AUTOMATED DETECTION OF HUMAN ERRORS BASED ON MULTIPLE PARTIAL STATE SPACES

D. Gamrad[1], H. Oberheid[2], D. Söffker[1]
[1]University of Duisburg-Essen, Germany, [2]German Aerospace Center, Germany

Corresponding author: D. Söffker, University of Duisburg-Essen, Chair of Dynamics and Control
47057 Duisburg, Lotharstraße 1, Germany, soeffker@uni-due.de

Abstract. This contribution proposes a novel approach to detect human errors automatically. The approach is based on previous work, where the interaction between human operators and technical systems is formalized using a Situation-Operator-Modeling approach and implemented using high-level Petri Nets. The analysis was based on an automatically generated complete state space of a Petri Net. Now, multiple partial state spaces of limited size are used to reduce the computational effort. Furthermore, the model of the considered Human-Machine-System is extended with respect to a detailed definition of sub goals. As an example, the formalization and automated detection of the human error 'rigidity' is presented. Compared to previous work of the authors, the handling of more complex systems and more extensive simulations now becomes possible.

1 Introduction

The development of assistant systems (for human driver as example, cf. [4, 11, 9]), supporting human operators during supervisory tasks [16] is a field with increasing significance and a wide range of applications. Typical examples are safety critical Human-Machine-Systems like the control and supervision of the operation of nuclear power plants or airplanes. In these scenarios the human operator's flexibility is in practice often used as a fallback level to handle unexpected dangerous situations. However, the human interaction behavior is also not free of various kinds of errors [3, 15]. Here, assistant systems analyzing the Human-Machine-Interaction (HMI) could help the human in critical situations.

This contribution proposes a novel approach to detect human errors automatically. The approach is based on previous work, where the interaction between human operators and technical systems is formalized using a Situation-Operator-Modeling (SOM) approach and implemented using high-level Petri Nets [6]. The analysis was based on an automatically generated complete state space. Now, multiple partial state spaces of limited size are used to reduce the computational effort.

First, the applied approach including the modeling with SOM, the simulation with software for high-level Petri Nets, and the analysis based on automatic generated state space is described in Section 2. Then, the implementation of the concept with an simulation environment named 'HMI Analysis Architecture' is presented in Section 3. In Section 4, the extension of the applied approach with respect to the usage of multiple partial state spaces and its technical realization is described. As an example, the formalization and detection of the human error 'rigidity' is given in Section 6. Finally, Section 7 presents an example simulation and experimental results followed by a summary and outline of future work.

2 Simulation and analysis of human behavior

In [6, 18] an approach for automated detection of human errors in human interaction with complex dynamical systems is presented by formal representation [18] and related automatic detection [6]. As a basis of the approach, the interaction between a human operator and a technical system is formalized using a Situation-Operator-Modeling (SOM) approach [18, 17]. The implementation of a SOM-based model of interactions is realized using Coloured Petri Nets (CPNs) [7, 8]. From the CPN model in [6] a full state space can be generated, which contains all possible situations of the system. The considered human errors are described by formal query functions in a generic manner and are detected by analyzing the state space. The whole sequence from modeling to analysis is described in the following and visualized in Figure 1.

2.1 From the real world to the model

The interaction between cognitive systems, like humans, higher animals, or specific technical systems and the environment can be formalized using the Situation-Operator-Modeling (SOM) approach [17]. Core of this approach is the assumption that changes of the considered parts of the real world are understood as a sequence of effects described by the items scenes and actions. The item situation, which is in contrast to McCarthy [10] a time-fixed, system-, and problem equivalent one, is used describing the internal system structure (as part of the real world). A situation consists of characteristics, which are linked via relations. The item operator is used to model actions which change scenes (modeled as situations) in time. An operator transfers a situation to the following by changing the characteristics, the relations or both.

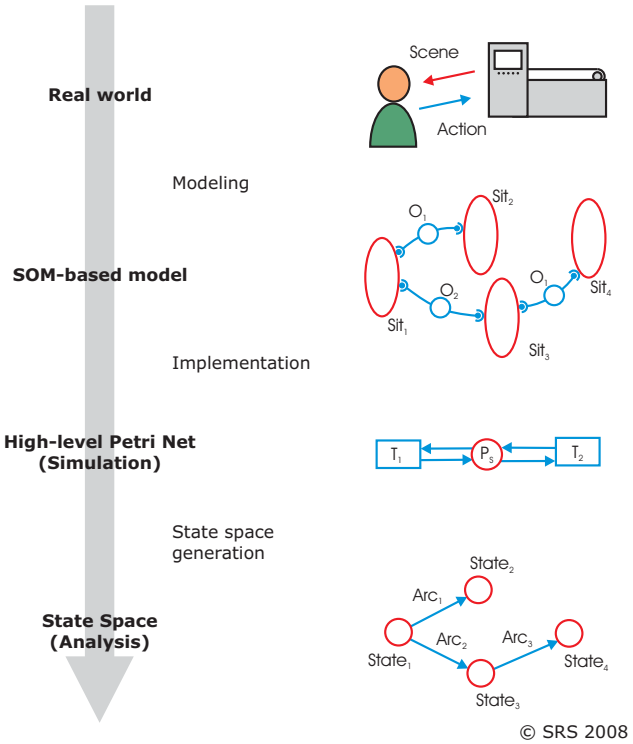

Figure 1: From modeling to analysis of interaction

The SOM approach can also be used to formalize human errors [17]. According to Dörner [3, 15], human errors are classified with respect to the interactions of humans in complex dynamical systems. Coming from psychology, word models are used to describe and distinguish different human errors, which are divided into the four main clusters goal elaboration, decision processing, control of the actions, and errors due to internal cognitive organization problems. As an example, Section 5.1 describes the SOM-based formalization and detection of the human error 'rigidity' according to Dörner's classification, but using instead of word-based description a description to be able to be used with processors, due to the formal base.

2.2 From the model to the simulation

The introduced SOM technique provides a qualitative modeling approach starting from a symbolic notation to structure and investigate human interaction with technical systems. In order to analyze complex systems with many degrees of freedom, computer-based representations of SOM models have to be built and simulated. The computer-based simulation of SOM-based models has formerly been realized in the form of both textual programming languages [1] and high-level Petri Net (HPN) formalisms [5]. The latter approach based on HPNs combines the advantages of a graphical representation with the benefits of formal executable model. In [5] the possibilities and benefits of developing specialized net patterns based on existing formalisms and tools for HPNs to simulate and analyze selected properties of SOM-based models are studied. According to [2], HPNs are defined as a class of net formalisms, which extends the classical Petri Net formalism originally introduced by Petri [13] (consisting of places, transitions, arcs, and black, uniform tokens) mainly in that HPNs allow several differing tokens (such as numerical values, data structures or complete software objects) at the same time on the same place and often support net hierarchization.

The motivation to consider HPN formalisms for the simulation of SOM-based models is due to a number of reasons. First of all, certain structural similarities exist between SOM and HPNs. They are both inherently bipartite formalisms consisting of an active element representing the system functionality and a passive element representing the system state. In this respect, a natural and intuitive correspondence exists between the notion of operators in SOM and transitions in HPNs. The meaning of the terms situation in SOM and of places (in combination with a token) in HPNs is also closely related.

Finally, some existing Petri Net software, such as CPN Tools, come with a well developed repertoire of basic graph theoretic analysis techniques. These are useful to prove individual properties on SOM-based models and allow analyzing interaction described by SOM-based approaches. The possibility of CPN Tools to generate and analyze model state spaces is one main reason for choosing CPN Tools as the basis for implementation in this work as will be detailed in the following.

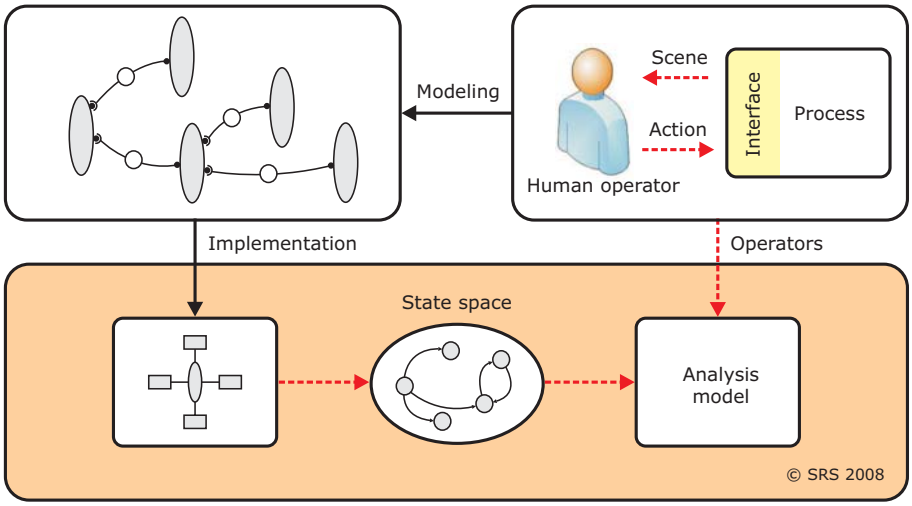

Figure 2: HMI Analysis Architecture with closed loop between state space and simulation

2.3 From the simulation to the analysis

State spaces of Coloured Petri Net models are discrete, directed graphs (digraphs). The digraph contains as nodes of the graph different states (markings), which the model may obtain, starting from a certain initial state and given the models constraints. The arcs of the graph represent possible transitions between these states. Adopting an appropriate implementation technique and net patterns to model SOM in CPN Tools as presented in [5], the states are interpreted as SOM-situations while the state transitions are associated with SOM-operators.

State spaces can be generated automatically in CPN tools. Depending on the properties of the model and available computational resources, the resulting state spaces may be complete (representing all reachable states) or only partial (representing a subset of all reachable states e.g. within a certain search depth with regard to the initial condition). Accordingly, it becomes possible to automatically determine a (partial or complete) set of reachable SOM-situations and calculate possible SOM-operator-sequences between these situations.

Using the calculated space of reachable situations is a helpful approach for the investigation of human errors in Human-Machine-System (HMS) interaction, since it allows relating the observed actions of the human operator to the (known) set of reachable situations and executable operators of the system. On the basis of the state space, it becomes possible to formally determine desirable (goal) and non-desirable (unsafe) states/situations reachable by the HMS and then check if the human operator's actions serve to approach (come closer to) a certain goal situation. It is also possible to observe if an action sequence serves to pursue one single goal consistently or iterates/jumps between various competing goals etc.

The automatic detection of human errors within the interaction of the system can be realized through formal state space query functions programmed in CPN Tools. In order to keep the functions reusable for different kinds of Human-Machine Systems, the queries should be built in a manner that the structure of the error detection is generic and remains the same independent of the specific system. Only the concrete definitions of what desirable/non-desirable/ goal situations mean in a certain application context are system specific and have to be exchanged.

3 HMI Analysis Architecture

The approach presented in Section 2 was already realized and tested successfully for different human errors by a simulation environment named HMI Analysis Architecture. In Figure 2, the connections between real world, modeling, and analysis are visualized. Here, the interaction between a human operator and a (real or simulated) process is modeled using the SOM approach. This model is implemented with high-level Petri Nets. From the Petri Nets describing the interaction of the real world a state space is generated. An analysis model uses the state space as well as the actions of the real world (modeled as operators) as inputs to evaluate the behavior of the human.

As an example, the interaction between a human operator (gamer) and a computer (arcade game [14]) was chosen. The arcade game enables the design of custom scenarios and is easy to replace by other simulated or real technical processes. The modeling and analysis of the interaction is realized by the software CPN Tools providing modeling and simulation of CPNs as well as formal analysis based on an automatically generated state space. In the following, the arcade game and the modeling of the interaction between human operator and arcade game is described in detail.

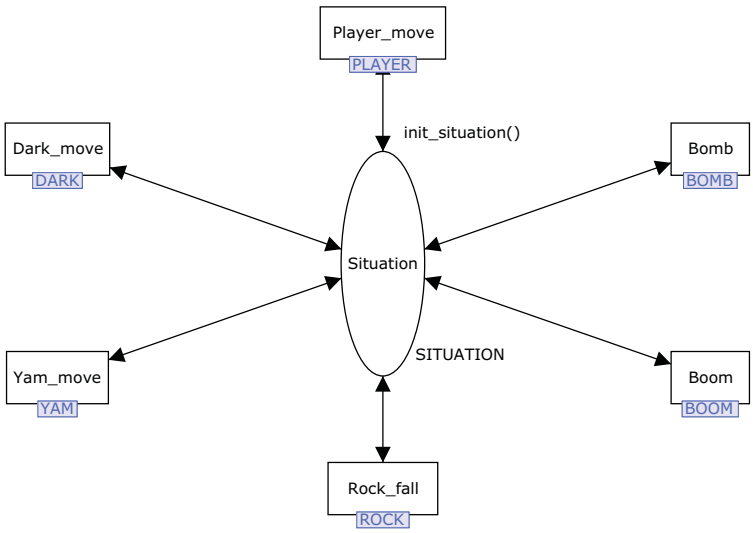

Figure 3: Modeling of interactions in CPN Tools

3.1 Arcade game application example

In this contribution, the specific process considered is an arcade style game (like 'Boulder Dash' or 'Sokoban') [14] providing a graphical interface to a human operator and a TCP/IP communication with CPN Tools. Within this formalizable synthetic environment an agent has to be controlled within a grid-based environment. The agent is able to perform six different kinds of actions ('move up', 'move down', 'move left', 'move right', 'snap field', and 'drop element'). The environment consists of static and dynamic elements with different behaviors combinable to complex scenarios using the integrated level editor. In general, the task of the human operator consists of first picking up a certain number of 'emeralds' and then finishing the level by leaving the scenario through an exit door.

In the current development state the arcade game was chosen due to its simple handling by editing own levels and custom elements. Given this possibility, environments can be designed, which purposely lead the human operator to commit certain errors. Nevertheless, the structure of the experimental environment and the developed functions for error detection are not specific to the arcade game. This enables the replacement of the arcade game by other simulated or real technical processes using the same concept and error detection mechanisms.

3.2 Modeling of arcade game interactions

The possible interactions within the arcade game are modeled using HPN-patterns for SOM-based models. In Figure 3, a part of the interaction model representing a subset of possible action consisting of the moving actions of the agent, the dropping and explosion of bombs, the falling of stones, and the moving actions of light- and dark-colored monsters is illustrated. All possible actions are connected to the same place. With an increasing number of actions the net can be designed more clearly by the usage of substitution transitions and fusion places. The functions of the transitions are detailed on subpages on a lower hierarchical level.

The situation is modeled by a place consisting of one token, which is a set of data types. Here, the situation consist of twenty characteristics, consisting of 'x-position of the agent', 'y-position of the agent', 'vitality of the agent', 'x-position of light-colored monsters', 'y-position of light-colored monsters', 'direction of light-colored monsters', 'x-position of dark-colored monsters', 'y-position of dark-colored monsters', 'direction of dark-colored monsters', 'positions of collectible emeralds', and 'number of needed points' etc. In the whole model, the actions of the agent as well as each action of the independent acting monsters are represented by transitions, which are linked to the place. The moving actions of the agent and the moving actions of all light-colored and dark-colored monsters respectively are represented by one transition, which combines several equal operators. Through the firing of a transition the corresponding operator is performed and the situation and token on the place respectively is changed.

4 Analysis of multiple partial state spaces (MPSS)

Using the approach demonstrated in [6], the online detection of human errors is possible for systems with small complexity. With growing complexity of the considered system, the computational effort for the generation of complete state spaces increases exponentially. Also, the execution of state-space-based queries needs increasingly more time and quickly becomes prohibitive. To solve this problem, a modified analysis approach based on the generation of multiple partial state spaces is proposed here, which is described in the following.

4.1 Conceptual idea

This contribution investigates a novel approach to detect human errors based on incomplete state spaces. Instead of attempting to generate one full state space a priori with all possible system states starting from the initial system state (before actually starting a simulation), multiple partial state spaces of limited size are computed a posteriori starting from observed intermediate states which actually occurred during simulations. Compared to the approach presented in [6] the handling of more complex systems and more extensive simulations now becomes possible.

Regarding the definition and detection of human errors based on incomplete state spaces (when not all reachable situations and situation trajectories are known) the modeling and the analysis also have to be modified. The goal directed behavior can no longer be evaluated with respect to a set of known final goal situations as it was possible in the case of a complete state space. To compensate for that, the definition of sub goal situations becomes necessary. Furthermore, the analysis of several state spaces may be required to calculate all necessary information for the error detection.

4.2 Technical realization

The technical basis of the employed approach builds on previous work [12]. There, a new technique was presented for the integrated and repetitive application of simulation and state space analysis within a single automated run of a Coloured Petri Net model. Conventionally, CPN Tools without extensions only supports either state space analysis or simulation starting from an initial model state. It then requires manual user interaction over the GUI when switching between or repeating these actions. This requirement makes the calculation of a large number of partial state spaces for different initial states inefficient or infeasible for practical purposes.

The technique and code extensions developed in [12], however, support the automatic (re)initialization of the CPN model and execution of state space analysis from an external process, supporting the iteration between simulation and state space analysis according to arbitrary control flows and passing of results from one simulation or analysis step to the next. All necessary functions to control CPN Tools in that way (reset, initialize, simulate, calculate state space, query state space) are defined in a separate code structure. In [12] the technique was applied to an example of a cognitive technical system. Due to the simplicity of the system, full state spaces could be calculated during each cycle. In this work presented here, the technique is extended to control the calculation of multiple partial state spaces for human error detection. The size of each state space can be defined in terms of maximum number of states, time horizon or calculation time according to the characteristics of the error which is searched for.

5 MPSS-based detection of human errors

As an example for the MPSS-based detection of human errors, this Sections describes the formalization and detection of the human error ’rigidity’ [3, 15] based on a full state space as well as the differences resulting from the extension with partial state spaces.

5.1 SOM-based formalization of rigidity

According to [3, 15] the human error ’rigidity’ describes an human behavior, where a human operator adhere rigidly to a previous planned strategy although due to external effects a change would be necessary and more efficient respectively.

The human error ’rigidity’ is visualized with SOM notation in Figure 4 [18, 17]. The situations (S_i) including the characteristics (A_i, B_i) and relations (R_i) are represented by gray ellipses and the operators (O_i) are represented by white circles. In the situation trajectory depicted in the lower part of the figure, the desired situation S_2 is not reached as planned in the upper part, due to external effects and disturbances. Instead, a different and unexpected situation S_{2a} is resulting. Given the new situation, O_2 will not lead to the desired goal due to the changed structural situation S_{2a}. The human error ’rigidity’ includes that the known and previously planned operator O_2 is nevertheless realized inconsiderately by the human operator, although the assumptions for its application no longer hold.

5.2 State-space-based detection of rigidity

Through the formalization of ’rigidity’ with formal query functions the automated detection is possible. Therefore, a set of possible final goal situations is calculated using the full state space. From this situations and the observation of user actions, a set of user goal situations (those final goal situations to which the user actions are directed) can be derived. Both the possible final goal situations and the user goal situations are updated after every observed action. The detection of ’rigidity’ itself is based on two conditions, which have to be fulfilled both. The first condition is fulfilled, if a detected user action is not directed to a final goal situations and the second condition is fulfilled, if the detected user action was directed to a previous user goal situations. This implicates the occurrence of a user independent action, which was not detected or ignored by the user corresponding to the definition of ’rigidity’.

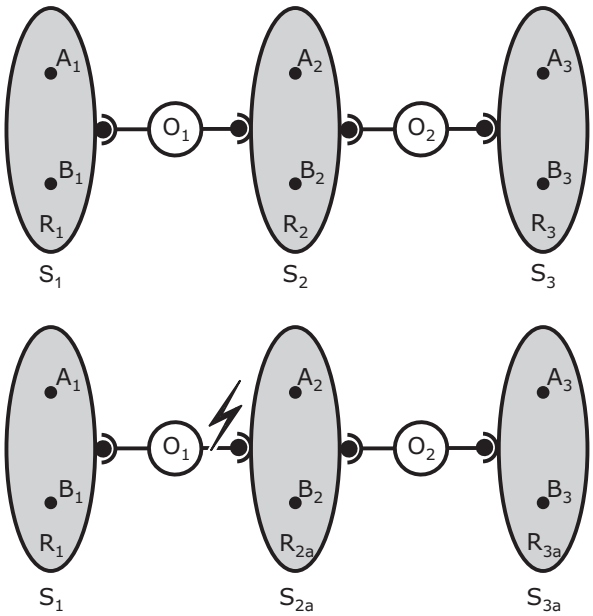

Figure 4: SOM-based formalization of the human error rigidity [18, 17]

5.3 MPSS-based detection of rigidity

As described, the detection of the human error 'rigidity' requires a set of known final goal situations. However, these situations are usually not contained in a partial state space. A first step to solve this problem is the definition of sub goal situations. Thus, goal situations could, but do not necessarily have to be, contained in a partial state space, which leads to the idea of multiple partial state spaces. If a partial state space is not large enough to offer all necessary information for the unambiguous detection of 'rigidity', a further state space could be calculated based on the previous analysis.

The MPSS-based detection of 'rigidity' is built on the conditions mentioned in Subsection 5.2. Here, a two-stage offline process is proposed. In the first step, each occurred action is analyzed based on a separate partial state space with a certain number of states x_s to exclude 'rigidity'. Hence, the number of states x_s is depending on the complexity of the considered system and has to be detected for every modeled interaction experimentally. All actions have to be considered as user actions, because of the fact that a certain action of the process could also be a 'waiting-action' of the user. So, 'rigidity' can be excluded by the negation of the three conditions

- the considered action is not goal-directed,
- the previous action was goal-directed, and
- the considered action would be directed to the goal neared by the previous action.

The critical situations, where 'rigidity' remains possible, are analyzed in the second step more detailed. Here, a larger partial state space is calculated to check for the defined conditions. If 'rigidity' can not be excluded again, the critical situations could be stored, e.g. for a closing discussion. It has to be mentioned that an MPSS-based analysis can not guarantee that an action is not directed to a goal, since the goal may lie beyond the horizon of the calculation.

At the beginning of an interaction sequence, it often can not be detected whether an action is goal directed or not. Due to this, all actions before the first action is detected as goal-directed are ignored. To minimize the ignored action sequence the number of states x_s can be enlarged.

6 Experimental example

The automated detection of the human error 'rigidity' was implemented and tested for the arcade game application introduced in section 3.1. At first, the used scenario is described followed by an example simulation. The detection of 'rigidity' is described and visualized by a subset of the related state space.

6.1 Scenario

The interface of the example scenario is shown in Figure 5. The human operator has to control the humanly looking agent. He has to pick up at least one emerald and leave the level by reaching the exit door in the lower right corner. Beside the agent, controlled by the human operator, there are two hostile monster agents. These agents move

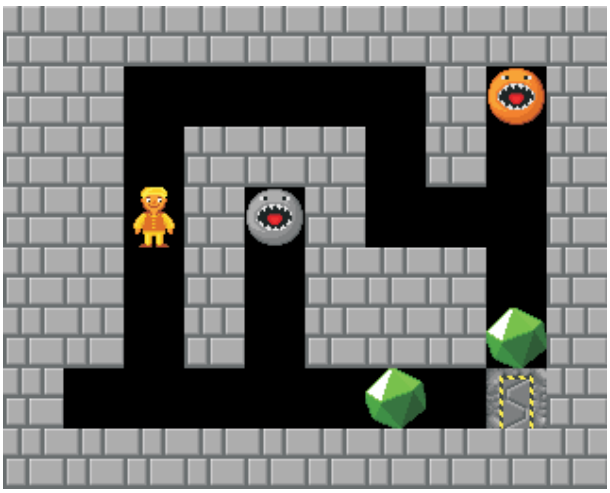

Figure 5: Initial scene of the example scenario

autonomously and pose a threat to the player's agent exposing different behaviors. The light-colored monster (on the right side) moves to one direction. If it reaches an emerald or a wall it changes the moving direction to the opposite side. The dark-colored monster (in the middle) moves to one direction, too. However, in the lowest position, it decides weather it keeps the vertical moving direction or changes to a horizontal moving direction. Furthermore, it is able to 'eat' the emeralds.

At the beginning of the game, the human operator gets a short instruction regarding the control of the agent and the goal of the level, which has to be reached. The behavior of the autonomously moving monsters is not explained and has to be determined by observation. Due to the similar look and the apparently equal behavior, the human operator could assume that the behaviors of both monsters are totally equal. Hence, the anti-clockwise way seems to be safer. After starting to control the agent to the lower emerald, the dark-colored monster could however change its behavior to horizontal movements and pick up the lower emerald before the operator. If this is not recognized by the human operator the human error 'rigidity' may occur. This means that the agent keeps on pursuing a goal, which is no longer reachable.

6.2 Simulation

Before the first action, the human operator observes the behavior of both monsters. After these monsters have returned to their initial locations and start again (Figure 5), the human operator starts to control the agent to the bottom emerald. The exact sequence of actions observed in this example is then: dark-monster-down (1), light-monster-down (2), agent-down (3), dark-monster-down (4), light-monster-down (5), agent-down (6), dark-monster-down (7), light-monster-down (8), agent-down (9), dark-monster-right (10), light-monster-up (11), and agent-right (12).

The full state space of the example scenario consists of 3661 states. Although, the analysis of the full state space would be possible, this easy scenario is chosen to keep the example clear and traceable. As described in Section 5.3, at the first partial state spaces are calculated after each action in the interaction sequence. The parameter x_s is set with the value 500. Before the 3rd action no subgoal is in the corresponding state spaces. Hence, the analysis ignores all previous actions and checks whether the following actions are directed to one of the subgoal contained in the partial state space. The 6th and 9th action (performed by the user) are directed to a subgoal. Due to the fact that all three conditions are fulfilled for the 12th action 'rigidity' is possible. In a second step, a larger state space with 1000 nodes is calculated. Here, the result of both analysis steps are equal.

In Figure 6, a subset of the calculated partial state space with the situation after the 9th action as origin is visualized. The ellipses represent situations, which are linked by the operators. The green-colored situations are user goal situations within the calculated state space. The dotted lines without inscriptions represent meta-operators consisting of a sequence of simple operators. The occurrence of the human error 'rigidity' is marked by the large circle.

At the beginning of the visualized sequence, it can been seen that the action of the dark-colored monster affects the behavior of the user relevantly. If the dark-colored monster moves up, the current user goal situations are still reachable. Otherwise, if the dark-colored monster moves to the right, no paths from the following situations to a subgoal are contained in the current state space. If the user ignores the action of the dark-colored monster and tries to realize the old strategy (left side of Figure 6) the action between the 7th and 17th situation fulfills all conditions for 'rigidity'.

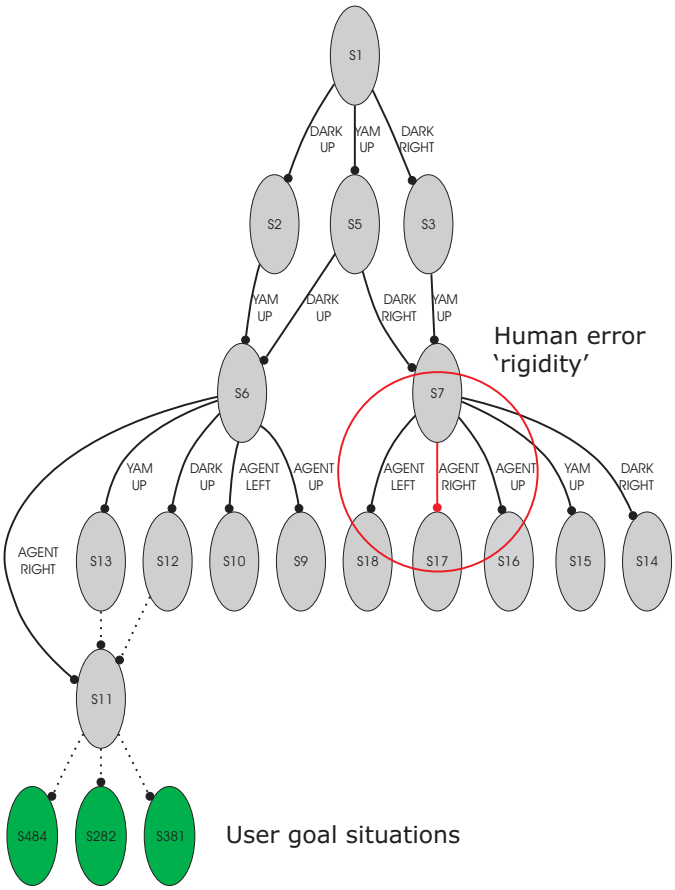

Figure 6: Example simulation

7 Summary and future work

The contribution presents an extension of an approach to simulate and analysis Human-Machine-Interaction. The interaction is formalized using the Situation-Operator-Modeling approach. The simulation and analysis is realized with the software CPN Tools providing the automatic generation of full or partial state spaces. In addition to previous work, the contribution focuses on the analysis of partial state spaces. Furthermore, a novel method realizing the automatic switch between simulation and state space generation, enables the analysis of several small space spaces. As an example the detection of the human errors 'rigidity' is described in detail followed by an example simulation.

In the future, the presented approach will be applied also to other kind of human errors. Furthermore, the analysis of human interaction behavior in real technical processes will be focused.

8 References

[1] Ahle, E. and Söffker, D.: *Interaction of intelligent and autonomous systems – part II: realization of cognitive technical systems*. In: Mathematical and Computer Modelling of Dynamical Systems, Vol. 14, No. 4, August 2008, 319–339.

[2] Baumgarten, B.: *Petri Netze – Grundlagen und Anwendungen*. 2nd ed., Spektrum Akademischer-Verlag, Heidelberg, 1996.

[3] Dörner D.: *Die Logik des Misslingens, Strategisches Denken in komplexen Situationen*. Rowohlt Verlag, 2002.

[4] Fiala, E.: *Lenken von Kraftfahrzeugen als kybernetische Aufgabe*. In: Automobiltechnische Zeitschrift 68, 1966, 156–162.

[5] Gamrad, D.: *Entwicklung von Mustern höherer Petri Netze zur rechnergestützten Simulation und Analyse von Situations-Operator-Modellen*. Diploma thesis, University of Duisburg-Essen, Chair of Dynamics and Control, 2006.

[6] Gamrad D., Oberheid H., Söffker D.: *Formalization and Automated Detection of Human Errors*. In: SICE International Conference on Instrumentation, Control and Information Technology, Tokyo, Japan, 2008, 1761–1766.

[7] Jensen K.: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Vol. 1-3. Springer-Verlag, 1997.

[8] Jensen K., Christensen S., Kristensen L. M.: *CPN Tools State Space Manual*. University of Aarhus, Department of Computer Science, Aarhus, 2006.
[9] Maurer M., and Stiller C.: *Fahrerassistenzsysteme mit maschineller Wahrnehmung*. Springer Verlag, Heidelberg, 2005.
[10] McCarthy, J.: *Situations, actions and causal laws*. Stanford University, 1963.
[11] Naab, K.: *Automatisierung bei der Fahrzeugführung im StraSSenverkehr*. In: at - Automatisierungstechnik 48, 2000, 211–223.
[12] Oberheid H., Gamrad D., Söffker D.: *Closed Loop State Space Analysis and Simulation for Cognitive Systems*. In: 8th International Conference on Application of Concurrency to System Design, Xian, China, 2008, 39–44.
[13] Petri C. A.: *Kommunikation mit Automaten*. Ph.D. thesis, University Bonn, Bonn, 1962.
[14] Rocks'n'Diamonds Website by Arcsoft Entertainment, http://www.artsoft.org/rocksndiamonds/.
[15] Schaub H.: *Modellierung der Handlungsorganisation*. Verlag Hans Huber, Bern, 1993.
[16] Sheridan T. B.: *Telerobotics, Automation and Human Supervisory Control*. The MIT Press, Cambridge, London, 1974.
[17] Söffker D.: *Systemtheoretische Modellbildung der wissensgeleiteten Mensch-Maschine-Interaktion*. Logos Wissenschaftsverlag, Berlin, 2003. - also: Habilitation Thesis, Bergische Universität Wuppertal, Germany, 2001.
[18] Söffker D.: *Understanding MMI from a system-theoretic view - Part I and Part II*. In: Proc. 9th IFAC, IFIP, IFORS, IEA Symposium Analysis, Design, and Evaluation of Human-Machine Systems, Atlanta, Georgia, USA, 2004.