An Approach to the Verification of Material Handling Systems

Thomas Klotz, Bernd Straube, Eva Fordran, Jürgen Haufe
Fraunhofer Institute for Integrated Circuits
Design Automation Division
Zeunerstr. 38, 01069 Dresden, Germany
thomas.klotz@eas.iis.fraunhofer.de

Frank Schulze, Karsten Turek, Thorsten Schmidt
Chair of Logistics Engineering
Institute of Material Handling and Industrial Engineering
Technische Universität Dresden, Germany
frank.schulze@mailbox.tu-dresden.de

Abstract

The development of correctly working logistic systems is a tedious task. On the one hand, the developer is faced with the increasing complexity of systems and shrinking time-to-markets, but on the other hand, the need for reliability and safety of the implemented controls becomes more and more important. Formal verification techniques such as model checking allow for proving whether a system completely fulfills its specification. Existing work, though, considered only the verification of single controllers, but did not analyze the behavior of a complete logistic system.

In this paper, an approach to the formal verification of material handling systems is presented. The approach is based on the definition of material handling system elements and their interconnection. Experimental results show that the approach can ensure the correct functionality of logistic systems.

1. Introduction

In the last couple of years, the impact of logistic systems increased significantly in a variety of industrial sectors, e.g. in manufacturing and transport. On the one hand, the correctness of the implemented systems plays a crucial role, but on the other hand, the design has to be achieved at a minimum cost. A typical field of application for logistic systems are baggage handling systems at airports. The increasing globalization and the fact that more and more people travel, result in ever-increasing demands to these systems. The design of such baggage handling systems, which may include several hundreds or even thousands of single system components, is a cumbersome process. As a result, there were more than a million travelers waiting for their delayed luggage – only considering European airports [2].

With increasing size and heterogeneity of logistic systems, the design and validation of the functional behavior of these systems becomes a hard to solve and error-prone task. Even though simulation tools provide a means to analyze and validate parts of the behavior of the logistic systems, these techniques are not able to make propositions about the complete behavior of the system. In contrast, formal methods are capable of proving the soundness of a system with regard to its specification. However, to the best of the authors’ knowledge, these methods have not been considered for verifying logistic systems yet.

In order to standardize the structure of the overall control of logistic systems, the VDI guideline 3628 [1] provides a definition for the abstraction layers of the control of automated material handling systems (cf. Fig. 1).

There has only been little work on the verification of material handling systems (MHS) [8] on higher layers of abstraction (e.g. layers MFCS and CSC) so far. In [6], the main advantages of introducing formal methods in the field of MHS are presented, and a simple formal model for a conveyor is given. The authors conclude that the modeling of MHS is non-trivial and time-consuming.

This paper presents an approach to the formal verification of material handling systems. The approach is based on the methodology of modeling single material handling system elements. By interconnecting these elements, a formal model of the logistic system is built, which thereafter can be fed into a model checker in order to check the properties on the model. With this, requirements to the logistic system can be automatically proven. The paper extends the authors’ previous work [13] by exemplarily showing how to enable global routing strategies, which in-

deed allow to specify and verify more complex MHS systems. Moreover, further MHS elements are presented, and more comprehensive experimental results are discussed.

The rest of the paper is structured as follows. Section 2 gives a brief explanation of model checking, which is the formal technique applied here. The modeling methodology for MHS is discussed in Section 3. The interconnection of MHS elements is thereafter presented in Section 4. The application of the approach is illustrated with two examples in Section 5. Finally, Section 6 concludes the paper and provides pointers for future research.

2. Model checking

Formal verification is the act of proving or disproving the correctness of the functionality of a system with respect to a specification using mathematical methods. Contrary, with validation methods, such as simulation and test, it is seldom possible to check the complete system, while formal verification methods are capable of making propositions with respect to the complete functional behavior of a system. The formal verification technique model checking [5] allows for automatically proving whether a set of properties representing the specification of a given system holds or its system model or not.

Fig. 2 sketches the process of model checking. Modeling involves the specification of the technical system in terms of a discrete model. For specifying this model, the formalism of finite state machines (FSM) [9] is often considered. Additionally, the given requirements to the system have to be formulated as a number of temporal logic properties. In this paper, Computation Tree Logic (CTL) [3] is utilized for describing the properties. If one or more properties do not hold on the model, a counterexample in terms of a simulation trace is presented to the user to ease debugging.

A drawback of model checking is the so-called state space explosion, which refers to the fact that the state space grows exponentially with the number of variables in the model. Thus, only finite state models with a certain size in terms of the number of states can be analyzed using model checking.

3. Modeling of MHS

In the proposed approach, the technical MHS is decomposed into a quantity of commonly used MHS components. For each of the MHS components, a separate behavioral model, the MHS element, is set up. This MHS model abstracts from the technical system in order to provide a discrete model. The model of the whole technical MHS is obtained by combining instances of the single MHS elements to a network of interconnected MHS elements. This model can then be used for formal verification, simulation, etc.

In the sequel, the behavior of some of the MHS elements is described by state transition tables. The behavior of an MHS element is modeled in a modular way, i.e. it is implicitly given as a network of FSMs. The set of states $S$ of an FSM can be empty in order to specify static behavior. For a state $s \in S$, the initial state is represented by $s(0)$ and the next state by $s'$. The corresponding output variable of a state variable is marked by a prefixed $o$.

In order to discretize space and time of the technical MHS, the following assumptions and restrictions are applied:

- The length of a cargo is 1 Length Unit (LU).
- At each discrete time step $t$, a cargo can move 0 or 1 LU.
- At each place, there can be only one or no cargo.
- There is only a finite set of cargos.
- The length of a conveyor is defined by $n$ LU with $n \in \mathbb{N}$.

In this paper, the following notation is used:

- $\mathbb{B}$ represents the set of Booleans $\mathbb{B} = \{0, 1\}$.
- The set $\mathbb{C}$ defines the types of cargos that have to be distinguished. A cargo type is specified by a letter such as $a$, $b$ etc.
- The special cargo type $L \in \mathbb{C}$ represents “no cargo”, i.e. a position where currently no cargo is placed.
- Don’t cares are represented by a “$\_\_\_$”.

Here, it is assumed that each cargo has the same size. This is typically the case in technical MHS, e.g. when standard pallets, trays, or bins are used as transport aids.

Figure 1. Layer concept defined in VDI 3628

Figure 2. Process of model checking
In addition, it is given that the receipt of a cargo always has to be acknowledged from the succeeding element, before the next cargo can be loaded. Consequently, there is never more than one cargo on an MHS element at a time. This is often the case for tray systems at large airports or pallet conveying systems.

Note: This assumption has been chosen in order to focus on certain types of MHS in first place. However, the proposed methodology can be easily extended to other types of MHS. For this, the behavior of the MHS elements has to be modeled accordingly.

The proposed modeling methodology can only be applied to technical MHS components where the aforementioned discretization is a feasible abstraction of the dynamic behavior of the components. Moreover, stochastic processes that occur in the MHS are not considered yet.

3.1. MHS elements
An MHS element \( M \) is defined as FSM denoted as the 6-tuple \( M = (I, O, S, s_0, \sigma, \lambda) \) such that,

- \( I \) is the set of input variables,
- \( O \) is the set of output variables,
- \( S \) represents the set of states,
- \( s_0 \) marks the initial state,
- \( \sigma \) is the set of transition functions \( \sigma : S \times I \rightarrow S \), and
- \( \lambda \) is the set of output functions \( \lambda : S \times I \rightarrow O \).

Variables and states take on values in some finite domain \( D \). For an MHS element with \( p \) preceding and \( s \) succeeding MHS elements, the set of input variables \( I \) is defined as \( I = (IN, Stop, Pre_1, ..., Prep, Ack_1, ..., Ack_s, Suc_1, ..., Suc_s) \):

- \( IN \) (C): for receiving cargos from the predecessor
- \( Stop \) (B): if \( Stop = 1 \), the MHS element halts
- \( Pre \) (B): \( Pre = 1 \) signals that the preceding MHS element wants to transfer a cargo
- \( Ack \) (B): the variable \( Ack \) becomes 1 whenever the current request is acknowledged, i.e. the last transported cargo of the MHS element arrived at the successor
- \( Suc \) (B): if \( Suc = 1 \), the succeeding MHS element is able to receive a cargo

Moreover, the set of output variables \( O \) is given as \( O = (OUT, Error, Give_1, ..., Give_s, Clear_1, ..., Clear_p, Take_1, ..., Take_p) \), with:

- \( OUT \) (C): for transferring cargos to the successor
- \( Error \) (B): \( Error \) becomes 1 whenever a malfunction has been detected, e.g. if an MHS element transports a cargo without having a request
- \( Give \) (B): if \( Give = 1 \), the MHS element wants to transfer a cargo to the successor
- \( Clear \) (B): becomes 1 whenever a cargo has been received from the predecessor
- \( Take \) (B): signals that this MHS element is able to receive a cargo, i.e. is not currently handling a request

Note: Even though the individual MHS elements are fault-free, errors can occur, e.g. when interconnecting these elements (cf. Section 4). These errors are reported by the \( Error \) variable.

Based on the definitions above, the behavior of the following MHS elements has been modeled:

- Conveyor: transports cargo from a preceding to a succeeding MHS element
- Turntable: couples two conveyors that are arranged in a right angle to each other
- Turntable with priority merge function: receives cargos from two different directions and transfers them to one subsequent MHS element; if both want to transport a cargo, one of the two predecessors is prioritized
- Turntable with divert function: conveys incoming cargos based on their type to one of the two succeeding MHS elements
- Picker: models the process of order picking

Moreover, a number of (non-technical) elements are needed to obtain a closed formal model of the MHS:

- Cargo source: generates arbitrary sequences of cargos of given types
- Cargo sink: to end the life-span of a cargo

Note: Here, only a limited number of MHS elements is discussed. Further MHS elements, though, can be added using the proposed methodology. For the modeling of some MHS elements, it may be necessary to add further inputs and/or outputs to the definition given above.

While in [13] the models of the conveyor and the turntable have been presented, the turntable with priority merge function is discussed in detail in Section 3.2, and the picker is introduced in Section 3.3.

3.2. Turntable with priority merge function
Turntables are used to couple two conveyors that are, in contrast to a straight conveyor line, arranged in a right angle to each other. The modeled turntable element gets a cargo loaded, turns, and then transfers the cargo to the next MHS element. Before the subsequent cargo can be loaded, the turntable has to move back to the loading position.

On the basis of the existing turntable model [13], further basic MHS elements can be derived by modifying its function and allowing more than one predecessor and/or successor. The turntable with priority merge function allows for receiving cargos from two different directions and transfers them to one subsequent MHS element. In the current model, one of the two predecessors is always prioritized in situations where both want to transport a cargo.
The variable \( shift \) controls whether the table is required to move (cf. Table 1).

The state variable \( req \) indicates whether the turntable is currently handling a request or not (cf. Table 2).

The chosen predecessor, from which a cargo is loaded, is stored in the state variable \( orig \) (cf. Table 5a on the next page). Note that there are two \( Take \) as well as two \( Clear \) variables necessary in order to communicate with two preceding MHS elements. The prioritization of one predecessor over the other is implemented by adapting the \( Take \) variable of the turntable (cf. Table 5a on the next page).

<table>
<thead>
<tr>
<th>( Stop )</th>
<th>( Pre )</th>
<th>( Take )</th>
<th>( Give )</th>
<th>( Suc )</th>
<th>( shift )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \text{Table 1. Turntable control: } shift \)

The variable \( \text{inp}_{\text{en}} \) (cf. Table 3) indicates that the turntable cargo path whether a new cargo is loaded from the preceding MHS element or not. A new cargo is loaded only if the turntable moves \((shift = 1)\), the predecessor wants to transfer a cargo \((Pre = 1)\), and the turntable does not currently handle a request \((Take = 1)\).

<table>
<thead>
<tr>
<th>( \text{Stop} )</th>
<th>( \text{d} )</th>
<th>( \text{opos} )</th>
<th>( \text{Pre} )</th>
<th>( \text{Ack} )</th>
<th>( \text{req} )</th>
<th>( \text{oreq} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \text{req}(0) = 0 \)

\( \text{Table 2. Turntable control: } req \)

Utilizing the \( Error \) variable (cf. Table 4a), a malfunction in the MHS model can be indicated, e.g. if the turntable receives an acknowledgment without having a request.

The turntable can only transfer a cargo to the succeeding MHS element (cf. Table 4b) if there is a cargo on the table \((d = 1)\), and the table is in the transferring position \((pos = 1)\). The receipt of a cargo is acknowledged once it has been detected by the sensor \( d \) which is defined as follows:

\[
d = \begin{cases} 1 & \text{if } c_1 \neq L \\ 0 & \text{else} \end{cases}
\]

The prioritization of one predecessor over the other is implemented by adapting the \( Take \) variable of the turntable (cf. Table 5a on the next page).

<table>
<thead>
<tr>
<th>( \text{Stop} )</th>
<th>( \text{o} )</th>
<th>( \text{pos} )</th>
<th>( \text{d} )</th>
<th>( \text{o} )</th>
<th>( \text{orig} )</th>
<th>( \text{Clear}_1 )</th>
<th>( \text{Clear}_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

By considering the origin of a received cargo \((orig)\), it is ensured that \( Clear_1 \) and \( Clear_2 \) will only be sent to the corresponding predecessor rather than to both at the same time (cf. Table 6).

The actual loading, storing, and transporting of a cargo are modeled in the cargo path of the turntable (cf. Table 7).

<table>
<thead>
<tr>
<th>( \text{IN} )</th>
<th>( c_1 )</th>
<th>( shift )</th>
<th>( \text{inp}_{\text{en}} )</th>
<th>( c_1 )</th>
<th>( \text{OUT} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( -v )</td>
<td>( \neg v )</td>
<td>( 0 )</td>
<td>( -v )</td>
<td>( v )</td>
<td></td>
</tr>
<tr>
<td>( u )</td>
<td>( v )</td>
<td>( 1 )</td>
<td>( u )</td>
<td>( v )</td>
<td></td>
</tr>
</tbody>
</table>

\( c_1(0) = L, u, v \in C \)

\( \text{Table 7. Turntable: cargo path behavior} \)

The state variable \( pos \) (cf. Table 8) models the current position of the turntable, where \( pos = 0 \) describes that the
The time may vary because of the picker is “chosen” non-deterministically whenever there are no articles on the cargo anymore, and thus, the cargo has to be adjusted:

- connecting and identifying the variables, the cargo path distinguishes between different MHS elements and/or MHS subnetworks which, in turn, are interconnected in an arbitrary order to a network that models the technical MHS. The resulting network itself can again be regarded as an MHS model. That means, an MHS model is a network of MHS elements.

In the presented model, it is assumed that the cargo type of a cargo changes whenever the picker has processed the cargo. The cargo type can e.g. represent the next destination of the cargo or the information whether there are no articles on the cargo anymore, and thus, the cargo has to be conveyed to an exit. \( \mathbb{T} \) defines the cargo types that may depart from the picker. Whenever a cargo is loaded (cf. Table 9), the type will be changed immediately to one of the types given in \( \mathbb{T} \).

To model the different amounts of time that are necessary to handle one order, the length \( l \) of the cargo path of the picker is “chosen” non-deterministically whenever a cargo is loaded (cf. Table 10). The time may vary between 1 and \( n \) time units.

Consequently, the sensor signals \( d_1 \) (for acknowledging the receipt of a cargo to the predecessor) and \( d_2 \) (position before the cargo is transferred to the successor) also have to be adjusted:

\[
\begin{align*}
Take_1 & \quad Take_2 & \quad orig & \quad orig' & \quad oorig \\
0 & 0 & 0 & b & b \\
0 & 1 & b & 1 & b \\
1 & 0 & b & 0 & b \\
1 & 1 & b & 0 & b \\
\end{align*}
\]

\( orig(0) = 0, b \in \mathbb{B} \)

\( (b) \ orig \)

\( m_{sel} \quad IN_1 \quad IN_2 \quad IN \)

\( 0 \quad u \quad \_ \quad u \\
1 \quad \_ \quad u \quad u \\
\)

\( (c) \ IN \)

Table 5. Priority merge control: \( Take_1, Take_2, \) \( orig \) and \( IN \)

<table>
<thead>
<tr>
<th>Stop</th>
<th>d</th>
<th>opos</th>
<th>Ack</th>
<th>oreq</th>
<th>Pre (_1)</th>
<th>Pre (_2)</th>
<th>Take (_1)</th>
<th>Take (_2)</th>
<th>( m_{sel} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) \( Take_1 \) and \( Take_2 \)

Table 6. Priority merge control: \( Take_1, Take_2, \) \( orig \) and \( IN \)

<table>
<thead>
<tr>
<th>( m_{sel} )</th>
<th>( IN_1 )</th>
<th>( IN_2 )</th>
<th>( IN )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( u )</td>
<td>( _ )</td>
<td>( u )</td>
</tr>
<tr>
<td>1</td>
<td>( _ )</td>
<td>( u )</td>
<td>( u )</td>
</tr>
</tbody>
</table>

\( (c) \ IN \)

Table 7. Priority merge control: \( Take_1, Take_2, \) \( orig \) and \( IN \)

\( \text{Table 8. Priority merge control: } Take_1, Take_2, \) \( orig \) and \( IN \)

Table 9. Picker: cargo path behavior

\[
\begin{align*}
IN & | c_1 & shift & inp\_en & pos & d_2 & c' & OUT & pos' \\
- & v & 0 & - & m & - & t & v & 1 \\
v & v & 1 & 1 & - & - & v & v & 0 \\
v & v & 1 & 0 & k & 0 & v & v & k+1 \\
v & v & 1 & 0 & - & 1 & v & v & 0 \\
\end{align*}
\]

\( pos(0) = 0; c_1(0) = L; u, v \in \mathbb{C}; t \in \mathbb{T}; 1 \leq k \leq L \)

\( m, k \in \mathbb{N}, 0 \leq m \leq n, 1 \leq k \leq n \)

Table 9. Picker: cargo path behavior

\( \text{Table 10. Picker: } l \)

\[
d_1 = \begin{cases} 
1 & \text{if } pos = 1 \\
0 & \text{else}
\end{cases} \\
d_2 = \begin{cases} 
1 & \text{if } pos = ol \\
0 & \text{else}
\end{cases}
\]

A corresponding example will be shown in Section 5.2.

4. Interconnection of MHS elements

MHS elements can be interconnected in an arbitrary order to a network that models the technical MHS. The resulting network itself can again be regarded as an MHS model. That means, an MHS model is a network of MHS elements and/or MHS subnetworks which, in turn, are such interconnections.

Fig. 3 shows an example of a network consisting of a series connection of three MHS elements. The subsequent MHS elements \( i - 1, i, \) and \( i + 1 \) are linked by connecting and identifying the variables \( \text{OUT}_{i-1} \rightarrow \text{IN}_1, \text{Give}_{i-1} \rightarrow \text{Pre}_{i}, \text{OUT}_{i} \rightarrow \text{IN}_{i+1}, \text{Give}_{i} \rightarrow \text{Pre}_{i+1}, \text{Clear}_{i} \rightarrow \text{Ack}_{i-1}, \) and \( \text{Take}_{i} \rightarrow \text{Suc}_{i-1}. \) The variables \( \text{Stop} \) and \( \text{Error} \) have to be connected with regard to the technical MHS, e.g. to model a global stop function or a comprehensive error detection.
5. Experimental results

In this section, two examples of MHSs are introduced and discussed in the sequel. The MHSs are modeled using interconnected MHS elements (cf. Section 3). With this, a number of properties are proven on the models. Therefore, the open-source model checker NuSMV [4] has been utilized. The properties are specified in CTL [3]. The properties have been checked using NuSMV 2.5.2 running on a desktop computer with 8GB RAM. The actual lengths of the conveyors are given in the figures.

5.1. Example 1

A typical example of an MHS is depicted in Fig. 5. Here, the incoming cargos from src enter a loop via the priority merge pm1, which prioritizes cargos from con6 over those from con1 in order to ensure that the cargos in the loop keep moving. In case con7 (con8) is ready to receive a cargo, all incoming cargos of type g (r) are routed at div1 (div2) to OUT1 (OUT2), respectively. All the cargos that could not be transported to one of the sinks in first place move on in the loop. The set of cargo types is given as $C = \{g, r, L\}$.

At first, the correct signaling between the MHS elements can be checked. Therefore, the Error signals of all of the MHS elements of the example are or-ed connected. Then, with the property

$$\text{AG} (\text{Error}=0)$$

it is proven that none of the Error signals ever becomes 1.

Next, the correct routing of the MHS system is analyzed (here exemplarily shown for OUT1):

$$\text{AG}(\text{OUT1}=g \mid \text{OUT1}=L)$$

This property states that at any time at OUT1 there can only be a cargo of type g or no cargo (type L), and consequently, no cargo of type r will ever reach OUT1.

A typical problem that may appear in MHS containing loops are deadlocks. By considering the property

$$\text{AG}(\text{con6.OUT}=L \rightarrow \text{AF}(\text{con6.OUT}=L))$$

it can be shown that there is no deadlock in the MHS system, i.e. all the cargos in the loop will be conveyed at some point in time. The property expresses that if there is a cargo at con6, then con6 will be empty eventually. However, the fairness constraint Stop = 0 has to be added to the model in order to successfully proof this property. Otherwise, the model checker presents a counterexample where the Stop signal is constantly 1.

5.2. Example 2

The second example is depicted in Fig. 6. Here, cargos of type s are provided from the cargo source src1 and, thereafter, enter a loop. The priority merge elements pm1, pm2 and pm3 ensure that all the cargos in the loop are prioritized.

All cargos of type s are conveyed to the high rise storage (HRS) element hrs. The model of the HRS abstracts from a technical HRS in such a way that the actual storing and retrieving of cargos from and to certain position in the storage are not considered. It is assumed that the HRS itself works correctly, and hence, the storing is modeled using the conveyor con7 and the sink snk2, whereas the retrieving is modeled as a source (src2) connected to a conveyor (con8).

The cargos from the HRS are of the type g, i.e. they have to be routed to the picker pick via div2. There, the cargos are processed to satisfy a given order. The MHS element div2 represents a turntable with divert function, but with an additional timer function: If there is a cargo of type g at div2 and the picker cannot handle the cargo (because it is currently working on another request), the cargo will wait up to three time units at div2, before it will be conveyed to con4. Afterwards, after one to four time steps, the cargos are transferred back from pick into
the loop via pm3. The resulting types s or b of the cargos mark whether the cargos have to be routed to the exit OUT or back to hrs.

Based on the given model, a number of properties are verified using model checking. First, it is proven whether all the cargos that leave from the picker are either of type b or s, so that they are routed to their correct destinations. Additionally, the case that the picker does not handle a cargo (pick.OUT = L) has to be included:

$$AG(pick.OUT=L \mid pick.OUT=s \mid pick.OUT=b)$$

After proving that the cargos are handled correctly at the picker, it is analyzed if only the cargos of type b, and thus, no other cargos, will ever reach OUT:

$$AG(OUT = b \mid OUT = L)$$

Then, the correct implementation of the waiting function at div2 is verified. With the following property, it is checked whether it can never happen that there is a cargo of type g placed at div2 for four (instead of the intended three) subsequent time steps:

$$AG(OUT = g \mid OUT = L)$$
AG !( (Stop=0 & div2.OUT=g) & EX (Stop=0 & div2.OUT=g)) & EX (Stop=0 & div2.OUT=g) & EX (Stop=0 & div2.OUT=g))

Note that the Stop signal has to be set to 0 in the property because otherwise a cargo stays at div2 as long as stop is active. Next, the routing is analyzed more in detail. By considering the properties

AG! (con3.OUT=g & div2.OUT=g & pick.Take=0)

and

AG! (pm2.OUT=g & con3.OUT=g & div2.OUT=g & pick.Take=0)

it can be checked whether cargos that are supposed to be handled by the picker may have to wait at the elements con3 and div2, or even at pm2, con3, and div2 at some point in time. The first property is false, and the second one is true, i.e., it may happen that cargos of type g have to wait at con3 and div2 but not at pm2, con3, and div2 simultaneously.

Since the latter can lead to blocking cargos in the loop, a global routing rule is added to the model: In case there are cargos of type g at con3 and div2 at a time, then the cargo at div2 does not have to wait up to three time steps at div2 but will be conveyed immediately to con4. As a result, the throughput of the overall MHS model is increased.

The correct realization of the routing rule is proven by the following property:

AG ( (con3.OUT=g & div2.OUT=g & pick.Take=0 & con4.Take=1) -> AX (con4.OUT=g))

The property expresses that if there are cargos of type g at con3 and div2, and the picker cannot handle them (pick.Take = 0), but con4 could do so (con4.Take = 1), then the cargo from div2 will be at con4 in the subsequent time step.

6. Conclusion and outlook

In this paper, an approach to the formal verification of MHS models has been presented. The approach is based on the modeling methodology of MHS elements that are proven to be sound. The functional behavior of a number of MHS elements has been explained. The feasibility of the proposed approach has been shown using examples. The examples demonstrate that by applying formal methods, complex requirements from the specification of the MHS can be automatically checked.

The addition of global routing strategies has been discussed in a case study. In ongoing research, a more general form of specifying global routing strategies has to be developed. This may include an abstract model of the material flow computer and its program to allow complex routing decisions and processes that can be proven.

In future work, the routing as well as the layout of the MHS will be automatically extracted from a given graphical representation, e.g., from a commercial MHS simulator. With this, the required MHS elements can be automatically selected from the list of modeled MHS elements (“model library”), interconnected according to the technical MHS, and then fed to a model checker for verification. As a result, the designer of the logistic system does not have to manually deal with the internals of the MHS elements.

As a long-term goal, the verified MHS model can serve as a basis for code generation.

References