Using Hybrid Automata to Support Human Factors Analysis in a Critical System

G. Doherty (g.doherty@r1.ac.uk)  
Rutherford Appleton Laboratory, United Kingdom

M. Massink (m.massink@cnrce.cnr.it) and G. Faconti  
(g.faconti@cnrce.cnr.it)  
CNR - Istituto CNUCE, Pisa, Italy

Abstract. A characteristic that many emerging technologies and interaction techniques have in common is a shift towards tighter coupling between human and computer. In addition to traditional discrete interaction, more continuous interaction techniques, such as gesture recognition, haptic feedback and animation, play an increasingly important role. Additionally, many supervisory control systems (such as flight deck systems) already have a strong continuous element. The complexity of these systems and the need for rigorous analysis of the human factors involved in their operation leads us to examine formal and possibly automated support for their analysis. The fact that these systems have important temporal aspects and potentially involve continuous variables, besides discrete events, motivates the application of hybrid systems modelling, which has the expressive power to encompass these issues. Essentially, we are concerned with human-factors related questions whose answers are dependent on interactions between the user and a complex, dynamic system.

In this paper we explore the use of hybrid automata, a formalism for hybrid systems, for the specification and analysis of interactive systems. To illustrate the approach we apply it to the analysis of an existing flight deck instrument for monitoring and controlling the hydraulics subsystem.

Keywords: Hybrid Automata, Human Factors, Critical Systems

1. Introduction

The distinguishing feature of the modelling and specification of interactive systems is the need to accommodate the user; for example, to formalise and analyse user requirements, and to conduct usability reasoning. The environment can also play a significant role, as it can impose constraints on both the system and the user, and the communication paths between them.

In existing approaches (see for example the review in [12]), interaction between user and system is assumed to be of a discrete and sequential nature. Such a view may be inadequate where interaction between user, system and environment contains continuous as well as discrete elements. The ‘continuous’ aspect of the system can take many forms, including, but not restricted to, continuous input and output devices, such as tracking systems to interpret human gesture, video, audio, and haptic devices. Certain types of interactive system have always had a continuous element - for example, many forms of supervisory control systems, such as flight deck systems and medical monitoring systems. Additionally, many emerging technologies, such as augmented reality, support
richer and more continuous interaction with the user, and hence applications using such techniques can also be viewed as hybrid systems [21].

If the models we build of such systems are to support reasoning about issues of usability and user requirements, then building a model of system behaviour is not enough - we must also have some means of referring to both the user and the environment. It has been proposed that usability issues can be better understood in terms of the conjoint behaviour of system and user, and that syntetic models [8, 9], which combine a formal system model with a representation of human cognition, support such an approach. In this paper, we do not consider the modelling of human cognition but rather apply models of user input and observation of system output, which afford the possibility of reasoning about user behaviour and inference. These models can take the form of constraints imposed by the limitations of the user or of the environment, or they may take the form of more explicit models of relevant aspects of the user or environment.

Automata provide a relatively simple formalism, with a convenient graphical representation, for specifying the behaviour of systems. Basically, an automaton consists of a number of locations, and a number of transitions which link these locations. System specifications typically involve several automata, synchronised on certain transitions. They include variables on which location invariants and transition guards are based. Recently, a number of interesting variants of automata, including timed and hybrid automata, have been developed that allow the specification of processes that evolve in a continuous rather than a discrete way. Timed automata include real valued clock variables which increase at a uniform rate [17, 3]. Hybrid automata [13], on which we focus in this paper, include analog variables which can change at arbitrary rates (both positive and negative). The continuous change of real valued variables in hybrid automata is specified by sets of differential equations, as is common practice in for example physics. The automata based formalisms not only provide a specification language with a graphical interpretation, but also allow for automatic verification of properties by means of reachability analysis provided by several model checking tools [3, 17, 14].

In this paper we focus on flight deck instrumentation concerning the hydraulics subsystem of an aircraft. This is based on a case study originally presented by Fields and Merriam [10], which involves analysis of the support the instrument provides to the user for the diagnosis of system failures (including issues of representation).

In the sections following we introduce the case study, and the HyTech formalism. In section 4 we examine the issue of reasoning about interactive systems with HyTech, and in sections 5 and 6 we proceed with the specification and analysis of the case study. We conclude with some discussion and issues for future work.
2. Case study on aircraft hydraulics

The case study we analyse in this paper is taken from the domain of aircraft hydraulics, and is based on the description by Fields and Merriam in [10]. The hydraulics system is vital to the safe operation of the aircraft as it is the means by which inputs from the pilot or autopilot are conveyed to the control surfaces (e.g. rudder and ailerons). Movement of the surfaces is achieved by servodynes which rely on hydraulic fluid. This fluid is supplied from reservoirs, and a number of valves determine which reservoir supplies a given servodyne.

In [10] a generic model of hydraulic systems is first presented, and a reduced (but realistic) version used as the basis of analysis. We base our treatment on the reduced model.

In the reduced model, only two control surfaces - rudder and aileron - are included. The operation of these surfaces is powered by servodynes; each surface has a primary and secondary servodyne. Hydraulic fluid is supplied to the servodynes from two reservoirs, the primary servodynes of each being connected to the blue reservoir, the secondary to the green reservoir. The valves between the reservoirs and servodynes are such that each surface is connected to only one tank at a time (see Fig. 1).

![Hydraulic System Diagram](image)

*Figure 1. Hydraulic System, from [10]*

Two goals are identified in relation to operating the system; firstly to preserve the safety of the system (maintaining hydraulic power to the control surfaces), and secondly to discover the cause of a problem. The user’s actions are hence closely tied with the process of reasoning about the possible faults in the system. As noted in [10], this type of activity is typical in process control settings, and hence we see the case study as representative of a class of applications. For consistency with the literature on diagnosis and repair, we use below the terminology (shown in italics) developed by Friedrich, Gottlob and Nejdl [11].

In order to be able to repair a system, the operator must have in mind what the purpose of the system is. In general the *system purpose* could be defined as “All components have to work correctly”. The purpose can be formulated in more detail for specific systems and be expressed in a logical sentence including temporal properties. In the case study the *system purpose* is the minimisation of leaks by the human operator, the motivation being that a control surface connected to an empty reservoir cannot be operated and hence
is a serious hazard. In the model, each reservoir can leak independently, as can each of the servodynes. Failures can occur in any combination of reservoirs and servodynes, although fluid is only lost through a leaky servoyne while the valve connecting it to one of the reservoirs is open. In reality there are many other components involved in the system, and many more possible types of system failure, but the model described here suffices to illustrate our approach.

In order to perform a diagnosis and repair process an operator needs to be able to influence the system state. The possibilities he has depend on the available set of diagnosis and repair actions. User operation of the system is by means of two switches, one for each control surface, which can be set by the user to either blue or green (see Fig. 2(a)). So in this case setting the switches are the diagnosis and repair actions.

For the diagnosis an operator must be able to gain knowledge about the system state. This is done via observations. Note that observations differ from actions in the sense that observations do not influence the system state whereas actions do. The level of fluid in each reservoir is presented to the user by means of a pie-chart like display (see Fig. 2(b)), where the loss of fluid can be observed by the pilot as a gradual decrease of the ‘filled’ portion of arc in the display (shaded grey in our diagram). The observation of the fluid level indicators are the only observations available to the pilot. Note that this description is based on the existing interface in the real aircraft rather than the alternative designs also discussed in [10].

Figure 2. Controls and status display, adapted from [10]

The case study is interesting since it includes both real-time issues surrounding the diagnosis and correction of leaks. There are analog (continuous) variables representing the fluid levels in the tanks (which can change continuously at various rates), and the continuous representation of those levels by the level indicators on the flight deck. These continuous aspects are combined with discrete controls operated by the pilot.

To illustrate how the diagnosis of leaks is tied to the observation of tank levels in the different switch configurations, consider the following scenario, where the rudder primary (R1) and aileron secondary (A2) servodynes are leaking. The sequence of switch settings, observations, and the ‘ideal’ user understanding of the system involved in the diagnosis are illustrated in ta-
ble I. Initially the user observes a decrease (L) in the quantity of the blue reservoir (BR) and no decrease (N) in the level of the green reservoir (GR). Setting both switches to green, a decrease in the level of the green reservoir is observed, leading to the conclusion that neither blue nor green reservoir are leaking. Setting the rudder back to blue leads to loss from both reservoirs, leading to the conclusion that both the rudder primary and aileron secondary servodynes are leaking, and toggling both switches leads the user to conclude that neither rudder secondary nor aileron primary are leaking, completing the users knowledge of the state of the system. Note that the results of earlier observations influence the conclusions the pilot can reach at each step of the sequence. That is, the reasoning process is history dependent.

Table I. Observations and inferences

<table>
<thead>
<tr>
<th>Step</th>
<th>Control State</th>
<th>Observation</th>
<th>Possible causes of leak</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rudder</td>
<td>Aileron</td>
<td>Blue</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>B</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>G</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>G</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>G</td>
<td>B</td>
<td>N</td>
</tr>
</tbody>
</table>

In the following sections we develop a formal model of the hydraulic sub-system interface and some aspects of the behaviour of a pilot using hybrid automata and an associated analysis tool (Hytech). The next two sections give a short introduction to hybrid automata and a general discussion on what kind of usability related analysis could be supported by them.

3. Hybrid automata

In this section we give an informal overview of the specification language for hybrid automata HyTech and the analysis capabilities of the associated tools. For details on the formal semantics, the use of the tools and for further references to articles on the theory underlying HyTech we refer to [14, 13].

3.1. HyTech specification language

A HyTech specification consists of a number of automata, a set of variable declarations and a number of analysis commands. Each automaton is named, and contains some initial conditions, a set of locations (with invariants), one of which is an initial state, a set of transitions (which may include guards and variable assignments), some of which may be labelled as urgent. Transitions may be synchronised between any number of automata (multi-part synchronisation), and variables are global, i.e. they may be referenced from
any automaton. Variable types are distinguished by the generality of the rate conditions which may be assigned to them.

Graphical manipulation of HyTech specifications is possible via the AutoGraph tool, and a converter which maps state transition diagrams in AutoGraph to the HyTech textual specification language [16].

3.2. HyTech model checking

Besides a specification language based on automata HyTech provides a tool for reachability analysis. With the tool a subclass of hybrid automata specifications can be automatically analysed. This subclass is the set of linear hybrid automata. These are hybrid automata in which all invariants and conditions are expressed as (finite) conjunctions of linear predicates. A linear predicate is an (in)equality between linear terms such as for example:

\[ \frac{1}{2}x + 5 \leq y \]

The coefficients in the predicates must be rationals. HyTech can compute the set of states of the parallel composition of linear hybrid automata. Given an initial region (a subset of the state space is called a region)\(^1\), it can compute the set of states that is reachable from the initial region by forward reachability analysis. Conversely, given a final set of states, it can compute the set of states from which the final region can be reached by using backward reachability analysis. As a side effect of reachability analysis a sequence of delay and transition steps can be generated that shows an example of how one set of states can be reached starting from another. This can be extremely helpful for what we could call ‘high-level debugging’ of a formal model of a system.

A third analysis feature is the use of models that contain parameters. HyTech can be used to synthesize automatically precise constraints on these parameters which are necessary to satisfy certain properties. In later sections we show how these kinds of analysis can be useful for the analysis of human computer interfaces.

4. Reasoning about interactive systems with HyTech

The approach we take is to model not only the system but also aspects of the user and potentially the environment by means of various automata. In this section we consider the types of interactive system properties we would like to analyse, the construction of the models for user, system and environment, and the ways in which we may link the models.

\(^1\) Note that in the context of HyTech specifications, the term region is used to refer to an arbitrary subset of the state space; in the underlying theory concerning model checking of automata specifications, this term has a more specific meaning.
4.1. Properties

There are many benefits associated with the construction of a formal specification, but the one we focus on here is the possibility of formally verifying whether certain properties hold on the combined specification of system, user, and environment, in this case via the capabilities of the HyTech model-checker. The basic capabilities of HyTech are based on reachability analysis. There are a number of ways in which this can be used in the context of human factors analysis:

**possible behaviours** One form of analysis is to examine whether it is possible for the system to reach desirable states or sets of states (for example those representing the user’s *goals*), or whether it is impossible to reach undesirable states (for example those which would compromise the safety of the system). This class of properties can be analysed through simple reachability analysis.

**impact of errors** From common types of human error (for example errors of omission, commission and transposition [15]), we can examine the impact, i.e. *“the effect that an action or sequence of actions has on the safe and successful operation of the system”*[4] that such errors might have. Encoding such a sequence in HyTech is possible, although less straightforward than simple reachability properties.

**state/display conformance** Where we have automata representing the presentation of the system state, or observation of this by the user, we can check conformance between the automata representing the system state, and those representing the presentation and/or observation. See for example Dix [6] for a discussion of various forms of state-display conformance.

**performance related properties** Since we can derive limits on timed and analog variables such that certain states are reachable for example, it is possible to examine performance related issues such as latency (see [1] for a treatment of latency related issues in a lip synchronisation protocol modelled in UPPAAL, a specification language for timed automata). Data on latencies can be compared to human factors and ergonomics data on human response times and perception thresholds, to ascertain whether system performance is compatible with the user’s capabilities.

A review of the use of machine assisted verification (including model checking, but not real-time or hybrid techniques) in the analysis of interactive systems can be found in Campos and Harrison [2]. Rushby [20] describes some preliminary work in using a model checker to look for inconsistencies between a user’s model of system operation and actual system behaviour which lead to automation surprises in an avionics case study [19].
4.2. Modelling user, system and environment

While automata formalisms are quite straightforward to apply to the specification of interactive systems, there are a number of levels of abstraction at which we can apply them. Specifically, we must decide what subset of the state space is covered by each location. We can opt for very low level models with a one-to-one correspondence between system states and automata locations, or for higher level models where the locations might represent system modes.

In this paper, we consider user models which consist of models for input, and possibly observation. An input model could simply capture the possible sequences of operations a user can invoke, or it can be more focussed towards a specific task, or even a strategy for achieving some goal, as might be encoded in a standard operating procedure. Different kinds of observation model may also be defined, with observation times perhaps related to rates of change of interface elements or the workload of the user at a certain point in a procedure. Not all relevant aspects of the user need to be included in the automaton itself, but may also be expressed in the analysis language. For example, a region definition may correspond to observation of the state of the system by the user. A detailed example of this is given in section 6.3.

With regard to the environment, we obviously wish to minimise the details included in the specification. Where some external process must be modelled, the variables involved are likely to be more important than the locations. Non-determinism as a means of abstraction can also play an important role.

4.3. Composing user, system and environment models

The potential for applying these specifications to usability reasoning derives from the way in which the user and system automata execute together, and hence it is worth considering how the models are composed. There are a number of ways in which automata representing user, system and environment can be linked:

- synchronisation transitions, including urgent transitions. Where transitions of different automata are synchronised (they have the same label), then they must occur simultaneously in all automata which contain the label. Thus by means of a synchronised transition, we can require that both system and user models make a transition at the same time.

- shared variables - use of a variable from another part of the model in a guard or invariant corresponds to an observation of some form. Hence where the user model references the value of a system variable, this corresponds to the user observing the variable - either in the presentation of the system, or directly from the environment.

When we consider the significance of such relationships between user and system models to usability reasoning, we find that there are a number of interesting semantic issues. Firstly, what does it mean when one automaton “waits” for the other? This seems to be an issue of initiative, and so whenever
we define synchronised transitions between system and user, we should consider the significance of the ‘waiting’ behaviour in terms of the interaction. Secondly, there can be a distinction between the occurrence of an event, for example in the system model, and perception of that event, which might correspond to a transition in the user model. Consider for example a form of ‘polling’ behaviour by a user who periodically checks some portion of the system display, rather than continuously monitoring it (an assumption which would be unreasonable in many application domains).

5. Hydraulics system specification

In this section, we describe a formal specification of the behavioural aspects of the hydraulics system interface introduced in section 2. We construct automata to represent both the system and the user.

5.1. System model

The system model comprises four automata, namely:

- those that model the servodyne leakage and valve state, one for each control surface.
- those that model the reservoirs, one for each reservoir.

We define two valve automata, ValveR for the rudder and ValveA for the aileron. The valve for each control surface can be set to either the blue or green reservoir, and correspondingly the primary or secondary servodyne. These transitions are given synchronisation labels sab, signifying ‘set aileron to blue’, srg signifying ‘set rudder to green’, and so on. Transitions are atomic and do not cost time. The locations with a label containing a B as the second letter are those where the blue reservoir has been selected, and those containing a G where the green reservoir has been selected.

Furthermore, each servodyne can leak independently, yielding a total of eight locations for each automaton. From a given situation, more leaks can occur, and so we have transitions (the unlabelled ones in Fig. 3) denoting the occurrence of (additional) leaks. These leaks are indicated in the location names by the last two letters, each of which can be either N - notleaking or L - leaking, the second last letter representing the state of the primary (blue) servodyne and the last letter that of the secondary (green) servodyne.

For each valve of the aileron an analog variable is introduced that models the amount, and also rate of leakage of the connected servodyne. For the servodynes of the aileron variables ba and ga model the amount of fluid leaking from the primary and the secondary aileron servodyne respectively, and their first time derivatives dBA and dGA (automatically provided by the tool) the respective rate of leakage. Note that when the valve connecting a leaky servodyne is closed the servodyne does not loose liquid. So, for example, in the location labelled by AGLL, standing for the aileron switched to the green
(secondary) servodyne where both servodynes are leaking, only the derivative regarding the leak in the secondary servodyne is non zero \((\text{dba}=0, \text{dga}=1)\).

![Valve A Automaton Diagram]

**Figure 3. Aileron Valve Automaton**

The initial state is indicated by a double circle and is a state where no leaks are present and the aileron is connected to the blue reservoir. A similar automaton is constructed for the valves and servodynes of the rudder with variables \(br\) and \(gr\), see Fig. 6.

The reservoir automaton (figure 4), includes three locations indicating the status of the reservoir which can be empty, leaking, and not leaking but not empty. For the green reservoir these have labels \(\text{emptyG}, \text{GRL}\) and \(\text{GRNL}\) respectively. Associated with the locations we have both invariants on the continuous variable \(g\) (the level of fluid in the reservoir) and conditions on the rate variable \(dg\), the rate at which fluid is lost being the sum of that lost from the servodynes connected to the green reservoir \((-dgr-dga)\) and leakage from the reservoir itself \((-1\) in the condition for location \(\text{GRL}\)). When the reservoir is empty, both the level of fluid and the rate of leakage are zero. In the initial state it is assumed that the reservoir contains an amount (10 units) of fluid as indicated by the starting condition \(\text{start}: g=10\) in the automaton box.

A similar automaton is constructed for the blue reservoir, see Fig. 6.

![Green Reservoir Automaton Diagram]

**Figure 4. Reservoir Automaton**
5.2. User model

There are many ways to model the user, each taking particular aspects of user behaviour into account by modelling certain assumptions. In this paper we use a user model that only reflects the possible operations a user can perform with the two switches of the interface. In effect, this is the minimally constrained user model; we presume the user can perform any combination of diagnosis and repair actions which the interface allows. However, the model is still rich enough to allow the analysis of some interesting properties of the interface, and it can serve as a basis to which further assumptions, such as the time a pilot needs to observe a decrease in the fluid levels, can be added at a later stage.

We model user input by means of a single automaton, in which the user can perform actions to move between the four possible combinations of switch settings; that both rudder and aileron are set to blue RBAB, that rudder is set to green and aileron to blue RGAB and so on, as illustrated in Fig. 5. The transitions have synchronisation labels that establish synchronisation between user actions and the two valves. Each transition is guarded by a constraint on the level of fluid in the reservoir to which the user wants to connect a servodyne.

![User Automaton Diagram](image)

*Figure 5. User Input Automaton*

As can be seen in the above diagrams, both valve/servodyne automata and the user input automaton are synchronised on four events (namely srb, srg, sab, sag), which set the rudder and aileron valves to the blue or green settings. These are the *diagnosis and repair actions*. The observations that are possible are the values and the rate of change of the fluid level indicators, modelled by the variables g and b. Note that synchronisation of the user and valve automata on these actions indicates that these transitions may occur, but not necessarily as soon as they are enabled (from the semantics of HyTech). This way a number of user actions are possible, rather than encoding a specific user behaviour. The complete specification (without an analysis section) is shown in Fig. 6.
5.2.1. Region-based observation model
The investigation of specific user behaviour can be analysed by modelling user observation and consequent reaction to the observations. User observation of the presentation of the system status, i.e. the change in the fluid level indicators, can be represented by automata. However for the purpose of most of the analyses discussed in this paper it was found to be simpler and more convenient to model observation as a set of regions defined in the analysis component of the HyTech specification. Each region corresponds to a user observationally-equivalent class of states. Details of this approach are discussed in the next section.

5.2.2. Hybrid observation model
Although most of the analysis is conducted with the observation model above, we could extend the user model to include delays between user observation of some system event, and input of some response by the user. As mentioned in section 4.1 above, these response times could be based on empirical data on human performance, allowing us to analyse the system taking into account the limitations of both the user and the technology. For example, let us assume that the time before a switching action is composed of the time it takes to observe the fluid level indicators and some time to think about the next step to take. This hybrid user model is shown in figure 7 (a). For the purposes of illustration, one observation and switching action have been modelled. The user starts by observing the level indicators. The actual time during which she looks depends on the rate of leakage, modelled by $\text{dlook} = \text{d}g + \text{db}$. When the rate is high, the indicators move faster and the pilot can observe more quickly that a leak is present (the variable $\text{look}$ reaches the value $0$ earlier and so the transition becomes enabled earlier). When a leak has been observed,
the pilot will think about what she has observed for a short while, modelled by the location Thinking. Here we assumed a thinking time of one fixed time unit. When the time to think has passed the pilot performs the proper action.

![Diagram](image)

*Figure 7. Hybrid observation models dependent on rate of leakage*

Alternatively we could *add* an automaton incorporating these real-time aspects to the existing user model. Such an automaton is shown in figure 7 (b). The analysis of both extensions are discussed in section 6.2.1.

### 5.3. Additional Assumptions

While we do not use them in the current case study, contextual assumptions and environmental factors can also be specified as automata. In [10] for example, it is assumed that no new leaks occur during the diagnosis. If we label all ‘new leak’ transitions with a separate synchronisation label for each automaton (eg. a leak for the leaks in the aileron valve), and construct an automaton (Assumption in Fig. 8) with reflexive transitions in the initial state which synchronises on these labels, and a transition to another state which does not synchronise on these labels (NoMoreLeaks in the figure), then by adding the conjunct loc[Assumption] = NoMoreLeaks to our analysis regions, we preclude the occurrence of new leaks during the diagnosis.

![Diagram](image)

*Figure 8. Assumption automaton*
At a practical level, assumptions in this form can be easily added and removed which makes it easier to perform analysis of the complete system under various assumptions.

6. Analysis

Having completed our specification of the system, we proceed with a number of analyses. The examples in the current section illustrate the approach and the possible kinds of analyses that can be performed with the model checker HyTech to support human factors analysis. The analyses discussed in this section use the region-based observation model with the exception of those discussed in section 6.2.1.

In addition to looking at possible behaviours and conducting performance related analysis, we illustrate an original form of task-driven analysis which concerns the possible inferences to be made by the user in the diagnosis of system faults. We show how the expressiveness of the analysis language of HyTech can be exploited to model the observations of the liquid indicators by the pilot when performing the diagnosis activity. This approach makes it possible to evaluate and compare the efficiency and efficacy of different diagnosis strategies.

Our other aim in this section is to illustrate how real-time and continuous elements can be included in analyses with a human factors focus. In particular we show how formal modelling with hybrid automata and associated model checking tools can support analyses driven by concepts such as user tasks and user inference, which are external to the system specification. This is not intended as an attempt to cover the many ways in which this is possible, but rather as a first exploration of an interesting area of analysis. This motivates further work, and in particular more detailed user models.

6.1. Possible behaviours

Questions regarding the possible behaviours the user can generate can be expressed via reachability conditions, and checked in a straightforward fashion. Sample traces, which reach a target region from an initial region, including timing information, can be constructed automatically from the model by HyTech. For example, let’s suppose the valves are switched to blue and only the primary servodyne of the rudder and the secondary servodyne of the aileron are leaking. This situation is defined as the region variable init_reg.

\[
\text{init\_reg} := \text{loc\[User\]}=\text{RBAB} \land \\
\text{loc\[ValveR\]}=\text{RBLN} \land \\
\text{loc\[ValveA\]}=\text{ABNL} \land \\
\text{loc\[GreenRes\]}=\text{GRNL} \land \\
\text{loc\[BlueRes\]}=\text{BRNL} \land \\
g=5 \land b=5 \land t=0;
\]
The observation of a decrease in the blue and green reservoir fluid quantity can be expressed as two regions. Leaking of ‘green’ fluid can be observed when the rudder is switched to green and the secondary servodyne is leaking, or when the aileron is switched to green and the secondary servodyne is leaking or when the green reservoir is leaking. A similar region can be defined for the observation of a decrease in the ‘blue’ fluid. In HyTech these regions are defined in the following way where $|$ denotes the logical ‘or’:

\[
green\_leak := (\text{loc[ValveR]}=\text{RGNL} \mid \text{loc[ValveR]}=\text{RGLL} \mid \\
\text{loc[ValveA]}=\text{AGNL} \mid \text{loc[ValveA]}=\text{AGLL} \mid \\
\text{loc[GreenRes]}=\text{GRL});
\]

\[
blue\_leak := (\text{loc[ValveR]}=\text{RBLL} \mid \text{loc[ValveR]}=\text{RBNL} \mid \\
\text{loc[ValveA]}=\text{ABLL} \mid \text{loc[ValveA]}=\text{ABNL} \mid \\
\text{loc[BlueRes]}=\text{BRL});
\]

### 6.1.1. Can we find a fix for a situation?
In the context of our case study, the first simple analysis is to let HyTech find out whether a fix can be reached starting from a particular ‘leaky’ situation\(^2\). In other words, can the system purpose be restored given a certain situation? We can let HyTech compute the set of states reachable from the initial region intersected with those states in which no leaks occur (defined as \text{fin.reg}).

\[
\text{fin.reg} := \neg\text{blue\_leak} \land \neg\text{green\_leak} \land g>0 \land b>0;
\]

\[
\text{reached} := \text{reach forward from init.reg endreach};
\]

If the intersection is not empty, we can let HyTech print a (shortest) trace from the leaking situation to a situation in which the problem is fixed. The result is shown below:

**Location:** \text{RGNL.RGRL.RGAB.BRNL.ABNL}

\[
b \leq 5 \quad \& \quad g + t \geq 5 \quad \& \quad b + t \geq 5 \quad \& \quad g \leq 5 \quad \& \quad 0 < g \quad \& \quad 0 < b
\]

\---- Generating trace to specified target region \----

**Time:** 0.00

**Location:** \text{RBNL.RGRL.RBGB.BRNL.ABNL}

\[
g = 5 \quad \& \quad gr = 0 \quad \& \quad ga = 0 \quad \& \quad b = 5 \quad \& \quad br = 0
\]

\[
\& \quad ba = 0 \quad \& \quad t = 0
\]

**VIA:** srg

**Time:** 0.00

**Location:** \text{RGNL.RGRL.RGAB.BRNL.ABNL}

\(\text{\footnotesize \footnote{\text{2} Here we define a fix in terms of stopping all leaks; in fact we could use a weaker definition in which no control surface is connected to an empty reservoir. We have carried out this analysis, which leads to more states in the solution space, but apply the simpler definition here for the purpose of illustration.}}\)

\[\text{}\]

\[\text{}\]

\[\text{}\]
\[ g = 5 \quad \& \quad gr = 0 \quad \& \quad ga = 0 \quad \& \quad b = 5 \quad \& \quad br = 0 \\
\& \quad ba = 0 \quad \& \quad t = 0 \]

The first three lines describe the set of states characterised by the intersection of \texttt{fin.reg} and \texttt{reached}. The first line consists of a location vector that lists the location of each automaton in the specification in the situation in which a fix for the leaking is realised. The list gives only the names of the automata locations. They belong respectively to the automata \texttt{ValveR}, \texttt{GreenRes}, \texttt{User}, \texttt{BlueRes} and \texttt{ValveA}. The order in which the location vector is reported is the same in each following analysis in this section. The next two lines give the conditions on the variables that have to be satisfied in that case.

The second part of the result concerns the trace starting from the initial region to the final region. In this case the shortest trace to a fix consists of just one action, namely switching the rudder to ‘green’, achieved by transition \texttt{src}. Since in this model we have not specified that switching activity performed by the pilot costs time, the transition can be performed in zero time units. Results on the timed extensions of the user model are discussed in the next section.

Further details on these constraints should first be investigated in order to keep the model as realistic as possible.

In this case study there are ‘leaky’ situations that obviously cannot be fixed by the pilot by means of the switches. One such situation is when both reservoirs are leaking. Forward reachability analysis of such a situation gives us indeed the answer that a fix is not possible.

6.1.2. \textit{For which situations can we find a fix?}

A more interesting question is for which ‘leaky’ situations a fix can be found and whether HyTech can provide us with an exact characterisation of all those situations. This question can be answered by using a backward reachability analysis, starting from a situation in which no leaks occur, i.e. the above defined region \texttt{fin.reg}. This leads to a set of 36 regions. Three representative ones are shown below.

\begin{align*}
\text{Location: } \texttt{RGNL.GRNL.RGAG.BRNLL.AGNL} \\
t &= 0 \quad \& \quad 0 < g \quad \& \quad 0 < b
\end{align*}

\begin{align*}
\text{Location: } \texttt{RGNL.GRNL.RGAG.BRNLL.AGLN} \\
t &= 0 \quad \& \quad 0 < g \quad \& \quad 0 < b
\end{align*}

\begin{align*}
\text{Location: } \texttt{RGNL.GRNL.RGAG.BRNLL.AGNN} \\
t &= 0 \quad \& \quad 0 < g \quad \& \quad 0 < b
\end{align*}

Inspection of all the reported states shows that the automata that model the reservoirs are always in the non-leaking location (\texttt{GRNL} and \texttt{BRNL}). Moreover, in none of the reported regions both valves of the rudder or both valves of the aileron are leaking at the same time. This gives us indeed exactly the conditions under which the pilot is able to find a fix.
6.2. PERFORMANCE RELATED PROPERTIES

Using HyTech, we can derive limits on the variables, including fluid levels and rates of leakage. To do this, we use a parameter variable and get the system to find the values of the parameters for which a solution can be found (some final region reached). For example, consider an analysis where we declare the initial levels of fluid in the tanks to be given by a parameter alpha. We define the target region to be one where neither tank is empty and the clock t has reached 3. In Fig. 9 below we show the analysis for an initial region where both reservoirs and both rudder valves are leaking. The hide statement is a form of existential quantification, allowing us to solve against the variables of interest. When we run the analysis we obtain the following output, which

```
Analysis
var init_reg, fin_reg_reached, fail_reg : region;
init_reg := loc[User]=RBA & loc[Valve]=BLL &
  loc[Valve]=ABNN & loc[GreenRes]=GRL &
  loc[BlueRes]=BRL g = alpha & b = alpha & t=0;
fail_reg := b=0 | g=0;
fin_reg := t=3 & ~fail_reg;
reached := reach forward from init_reg endreach;
if empty(reached & fin_reg) then
  prints "Situation is not reachable";
else
  print omit all locations hide non_parameters in reached&fin_reg endhide;
endif;
```

*Figure 9. Analysis to derive constraints*

gives the range of values of the parameter and for which the final region is reachable.

**Composing automata ****

........Number of iterations required for reachability: 8

\[ 2\alpha > 9 \]

Examination of more detailed output shows that the traces requiring the minimal alpha value of 9/2 are those which involve switching the rudder valve from the primary to the secondary reservoir at t=1.5, hence for each tank, 3 units are lost from the reservoir leak, and 1.5 units from the rudder servodyne leaks.

The analysis above involved deriving a constraint on the analog variables of the system. Investigation of the traces of behaviour corresponding to the bounds of the system performance (i.e. minimal fluid loss) gives information regarding the user performance which corresponds to these bounds. We could also solve directly against clock variables related to user performance, for example to derive the maximum time which can elapse before some action must be taken by the user to preserve system safety.
6.2.1. *Hybrid observation model*

The hybrid model presented in section 5.2.2 allows us to illustrate the following time dependent behaviour of system and user. As in the previous performance analysis we define an initial region where the aileron valves are not leaking, but the rudder ones are. Furthermore both reservoirs are leaking. Initially both aileron and rudder servodynes are set to blue. The liquid in the reservoirs is a parameter. The final region is such that three time units have passed since the start, the reservoirs are not empty and the rudder valve is set to blue. Given this situation and the more complex behaviour of the pilot, we would like to know what the minimal amount of fluid needs to be to allow the pilot to fix the system. HyTech gives us the following result, where \( \text{alpha} \) again stands for the fluid level in the reservoirs:

*Composing automata ****

............Number of iterations required for reachability: 12

3\( \alpha \) > 14

This example clearly shows how real-time aspects of user behaviour could be made dependent on real-time behaviour of the system, something that is a useful tool in the analysis of interfaces. It also shows that these complicated dependencies can make it hard to reason about the model without tool support such as a model checker. The observation model of figure 7 (b) gives the same result as above, but allows analysis for different initial situations and is therefore more general. The price to pay is the larger state-space it generates and therefore more computational resources for the analysis.

6.3. *Diagnosis activity*

Besides finding a fix for a situation in which leaking of hydraulic fluid occurs, the goal of the pilot is to discover as soon as possible and as precisely as possible which components in the hydraulics system are leaking. As we have seen in the description of the case study in section 2 the pilot is supposed to follow a certain diagnosis strategy in which she sets the switches to different positions and observes the corresponding changes in the fluid level indicators. In this strategy, the order in which the different positions of the switches are examined is important because this influences the inferences about the status of the system components that the pilot can make in each step of the diagnosis. Moreover, the pilot has to remember the observations made in previous steps of the process, in order to reach a complete assessment of component failures at the end of the diagnosis.

A first analysis that seems useful is to check whether the diagnosis strategy proposed works for particular ‘leaky’ situations. This could be done in several ways. One straightforward way would be to model the diagnosis behaviour of the pilot as an automaton. Given the number of possible situations, user observations and inferences that can be made, this approach seems rather cumbersome and unlikely to scale up when more difficult cases would be analysed.
A more indirect, but simpler, way to reach the same goal is to model observation by the pilot and consequent actions via the reachability analysis language of HyTech. This is the approach we follow in this section, using the model of figure 6.

We start the analysis from a region that characterises the situation in which the user has switched the valves of the rudder and aileron to the blue reservoir (i.e. \( \text{loc[User]} = \text{RBAB} \)). In this situation the pilot observes a decrease in the blue reservoir fluid quantity and no decrease in the green reservoir fluid quantity.

Let us assume that the pilot follows the diagnosis strategy indicated in table I, going through the various settings from top to bottom. The first situation of the diagnosis can then be defined as:

\[
\text{sit1} := \text{loc[User]} = \text{RBAB} \land \text{blue_leak} \land \neg\text{green_leak} \land t=1 \land g>0 \land b>0 \land \neg\text{loc[GreenRes]}=\text{emptyG} \land \neg\text{loc[BlueRes]}=\text{emptyB};
\]

Starting from this situation we would be interested in the set of all logically possible worlds. A possible world for a relevant time interval and a given set of initial conditions, observations and actions includes the current state of all components and parameters in the system as well as the history and the future of the system. These can be obtained by using a forward reachability search starting from sit1.

\[
\text{s1} := \text{reach forward from sit1 endreach};
\]

We are now interested in those possible future worlds that are consistent with the second situation in which the user has switched both the rudder and the aileron to green and observes a decrease in the green fluid but not in the blue while the two reservoirs are still not empty. The second situation, assuming that the pilot needs 1 time unit to observe and interpret the level indicators\(^3\), can be characterised as:

\[
\text{sit2} := \text{loc[User]} = \text{RGAG} \land \neg\text{blue_leak} \land \text{green_leak} \land t=2 \land g>0 \land b>0;
\]

The set of possible worlds s1 that are consistent with sit2 can be obtained by taking their intersection. Forward reachability search, starting from this intersection, gives the set of possible worlds that are consistent with the actions and observations of the pilot in sit1 and sit2.

\[
\text{s2} := \text{reach forward from s1} \land \text{sit2 endreach};
\]

Region s2 in its turn can be intersected with the region describing the next situation the pilot may encounter when she switches the aileron back to

\(^3\) Whether or not this is a realistic assumption, it is important that it has been made explicit, and in any case suffices to illustrate the approach.
blue and observes that both the fluids stopped decreasing (sit3 below). Note that this step corresponds to going from step 2 to step 4 in the diagnosis steps reported in Table 1, skipping the third step listed there.

\[
sit3 := \text{loc}[\text{User}]=\text{RGAB} \land \neg \text{green\_leak} \land \neg \text{blue\_leak} \land t=3 \\
g>0 \land b>0;
\]

The resulting regions printed by HyTech of for example \(s1 \land sit2\) and \(s2 \land sit3\) give us an indication of the size and complexity of the resulting regions, and thus of the solution space the pilot must reason about. The resulting region of \(s2 \land sit3\), i.e. the possible worlds that are consistent with the actions and observations in the three situations, contains only one element:

**Location:** RGLN.GRLG.RGAB.BRNL.ABNL
\[
t = 3 \land 0 < g \land 0 < b
\]

This gives exactly one possibility for the locations of the automata as the result of the diagnosis followed by the pilot. This unique result indicates that the pilot should reach only one conclusion about the status of the components of the hydraulics system, assuming no errors in reasoning are made. The conclusion in this case is that the primary servodyne of the rudder is leaking, but the secondary is not (RGLN), the green reservoir is not leaking (GRLN), the blue reservoir is not leaking (BRNL) and the primary servodyne of the aileron is not leaking, but the secondary is (ABNL). This result corresponds to the combination of the derivations following steps 1, 2 and 4 in Table 1. Note that for this diagnosis we have not made any assumptions about the possibility that hydraulic components may start leaking while the pilot is performing the diagnosis.

The above analysis shows that in that particular diagnosis, with the described observations by the pilot, a precise and unique assessment of the status of the hydraulic components can be reached without excluding in advance that no component starts leaking during the diagnosis. It is also interesting to take a look at the region \(s1 \land sit2\) that gives the states that can be reached right after the second observation (step 2) by the pilot. It gives 21 possible combinations of locations. From the detailed output we can derive what the pilot could have correctly concluded about the status of the components after the second step in the diagnosis, namely that the blue reservoir is not leaking (BRNL in each possible location vector). But, for example, the pilot can no longer be sure that the green reservoir is not leaking, although he could conclude this correctly after the first observation. This is explained by the fact that the full result obtained from HyTech also takes into account that the green reservoir could have started leaking during the diagnosis.

If we assume that this is not the case, as the pilot is likely to do in order to reduce the number of possibilities he has to deal with, the number of combinations is reduced to 9. They are given below without the corresponding constraints on the analog variables:

**Location:** RGLN.GRLG.RGAG.BRNL.AGLL

hytech.tex; 18/07/2000; 14:19; p.20
The result now coincides with the derivations reported in [10] (and table I), in which it is stated that the pilot can conclude at this point that at least one of the primary servodynes is leaking and one of the secondary servodynes, i.e. in none of the locations the combinations RGN* and AGN* or RG*N and AG*N occurs, where * stands for N or L. The result can be obtained automatically by adding as an extra constraint to sit2 the assumption of the pilot that the green reservoir is still not leaking (loc[GreenRes]=GRNL). This is also one way to make underlying assumptions concerning the validity of the diagnosis strategy explicit.

6.4. Scaleability

The above shows that with the support of HyTech complex situations can be analysed, that would be difficult to perform accurately with pencil and paper. The analysis of diagnosis methods can be performed more systematically and thoroughly. Our view is that the case study lies on the boundary of what can reasonably and reliably be analysed by hand. From this, a valid question is how the approach scales up to more complex systems.

Adding another control surface increases the complexity of both the task faced by the pilot, and the analysis conducted by the designer, by an order of magnitude. Our experiments have shown that this brings the analysis times up from the order of seconds and tens of seconds to the order of minutes and tens of minutes, but is still well within the capability of current technology. Another point of importance in this respect is that if the reasoning concerning the uae of the system is of a very high level of complexity, then it may well be that the complexity of the system will be problematic for the operator. From this, we feel that the level of complexity which can be accommodated by the technology is sufficient to justify it’s use with real systems.

7. Summary and Discussion

In this paper we have examined the possibility for providing formal support for reasoning about human factors in complex, continuous interactive systems. We believe that if formal models of interactive systems are to be useful as a basis for usability reasoning, then they must include reference to the user [7], and if we are to reason about the operation of the system within a certain context, then environmental aspects must also be considered. A significant
benefit of the type of formalisation and analysis described here is in making explicit any assumptions about the context of an analysis, whether it concerns environmental factors, the operational context (for example standard operating procedures), or the capabilities and performance limits of the user.

Starting from our concerns with the specification and analysis of modern interactive systems, we have shown that hybrid automata provide a useful approach to modelling interactive systems with a real-time or continuous aspect. In our approach, we have modelled both system and user as automata, with links between them via synchronised transitions and shared variables. On the basis of these models, a number of forms of analysis are possible; by translating our questions into a form which can be checked by the tools, we can look at a variety of properties pertaining to possible behaviours, their sequencing, and their timing. These forms of analysis are similar to those commonly carried out in the verification of critical systems, however we cannot stress enough that it is their ability to answer human-factors related questions and their interpretation in this context which we are interested in.

The background to this case study was the diagnosis of failures in a supervisory control setting; an important and critical activity in which the operator of the system is responsible for the safe operation of the system. Careful design and analysis of diagnosis strategies is therefore very important, and motivates the application of a rigorous approach. In the case study, we used as background the theory of diagnosis of [11]. In other areas of application (for example, planning) we might exploit a different domain specific theory.

Task driven analysis, such as the analysis of inferences involved in human performed diagnosis of system faults, has long been established in the field of human computer interaction (see for example [5]). However, the approach to the formal support of such analysis in the context of critical systems which we have presented is a new application of hybrid automata.

Our modelling of the user has been rather simple so far; another interesting area to explore would be the derivation of more detailed user models, particularly with a psychological or ergonomic justification, for example further modelling of user observation, thinking and reaction times. Toolkits of certain common user behaviours (eg. observation based on polling) are also an interesting possibility.

Acknowledgements

We would like to thank the anonymous reviewers and participants of the ERCIM workshop on Formal Methods in Industrial Critical Systems for their helpful comments and suggestions. This work was supported by the TACIT network under the European Union TMR programme, contract ERB FMRX CT97 0133.
References
