JAS-95 Lite: Modelling and Formal Analysis of Dynamic Properties

Simin Nadjm-Tehrani
(simin@ida.liu.se)
Dept. of Computer and info. Science
Linköping University
S-581 83 Linköping, Sweden

Jan-Erik Strömberg
(janerik@s3.kth.se)
Dept. of Signals, Sensors & Systems
Royal Institute of Technology
S-100 44 Stockholm, Sweden

Abstract

In this report we present the multi-disciplinary work performed in the first phase of the COHSY project concerning generation of models and analysis of *hybrid* systems – mathematical models including both continuous and discrete elements. The participants of the project were SAAB Military aircraft, Volvo aerospace, and VOAC hydraulics, as well as three departments from Linköping University: Computer Science, Electrical engineering and Mechanical engineering.

The report addresses modelling and formal verification of a fictitious system, the landing gear of an aircraft referred to as JAS-95 Lite, which involves hydro-mechanical and electro-mechanical sensors and actuators as well as electronic and software modules performing diagnosis and control. The technical system, is moreover in dynamic interaction with a human operator (the pilot). The main aim of this work has been to mathematically prove that specified requirements are satisfied by given design specifications for the controller and alternative models for the physical environment. An architectural model is used to facilitate the combination of alternative configurations.

The report provides a summary of several tracks of activity, a major one being the application and illustration of state of the art techniques in physical modelling of the hardware, and mathematical modelling and verification of the closed loop system. The languages used for modelling range from engineering schematic diagrams to Bond Graphs, hybrid transition systems, hybrid automata, and the temporal logic Extended Duration Calculus. It also provides some insights into modelling in synchronous languages for high level specification of real-time programs – the interest being the investigation of the applicability of tools available for analysis and subsequent code generation from high level designs. Two languages from this family are examined in the context of the case study: discrete models in Esterel and timed models in statecharts as implemented in the tool Statemate. Parts of this work have been presented at a number of international conferences [20, 21, 30, 34, 29].

Contents

Chapter 1

Introduction

1.1 Scope of the report

In this report we present the multi-disciplinary work performed in the first two years of the COHSY project concerning generation of models and analysis of *hybrid* systems – mathematical models including both continuous and discrete elements. An earlier report [6] can be consulted for a general description of the project. This report addresses modelling and formal verification of a fictitious system, referred to as JAS-95 Lite, involving hydro-mechanical and electro-mechanical sensors and actuators as well as electronic and software modules performing diagnosis and control. The technical system, hereafter referred to as "the system", is moreover in dynamic interaction with a human operator (the pilot). The aim of this work is to prove that specified requirements are satisfied by a given design specification for the controller(s) and the physical hardware and environment configuration.

The work reported here has been partially presented in international conferences in the form of shorter articles. This report can primarily be seen as the full version of [20, 21, 30]. It also provides the context to the work reported in [34]. Some extentions to the work reported here will further appear in [29].

In order to formally verify the properties of a given system, we first need a mathematical model of it. However, models suitable for formal verification are not generated in the development of computer-based systems as a matter of routine. Thus, a significant part of a verification effort concerns methods for deriving modular mathematical models from the existing informal engineering documents. In this modelling process we give as much weight to the physical environment as to the controlling software. The models of the hardware and the software are typically fundamentally different in their character. Hence, different competences are required in development and documentation of these models. However, it is the composition of the models for these two parts, also referred to as the *closed loop* model, which is needed for proving the properties we are interested in.

To derive mathematical models from engineering documents, is an inherently difficult task. Even if international document standards are commonly adopted today, the mathematical interpretation of such documents still is rather ambiguous. We have approached this problem by adopting an iterative process in which models are successively refined as a result of earlier analysis. The pragmatic reason for this is the following. The level of detail in the model is dependent on factors such as the requirement specification, assumptions made about the physical components, and the choice of verification technique. Hence, in real-world applications, the modelling and the analysis processes must go hand-in-hand. For this kind of iterative process to be realistic in a multi-disciplinary setting, the formal models derived and the assumptions made must be easy to convey to the domain specialists. Also, new assumptions derived from results made available by earlier analysis, must be easy to incorporate into the model at any time.

The above process necessitates the employment of a range of formalisms and methodologies in our application. To mention a few, we have, at the physical level of abstraction, ideal physical model diagrams and switched bond graphs [31]. At the mathematical level of abstraction, we have employed hybrid transition systems (HTS), and an architecture for the top-level decomposition of hybrid systems [19, 18]. This modular framework allows us to plug in a refined version of an arbitrary module without having to consider (or redo) the rest of the model. In connection with the verification technique, we have experimented with several other mathematical modelling languages: Linear Hybrid Auatomata (LHA) [2] and the Extended Duration Calculus (EDC) [4] from the hybrid family of languages, Esterel (I/O automata) and statecharts from the synchronous family of languages.

1.2 Selected example

The issues above are illustrated in the context of one of the selected examples in [6]; that is, the subsystem in JAS-95 Lite in which the response to the landing command by a pilot are analyzed (see Figure 1.1). When deriving

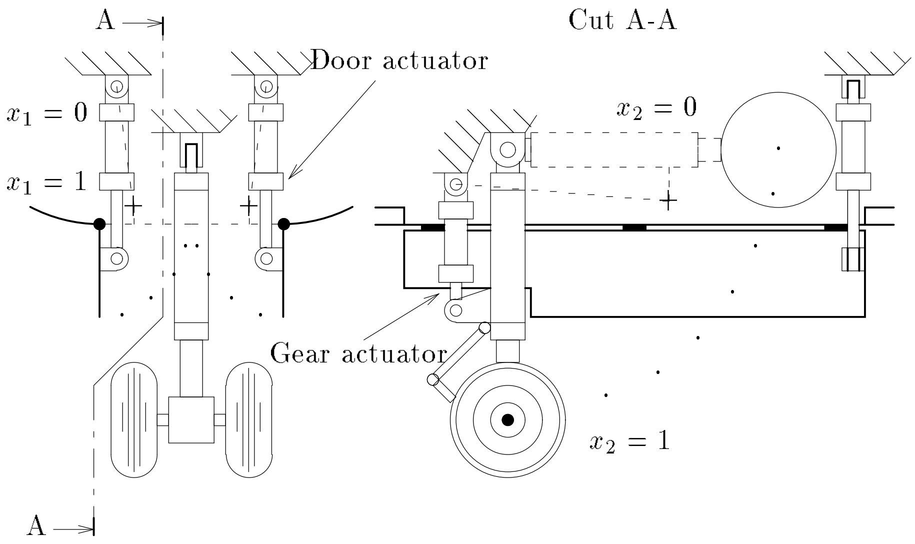


Figure 1.1: The mechanical view of the system studied.

the mathematical models we note that some major elements of the physical configuration have already been fixed due to other demands on the aircraft as a whole (weight etc.). Also, some back up mechanisms have already been envisaged. That is, under abnormal situations there should always be possible to initiate landing using a reserve power supply. Thus we have started the modelling activity at an architectural level.

A basic problem at this level is that the segregation of the landing gear subsystem from the rest of the aircraft is not a straight-forward task. This is due to the fact that achieving the goals of the landing gear is not solely dependent on the functional correctness of a single controller. Rather, the proper operation of the landing gear is possible under certain conditions in the rest of the aircraft. For example the physical operation of the mechanical door and gear components is dependent on hydraulic power at certain locations in the hydraulic power supply system and at certain times. Thus, in order to identify the mode of operation in the gear control system there should be another module for monitoring and diagnosis of the hydraulic power supply.

Based on the above observations we have identified a sub-system consisting of four blocks for modelling and analysis. Figure 1.2 shows the topology of the system under study, where the full arrows denote directions of infor-

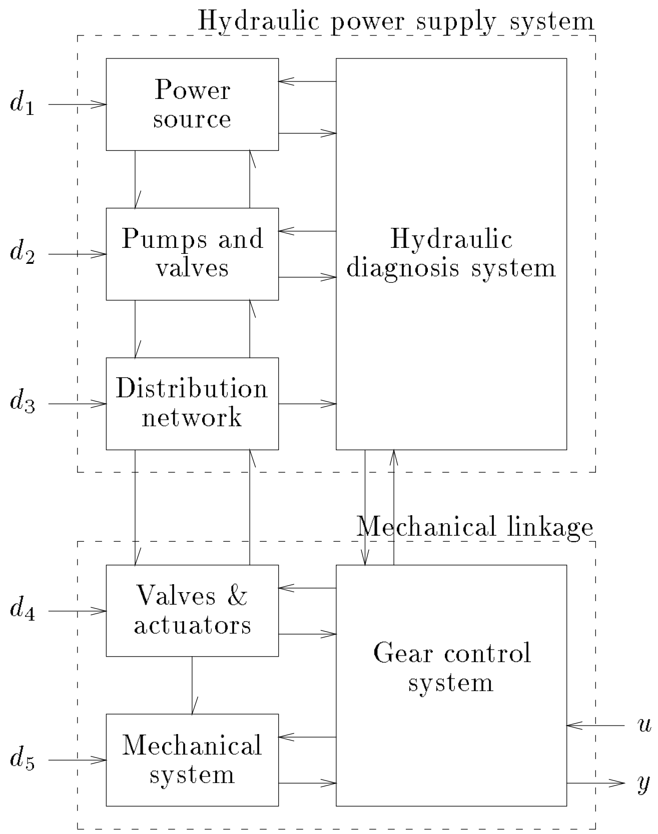


Figure 1.2: An abstraction of the landing gear system.

mation exchange and half arrows directions of (positive) energy exchange. The inputs denoted by d_i are abstractions of other activities in the aircraft treated as disturbances, u denotes pilot commands, and y denotes system state information delivered to the pilot.

1.3 Summary of the report

The work in the project has led to derivation of physical and mathematical abstractions for all four components of the system. At the first stage, models for the hydro-mechanical subsystems (in terms of schematic engineering diagrams) have been developed. These models have been transformed to the energy-based graphical language of switched bond graphs. The latter step significantly aids the derivation of mathematical models for mode-switching physical systems. These are physical systems which can not be described by one set of differential and algebraic equations (DAEs); rather different sets of DAEs are needed depending on the discrete mode in the system. Next, models for diagnosis and controller modules (to be implemented in hardware or software) have been developed. The model for the diagnosis unit necessarily includes a mapping from continuous variables to discrete indications of leakage. This model and the various impacts of the leakage model on the closed loop system has not been integrated in the verification models presented in the current report.

As for dynamic properties, the modelling phase was followed by proofs of the following three properties for several models of the landing gear system.

- The door and the gear do not collide under movement
- Whenever the pilot issues the landing (airborne) command, the gear will be out (in) and the door closed within T seconds.

The above modelling activity has made it possible to investigate different verification techniques on the same example. First, we have tried composition of the coarsest model of the door and gear with a timed dynamic controller, and applied the symbolic model checking techniques implemented in the HyTech verification tool. The physical models could be used to justify that under certain hydraulic power modes the operation of the door and the gear could be approximated by time-linear evolutions. Hence, the safety and timeliness of the controller under these conditions can be tested using automatic methods. The same underlying models were also used to experiment with other, more familiar modelling and verification techniques: the Statemate environment for the language of statecharts, and the development environment for Esterel programs (translated into Input/Output machines by Mauto).

Second, for the same (time-linear) model of the physical system we have analyzed its composition with a different controller (consisting of a static mapping). This composition was then subjected to a manual proof within Extended Duration Calculus leading to a proof of the same 3 properties. The point with this hand proof is that it provides the necessary structure over which the same proofs for a non-linear model of the hydro-mechanical system can be based. This corresponds to a mathematical model for a more refined version of the hydraulic supply system in which the hydralic pressure is not assumed to be constant, it is rather assumed to be regulated by a hydro-mechanic pump.

The structure of the report is as follows. Chapter 2 presents the steps for translating engineering schematic diagrams (or appartus models) of the physical environment to physically based models in Switched Bond Graphs. This work leads to the mathematical models on which the formal verification is based. The chapter also provides an abstract architectural breakdown of the closed loop system. Chapter 4 refines the architectural model by filling alternative models for the constituents. It also introduces the notion of hybrid transition systems used for mathematical modelling of the closed loop system. Chapter 5 is devoted to the automatic verification techniques applied to this example, and chapter 6 deals with the deductive proofs in EDC which are extended to the nonlinear environment model. In the final chapter we comment on observations and the insights gained during these studies.

Acknowledgements

This research was partially funded by the Swedish National Board for Industrial and Technical Development (NUTEK). Also, the authors wish to thank Göran Backlund and his associates at Saab Military Aircraft, Linköping and Arne Jansson at the division of Fluid Power, dept. of Mechanical Engineering, Linköping University, for many valuable contributions and comments on the hydraulic and mechanical models of which only a fraction were presented here.

Chapter 2

From engineering documents to mathematical models

The chapter illustrates the multi-paradigm modelling approach leading from the documents currently produced by domain experts to the mathematical models suitable for formal verification in the following chapters.

2.1 Apparatus modelling of hardware

We let the term 'apparatus' refer to anything that is controlled by a discrete controller and of which the behaviour is described by physical laws. Note that this excludes software components. Apparatus models cover electromechanical or electrohydraulic actuators, the systems governed by these actuators and the sensors used to measure the effect of the actuation. As an example, Figure 2.1 depicts an apparatus model of a system involving three different physical domains: electric, hydraulic and mechanic. This particular diagram is a model of the landing gear subsystem of an aircraft.

Industrial documentation of the components constituting an apparatus, whether they are electric, hydraulic or mechanic, is typically informal. The icons used in Figure 2.1 are fairly well standardized in as far as they refer to *functions* within the system, implemented by specific devices. However, they are ambiguous with respect to the underlying physics: it is not clear which physical effects must be taken into account. The industrial description for such devices typically consists of a well defined 'item' or 'order code' number associated with each device.

As a matter of fact, it is not even possible to precisely define the un-

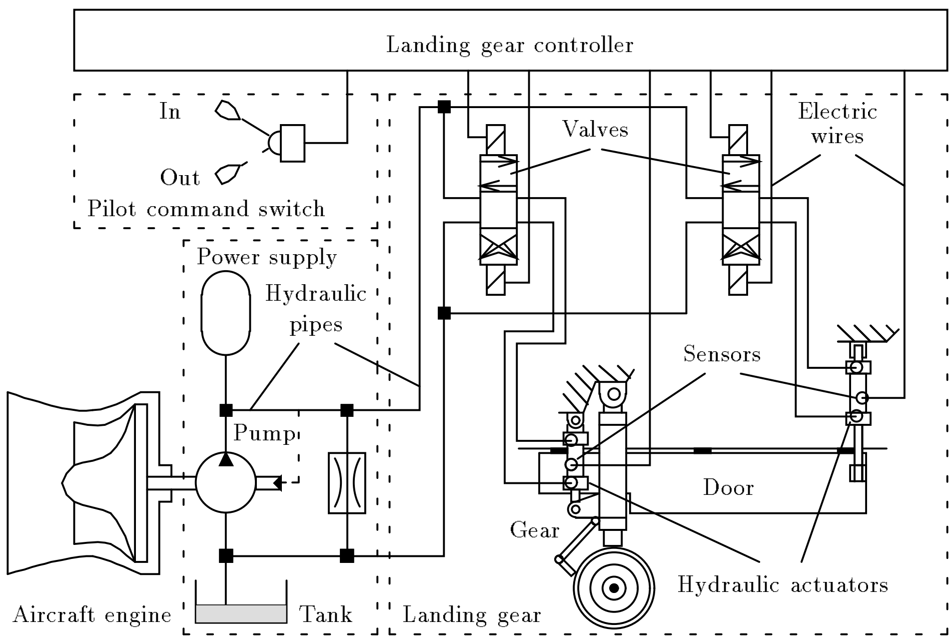


Figure 2.1: The landing gear system as documented by a mechanical engineer.

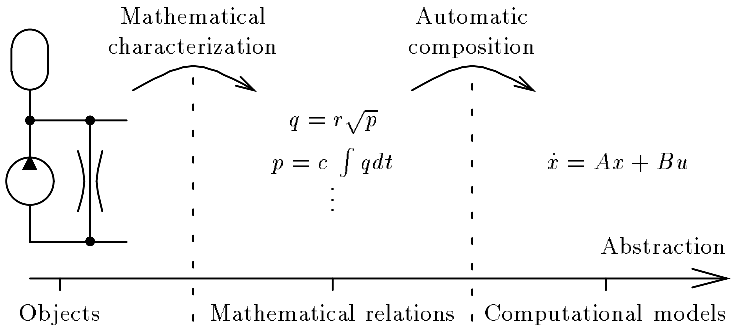


Figure 2.2: The structure of 'object oriented' modelling languages such as Omola/Dymola.

derlying physics of a device component. The reason for this is that the behaviour of a component depends on the circumstances in which the device is used and its characteristics with respect to accuracy, timing, dynamic range etc. Hence, the type of 'models' typically used in industry are in themselves not informative enough for determining a unique mathematical behaviour model suitable for e.g. formal verification.

Nevertheless, it is possible to link, in an organized way, mathematical models to each of the icons (objects) used in schematic diagrams, provided we know the context in which the objects are being used. This is the approach adopted in so called 'object oriented modelling languages' such as Omola [15, 3] and Dymola [5]; see Figure 2.2. Links between objects and mathematical relations describing the behaviour of each object are here made explicit. Once the behaviour of the individual objects has been specified, the composed overall model can be automatically generated. This compositional approach is of course necessary in order to allow for the modelling of real-sized systems.

Although the object oriented approach solves some problems, there are still some unresolved ones. For example, a major problem is how to describe the assumptions underlying the mathematical device models in a standardized way. Also, we have the more fundamental problem of having to link behaviour models to unique objects. This is not always possible since behaviour ever so often emerges from the interaction *between* two or more devices or objects. In 'object oriented' languages such as Omola/Dymola, this problem is typically circumvented by introducing abstract 'interaction objects'.

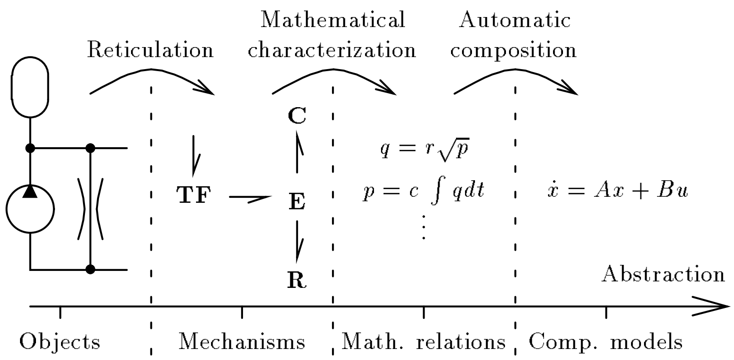


Figure 2.3: The structure of the energy oriented bond graph language.

A clarifying example of an interaction effect is *friction*. Friction is the effect of two bodies moving relative to each other. In languages like Omola or Dymola a 'friction object' is to be inserted in between the objects representing the two moving bodies. Hence, in that case the clear relationship between the objects in the original schematic diagram and the Omola/Dymola type of representation cannot be maintained. Thus we get an intermingling of models for concrete devices with models of abstract interactions. To summarize, in these models the links between objects and mathematical relations are made explicit, but the assumptions made in the mathematical characterization are not. Also, abstract 'interaction objects' typically have to be introduced.

2.2 Physical modelling

Bond graphs [24, 13], provide a neater solution to the fundamental modelling problems mentioned in the previous chapter. This is done by introducing an additional modelling formalism 'in between' the concrete device oriented *technical level* and the more abstract mathematical level; see Figure 2.3.

In this chapter we first give a brief introduction to classical Bond graphs and their swithed extension, followed by a BG model of the landing gear of JAS-95 Lite.

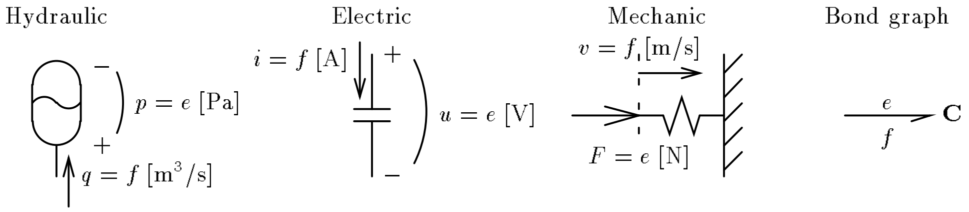


Figure 2.4: The bond graph representation of three different storage mechanisms.

2.2.1 Classical Bond Graphs

As opposed to the Dymola/Omola type of language, the bond graph language is *not* a programming language. Bond graphs represent the energetic interaction structure of an apparatus and nothing else. This also implies that bond graphs are restricted to systems which can be naturally represented using energy as the basic concept. The advantage of adopting energy concepts is that they are well understood and have a common meaning to almost all classical engineers; no matter the particular engineering domain.

The original bond graph language consists of only nine basic elements. These nine elements form the nodes of a graph connected by half-arrow shaped *bonds* representing exchange of energy. The elements represent energy processes. With each bond a pair of power variables *effort* and *flow* is associated, with $e \times f = power$. Effort and flow are generalisations of force and velocity, voltage and current, temperature and entropy flow in different domains.

The nine energy processes defined by the language fall into one of the following five categories: (1) sources ($\mathbf{Se}, \mathbf{Sf}$), (2) storages ($\mathbf{C}, \mathbf{I}$), (3) dissipations ($\mathbf{R}$), (4) distributions ($\mathbf{E}, \mathbf{F}$) and (5) conversions ($\mathbf{TF}, \mathbf{GY}$). Each of these processes corresponds to a well defined ideal physical mechanism. For example, a bond graph **C**-element represents an idealized model of a hydraulic accumulator, an electric capacitor or a mechanic mass-less spring (see Figure 2.4). By selecting and combining these ideal elements, the modeller is forced to express every assumption in a standard and formal way.

The first phase in the bond graph modelling procedure is referred to as *reticulation* [24]. As a matter of fact, this inherently difficult modelling phase is always present no matter the modelling approach employed. Based on the purpose of the model and the context in which the devices are used it is here decided which physical effects to consider. Bond graphs are unique in that they force the modeller to represent this phase and the decisions made explicitly. Moreover, bond graphs accommodate the modeller with the smallest possible set of primitive concepts sufficient for a wide class of apparatus. Of course bond graphs provide little or no support when systems outside this class are considered.

Once the system has been reticulated successfully, i.e. the bond graph has been drawn, the mathematical characterization of the nodes in the graph is in most cases straightforward. This is due to the fact that each node is associated with a well defined mathematical structure: the *constitutive relation*. Note that a constitutive relation is a relation in the original mathematical sense; *not* a computation. Therefore, a bond graph is *acausal*, i.e. a declarative description of a computational model. The composition of the individual relations into a complete computational whole is completely algorithmic. The result of automatic composition is a system of differential and algebraic equations (DAE). This provides a well-defined mathematical semantics for bond graphs [31].

2.2.2 Switched Bond Graphs

Classical bond graphs allow for finite flows of energy only and are thus restricted to modelling continuous changes. However, many practical devices undergo abrupt changes in their behaviour, mostly due to external or internal discrete control. Such devices will be referred to as *mode-switching* systems.

In order to carry over the bond graph methodology to mode-switching systems, an extension referred to as the *switched bond graph* was introduced in 1993 [32, 33, 31, 28]. The idea is to allow for reticulation of hybrid systems without loosing any of the conceptual clarity of the traditional bond graph language. The main ingredient in the extension is a new ideal element, *viz.* the *ideal generalized switch* (**Sw**) [32, 28]. As opposed to the other bond graph elements, the **Sw**-element is *not* associated with a constitutive mathematical relation over effort and flow. Instead, it is associated with a discrete mathematical structure determining the discrete state of the switch. Once all elements of a switched bond graph have been mathematically characterized – including the **Sw**-elements – it can be automatically converted into a computational hybrid system: two sets of DAEs and boolean transition conditions. Further details on how to characterize individual **Sw**-elements and how to compose a switched bond graph to a computational whole, is provided in [31, 28].

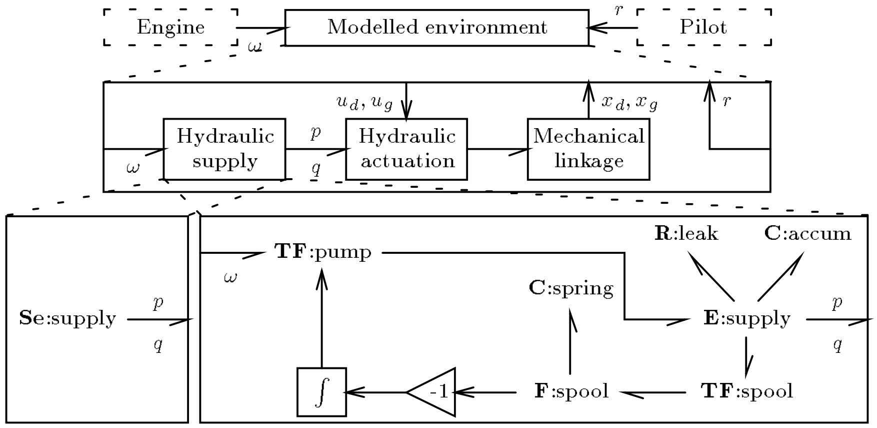


Figure 2.5: The physical environment models. Left: the coarsest model of the hydraulic supply. Right: the model including a hydro-mechanically regulated pump.

2.2.3 The Landing Gear Models

The landing gear system has been chosen as one of the test cases in an ongoing multi-disciplinary research effort in cooperation with aerospace industries in Sweden. The project concerns evaluation and development of tools and methods for the modelling and formal analysis of complex hybrid systems.

The system involves hydro-mechanic, electro-hydraulic and electro-mechanic hardware components interacting with hardwired as well as software based modules performing diagnostic and controlling tasks. The system contains continuous control loops, implemented in hydro-mechanic hardware, and discrete control loops, implemented in hardware and software. Finally, the system interacts with a human operator, namely the pilot, who in turn critically depends on its proper operation.

The hardware components in the landing gear system consist of the landing gear itself, i.e. the wheel–and–suspension system, a pair of doors protecting the gear during flight and landing, and a pair of hydraulic actuators for the manouvering of the gear and the doors; see also Figure 2.1. As part of the landing gear system, the hydraulic power supply system provides the hydro-mechanic energy needed to manoeuvre the gear and the doors under all operating conditions.

The software components implement a discrete controller for the door and gear actuators, as well as a diagnosis module which detects malfunctions in the hydraulic power supply system.

For the purpose of illustration, we present two different bond graph models of the hydraulic power supply subsystem and a single model of the door–and–gear subsystem. These models differ in the assumptions made about the subsystems, and therefore illustrate the capability of bond graphs to make such assumptions clear.

In the coarsest model of the hydraulic power supply we represent the assumption that the hydraulic power supply system behaves as an ideal constant pressure source; see the lowest left-most box in Figure 2.5 where p and q stand for hydraulic supply pressure and flow respectively. Combining it with the model of the door–and–gear subsystem, the overall plant model converts to a simple time-linear dynamic system

$$\dot{x}_d = \alpha_d \, u_d, \quad x_d \in [0,1] \tag{2.1}$$

$$\dot{x}_g = \alpha_g \, u_g, \quad x_g \in [0,1] \tag{2.2}$$

where $\alpha_d, \alpha_g > 0$ are constants (time-invariants), x_d (x_g) the normalized door (gear) position and u_d (u_g) $\in \{-1, 0, +1\}$ is the three-valued door (gear) control signal.

In the more detailed model of the power supply system (lowest right-most box in Figure 2.5), we no longer assume that the hydraulic pressure is ideal. Instead we make an explicit model of the hydro-mechanic regulator which in fact attempts to make the pressure 'as constant as possible' given a particular engine velocity ω. In this case the overall bond graph converts to a non-linear dynamic system

$$\dot{x}_d = \alpha'_d \, p \, u_d, \quad x_d \in [0,1] \tag{2.3}$$

$$\dot{x}_g = \alpha'_g \, p \, u_g, \quad x_g \in [0,1] \tag{2.4}$$

$$\dot{p} = -(\gamma + \gamma_d \, |u_d| + \gamma_g \, |u_g|) \, p + \beta \tag{2.5}$$

where $\alpha'_d, \alpha'_g, \gamma_d, \gamma_g > 0$ are constants and p is the hydraulic supply pressure. The quotient β/γ is a monotonic function of the aircraft engine velocity ω.

These models are only a small selection of models we have derived. Other models consisting of some fifty bond graph elements have been efficiently developed in cooperation with hardware experts from Saab Military Aircraft and the Dept. of Fluid Power at Linköping University. The efficiency of this iterative modelling process has significantly gained from the use of bond graphs. By means of the bond graphs we were able to communicate our models with all engineers from the industrial partners and were able to sort out misunderstandings and misconceptions at an early stage of the modelling process.

The DAE models derived can now be plugged into the model of the closed loop system as required. To do this we use a hybrid modelling architecture [17] described in the next section.

2.3 The architectural model

This section advocates a framework for putting together the environment models derived earlier with mathematical models for the other parts of the system. It can be used for refining the overall architecture in Figure 1.2 by making the interface between the controller/diagnoser and the physical world more explicit.

2.3.1 A generic decomposition

The generic architecture as depicted in Figure 2.6, accomodates both the mathematical models for the physical environment and the mathematical models for the discrete controllers and supervisory systems. It has been adopted from the multi-layer architecture for hierarchical control suggested earlier [17]. This architectural decomposition is used as a framework for verification and makes the interfaces (as dictated by the choice of sensors and actuators) explicit.

The discrete controller consists of a *selector* asynchronously reacting on discrete events e detected by the *characterizer*. The characterizer generates these events using a classification function over the real valued environment variables z. The output of the selector is the discrete choices c of control algorithms implemented by the *effector*. The *environment* is driven by the real valued control variables u. The unpredictability of the environment is made explicit by the disturbance variable v. This variable is typically used for allowing uncertainties in sensor measurements. In addition, it may be used to model major elements of the environment not under control.

2.3.2 Framework for iterative modelling

This architecture has been used in a number of different applications and aids the process of modelling the closed loop system for the purpose of verification. In the landing gear study we have used the architecture as a

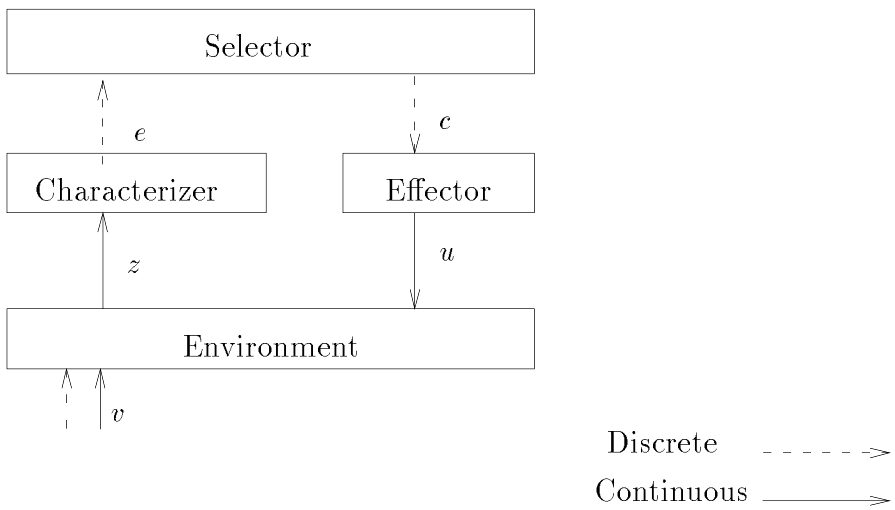


Figure 2.6: A generic architecture for decomposition of hybrid systems

framework for iterative modelling of the closed loop system. It was thus used to plug in different models of the environment which keeping the same selector, characterizer and effector; or different selectors while keeping the same environment model. Thus, we have been able to study and illustrate different verification techniques depending on the complexity (granularity) of the closed loop model and the property to be verified.

In the following chapter we will show the details of the alternative landing gear models by filling the boxes in the above architecture.

Chapter 3

Hybrid mathematical models

In this chapter we describe how the models for the closed loop system can be derived using the alternative mathematical models for the hardware derived earlier. We recall that the alternative hardware models were based on alternative assumptions during physical modelling. Similarly, alternative mathematical models of the designed controller can be established and plugged into the decomposition architechture presented earlier.

3.1 Models of the controller

In what follows we give two alternative design models for the landing gear controller. These different models reflect assumptions made about the alternative implementations. The first one is a dynamic selector and models a software implementation of the controller incorporating delays. The model includes discrete state transitions where the label $[l, u]$ on each transition may be used to represent the lower and upperbound on the delay associated with that transition. These delays may be used to model communication and computation delays within the computer system to the extent that they are relevant.

The second model is a static selector which corresponds to an implementation of the controller in hardwired electronic, and hence does not incorporate any delays. The first controller may be considered as that used under normal operation, whereas the second controller being a backup-system taking over in emergency situations. The two different controller models are

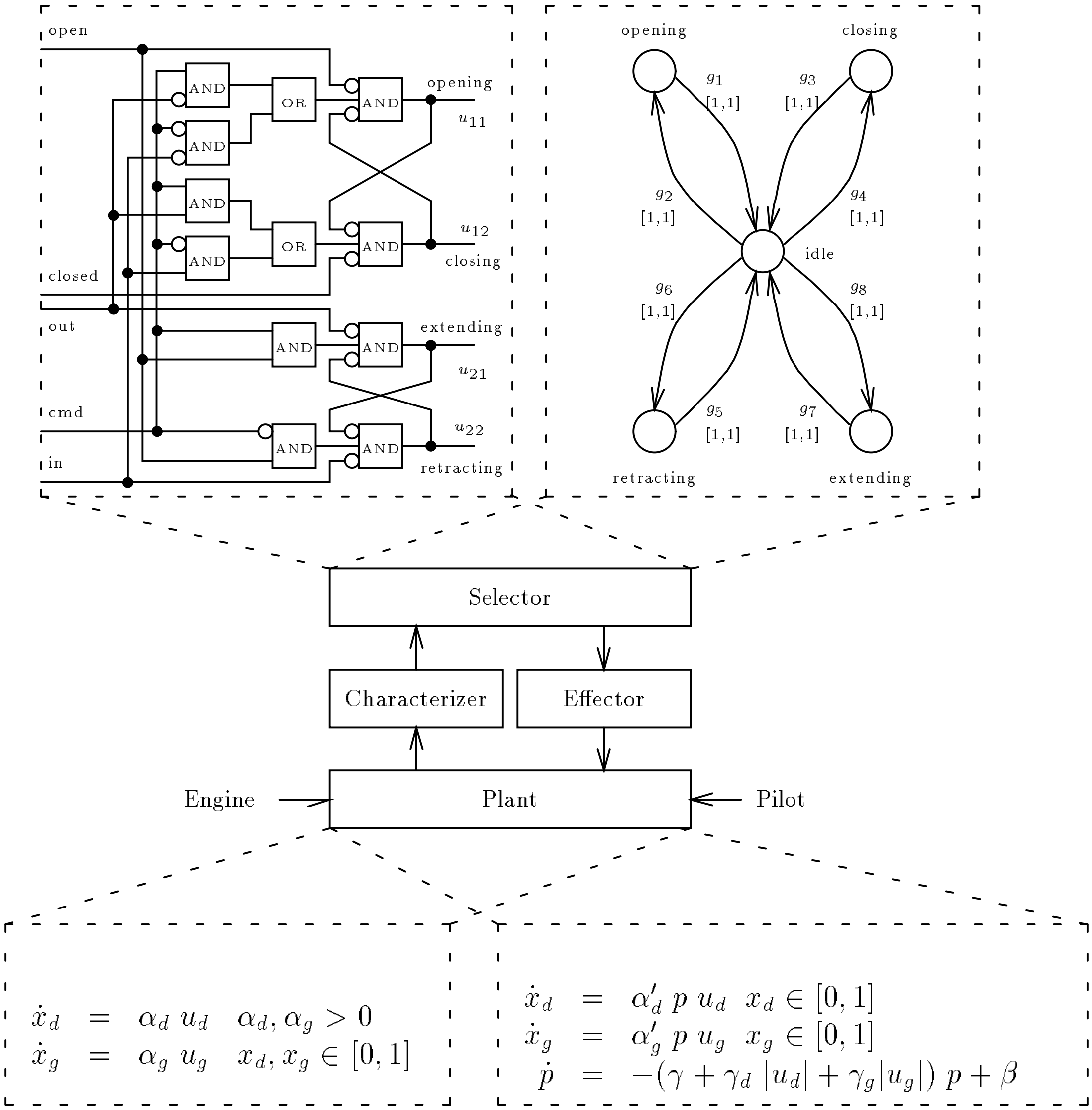


Figure 3.1: Alternative mathematical models for the plant and the controller giving rise to 4 different closed loop models.

outlined in Figure 3.1. Note that in this simple version, we have associated deterministic unit delays with every transition.

As it can be seen in the figure, the inputs to the static selector are discrete states *open, closed, in, out, cmd.* These events are determined by the characterizer based on the sensed values of the door and gear position and the current pilot command. The same inputs are also used to determine the somewhat more complex conditions (guards) g_i used in the dynamic selector. The following formulas describe the condition for each change of state in the dynamic selector:

$$\begin{aligned}
g_1 &\equiv open \vee [7] \vee [1] \\
g_2 &\equiv (\neg cmd \wedge ([8] \vee [2])) \vee (cmd \wedge ([9] \vee [3])) \\
g_3 &\equiv closed \vee [7] \vee [4] \\
g_4 &\equiv (cmd \wedge ([8] \vee [5])) \vee (\neg cmd \wedge ([9] \vee [6])) \\
g_5 &\equiv out \vee [9] \vee [7] \\
g_6 &\equiv (cmd \wedge ([4] \vee [6])) \vee [1] \\
g_7 &\equiv in \vee [1] \vee [7] \\
g_8 &\equiv \neg cmd \wedge ([4] \vee [5])
\end{aligned}$$

and where the conditions $[k]$ are given by

$$\begin{aligned}
[1] &\equiv \neg open \wedge closed \wedge \neg in \wedge \neg out \\
[2] &\equiv \neg open \wedge closed \wedge \neg in \wedge out \\
[3] &\equiv \neg open \wedge closed \wedge in \wedge \neg out \\
[4] &\equiv open \wedge \neg closed \wedge \neg in \wedge \neg out \\
[5] &\equiv open \wedge \neg closed \wedge \neg in \wedge out \\
[6] &\equiv open \wedge \neg closed \wedge in \wedge \neg out \\
[7] &\equiv \neg open \wedge \neg closed \wedge \neg in \wedge \neg out \\
[8] &\equiv \neg open \wedge \neg closed \wedge \neg in \wedge out \\
[9] &\equiv \neg open \wedge \neg closed \wedge in \wedge \neg out
\end{aligned}$$

3.2 Closed loop models

Figure 3.1 combines the alternative models presented earlier in a uniform framework. In what follows we show the mathematical closed loop models

Selector

$$
\begin{array}{rcl}
((cmd \wedge \neg out) \vee (\neg cmd \wedge \neg in)) \wedge \neg open \wedge \neg u_{12} & \Leftrightarrow & u_{11} \\
((cmd \wedge out) \vee (\neg cmd \wedge in)) \wedge \neg closed \wedge \neg u_{11} & \Leftrightarrow & u_{12} \\
cmd \wedge open \wedge \neg out \wedge \neg u_{22} & \Leftrightarrow & u_{21} \\
\neg cmd \wedge open \wedge \neg in \wedge \neg u_{21} & \Leftrightarrow & u_{22}
\end{array}
$$

Characterizer

$$
\begin{array}{rcl}
open & \Leftrightarrow & x_d \geq 1 \\
closed & \Leftrightarrow & x_d \leq 0 \\
in & \Leftrightarrow & x_g \leq 0 \\
out & \Leftrightarrow & x_g \geq 1 \\
cmd & \Leftrightarrow & r > 0
\end{array}
$$

Effector

$$
\begin{array}{rcl}
u_{11} \wedge \neg u_{12} & \Leftrightarrow & u_d = 1 \\
(\neg u_{11} \wedge \neg u_{12}) \vee (u_{11} \wedge u_{12}) & \Leftrightarrow & u_d = 0 \\
\neg u_{11} \wedge u_{12} & \Leftrightarrow & u_d = -1 \\
u_{21} \wedge \neg u_{22} & \Leftrightarrow & u_g = 1 \\
(\neg u_{21} \wedge \neg u_{22}) \vee (u_{21} \wedge u_{22}) & \Leftrightarrow & u_g = 0 \\
\neg u_{21} \wedge u_{22} & \Leftrightarrow & u_g = -1
\end{array}
$$

Environment

$$
\begin{array}{rclcrcl}
\dot{x}_d & = & \alpha_d \, u_d & \qquad & \alpha_d, \alpha_g & > & 0 \\
\dot{x}_g & = & \alpha_g \, u_g & \qquad & x_d, x_g & \in & [0,1]
\end{array}
$$

r Pilot command

Figure 3.2: The system built up from the time-linear environment model and the static selector.

for 3 of the 4 resulting models obtained by composing different selector and environment models. To begin with we take the coarsest model of the environment which gives a time-linear evolution for the door and gear positions. This corresponds to choosing either of the selector models in Figure 3.1 and composing it with the differential equation model in the bottom left corner of the figure. To achieve this we will further make the mappings in the interface between the controller and its environment (i.e. the characterizer and the effector) explicit.

First we obtain a model of the system consisting of the static selector and the coarse environment model as depicted in Figure 3.2. This model is clear enough from a representational point of view. However, the concrete mathematical models are presented in chapter 5. There, in order to prove the desired properties of the model we will embed the different system components in an interval based temporal logic, Extended Duration Calculus. The proof system of EDC is then used for carrying out the logical reasoning about the intended properties. Note also that keeping the same selector and changing the environment model to a more realistic one is simply achieved by plugging in the bottom right model of the environment in Figure 3.1. Chapter 5 also includes the EDC model based on this architectural decomposition.

Next we consider the combination of the dynamic selector (top right box in Figure 3.1) with the time-linear environment model. In this case the somewhat simpler effector mapping from the discrete states of the selector to the parameters of the environment model is shown in Figure 3.3.

This closed loop combination gives rise to a compact model of the system expressed in hybrid transition systems. Here we give an informal introduction to this formalism followed by the actual model resulting from the above composition.

3.3 Hybrid Transition Systems

In [18] we formally define *hybrid transition systems* (HTS) as a modular extention of *timed transition systems* [11]. Here we give an informal account of the HTS formalism. In HTS, the system variables are partitioned into disjoint sets of input and state variables. Both input and state variables may be either continuous or discrete. The behaviour of the system is modelled by phases of continuous activity interleaved with discrete mode changes. In each mode, the change of continuous state variables is defined by differential *or* algebraic equations, and the value of the discrete variables remains constant. The change in each state variable is defined in terms the other state and (or) input variables, given in terms of DAE in state space form.

Discrete mode changes take place only if the *guard* to some outgoing transition becomes true. A guard is a boolean condition over the input and state variables and elements from their value domains.

We use *hybrid transition graphs* for representing HTS. Each mode m of the system is represented by a node in the graph. With each node we associate one set of differential and algebraic equations, e.g. $x = G_m(x, u)$ and $\dot{x} = F_{m'}(x, u)$ provided in Figure 3.4. Each discrete transition in the system is represented by an arc in the graph labelled with a guard g and a pair of timing constraints $[\mathbf{l}, \mathbf{u}]$. Transitions are taken instantaneously whereas the delays $\mathbf{l}$ and $\mathbf{u}$ belong to non-negative real numbers. These timing constraints are essentially used to model delays in controllers. Hence, for plant models $\mathbf{l} = \mathbf{u} = 0$ except where explicit actuator delays are modelled.

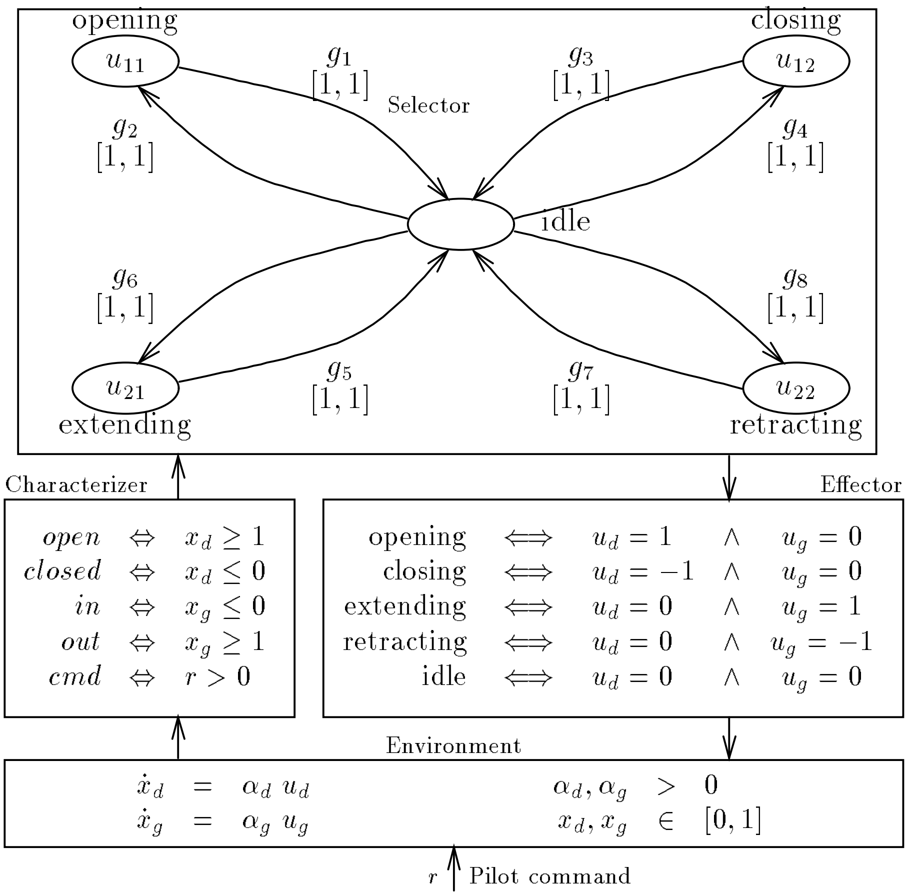


Figure 3.3: The system built up from the dynamic selector and the time-linear environment.

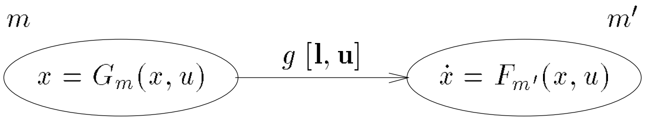


Figure 3.4: An example hybrid transition graph.

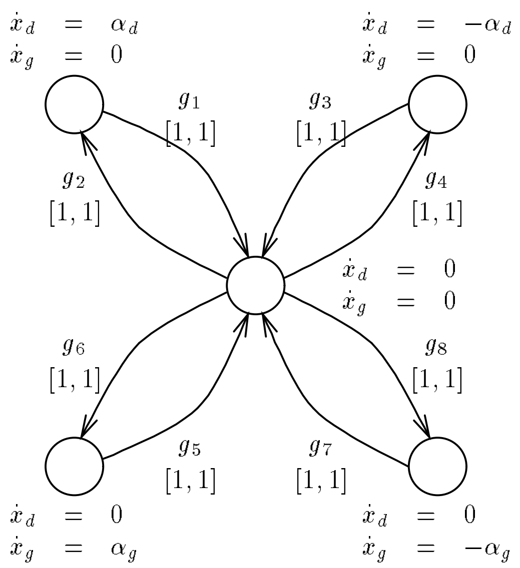


Figure 3.5: A HTS model of the closed loop system incorporating the timed selector and the time-linear hardware model.

3.3.1 Operational semantics

The operational semantics can be informally described as follows (see [18] for a formal definition). The system remains in its current mode m while the guard g is not true. If g remains true over an interval of length $\mathbf{l}$, then the system *may* move to the next mode m'. If g remains true over an interval $\mathbf{u}$, then the system *must* change its mode to m'. In other words the system changes its mode from m to m' within $\mathbf{l}$ and $\mathbf{u}$ time units. If $\mathbf{l} = \mathbf{u} = 0$, then the mode change takes place instantanuously as in the case of models for physical systems. On the other hand, $\mathbf{l} = \mathbf{u} \neq 0$, can be used to model deterministic delays in (cyclic) computational processes.

When a continuous variable is defined by a differential equation in a mode, then the initial value of that variable is the same as its final value in the previous mode. Otherwise, if a continuous variable is defined by an algebraic equation, the value of the variable may change discontinuously in the transition. The new value will then be defined in terms of the other variables in accordance with the algebraic equation.

3.3.2 Parallel composition

Parallel composition is defined as a binary operator which produces a new HTS from two existing HTS's. The constituent subsystems are required to have no common state variables, but otherwise communicate through shared variables. The composition operator resembles the product operation in automata theory in the sense that the set of modes of the new HTS is a Cartesian product of the modes of the two original subsystems. The new set of state variables is the union of the subsystems' state variables, and the set of input variables is the union of the respective input variables reduced by shared variables (used for internal communication). Since the state variables in the subsystems are disjoint, the set of differential and algebraic equations in each new mode is the union of equations existing in the original subsystems restricted to the state variables of the composed system. Each transition in each subsystem gives rise to several transitions in the composed system, modelling the case where one subsystem changes its mode and the other remains in its current mode.

3.3.3 HTS representation of the landing gear

The generic architecture and the modular character of hybrid transition systems can be used to obtain a compact mathematical representation of the closed loop system as depicted in Figure 3.5 where the guards g_i are defined in terms of x_g, x_d and r by translating the formulas over the events *in, out, open, closed, cmd* provided earlier using the characterizer mappings in Figure 3.3.

In the chapters which follow we use the above models to analyze the safety and timeliness properties of the closed loop system. In particular, the above HTS model will be analysed using automatic verification techniques.

Chapter 4

Automatic verification

In this chapter we give a brief survey of a number of automatic verification techniques applied to the landing gear models developed earlier. The tools employed in this study include both research prototypes and commercial modelling tools. Sections 4.3 to 4.4 provide more detail on this work and section 4.5 includes some comparative remarks.

4.1 Requirements Specifications

In order to analyse the models derived by means of formal verification techniques we need to state the essential requirements about the system in a formal language. According to our experience, in complex engineering applications there are great difficulties in capturing these requirements (even when stated informally) in a systemmatic manner. However, despite the importance of the topic, here we leave out the issues connected with *requirements engineering*, the systemmatic decomposition of higher level requiremnts into lower level reqiurements, tracibility, and so on. For the purpose of studying the verification techniques, we simply assume that certain safety and timeliness requirements can be identified and stated.

Using a mathematical model of the closed loop system, our main concern has been to study two safety and timeliness properties of the system, that is:

- Safety: the closed loop model guarantees that the door and the gear will not collide in movement;

- Timeliness: given that the pilot command is not changed more frequently than every t seconds, the composed system will have the property that a pilot command leads to the desirable state ("closed and extended" or "closed and retracted") within d seconds ($t \geq d$).

In what follows, and also in the next chapter we will show variations of methods applied to verify these properties formally.

4.2 State space search in Statemate

A primitive technique for detecting violation of safety properties uses systematic and exhaustive search of the state space. This method is obviously not optimal for large systems but it is unfortunately the only verification technique available in commercial environments for high level design descriptions of complex systems[1].

One such specification tool is Statemate which supports development of high level hierarchical specifications and code generation from statecharts. The "dynamic check" facility in the tool Statemate falls into the above category and should be seen as an organized simulation of a particular design description until the proof of violation of a certain property is obtained. The exploration of the state space follows all possible execution branches according to the version of statecharts semantics implemented in Statemate [10].

Note that considering Statemate as a semi-formal software engineering tool is due to the fact that the formal semantics exists for a fragment of the features supported by the tool. For example, the different model views, i.e. the dataflow like language of activity charts and the architectural view provided by module charts, have not been mathematically connected to the statecharts language. Also, formal semantics has only been provided for the pure event-based fraction of statecharts although state variables with value domains ranging over reals or integers are also supported by the tool[2]. The latter feature was exploited in representing the closed loop model for the landing gear in order to stay as close to the hybrid mathematical model as possible. We found that modelling controllers with complex control struc-

[1] We classify system specifications in propositional logic, as supported by NP Tools (Logikkonsult AB) for example, as low level descriptions into which higher level system specifications may well be transformed eventually.

[2] Though full formal semantics for Statemate languages is a topic for active research. So is the translation of statechart like languages to boolean equations or BDDs.

tures is not a problem but continuous plant models have to be represented as discrete time (difference) equations.

The landing gear model at the top most level can be presented in Statemate according to Figure 6.1 in the Appendix. It includes a test environment by parallel composition with a pilot model and a clock which can be reset for verifying timeliness properties (a timer).

The door and gear are represented as state machines with one state, where the event of entering the state at every system time step coincides with incrementing the value of the door (respectively gear) position by a constant value. The model for the door is depicted as the statechart in Figure 6.2. Note that the continuous mathematical model is thus transformed to a discrete time model using a specific sampling interval.

The model of the controller corresponds in a natural way to the dynamic model at the right hand top of Figure 3.1. where an intermediate state is introduced to model waiting of the system for the duration of the transition delay. This is depicted in Figure 6.3.

Next we show how the requirements specifications are modelled. To check each requirement we devise a watchdog which observes the variables of all the system components and takes the transition from a safe to an alarm state whenever the required property is violated. the watchdog for the timeliness property is shown in Figure 6.4.

The search for reachable unsafe states (or overdue responses) is then taken care of by the use of *dynamic check* analysis function in Statemate over the landing gear test system together with the watchdog.

4.3 Symbolic model checking with HyTech

Within algorithmic techniques for hybrid models a specific subclass of models dominate the scene. These are hybrid systems in which the continuous variables change at a linear rate of time (e.g. our coarsest model of the hydromechanics). One of the suggested methods is symbolic model checking, originally developed for purely discrete models using representations in binary decision diagrams (BDD) [16]. Nicollin et al. and Alur et al. extend this method for proving properties of timed and hybrid stystems [22, 2]. Linear hybrid automata (LHA) is an example formalism [1] in which symbolic model checking can be used as a verification technique.

This method works on partitions of the infinite state space of the system. Symbolic manipulation is performed on finite descriptions (polyhedra) of

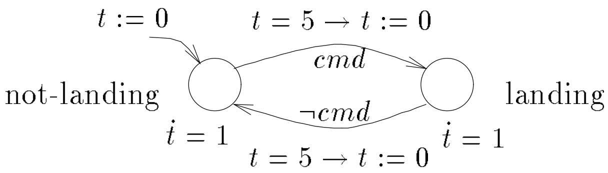


Figure 4.1: A possible model for the pilot in LHA: changing commands every 5 seconds.

infinite sets of points in the n-dimensional state space. More precisely, the set of reachable states is computed symbolically by iterative computations of the fixed point of a function, whereby the unreachability of unsafe states can be proved if the fixed point computation terminates. The problem is in general undecidable for hybrid systems (even with the restriction to LHA), but there are suggestions that computations terminate for most practical cases. This is a proposition which we have tested in this practical example.

The symbolic model checking tool that we have tested is the HyTech prototype implemented by Henzinger et. al. The tool accepts a model of the hybrid system and its surrounding environment (the pilot) in a restricted form of LHA. It also requires a set of "bad" states characterized as a boolean expression over the continuous state variables. In our example bad states are those in which the the door is not open and the gear is moving. The hybrid model of the landing gear was therefore transformed to a form suitable for this software tool.

The hybrid model of the landing gear was a transformed version of Figure 3.5 to the (textual) syntax accepted by HyTech (later versions of HyTech have a graphical interface, but this work used the 1994 version). This involved several straight forward but tedious transformations, mainly due to not accepting non-convex guards and invariances. The model for alternative pilots can be provided as different LHA; for example, Figure 4.1 presents a pilot which changes its landing command every 5 seconds.

Our experiments show the viability of the technique for applications with subcomponents which can be assumed to have time-linear evolutions. At the same time they point out the need for a higher level representation of the design than that accepted by HyTech. After splitting the transitions and locations and when several local clocks are added to a system, the representation in HyTech is far from the simple compact model presented in Figure 3.5.

The results of the experiments with the above example can be summarized as follows:

- All experiments with the tool terminated within seconds.
- The hybrid model of the landing gear was verified to have the safety property "the door is always open when the gear is in movement". This was proved with many different pilot models, starting from an initial configuration in which the controller was idle, the door closed and the gear in.
- For a door which is opened (closed) at a rate of 2 (-2) and a gear which is extended (retracted) at a rate of 1 (-1), and for different pilots who change the landing command at different rates, several experiments were performed to check the timeliness requirements. For example, whether it is always the case that the door is closed and the gear extended within 7 seconds of issuing the landing command (and similarly for the not-landing command), if the pilot changes the command every 5 seconds.

4.4 Verification of safety properties in Esterel

A common approach to modelling and verification of discrete control systems is the discrete events approach, whereby the system is represented as parallel composition of a set of finite automata. The requirements may be represented according to different preferences (e.g. as formulas in a temporal logic or as finite automata). Although the theory for synthesis of discrete controllers has existed for a while [25], its application to verification requires obtaining natural discrete models of the plant, and practical automatic verification tools.

In this section we report on an application of these ideas within the development environment of Esterel – a language falling in the framework of synchronous languages[8]. This family of languages have the benefit of a formal semantics (as synchronous I/O machines or Mealy automata) and an intuitive appeal within the engineering community. One may question why is formal verification in such a setting worth some attention when recent advances in verification techniques are within the timed and hybrid domains? One argument is the attractiveness of code generation capabilities in the development environments for this class of languages. Once the model of a controller in interaction with a plant has been formally verified for certain properties, then the high level controller representation can be automatically translated to C, Ada or a circuit design language. Furthermore, models in Esterel and statecharts with the above underlying semantics can be bridged to models in the data flow languages of Lustre and Signal (with the associated commercial tools e.g. SAO+).

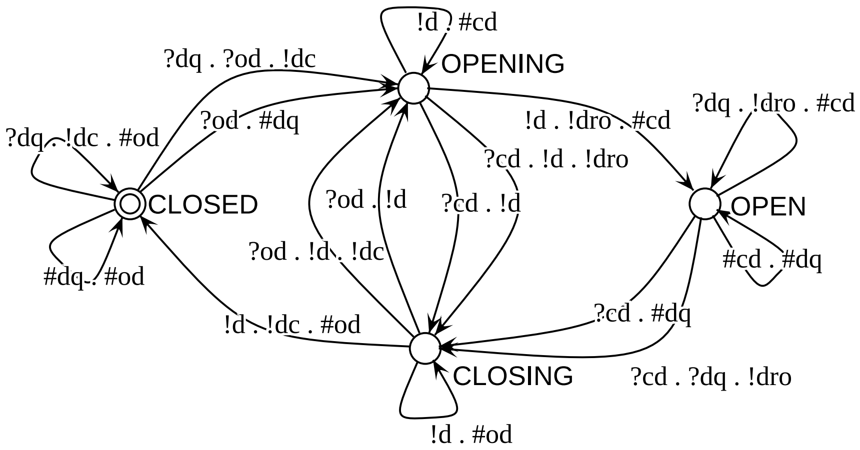
Our application of these techniques to the landing gear example rests on two basic ideas. First, that realistic discrete models of the mechanics and hydraulics can be developed once the physically driven models are available and well understood. Second, that the safety requirements can easily be encoded as synchronous observers [9] of the plant and the controller. An observer is a reactive state machine which "watches" the emitted signals of another machine. If a particular safety property is violated the observer emits an alarm (α signal). Formal verification amounts to checking that a parallel composition of the observer and the system does not emit alarms.

This approach suits verifying safety properties (i.e. "something bad will not happen"). For bounded response properties (i.e. "something good will happen within x steps") formulas of temporal logic are easier to specify. Though there exists a system which gives an automatic translation of such a property in temporal logic to a synchronous observer [12].

In [34] we report on whether the proof of a property of the system should be proved on a model of the composition of its subsystems, or that it should be split into different properties of the subsystems and proved separately for each subsystem. The study of the compositional versus decompositional proof techniques in the case of tightly interacting systems is exemplified by providing the numbers of states and transitions (and hence the size of proofs and intermediate proof steps) in both compositional and decompositional methods.

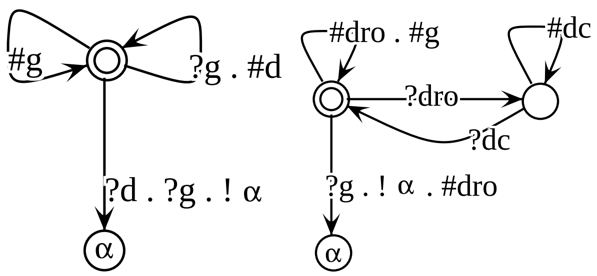
In what follows we include a number of models developed for the purpose of the study in [34]. For example Figure 4.2 depicts the door model as an I/O machine automatically obtained from the Esterel compilation process and drawn by the autograph package. Transitions on the graphs are signal operator pairs concatenated by full stops. The '!' operator represents that a signal is being output, a '?' operator represents signal input, and a '#' is an operator representing the absence of a signal from the input of a reaction.

The machine in figure 4.2 has four states, the double ringed one representing the initial state. The two states shown here on the left and right of the diagram represent the fact that the door is stationary, and open or closed respectively. The top and bottom states represent the movement of



inputs		**outputs**	
od	open door	dro	door open
cd	close door	dc	door closed
dq	door query	d	door in motion

Figure 4.2: A model of the door as an I/O machine showing input and output signals



inputs		**outputs**	
d	door in motion	α	alarm
g	gear in motion		
dro	door open		
dc	door closed		

Figure 4.3: An observer of the safety property that the door and gear do not collide. The actual observer is the parallel composition of these two machines.

the door.

The door can receive three inputs: **od** and **cd** are requests from the controller for the door to be opened and closed respectively, **dq** is a request from the controller for the door to report its state. As we can see from the transitions if the door is stationary (in states OPEN or CLOSED) and it receives a **dq**, it will report its current state with the outputs **dro** or **dc**, (door open and closed). If the door is in motion, the **dq** signal is ignored. On the transition from OPENING to the the OPEN state the door emits a **dro** without prompting, (and of course a **dc** for the corresponding CLOSING case). The signal **d** is a signal indicating the movement of the door. It is emitted spontaneously on an empty input whenever the position of the door is updated i.e. when the door moves.

The model for the gear is very similar in structure only the opening and closing actions are replaced by extending and retracting.

The property that we are interested in verifying is that the landing gear and the door will not collide. This property we prove by proving the stronger property that the door should always be open and stationary when the landing gear is in motion. This second property can be seen as two properties, that the door and gear are never moving at the same time, and that the door is always open when the gear is in motion. Notice that both properties, are properties of the composed system, and cannot hold over just the plant or controller.

These two properties can be checked by a pair of observer machines combined in parallel. Figure 4.3 shows these machines. The first simply emits an alarm (α signal) and halts whenever an event occurs containing the signals **g** and **d** at the same time, indicating that the door and gear have moved at the same time. The second machine watches the signals **dro** and **dc** which are sensor states from the plant emitted when the door is open and closed respectively. This second machine emits an alarm if **g** ever occurs when the sensors have indicated that the door is not open.

In order to check that the system satisfies the safety property, the complete observer was combined in parallel with the system, and the resulting state machine checked for the emission of α signals. This verification exercise is fully described in [34].

4.5 Discussion

Each of the approaches described above have positive and negative characteristics.

4.5.1 Comparison of tools

Our Statemate and Esterel verifications were based on standard search in the state space whereas HyTech uses a more advanced representation of the search space. This is useful for larger systems or systems with infinite state spaces. Though later versions of the Esterel environment provide translations to BDDs and thus facilitate symbolic model checking. Note that HyTech is a representative for a family of symbolic model checkers (including SMC, Uppaal, Kronos, Spin) all developed in recent years based on similar techniques but with differing levels of expressivity in the used languages. Such symbolic model checking packages can be incorporated in future commercial verification tools to make their application to realistic examples more practical.

As far as user interface is concerned, Statemate is a stable commercial tool with a reasonable (but far from perfect) user interface, whereas the other two tools are advanced research prototypes with less user support. The Esterel and Statemate environments support automatic translation to code (C, Ada or circuits) whereas HyTech is essentially an analysis tool.

Another practical aspect of the verification activity is the need for some indication of the reason for failure to comply with a requirement. As models get more complex, the compliance of the design for a dynamic system with its requirement specifications becomes more of an exception than a rule. Hence, trace facilities for finding the cause of reachability of the "bad" states is an essential ingredient. None of the tool versions examined could give diagnostic information to trace the cause of the reachability of bad states. They simply report that a bad state was reachable, and return the set of reachable bad states.

4.5.2 Modelling and language issues

Obviously none of the languages support development of the natural (hybrid) model provided earlier as an HTS in Figure 3.5. However, statecharts could be used in this particular case to get models with timers and integrators, preserving much of the structure in the HTS. This was dependent on a translation of the differential equations to difference equations.

The translation from the HTS to LHA, which was needed by HyTech, necessitates the addition of invariances at the different locations of the LHA. The motivation for invariances in the LHA formalism is the need to force progress which in turn allows the representation of upper bounds on enabling of the guards.

The operational semantics of HTS ensures progress if the guard of some outgoing transition from a mode is satisfied. The motivation there is the intrinsic (causal) determinism in physical plants and the need for reactive and deterministic behaviour in real-time controllers (i.e every input in every state has a well-defined and unique reaction). Hence, for the deterministic example we have here, the invariance at an LHA location simply becomes the "inverse" of the guard(s) for the outgoing transition(s).

Having time-determinism (the same lower and upperbound on each transition) also makes it straight forward for translation to an LHA. A clock has to be reset at the entry to each location and advanced until the deterministic bound is reached. However, if there are a lot of transitions with nondert-erministic timing constraints (different lower and upper bounds) then the LHA model becomes much more clottered than the HTS model.

The above discussion can be summarized as follows. The form and content of the model subject to analysis is determined by several factors. In particular, the property we have to prove about the model, the analysis technique adopted and the available tool have a major impact. It is therefore of utter importance that the translations between different modelling levels (languages) preserves all the necessary properties of the model. Thus modelling continues to be an important step within the formal verification process.

Chapter 5

Verifying non-linear hybrid models

In this chapter we adopt a deductive approach to verification of hybrid models. The approach builds on representation of the closed loop model and the required properties in a variant of interval temporal logic called *Extended Duration Calculus* (EDC). Then verifying that the requirements are met by the given model are achieved through proving theorems in the proof system of the logic.

In this chapter we show the details of the proofs for two of the four closed loop models presented earlier in the report. First we show how proofs in EDC can be used to verify the design consisting of the static controller and the coarse environment model. Next, we show that the proof for the closed loop system consisting of the static controller and the non-linear environment model can be carried out by just adding two extra lemmas and preserving the proof structure for the simpler plant model.

5.1 A brief introduction to EDC

We use the version of EDC in accordance with [26] in which a mathematical theory about state functions is assumed as given. The syntactic constituents of the language are *state names* ($f, g, \dot{f}, \dot{g}$, etc) denoting a real valued function on time, *boolean state names* denoting a boolean valued function on time, operators and relations on real numbers, and propositional logic operators, resulting in state expressions and state assertions.

For example, here state names $x_d, x_g, \dot{x}_d, \dot{x}_g$, boolean state names *open* and *closed*, as well as the other boolean state names are constituents in the mathematical theory over which assertions can be made. Furthermore, EDC uses the notions of 'length of an interval' and 'duration'. The length of an interval is equal to the duration of the constant boolean state *true* (which has a value 1 at any time), expressed as $\ell = \int true$. Furthermore, for Q being an assertion, the notion of Q holding over an interval (represented by $\lceil Q \rceil$) is related to durations by the following definition:

$$\lceil Q \rceil \widehat{=} (\int Q = \ell) \wedge (\ell > 0)$$

Informally speaking, state assertions are evaluated to 0 or 1 whereby their durations can be reasoned about. In particular, *true* and *false* as state assertions evaluate to 1 and 0 respectively.

Atomic duration formulas have the form $\lceil Q \rceil$ or $\triangleleft(dt_1, \ldots, dt_n)$ where $\triangleleft$ is an n-ary relation on reals and dt_i are duration terms. Duration terms have essentially the form $\int se$ (for state expression se), $\mathbf{b}.se$ and $\mathbf{e}.se$ denoting the value of the state expression se at the beginning and end of an interval respectively, and terms built with operators on reals. Compound duration formulas are build from atomic formulas using the usual logical connectives as well as the chop connective of interval tempotal logic (denoted by ;) which chops an interval in two parts. Formulas are interpreted over an interval [b,e] in an interpretation $\mathcal{I}$. An interpretation $\mathcal{I}$ assigns values to state names, types and operator symbols of the language. In particular, $\mathcal{I}(\ell)[b,e] = e - b$.

Semantics of the chop operator is defined as follows. The meaning of the formula $D_1; D_2$ in an interpretation $\mathcal{I}$ over an interval $[b, e]$ is defined as:

$$\begin{aligned} \mathcal{V}_\mathcal{I}, [b,e] \models D_1; D_2 \quad \text{iff} \quad & \mathcal{V}_\mathcal{I}, [b,m] \models D_1 \text{ and } \mathcal{V}_\mathcal{I}, [m,e] \models D_2 \\ & \text{for some } m \in [b,e]. \end{aligned}$$

A formula D is satisfiable in an interpretation $\mathcal{I}$ (written $\mathcal{V}_\mathcal{I} \models D$) iff $\mathcal{V}_\mathcal{I}, [0,t] \models D$ for all $t \in$ TIME. The set of time points TIME usually chosen to be $\mathbb{R}^{\geq 0}$. A formula D is valid (written $\models D$) iff $\mathcal{V}_\mathcal{I} \models D$ for every interpretation $\mathcal{I}$.

The following abbreviations are introduced at the formula level:

$\lceil\,\rceil$	$\widehat{=}$	$\ell = 0$	the empty interval
$\Diamond D$	$\widehat{=}$	$true; D; true$	D holds in some sub-interval
$\Box D$	$\widehat{=}$	$\neg\Diamond\neg D$	D holds in every sub-interval

Next we present a number of proof rules (laws) from the proof system of EDC [26]. which will be used in the proofs later on in the chapter.

P-And:

$$\lceil P \wedge Q \rceil \Leftrightarrow \lceil P \rceil \wedge \lceil Q \rceil$$

P-Always:

$$\lceil P \rceil \Rightarrow \Box (\lceil P \rceil \vee \lceil\, \rceil)$$

Somewhere-Neg:

$$\neg \Diamond D \Leftrightarrow \Box \neg D \text{ and } \Diamond \neg D \Leftrightarrow \neg \Box D$$

Always-intro:

$$\Box D \wedge (D_1 \,;\, D2) \Rightarrow (\Box D \wedge D_1) \,;\, (\Box D \wedge D2)$$

Always-Once-Somewhere:

$$\Box D \Rightarrow D \text{ and } D \Rightarrow \Diamond D$$

Zero:

$$\int false = 0$$

Chop-false:

$$D \,;\, false \Rightarrow false \text{ and } false \,;\, D \Rightarrow false$$

Chop-P-Or:

$$\lceil P_1 \vee P_2 \rceil \,;\, true \Leftrightarrow \lceil P_1 \rceil \,;\, true \vee \lceil P_2 \rceil \,;\, true$$

Dur-Chop:

$$(\int P = r_1) \,;\, (\int P = r_2) \Leftrightarrow \int P = r_1 + r_2$$

Chop-Exists: *provided v does not occur free in D_1*

$$D_1 \,;\, (\exists v : T \bullet D_2) \Leftrightarrow \exists v : T \bullet D_1 \,;\, D_2$$

Continuity:

$$\mathbf{e}.se = v_1 \,;\, \mathbf{b}.se = v_2 \Rightarrow v_1 = v_2$$

Note that the last law applies only to continuous state expressions. In the proofs which follow we sometimes use **analysis** where the current line can be motivated by the immediately preceding line and real arithmetic. As well as the above rules, the proofs which will follow will be using another motivation denoted by **PL**, whenever the preceding line in the proof leads to the current line according to the rules in predicate logic. We further include two general EDC results which will also be used in later proofs.

Lemma 1 *For duration formulas D_1 and D_2*

$$\Box D_1 \wedge (D_2 \,;\, D_3) \Rightarrow (D_1 \wedge D_2) \,;\, (D_1 \wedge D_3)$$

$\Box$

Proof:

$$\begin{array}{ll} & \Box\, D_1 \wedge (D_2 \,;\, D_3) \\ \Rightarrow & \{\text{Always-intro}\} \\ & (\,\Box\, D_1 \wedge D_2) \,;\, (\,\Box\, D_1 \wedge D_3) \\ \Rightarrow & \{\text{Always-Once-Somewhere}\} \\ & (D_1 \wedge D_2) \,;\, (D_1 \wedge D_3) \end{array}$$

$\Box$

Lemma 2 *For state assertions P_1, P_2 and P_3*

$$\lceil P_1 \rceil \wedge (\lceil P_2 \rceil \,;\, \lceil P_3 \rceil) \Rightarrow (\lceil P_1 \rceil \wedge \lceil P_2 \rceil) \,;\, (\lceil P_1 \rceil \wedge \lceil P_3 \rceil)$$

$\Box$

Proof:

$$\begin{array}{ll} & \lceil P_1 \rceil \wedge (\lceil P_2 \rceil \,;\, \lceil P_3 \rceil) \\ \Rightarrow & \{\text{P-Always}\} \\ & \Box\,(\lceil P_1 \rceil \vee \lceil\,\rceil) \wedge (\lceil P_2 \rceil \,;\, \lceil P_3 \rceil) \\ \Rightarrow & \{\text{Always-intro}\} \\ & (\,\Box\,(\lceil P_1 \rceil \vee \lceil\,\rceil) \wedge \lceil P_2 \rceil) \,;\, (\,\Box\,(\lceil P_1 \rceil \vee \lceil\,\rceil) \wedge \lceil P_3 \rceil) \\ \Rightarrow & \{\text{Always-Once-Somewhere}\} \\ & ((\lceil P_1 \rceil \vee \lceil\,\rceil) \wedge \lceil P_2 \rceil) \,;\, ((\lceil P_1 \rceil \vee \lceil\,\rceil) \wedge \lceil P_3 \rceil) \\ \Rightarrow & \{\text{PL}\} \\ & ((\lceil P_1 \rceil \wedge \lceil P_2 \rceil) \vee (\lceil\,\rceil \wedge \lceil P_2 \rceil)) \,;\, ((\lceil P_1 \rceil \wedge \lceil P_3 \rceil) \vee (\lceil\,\rceil \wedge \lceil P_3 \rceil)) \\ \Rightarrow & \{\text{Defn, analysis}\} \\ & ((\lceil P_1 \rceil \wedge \lceil P_2 \rceil) \vee \mathit{false}) \,;\, ((\lceil P_1 \rceil \wedge \lceil P_3 \rceil) \vee \mathit{false}) \\ \Rightarrow & \{\text{PL}\} \\ & (\lceil P_1 \rceil \wedge \lceil P_2 \rceil) \,;\, (\lceil P_1 \rceil \wedge \lceil P_3 \rceil) \end{array}$$

$\Box$

Name: Signature	*Type: Description*
$x_d : \mathbb{R} \to \mathbb{R}$	Plant state: door position (closed: 0, opened: 1)
$x_g : \mathbb{R} \to \mathbb{R}$	Plant state: gear position (retracted: 0, extended: 1)
$\alpha_d : \mathbb{R}$	Plant parameter: door speed
$\alpha_g : \mathbb{R}$	Plant parameter: gear speed
$u_{11} : \mathbb{R} \to \{0,1\}$	Controller command: open door (hold: 0, open: 1)
$u_{12} : \mathbb{R} \to \{0,1\}$	Controller command: close door (hold: 0, close: 1)
$u_{21} : \mathbb{R} \to \{0,1\}$	Controller command: extend gear (hold: 0, extend: 1)
$u_{22} : \mathbb{R} \to \{0,1\}$	Controller command: retract gear (hold: 0, retract: 1)
$r : \mathbb{R} \to \{0,1\}$	Pilot command: extend gear (retract: 0, extend: 1)

Table 5.1: Declaration of states and global variables.

5.2 EDC model of the landing gear

Table 5.1 gives the declaration of the constituents in the coarse plant model of the landing gear.

Using the above notation and the architectural breakdown in Figure 3.2, the closed loop system for the landing gear is represented by

$$S \equiv P \wedge C$$

where the plant model

$$P \equiv Env \wedge Eff$$

and the controller model

$$C \equiv Char \wedge Sel$$

Here Env, $Char$, Sel, and Eff are assertions within the underlying mathematical theory and defined in accordance with the formulas provided in the appropriate boxes in Figure 3.2. Then the behaviour of the system over an interval of time can be represented by lifted duration formulas over these assertions.

Note that the composition of the environment and effector model leads to the following two sets of constraints on the door and the gear parts of the model respectively. That is, $\lceil P \rceil \Rightarrow \lceil P_1 \rceil \wedge \lceil P_2 \rceil$.

$$P_1 \ \widehat{=} \ \begin{cases} (u_{11} \wedge \neg u_{12} & \Rightarrow \ \dot{x}_d = \alpha_d) \wedge \\ ((\neg u_{11} \wedge \neg u_{12}) \vee (u_{11} \wedge u_{12}) & \Rightarrow \ \dot{x}_d = 0) \wedge \\ (\neg u_{11} \wedge u_{12} & \Rightarrow \ \dot{x}_d = -\alpha_d) \\ 0 \leq x_d \leq 1 & \\ 0 \leq \alpha_d \leq 1 & \end{cases} \tag{5.1}$$

$$P_2 \ \widehat{=} \ \begin{cases} (u_{21} \wedge \neg u_{22} & \Rightarrow \ \dot{x}_g = \alpha_g) \wedge \\ ((\neg u_{21} \wedge \neg u_{22}) \vee (u_{21} \wedge u_{22}) & \Rightarrow \ \dot{x}_g = 0) \wedge \\ (\neg u_{21} \wedge u_{22} & \Rightarrow \ \dot{x}_g = -\alpha_g) \\ 0 \leq x_g \leq 1 & \\ 0 \leq \alpha_g \leq 1 & \end{cases} \tag{5.2}$$

Likewise, the selector and characterizer models can be combined to the give the following EDC formulation:

$$C \widehat{=} \tag{5.3}$$

$$\begin{cases} (r > 0 \wedge \neg(x_g \geq 1)) \vee (r \leq 0 \wedge \neg(x_g \leq 0)) \wedge \neg(x_d \geq 1) \wedge \neg u_{12} & \Leftrightarrow \ u_{11} \\ ((r > 0 \wedge (x_g \geq 1)) \vee (r \leq 0 \wedge (x_g \leq 0))) \wedge \neg(x_d \leq 0) \wedge \neg u_{11} & \Leftrightarrow \ u_{12} \\ r > 0 \wedge (x_d \geq 1) \wedge \neg(x_g \geq 1) \wedge \neg u_{22} & \Leftrightarrow \ u_{21} \\ r \leq 0 \wedge (x_d \geq 1) \wedge \neg(x_g \leq 0) \wedge \neg u_{21} & \Leftrightarrow \ u_{22} \end{cases}$$

Given the static controller model, for example, it is possible to verify e.g.

$$\lceil C \rceil \Rightarrow \lceil \neg(u_{11} \wedge u_{12}) \rceil \tag{5.4}$$

$$\lceil C \rceil \Rightarrow \lceil \neg(u_{21} \wedge u_{22}) \rceil \tag{5.5}$$

i.e. open and close (extend and retract) commands can never be issued simultaneously. Also, the overall system model can be summarized by the following invariant

$$\Box \, (\neg \lceil \neg S \rceil \wedge continuous(x_d) \wedge continuous(x_g)).$$

Note that different mathematical models for the closed loop system are simply obtained by replacing some assertion with a new one. This will be illustrated in more detail in section 5.6.

5.3 Requirement specifications in EDC

We now present the EDC formulations of the two closed loop system properties which were presented in the last chapter. First we consider the safety property, i.e. the fundamental requirement that the gear must never move when the door is not fully open. This requirement R_1 may be formalized as

$$R_1 \quad \equiv \quad \Box \neg \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \tag{5.6}$$

The second performance (or timeliness) property requires that the extension (or retraction) to be completed within T time units, is presented as follows. These requirements may now be formalized by the two formulae R_2 and R_3.

$$R_2 \quad \equiv \quad \Box \neg((\lceil r > 0 \rceil \wedge \ell = T) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil) \tag{5.7}$$

$$R_3 \quad \equiv \quad \Box \neg((\lceil r \leq 0 \rceil \wedge \ell = T) \,;\, \lceil \neg(x_g \leq 0 \wedge x_d \leq 0) \rceil) \tag{5.8}$$

In other words, it is never the case that the landing (no landing) command is in operation for an interval of length T and the desired effect has not been achieved immediately after the interval. In order to verify these properties we thus have to prove

$$\lceil S \rceil \Rightarrow R_1 \wedge R_2 \wedge R_3$$

In the following two sections we give the proofs for requirements R_1 and R_2. The proof for R_3 is analogous to the one for R_2 and is therefore omitted.

5.4 Verification of R_1

In order to verify the first requirement we employ proof by contradiction within the proof system of EDC [26]. But before that we need to state the following lemma which gives a property of the application model as presented in Section 5.2.

Lemma 3 $\lceil P \wedge C \rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \Rightarrow false$ □

Proof:

$$
\begin{array}{ll}
 & \lceil P \wedge C \rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{P-And}\} \\
 & \lceil P \wedge C \wedge \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{5.2, PL}\} \\
 & \lceil C \wedge \neg((\neg u21 \wedge \neg u22) \vee (u21 \wedge u22)) \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{PL}\} \\
 & \lceil C \wedge ((u21 \wedge \neg u22) \vee (\neg u21 \wedge u22)) \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{5.3, PL}\} \\
 & \lceil ((x_d \geq 1) \vee (x_d \geq 1)) \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{PL}\} \\
 & \lceil false \rceil \\
\Rightarrow & \{\text{Defn}\} \\
 & \int false = \ell > 0 \\
\Rightarrow & \{\text{Zero}\} \\
 & 0 = \ell > 0 \\
\Rightarrow & \{\text{analysis}\} \\
 & false
\end{array}
$$

□

Next we present the proof of the safety property.

Theorem 1

$$\lceil S \rceil \wedge \neg R_1 \Rightarrow false$$

□

Proof:

$$
\begin{array}{ll}
 & \lceil P \wedge C \rceil \wedge \neg \Box \neg \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{Somewhere-Neg, PL}\} \\
 & \lceil P \wedge C \rceil \wedge \Diamond \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{Defn}\} \\
 & \lceil P \wedge C \rceil \wedge true \mathbin{;} \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \mathbin{;} true \\
\Rightarrow & \{\text{P-Always}\} \\
 & \Box (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge (true \mathbin{;} \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \mathbin{;} true) \\
\Rightarrow & \{\text{Lemma 1}\} \\
 & ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge true) \mathbin{;} ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil) \mathbin{;} \\
 & ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge true) \\
\Rightarrow & \{\text{Lemma 3, PL}\} \\
 & (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \mathbin{;} (\lceil\,\rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil) \mathbin{;} (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \\
\Rightarrow & \{\text{Defn}\} \\
 & (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \mathbin{;} false \mathbin{;} (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \\
\Rightarrow & \{\text{Chop-false}\} \\
 & false
\end{array}
$$

$\Box$

5.5 Verification of R_2

To prove the timeliness property we first need to make the structure of the application more explicit. This conceptualization then provides the necessary structure over which a case-based proof can be carried out.

First, going back to our closed loop model as depicted in Figure 3.2 and the mathematical formulation of the controller in equation (5.3) we note that the controller represents the combination of a static selector and a static characterizer. Whereas the plant equations (5.1) and (5.2) can be seen as a combination of the static effector and the dynamic environment model.

Let's consider what the HTS closed loop model for this configuration would look like[1]. With no knowledge about the controller, the model of the environment could be represented by a hybrid transition system with two differential equations in each mode. The derivatives of the door and gear

[1]The analysis which follows here performs a kind of application-based simplification (minimization) of the HTS model which could have been formally obtained. In general, we need formal rewrite rules and tools to obtain the simplest form for each application.

positions ($\dot{x}_i$) could then take any of the values $(-\alpha_i, 0, \alpha_i)$ for $i \in \{d, g\}$ in different modes. The system would then have 9 modes and would change its mode depending on various choices of effector output. There would therefore be 8 transitions into each mode from all the other 8 modes, and 8 transitions out of each mode each going to one of the other 8 modes.

The model of the closed loop system, however, contains more structure. First, not all 9 combinations of effector output are allowed by the combination of the effector and the selector. In other words, the composition of the characterizer, selector and effector mappings is not surjective. To see this note that if the effector is seen as a function

$$\mathcal{E} : \mathbb{B}^4 \to \{\langle u_d, u_g \rangle \mid u_d \in \{-\alpha_d, 0, \alpha_d\} \text{ and } u_g \in \{-\alpha_g, 0, \alpha_g\}\}$$

then it is surjective. However, the selector mapping $\mathcal{S} : \mathbb{B}^5 \to \mathbb{B}^4$ is not surjective. More specifically, some combinations of $\langle u_{11}, u_{12}, u_{21}, u_{22} \rangle$ are not in the range of the selector function. For example, u_{11} and u_{12} are not allowed to be true at the same time (see equation 5.4). Also, if the landing command is active then based on the selector equations it is possible to eliminate some other combinations of u_{ij}. The following lemma formalizes this property of the selector-effector composition. Note that the state regions S_i provide a covering of the state space in presence of the landing command.

Lemma 4 *Let S_i be defined according to the following combinations of state variables and pilot command:*

$$\begin{array}{rcl}
S_1 & \equiv & (((0 < x_d < 1) \wedge (0 < x_g < 1)) \vee \\
 & & ((x_d \leq 0) \wedge (0 < x_g < 1)) \vee \\
 & & ((x_d \leq 0) \wedge (x_g \leq 0)) \vee \\
 & & ((0 < x_d < 1) \wedge (x_g \leq 0))) \wedge r > 0 \\
S_2 & \equiv & (((x_d \geq 1) \wedge (0 < x_g < 1)) \vee \\
 & & ((x_d \geq 1) \wedge (x_g \leq 0))) \wedge r > 0 \\
S_3 & \equiv & (((0 < x_d < 1) \wedge (x_g \geq 1)) \vee \\
 & & ((x_d \geq 1) \wedge (x_g \geq 1))) \wedge r > 0 \\
S_4 & \equiv & (x_d \leq 0) \wedge (x_g \geq 1) \wedge r > 0
\end{array}$$

Then we have

$$\lceil P \rceil \wedge \lceil r > 0 \rceil \Rightarrow \lceil S_1 \vee S_2 \vee S_3 \vee S_4 \rceil$$

□

Lemma 5 *Let S_i be defined according to Lemma 4. Then the following statements can be derived using propositional logic:*

$$\lceil S_1 \wedge C \rceil \Rightarrow \lceil u_{11} \wedge \neg u_{12} \wedge \neg u_{21} \wedge \neg u_{22} \rceil$$

$$\lceil S_2 \wedge C \rceil \Rightarrow \lceil \neg u_{11} \wedge \neg u_{12} \wedge u_{21} \wedge \neg u_{22} \rceil$$

$$\lceil S_3 \wedge C \rceil \Rightarrow \lceil \neg u_{11} \wedge u_{12} \wedge u_{21} \wedge \neg u_{22} \rceil$$

$$\lceil S_4 \wedge C \rceil \Rightarrow \lceil \neg u_{11} \wedge \neg u_{12} \wedge \neg u_{21} \wedge \neg u_{22} \rceil$$

□

Note that the above result implies that while the landing command is in operation, no combinations of u_{ij} other than those listed above are possible. This is a direct corollary to Lemma 4. It rests on the fact that the domain of the function composed from the characterizer and selector functions has been completely covered by the conditions on the left hand side of $\Rightarrow$ in the above formulas. Based on this result and the effector mapping described in P_1 and P_2, we can now state the following lemma, again proved using propositional logic and Lemma 5.

Lemma 6 *Let S_i be defined according to Lemma 4. Then we have*

$$\lceil S_1 \rceil \wedge \lceil P \wedge C \rceil \Rightarrow \lceil \dot{x}_d = \alpha_d \wedge \dot{x}_g = 0 \rceil$$

$$\lceil S_2 \rceil \wedge \lceil P \wedge C \rceil \Rightarrow \lceil \dot{x}_d = 0 \wedge \dot{x}_g = \alpha_g \rceil$$

$$\lceil S_3 \rceil \wedge \lceil P \wedge C \rceil \Rightarrow \lceil \dot{x}_d = -\alpha_d \wedge \dot{x}_g = 0 \rceil$$

$$\lceil S_4 \rceil \wedge \lceil P \wedge C \rceil \Rightarrow \lceil \dot{x}_d = 0 \wedge \dot{x}_g = 0 \rceil$$

□

This lemma provides the necessary structure over which the proof of the timeliness property R_2 can be carried out. To make the structure of the proof for the main theorem more visible we present a fraction of a hybrid transition system which can be derived for the closed loop system. Each mode in this system corresponds to a region in the continuous state space given that the pilot command is in operation[2].

[2] A similar but slightly different fraction can also be derived for the case when $r \leq 0$ holds, in order to prove the property R_3.

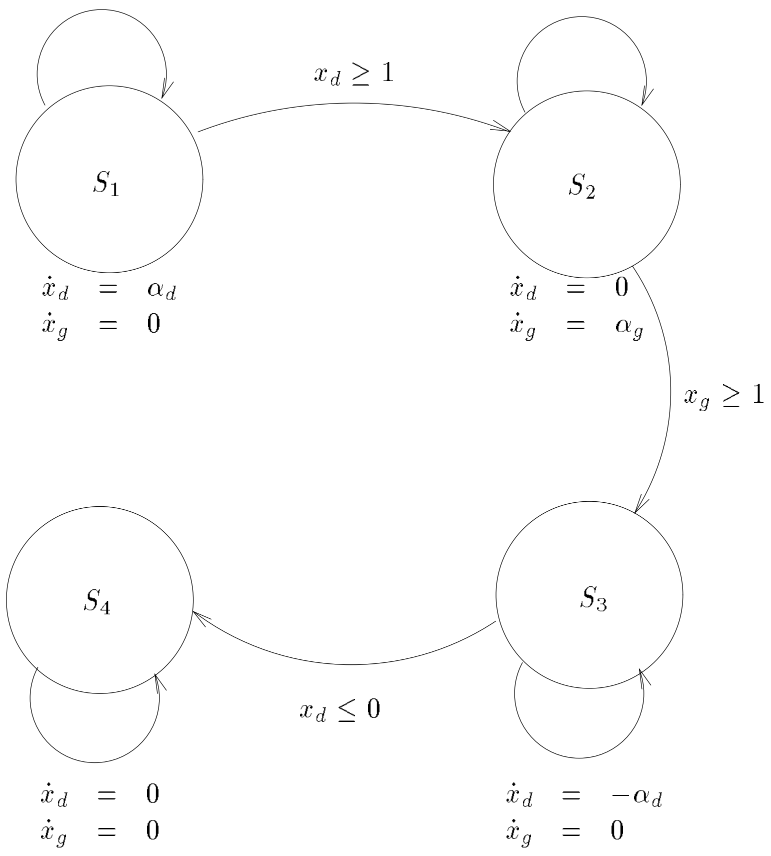


Figure 5.1: State space regions of the closed loop system with landing command in operation.

Figure 5.1 associates one mode with each of the state space regions S_i in Lemma 4 above. Each mode in the graph is annotated with the corresponding door and gear equations derived in Lemma 6. Note that reflexive transitions have an implicit "otherwise" condition. They have been drawn to aid visualization: if the guard of the outgoing transition is not true the system remains in the current S_i region. In particular, the system will remain in the S_4 region unless the landing command is altered – a transition condition for moving out to the other half of the model which has not been depicted.

Based on this pictorial representation and using the definition of regions S_i we can justify the choice of arcs in the graph together with their respective labels. This analysis is formalized in the following EDC formulation.

Lemma 7 *For S_i as defined in Lemma 4 we have:*

(1) $(\lceil S_1 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \Rightarrow (\lceil S_1 \rceil \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, true))$

(2) $(\lceil S_2 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \Rightarrow (\lceil S_2 \rceil \vee (\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true))$

(3) $(\lceil S_3 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \Rightarrow (\lceil S_3 \rceil \vee (\lceil S_3 \rceil \,;\, \lceil S_4 \rceil \,;\, true))$

(4) $(\lceil S_4 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \Rightarrow \lceil S_4 \rceil$ *and* $\lceil S_4 \rceil \Rightarrow (true \,;\, \lceil S_4 \rceil)$

□

The first formula, for example, states that if S_1 holds during a sub-interval which starts an interval, then it will either continue to hold during the rest of the interval, or the sub-interval will be followed by an adjacent sub-interval in which S_2 holds.

Proof: By straight forward mathematical analysis, we can motivate the transition from S_1 to S_2 as follows: based on Lemma 6 the value of x_g in the region S_1 is constant. Therefore, during every interval in which the landing command remains constant $(r > 0)$, if the state of the system should move out of this region, it should do so due to the value of x_d. That is, x_d should increase beyond the bounds set by the region. This will take the system to state S_2 with $x_d \geq 1$ as a condition. Similar reasoning can be performed for other transitions. □

To give the proof for satisfiability of R_2 we need to make the following assumption on the constants α_d and α_g. Alternatively stated, we obtain these additional constraints as a by-product of the main proof. We assume that $2/\alpha_d + 1/\alpha_g \leq T$. Now we proceed with two more results before the main proof is presented. First, we show that niether of $S_1, \ldots, S_3$ can last for as long as T time units.

Lemma 8 *Let S_i be defined as in Lemma 4. Then we have:*

(1) $\lceil S_1 \rceil \wedge \ell = T \Rightarrow false$

(2) $\lceil S_2 \rceil \wedge \ell = T \Rightarrow false$

(3) $\lceil S_3 \rceil \wedge \ell = T \Rightarrow false$

$\square$

Proof: Below we show the proof for (1). (2) and (3) are proved analogously.

$$
\begin{array}{ll}
 & \lceil S_1 \rceil \wedge \ell = T \\
\Rightarrow & \{\text{Lemma 6}\} \\
 & \lceil \dot{x}_d = -\alpha_d \wedge \dot{x}_g = 0 \rceil \wedge \ell = T \\
\Rightarrow & \{\text{analysis}\} \\
 & \ell \leq 1/\alpha_d \wedge \ell = T \\
\Rightarrow & \{\text{analysis}\} \\
 & false
\end{array}
$$

$\square$

Next, we show that reaching region S_4 contradicts violation of R_2.

Lemma 9 *Let S_4 be defined as in Lemma 4. Then we have:*

$$(\lceil P \wedge C \rceil \wedge true \,;\, \lceil S_4 \rceil) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \Rightarrow false$$

$\square$

Proof:

$$
\begin{array}{ll}
& (\lceil P \wedge C \rceil \wedge true \; ; \lceil S_4 \rceil) \; ; \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{Lemma 2, Lemma 6}\} \\
& (\lceil P \wedge C \rceil \wedge true) \; ; (\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil \wedge \lceil \dot{x}_d = 0 \wedge \dot{x}_g = 0 \rceil) \; ; \\
& \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{analysis, PL}\} \\
& \lceil P \wedge C \rceil \; ; \\
& (\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil \wedge (\exists v_1 v_2 : \mathbb{R} \bullet \mathbf{b}.x_g = \mathbf{e}.x_g = v_1 \wedge \mathbf{b}.x_d = \mathbf{e}.x_d = v_2)) \; ; \\
& \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{Chop-Exists}\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet \lceil P \wedge C \rceil \; ; \\
& (\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil \wedge \mathbf{b}.x_g = \mathbf{e}.x_g = v_1 \wedge \mathbf{b}.x_d = \mathbf{e}.x_g = v_2) \; ; \\
& \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Continuity}\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet \lceil P \wedge C \rceil \; ; \\
& (\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil \wedge \mathbf{b}.x_g = \mathbf{e}.x_g = v_1 \wedge \mathbf{b}.x_d = \mathbf{e}.x_d = v_2) \; ; \\
& (\mathbf{b}.x_g = v_1 \wedge \mathbf{b}.x_d = v_2 \wedge \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil)) \\
\Rightarrow & \{\text{Defn } S_4\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet \lceil P \wedge C \rceil \; ; (v_1 \geq 1 \wedge v_2 \leq 0) \; ; \\
& (\mathbf{b}.x_g = v_1 \wedge \mathbf{b}.x_d = v_2 \wedge \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil)) \\
\Rightarrow & \{\text{analysis}\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet \lceil P \wedge C \rceil \; ; (v_1 \geq 1 \wedge v_2 \leq 0) \; ; false) \\
\Rightarrow & \{\text{Chop-Exists}\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet \lceil P \wedge C \rceil \; ; (v_1 \geq 1 \wedge v_2 \leq 0)) \; ; false \\
\Rightarrow & \{\text{Chop-false}\} \\
& false
\end{array}
$$

□

We are now in a position to carry out a complete proof of property R_2.

Theorem 2 *Let* $2/\alpha_d + 1/\alpha_g \leq T$. *Then* $\lceil S \rceil \wedge \neg R_2 \Rightarrow false$ □

Proof:

$$\begin{array}{ll}
& \lceil P \wedge C \rceil \wedge \neg \Box \neg ((\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Somewhere-Neg, PL}\} \\
& \lceil P \wedge C \rceil \wedge \Diamond ((\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Defn}\} \\
& \lceil P \wedge C \rceil \wedge (true \, ; (\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil \, ; true) \\
\Rightarrow & \{\text{P-Always}\} \\
& \Box (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge \\
& (true \, ; (\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil \, ; true) \\
\Rightarrow & \{\text{Lemma 1, PL}\} \\
& (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \, ; \\
& (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge ((\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \, ; \\
& (\lceil\,\rceil \vee \lceil P \wedge C \rceil)
\end{array}$$

Considering the middle intervals:

$$\begin{array}{ll}
& (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge ((\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{PL}\} \\
& (\lceil\,\rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \vee \\
& (\lceil P \wedge C \rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Defn}\} \\
& false \vee (\lceil P \wedge C \rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \, ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 2, PL}\} \\
& (\lceil P \wedge C \rceil \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{P-And, Lemma 4}\} \\
& (\lceil P \wedge C \rceil \wedge \lceil S_1 \vee S_2 \vee S_3 \vee S_4 \rceil \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{??\} \\
& \lceil P \wedge C \rceil \wedge \\
& (\lceil S_1 \rceil \, ; true \vee \lceil S_2 \rceil \, ; true \vee \lceil S_3 \rceil \, ; true \vee \lceil S_4 \rceil \, ; true) \wedge \\
& \lceil r > 0 \rceil \wedge \ell = T \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)
\end{array}$$

We prove each case separately, starting from the case where S_4 holds at the beginning of the first subinterval:

Case 1:

$$
\begin{aligned}
& (\lceil P \wedge C \rceil \wedge (\lceil S_4 \rceil \, ; true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (true \, ; \lceil S_4 \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 9}\} \\
& false
\end{aligned}
$$

The next three cases cover the possibilities that the first subinterval is started by S_3, S_2, and S_1 respectively.
Case 2:

$$
\begin{aligned}
& (\lceil P \wedge C \rceil \wedge (\lceil S_3 \rceil \, ; true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_3 \rceil \vee (\lceil S_3 \rceil \, ; \lceil S_4 \rceil \, ; true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 8, PL}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_3 \rceil \, ; \lceil S_4 \rceil \, ; true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_3 \rceil \, ; true \, ; \lceil S_4 \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \, ; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow \; & \{\text{Lemma 9, Chop-false}\} \\
& false
\end{aligned}
$$

Case 3:

$$
\begin{array}{rl}
& (\lceil P \wedge C \rceil \wedge (\lceil S_2 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_2 \rceil \vee (\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 8, PL}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge ((\lceil S_2 \rceil \,;\, \lceil S_3 \rceil) \vee (\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, \lceil S_4 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge ((\lceil S_2 \rceil \,;\, \lceil S_3 \rceil) \vee (\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true \,;\, \lceil S_4 \rceil)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 9, Chop-false}\} \\
& ((\lceil S_2 \rceil \,;\, \lceil S_3 \rceil \vee false) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{analysis, PL}\} \\
& (\ell \leq 1/\alpha_g \,;\, \ell \leq 1/\alpha_d \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Dur-Chop, analysis}\} \\
& false \,;\, (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Chop-false}\} \\
& false
\end{array}
$$

Case 4:

$$
\begin{array}{ll}
& (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S1 \rceil \vee (\lceil S1 \rceil \,;\, \lceil S_2 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 8, PL}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7}\} \\
& (\lceil P \wedge C \rceil \wedge (((\lceil S_1 \rceil \,;\, \lceil S_2 \rceil) \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true)) \wedge \\
& \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{analysis, PL}\} \\
& (\lceil P \wedge C \rceil \wedge \\
& ((\ell \leq 1/\alpha_d \,;\, \ell \leq 1/\alpha_g \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Dur-Chop, analysis}\} \\
& (\lceil P \wedge C \rceil \wedge (false \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7, PL}\} \\
& (\lceil P \wedge C \rceil \wedge \\
& (((\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil) \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, \lceil S_4 \rceil \,;\, true)) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{analysis}\} \\
& (\lceil P \wedge C \rceil \wedge \\
& ((\ell \leq 1/\alpha_d \,;\, \ell \leq 1/\alpha_g \,;\, \ell \leq 1/\alpha_d \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, \lceil S_4 \rceil \,;\, true)) \wedge \\
& \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Dur-Chop, analysis}\} \\
& (\lceil P \wedge C \rceil \wedge (false \vee (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, \lceil S_4 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7, PL}\} \\
& (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \,;\, \lceil S_2 \rceil \,;\, \lceil S_3 \rceil \,;\, true \,;\, \lceil S_4 \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 9, Chop-false}\} \\
& false
\end{array}
$$

Based on the last four cases and through two further applications of **Chop-false** we can conclude the proof for

$$\lceil S \rceil \wedge \neg R_2 \Rightarrow false$$

$\square$

5.6 Verification of the non-linear hybrid model

We now present the additional proof steps necessary to show that the same properties hold in presence of the non-linear environment model. Let the environment model in Env be replaced by Env' to give the mathematical model S'. With this refined physical model of the hydraulic supply system (corresponding to the bond graph to the right of Figure 2.5) we have:

$$\begin{aligned}
\dot{x}_d &= \alpha'_d \, p \, u_d \\
\dot{x}_g &= \alpha'_g \, p \, u_g \\
\dot{p} &= -(\gamma + \gamma_d \, |u_d| + \gamma_g \, |u_g|) \, p + \beta
\end{aligned}$$

where $\alpha'_d, \alpha'_g, \beta, \gamma, \gamma_d, \gamma_g > 0$ are constants, $x_d, x_g \in [0, 1]$ are the positions of the door and the gear actuators as before and where p is the hydraulic supply pressure.

First we note that the proofs for lemma 3 and Theorem 1 are not affected by the new plant model. Hence, the safety property continues to hold.

Next, we study the proof of the timeliness property R_2. The coarse model S was shown to have this property under the assumption that $2/\alpha_d + 1/\alpha_g \leq T$. We need similar assumptions in the case of S', only the difference is that due to the higher complexity in the model more insights from the physical world are needed to formulate and validate the assumptions. More specifically, the new proof requires that the values of the constants $\gamma, \gamma_d, \gamma_g, \beta$ and the pressure level p are made explicit. But before that we review the impact of the new plant model on the existing lemmas. We note that lemma 5 still holds since it is based on the combination of the characterizer and the selector which remain unchanged. Lemma 6 will be restated as the following lemma taking account of Env' equations.

Lemma 10 *Let S_i be defined according to Lemma 4. Let $P' \widehat{=} Env' \wedge Eff$. Then we have*

$$\lceil S_1 \rceil \wedge \lceil P' \wedge C \rceil \Rightarrow \lceil \dot{x_d} = \alpha'_d p \wedge \dot{x_g} = 0 \rceil \wedge \lceil \dot{p} = -(\gamma + \gamma_d) p + \beta \rceil$$

$$\lceil S_2 \rceil \wedge \lceil P' \wedge C \rceil \Rightarrow \lceil \dot{x_d} = 0 \wedge \dot{x_g} = \alpha'_g p \rceil \wedge \lceil \dot{p} = -(\gamma + \gamma_g)p + \beta \rceil$$

$$\lceil S_3 \rceil \wedge \lceil P' \wedge C \rceil \Rightarrow \lceil \dot{x_d} = -\alpha'_d p \wedge \dot{x_g} = 0 \rceil \wedge \lceil \dot{p} = -(\gamma + \gamma_d)p + \beta \rceil$$

$$\lceil S_4 \rceil \wedge \lceil P' \wedge C \rceil \Rightarrow \lceil \dot{x_d} = 0 \wedge \dot{x_g} = 0 \rceil \wedge \lceil \dot{p} = -\gamma p + \beta \rceil$$

□

The above lemma can then be used to redraw the Figure 5.1 to get a similar figure which covers the state space of S' in presence of $r > 0$. This is done by simply replacing the differential equations in Figure 5.1 with the equations on the right hand side of implications in Lemma 10. Based on the above lemma, Lemma 9 will continue to hold for S'. The new structure and the following lemma can then be used to show that the earlier result about S stated in Lemma 7 continues to hold for S'.

Lemma 11 *Let* $S' \equiv P' \wedge C$. *Let* $\hat{\gamma} = max\,(\gamma + \gamma_d, \gamma + \gamma_g)$. *Let* $p(0) \geq \hat{p} = \beta/\hat{\gamma}$. *Then* $p(t) \geq \hat{p} > 0$ *for all* $t \geq 0$. □

Proof: at the same time (Lemma 10). Based on physical knowledge we know that β is positive constant dependable on the engine velocity ω (for a particular ω we get a particular β); also that γ_d and γ_g are positive. We show that if p is positive to begin with, it will remain positive under all circumstances. We consider two cases.
Case 1: $p(0) = \hat{p}$. Then $\dot{p} = 0$ and hence $p = \hat{p}$ at all times. This case reduces the problem to the coarse model analysed earlier.
Case 2: $p(0) > \hat{p}$. Then $\dot{p} < 0$ and hence p will eventually reach $\hat{p}$, in which case the argument under case 1 will continue to hold.

□

Thus, $\hat{p}$ is the minimum value that p can take if the system is initialized with adequate pressure. Next we show that the positions of the door and gear will monotonically increase (decrease) in every S_i region.

Lemma 12 *For any interval in which* u_d *and* u_g *have fixed values,* p *is monotonic in* t. □

Therefore, the earlier transition relations between the regions hold even in the new system S'. That is, Lemma 7 holds also for S'. The last two results can be used to derive the upperbound for the duration of staying in each region in terms of the minimal pressure value $\hat{p}$.

Theorem 3 *Let* $\frac{2}{\hat{p}\alpha'_d} + \frac{1}{\hat{p}\alpha'_g} \leq T$. *Then*

$$\lceil S' \rceil \wedge \neg R_2 \Rightarrow false$$

$\Box$

Proof: We follow the same proof structure as in Theorem 2 only with the initial assumption that $\frac{2}{\hat{p}\alpha'_d} + \frac{1}{\hat{p}\alpha'_g} \leq T$. In addition, we systematically replace the references to Lemma 6 and the associated right hand side differential equations with references to Lemma 10 and the new equations from Env' as given in Lemma 10. $\Box$

5.7 Discussion

Going back to our generic hybrid architecture, the work reported in this chapter illustrates a possible technique when the complexities and the dynamics of the closed loop system lies in the lower half (the environment model). Typical for these cases is that a fair amount of the formal verification is mathematical analysis over the real-valued variables. This can be contrasted with the methods illustrated in chapter 4 which were more suited to cases where the complexity arises due to the size or non-deterministic timing characteristics of the upper half of the architecture (discrete controllers). This study is to our knowledge the first application of formal deductive proofs to hybrid systems with non-linear DAEs.

It should be obvious from the illustration of the method that the EDC formulation of the closed loop model, using the architectural decomposition and the mathematical models for each component, were easy to obtain.

The EDC proofs of course require a general familiarity with constructing logical proofs. However, having this familiarity, the use of EDC proof system was not particularly difficult. Nevertheless, since similar proof tactics (contradiction, case analysis, etc) tend to appear in different applications, the use of mechanized proof assistants is recommendable. Thus, the next step for making this type of analysis worthwhile in a larger example is the use of theorem provers. A mechanical prover for Duration Calulus implemented on top of PVS [23] exists and will be subject study in future works [27].

Another reflection over the work performed is the close connection between the region based transition system (Figure 5.1) and the structure of the proof. Much of the literature on formal verification assumes that the model (axiomatization) of the system to be analyzed exists apriori. This formalization is additionally assumed to be in a form suitable for application of the given proof technique. HyTech could for example analyze the transition system based on the coarse model automatically, thereby eliminating the need for the hand proof in the time-linear model. However, the exercise illustrates that the very derivation of this model may be nontrivial – or rather, once a satisfactory model is found the proof work is more or less routine. Moreover, the insights gained while modelling can be reused for verifying later refinements of the model (e.g. for the non-linear case). Thus modelling remains a cornerstone of the verification activities even in this class of models.

Chapter 6

Concluding remarks

In this report we have studied the application of a range of modelling and verification techniques to a fictitious aircraft landing gear system consisting of mechanical, hydraulical, electronic and software components.

We have initiated the study at the informal engineering documents describing the physical components, derived bond graph and mathematical models for the environment systematically, and then combined them with the mathematical model of the discrete controller(s).

We used a generic architecture to derive alternative closed loop models based on alternative plant and controller models. The plant models ranged from simple time-linear models to non-linear models allowing variable hydraulic pressure.

We also illustrated different formal verification techniques, ranging from state space search in Statemate to hand proofs in Extended Duration Calculus to arrive at proofs of safety and timeliness properties of the different closed loop models. While symbolic model checking techniques could be used for verifying complex timed dynamic controllers with a simple model of the environment, the proof with the next (refined) model of the environment necessitated deductive reasoning in a real-time temporal logic (EDC).

The experience with the different modelling and verification techniques confirms that a principled derivation of the models remains a cornerstone of the verification activities. For smaller systems it is possible to manually translate between model types while preserving their essential properties; but applications in an industrial setting require more support tools for derivation and management of models.

Appendix

In this appendix we provide a series of plots obtained from a specification of the landing gear example in the tool Statemate. These statecharts are commented upon in section 4.2 of chapter 4.

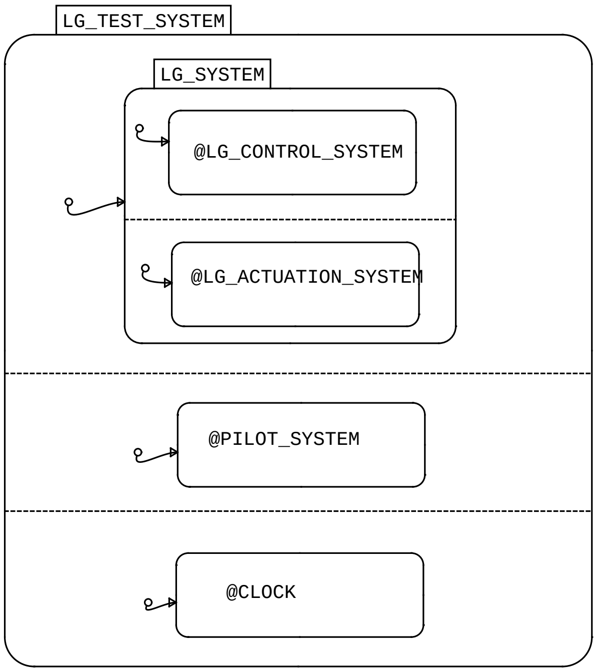


Figure 6.1: The top level model of the landing gear system in Statemate.

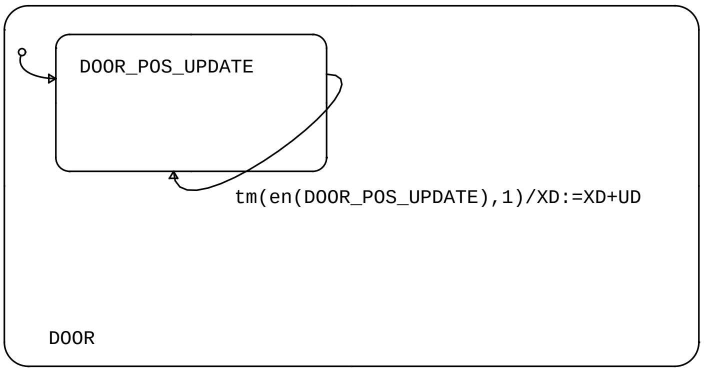


Figure 6.2: Modelling of the door in Statemate.

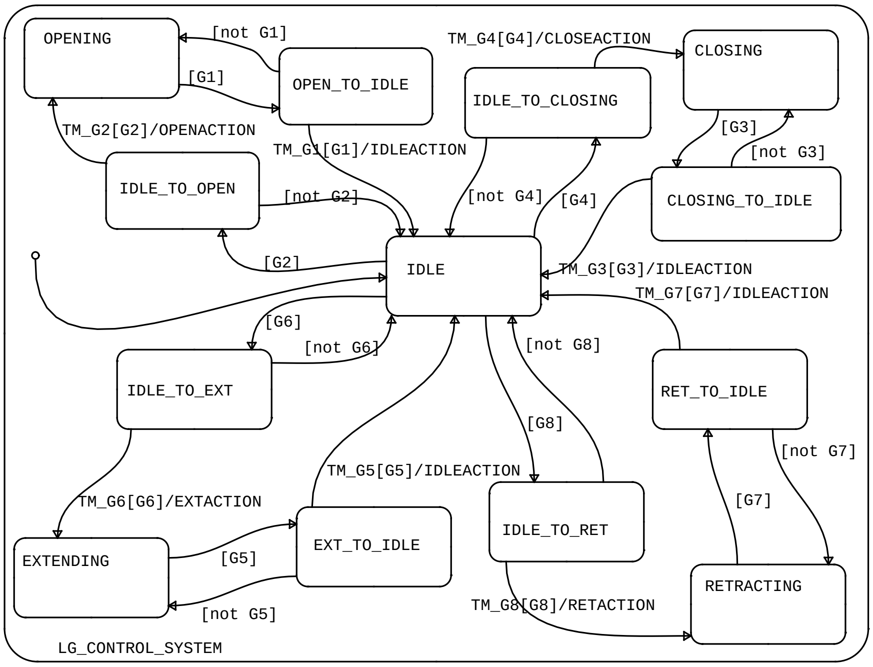


Figure 6.3: The model of the timed dynamic controller in Statemate.

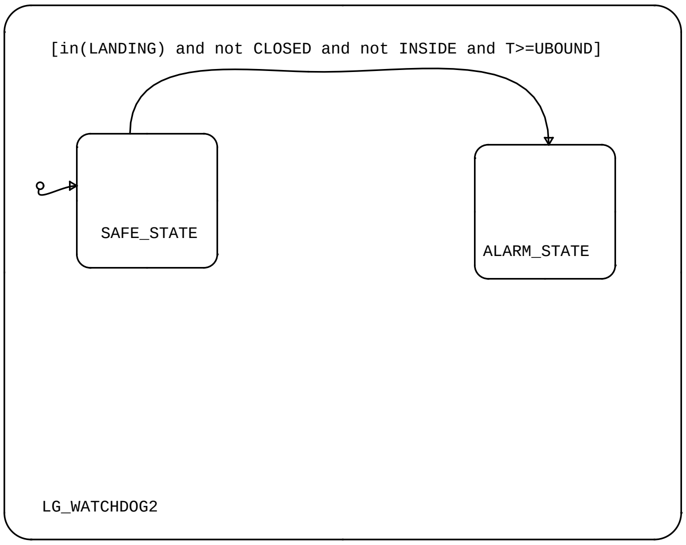


Figure 6.4: A Statemate watchdog implementing an observer for the timeliness property.

Bibliography

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P-H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In Grossman et al. [7], pages 209–229.

[2] R. Alur, T.A. Henzinger, and P-H. Ho. Automatic Symbolic Verification of Embedded Systems. In *Proc. of 14th Annual Real Time Systems Symposium*, pages 2–11. Computer Society Press, 1993.

[3] M. Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems.* PhD thesis, Lund Institute of Technology, Lund, 1994. Dissertation no. ISRN LUTFD2/TFRT-1043-SE.

[4] Z. Chaochen, A. P. Ravn, and M. R. Hansen. An Extended Duration Calculus for Hybrid Real-Time Systems. In Grossman et al. [7], pages 36–59.

[5] H. Elmqvist. *Dymola User's Manual.* Dynasim AB, Lund, 1994.

[6] R. Germundsson, J. Gunnarsson, A. Jansson, P. Krus, M. Morin, S. Nadjm-Tehrani, J. Plantin, M. Sethson, and J-E. Strömberg. Complex Hybrid Systems I, A Study of Available tools and Specification of Planned Work. Technical Report LITH-IDA-R-94-29, Dept. of Computer and Information Science, Linköping University, September 1994.

[7] R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Proc. Workshop on Theory of Hybrid Systems, October 1992, LNCS 736*, Lyngby, Denmark, 1993. Springer Verlag.

[8] N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.

[9] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Third Int. Conf. on Algebraic Methodology and Software Technology, AMAST'93*, Twente, June 1993. Workshops in Computing, Springer Verlag.

[10] D. Harel and A. Naamad. The Statemate Semantics of Statecharts. *ACM Transactions on Software Engineering Methods*, 1996.

[11] T.A. Henzinger, Z. Manna, and A. Pnueli. Timed Transition Systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G.Rozenberg, editors, *Real-Time: Theory in Practice, Proc. REX workshop 1991*, pages 226–251. Springer Verlag, LNCS 600, 1992.

[12] L.J. Jagadeesan, C. Puchol, and J.E. Von Olnhausen. A Formal Approach to Reactive Systems Software: A Telecommunications Application in Esterel. *Formal Methods in System Design*, 8:123–151, 1996.

[13] D.C. Karnopp, R.C. Rosenberg, and D. Margolis. *System dynamics - A unified approach (2nd edition)*. John Wiley & Sons, New York, 1990.

[14] H. Langmaack, W.-P. de Roever, and J. Vytopil, editors. *Proc. of the 3rd. International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 863*. Springer Verlag, 1994.

[15] S.E. Mattsson and M. Andersson. Omola - An object oriented modeling language. In Jamshidi and Herget, editors, *Recent advances in computer-aided control systems engineering*, volume 9, pages 291–310, 1993.

[16] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[17] M. Morin, S. Nadjm-Tehrani, P. Österling, and E. Sandewall. Real-Time Hierarchical Control. *IEEE Software*, 9(5):51–57, September 1992.

[18] S. Nadjm-Tehrani. Reactive Systems in Physical Environments. In *Hybrid Problems, Hybrid Solutions – proc. of the 10th. Biennial Conference on AI and Cognitive Science*, pages 169–180. IOS Press, April 1995.

[19] S. Nadjm-Tehrani and J-E. Strömberg. From Physical modelling to Compositional models of Hybrid Systems. In Langmaack et al. [14], pages 583–604.

[20] S. Nadjm-Tehrani and J-E. Strömberg. Modelling and Formal Analysis of an Aircraft Landing Gear System. In *Proc. of the 2nd. European workshop on Real-time and Hybrid Systems*, pages 239–246. VERIMAG Laboratory, Grenoble, May 1995.

[21] S. Nadjm-Tehrani and J-E. Strömberg. Proving Dynamic Properties in an Aerospace Application. In *Proc. of the 16th International Symposium on Real-time Systems*, pages 2–10. IEEE Computer Society Press, December 1995.

[22] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An Approach to Description and Analysis of Hybrid Systems. In Grossman et al. [7], pages 149–178.

[23] N. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *In Proc. 11th International Conference on Automated Deduction, LNCS 607.* Springer Verlag, 1992.

[24] Henry M. Paynter. *Analysis and design of engineering systems.* MIT Press, Cambridge, M.A., 1961.

[25] P.J.G. Ramadge and W.M. Wonham. The Control of Discrete Event Systems. *Proceedings of the IEEE*, 77:81–97, 1989.

[26] A.P. Ravn. Design of Embedded Real-time Computing Systems. Technical Report ID-TR:1995-170, Dept. of Computer Science, Technical University of Denmark, October 1995.

[27] J. U. Skakkebaek and N. Shankar. Towards a Duration Calculus Proof Assistant in PVS. In Langmaack et al. [14], pages 660–679.

[28] U. Söderman. *Conceptual modelling of mode switching physical systems.* PhD thesis, Linköping University, Linköping, 1995. Dissertation no. 375.

[29] J.-E. Strömberg and S. Nadjm-Tehrani. Hybrid Systems Verification Combining Duration Calculus and Bond Graphs. In *Proc. IFAC-IFIP-IMACS Conference on Control of Industrial Systems.* IFAC, 1997. To appear.

[30] J.-E. Strömberg, S. Nadjm-Tehrani, and J. Top. Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems. In *Proc. of the DIMACS International Workshop on Verification and Control of Hybrid Systems, LNCS 1066*, pages 282–293. Springer Verlag, 1996.

[31] J.E. Strömberg. *A mode switching modelling philosophy.* PhD thesis, Linköping University, Linköping, 1994. Dissertation no. 353.

[32] J.E. Strömberg, J.L. Top, and U. Söderman. Variable causality in bond graphs caused by discrete effects. In *Proc. First Int. Conf. on Bond Graph Modeling (ICBGM '93)*, number 2 in SCS Simulation Series, volume 25, pages 115–119, San Diego, 1993.

[33] J.L. Top. *Conceptual modelling of physical systems.* PhD thesis, University of Twente, Enschede, 1993.

[34] M. Westhead and S. Nadjm-Tehrani. Verification of Embedded Systems using Synchronous Observers. In *Proceedings of the 4th International Conference on Formal Techniques in Real-time and Fault-tolerant Systems, LNCS 1135*, pages 405–419. Springer Verlag, September 1996.