Flexible Assembly Line: ARGESIM
Comparison 2

S.M. Riezebos
Flexible Assembly Line

ARGESIM Comparison 2
Bachelor Final Project
SE: ......

S.M. Riezebos

Supervisor: Dr. Ir. D.A. van Beek
Advisor: ir. L. Swartjes

Eindhoven, June 11, 2013
Abstract

In this report a benchmark for simulations systems is done. It is Comparison 2 of the ARGESIM group simulation benchmarks. A flexible assembly line is modeled and simulated using CIF 3 [3].
The assembly line is a system of conveyor belts where pallets are circling through the system until 4 assembly processes have been completed. The number of pallets is the main input variable.
The tasks that are defined by the ARGESIM group consist of modeling the assembly line and finding the throughput and flowtime with variable numbers of pallets. Finally an optimal number of pallets needs to be found.

The model is made using CIF 3 and it includes a visualization of the simulated assembly line. The simulation gives reliable results for an input between 1 and 30 pallets. Above 30 the flowtime results become unstable, above 53 the model goes into deadlock due to full conveyor belts.
The calculated optimal number of pallets is 13. This number results in a throughput of 180 pallets/h and an average flowtime of 260 s.
Contents

1 Introduction 2
  1.1 ARGESIM Comparison .................................. 2
  1.2 Assignment tasks and criteria ........................... 3

2 Modeling 5
  2.1 Assumptions ........................................... 5
  2.2 General criteria ........................................ 5
  2.3 Control Strategy ....................................... 5
  2.4 Model details .......................................... 6
  2.5 Visualization .......................................... 8

3 Simulation results 10

4 Validation 12
  4.1 Model validation ....................................... 12
  4.2 Comparing to other ARGESIM entries ..................... 12

5 Conclusion and Recommendations 15
  5.1 Conclusion ............................................. 15
  5.2 Recommendations ...................................... 15

A Table of results 18

B CIF 3 Model 19
Chapter 1

Introduction

1.1 ARGESIM Comparison 2

This report describes the modeling, simulation and verification of ARGESIM Comparison 2 using version 3 of the Compositional Interchange Format (CIF 3). The Comparison 2 (C2) case [1] is one of 19 benchmarks for simulation systems. These benchmarks have been developed by the ARGESIM Group. The subject of C2 is a flexible assembly system. The main features that are tested are:

- The possibility to define and combine submodels
- The method to describe complex control strategies

The assembly line consists out of 8 similar subsystems that are visualized in Figure 1.1.1:

Each subsystem has two switches, two conveyor belts and one workstation. A pallet enters the subsystem at Sx. When all criteria to move the pallet to the workstation are met it will move up to B2 and Ax, otherwise it will move to B1. This can be seen in Figure 1.1.2: Apart from the given specifications there are some other rules that apply to the system:

- Pallets that are moved back from a workstation through Sy have priority over pallets that are arriving from conveyor belt B1
- Pallets that exit the third subsystem A2 and A6 can move directly to the next subsystem. So where the normal space between subsystems is 0.4 m these spaces are modeled as 0.0 m
The switches are not allowed to function as buffers. If a pallet enters a switch Sx or Sy it must be processed immediately.

Operation A6 is a substitute for operation A3, A4 or A5. It can do one of the three operations.

Operation A2 can only occur when either A3, A4 and A5 are all completed or they all are yet to be completed.

Workstation A1 unloads parts from pallets that have completed all necessary processes and loads new unprocessed parts on these pallets.

The conveyor belt movement speed is 0.3 m/s

Any moving up or down in Sx or Sy takes 2 s

Pallets have a size of 0.36 m

Every machine Ax has a deterministic operation time, these times are shown in Table 1.1.1.

<table>
<thead>
<tr>
<th>Station</th>
<th>Operation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>15</td>
</tr>
<tr>
<td>A2</td>
<td>60</td>
</tr>
<tr>
<td>A3</td>
<td>20</td>
</tr>
<tr>
<td>A4</td>
<td>20</td>
</tr>
<tr>
<td>A5</td>
<td>20</td>
</tr>
<tr>
<td>A6</td>
<td>30</td>
</tr>
</tbody>
</table>

1.2 Assignment tasks and criteria

The ARGESIM comparison definition states a few general criteria and tasks that should be performed. The tasks are described as follows:

- Task a: Method of control strategy description and method of statistical evaluation
• Task b: Simulation results: total throughput and average throughput time

• Task c: Simulation results: optimization of the number of pallets with respect to maximal throughput and minimal throughput times

For each task there are criteria that should be evaluated. The model description and structure need to be evaluated in general. For Task a the control strategy type must be chosen. For Task b the variation of the number of pallets in the system is prescribed. For Task c the optimization method must be described and the target must be chosen. Either the optimization should be done for throughput first and then for flowtime, or as a compromise between both. Furthermore, the simulated time should always be 10 hours, the throughput and throughput time should be measured only for the last 8 hours of the simulated time.
Chapter 2

Modeling

In this chapter the creation of the model is evaluated, it will include the assumptions and criteria that are used to make the model, the control strategy, as well as a description of the most important details of the model. Furthermore the visualization is explained.

2.1 Assumptions

In the project definition most information about the flexible assembly system is clear. There are a few parts that are open to interpretation or not completely defined. Therefore the following assumptions are made:

- After being processed in a workstation a pallet directly transfers back to conveyor belt B1 through Sy. Switch Sy is positioned directly at the exit of workstation Ax.
- It is possible to temporarily hold a pallet before in front of Sy in case another pallet is being transferred from conveyor belt B2, which has priority by definition.
- All conveyor belts are divided into parts of 0.4 m, these parts are assumed to hold exactly one pallet. The 4 cm difference is assumed to be a safety margin.

2.2 General criteria

As general criteria for the case the model type and structure should be evaluated. The model type is textual and graphical. Using CIF3 the model can be simulated using SVG visualization, while the textual results can be presented simultaneously. In the model there are submodels for different parts of the system, and functions are used to determine the outcome of operations.

2.3 Control Strategy

The control strategy is partially predefined by the definition of the case. Workstation A1 is defined as both generator and exit process, which implements a pull strategy. The criteria to move a pallet up to a workstation are also predefined. Furthermore some of the assignment criteria apply to the control strategy. A pallet that has finished a machine operation has
priority over a pallet arriving from a belt B1, this is controlled by checking whether a pallet will be finishing a machine operation within the needed movement time to go through a switch. If this is the case the pallet on B1 is temporarily stopped until the movement from the machine has been finished.

Apart from this, the control strategy of the conveyor belts and machines is predetermined. All belts receive pallets, then they move across the belt and are sent to the next part. All machines receive pallets, keep them until the operation is finished and send them to the next switch. The only exception is machine A6. This machine can do one out of three possible operations. It can replace either A3, A4 or A5. The current control strategy checks whether operation A3 is necessary and acts accordingly. If A3 has already been completed it will do A4, if necessary, and otherwise A5. This order of operations should not influence the average flowtime substantially, because the shortage of each of these 3 operations will equalize when another one is completed in a lot of the other pallets.

The only part where an advanced control strategy could be used is the switch. This could be done globally, for example by keeping track of the status of all pallets and having them go through the workstations as efficiently as possible. This would require extensive knowledge of control strategies and CIF 3 programming.

Currently the control strategy is mostly local. Most switches receive a pallet, check whether the status of the pallet matches the criteria to be moved up to the next machine and then try to send it up to the buffer belt. If it is full the pallet will be send on to the bottom belt. There are two exceptions to this. The switches in front of the first and second A2 machines have a control strategy that is not local. When the machine criteria are met these switches also check the bottom belt, switch and buffer belt for the next machine. If these do not contain pallets that also meet the criteria it will send the current pallet to the next machine to distribute the load across the three A2 machines. This improves throughput time for simulations with a low number of pallets.

### 2.4 Model details

The CIF3 model (Appendix B) is programmed using the Eclipse software. This software package is also used for visualization and programming the link between the model and the visualization. The model can be divided into three parts. The submodel definitions, the instantiations, and the functions. The submodels contain a definition for a normal machine and switch, as well as a separate definition for switch 2-1 and 2-2 and machine A1. There is a definition for all four possible belt lengths. Multiple versions of these submodels are instantiated. They all have at least one channel for receiving pallets and one for sending them. The type ”pallet” is defined as a tuple with an integer (id number), a real number (generation time) and four integers for the machine operations (0 for operations to be done and 1 for finished operations), booleans where used for the operations at first but these were replaced with integers because those were better to link to the visualization at the time. The last part are the functions, these are used in the submodels to determine the outcome of certain events.

In the model it would be logical to make the movements across all conveyor belts continuous, by doing this pallets would move through the system with a constant velocity. The problem with this approach is that it works good for a constant number of pallets, but it would take a lot of reprogramming every time the number of pallets is changed. Therefore the belts are
modeled with another approach:
The model can be seen as a collection of spaces for pallets, each machine and switch contains one space and each belt two to five spaces. When the model is running, pallets move from space to space, each movement is constrained by control strategy and movement times. The pallets start at A1, move through the system until all operations are completed and then return to A1 where they are reinitialized. Because of these discrete spaces the model looks somewhat different than the system would look in reality.

```python
def Machine(alg int atype; alg real ct; event receive; alg pallet pl):
    event send;
    disc pallet im = leeg;
    cont t = 0.0;

    location wait:
        initial;
        equation t' = 0.0;
        event receive? when im = leeg do im := operate(pl, atype), t := ot goto operation;

    location operation:
        equation t' = -1.0;
        event send! when im != leeg and t <= 0.0 do im := leeg goto wait;
    end

Figure 2.4.1 – The CIF 3 code for a machine
```

The middle part of the model contains the submodels. The part of the model that defines a machine can be seen in Figure 2.4.1. The machine basically waits until it can receive a pallet, then sets the operation time and waits until it can send the pallet away again. The function operate is used to change the correct integer in the pallet from 0 to 1. A machine can only receive when the space for a pallet (im) is empty.

The conveyor belts are similar, but they have multiple spaces and the there is a set movement time for every space instead of an operation time. The events in the conveyor belts run continuously, therefore it is possible that for example a pallet moves from space 1 to 2 while at the same time a pallet stays in space 3, the movement time is set when the pallet enters the belt. An alternative would be to force pallets to move simultaneously, this would make the movement seem more realistic. The reason this approach is not used is that it could extend flowtime: sometimes a pallet that is ready to move, would have to wait for the movement time of another pallet to finish. The conveyor belts cover a large part of the model.

Another important part is the switch as shown in Figure 2.4.2. There are two channels for receiving and two for sending. receive1 receives from the previous machine, it adds 2 s of movement time for switching. receive2 receives from the previous belt B1 unless receive1 will receive within 4/3 s. send1 sends to the buffer belt before the next machine when criteria are met. If send1 is unable to send in time the pallet will be send through send2. When criteria are not met the pallet will also be sent through send2. The function shortcheck is used because switch 3 and 6 have no movement time.

The last part of the model are the functions. Function shortcheck is used because the movement time for switch 3 and 6 is 0 s. Function crit checks whether a pallet on a switch meets the criteria for the next machine. Function operate changes the pallet integer corresponding to the machine type and function bufcheck is used in the switches in front of A2-1 and A2-2 to check the upcoming spaces for pallets where operation A2 is still to be done.
2.5 Visualization

The visualization during simulation can be seen in Figure 2.5.1. The image is made with Inkscape software. Some properties of the image are linked to variables of the model with a cifsvg file using Eclipse.

Every pallet space in the model is visualized by a cube, containing the status of the pallet as shown in Figure 2.5.2. The number is the ID number of the pallet and the colored blocks are the status of operations A2, A3, A4 and A5. A block is green, if the corresponding operation is finished, and red if it is not. An empty space has ID number 0 and only red blocks.

Above the visualized system the current system time and the processed pallet count are stated. The flowtime and throughput are calculated after the time reaches 7200 s, as stated by the assignment. A machine contains a pallet space as well as the amount of time until operation is complete. All variables in the visualization directly correspond to a variable in the model.

If continuous movements would have been used in the model it would also have been possible...
to use continuous movements in the visualization. But modeling this approach is different in a lot of ways. It would probably make the visualization seem much more realistic. Nevertheless, the results of the currently used model and visualization give a very close approximation of the alternative.
Chapter 3

Simulation results

This chapter describes the results of simulation. The simulations are done, as prescribed by task b and task c. For each pallet number a simulation of 10 hours is done, the last 8 hours are used to calculate the results. During simulation, it is possible that multiple transitions are possible simultaneously. To make sure that the results are replicable, the simulation is set to choose the first transition. An overview of the results is shown in Table 3.0.1, the full results of simulation can be found in Appendix A.

<table>
<thead>
<tr>
<th>WIP</th>
<th>Flowtime</th>
<th>8 hour throughput</th>
<th>Computer time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>266,7</td>
<td>1080,2</td>
<td>00:38:00</td>
</tr>
<tr>
<td>11</td>
<td>264,0</td>
<td>1200,7</td>
<td>00:39:00</td>
</tr>
<tr>
<td>12</td>
<td>261,8</td>
<td>1320,9</td>
<td>00:41:00</td>
</tr>
<tr>
<td>13</td>
<td>260,0</td>
<td>1440,2</td>
<td>00:43:00</td>
</tr>
<tr>
<td>14</td>
<td>280,0</td>
<td>1440,2</td>
<td>00:45:00</td>
</tr>
<tr>
<td>15</td>
<td>300,0</td>
<td>1441,0</td>
<td>00:44:00</td>
</tr>
<tr>
<td>16</td>
<td>300,0</td>
<td>1440,9</td>
<td>00:53:00</td>
</tr>
<tr>
<td>17</td>
<td>320,0</td>
<td>1440,6</td>
<td>00:54:00</td>
</tr>
<tr>
<td>18</td>
<td>300,0</td>
<td>1440,9</td>
<td>01:03:00</td>
</tr>
<tr>
<td>19</td>
<td>300,0</td>
<td>1440,9</td>
<td>01:08:00</td>
</tr>
<tr>
<td>20</td>
<td>400,0</td>
<td>1440,4</td>
<td>01:11:00</td>
</tr>
<tr>
<td>25</td>
<td>500,5</td>
<td>1441,1</td>
<td>01:53:00</td>
</tr>
<tr>
<td>30</td>
<td>600,3</td>
<td>1442,3</td>
<td>02:18:00</td>
</tr>
<tr>
<td>35</td>
<td>525,2</td>
<td>1441,6</td>
<td>02:36:00</td>
</tr>
<tr>
<td>40</td>
<td>720,0</td>
<td>1440,1</td>
<td>02:46:00</td>
</tr>
<tr>
<td>45</td>
<td>897,9</td>
<td>1442,0</td>
<td>00:00:00</td>
</tr>
<tr>
<td>50</td>
<td>999,0</td>
<td>1441,8</td>
<td>00:00:00</td>
</tr>
</tbody>
</table>

Task b is to calculate flowtime and throughput from 20 to 60 pallets. The model is unable to calculate these above 53 pallets, the cause of this is that at some point all switches and bottom belts are full, this causes the system to lock. The throughput is constant for all values between 20 and 60. This is because above 12 pallets machine A2 will become a bottleneck. This is visualized in Figure 3.0.1. Between 10 and 13
the throughput follows a linear path.

![Figure 3.0.1 – Throughput vs. amount of pallets](image)

![Figure 3.0.2 – Flowtime vs. amount of pallets](image)

For the flowtime, most simulations with more than 12 pallets are linear. This is not surprising because above 12 the workstations A2 run at full capacity. Any pallets that are added above this number will only result in more flowtime. Above 30 pallets a lot of simulations stop following the expected linear path. This is probably an error in the model. The cause of this is error has not been found. All simulations and visualizations seem exactly the same as the ones that give correct results.
Chapter 4

Validation

This chapter shows the validation of the model. First the model itself is considered, then the results are compared to those of other ARGESIM entries, these results should not be very different.

4.1 Model validation

In order to validate the model a few tests are done. First the model is simulated with only 1 pallet, this works and gives low throughput (19 pallets/h) and low flowtime (195 s). The upper limit of the model is also found, this is 53 pallets. Above 53, all switches and lower belts are full at a certain point in the simulation. When this happens a deadlock occurs as all belt spaces are unable to receive a new pallet. The spaces need to be emptied before they can receive a new pallet.

Another important point is that there are flowtime results lower than expected. These only occur for specific pallet numbers while both above and below the strange results there are still normal results. Most of the errors occur between 30 and 50 pallets. The fact that it occurs mostly with high pallet numbers might suggest that the model is somehow instable if such a high number of events occur simultaneously.

The last important point for validation is the visualization, this is a very useful tool to check whether the model works correctly. During the process of making the model a lot of errors have been found using the visualization. In the end no more visual errors have been found.

4.2 Comparing to other ARGESIM entries

There are a lot of other entries that have modeled and simulated ARGESIM comparison 2. The results have been published on the ARGESIM website [2]. When comparing these results to the results for CIF 3 Figure 4.2.1 and Figure 4.2.1 are made. The results are all quite similar, there are some differences between all results but these can be explained. The assumptions that are made to complement the case definition can be different. Furthermore every participant uses a control strategy. A different strategy can also lead to different results. The throughput gives results that are similar to some of the other entries. This indicates that the CIF 3 model gives correct results. Furthermore this model is not the only model without results for high pallet numbers. Other models also seem to have trouble with simulations for
high pallet number.

Figure 4.2.1 – Throughput vs. amount of pallets
The results regarding flowtime are partially very similar to the others. Again the results above 30 seem to have a lot of errors. The other models do not seem to have these errors. It is also possible that the other models did not find any errors because they only have results for every 5-10 pallet numbers. For CIF 3, errors would not be visible if only those results are considered.

Using all these results it can be concluded that for pallet numbers below 30 the model is correct.
Chapter 5

Conclusion and Recommendations

5.1 Conclusion

The conclusion can be split into two categories. One part about this specific model and results and one part about the performance and capabilities of CIF 3. The model itself gives good and valid results for throughput. Especially when comparing the results to other ARGESIM entries the model looks valid. For the results regarding flowtime this is a different story. There are errors for high pallet numbers which indicate that something is not working correctly. This part of the model probably contains an error. The visual part of the model gives a clear image of what is happening during a simulation. It currently works with a sufficient speed to be able to give a quick overview of how this system works. Therefore this part seems to be correct as well.

The other part is a general conclusion about CIF 3. The language is capable to be used for this comparison. There is a lot of functionality that can be used perfectly to accomplish the goals of this benchmark. The use of events and locations makes it easy to define all processes that should be running simultaneously. The definition of submodels with different channels and variables causes the final model to be very compact. Being able to refer variables from other submodels is also very useful, especially when making a more complex control strategy. The visualization is a useful tool in many aspects, mainly presenting a model in a form that is quickly comprehensible. It is also useful during development because often the location of an error is recognizable from the visualization. CIF 3 was in development during this project, this caused a few issues during the development of the model. Mostly these consisted out of workarounds as some parts of CIF 3 were not fully developed yet. In the end almost everything was working.

5.2 Recommendations

The current model uses discrete places to simulate all conveyor belts. It would be more realistic to use a continuous model. This could also prevent the inability to run above 50 pallets. If this is also implemented in the visualization, this would make it much clearer how the system works.
The control strategy is highly local in the current system, this can cause pallets to circle the system unnecessarily. If a global control strategy were to be implemented this can be prevented by checking the status of upcoming pallets and machines for the whole system and then choosing the most efficient routes for the pallets.
Bibliography


### Appendix A

#### Table of results

<table>
<thead>
<tr>
<th>WIP</th>
<th>Flowtime</th>
<th>Throughput</th>
<th>WIP</th>
<th>Flowtime</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>266.7</td>
<td>135.0</td>
<td>32</td>
<td>593.0</td>
<td>180.2</td>
</tr>
<tr>
<td>11</td>
<td>264.0</td>
<td>150.1</td>
<td>33</td>
<td>652.9</td>
<td>179.8</td>
</tr>
<tr>
<td>12</td>
<td>261.8</td>
<td>165.1</td>
<td>34</td>
<td>620.0</td>
<td>180.2</td>
</tr>
<tr>
<td>13</td>
<td>260.0</td>
<td>180.0</td>
<td>35</td>
<td>525.2</td>
<td>180.2</td>
</tr>
<tr>
<td>14</td>
<td>280.0</td>
<td>180.0</td>
<td>36</td>
<td>609.5</td>
<td>180.1</td>
</tr>
<tr>
<td>15</td>
<td>300.0</td>
<td>180.1</td>
<td>37</td>
<td>564.1</td>
<td>180.2</td>
</tr>
<tr>
<td>16</td>
<td>300.0</td>
<td>180.1</td>
<td>38</td>
<td>619.8</td>
<td>180.1</td>
</tr>
<tr>
<td>17</td>
<td>320.0</td>
<td>180.1</td>
<td>39</td>
<td>586.1</td>
<td>180.4</td>
</tr>
<tr>
<td>18</td>
<td>300.0</td>
<td>180.1</td>
<td>40</td>
<td>720.0</td>
<td>180.0</td>
</tr>
<tr>
<td>19</td>
<td>300.0</td>
<td>180.1</td>
<td>41</td>
<td>724.3</td>
<td>180.1</td>
</tr>
<tr>
<td>20</td>
<td>400.0</td>
<td>180.1</td>
<td>42</td>
<td>485.1</td>
<td>180.0</td>
</tr>
<tr>
<td>21</td>
<td>419.4</td>
<td>180.1</td>
<td>43</td>
<td>862.0</td>
<td>180.0</td>
</tr>
<tr>
<td>22</td>
<td>440.4</td>
<td>180.0</td>
<td>44</td>
<td>720.0</td>
<td>180.2</td>
</tr>
<tr>
<td>23</td>
<td>460.6</td>
<td>180.3</td>
<td>45</td>
<td>897.9</td>
<td>180.3</td>
</tr>
<tr>
<td>24</td>
<td>480.0</td>
<td>180.1</td>
<td>46</td>
<td>719.2</td>
<td>180.0</td>
</tr>
<tr>
<td>25</td>
<td>500.5</td>
<td>180.1</td>
<td>47</td>
<td>834.7</td>
<td>180.2</td>
</tr>
<tr>
<td>26</td>
<td>519.5</td>
<td>180.0</td>
<td>48</td>
<td>740.0</td>
<td>180.1</td>
</tr>
<tr>
<td>27</td>
<td>539.8</td>
<td>180.3</td>
<td>49</td>
<td>761.2</td>
<td>180.3</td>
</tr>
<tr>
<td>28</td>
<td>483.5</td>
<td>179.9</td>
<td>50</td>
<td>999.0</td>
<td>180.2</td>
</tr>
<tr>
<td>29</td>
<td>577.1</td>
<td>179.8</td>
<td>51</td>
<td>1.020.5</td>
<td>180.2</td>
</tr>
<tr>
<td>30</td>
<td>600.3</td>
<td>180.3</td>
<td>52</td>
<td>1.014.2</td>
<td>179.8</td>
</tr>
<tr>
<td>31</td>
<td>620.0</td>
<td>180.2</td>
<td>53</td>
<td>1.058.9</td>
<td>180.1</td>
</tr>
</tbody>
</table>
Appendix B

CIF 3 Model

1 // Definition of workstations
2 a1: A1(20, b1.send, b1.buf1);
3 a21: Machine(2, 60.0, b21.send, b21.buf1);
4 a22: Machine(2, 60.0, b22.send, b22.buf1);
5 a23: Machine(2, 60.0, b23.send, b23.buf1);
6 a3: Machine(3, 20.0, b3.send, b3.buf1);
7 a4: Machine(4, 20.0, b4.send, b4.buf1);
8 a5: Machine(5, 20.0, b5.send, b5.buf1);
9 a6: Machine(6, 30.0, b6.send, b6.buf1);

10 // Definition of switches
11 s1: Switch(1, a6.t, a6.send, belt6.send, a6.im, belt6.buf1);
12 s21: Switch2(2, a1.ot, a1.send, belt1.send, a1.ia, belt1.buf1,
13   belt21.buf4, belt21.buf3, belt21.buf2, belt21.buf1,
14   s22.is, b22.buf2, b22.buf1);
15 s22: Switch2(2, a21.t, a21.send, belt21.send, a21.im,
16   belt21.buf1, belt21.buf4, belt21.buf3, belt21.buf2,
17   belt22.buf1, s23.is, b23.buf2, b23.buf1);
18 s23: Switch(2, a22.t, a22.send, belt22.send, a22.im, belt22.buf1);
19 s3: Switch(3, a23.t, a23.send, belt23.send, a23.im, belt23.buf1);
20 s4: Switch(4, a3.t, a3.send, belt3.send, a3.im, belt3.buf1);
21 s5: Switch(5, a4.t, a4.send, belt4.send, a4.im, belt4.buf1);
22 s6: Switch(6, a5.t, a5.send, belt5.send, a5.im, belt5.buf1);

25 // Definition of buffer belts
26 b1: Belt3(s1.send1, s1.is);
27 b21: Belt2(s21.send1, s21.is);
28 b22: Belt2(s22.send1, s22.is);
29 b23: Belt2(s23.send1, s23.is);
30 b3: Belt2(s3.send1, s3.is);
31 b4: Belt2(s4.send1, s4.is);
32 b5: Belt2(s5.send1, s5.is);
33 b6: Belt3(s6.send1, s6.is);

35 // Definition of belts
36 belt1: Belt5(s1.send2, s1.is);
37 belt21: Belt4(s21.send2, s21.is);
belt22 : Belt4(s22.send2, s22.is);
belt23 : Belt4(s23.send2, s23.is);
belt3 : Belt4(s3.send2, s3.is);
belt4 : Belt4(s4.send2, s4.is);
belt5 : Belt4(s5.send2, s5.is);
belt6 : Belt5(s6.send2, s6.is);

// Definition of pallet and empty pallet
type pallet = tuple (int id1; real t0; int a2; int a3; int a4; int a5);
const pallet leeg = (0,0,0,0,0,0);

// Definition of workstation A1, the generator and exit
def A1(alg int WIP; event receive; alg pallet p1):
  event send;
  cont ot = 0.0, gt = 0.0 der 1.0;
disc pallet ia = (start,gt,0,0,0,0);
disc int start = 1, count0 = 0, count1 = 0;
disc real flow = 0.0, through = 0.0;

// This location is only used to generate pallets at the start of simulation
location initialize:
  initial;
equation ot' = 0.0;
event send! when start <= WIP do start := start + 1,
   ia := (start+1,gt,0,0,0,0);
when start > WIP do ia := leeg goto wait;

// The separate events are made because flowtime and throughput
// should only be calculated after 7200 seconds
location wait:
equation ot' = 0.0;
event receive? when ia = leeg and gt <= 7200.0
  do ia := (p1[id1],gt,0,0,0,0), ot := 15.0,
    count0 := count0 +1, count1 := count1 + 1
  goto operation;
event receive? when ia = leeg and gt > 7200.0
  do ia := (p1[id1],gt,0,0,0,0), ot := 15.0,
    through := (count1-count0+1)/((gt-7200.0)/3600),
    flow := ((count1-count0)*flow + (gt-p1[t0]))/(count1-count0+1),
    count1 := count1 +1 goto operation;

location operation:
equation ot' = -1.0;
event send! when ot <= 0.0
  do ia := leeg goto wait;
end

// Other workstations are just a simplified version of A1
def Machine(alg int atype; alg real ot; event receive; alg pallet p1):
  event send;
disc pallet im = leeg;
cont t = 0.0;

location wait:
initial;
equation t’ = 0.0;
event receive? when im = leeg do im := operate(p1,atype), t :=

  goto operation;

location operation:
equation t’ = -1.0;
event send! when im != leeg and t <= 0.0 do im := leeg goto

wait;
end

// All buffer and belt definitions are identical, only the belt size
// changes
// Belt5 and Belt4 are used for belts B1
def Belt5(event receive; alg pallet p1):
  event send;
disc pallet buf5 = leeg, buf4 = leeg, buf3 = leeg,
  buf2 = leeg, buf1 = leeg;
  cont mt5 = 0.0 der 1.0, mt4 = 0.0 der 1.0, mt3 = 0.0 der 1.0,
  mt2 = 0.0 der 1.0, mt1 = 0.0 der 1.0;

  // Belts are split into discrete places where pallets stay
  // for the exact amount of time it would take to move through the
  // place continuously
  location:
  initial;
  event receive? when buf5 = leeg do buf5 := p1, mt5 := 0.0;
  when buf4 = leeg and buf5 != leeg and mt5 >= 4.0/3.0
    do buf4 := buf5, buf5 := leeg, mt4 := 0.0;
  when buf3 = leeg and buf4 != leeg and mt4 >= 4.0/3.0
    do buf3 := buf4, buf4 := leeg, mt3 := 0.0;
  when buf2 = leeg and buf3 != leeg and mt3 >= 4.0/3.0
    do buf2 := buf3, buf3 := leeg, mt2 := 0.0;
  when buf1 = leeg and buf2 != leeg and mt2 >= 4.0/3.0
    do buf1 := buf2, buf2 := leeg, mt1 := 0.0;
  event send! when buf1 != leeg and mt1 >= 4.0/3.0
    do buf1 := leeg;
end

def Belt4(event receive; alg pallet p1):
  event send;
disc pallet buf4 = leeg, buf3 = leeg,
  buf2 = leeg, buf1 = leeg;
  cont mt4 = 0.0 der 1.0, mt3 = 0.0 der 1.0,
  mt2 = 0.0 der 1.0, mt1 = 0.0 der 1.0;
location:
initial;

event receive? when buf4 = leeg do buf4 := p1, mt4 := 0.0;

when buf3 = leeg and buf4 != leeg and mt4 >= 4.0/3.0
do buf3 := buf4, buf4 := leeg, mt3 := 0.0;
when buf2 = leeg and buf3 != leeg and mt3 >= 4.0/3.0
do buf2 := buf3, buf3 := leeg, mt2 := 0.0;
when buf1 = leeg and buf2 != leeg and mt2 >= 4.0/3.0
do buf1 := buf2, buf2 := leeg, mt1 := 0.0;

event send! when buf1 != leeg and mt1 >= 4.0/3.0
do buf1 := leeg;

end

// Belt3 and Belt2 are used for buffer belts B2 and have an extra delay
to account for switching time

def Belt3(event receive; alg pallet p1):
  event send;
  disc pallet buf3 = leeg, buf2 = leeg,
  buf1 = leeg;
  cont mt3 = 0.0 der 1.0, mt2 = 0.0 der 1.0, mt1 = 0.0 der 1.0;

location:
initial;

event receive? when buf3 = leeg do buf3 := p1, mt3 := 0.0;
when buf2 = leeg and buf3 != leeg and mt3 >= 2.0 + 4.0/3.0
do buf2 := buf3, buf3 := leeg, mt2 := 0.0;
when buf1 = leeg and buf2 != leeg and mt2 >= 4.0/3.0
do buf1 := buf2, buf2 := leeg, mt1 := 0.0;

event send! when buf1 != leeg and mt1 >= 4.0/3.0
do buf1 := leeg;
end

def Belt2(event receive; alg pallet p1):
  event send;
  disc pallet buf2 = leeg, buf1 = leeg;
  cont mt2 = 0.0 der 1.0, mt1 = 0.0 der 1.0;

location:
initial;

event receive? when buf2 = leeg do mt2 := 0.0, buf2 := p1;
when buf1 = leeg and buf2 != leeg and mt2 >= 2.0 + 4.0/3.0
do buf1 := buf2, buf2 := leeg, mt1 := 0.0;

event send! when buf1 != leeg and mt1 >= 4.0/3.0
do buf1 := leeg;
end

def Switch(alg int atype; alg real pcheck; event receive1, receive2; alg pallet p1, p2):
  event send1, send2;
  disc pallet is = leeg;
  cont mt = 0.0 der 1.0;

location:
//receive1 has priority over receive2 when it is about to receive

[Initial]

event receive1? when is = leeg
do is := p1, mt := 0.0;

[Event receive2? when is = leeg and]
(pcheck >= 4.0/3.0 or pcheck <= 0)
do is := p2, mt := 2.0;

event send1! when is != leeg and crit(is,atype) and
mt >= (2.0 + shortcheck(atype))
do is := leeg;

//send2 will send when criteria are not met or when send1 does not send in time

event send2! when is != leeg and
(nott crit(is,atype) and mt >= (2.0 + shortcheck(atype))
or (crit(is,atype) and
mt >= (2.0 + shortcheck(atype) + 0.001)))
do is := leeg;
end

//The only difference with a normal switch is that Switch2 also
//checks if the next machine A2 has enough space to receive

def Switch2(alg int atype; alg real pcheck; event receive1, receive2;
alg pallet p1, p2, check1, check2, check3,
check4, check5, check6, check7):

[Event receive1? when is = leeg]
do is := p1, mt := 0.0;

[Event receive2? when is = leeg and]
(pcheck >= 4.0/3.0 or pcheck <= 0)
do is := p2, mt := 2.0;

event send1! when is != leeg and crit(is,atype) and
mt >= (2.0 + shortcheck(atype)) and
bufcheck(check1,check2,check3,check4,check5,check6,
check7) > 1
do is := leeg;

[Event send2! when is != leeg and]
(nott crit(is,atype) and mt >= (2.0 + shortcheck(atype))
or (crit(is,atype) and mt >= (2.0 + shortcheck(atype) + 0.001)))
do is := leeg;
end

//shortcheck is used because A3 and A6 do not have space between switches

func real shortcheck (int atype):
if atype = 3 or atype = 6:
    return 0.0;
else
    return 4.0/3.0;
end

// Function crit determines whether a pallet is allowed to move up to a workstation
func bool crit (pallet p; int a):
    bool b;

    if a = 1 and p = (p[id1],p[t0],1,1,1,1):
        b := true;
    end
    if a = 2 and p = (p[id1],p[t0],0,1,1,1):
        b := true;
    end
    if a = 2 and p = (p[id1],p[t0],0,0,0,0):
        b := true;
    end
    if a = 3 and p[a3] = 0:
        b := true;
    end
    if a = 4 and p[a4] = 0:
        b := true;
    end
    if a = 5 and p[a5] = 0:
        b := true;
    end
    if a = 6 and (p[a3] = 0 or p[a4] = 0 or p[a5] = 0):
        b := true;
    end
    return b;
end

// Function operate changes the integer corresponding to machine type
func pallet operate (pallet p; int a):
    pallet p1;

    if p != leeg:
        p1 := p;
        if a = 2:
            p1[a2] := 1;
        end
        if a = 3:
            p1[a3] := 1;
        end
        if a = 4:
            p1[a4] := 1;
        end
        if a = 5:
            p1[a5] := 1;
    end
if a = 6 and p[a3] = 0:
    p1[a3] := 1;
end

if a = 6 and p = (p[id1],p[t0],p[a2],1,0,p[a5]):
    p1[a4] := 1;
end

if a = 6 and p = (p[id1],p[t0],p[a2],1,1,0):
    p1[a5] := 1;
end
else
    p1 := (99,-1.0,1,1,1,1);
end

return p1;

// This function is used in switch2 to determine whether
// the pallet should go to the next workstation A2

func int bufcheck (pallet p1,p2,p3,p4,p5,p6,p7):
    int i;

    if p1[id1]!= 0 and p1[a2] = 0:
        i := i + 1;
    end

    if p2[id1]!= 0 and p2[a2] = 0:
        i := i + 1;
    end

    if p3[id1]!= 0 and p3[a2] = 0:
        i := i + 1;
    end

    if p4[id1]!= 0 and p4[a2] = 0:
        i := i + 1;
    end

    if p5[id1]!= 0 and p5[a2] = 0:
        i := i + 1;
    end

    if p6[id1]!= 0 and p6[a2] = 0:
        i := i + 1;
    end

    if p7[id1]!= 0 and p7[a2] = 0:
        i := i + 1;
    end

    return i;