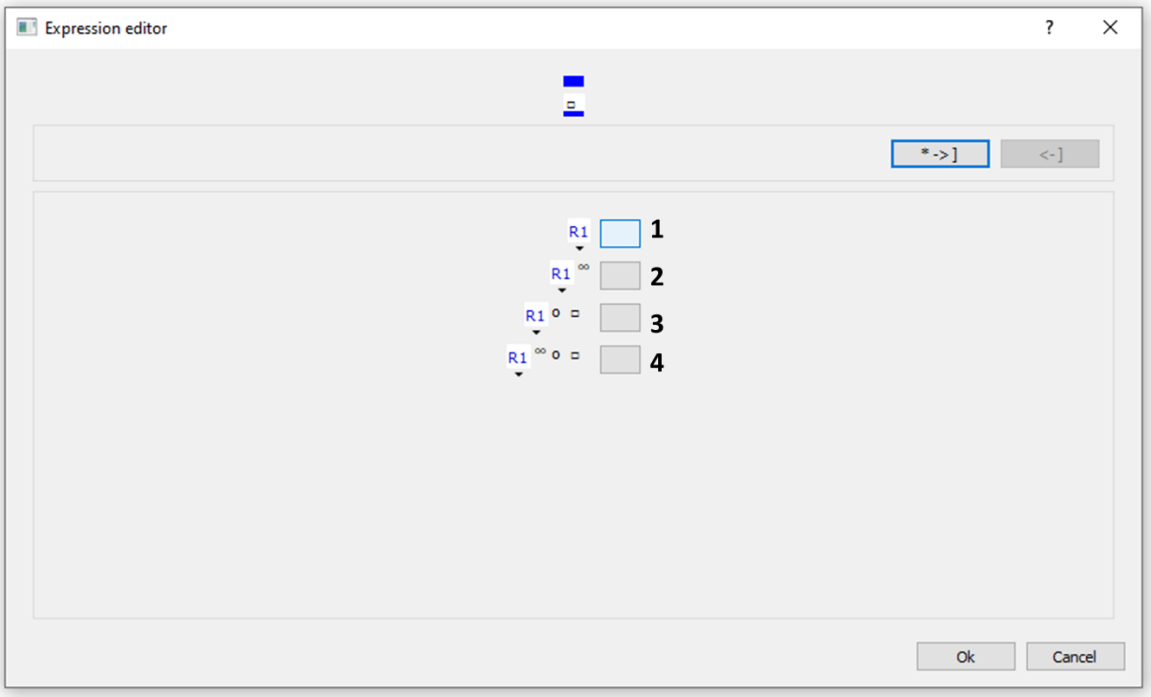
Add an alternative branch, position the mouse cursor over the interactive "block-progenitor" strip located below the condition block similar to below:



Remark 9: To delete a block, hover over the block and right-click with the mouse. Then, select the "Remove the block" action from the context menu.

Remark 10: To modify a condition block expression, hover over the block and double-click the left mouse button. The expression editor, similar to the one in the expression creator (e.g., Functions, Numeric Expressions, and Logical Expressions), will appear. Utilize it to adjust the expression, such as setting specific conditions.

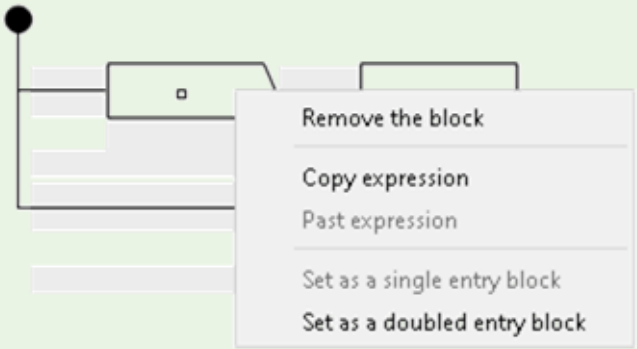
Remark 11: To modify an edition block expression, hover over the block and double-click the left mouse button. The expression editor will appear, resembling the expression creator (as seen earlier), featuring the following layout:



(1) Single edition.
(2) Manifolded edition.
(3) Single edition composition.
(4) Manifolded edition composition.

Utilize the expression editor to craft a suitable expression.
Once the algorithm flow-chart is created, click the "Ok" button to save it in the project and close the algorithm flow-chart editor.

Remark 12: Each initial block in every branch of the block flowchart can be labeled as a "single/double entry" block. A "double entry" block signifies that the control flow will traverse that block (and its branch) twice — once initially, and then again after all other branches have been traversed. To designate this, simply access the context menu of the block and select the appropriate option.



Remark 13: For undo and redo operations, navigate to the main menu, select "Edit," and choose either "Undo" or "Redo."

10. Programs

In the context of the EBC, a program is a set of instructions that specifies the logic for controlling traffic signals. The control logic is structured hierarchically to accommodate various time scales of control. The program defines the construction logic for the signal diagram based on events map. At higher levels, meta-programs define the interchanging logic between programs at the lower level. The program includes specifications for the start-up/finish logic of the signal diagram, as well as settings for normalization and support event interchange delay. This hierarchy comprises four levels:

- Level 0 (Basic Level): This level focuses on “seconds” time scale control and is governed by a specific program. This program outlines the logic for constructing the signal diagram.
- Level 1 (Hourly Level): Operating on the “minutes” time scale, this level is guided by a meta-program. This meta-program defines the logic for the interchange of programs at Level 0.
- Level 2 (Daily Level): Geared towards “hours” time scale control, this level is shaped by a meta-program that dictates the logic for interchanging programs at Level 1.
- Level 3 (Yearly Level): Operating on “days” time scale control, this level follows a meta-program which outlines the logic for interchanging programs at Level 2.

To specify a basic level control program for events-based control, the following is required:

1. Program Basic Diagram Specification:
    1.1. Provide a concrete events map as the foundation for constructing the signal diagram.
2. Program Basic Diagram Start-up/Finish Logic Specification:
    2.1. Define the main cyclic pattern of the events map, determining the cutting place for the operative signal map during operating breaks or new starts.
    2.2. Specify normalization settings (e.g., coordination), including:
        2.2.1. Define how much time of main aftereffects could be skipped when signal diagram is normalizing.
        2.2.2. Define how long (maximum) the support event could be delayed when signal diagram is normalizing.
        2.2.3. Define a global synchronization point (e.g., offset) for the support event for coordination.
3. Adaptation Algorithm (if needed): Specify an adaptation algorithm if adaptation is required.

In essence, this framework ensures a structured and well-defined approach to creating a basic level control program in the events-based control system.

Graphical User Interface – Events-Based Control Program

To define an events-based control program, proceed with the following steps:

STEP 1: Navigate to the “Programs” tab (**Figure *67***)

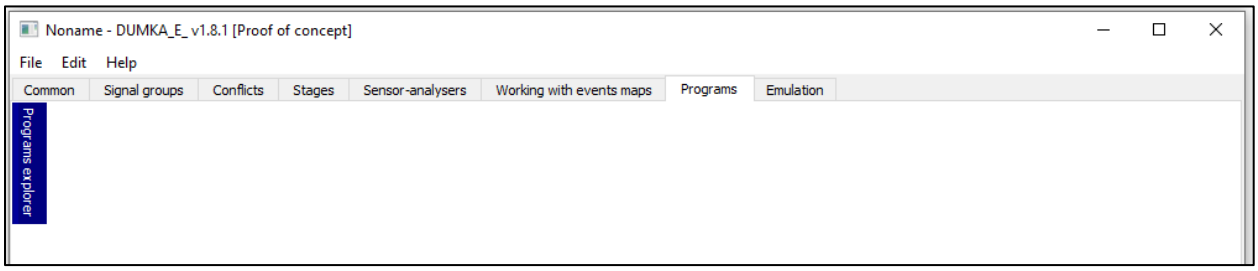


Figure 67. Access to Programs

STEP 2: Next, navigate to the “Programs Explorer” located on the left side of the page. Click on it to open the programs bank content browser (**Figure *68***).

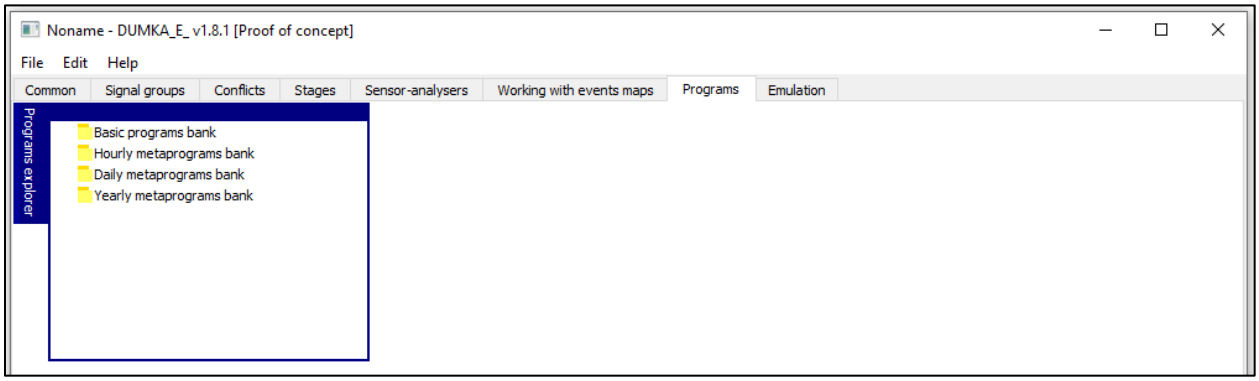


Figure 68. Accessing Programs Bank

STEP 3: In the “Programs Explorer” window, choose the “Basic Programs Bank” folder by clicking the left mouse button. Afterward, access the context menu by right clicking the mouse. Then, initiate the creation of a program by clicking the “Add Program...” action with the left mouse button. This action will prompt the program creation dialog to appear (**Figure *69***).

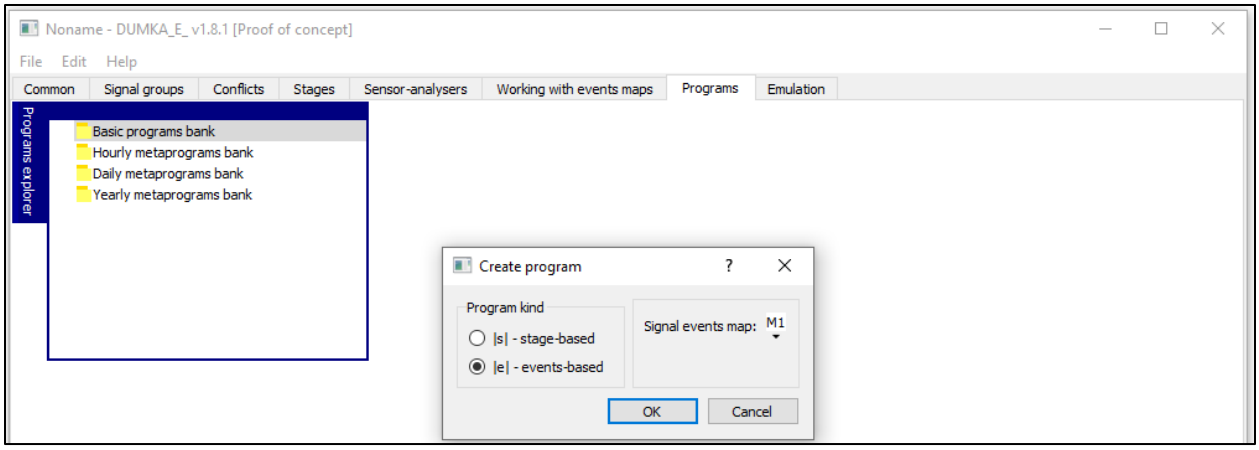


Figure 69. Add Program Action

STEP 4: If necessary, select the suitable "program kind", and click the "Ok" button. This will generate a new default program (automatically identified as |e|, which is "event-based"), and its representation will be displayed on the page.

Figure *70* illustrates the options within the program.

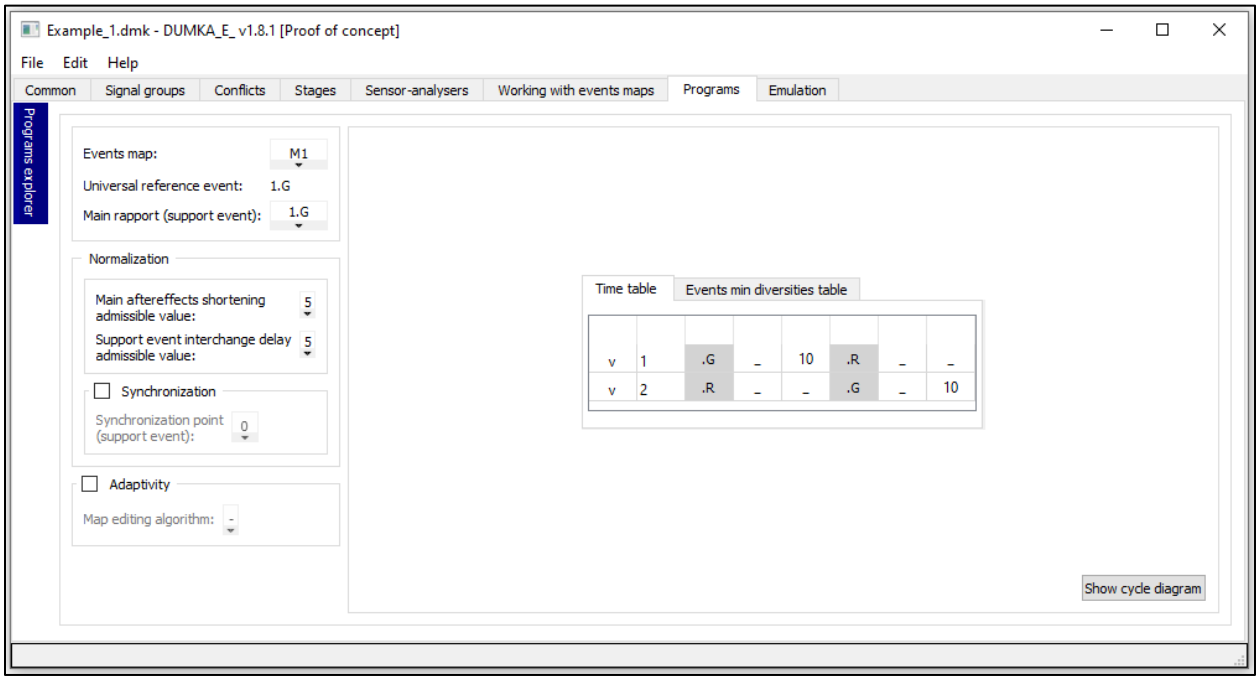


Figure 70. Overview of Programs

If needed, make any necessary modifications as explained in the following steps.

STEP 5: To adjust after-effect times, edit the "Time Table". The sections that are editable are:

i) minimum aftereffect time and ii) main aftereffect time (highlighted in blue in **Figure *71***).

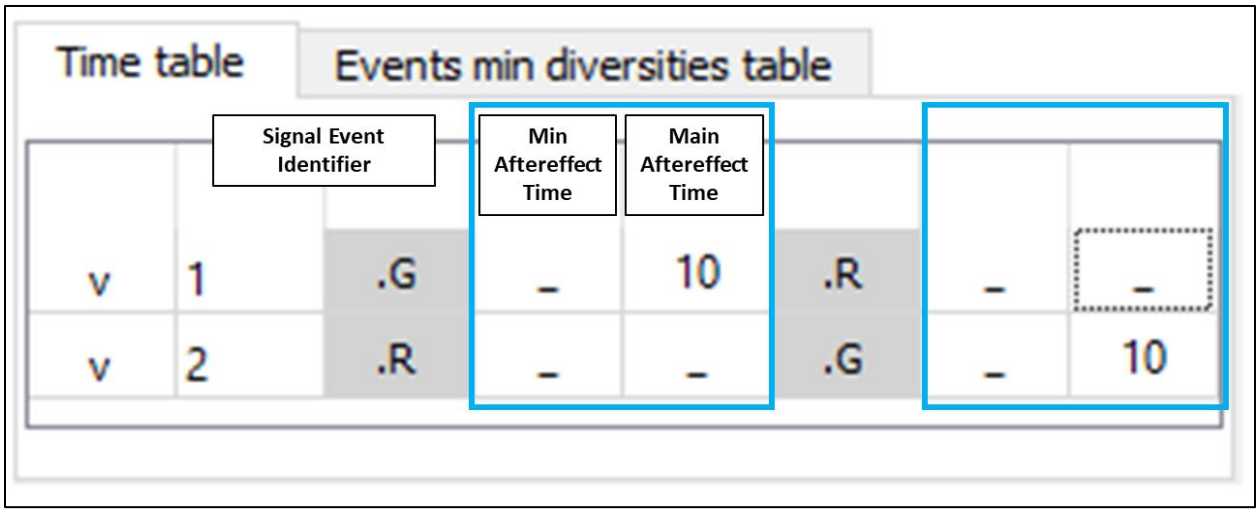


Figure 71. Time Table Components

It should be noted that the symbol "-" is used for the 1 second time. To edit the values, double click on the square and choose the appropriate value (see **Figure *72***).

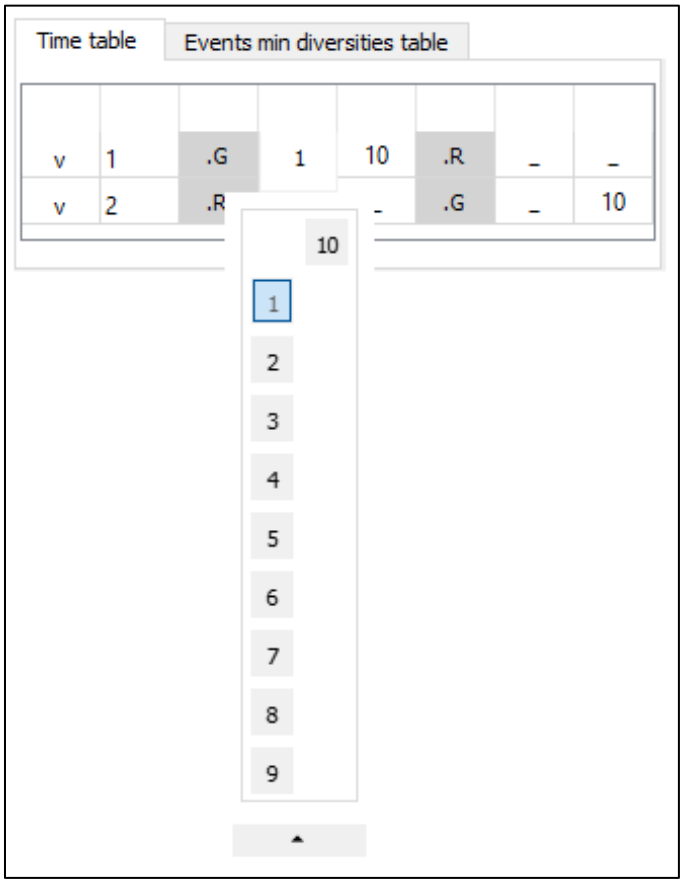


Figure 72. Edit Time Table Values

STEP 6: To adjust events diversity times (*Event Temporal Separation*), navigate to the "Events min diversities table" tab and edit the corresponding table same as above (**Figure *73***).

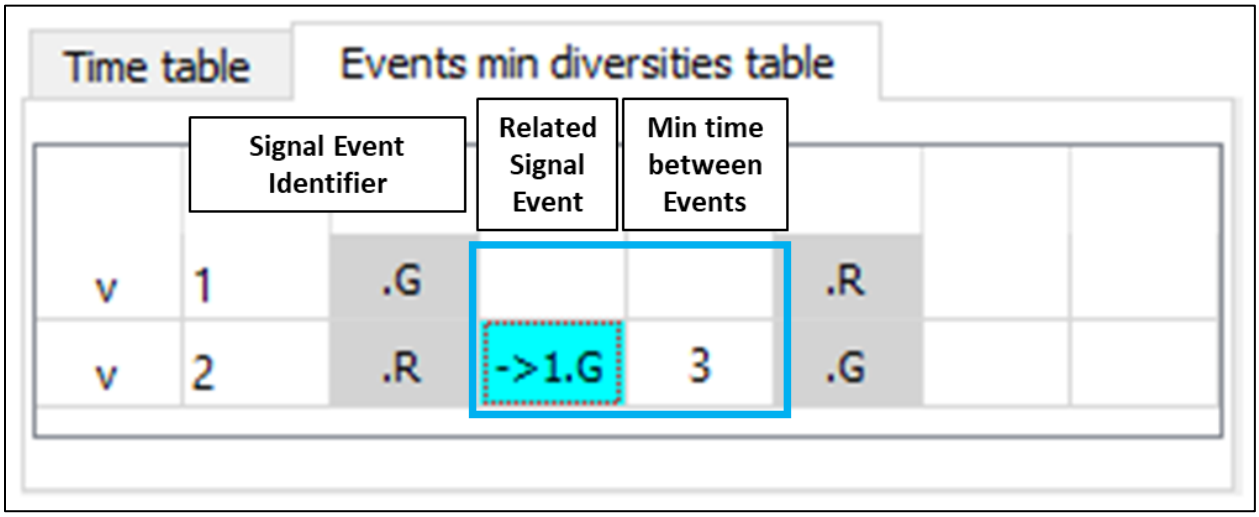


Figure 73. Events Minimum Diversities Components

To view the representation of the basic signal diagram (shown with number "2" in **Figure *74***), click the "Show cycle diagram" button (shown with number "1" below).

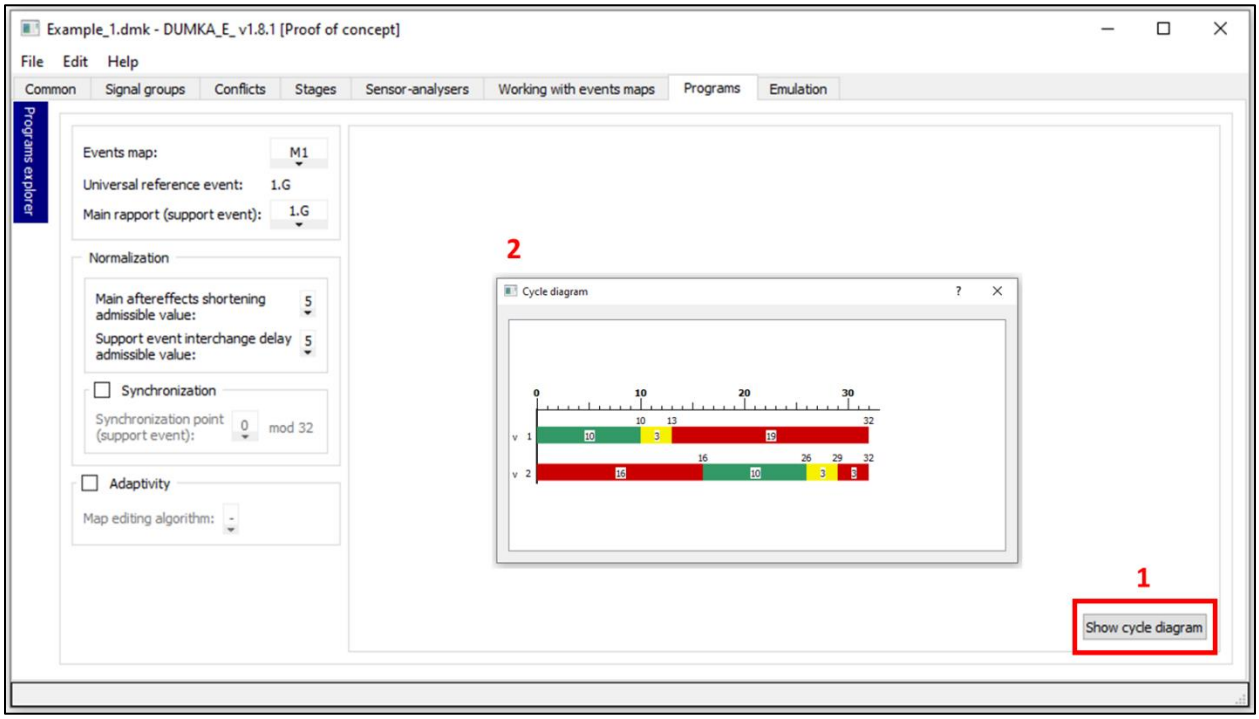


Figure 74. Signal Diagram Representation

To execute "undo/redo" operations, navigate to the main menu, select "Edit," and choose either "Undo" or "Redo."

11. Emulation

The system incorporates a virtual traffic controller. Upon validation, the current project will be automatically loaded into the virtual traffic controller, allowing emulation of its operation based on project settings. Key concepts of the controller include:

Control Unit: A control unit is the driving force behind the controller's operations. Typically, a distinct control unit is allocated for each control level: basic, hourly (meta), daily (meta), and yearly (meta) levels. These units follow a hierarchical structure, with each level being subordinated to the one above it.

Operational Control Element (OCE): The Operational Control Element (OCE) guides the activity of the control unit. There are various kinds of OCEs, each associated with specific functionalities:

For the basic level unit:

- All-motions forbidding fixed stage meso-routine (“all red” executor).
- All-motions authorizing with caution fixed stage meso-routine (“yellow flashing” executor).
- Fixed stage routine (executor for a stage from the special stages collection).
- Dynamic stage routine (executor for dynamically designed stages).
- Engineer program (executor for a traffic engineer-designed program).

For the meta-level unit:

- Engineer program (executor for a traffic engineer-designed meta-program).
- Provider routine (executor that provides a specified lower-level control activity without performing any additional actions).

Leading OCE: The leading OCE, distinct from a provider routine OCE, operates at the highest control level.

Control unit modes are defined for both basic level and metalevel units, for the basic level unit:

- OCE execution activating mode (lasts until a new OCE management manifestation on the signal horizon).
- OCE execution normalizing mode (lasts until the transition process caused by OCE changeover is complete).
- OCE normal execution.
- OCE execution completing mode (lasts until the current OCE finishes).

For the metalevel unit:

- OCE execution normalizing mode (lasts until the transition process caused by OCE changeover is complete).
- OCE normal execution.
- OCE execution completing mode (lasts until the current OCE finishes).

The controller can execute the following external commands:

1. Reset: Resets the controller, initiating the “all off” stage executor as the initial OCE.
2. Call leading engineer program: Requests the “engineer program” OCE as the leading OCE.
3. Call leading fixed stage routine: Requests the “fixed stage routine” OCE as the leading OCE.

A request for an OCE includes the following arguments:

- OCE id
- OCE changeover urgency (influencing the urgency of interrupting the current OCE)
- OCE changeover stringency (determining whether to interrupt the current OCE if it is identical to the requested one).

Graphical User Interface – Emulation

To emulate the functioning of the controller, follow these steps:

STEP 1: Navigate to the “Emulation” tab (**Figure *75***).

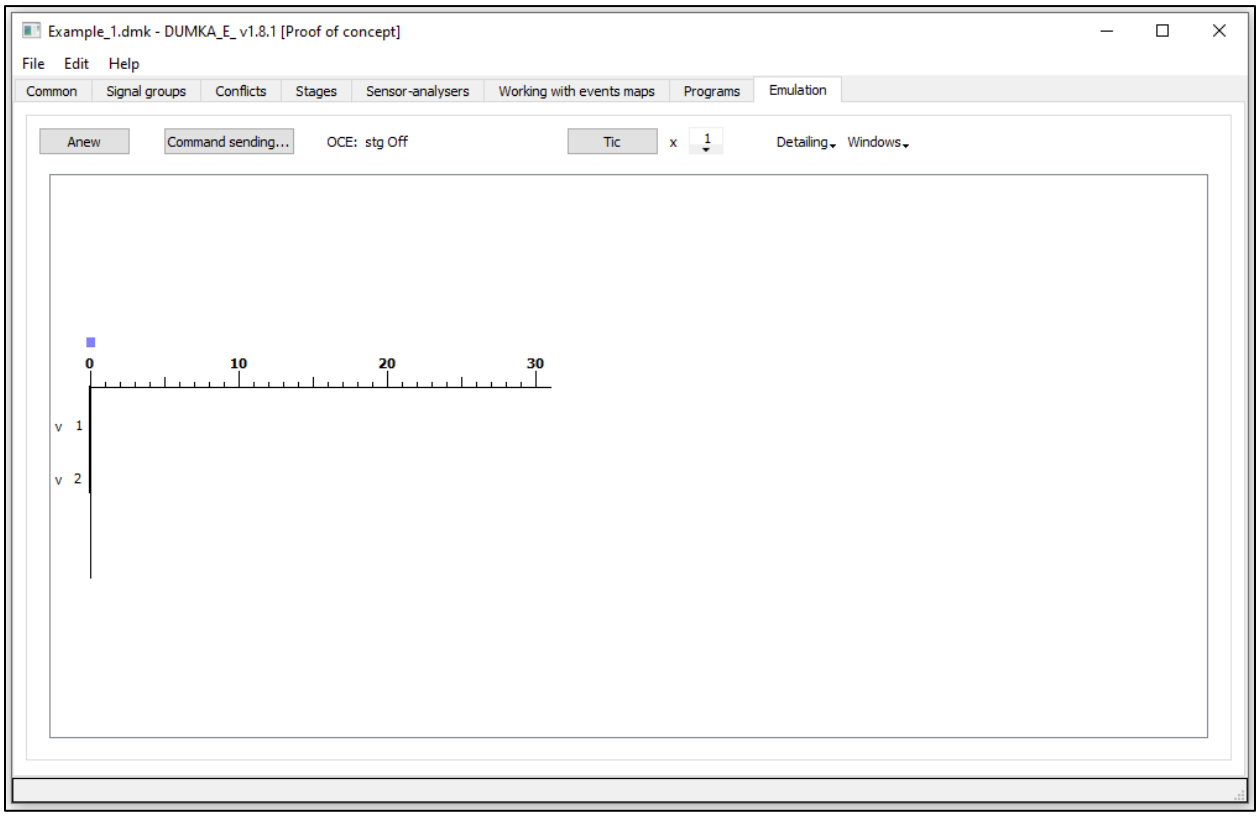


Figure 75. Emulation Tab Overview

STEP 2: To fine-tune the operational representation, click the “Detailing” flat-button and select or deselect the corresponding items (**Figure *76***).

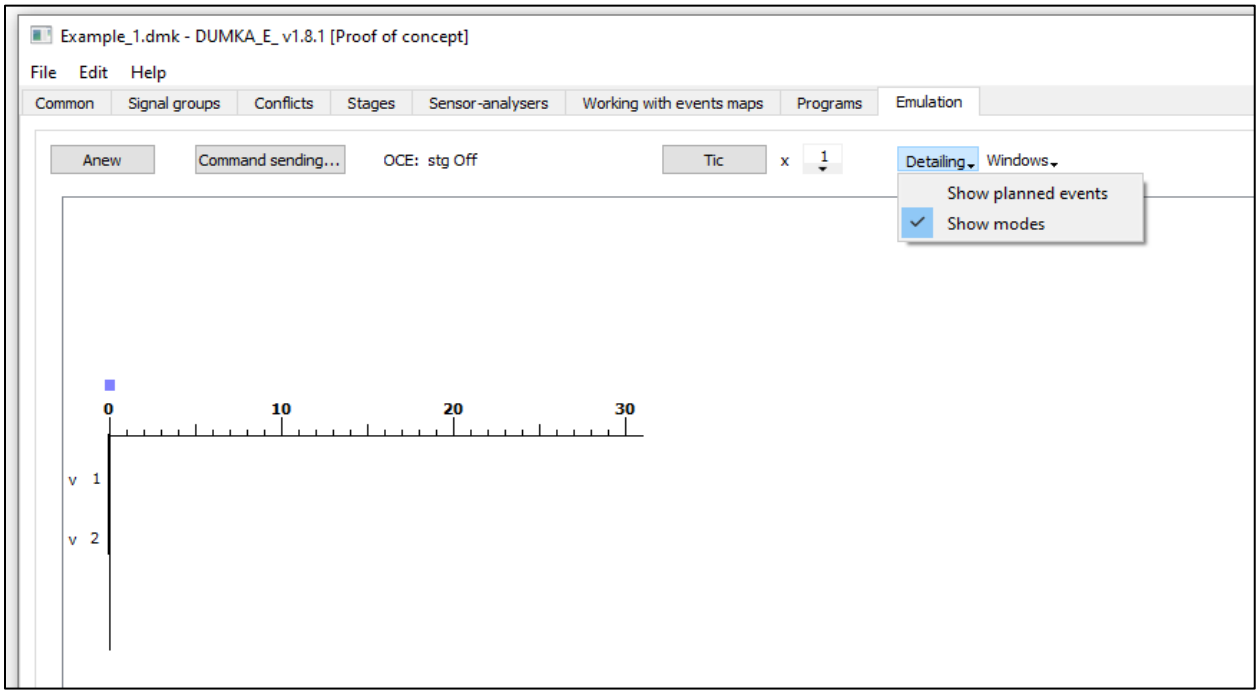


Figure 76. Representation Details

STEP 3: To simulate detector inputs, position the mouse cursor on the graphical representation area, right-click, and select the "Add detector emulation..." action. This will open the detector emulation dialog (**Figure *77***).

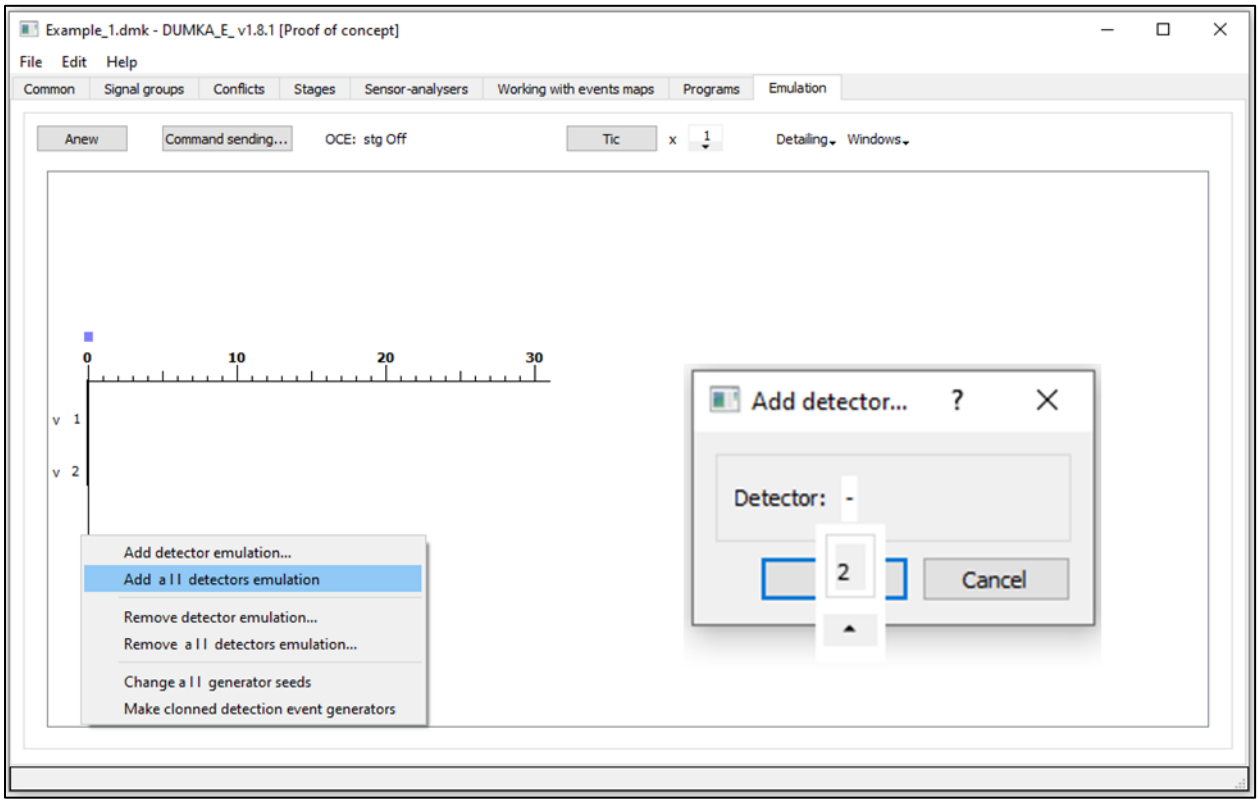


Figure 77. Add Detectors to Emulation

Select a detector identifier for emulation and click the "Ok" button. The emulated detector will be visually represented as a blue rectangle labeled with the detector ID, indicating its tic-state.

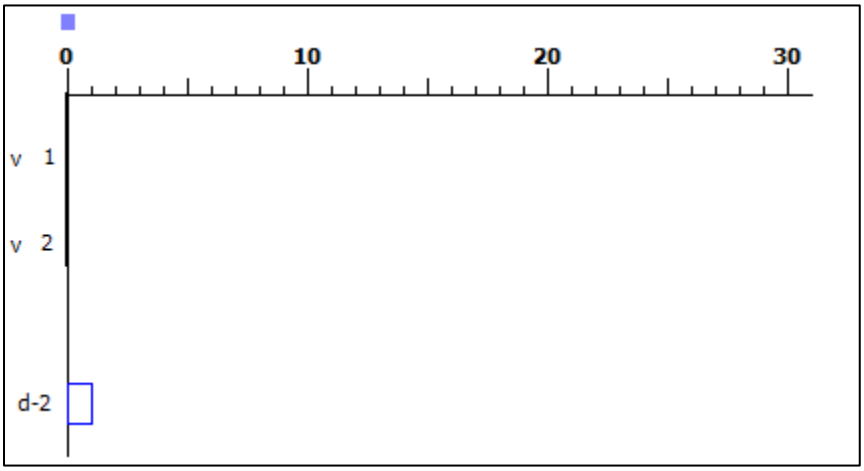
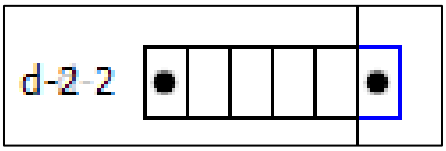


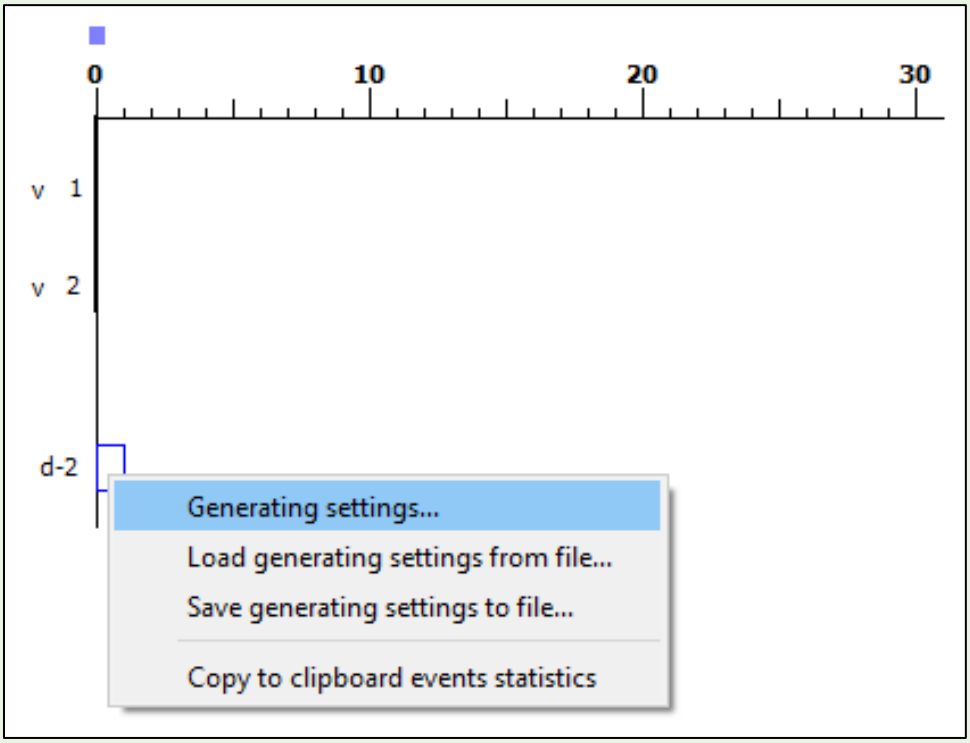
Figure 78. Added Detector Emulation

It should be noted that by default, detector emulation is set to "manually driven," meaning that the occurrence of object presence detection events is controlled by user manipulation.

STEP 4: To manually enable or disable the occurrence of object presence detection events for a specific second, hover over the detector tic-state rectangle and double-click the left mouse button. The appearance or disappearance of a black point in the rectangle corresponds to the presence or absence of an object detection event occurrence.



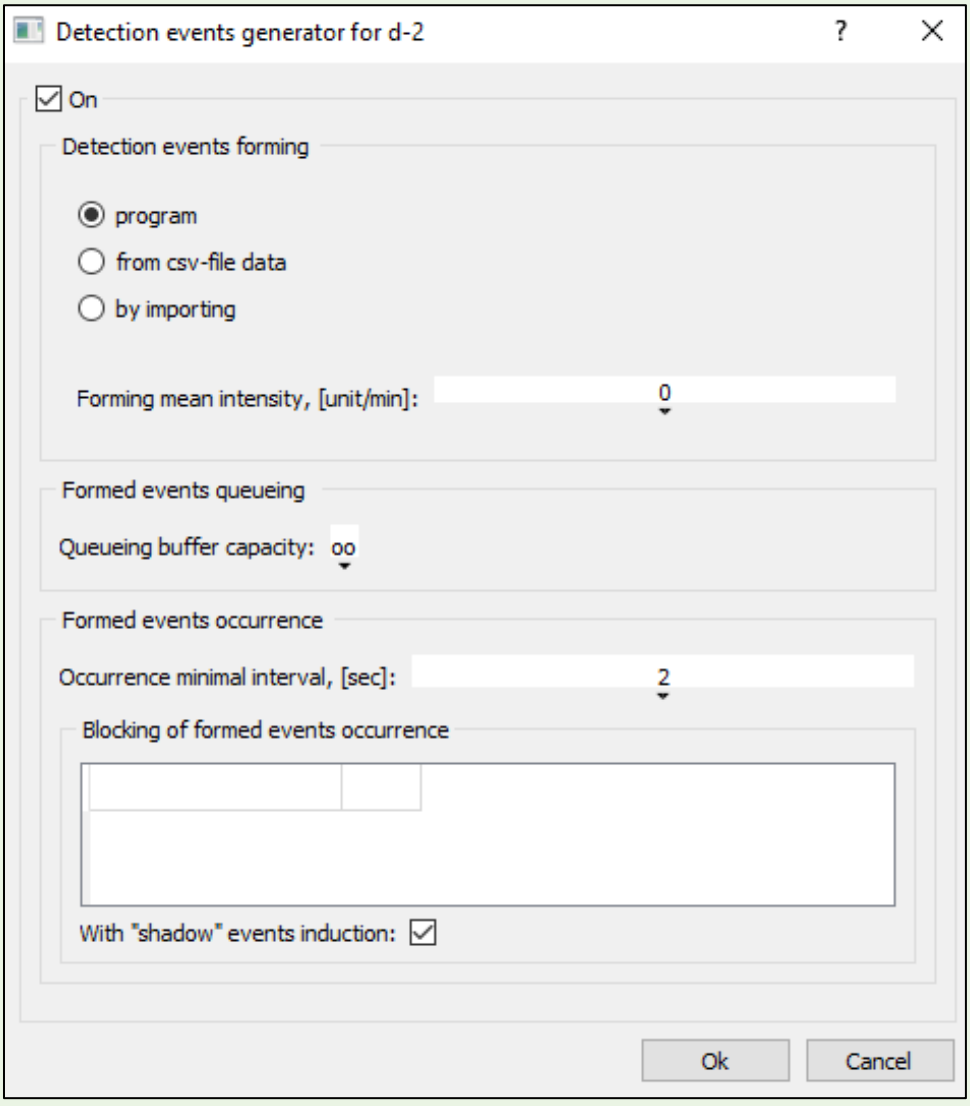
Remark 1: To configure automatic generation of object presence detection events occurrences, hover over any detector tic-state rectangle, right-click, and select the "Generating settings..." option from the context menu.



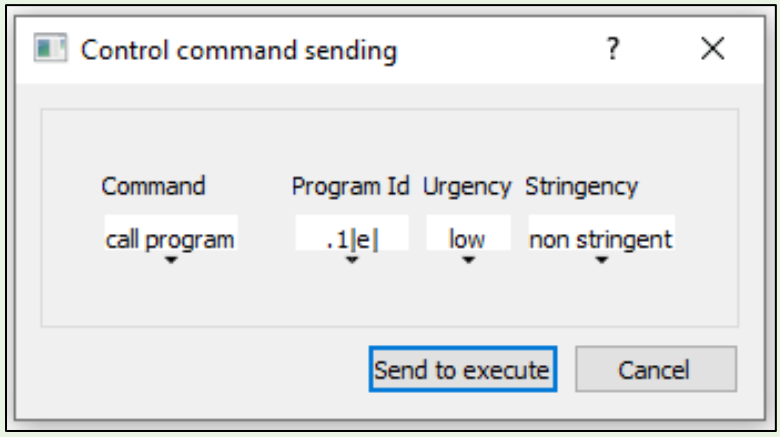
This action will open the detector detection events generator window shown below.

- The "On" checkbox corresponds to the generator's "on"/"off" state.
- "Intensity" represents the rate of events generation, often interpreted as the arrival rate of vehicles or pedestrians.
- "Occurrence interval" indicates the rate at which generated events occur, interpreted as the servicing rate of vehicles or pedestrians.
- "Blocking of formed events occurrence" refers to the signal group number whose forbidding signal will block the occurrence of generated events, typically associated with the signal group controlling the approach of road users.

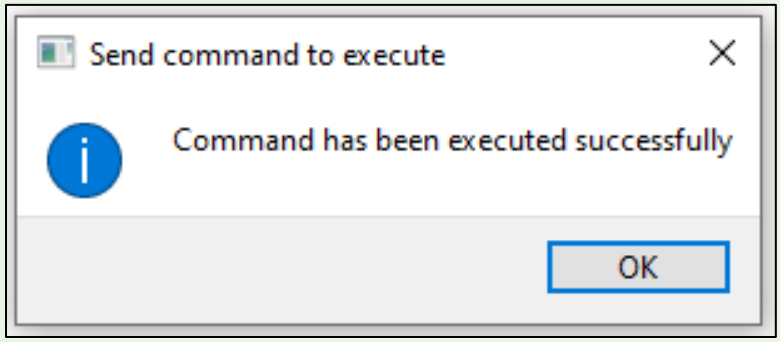
Any changes made to detector editions result in an outcome similar to pressing the "Anew" button (explained in Remark 3).



Remark 2: Click the "Command sending..." button to access the command sending dialog. From there, select the desired command, such as "call program," and click "Send to execute."



A confirmation message will appear in the information box, indicating the successful execution of the command.



STEP 5: To fast-forward the controller's operating time, use the “Tic” button. Each press of the button advances the controller time by one second, multiplied by the specified multiplier (highlighted in red in **Figure *79***). This action triggers the drawing of the corresponding operative signal diagram and related details easier and faster.

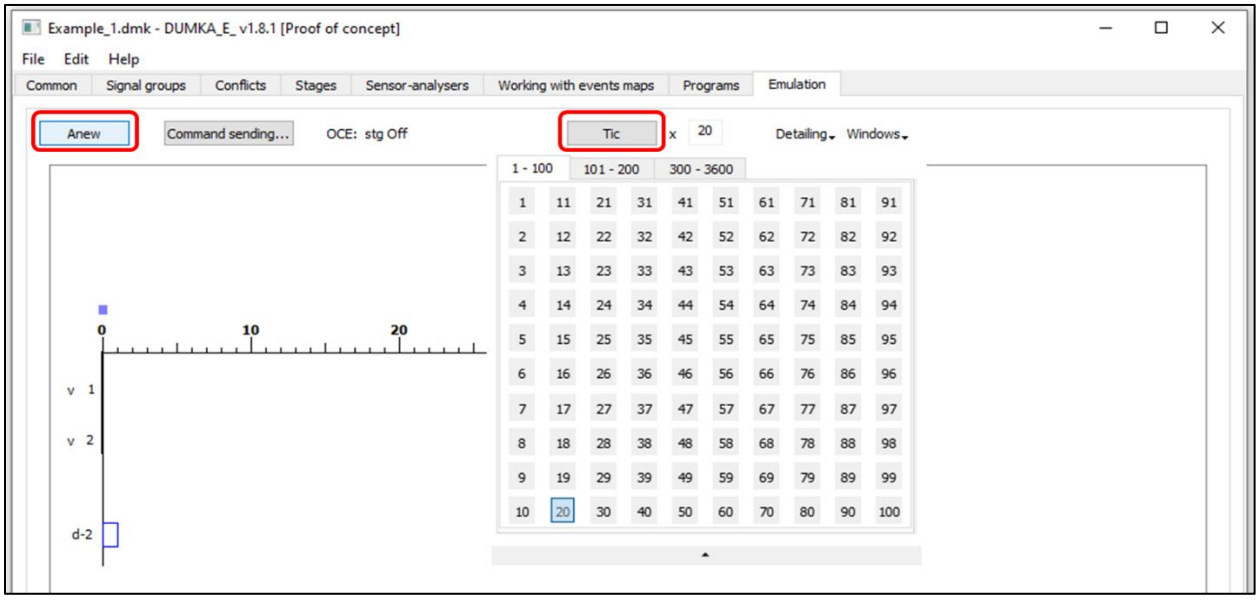


Figure 79. Fast-forwarding the Emulation Time

To clear the drawn diagram and reset the controller, press the “Anew” button (highlighted in red in **Figure *79***).

STEP 6: To trace the OCE events-based program operative signal events map editing algorithm, click the "Windows" flat-button and enable the "Algorithm tracing window." The corresponding tracing window will be displayed. The green color in the tracing window corresponds to the control flow path, while lilac indicates inexecution, often due to reasons such as the edition being inexecutable at the given moment.

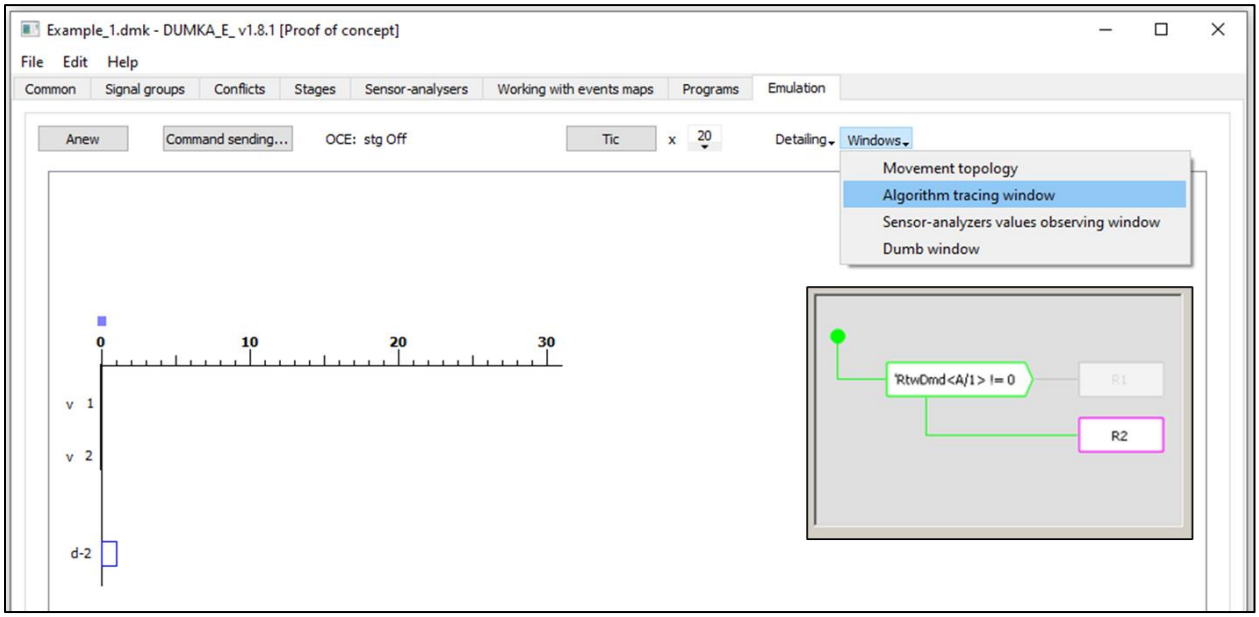
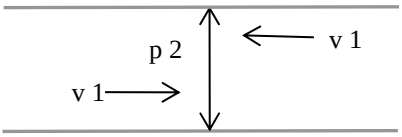


Figure 80. Algorithm Tracing Window

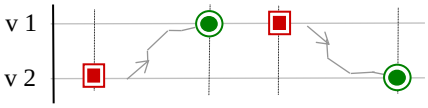
12. Examples

Example 1: Pedestrian Actuated Control

Pedestrian actuated control is a traffic control system that empowers pedestrians to activate changes in traffic signals (e.g., primarily through devices such as push buttons). Figure below shows the simple geometry for showcasing this example:

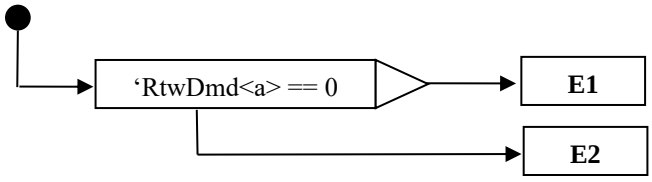


The Signal Event Graph corresponding to the figure above shows the precedence relations between the signal events in a cyclic pattern as shown in the figure below:



Algorithm 1: Delayed start of green signal for pedestrian crossing (p2)

The flow-chart below shows how the algorithm makes the decision, between two rules/editions (E1 and E2), based on the detected demand of the pedestrian crossing (p2).



Editions (E1 and E2) are represented below:

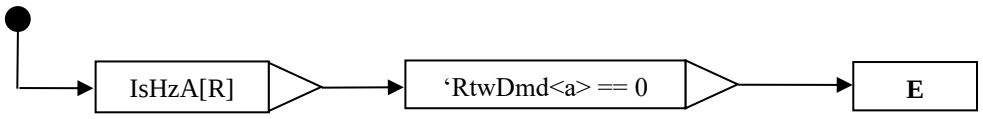
Interpretation: If there is no demand for pedestrian crossing (p2), then execute R1.

- E1 involves decreasing the level of urgency for the red signal event for v1, consequently delaying it along with all subsequent events.

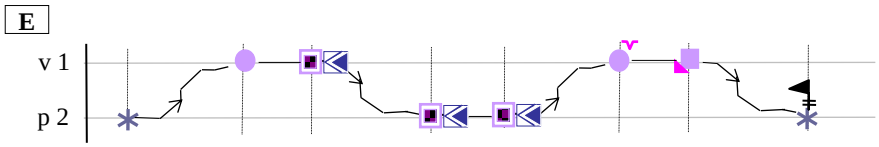
On the contrary, if there is a demand for pedestrian crossing (p2), then execute R2.

- E2 entails raising the level of urgency for the pedestrian green signal event.

Algorithm 2: Skipped green signal for pedestrian crossing (p2)



Edition (E) is represented below:

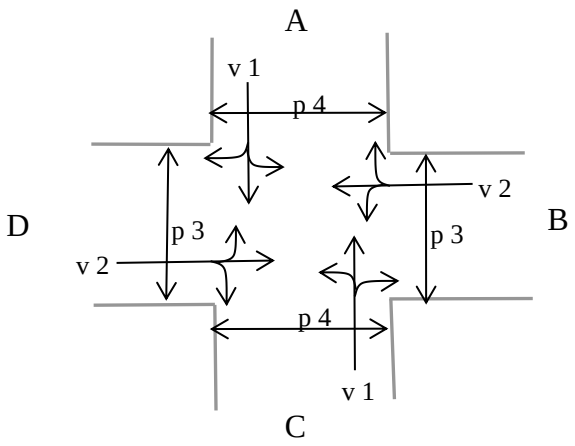


Interpretation: If by the decision-making point, the algorithm detects no demand for pedestrian crossing (p2), executes E.

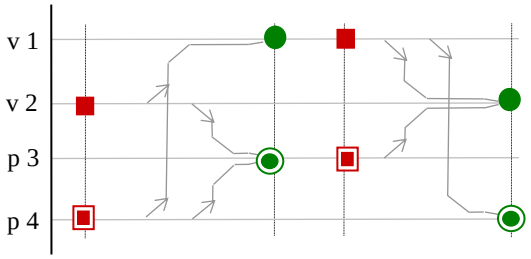
- E involves skipping the pedestrian service to avoid disrupting the overall coordination for v1.

Example 2: Vehicle Actuated Control

Vehicle actuated control (semi- or fully-) is a traffic control system designed to respond dynamically to the presence or absence of vehicles at intersections. Figure below shows the simple geometry for showcasing this example:

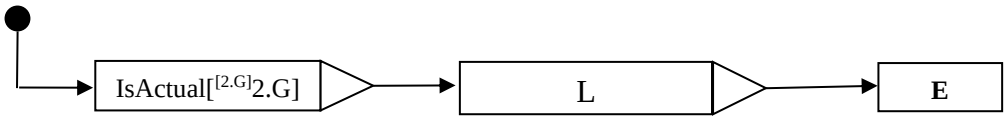


The Signal Event Graph corresponding to the figure above shows the precedence relations between the signal events in a cyclic pattern as shown in the figure below:

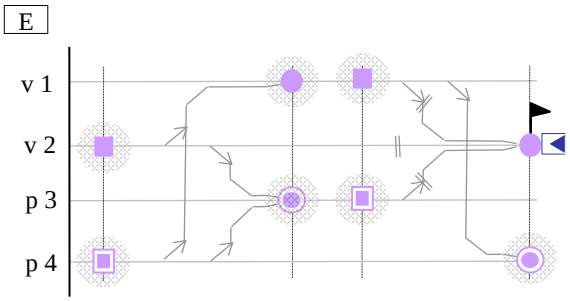


Algorithm 1: Semi-Actuated (based on gap-out)

Symbol	Logical Condition
L	'RtwUsgGpg<B> == 1 **AND** 'RtwUsgGpg<D> == 1



Edition (E) is represented below:

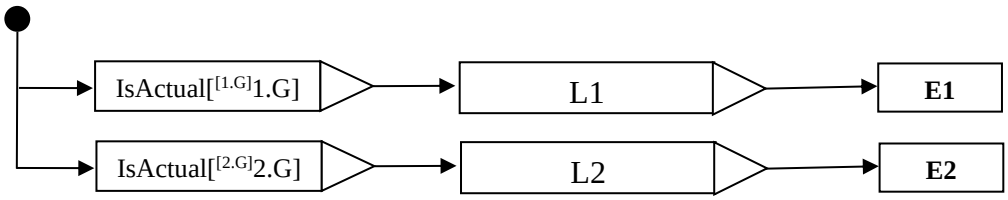


Interpretation: If the current signal indicates that signal group V2 is in the green phase and a gap is detected, proceed to execute E.

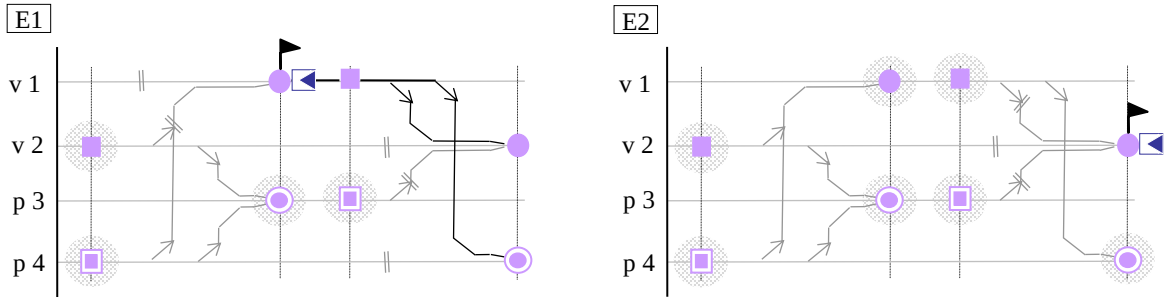
- E involves reducing the planned duration of the green signal for v2 and terminates it (gaps out).

Algorithm 2: Fully Actuated (based on gap-out)

Symbol	Logical condition
L1	'RtwUsgGpg<A> == 1 AND 'RtwUsgGpg<C> == 1
L2	'RtwUsgGpg<B> == 1 AND 'RtwUsgGpg<D> == 1



Editions (E1 and E2) are represented below:



Interpretation: If the current signal indicates that signal group V1 is in the green phase and a gap is detected in both approaches of intersection (A and C), proceed to execute E1.

- E1 involves reducing the planned duration of the green signal for v1 and terminates it (gaps out).

If the current signal indicates that signal group V2 is in the green phase and a gap is detected in both approaches of intersection (B and D), proceed to execute E2.

- E2 involves reducing the planned duration of the green signal for v2 and terminates it (gaps out).

Appendix A: Primary and Transitional Control, Vienna Convention

Table 10. Primary Control Signals for Vehicles

Primary Control Signal Name	Representation		Controlling logical meaning
	Graphical	Textual	
Vehicular			
Logically Vehicular Red Flashing		Rf	as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Red		R	as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Common Green		G	as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Additional Green		g	as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Accented Green		$\bar{G}$	as in the **extended** Vienna Convention in p. 10 for corresponding traffic lights lighting
Logically Vehicular Yellow Flashing		Yf	as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Absent		A	as in the Vienna Convention for corresponding traffic lights lighting

Table 11. Primary Control Signals for Pedestrians

Primary Control Signal Name	Representation	Controlling logical meaning
	Graphical	
Pedestrian		
Logically Pedestrian Red		as in the Vienna Convention for corresponding traffic lights lighting
Logically Pedestrian Green		as in the Vienna Convention for corresponding traffic lights lighting
Logically Pedestrian Absent		as in the Vienna Convention for corresponding traffic lights lighting

Primary Control Signal Name	Representation		Controlling logical meaning
	Graphical	Textual	
Vehicular			
Logically Vehicular Yellow			as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Yellow Alternative			as in the **extended** Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Red and Yellow			as in the Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Green Flashing			as in the Vienna Convention for corresponding traffic lights lighting

Logically Vehicular Accented Green Flashing			as in the **extended** Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Additional Green Flashing			as in the **extended** Vienna Convention for corresponding traffic lights lighting
Logically Vehicular Additional Green and Yellow			as in the Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Additional Green and Alternative Yellow			as in the Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Additional Green Flashing and Red			as in the **extended** Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Additional Green and Yellow and Red			as in the Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Additional Green Flashing and Yellow and Red			as in the **extended** Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Accented Additional Green and Green Flashing			as in the **extended** Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)
Logically Vehicular Accented Additional Green Flashing and Green			as in the **extended** Vienna Convention for corresponding traffic lights lighting (provided a common direction/direction controlling is)

Appendix B: Signal Schemes Examples

Traffic lights system T

Canonical lights layout

1. Signal scheme [T.1.x]$_o$

1.1 Movement geometrical directions ▲ – any

Lights form aspects specification.

General control signals	Implementing					
	Primary form	Transitional forms				
			 A			
	logically vehicular red	logically vehicular yellow	logically vehicular alternative	logically vehicular yellow and red		
						
	logically vehicular green	logically vehicular green flashing				
						
	logically vehicular yellow flashing					
						
	logically vehicular absent					

Transitions specification

from general control signal: \ to general control signal:								
		E	B	E	B	E	B	E	B
								
									
			 A						
			 A						

2. Signal scheme [T.2.x]$_o$

2.1 Movement geometrical directions ▲ – any

Lights form aspects specification.

General control signals	Implementing					
	Primary form	Transitional forms				
			 A			
	logically vehicular red	logically vehicular yellow	logically vehicular alternative	logically vehicular yellow and red		
						
	logically green accented	logically green accented flashing				
						

	logically vehicular yellow flashing					
						
	logically vehicular absent					

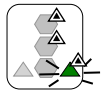
Transitions specification

from general control signal:	to general control signal:								
								
	E	B	E	B	E	B	E	B
								
								
		 A						
		 A						

3. Signal scheme [T.x.e]$_A$ (exclusive additional)

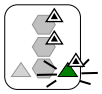
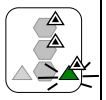
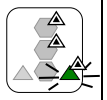
3.1 Movement geometrical directions ▲ – right-oriented (from 0 to 90 grad)

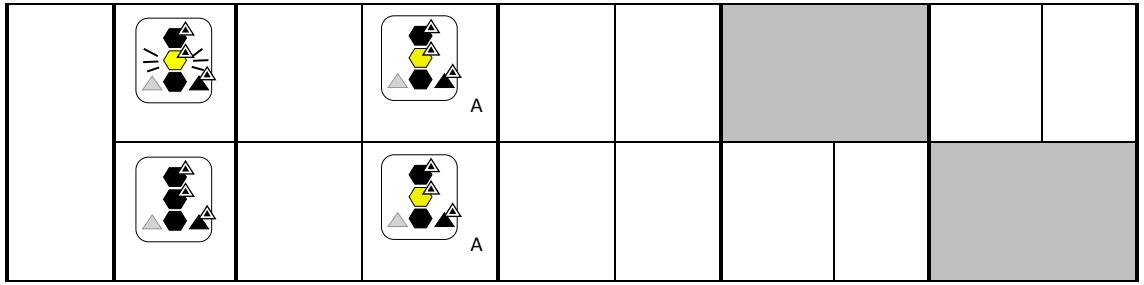
Lights form aspects specification.

General control signals	Implementing					
	Primary form	Transitional forms				
			 A			
	logically vehicular red	logically vehicular yellow	logically vehicular alternative	logically vehicular yellow and red		
						
	logically vehicular additional green	logically vehicular additional green flashing				

						
	logically vehicular yellow flashing					
						
	logically vehicular absent					

Transitions specification

		to general control signal:							
									
		E	B	E	B	B	B	E	B
from general control signal:									
									



3.2 Movement geometrical directions – left-oriented (> 90 grad)

[similar to 3.1]

4. Signal scheme [T.x.o]$_A$ (additional)

4.1 Movement geometrical directions – right-oriented (from 0 to 90 grad)

Lights form aspects specification.

	Implementing					
	Primary form	Transitional forms				
General control signals			 A			
	logically vehicular red	logically vehicular yellow	logically vehicular alternative	logically vehicular yellow and red		
						
	logically vehicular green	logically vehicular green flashing				
			 A			

	logically vehicular additional green	logically vehicular additional green and yellow	logically vehicular additional green and alternative yellow	logically vehicular additional green flashing and red	logically vehicular additional green and yellow and red	logically vehicular additional green flashing and yellow and red
						
	logically green accented	logically green accented flashing	logically vehicular accented additional green and green flashing	logically vehicular accented additional green flashing and green		
						
	logically vehicular yellow flashing					
						
	logically vehicular absent					

Transitions specification

from general control signal:	to general control signal:											
		E	B	E	B	E	B	B	B	E	B	E	B
			A				A						
			A				A						

4.2 Movement geometrical directions ⟁ –left-oriented (> 90 grad)

[similar to 4.1]

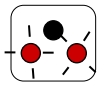
Traffic lights system W

Canonical lights layout

1. Signal scheme $[W]_o$

1.1 Movement geometrical directions ▲ – any

Lights form aspects specification.

	Implementing					
General control signals	Primary form	Transitional forms				
						
	logically vehicular red flashing					
						

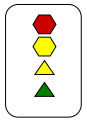
	logically vehicular yellow flashing					
						
	logically vehicular absent					

Transitions specification

from general control signal:	to general control signal:					
						
	E	B	E	B	E	B
						
						
						

Traffic lights system TE

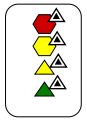
Canonical lights layout

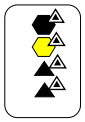


1. Signal scheme [TE.2]$_o$

1.1 Movement geometrical directions ▲ – any

Lights form aspects specification.



General control signals	Implementing					
	Primary form	Transitional forms				
			 A			
	logically vehicular red	logically vehicular yellow	logically vehicular alternative	logically vehicular yellow and red		
						
	logically green accented	logically green accented flashing				
						
	logically vehicular yellow flashing					
						

	logically vehicular absent					

Transitions specification

from general control signal: \ to general control signal:								
	E	B	E	B	E	B	E	B
		A						
		A						

Traffic lights system p

Canonical lights layout

1. Signal scheme [p]$_o$

1.1 Movement geometrical directions ▲ – any

Lights form aspects specification.

General control signals	Implementing					
	Primary form	Transitional forms				
			 A			
	logically pedestrian red	logically pedestrian yellow	logically pedestrian yellow alternative			
						

	logically pedestrian green					
						
	logically pedestrian absent					

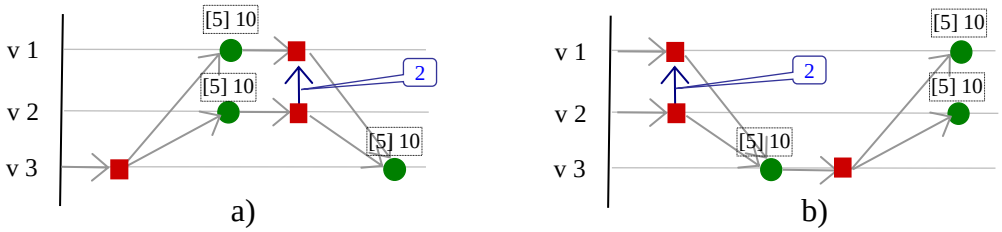
Transitions specification

from general control signal:	to general control signal:					
						
	E	B	E	B	E	B
						
						
		 A				

Appendix C: Cyclic Patterns

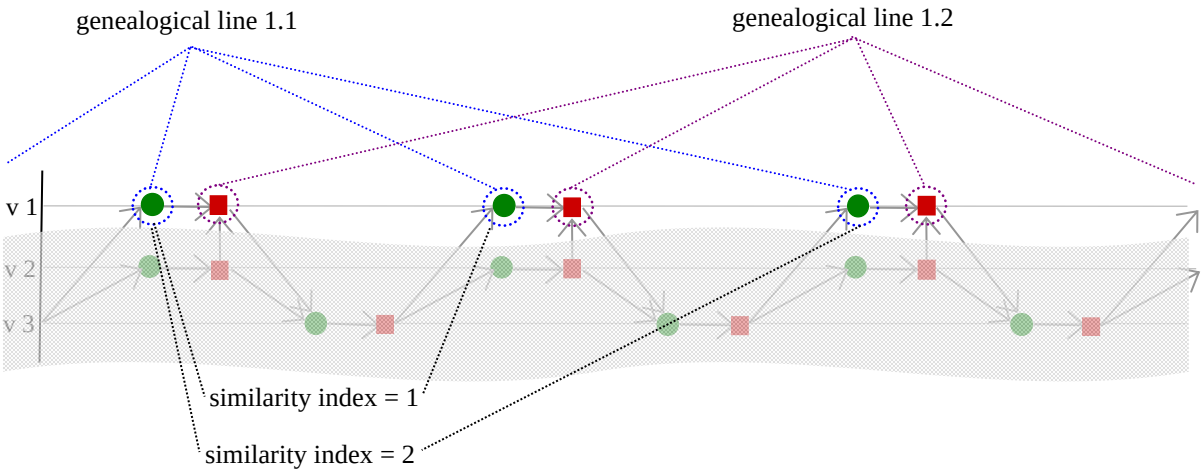
An events map is *regular* if it can be presented as made of repetitions of the same finite fragment. The corresponding repeating fragment is *a rapport.*

A *prime* rapport – the rapport which cannot be presented as made of repetitions of another rapport. Otherwise, the rapport is *compound.*



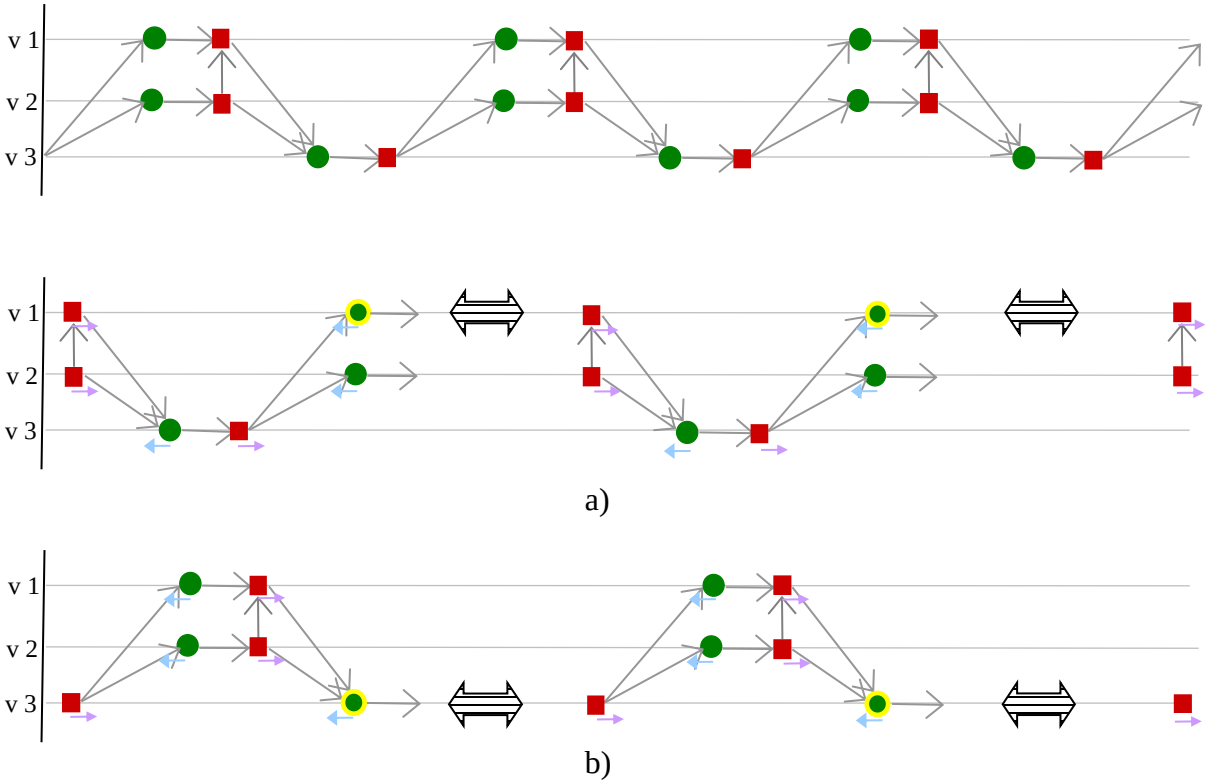
Prime rapports variants of the same signal map

"Cloned" by prime rapport repetitions events are *similar* events or "*twins*"-events and form particular *genealogical line*. Difference in ordinal numbers of "cloning" of one "twin" relative to another is *similarity index*.



Signal events map is *g-strongly connected* if there is a directed (by precedence relation) path between events of every pair of genealogical lines.

Regular g-strongly connected signal events map *natural sectioning* is its partitioning basing on the events occurrence grouping in the signal diagram built on the map in situation where main aftereffect of a chosen series of similar events tends to infinity. The resulting partition element is *natural rapport*. The corresponding event (and its "twins") is (*natural*) *rapport support event* (see fig. below).



Example of the natural sectioning variants.

The support events is highlighted with yellow.

A regular signal map is *3g-strongly connected* if for any pair of genealogical lines there is a path from event of first genealogical line to similar event with similarity index not greater 3 which includes an event of second genealogical line.

Signal event is *proper* if its signal differs from one of a previous event.

Universal reference event of prime rapport is a first in order of signal group numeration event which fulfill conditions:

1) its signal gives right of way.

2) its occurrence can be directly observed and recognized within prime rapport (i.e. it is a proper event, and it is unique event with such signal within signal group and rapport);

In cases where several events fulfill the conditions 1), 2), the event with a signal with lowest rank to be chosen as the universal:

Table

Meta-Signal	rank
logically vehicular accented green	0
logically vehicular common green	1
logically vehicular additional green	2
logically vehicular yellow flashing	3
logically vehicular absent	4
logically pedestrian green	5
logically pedestrian absent	6

Prime rapport event *universal identifier* includes:

1) event signal group number.

2) event control signal.

and,

3) if the event with the control signal is not unique for the signal group, additionally,

3.1) ordinal number of occurrences of the event (or its “twin”) in a range of proper events with the given signal on the signal group after (by precedence relation) universal signal event. (Represented with a left upper index in the form ‘/’ – for 2, ‘//’ – for 3, etc., see figure below);

3.2) ordinal number of occurrences of the event (or its “twin”) with the given signal on the signal group within continuous series of events with the same signal. (Represented with a right lower index, ‘1’ – for 2, ‘2’ – for 3, etc.).