On the Control of the Paint Factory Scale Model

– Case Study of a Production System Located at the TU/e –

Olivier Boutin and Jan H. van Schuppen

olivier.research@gmail.com – J.H.van.Schuppen@cwi.nl

November 14, 2011

1 Introduction

This report is about the modelling and control of a paint factory scale model, located at the *Technische Universiteit Eindhoven*[1] (TU/e) in Eindhoven, Netherlands. The goal of this study is to apply supervisory control theory (SCT), as defined by Peter J. Ramadge and W. Murray Wonham in their well-known article [Ramadge and Wonham, 1989], on a quite realistic system. But the classical state space explosion problem of SCT arise also in this case study. Thus we decided to apply the theory of coordination control, in order to synthesise several smaller supervisors and overcome this complexity issue. The two next sections describe the physical elements that take part in the scale model and its desired behaviour. A first attempt at TU/e to fully model the system using the SCT framework by a student had shown[2] that the system is too complex for the computation of a single supervisor using standard SCT [Hamer, 2007]. We will see in section 4 how far this student and other ones have managed to model the physical system and compute a supervisor for it.Section 5 deals with our current proposition to model this system, thanks to the theory of coordination control, and includes a discussion of the results at the end.

2 Physical Description

The goal of the paint factory is to deliver to a client any colour this person orders, after some due processing time. Only the 3 primary colours are stored in the plant and the system may have to mix some of them to deliver the desired colour. Up to 12 consecutive orders can be delivered in a row into cups, that the client will eventually collect. Every time a cup is collected, a new order can be made. After a given colour has flowed into some of the internal pipes, the latter usually have to be cleaned by letting cleaning fluid flow into them. This dirty fluid is eventually flushed into a waste vessel.

[1] "Eindhoven University of Technology" in English.

[2] But only to some extent – see the discussions in Section 4.2.

The paint factory is a complex system composed of many physical elements, a picture of which can be found in Figure 1.

Figure 1: View of the whole paint factory

We will not consider in this report the underlying electronic part of this system, of which description can be found in [van Rooy, 2007]. Thus we leave the electric switchboard out of our concern and focus on the physical elements, as displayed in Figure 2. These elements are described as follows.

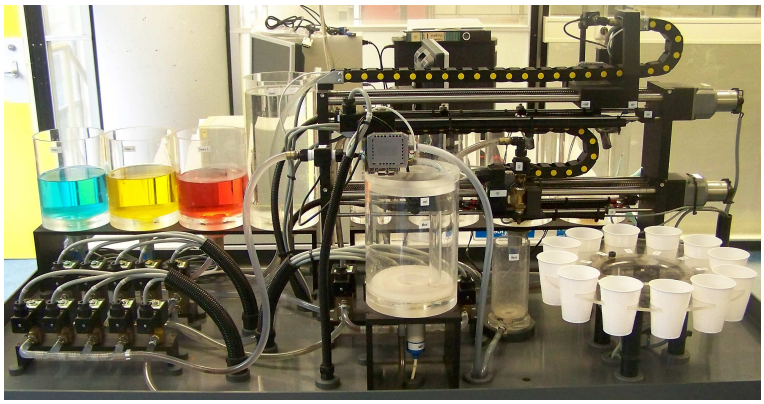

Figure 2: Closer view of the physical system

The vessels: A total of 9 vessels take part in the paint factory. 4 of them contain the inputs (the 3 primary colours and a cleaning fluid, as displayed in Figure 3); another one collects different quantities of primary colours in order to mix them; and three other vessels are buffers enabling parallel operations to take place or to store such colours for future use (see the picture of Figure 5, taken from the back of the plant). The last vessel is to collect all the waste, i.e. all the colour fluids that are not of interest for the client and need to be cleaned out.

Figure 3: The input vessels

The pipes: The fluids can flow in the system thanks to pipes, connecting the different interfaces of the other physical elements, until they eventually fill cups located at the end of the system, or go to the waste. Cleaning fluid is to be used to clean a given pipe/vessel after some coloured fluid has flowed down it. This is why a direct connection is needed from the cleaning fluid vessel to every pipe/vessel of the system.

The pumps: A pump is located downstream each vessel, in order to drain specific quantities of fluid into the pipes downstream these vessels.

The valves: The system includes a total of 18 valves. 11 of them have 2 connection points (1 input and 1 output) and can either be open or closed. They are used to retain the fluids in given parts of the system while other things happen simultaneously in the rest of the plant. The other 7 ones have 3 connection points, are always open and connect an input (respectively output) to one of two possible corresponding outputs (resp. input). They act as routing switches. See the translucent tubes in Figure 4; the top row is composed of routing switches (2 output tubes) and the bottom row is composed of open/closed valves.

Figure 4: The two kinds of valves in the paint factory

The overhead taps: The paint factory has two overhead mobile taps[3], that are used when a fluid is to be poured into one of several containers. Using such a mobile device appeared to be more convenient than pluging all the necessary pipes. Both of the taps consist in a motor and a pouring device in the form of a nozzle that drive the incoming flow to a desired container. A sensor is located on top of each of them. The first overhead tap drives either a mixed colour into a given buffer vessel, or some cleaning fluid, that have been used into the mixing vessel, to the waste (see Figure 5). The second one drives either an ordered colour to the exit point, the turntable (see below), or some cleaning fluid, used in the pipe for ordered primary colours or the pipe downstream the buffer vessels, to the waste.

The turntable: The turntable is a rotative holder containing 12 cups, displayed in Figure 6. The rotation of this element is also driven by a motor together with sensors, in order to know the current position and decide when to stop turning.

A schematic design of these elements and the way they are connected can be found in [van Rooy, 2007, Figure 1.2, page 3] and is reproduced in Figure 7. All the labels in Sections 3 and 4 will be based on that figure.

Even though these physical parts are well connected in the paint factory, one needs to express specifications for that plant, so that the sequential order of events is respected, the colours are not mixed if it is not required, a colour

[3]These components were called "slides" in all the student reports, apparently because this element, together with the screw, was assimilated to a slider-crank mechanism – see `www.britannica.com/EBchecked/topic/548729/slider-crank-mechanism`, where a crank is used to change a circular into a repetitive up-and-down or back-and-forth motion of a slide. But this repetitive property is not prevalent, nor the tap is actually sliding back and forth *on top of a surface* but *along some tubes*. This is why we do not stick to this name.

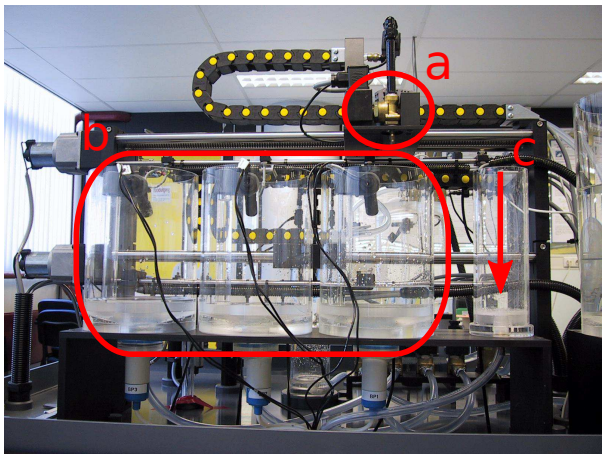

Figure 5: The tap (a) on top of the buffer vessels (b). The extra vessel (c) leads directly to the waste

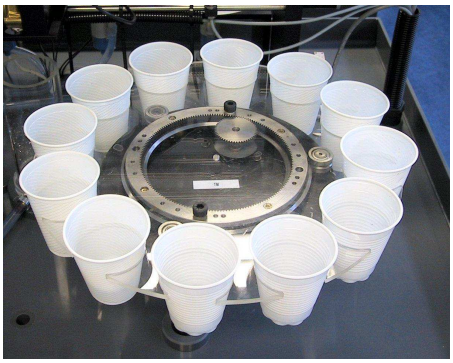

Figure 6: The turntable and its cups

fluid should not be flushed to waste, and so on. To sum up, the goal of the specifications is to prevent any undesired event to happen.

3 Specifications

As a matter of fact, we found no single formal specifications in the documentation of the paint factory. They are spread amongst the different reports that were available, and need to be gathered. Here is an attempt to provide a list of specifications, as complete as possible, leading to a proper behaviour of the system. The different rules that the system has to follow are numbered, in order to come back to them more easily in the remainder of the report.

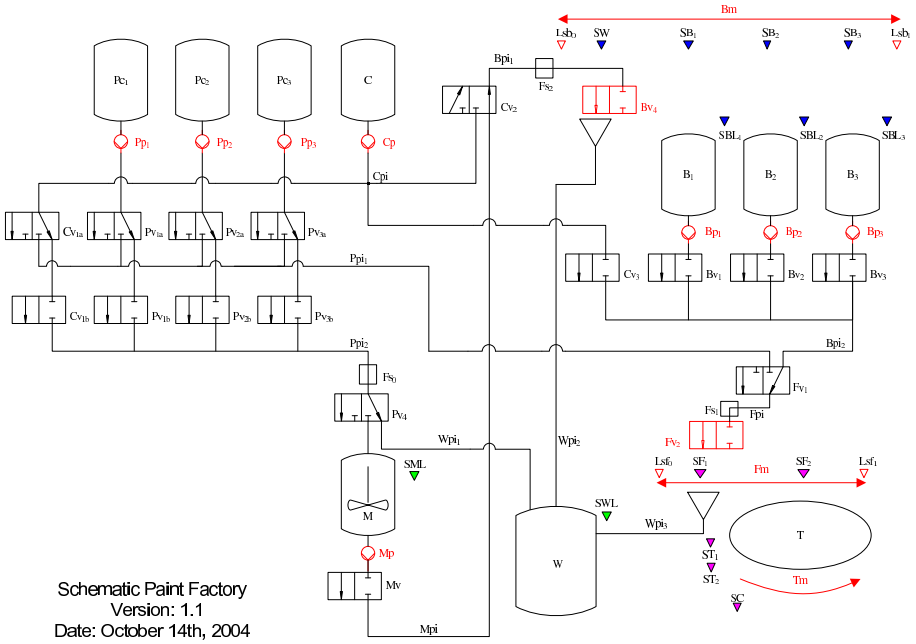

Figure 7: Schematic design of the paint factory

3.1 Path

Rules related to the proper setting of a path thanks to the valves.

Spec1 Before a fluid is pumped into a pipe, the downstream switch valves on the path to the next vessel or the exit point should be set up in the correct position. And then, these valves should not move before the end of the pumping.

Spec2 The valves shutting the pipes of the two overhead taps (B_{v_4} and F_{v_2}) should not open when the corresponding tap is moving.

Spec3 Only one of the downstream valves of the cleaning fluid vessel (i.e. $C_{v_{1b}}$, C_{v_2} and C_{v_3}) can be open at any time.

3.2 Flow

Rules related to the pumps, the pipes and the elements these pipes feed into.

Spec4 No more than one fluid (be it coloured or a cleaning one) can flow down a pipe at a time.

Spec5 When a fluid is pumped out into another vessel, the fluid into the latter should not be pumped out at the same time.

Spec6 A cup cannot be filled twice before its content is collected.

3.3 Cleaning

Rules related to the cleaning vessel and some pipes.

Spec7 A pipe has to be cleaned each time some colour fluid has flowed into it and a vessel has to be cleaned when it is emptied.

Spec8 A (dirty) cleaning fluid should only exit the system into the waste vessel and a colour fluid should only exit the system into a cup.

4 State of the Art at the TU/e

Several students of the TU/e have worked on that scale model in the scope of projects, using automata[4]. Four reports are available ([van den Bremer, 2005], [Hamer, 2007], [Roberts, 2008] and [Hoefnagels, 2008]) and we will discuss here how they have dealt with the complexity and the decomposition of the system into sub-systems.

It is worth mentioning that another report dealing with the modelling of the Paint Factory is also available [Triepels, 2006]. But what this student calls supervisors are merely schedulers, designed in an automata framework using UPPAAL [Behrmann et al., 2004]. As this does not fit in a supervisory control scope, we will not present it here.

4.1 Work by [van den Bremer, 2005]

This project has been undertaken at a Bachelor's degree level. This is mostly why, in our opinion, the proposed models are not really relevant; especially because of the simplifying assumptions made for the whole model to be manageable. Another reason is that it seems to be the very first student work dealing with that scale model. So he could not build up on previous work and knowledge. In fact, the supervisory synthesis work has not actually been finished. The synthesis steps using the TCT tool[5] are provided, but the computed supervisor has not been validated[6].

Every controllable physical element has been modelled in that report. That is to say, according to the list provided in Section 2: valves (grouped as so-called manifolds), pumps and motors.

We state now what are the interesting points and what we would like to avoid in our modelling of the system, justifying why. First, let us list the identified interesting points:

[4] Formally speaking, we should talk about generators, because the transition functions of the models are only partial.

[5] Available at `www.control.utoronto.ca/~wonham/`.

[6] As a matter of figures, the supervisory controller obtained with this modelling (though wrong) has 7,560 states and 171,336 transitions.

- Grouping the valves in so-called manifolds, depending on their role in the system. These can be seen as sub-systems, although the models provided in this report are not so reliable.

- The idea of trying to avoid cleaning when it is not needed, to save paint and time, by specifying the last colour used in a pipe/vessel. Then the latter would not be cleaned for nothing in case a further order with the exact same colour happens in the future. But it might also be a loss of time to always wait for future order before cleaning the pipes and vessels. Indeed, every new job would have to wait for the cleaning steps first, and could not be executed right away. Also, this modelling might quickly lead to a state space "explosion", by having to store the last used colour as extra information in the state of a physical system part.

We also have gathered everything that we considered to be a drawback into the following list:

- The level of abstraction of the system seems very low. This naturally leads to a lot of small models to take into account. Because he realised that the generator of the whole plant would include too many states then, the student made simplifying assumptions that actually allows some undesired behaviours, (e. g. "opening" or "closing" events for the valves can happen at any time, which is not reliable). Moreover, when grouping some models of the physical elements together, too many considerations are taken into account, so that a lot of self-loops have to be included in the generators.

- Even though two kinds of valves co-exist, all of them are modelled in the same way. In the report, each valve is said to be either "open" or "closed", although the valve switches are in facts always open, and connect either one of two entry (respectively exit) points to the corresponding exit (resp. entry) point. This is quite confusing when retrieving what each event of the generators actually means.

- Several times, there is a confusion between the necessity of disabling the opening of a valve in a given state of the system and the uselessness of enabling it in that same state.

- There are some modelling mistakes (forgotten events and initial/marked states).

- Sub-models are grouped regarding their kind and not their function. This leads to inconsistent consideration, especially regarding the pumps.

- No sensors are taken into account. So the system never knows when the right quantities of fluid are reached at some checkpoints, and the overhead taps never know where they are actually located.

4.2 Work by [Hamer, 2007]

This report is a master's thesis. Even though, there is no direct reference to it, this work seems to be the extension of the one reported in [van den Bremer, 2005]. In any case, the models are similar, to a large extent. Using so-called model based engineering[7], the system is split into components. The chosen approach is to model every single controllable part (i.e. valves, pumps and motors) as a component, using the automata theory formalism.

4.2.1 On the Models of the Physical System

The given models of the pumps are actually merged with the ones of the corresponding switch valves. Indeed, whatever the pump is, each model usually only contains 2 states ("on" and "off") and 2 transitions (to turn it on or off). Whereas here, events of the transitions of the provided models also embed the instruction of what destination the fluid will eventually reach, which is a matter of the switch valves. Some valves are grouped in so-called valve matrices, which correspond to what were called manifolds in [van den Bremer, 2005]. This time, the model of the physical system is mixed with the one of the specification, since they already state that only one fluid can flow in a pipe at a time. The level of abstraction of the motors is really high, since no information about their position is used (see Section E.3 of that report). Only the paths that the fluids have to take are used to drive the succession of the states of the motors. On the one hand, this helps to have smaller models (which is a big issue in the SCT framework), but this might be too abstracted in this very case. Since only the *controllable* parts of the system are modelled, all the sensor part is actually included in the events ending a given task, which should actually be considered *uncontrollable*. The only kind of specified tasks (or events) are pumping ones. So they all rely on sensors that tell us when to stop (when the desired quantity of fluid is delivered). Thus, this is inconsistent with the generators given for the controllable parts. Another problem is that in most models, different reasons to turn on or off a physical part are mixed in the model of that very part, so that erroneous signals can lead to an unwanted state of the system.

4.2.2 On the Models of the Specifications

Only two models of the specification are given, since the missing ones are said to be similar[8]. The model for the mixing buffer allows for only doing one colour mix and does not enable a mix based on 3 colours, so it is not re-usable in our work as such. Also, the self-loops included in the specification models do not seem to be accurate, since for instance the event "stop pumping from mixer to buffer 1" appears in almost every states of the specification for buffer 1 (see Figure E.23 of that report).

[7] A modelling framework taught at the TU/e.

[8] As a matter of figures, the supervisory controller obtained with this modelling (though wrong to our point of view) has approximately 20,000 states and 85,000 transitions.

4.2.3 Other Remarks

Finally, a scenario of a client ordering a mix of 2 colours (primary colours 1 and 2) is provided, in the form of a generator. But the given sequence actually also includes the interaction with primary colour 3, which is undesired. Such an order should only include desired states and specify what the system is expected to do, and not tell how to recover from an error.

Moreover, the validation of the work on the files has not been undertaken, due to a lack of time.

4.3 Work by [Roberts, 2008]

The goal of this master's thesis work is to investigate a different supervisory control scheme, as defined by E. Tronci in [Tronci, 1998]. But the formalism and synthesis of the specifications provided in that report are not suitable for SCT. The main drawback of Tronci's method is that no distinction can be made between controllable and uncontrollable events. Indeed, uncontrollable events that lead to a plant failure are modelled thanks to an extra failure state. Hence the controllability condition, as defined by W. Murray Wonham [Wonham, 2005], might not be satisfied. Nevertheless, the modelling of the plant is based on automata and some issues are still relevant in the design of our modelling method.

In that report, schedulers are integrated as plant components. The student compares two ways of including them in the models. In fact, the second automata is actually just a projection of the first one, withdrawing the stopping events of a pumping operation. This student prefers the second alternative because it has less states and transitions, but this is a mere consequence of that projection. It does ensure that any pipe is cleaned after it is used, but one would still need specifications to make sure the sequences of event of a pumping operation are executed properly, for instance. The problem raised here is related to the choice of the representative events of an operation (i.e. are the valve events more representative than any other one?). Another example of such a choice is for the so-called "output guarantee" property. The events that have been chosen to represent the fact that enough fluid have flowed down a given pipe is when valve F_{v_2} is opened and then closed (the liquid has been sent out of the system). It would have been much closer to reality to take the sensors signals into account.

The optimal controller synthesised by the scheme of Tronci takes into account the required languages (i.e. the actual orders of the client). The students states that a downside of Tronci's method is that everytime a desired output is added to the system, the controller needs to be synthesised again. Nevertheless, if one does not stick to the admissible specification language, the same drawback remains in SCT. This scheduling part refers to the required behaviour, and is not taken into account in the contributions of W. Murray Wonham and his co-authors. Only in [Cassandras and Lafortune, 2008, Section 3.4.4] appears a interest for taking into account such requirements in the supervisor synthesis. In that report, the alternative of computing a controller only for the admissible

language and drive the production thanks to a side or upper level real-time optimiser is suggested and could be investigated. In facts, in Figure 5.8 of his report, this student gives a sketch of what could be a hierarchical way of modelling such a plant, with a resource controller at the lower (physical) layer, and both a supervisory controller (for the admissible tasks) and a task controller (for the sequence of events to be executed) at a higher level. In Section F.2 is also suggested a hierarchical way a decomposing tasks, in terms of primary colours needed for the secondary-colour orders. Hierarchical modelling is at the heart of complexity management and shall be integrated in our method.

Moreover, we would like to give two more comments about the actual correctness of that study. The optimisation algorithm is only based on seeking a shortest path in the state space (i.e. the least number of transitions to reach a final state). But this criterion is heavily related with the modeller's skills and should be more closely related to a physical system measurement. And altough the paint factory can deliver any colour, only the six primary and secondary colours are taken into account in that report.

4.4 Work by [Hoefnagels, 2008]

That report is also a master's thesis. Its content is the continuation of the previous reports dealing with SCT, i.e. [van den Bremer, 2005] and [Hamer, 2007]. After the conclusion of [Hamer, 2007] regarding the state space explosion of the paint factory model, the alternative of the state tree structures (STS), as defined by Ma and Wonham [Ma and Wonham, 2005], has been invastigated.

4.4.1 On the Models of the Physical System

A first interesting point is that the whole system is divided into logical parts, namely:

- The primary colours sub-system.
- The cleaning sub-system.
- The mixing vessel sub-system.
- The buffer sub-system.
- The filling station sub-system.
- The flow sensors.

Some abstraction has been used not to have to model every single component. For instance, the pumps are considered to be controlled together with the valves. This allows for simplified generators and the actual actions for these resources is left to be adapted at a lower level.

4.4.2 On the Models of the Specifications

When specifications are logical statements (especially of the kind of mutual exclusions), they are modelled thanks to logical expressions using boolean variables. Otherwise, when a constraint corresponds to a sequential order of events, like a recipe, a generator is used to model it in a formal way. The fact that the boolean variables of these logical expressions should not have the same name as the variables of the binary decision diagrams is quite confusing and seems to be part of the suggested way of modelling specifications thanks to STS. In facts, the former should not include underscore characters ('_') in their names, to avoid problems of name clashes. Moreover, no links are given between the boolean variables used in a logical expression of the report and the corresponding states of a previously defined generator, which does not help reading.

The specification for the mixing operation, which is the most complex one, seems not to be accurate (see Figure A.7 of that report). In particular, the consecutive colour delivery processes are wrongly interpreted and some internal states leading to valve closings/openings are missing. Also, the generators of the two overhead taps are quite different in accuracy. Indeed, Figure A.22 of that report seems to only model the behaviour of the motor driving the overhead tap upstream the filling station, while the overhead tap upstream the buffer sub-system is decomposed into Figure A.14 for the behaviour of the motor and Figure A.15 for the one of the actual overhead tap. Moreover, the generator of Figure A.15 is not really consistent, because we do not see the point of defining extra states to express the last direction (right or left) used by the overhead tap before stopping. To take into account these moving directions, only the controllable events "left" and "right" are necessary. In addition, it seems that, instead of being abstracted, hardware issues have been included in the model because of the use of an uncontrollable event while the device is moving between two sensors. We think this event is meant to represent the low level signal "keep moving in this direction" sent to the motor every period of the system clock, which is not to be considered at our level of modelling.

4.4.3 Other Remarks

As in [van den Bremer, 2005], this student tends to talk about "opening" or "closing" switches, which does not really seem convenient to us (see our comments in Section 4.1).

In this report, the STS framework is said to be suitable for this complex system but there is no justification whatsoever[9].

[9] As a matter of figures, only some measures on the binary decision diagrams of the STS are available. So we cannot really compare it with the previous works. The sizes of the computed specifications, optimal controlled behaviour and of all control functions are respectively 638, 44,619 and 715 – the unit seeming to be the number of nodes.

5 Our Proposition

In order to overcome the underlying complexity of the paint factory, we have decided to use the coordination control framework. Its goal is to dedicate a specific controller to administrate the shared elements of a system and let all sub-systems deal internally with their local elements, thanks to dedicated controllers making sure the specifications are satisfied. The advantage is to cope with state space explosion and move from a monolithic control to a distributed one [Komenda et al., 2010].

5.1 Description of the Case Study

We have decided to address only a sub-part of the whole system. This study focuses on controlling common pipe P_{pi_2}, that is used to drive the primary colours to the mixing vessel. This is the first critical section to manage in the paths from the primary colour vessels to the turntable – unless only pure primary colours are always ordered. This sub-system includes all the input fluids of the system: the 3 primary colours to be mixed in the mixing vessel and the cleaning fluid. The controllers that we want to synthesise for this smaller system are to make sure that the 4 fluids do not get mixed in pipe P_{pi_2} (see Spec4).

For our convenience, we have relabeled some of the physical elements. Table 1 shows how this renaming has been applied.

[van Rooy, 2007]	Our modelling
P_{p_1}	P_B
P_{p_2}	P_Y
P_{p_3}	P_R
$P_{v_{1a}}$	S_1
$P_{v_{2a}}$	S_2
$P_{v_{3a}}$	S_3
$P_{v_{1b}}$	V_B
$P_{v_{1b}}$	V_Y
$P_{v_{1b}}$	V_R
P_{v_4}	S_4
M_p	P_M
M_v	V_M
C_p	P_C
$C_{v_{1a}}$	S_7
$C_{v_{1b}}$	V_C

Table 1: Renaming conventions

A schema of the sub-system we are dealing with is depicted in Figure 8.

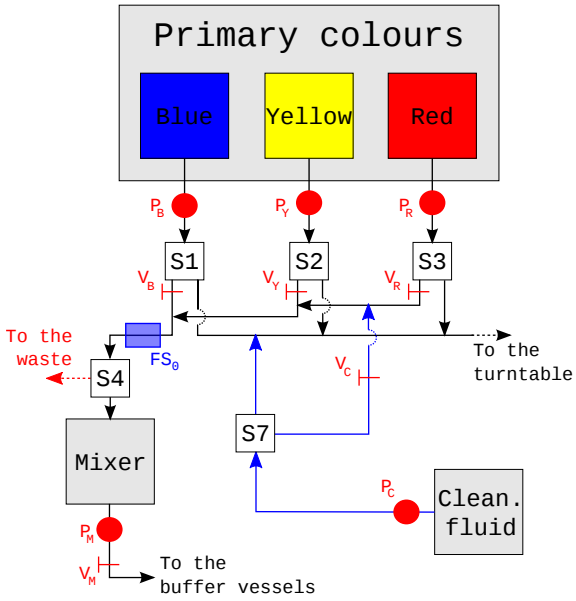

Figure 8: Schematic view of the sub-system we study

At the initial state, we assume that all the pumps are off, all the open/closed valves are closed and all the switches are on the left position. The possible positions of the switch valves (left/right) are taken from the point of view of the flow. For instance, if we want to drive some blue colour from its vessel to the mixer, Switch S_4 is already in the correct position and we will have to open V_B, flip S_1 to the right and eventually turn on P_B (V_M is already closed so there is no risk of a leak in the mixing vessel).

Based on the previously mentioned drawbacks of the available studies for this system, we have decided to model the system from scratch. All the models we have designed will be described in the following sub-sections.

5.2 The Models

The system can be decomposed in both an horizontal and a vertical ways. The plant can be split into sub-systems, in a horizontal fashion, and then be refined hierarchically with layers, in a vertical fashion.

At each given layer, none of the components is supposed to communicate information, events or signals, with the other ones. Communication is only done with elements of the layers directly above or below. For this case study, the bottom-most layer is a physical layer including all the atomic models of all the parts. These parts models are then merged in order to get operations, only

including the needed parts. Afterward, the operations are coordinated thanks to local controllers and a coordinator for each shared section.

The coordinators are heavily related to what is shared. The system is to deliver products, and some available paths between two vessels in the system are shared between several pumping operations. This lead us to consider that the coordinators are the shared pipes, in terms of all the parts located in the shared paths (flow sensors, switches and valves). For the time being, only one pipe is shared so we only need one coordinator.

Our following systemic view of the system is depicted in Figure 9.

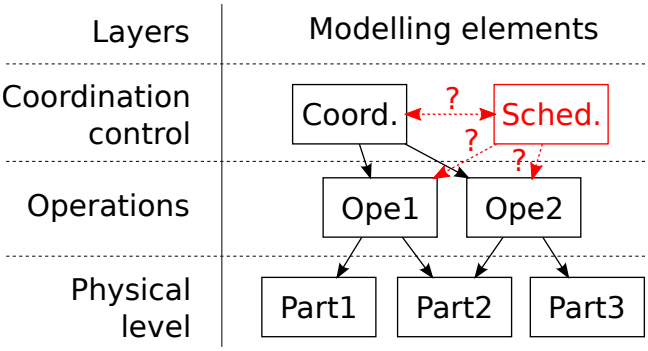

Figure 9: Hierarchical view of the paint factory

It seems the system needs to be influenced by both the coordinators (that deal with the *admissible* language) and the schedulers (that deal with the *required* language). For the time begin, we only tackle the admissible language. We will inquire in a later stage how to also deal with the required one, following the track of [Cassandras and Lafortune, 2008, Section 3.4.4]. Also, when the system will be complex enough, it is still arguable whether or not the different coordinators will themselves need to be coordinated, until a top-level layer for the colour orders[10].

A missing part in our modelling procedure is how to get the operation models, from the ones of the different physical parts. Our feeling is that we could find them by inspecting in the state space of the system, and not of the automata, how to pump the fluids as desired (how to go from the state "the fluid is in this vessel" to the state "so many quantity of the same fluid is in that vessel"?). Then, using a minimisable criterion, we could get an optimal recipe of actions to take; in other words: the operation models we have designed by hand. This could be a solution to automate this design. Some references of insterest are [Feng and Wonham, 2010] and [Ricker and Caillaud, 2007], where algorithms or general approaches could be found.

After discussing the way of decomposing the system into sub-models, let us describe the models themselves.

[10] In other words: does a 4^{th} layer exist?

A pumping operation consists in 3 parts: the actual process, its preconditions, and the way to go back to the idle state. The process is to be delimited by representative events, because we need to be able to know when some fluid is pumped into the pipe and when it is not the case. We figured the representative events of a pumping operation are the ones actually driving the fluids, that is to say the events turning on or off a pump. This in turn marks out a critical section in the graph, according to the specifications (see Section 5.2). Indeed, Spec4 tells us that when a pump is operating, no other one should pump anything in the same pipe. For an operation to execute properly, we first need to set up the corresponding path correctly (moving all the parts in any order). In the end, we get back to the idle state of the sub-system putting all the local parts to their own idle state. Also note that, because FS_0 and S_4 are shared parts, the events related to them should appear in each sub-system. All the other events should be local to each corresponding sub-system. Nevertheless, as a current simplification, since S_4, V_M and P_M are supposed to stand still, we will abstract these valves and pump away. All these considerations have lead us to the generator of Figure 10, which we have designed thanks to the DESTool software [Moor, 2011], a graphical interface based on the libFAUDES library [Moor et al., 2008].

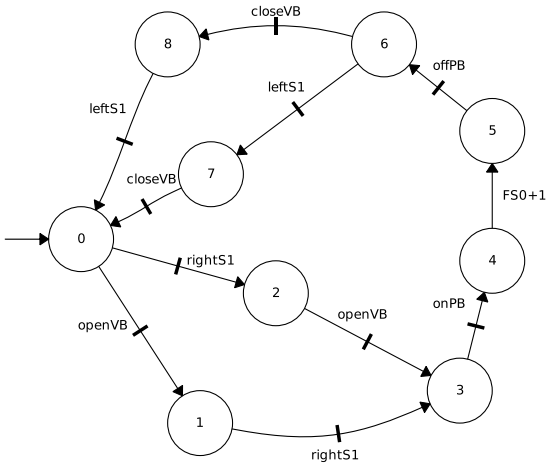

Figure 10: First draft of a pumping operation

Note that the single transition between states 4 and 5 is an abstraction of the actual needed quantity (event *FS0+1* meaning "flow sensor 0 has detected one more unit of fluid"). Flow sensors are considered accurate enough to only deal with integer quantities to mix. We also assume that the events will always happen as soon as possible. So we will not deal with the emergency character of events that must happen before something wrong happens (like stopping a pump before too much fluid has been pumped out).

In order to apply the coordination control framework, we need the sub-system to have specific properties. If the sub-systems are conditionally independent, and the specifications conditionally decomposable and controllable, then we know that the synthesised supervisors will be optimal (size-wise). In any case, provided extra specific properties, supremal conditionally controllable sublanguages can be computed [Komenda et al., 2010]. Let us discuss these properties and the consequences on the modelling if we want to satisfy them.

5.2.1 Nonblockingness of the Sub-systems and of the Plant

The theorems and algorithms available in [Komenda et al., 2010] assume prefix-closed languages (i.e. for which all states are marked). Some improvement, that can be found in [Komenda et al., 2011], allows for nonprefix-closed languages. But the authors expect a new so-called conditional closeness property, which is only provided for 2 competing sub-systems. While waiting for the proof of the correctness of the algorithms for any number of competing sub-systems, we will assume that this property holds true in our case.

To make the sub-systems nonblocking, one needs to mark at least one state, which expresses a final state or some state that we want to ultimately reach. As the operations are a never-ending underlying process, we think that the starting/idle state is an appropriate candidate to be marked. As any other state does not lead to a legitimate idle state, we will only mark that one.

Besides, event *FS0+1* is shared between all sub-systems. And any pumping operation is to be allowed, except when a competing sub-system has already been granted access to the corresponding critical section. Then, *FS0+1* self-loops have to be added to all the states that are not in the critical section. Otherwise, the whole system would tend to turn on all the pumps before counting anything. But this is strictly forbidden by the supervisor, and would thus lead to an uncontrollable and unrealistic controlled system.

A new version of the model of a pumping operation is given in Figure 11.

5.2.2 Output Control Consistency

Definition 1 (Output Control Consistency [Wong and Wonham, 1996])**.** *A projection $P : E^* \to E_k^*$, where $E_k \subseteq E$, is* output control consistent *(OCC) for a language $L \subseteq E^*$ if for all strings $s \in \overline{L}$ of the form*

$$s = \sigma_1 \dots \sigma_l \text{ or } s = s'\sigma_0\sigma_1 \dots \sigma_l, l \geq 1,$$

where $s' \in E^*, \sigma_0, \sigma_l \in E_k$ and $\sigma_i \in E \setminus E_k$, for $i = 1, 2, \dots, l-1$, $\sigma_l \in E_u$, then $\sigma_i \in E_u$, for $i = 1, \dots, l-1$.

In our case, this property implies that all local events that can precede an uncontrollable coordinated event should also be uncontrollable. In our way of modelling, the detection of a unit of fluid in Pipe P_{pi_2} by FS_0 (triggered by event *FS0+1*) is uncontrollable, because this event cannot be launched as a result of the action of a button, for instance. Flow sensors act on their own and signal when they have detected some quantity of fluid.

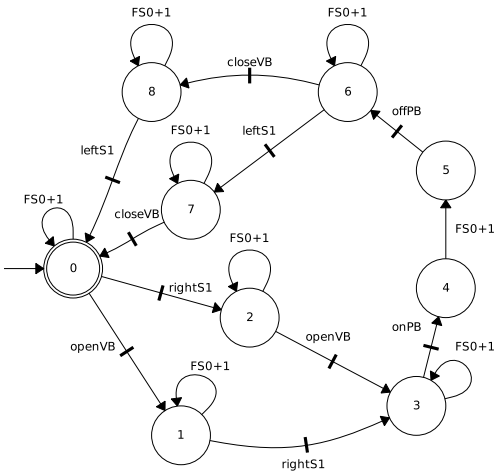

Figure 11: A nonblocking pumping operation

It turns out that projections P_k^{i+k} are not OCC[11]. Indeed, for OCC to be satisfied, either *FS0+1* should be controllable, or only located in the critical section, or all the local events of a sub-system should be uncontrollable. But none of these choices is possible in our way of modelling the system. Nevertheless, if we try and compute the supremal controllers for our sub-systems, we get correct results. It then seems that some hypothesis needs to be loosen in the theory, or that some specific set-up needs to be included in the allowed cases. Our intuition is that it is because, during the real-time execution of the different operations, event *FS0+1* will actually *always* be preceded by a coordinated controllable event (i.e. turning on the corresponding pump), but this is not known by the competing sub-systems.

As we cannot satisfy this property, we choose not to modify the previous version of the model of a pumping operation.

5.2.3 Language Observability

Definition 2 (Observer property [Wong and Wonham, 1996])**.** *A projection $P : E^* \to E_k^*$, where $E_k \subseteq E$, is an* L-observer *for a language $L \subseteq E^*$ if for all strings $t \in P(L)$ and $s \in \overline{L}$, $P(s) \leq t$ implies that there exists $u \in E^*$ such that $su \in L$ and $P(su) = t$.*

In the coordination control framework, we need the projections P_k^{i+k} to be $(P_i^{i+k})^{-1}(L_i)$-observers. This property implies that all the strings in a sub-system using only local events (i.e. that would be projected to ε using the alphabet of the coordinator) should eventually lead to the initial state.

[11] In facts, we check whether they are *local* control consistent, simply because OCC is not implemented in our tool. It is known that LCC implies OCC [Schmidt and Breindl, 2008].

The event set of the coordinator includes at least all the shared events. This means that *FS0+1* has to be part of it. Thus, all the states that are co-reachable from state 4 should either include transitions with events that will be added to the event set of the coordinator, or transitions enabling to eventually come back to state 0. As a matter of fact, if a valve or a switch is put back to its idle position, we somehow undo the process. So we have chosen to add transitions pointing back to states 0, 1 and 2. Then, we have added event *onPB* to the event set of the coordinator, because this cannot be undone before the right quantity of fluid has flowed down the pipe.

Then, as a matter of consistency, we have also added the transitions back to state 6 from states 7 and 8[12]. This new version of the model of a pumping operation is depicted in Figure 12.

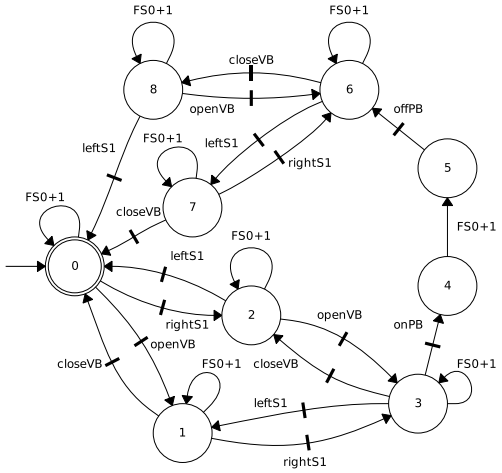

Figure 12: A nonblocking pumping operation satisfying the observer property

Because switch S_4 should not move during such a pumping operation, the events related to it do not appear in the model. Nevertheless, S_4 is definitely involved in the cleaning operation, and events *leftS4* and *rightS4* are in the coordinator model because S_4 is in the common path. One would like to design the cleaning operation after the pattern of Figure 12. But, to satisfy the observer property for the cleaning operation, we should withdraw the transitions leading back to a previous state after the actual pumping of the cleaning fluid (because events *leftS4* and *rightS4* are now involved). Nevertheless, it appears that this property is not needed either, as long as we have conditional controllability of the specifications; which is the case with the model of Figure 13, that we will use in the remainder of this study.

[12]We do not add transitions back to states 7 and 8 from state 0. They would mean that we have started a new process and are actually already included in the model as the transitions to states 1 and 2 from state 0.

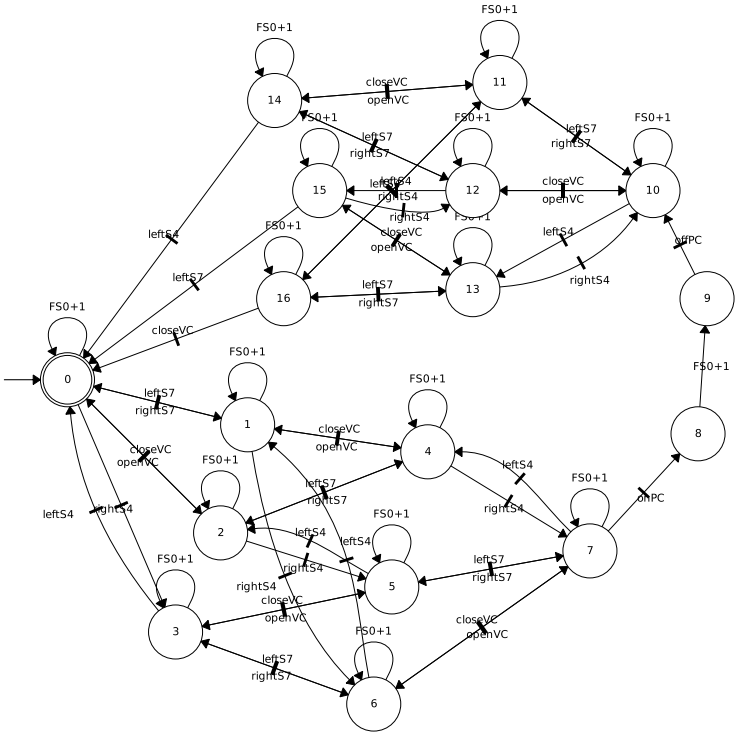

Figure 13: The cleaning operation

Specifications are meant to avoid undesired states (what is allowed or not), but never actually tell how to do things. This is why the recipes of the operations should not appear in the specifications.

So far, we are only addressing Spec4, which implies that, out of pumps P_R, P_Y, P_B and P_C, only one of them can be turned on at a time. This specification is an implementation of the classical mutual exclusion pattern. We first express the relationship between two such pumps, as depicted in Figure 14 (where the local events of each sub-system have been gathered around the idle state 0), and then we compose all the sub-specifications together, to get the whole Spec4. To be maximally permissive and allow as much parallelism as possible, all the local events of each sub-systems are always allowed while another one is operating, be it already in the critical section or not. For technical reasons, we had to put a dummy *FS0+1* event on the idle state as well. Indeed, as the different processes will be executed, this very event will actually never happen, because each of the mutual exclusions state that it should only happen after turning on a pump. But it is needed in fact, as an uncontrollable event allowed in the idle state of all the sub-systems, to make sure that the specifications are controllable, following the definition [Ramadge and Wonham, 1989].

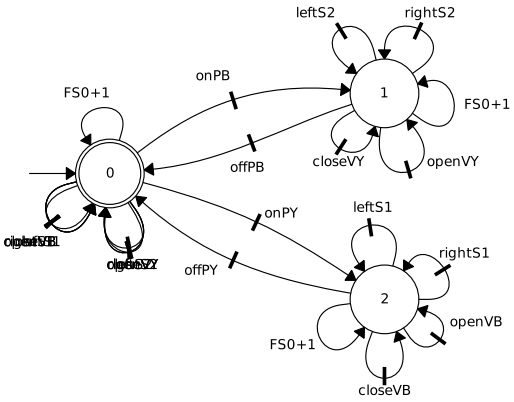

Figure 14: Mutual exclusion between the pumping of blue and yellow fluids in pipe P_{pi_2}

Figure 15 shows what are the whole specifications after composing the 6 binary mutual exclusion specification models.

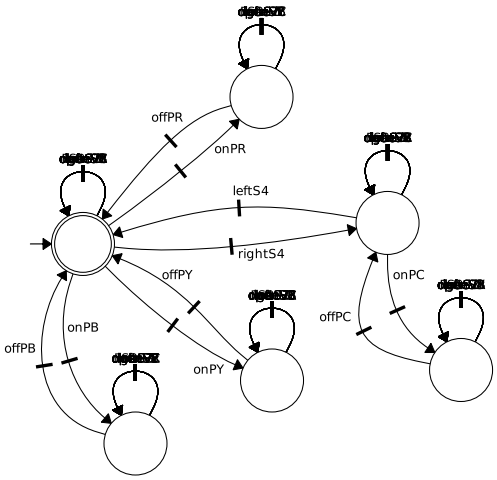

Figure 15: Mutual exclusion between all the fluids in pipe P_{pi_2}

According to the coordination control framework, the coordinator, denoted G_k, should add nothing new to the whole system. It is actually to be taken as the merging of some splitting of the sub-systems, as depicted in Figure 16, in terms of event sets. Note that sometimes, more events are included in the coordinator than the mere intersection of the event sets of the different sub-systems.

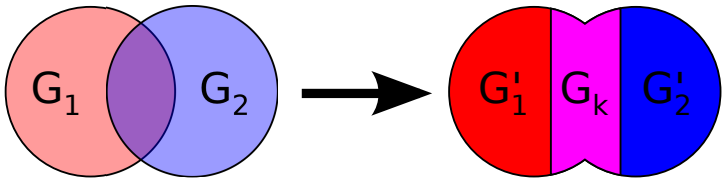

Figure 16: How to get the coordinator from the sub-systems

As stated before, events *FS0+1* and *onPB* have to be included in the coordinator event set. But for the mutual exclusion to operate well, we still need to know when a sub-system has released the critical section. This is why the events related to turning off the pumps also have to be included in this event set. Then, we need to extract the model of the coordinator from the one of the plant. However, we would like to avoid to compute the model of the plant, being the composition of all sub-systems. But we can take advantage of the fact that the intersection of the event sets of all sub-systems is be to included in G_k. This allows us for constructing the model of the coordinator as the composition of the projection of each subsystem on the event set of the coordinator that we have just designed. In other words and in our case:

$$P_k(L) = P_k(G_1||G_2||G_3||G_4) = P_k(G_1)||P_k(G_2)||P_k(G_3)||P_k(G_4).$$

In the end, we get the coordinator model depicted in Figure 17. Note that, as such, it does not satisfy the specifications (several pumps can be activated at the same time). So will need to apply a controller to it.

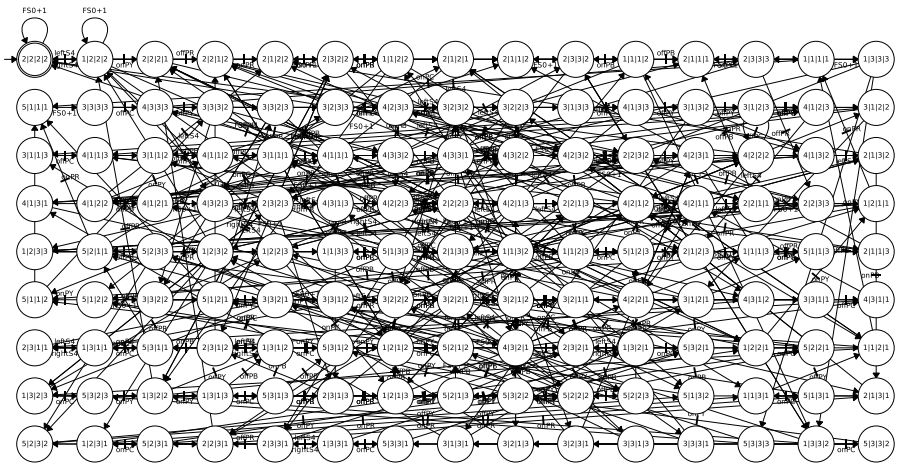

Figure 17: Model of the coordinator of pipe P_{pi_2}

As a matter of fact, figures of the models we deal with are gathered in Table 2. $G_i, i \in [1..3]$ corresponds to the pumping of a primary colour, whereas G_4 is the pumping of the cleaning fluid. It is easy to note that the state space of the whole system is much greater than the one of each sub-system.

	Whole plant	G_1	G_2	G_3	G_4	G_k
♯ states	15309	9	9	9	17	135
♯ events	27	7	7	7	9	11
♯ transitions	134899	24	24	24	63	437

Table 2: Sizes of the plant generators

5.3 The Control Synthesis

Thanks to the algorithms given in [Komenda et al., 2011], we get the following supervisors.

First, we compute the supervisor for the coordinator. As we can see in Figure 18, this controller indeed makes sure that the pumping operations cannot happen at the same time.

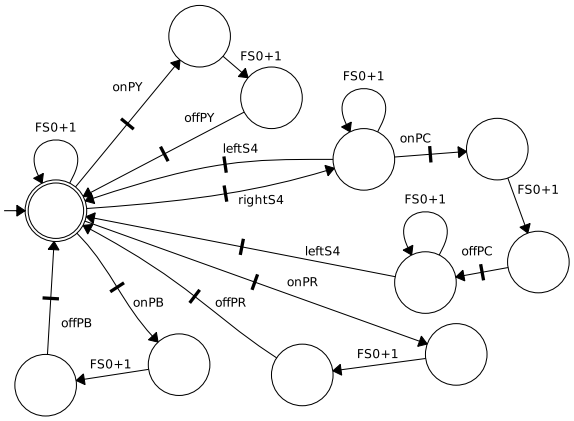

Figure 18: Optimal controller for G_k

Then, provided this supervisor, we can compute the ones for the coordinated sub-systems. As an example, the controller for the blue pumping operation is depicted in Figure 19. The pattern for the pumping operation is still appearing (see Figure 12), and all the states and transitions around this are meant to deal with the parallelism of the pumping operations: as long as we are not in the critical section, any other sub-system can pump some fluid in the shared pipe.

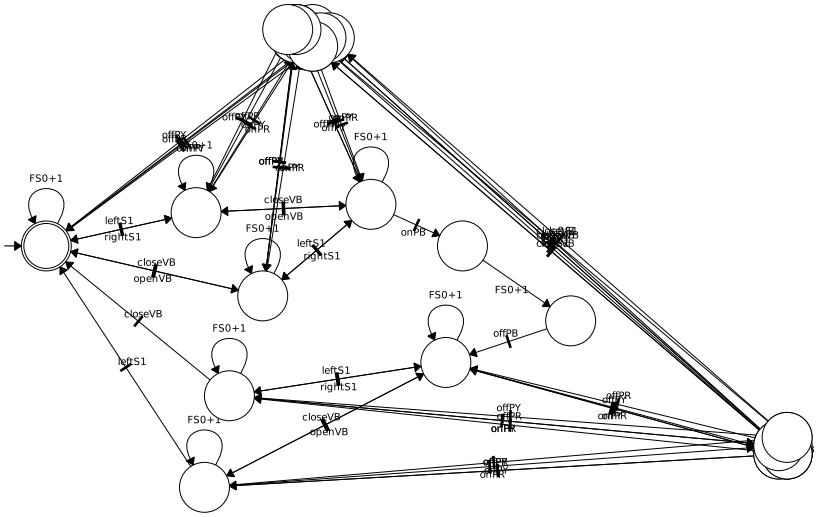

Figure 19: Optimal controller for G_1

The synthesised supervisors are 40 to 400 times smaller than the one that would have been computed for the whole system, as displayed in Table 4.

	Specs	Sup_{plant}	$Sup_{G_i}, i \in [1..3]$	Sup_{G_4}	Sup_{G_k}
♯ states	6	9261	65	63	11
♯ events	27	27	15	15	11
♯ transitions	96	85750	234	226	18

Table 3: Sizes of the supervisor generators

It is interesting to note that even though G_4 is more complex than the other pumping operations (see Table 2), its supervisors is less complex than the ones synthesised for the other operations. We have not found any reason for that yet.

A conjecture about the conditions for the framework to be applicable is that the language of the supervisor of the coordinator be included in the ones for the coordinated sub-systems, that is

$$\forall i, SupC_k \subseteq P_k(SupC_{i \cup k}). \tag{1}$$

It is indeed the case with our models and this could solve the failure to satisfy the OCC and observability properties. But, this workaround implies to first try and compute the supervisors, even though some properties are missing, in the hope that it would be fine in the end, which is not really convenient.

5.4 Concluding Remarks

What we have presented so far seems to be the "best" solution to us. But before getting something working on, we had taken several choices that appeared not to lead to a correct modelling. In order to help other modelers and to share our knowledge, we will come back here on such "false good ideas"...

Unlike in the work of Subsection 4.2, we decided to explicitly take the events of the pumps into account. Indeed, we needed *representative* events to consider the actual pumping process. We started using signals, but this was artificial and it added unnecessary complexity, in terms of extra events, states and transitions.

We have also tried to assign a colour to the counting events of a flow sensor. But, apart from the fact that this is not how it behaves in reality, if the coordinator becomes empty (the counting events not being shared any more), then the composition of the local supervisors becomes too big to be used (because no synchronisation can actually operate).

Moreover, for the model of the cleaning operation depicted in Figure 13, it was indeed possible not to add the transitions leading back to a previous state after state 10 is reached. This would lead to an observable model and some different figures in the synthesis of the supervisor, as displayed in bold characters in Table 4. It appears that some models are less complex, but complexity should not be the only criterion. Indeed, the models would be less close to reality and less consistent in our opinion.

	Whole plant	G_4	G_k	Specs	Sup_{plant}	Sup_{G_4}	Sup_{G_k}
♯ states	**12393**	17	135	6	**7889**	**59**	11
♯ events	27	9	11	27	27	15	11
♯ transitions	**106364**	**60**	437	96	**71001**	**207**	18

Table 4: Different sizes for an observable cleaning operation

5.5 Suggestions for Extension of Work

In order to control the complete scale model, we could investigate the following points:

- Add more operations, and the related parts and specifications.
- Model the system thanks to Petri nets, so that the synchronisations/parallelisms appear clearly, and then decompose the system according to the synchronisations.
- Think about an initialising process, that would clean everything first, in case of previous emergency breakdown.
- Work with unobservable events, e.g. clogs in the pipes.

Below are some pending questions that should be adressed in the next stages of this study.

1. How to cope with the *required* language and not only the *admissible* one (as expressed in [Cassandras and Lafortune, 2008])? The actual goal of the system is to eventually deliver orders. So we will also have to effectively tell the system what to do. In addition, does it make sense to hierarchically group and control the different coordinators? (see the discussions in Section 5.2)

2. How to get the sub-systems (i.e. the operations) from the actual models of the physical parts? Isn't related somehow to the specifications? (see the discussion in Sections 5.2, and 3.1)

3. Could it be possible to infer the so-called post-condition of relation (1), without having to compute everything for nothing?

4. Is there really a sensor that detects when some liquid in a vessel is below a certain level? (as stated in [Roberts, 2008, Section F.3, page 92])

References

[Behrmann et al., 2004] Behrmann, G., David, A., and Larsen, K. G. (2004). A Tutorial on UPPAAL. `www.uppaal.com`.

[Cassandras and Lafortune, 2008] Cassandras, C. G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, 2nd edition.

[Feng and Wonham, 2010] Feng, L. and Wonham, W. M. (2010). On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems*, 20(1):63 – 102.

[Hamer, 2007] Hamer, J. (2007). Model Based Engineering of a Paint Factory with Supervisory Control Theory. Master's thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands. Report SE 420492, System Engineering Group.

[Hoefnagels, 2008] Hoefnagels, N. (2008). Supervisory machine control based on State Tree Structures for the paint factory model. Master's thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands. Report SE 420519, System Engineering Group.

[Komenda et al., 2010] Komenda, J., Masopust, T., and van Schuppen, J. H. (2010). Synthesis of Safe Sublanguages satisfying Global Specification using Coordination Scheme for Discrete-Event Systems. In *Proceedings of the 10th International Workshop on Discrete Event Systems, WODES'10*, pages 436 – 441, Berlin. Technische Universität Berlin.

[Komenda et al., 2011] Komenda, J., Masopust, T., and van Schuppen, J. H. (2011). Coordinated Control of Discrete Event Systems with Nonprefix-Closed Languages. In *IFAC World Congress*, Milan, Italy. To be published.

[Ma and Wonham, 2005] Ma, C. and Wonham, W. M. (2005). *Nonblocking Supervisory Control of State Tree Structures*, volume 317 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, Berlin.

[Moor, 2011] Moor, T. (2011). DESTool. `www.rt.eei.uni-erlangen.de/FGdes/destool/index.html`. Retrieved on 18 July 2011.

[Moor et al., 2008] Moor, T., Schmidt, K., and Perk, S. (2008). libFAUDES – An open source C++ library for discrete event systems. In *Proceedings of the 9th International Workshop on Discrete Event Systems, WODES'08*, Gothenburg, Sweden.

[Ramadge and Wonham, 1989] Ramadge, P. J. and Wonham, W. M. (1989). The Control of Discrete Event Systems. *Proceedings of the IEEE*, 77(1):81 – 98.

[Ricker and Caillaud, 2007] Ricker, S. L. and Caillaud, B. (2007). Mind the Gap: Expanding Communication Options in Decentralized Discrete-Event Control. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 5924 – 5929, New Orleans, Louisiana, USA.

[Roberts, 2008] Roberts, R. J. Y. (2008). Synthesis of supervisory control based on logic specifications. Master's thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands. Report SE 420525, System Engineering Group.

[Schmidt and Breindl, 2008] Schmidt, K. and Breindl, C. (2008). On Maximal Permissiveness of Hierarchical and Modular Supervisory Control Approaches for Discrete Event Systems. In *Proceedings of the 9th International Workshop on Discrete Event Systems, WODES'08*, pages 462 – 467, Gothenburg, Sweden.

[Triepels, 2006] Triepels, J. (2006). Model Based Engineering of a paint factory with UPPAAL. Master's thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands. Report SE 420490, System Engineering Group.

[Tronci, 1998] Tronci, E. (1998). Automatic synthesis of controllers from formal specifications. In *Proceedings of the 2nd IEEE International Conference on Formal Engineering Methods*, Brisbane, Australia.

[van den Bremer, 2005] van den Bremer, W. A. P. (2005). Design of a Supervisory Controller for the SE-Paint Factory. Bachelor's Thesis, Technische Universiteit Eindhoven, Eindhoven, Netherlands. BEP report SE 420429, System Engineering Group.

[van Rooy, 2007] van Rooy, H. W. A. M. (2007). The Paint Factory Reference Manual. Technical Report SE No Number, Technische Universiteit Eindhoven, Eindhoven, Netherlands.

[Wong and Wonham, 1996] Wong, K. C. and Wonham, W. M. (1996). Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 6(3):241 – 273.

[Wonham, 2005] Wonham, W. M. (2005). Lecture notes on control of discrete-event systems. University of Toronto, Department ECE. Lecture notes.