, , 1–37 ()

Formal Verification of Dynamic Properties in an Aerospace Application

SIMIN NADJM-TEHRANI simin@ida.liu.se
Dept. of Computer and info. Science,
Linköping University, S-581 83 Linköping, Sweden

JAN-ERIK STRÖMBERG janerik@dst.se
DST Control AB, Mjärdevi Science Park,
Teknikringen 6, S-583 30 Linköping, Sweden

Editor: Thomas Henzinger

Abstract. Formal verification of computer-based engineering systems is only meaningful if the mathematical models used are derived systematically, recording the assumptions made at each modelling stage. In this paper we give an exposition of research efforts in cooperation with aerospace industries in Sweden. We emphasize the need for modelling techniques and languages covering the whole spectrum from informal engineering documents, to hybrid mathematical models. In this modelling process we give as much weight to the physical environment as to the controlling software. In particular, we report on our experience using switched bond graphs for the modelling of hardware components in hybrid systems. We present the basic ideas underlying bond graphs and illustrate the approach by modelling an aircraft landing gear system. This system consists of actuating hydromechanic and electromechanic hardware, as well as controlling components implemented in software and electronics. We present a detailed analysis of the closed loop system with respect to safety and timeliness properties. The proofs are carried out within the proof system of Extended Duration Calculus.

Keywords: formal verification, hybrid system, physical modelling, bond graph, aerospace application, Duration Calculus

1. Introduction

In this article we present some insights gained from the work performed in a multidisciplinary project in cooperation with the Swedish aerospace industries, by computer science, electrical engineering, and mechanical engineering departments at Linköping university. The project concerns generation of models and analysis of *hybrid* systems – mathematical models including both continuous and discrete elements. The article addresses modelling and formal verification of a system, involving hydromechanical and electromechanical 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).

Our main thesis is that verifying properties of computer based systems benefits from modelling both the embedded controller and its surrounding physical environment. Furthermore, unless care is taken in developing the mathematical models which form the basis of formal verification, the results of verification can not be relied upon. Therefore, rather than verifying properties of ad hoc models we take a number of steps for systematically deriving the models from engineering design documents. The models of the hardware and the software are typically fundamentally different in their character[1]. 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. The work reported here has been partially presented in international conferences in the form of shorter articles. This article can therefore be seen as the full version of [21, 32].

Deriving mathematical models from engineering documents, is an inherently difficult task. 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 [33]. At the mathematical level of abstraction, we have employed hybrid transition systems (HTS) [18, 19], and an architecture for the top-level decomposition of hybrid systems [17]. 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 Automata (LHA) [1] with the tool HyTech, and Extended Duration Calculus (EDC) [6] from the hybrid family of languages, as well as Esterel (I/O automata) [4] and statecharts [12] from the synchronous family of languages.

In this article we concentrate on the derivation of mathematical models for a landing gear system and the analysis of safety and timeliness properties of the system within EDC. The application of the other verification techniques are covered in a longer technical report [22].

1.1. The Example

The techniques above are illustrated in the context of a landing gear in which the response to the landing command by a pilot are analysed (see Figure 1). When deriving 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 land-

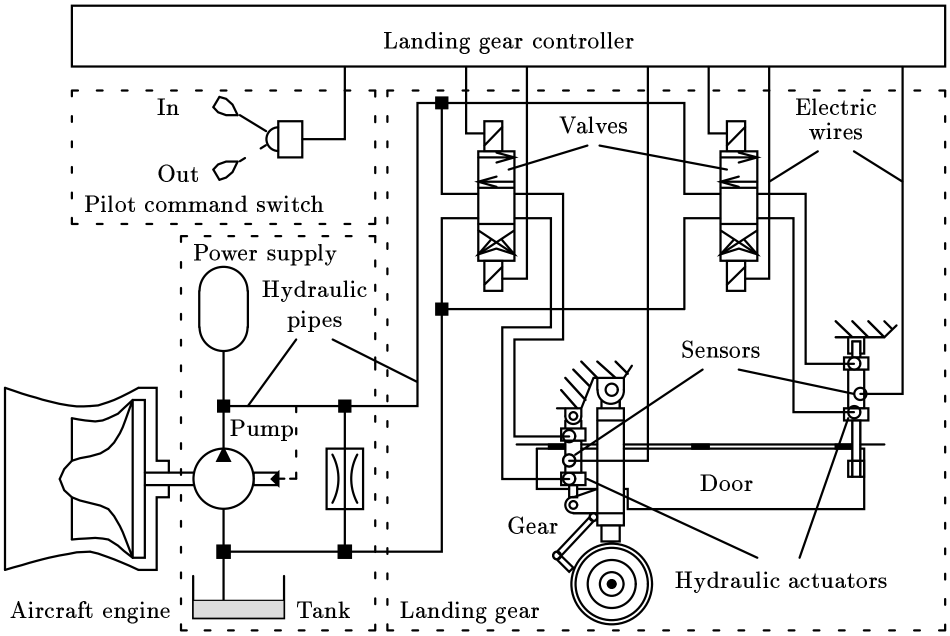


Figure 1. The landing gear system as documented by a mechanical engineer.

ing 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. The load on the hydraulic system itself is dependent on several other "clients" to which hydraulic pressure is delivered. Thus, in order to identify the mode of operation in the landing gear control system there exists another module for monitoring and diagnosis of the hydraulic power supply.

Based on the above observations we have identified a sub-system consisting of two blocks for modelling and analysis. The two blocks cover the power supply and the mechanical linkage parts respectively – each consisting in turn of physical mechanisms and control subsystems. Figure 2 shows the topology of the system under study, where the full arrows denote directions of information exchange and half arrows directions of (positive) energy exchange. In other words, full arrows correspond to signals transmitted (one-way) via some abstract communication channel (e.g. computer programs and communication protocols) and half arrows to bilateral

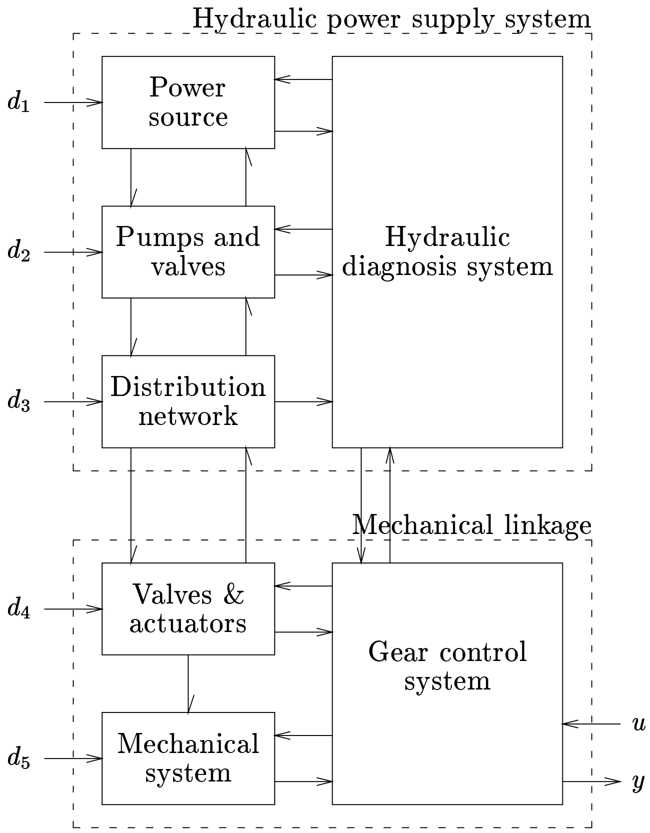


Figure 2. An abstraction of the landing gear system.

signals transmitted via some physical structure (e.g. electromechanical or hydromechanical hardware). 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.

1.2. Summary of the Work

The work in the project has led to derivation of physical and mathematical abstractions for all components of the system. In the current presentation we exclude the diagnosis subsystem, since only proofs pertaining to one of its modes are presented here.

At the first stage, models for the hydromechanical 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 (DAE); rather different sets of DAE 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 modelling phase was followed by proofs of the following three properties for several versions of landing gear models.

- 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.

This article presents the proofs for a static controller in combination with two different environment models: one in which the environment variables change as a linear function of time, and the second where the environment model is a non-linear system of DAE. This corresponds to a mathematical model for a more refined version of the hydraulic supply system in which the hydraulic pressure is not assumed to be constant as in the first case. It is rather assumed to be regulated by a hydromechanic pump.

The model of the composed system was in both cases analysed using the proof system of EDC, leading to proofs for the above properties. While the closed loop model based on the time-linear evolutions in the environment could also be analysed using model checking (e.g. HyTech), the non-linear model required a combination of analysis and logical deductions. Thus, the point with the proof of the time-linear model with EDC was to provide the necessary structure over which the same proofs for the non-linear model of the hydromechanical system could be based.

2. From Schematic Diagrams to Mathematical Models

We let the term 'apparatus' refer to an object for which the behaviour is described by physical laws. By behaviour we mean the evolution in time of measurable physical quantities such as pressure, flow, velocity, current etc. In the context of computer controlled systems, apparatus models generally cover models of actuators, the plant actuated by these actuators, and the sensors used to measure the effects of the actuation. As a consequence, apparatus models generally involve several different physical domains. The apparatus model in Figure 1 involves at least three different physical domains: electric, hydraulic and mechanic.

Industrial apparatus models are typically provided in terms of *schematic diagrams*. Within each physical domain there is a set of generally accepted icons representing more or less complex standard apparatuses such as pilot controlled valves, check valves, double-acting cylinders etc. These icons are fairly well defined in the sense that they represent specific functions, e.g. electrical control of flow, rectification of flow, mechanical actuation by flow etc. They are well understood by experts within the same physical domain, but are often difficult to comprehend by experts outside that domain.

Also, the icons used in schematic drawings are ambiguous with respect to the underlying physics; it is not clear from the syntax which physical effects are taken into account. The reason for this is that the behaviour of an apparatus depends on the circumstances in which it is used. Often the effects needed to be considered depend on the required accuracy of the model, timing properties, dynamic range etc. Moreover, certain interaction effects are not so naturally captured in terms of apparatus models – an example of such interaction effects being friction. Hence, the type of models typically used in industry are in themselves not informative enough for determining a unique mathematical model suitable for e.g. formal verification. From this perspective there is therefore a genuine need for modelling according to physical principles.

2.1. Physical Modelling

In our project the apparatus models provided by the industry were made precise by means of *switched bond graphs.* Switched bond graphs [34, 35, 33, 30] are based on the graphical bond graph language [15], earlier introduced by Henry M. Paynter [25]. The language was originally proposed as a means to bridge the gap between different physical domain experts in the context of complex industrial plants.

To meet these requirements, bond graphs are based on *energy* concepts. By means of a set of well defined primitive energy concepts, bond graphs can be precisely interpreted by experts in several different domains since they have a precise mathematical and physical interpretation. Further, the bond graph concepts are powerful enough to capture even very complex mechanisms using only a few symbols.

2.2. Elementary Bond Graph Concepts

More specifically, bond graphs are based on the theory known as 'first principle of energy conservation'. As a consequence, the nodes in a bond graph represent degenerate primitive energy processes and the directed arcs represent *potential* flow of energy. The arcs are also referred to as *power bonds*, or simply *bonds.* The direction of the bonds represent the direction of *positive* energy flow. Every bond is associated with a pair (e, f) of variables referred to as *generalised power variables.* The power variable e is further referred to as *effort* and the variable f as *flow.*

For a given physical domain, the generalised power variables represent a pair of *physical quantities* such that $[ef] =$ Watts, where $[q]$ denotes the *unit* of the physical quantity represented by q. Recall that a physical quantity is a pair (value, unit) and that a variable q representing it is a real-valued function on time, i.e. $q : \mathbb{R} \to \mathbb{R}$.

By means of e and f the semantics of a bond can now be summarised:

- $ef \geq 0$ iff the direction of energy flow coincides with the direction of the bond.

- $|ef|$ represents the magnitude of the energy flow, i.e. the power transferred by the bond.

EXAMPLE: Consider the hydraulic domain. Let e represent pressure, i.e. $[e] = \mathrm{N/m^2}$, and f volumetric flow, i.e. $[f] = \mathrm{m^3/sec}$. Then we have $[ef] = [e][f] = \frac{\mathrm{N}}{\mathrm{m^2}}\frac{\mathrm{m^3}}{\mathrm{sec}} = \frac{\mathrm{Nm}}{\mathrm{sec}} = \mathrm{Watts}$, i.e. the pair (e, f) is indeed an instance of power variables. In the same way we show that voltage (e) and current (f) is a pair of power variables in the electrical domain and that torque (e) and angular velocity (f) is a pair in the rotating mechanical domain. $\square$

Bond graphs define a necessary and sufficient set of primitives for the modelling of a wide range of practical systems. The necessary and sufficent set of primitives consists of five elements, but normally a more practical set of nine elements is used. These nine elements are grouped into five categories as follows:

1. **Ideal sources:** ideal source of effort denoted **Se** and ideal source of flow denoted **Sf**.
2. **Ideal storages:** ideal storage of effort denoted **I** and ideal storage of flow denoted **C**.
3. **Ideal dissipation:** ideal energy dissipation denoted **R**.
4. **Ideal conversions:** ideal transformer denoted **TF** and ideal gyrator denoted **GY**.
5. **Ideal distributions:** common effort junction denoted **E** and common flow junction denoted **F**.

For example, the following instances of the abstract energy primitives are found in the hydraulic domain: a wide container (**Se**), accumulator (**C**), hydraulic restriction (**R**), hydromechanic pump (**TF**), parallel connection of components (**E**), and series connection of components (**F**).

It is important to note that the nine energy primitives are degenerate ideals. Hence, 'real' phenomena found in nature must be modelled by combining two or more of these primitives. Typically the nodes of a bond graph are labelled by the type (**Se**, **Sf**, **C**, ...) and a graph-unique identity. Hence a bond graph node is labelled **X:id**, where **X** is the type and **id** the unique identity.

2.3. Constitutive Relations

From an energy perspective, the behaviour of a physical mechanism is an n-ary relation over entities representing power variable pairs. Hence we have $n = 2m, m \in \mathbb{N}$, and m is sometimes referred to as the number of *power ports.*

For the ideal primitive mechanisms defined above, the set of possible behaviours is further restricted. The formal definition of this structure will be exemplified in what follows; further details, though informal, can be found in e.g. [2].

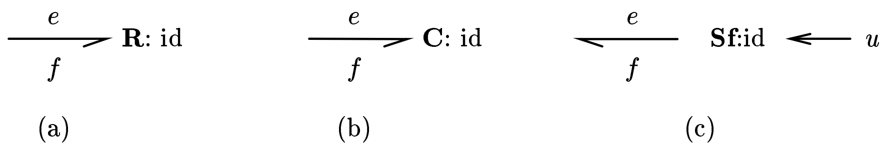


Figure 3. Some example bond graph primitives.

2.3.1. The Ideal Dissipation **R** Consider the primitive ideal dissipation, i.e. the **R**-element as depicted in Figure 3 (a). As a one-port **R**-element its behaviour is a relation over two time functions. Let e, f represent the power variables of the **R**-element. Then $\Phi \subseteq \mathbb{R}^2$, where $(e(t), f(t)) \in \Phi$ for all t, is a *constitutive relation* with the following properties:

1. $(0, 0) \in \Phi$.

2. $e(t)\, f(t) \geq 0$.

3. There exists a bijective function $\Phi_f : \mathbb{R} \to \mathbb{R}$ such that $(e(t), f(t)) \in \Phi$ iff $\Phi_f(e(t)) = f(t)$.

The first two requirements ensure that an ideal dissipation can never *generate* energy. The third requirement ensures existence of an inverse function Φ_e such that $(e(t), f(t)) \in \Phi$ implies $\Phi_e(f(t)) = e(t)$.

EXAMPLE: Consider a long narrow pipe carrying a hydraulic fluid. Let e be the pressure drop across, and f be the flow through the pipe. Let us assume the flow is *laminar*. Then, according to any standard book in fluid power,

$$\Phi = \{(e(t), f(t)) \mid k_1\, e(t) - f(t) = 0 \text{ for a specific } k_1 \in \mathbb{R}^+\}$$

where k_1 is a *physical parameter* computed from properties of the pipe (e.g. friction) and the fluid (e.g. viscousity). □

2.3.2. The Ideal Flow Storage **C** Consider the primitive ideal flow storage, i.e. the **C**-element as depicted in Figure 3 (b). Let e, f represent the power variables of the element, and x a *state variable* defined through $\dot{x} - f = 0$. By definition, x represents a physical quantity which is referred to as the *energy state* of the mechanism. If for instance $[f] = \mathrm{m}^3/\mathrm{sec}$, then it follows that $[x] = \mathrm{m}^3$, i.e. volume, which further supports the intuition behind the energy state concept.

Provided $\dot{x}(t)$ exists for all t, we have $(e(t), x(t)) \in \Phi$ for all t where $\Phi \subseteq \mathbb{R}^2$ is a relation with the same properties as the constitutive relation defined for the **R**-element.

EXAMPLE: Consider an open hydraulic accumulator. The hydraulic pressure e at the bottom of the accumulator (in which the fluid is affected by gravity g only) is proportional to the height h of the fluid level. Hence, if the cross section area A is

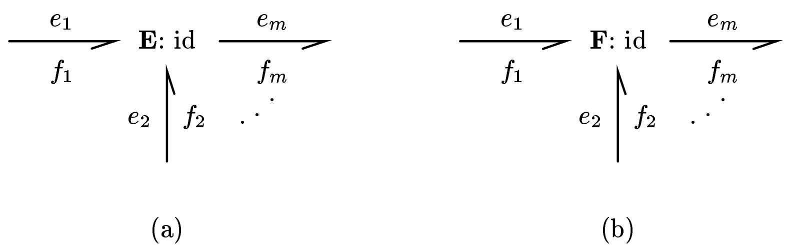


Figure 4. The two primitive ideal distribution elements. In this particular example $\sigma_1 = \sigma_2 = -\sigma_m = 1$.

constant with respect to the height, the pressure is proportional to the accumulated flow f. Let x be accumulated flow and ρ the density of the fluid. Then, for all t, we get

$$\Phi = \{(e(t), x(t)) \mid e(t) - k_2\, x(t) = 0 \text{ and } \dot{x}(t) - f(t) = 0\}$$

where $k_2 = \rho g/A$. □

2.3.3. The Ideal Flow Source **Sf** Consider the primitive ideal flow source **Sf** as depicted in Figure 3 (c). Let e, f be the power variables of the element and u an independent physical quantity referred to as the *modulation signal* or *control signal*. The full arrow in Figure 3 (c) underlines the fact that u represents a causally directed entity. We now have $(e(t), f(t)) \in \Phi$ for all t and $\Phi = \{(e(t), f(t)) \mid u(t) - f(t) = 0\}$. Note that an ideal effort source actually does not restrict e in any way.

2.3.4. The Ideal Distribution Elements Consider the primitive ideal junctions **E** and **F** as depicted in Figure 4. We note that the elements are *multi-ports* in that $m > 1$. As the name suggests, distribution elements are used to model interaction between other primitives, and hence $m \geq 2$ follows as an immediate consequence.

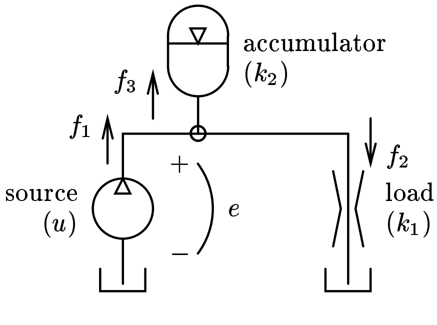
Let e_i, f_i be the power variables of bond i. For an **E**-junction we have, for all t

$$\{(e_1(t), f_1(t), \ldots, e_m(t), f_m(t)) \mid \sum_{i=1}^{m} \sigma_i\, f_i = 0 \text{ and } \forall_{i=1}^{m-1} e_i - e_{i+1} = 0\}$$

and for an **F**-junction

$$\{(e_1(t), f_1(t), \ldots, e_m(t), f_m(t)) \mid \sum_{i=1}^{m} \sigma_i\, e_i = 0 \text{ and } \forall_{i=1}^{m-1} f_i - f_{i+1} = 0\}$$

where $\sigma_i \in \{-1, 1\}$ represents the relative direction of bond i. If bond i is directed *towards* the junction, then $\sigma_i = 1$, otherwise $\sigma_i = -1$.



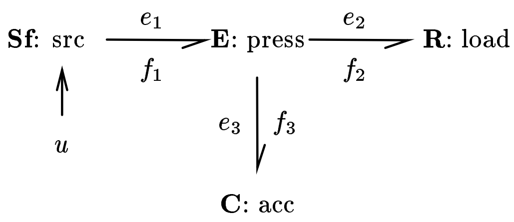


Figure 5. (a) A simple hydraulic circuit. (b) A first bond graph model of the circuit. Note that the system pressure $e = e_1 = e_2 = e_3$ is the same for all three apparatuses.

2.4. Model Composition

The composition of primitive ideals to form models of real systems is obtained after a series of steps. Here we simply outline the modelling process by providing a simple example.

EXAMPLE: Consider the hydraulic system as represented by the schematic drawing in Figure 5 (a). The flow source provides a flow f_1 proportional to the control signal u, the hydraulic load is essentially dissipative and the accumulator nearly loss-free. The interaction between the source, the accumulator and the load is such that the pressures e_1, e_2, e_3 at the ports of the mechanisms are all the same. Consequently, Figure 5 (b) depicts a first potential bond graph model of the system.

Combining the constitutive relations of the individual mechanisms, we get the following set of equations (omitting t for clarity):

$$\begin{aligned}
u - f_1 &= 0 \\
k_1\, e_2 - f_2 &= 0 \\
\dot{x} - f_3 &= 0 \\
e_3 - k_2\, x &= 0 \\
f_1 - f_2 - f_3 &= 0 \\
e_1 - e_2 &= 0 \\
e_2 - e_3 &= 0
\end{aligned}$$

This set of equations uniquely defines the behaviour of the system once the initial value $x(t_0)$ is known. □

2.5. *Causality in Bond Graphs*

We have now seen how a given bond graph and a set of constitutive relations maps to a mathematical model of the underlying system. We have already observed that such a model is a non-minimal set of equations which can in general be difficult to solve explicitly. A preferred alternative is a sequence of directed assignment statements such that unknowns can be immediately and sequentially computed from the knowns on the right hand side. Such a model is sometimes referred to as a *computational model.*

Such a causal computational model requires the model variables to be ordered in a specific *cause-effect* relationship. Bond graph theory comes with algorithms for assigning causality to a given graph [36]. The choice of algorithm depends on the specific form of computational model required as well as on the complexity of the bond graph itself. There are graph oriented algorithms suitable for manual calculations as well as algorithms better suited for machine implementation. One classical algorithm is the so called *Sequential Causality Assignment Procedure* (SCAP) which is used to derive classical state space forms consisting of ordinary differential equations only.

EXAMPLE: Applying SCAP on the model in Section 2.4 we obtain

$$\dot{x} = -k_1 k_2 \, x + u$$

□

The above equation is the basis for the physical model of the landing gear appearing in our application.

2.6. *The Switched Extension*

Classical bond graphs only allow for finite flows of energy and are thus restricted to the modelling of continuous change over time only. However, many practical apparatuses undergo abrupt changes in their behaviour, usually due to external or internal discrete control. Examples of such apparatuses are diodes, check valves, relays and devices with end-stops. Systems consisting of at least one such apparatus will be referred to as *mode-switching* systems.

Switched bond graphs were introduced to extend bond graphs to mode-switching physical system while maintaining the classical bond graph theory. The main ingredient in the extension is a new ideal degenerate element, namely the *ideal generalised switch* (**Sw**). 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 Boolean state of the switch. The two states of the **Sw**-element are referred to as the *effort* and the *flow* state. The reason for these terms is the following: In state effort the **Sw**-element is equivalent with an **Se**-element for which $u = 0$ for all t, and in state flow it is equivalent with an **Sf**-element for which $u = 0$. Given a switched

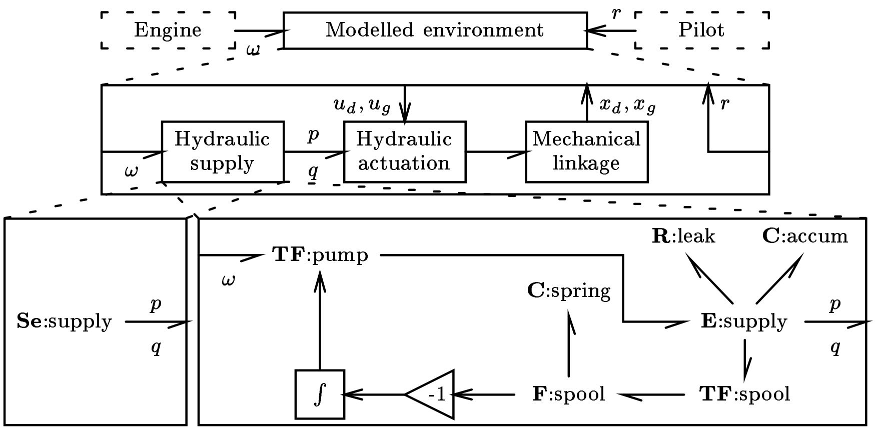


Figure 6. The physical environment models. Left: the coarsest model of the hydraulic supply. Right: the model including a hydromechanically regulated pump.

bond graph with p **Sw**-elements, once all the elements have been mathematically characterised, we get a computational hybrid system with $\leq 2^p$ sets of DAE, and a set of Boolean transition conditions. In this paper we do not elaborate further on the switched versions of the landing gear models. The interested reader is referred to [33, 30, 31].

2.7. The Landing Gear Models

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 (Figure 1). As part of the landing gear system, the hydraulic power supply system provides the hydromechanic energy needed to manoeuvre the gear and the doors under all operating conditions.

For the purpose of illustration, we present two different bond graph models of the hydraulic power supply subsystem only 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 6 where p and q stand for hydraulic 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

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

where $\alpha_d, \alpha_g > 0$ are constants (time-invariants), x_d (x_g) the normalised 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 6), we no longer assume that the hydraulic pressure is ideal. Instead we make an explicit model of the hydromechanic 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

$$\begin{aligned} \dot{x}_d &= \alpha'_d\, p\, u_d, \quad x_d \in [0,1] \\ \dot{x}_g &= \alpha'_g\, p\, u_g, \quad x_g \in [0,1] \\ \dot{p} &= -(\gamma + \gamma_d\, |u_d| + \gamma_g\, |u_g|)\, p + \beta \end{aligned}$$

where $\alpha'_d, \alpha'_g, \gamma_d, \gamma_g > 0$ are constants. 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 Aerospace and the Department 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.

3. Framework for Iterative Modelling

This section describes 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 2 by making the interface between the controller (diagnoser) and the physical world more explicit.

The generic architecture as depicted in Figure 7, accommodates 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 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.

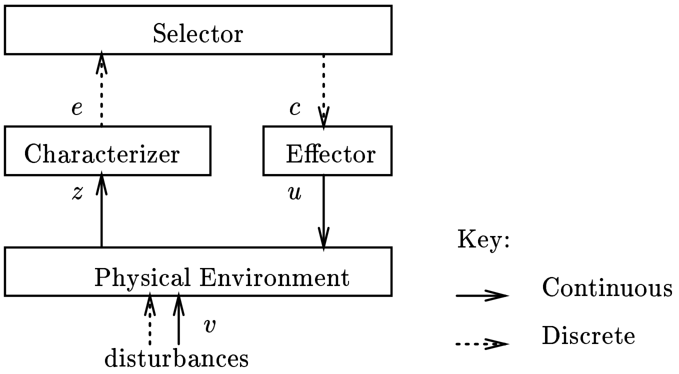


Figure 7. A generic architecture for decomposition of hybrid systems

In addition, it may be used to model major elements of the environment not under control.

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 framework for iterative modelling of the closed loop system. It was thus used to plug in different models of the environment while 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 section we will show the details of two alternative landing gear models by filling the boxes in the above architecture.

4. Hybrid Mathematical Models

In this section we describe how the models for the closed loop system can be derived using the mathematical models for the hardware derived earlier. Here, we concentrate on a static selector as depicted in Figure 8. This model can be a representation of a hard-wired electronics implementation of the controller, or alternatively, a model obtained from the compilation of a synchronous software model, e.g. in Esterel or Lustre [9]. Analysis of similar properties in presence of a dynamic selector with communication and computation delays can be found elsewhere [21, 22].

Next we will make the mappings in the interface between the controller and its environment (i.e. the characterizer and the effector) explicit.

As it can be seen in Figure 9, the inputs to the static selector are discrete states *open, closed, in, out, cmd.* Changes in these states are determined by the characterizer based on the sensed values of the door and gear position and the current pilot command. Note also that keeping the same selector and changing the environment

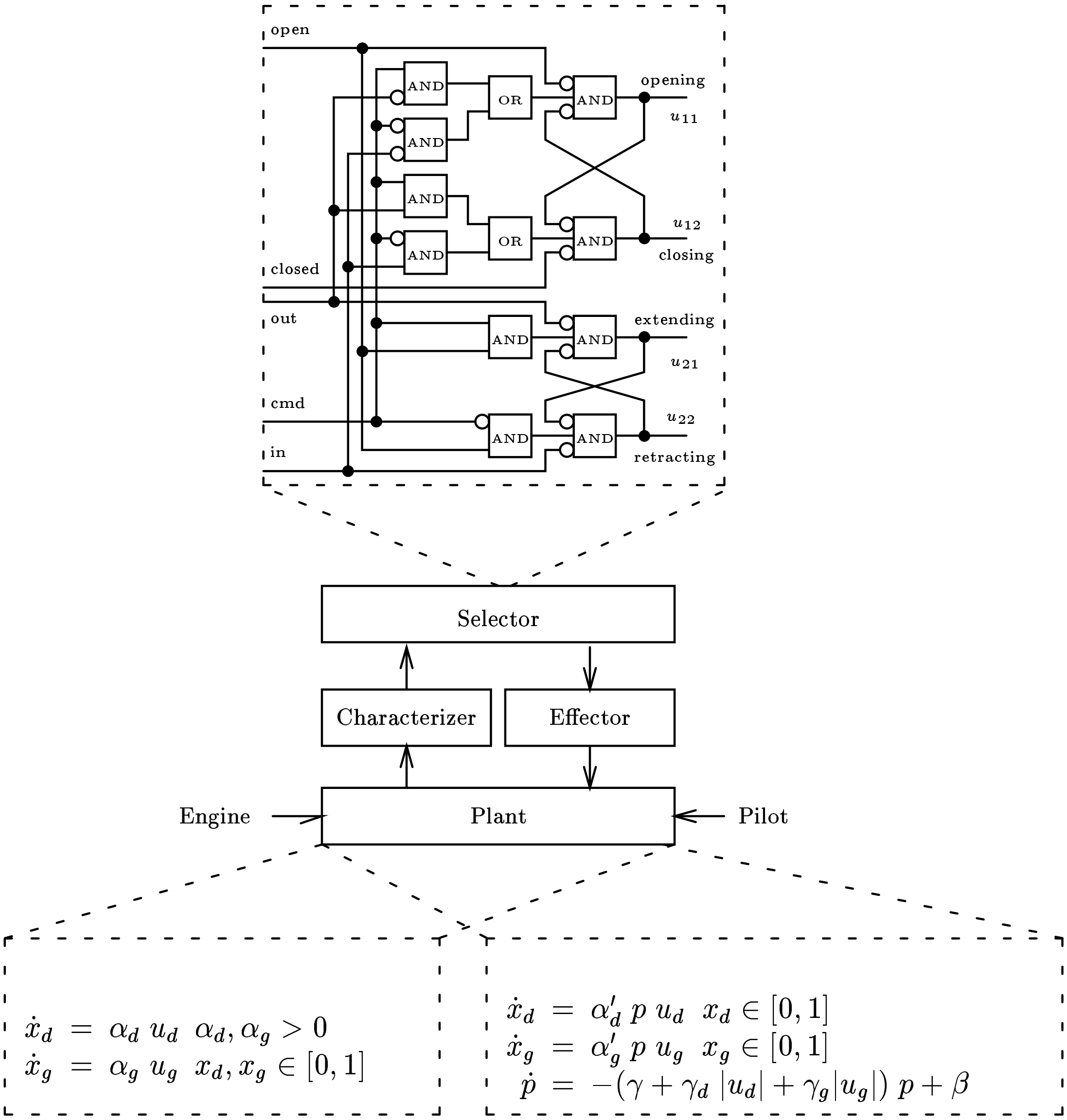


Figure 8. A static selector operating in two alternative physical environments.

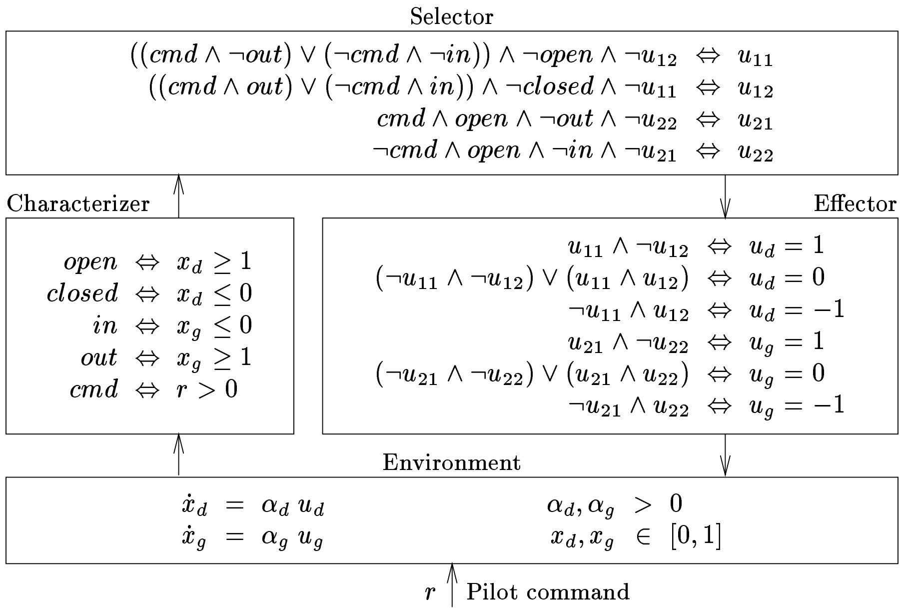


Figure 9. The system built up from the time-linear environment model and the static selector.

model to a more realistic one is simply achieved by plugging in the finer non-linear model in the architectural decomposition.

To analyse the model first we need to represent the closed loop model and the required properties in an extension of the interval temporal logic *Duration Calculus* (DC). Then verifying that the requirements are met by the given model is achieved through proving theorems in the proof system of the logic.

4.1. Brief Introduction to EDC

We use the extension of Duration Calculus in accordance with [27] in which a mathematical theory about state functions is assumed as given. The logical foundations of DC can be found in [11]. Here we give a brief exposition of the extended language. The syntactic constituents of EDC are *state names* denoting functions on time (in our case real valued functions $x_d, x_g, \dot{x}_d, \dot{x}_g$, etc), including *Boolean state names* denoting a Boolean valued function on time (in our case *open*, *closed*, etc). *State expressions* and *state assertions* are built from variables, functions and relations on real numbers, and propositional logic operators in a standard form. In our example we have $u_d\alpha_d$ as an state expression, and $x_g \leq 1$ as well as $P \wedge C$ as state assertions.

State expressions and assertions are evaluated in accordance with interpretations of states and operators. Assertions evaluate to 0 or 1 in a given interpretation. In particular, *true* and *false* as state assertions evaluate to 1 and 0 respectively at all times. *Durations* are formed from state expressions and assertions, using the symbol $\int$. Thus, for an assertion P, $\int P$ denotes the duration of P holding over an interval. Its semantics with respect to interpretation $\mathcal{I}$ and interval $[b, e]$ is defined by the integral $\int_b^e \mathcal{I}(P)dt$, where $\mathcal{I}$ assigns values of the right type to state names, and defines the (non standard) operator symbols of the language.

The *length* ℓ of an interval, is defined by the duration of *true* holding over the interval, expressed as $\ell \widehat{=} \int true$. Thus, the semantics for ℓ, in any interpretation $\mathcal{I}$ over any interval $[b, e]$, is defined by $\mathcal{I}(\ell)[b, e] = e - b$. 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:

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

Atomic duration formulas have the form $\lceil Q \rceil$ or $\triangleleft(dterm_1, \ldots, dterm_n)$ where $\triangleleft$ is an n-ary relation on reals and $dterm_i$ are duration terms. Duration terms essentially have 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 built from atomic formulas using the usual logical connectives as well as the chop connective of interval temporal logic (denoted by ;) which chops an interval in two parts.

The truth of a formula D, determined over an interval $[b, e]$ in an interpretation $\mathcal{I}$, is denoted $\mathcal{V}_\mathcal{I}, [b, e] \models D$. Specifically, the semantics of formulas built using the chop operator is defined as follows. For duration formulas D_1 and D_2, we have:

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

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 $t \in \mathbb{R}^{\geq 0}$. 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

To avoid excessive use of parentheses, precedence rules for the operators in the language are provided [27] – e.g. $\neg$, $\Box$ and $\Diamond$ bind stronger than chop (;) which binds stronger than the other logical operators.

Next we present a number of proof rules (laws) from the proof system of EDC [27] which will be used in the proofs later on in the paper.

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:

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

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 may use 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 a general EDC result which is used in later proofs.

LEMMA 1 *For duration formulas D_1 and D_2 and state assertion P*

$$\lceil P \rceil \wedge (D_2 \,;\, D_2) \Rightarrow ((\lceil P \rceil \vee \lceil\, \rceil) \wedge D_1) \,;\, ((\lceil P \rceil \vee \lceil\, \rceil) \wedge D_2)$$

Table 1. Declaration of states and global variables.

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)

Proof:

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

■

4.2. EDC Model of the Landing Gear

Table 1 gives the declaration of the constituents in the coarse plant model of the landing gear, the only variables determined externally being the reference pilot signal (r).

Using the above notation and the architectural breakdown in Figure 9, 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 9. 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, $\lceil P \rceil \Rightarrow \lceil P_1 \rceil \wedge \lceil P_2 \rceil$.

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

$$P_2 \;\widehat{=}\; \begin{cases} \begin{array}{rcl} (u_{21} \wedge \neg u_{22} & \Rightarrow & \dot{x}_g = \alpha_g) \wedge \\ ((\neg u_{21} \wedge \neg u_{22}) \vee (u_{21} \wedge u_{22}) & \Rightarrow & \dot{x}_g = 0) \wedge \\ (\neg u_{21} \wedge u_{22} & \Rightarrow & \dot{x}_g = -\alpha_g) \\ 0 \leq x_g \leq 1 & & \\ 0 \leq \alpha_g \leq 1 & & \end{array} \end{cases} \tag{2}$$

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

$$C \widehat{=} \tag{3}$$

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

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

$$\lceil C \rceil \Rightarrow \lceil \neg(u_{11} \wedge u_{12}) \rceil \tag{4}$$

$$\lceil C \rceil \Rightarrow \lceil \neg(u_{21} \wedge u_{22}) \rceil \tag{5}$$

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

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

4.3. Requirement Specifications in EDC

We now present the EDC formulations of the three closed loop system properties which were presented in the introduction. First we consider the safety property that the door and gear do not collide in operation. The property is proved by proving the stronger property that the gear must never move when the door is not fully open. This requirement R_1 may be formalised as

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

The timeliness properties require that the extension (or retraction) to be completed within T time units. These requirements may now be formalised by the two formulas R_2 and R_3.

$$\begin{aligned} R_2 \;&\equiv\; \Box\neg((\lceil r > 0 \rceil \wedge \ell = T) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil) \\ R_3 \;&\equiv\; \Box\neg((\lceil r \leq 0 \rceil \wedge \ell = T) \,;\, \lceil \neg(x_g \leq 0 \wedge x_d \leq 0) \rceil) \end{aligned}$$

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

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

5. Verification of the Requirements

In what follows we give the proofs that the closed loop system satisfies requirements R_1 and R_2. The proof for R_3 is analogous to the one for R_2 and is therefore omitted. First we show how proofs in EDC can be used to verify the design in presence of the coarse environment model. Next, in section 6, we show another proof for the closed loop model, now including the non-linear environment model instead. This is carried out by just adding three extra lemmas and preserving the proof structure for the simpler environment model.

5.1. Verification of $\boldsymbol{R_1}$

In order to verify the first requirement we employ proof by contradiction within the proof system of EDC [27]. But before that we need to state the following lemma which gives a property of the application model as presented in 4.2.

LEMMA 2 $\lceil P \wedge C \rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \Rightarrow \mathit{false}$

Proof:

$$
\begin{array}{ll}
 & \lceil P \wedge C \rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{P-And}\} \\
 & \lceil P \wedge C \wedge \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{2, \text{PL}\} \\
 & \lceil C \wedge \neg((\neg u_{21} \wedge \neg u_{22}) \vee (u_{21} \wedge u_{22})) \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{PL}\} \\
 & \lceil C \wedge ((u_{21} \wedge \neg u_{22}) \vee (\neg u_{21} \wedge u_{22})) \wedge x_d < 1 \rceil \\
\Rightarrow & \{3, \text{PL}\} \\
 & \lceil ((x_d \geq 1) \vee (x_d \geq 1)) \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{PL}\} \\
 & \lceil \mathit{false} \rceil \\
\Rightarrow & \{\text{Defn}\} \\
 & \int \mathit{false} = \ell > 0 \\
\Rightarrow & \{\text{Zero}\} \\
 & 0 = \ell > 0 \\
\Rightarrow & \{\text{analysis}\} \\
 & \mathit{false}
\end{array}
$$

■

Now we can prove the safety property.

THEOREM 1 $\lceil S \rceil \wedge \neg R_1 \Rightarrow \mathit{false}$

Proof:

$$
\begin{array}{ll}
 & \lceil P \wedge C \rceil \wedge \neg \Box \neg \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{Somewhere-Neg, PL}\} \\
 & \lceil P \wedge C \rceil \wedge \Diamond \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \\
\Rightarrow & \{\text{Defn}\} \\
 & \lceil P \wedge C \rceil \wedge \mathit{true} \,;\, \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil \,;\, \mathit{true} \\
\Rightarrow & \{\text{Lemma 1}\} \\
 & ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge \mathit{true}) \,;\, ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil) \,; \\
 & ((\lceil\,\rceil \vee \lceil P \wedge C \rceil) \wedge \mathit{true}) \\
\Rightarrow & \{\text{Lemma 2, PL}\} \\
 & (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \,;\, (\lceil\,\rceil \wedge \lceil \dot{x}_g \neq 0 \wedge x_d < 1 \rceil) \,;\, (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \\
\Rightarrow & \{\text{Defn}\} \\
 & (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \,;\, \mathit{false} \,;\, (\lceil\,\rceil \vee \lceil P \wedge C \rceil) \\
\Rightarrow & \{\text{Chop-false}\} \\
 & \mathit{false}
\end{array}
$$

■

5.2. *Verification of $\boldsymbol{R_2}$*

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

5.2.1. Structure of Closed Loop System First, going back to our closed loop model as depicted in Figure 9 and the mathematical formulation of the controller in equation (3) we note that the controller represents the combination of a static selector and a static characterizer, whereas the plant equations (1), (2) can be seen as a combination of the static effector and the dynamic environment model.

Using hybrid transition systems [18, 20, 19] for modelling the environment, phases of continuous activity are interleaved with discrete mode transitions. The system stays in a mode where its behaviour is defined by a set of DAE until the guard of some transition out of the mode is true, whereby it immediately moves to the next mode as defined by a different set of DAE. This compares with a hybrid automaton in which taking an enabled discrete transition is not optional but progress is enforced.[2].

Using different expertise for deriving the models necessitates that the model of a controller and its environment are modularly developed. 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 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 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. A parallel composition of the plant and controller would then give us a hybrid transition system in which the reachability of some modes would be constrained. In what follows we provide the structure of the closed loop system after application-based simplification (minimisation) of the hybrid transition system[3].

The following properties of the closed loop model gives us the underlying 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

$$\mathcal{E} : \mathbb{B}^4 \to \{\langle u_d, u_g \rangle \mid u_d \in \{-1, 0, 1\} \text{ and } u_g \in \{-1, 0, 1\}\}$$

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 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}. Lemma 4 formalises this property of the selector-effector composition. The allowed values of u_{ij}, however, depend on a partitioning of the state space given below. Note that state regions S_i provide a covering of the state space in presence of the landing command.

LEMMA 3 *Let S_i be defined according to the following combinations of state variables and pilot command:*

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

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 4 *Let S_i be defined according to Lemma 3. Then the following statements can be derived using propositional logic:*

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

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

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

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

Note that the above result implies that while the landing command is in operation (implicit in the region conditions), no combinations of u_{ij} other than those listed above are possible. This means that in none of the given regions, no pairs of door and gear movement commands are issued simultaneously by the selector. This is a direct corollary to Lemma 3. 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 4.

LEMMA 5 *Let S_i be defined according to Lemma 3. Then we have*

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

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

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

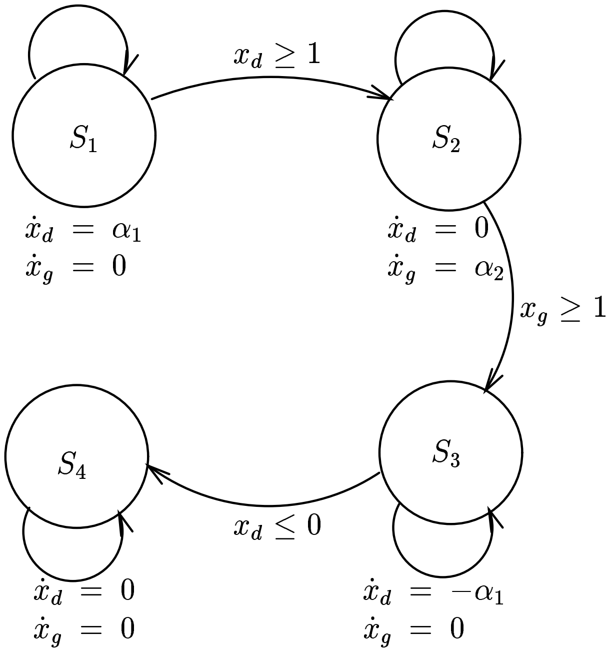


Figure 10. State space regions of the closed loop system with landing command in operation.

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

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 is derived for the closed loop system. With each mode in this system we associate a region in the continuous state space given that the pilot command is in operation[4].

Figure 10 associates one mode with each of the state space regions S_i in Lemma 3 above. Each mode in the graph is annotated with the corresponding door and gear equations given in Lemma 5. Note that reflexive transitions have an implicit "otherwise" condition. They have only been drawn to simplify visualisation: 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 for moving out to the other half of the model which has not been depicted.

Based on Lemma 3 and Lemma 5 we can provide a pictorial representation of the mode changes, where the choice of arcs and their respective labels are justified by characteristics of each region S_i. This analysis is formalised in the following EDC formulation. The first formula, for example, states that if S_1 holds during a subinterval which starts an interval, then it will either continue to hold during the rest of the interval, or the subinterval will be followed by an adjacent subinterval in which S_2 holds.

LEMMA 6 *For S_i as defined in Lemma 3 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* $\lceil S_4 \rceil \Rightarrow (true \,;\, \lceil S_4 \rceil)$

Proof: By straight forward mathematical analysis, we can motivate the transition from S_1 to S_2 as follows: based on Lemma 5 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. ■

5.2.2. The Proof Sketch for R_2 The above structure is the basis for the following proof steps which lead to the proof of R_2. First, we show that if the system arrives at S_4 then we can be sure that the condition stipulated in the second subinterval of R_2 is not violated (i.e. the door not fully closed or the gear not fully extended) – this is shown in Lemma 7. Hence, it suffices to show that from any state, if $r > 0$ persists, then the system will reach S_4 within the time limit stipulated in the first subinterval of R_2. To begin with, we show that the duration of stay in each mode is limited according to Lemma 8. Then the main theorem shows that starting anywhere in the state space will take us to S_4 within the allowed time T *provided that* the following relation holds: $\frac{2}{\alpha_d} + \frac{1}{\alpha_g} \leq T$. Alternatively stated, we obtain these additional constraints as a by-product of the main proof. The proofs of the lemmas and the main theorem can be found in the appendix.

LEMMA 7 *Let S_4 be defined as in Lemma 3. Then we have:*

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

Next, we show the constraints over the duration of stay in each of $S_1, \ldots, S_3$.

LEMMA 8 *Let S_i be defined as in Lemma 3. Then we have:*

(1) $\lceil P \wedge C \rceil \wedge \lceil S_1 \rceil \Rightarrow \ell \leq \frac{1}{\alpha_d}$

(2) $\lceil P \wedge C \rceil \wedge \lceil S_2 \rceil \Rightarrow \ell \leq \frac{1}{\alpha_g}$

(3) $\lceil P \wedge C \rceil \wedge \lceil S_3 \rceil \Rightarrow \ell \leq \frac{1}{\alpha_d}$

This will in turn imply that, under the above constraint over α_d, α_g and T, neither of these modes can last for as long as T time units.

COROLLARY 1 *Let* $\frac{1}{\alpha_d} \leq T$ *and* $\frac{1}{\alpha_g} \leq T$. *Then for* $i \in \{1,2,3\}$ *we have*

$$\lceil S_i \rceil \wedge \ell = T \Rightarrow false$$

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

6. Verifying 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'. We recall the refined physical model of the hydraulic supply system (corresponding to the bond graph to the right of Figure 6) 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 2 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 4 still holds since it is based on the combination of the characterizer and the selector which remain unchanged. Lemma 5 will be restated as the following lemma taking account of *Env'* equations.

LEMMA 9 *Let* S_i *be defined according to Lemma 3. Let* $P' \mathrel{\widehat{=}} Env' \wedge Eff$. *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 Figure 10 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 10 with the equations on the right hand side of implications in Lemma 9. Based on the above lemma, Lemma 7 will continue to hold for S'. Furthermore, the new structure and the following lemma can be used to show that the earlier result about S stated in Lemma 6 continues to hold for S'.

LEMMA 10 *Consider* $S' \equiv P' \wedge C$. *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:
Based on physical knowledge we know that β is a 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 initialised with adequate pressure. Next we show that the positions of the door and gear will monotonically increase (decrease) in every S_i region except for S_4.

LEMMA 11 *For any interval in which* u_d *and* u_g *have fixed values,* p *is monotonic in* t.

Proof: Follows from Lemma 9 and Lemma 10. ■

It is therefore the case that the earlier transition relation between the modes (regions) holds even in the new system S'. In other words, Lemma 6 holds also for S'. The last two results can be used to derive the upper bound for the duration of staying in each region in terms of the minimal pressure value $\hat{p}$, in a similar fashion to the non-linear case. Only the duration of stay in each mode is now also dependent on the minimal pressure $\hat{p}$. The higher the pressure the shorter time it takes for moving from an end position (for the door or gear) to another end position.

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

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

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 5 and the associated right hand side differential equations, with references to Lemma 9 and the new equations from Env' as given in Lemma 9.

■

7. Concluding Remarks

We have reported here on an approach to formal verification which is initiated at the informal engineering documents. We have described 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, to obtain models suitable for formal verification.

We used a generic architecture to derive alternative closed loop models based on alternative environment models. These ranged from simple time-linear models to non-linear models allowing variable hydraulic pressure. We also illustrated the use of proofs in Extended Duration Calculus to ensure safety and timeliness properties of the different closed loop models.

7.1. The Insights Gained

Going back to our generic hybrid architecture, the work reported in this article illustrates a possible technique when the complexity 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 other methods illustrated in [21, 22], for 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 among the first applications of formal deductive proofs to hybrid systems with non-linear DAE.

It should be obvious from the illustration of the method that, aided by the architectural decomposition, the EDC formulations of the closed loop model were easy to obtain. Moreover, switching from one plant model to another simply amounts to changing one conjunct in the formula representing S (the closed loop system) with another (representing the refined plant model).

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 mechanised 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. PC/DC, a mechanical prover for Duration Calculus implemented on top of PVS [24] has existed for some years [29]. An attempt to mechanically check some of our hand proofs was successfully carried out [23]. There is, however, more work to be done; tedious rewriting to bring the sequent in a special form can be eliminated by provision of more infrastructure in the prover. The PC/DC proof for Lemma 2 which requires 8 hand proof steps, for example, required 140 proof steps.

Another reflection over the work performed is the close connection between the region based transition system (Figure 10) and the structure of the proof. Much of the literature on formal verification assumes that the model (or axiomatisation) of the system to be analysed exists a priori. This formalisation is additionally assumed to be in a form suitable for the application of the given proof technique. HyTech [14] could for example analyse the transition system based on the coarse model automatically, thereby eliminating the need for the EDC proof in the time-linear case. 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).

7.2. The Wider Perspective

Fundamental research in formal methods and its deployment in several real world applications has seen a major progress over the past decade. Among the notable examples are the application of formal methods and tools to the London air traffic management system [10], avionic software for the Lockheed C130J [7], the Traffic Collision avoidance system (TCAS) [13], the Motorola 68020 microcode processor [5], and the proof of correctness for an SRT division algorithm [28].

However, despite the impressive results, it is clear that the road to application of formal techniques as a routine step within the system development process is quite a long and bumpy one. To attack the problems along this road research in two directions is needed. There is a need for basic research in formal languages, data structures and algorithms for specification and reasoning about complex embedded systems. There is also a requirement to study the needs of application domain engineers, the suitable models and abstractions in different domains (e.g. telecommunications, aerospace, electronic design, etc.) and to obtain suitable interfaces accessible to non formal method experts. The key to success is to get a good synergy between both of these tracks of research.

The first phase of the reported project started along the second track. Our work confirms conclusions reported in similar projects [8], that analysis by deductive techniques requires a great deal of user knowledge and interaction. We note that this is also true with algorithmic approaches. When applying model-checking in our example, the bulk of the work was in justifying the time-linearity assumption by the physical modelling activity, and presenting the exact hardware assumptions under which this verification result makes sense.

Another approach that we studied may be labelled the 'discrete events approach', whereby the system is represented as parallel composition of a set of finite automata. Although the theory for synthesis of discrete controllers has existed for a while [26], its application to verification requires obtaining natural discrete models of the plant, and practical automatic verification tools.

Our study of the landing gear example in this framework was carried out within the development environment of Esterel/Mauto [3]. The synchronous 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 attraction being the possibility of automatic code generation. Once the model of a controller in interaction with a plant has been formally verified for certain properties, then the high level controller model can be automatically translated to C, Ada or a circuit design language.

Again, our application of these techniques rests on the fact that realistic discrete models of the mechanics and hydraulics could be developed based on the physically driven models.

To sum up, our experience with 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.

Acknowledgments

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 Aerospace, and Arne Jansson at the division of Fluid Power, Department 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. The work by the second author was partly carried out while at the Department of Electrical Engineering at Linköping University.

Appendix

Proofs for the timeliness property

Lemma 7

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 1}\} \\
& ((\lceil P \wedge C \rceil \vee \lceil\,\rceil) \wedge true) \,;\, ((\lceil P \wedge C \rceil \vee \lceil\,\rceil) \wedge \lceil S_4 \rceil) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{PL}\} \\
& (\lceil P \wedge C \rceil \vee \lceil\,\rceil) \,;\, ((\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil) \vee (\lceil\,\rceil \wedge \lceil S_4 \rceil)) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{Defn, PL}\} \\
& (\lceil P \wedge C \rceil \vee \lceil\,\rceil) \,;\, (\lceil P \wedge C \rceil \wedge \lceil S_4 \rceil) \,;\, \lceil \neg(x_g \geq 1 \wedge x_d \leq 0) \rceil \\
\Rightarrow & \{\text{Lemma 5}\} \\
& (\lceil P \wedge C \rceil \vee \lceil\,\rceil) \,;\, (\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 \vee \lceil\,\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 \vee \lceil\,\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 \vee \lceil\,\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{Lemma 3}\} \\
& (\exists v_1 v_2 : \mathbb{R} \bullet (\lceil P \wedge C \rceil \vee \lceil\,\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 \vee \lceil\,\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 \vee \lceil\,\rceil) \,;\, (v_1 \geq 1 \wedge v_2 \leq 0)) \,;\, false \\
\Rightarrow & \{\text{Chop-false}\} \\
& false
\end{array}
$$

■

Lemma 8

Proof:
We show the proof for (1) which is a proof by contradiction. Proofs for (2) and (3) are analogous.

$$\lceil P \wedge C \rceil \wedge \lceil S_1 \rceil \wedge \ell > \tfrac{1}{\alpha_d}$$
$\Rightarrow$ {Lemma 3,Lemma 5, P-And}
$$\lceil P \wedge C \wedge x < 1 \wedge \dot{x}_d = \alpha_d \wedge \dot{x}_g = 0 \rceil \wedge \ell > \tfrac{1}{\alpha_d}$$
$\Rightarrow$ {Defn}
$$\lceil x \geq 0 \wedge x < 1 \wedge \dot{x}_d = \alpha_d \wedge \dot{x}_g = 0 \rceil \wedge \ell > \tfrac{1}{\alpha_d}$$
$\Rightarrow$ {analysis}
$$\ell \leq \tfrac{1}{\alpha_d} \wedge \ell > \tfrac{1}{\alpha_d}$$
$\Rightarrow$ {PL}
$$false$$

■

Theorem2

Proof:

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

Considering the middle intervals:

$$(\lceil \, \rceil \vee \lceil P \wedge C \rceil) \wedge ((\lceil r > 0 \rceil \wedge \ell = T) \; ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {PL}
$$(\lceil \, \rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \; ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \vee$$
$$(\lceil P \wedge C \rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \; ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {Defn, PL, Chop-false}
$$false \vee (\lceil P \wedge C \rceil \wedge (\lceil r > 0 \rceil \wedge \ell = T) \; ; \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {Lemma 1}
$$((\lceil \, \rceil \vee \lceil P \wedge C \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \; ; ((\lceil \, \rceil \vee \lceil P \wedge C \rceil) \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {Defn, PL}
$$(\lceil P \wedge C \rceil \wedge \lceil r > 0 \rceil \wedge \ell = T) \; ; (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {P-And, Lemma 3}
$$(\lceil P \wedge C \rceil \wedge \lceil S_1 \vee S_2 \vee S_3 \vee S_4 \rceil \wedge \lceil r > 0 \rceil \wedge \ell = T) \; ;$$
$$(\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$
$\Rightarrow$ {Chop-P-Or}
$$(\lceil P \wedge C \rceil \wedge$$
$$(\lceil S_1 \rceil \; ; true \vee \lceil S_2 \rceil \; ; true \vee \lceil S_3 \rceil \; ; true \vee \lceil S_4 \rceil \; ; true) \wedge$$
$$\lceil r > 0 \rceil \wedge \ell = T) \; ;$$
$$(\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil)$$

We prove each case separately. The first three cases cover the possibilities that the first subinterval is started by S_1, S_2, and S_3 respectively. Cases 2 and 3 essentially appear as a subproof in case 1, which is the longest subproof and will be shown here.
Case 1:

$$
\begin{array}{ll}
 & (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \mathbin{;} \mathit{true}) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 6}\} \\
 & (\lceil P \wedge C \rceil \wedge (\lceil S1 \rceil \vee (\lceil S1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \mathit{true})) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Corollary 1, PL}\} \\
 & (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \mathit{true}) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 6}\} \\
 & (\lceil P \wedge C \rceil \wedge (((\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil) \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \mathit{true})) \wedge \\
 & \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\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 \\
 & ((\ell \leq 1/\alpha_d \mathbin{;} \ell \leq 1/\alpha_g \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \mathit{true})) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\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 (\mathit{false} \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \mathit{true})) \wedge \lceil r > 0 \rceil \wedge \ell = T \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 6, PL}\} \\
 & (\lceil P \wedge C \rceil \wedge \\
 & (((\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil) \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \lceil S_4 \rceil \mathbin{;} \mathit{true})) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\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 \mathbin{;} \ell \leq 1/\alpha_g \mathbin{;} \ell \leq 1/\alpha_d \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \lceil S_4 \rceil \mathbin{;} \mathit{true})) \wedge \\
 & \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\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 (\mathit{false} \vee (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \lceil S_4 \rceil \mathbin{;} \mathit{true}) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 6, PL}\} \\
 & (\lceil P \wedge C \rceil \wedge (\lceil S_1 \rceil \mathbin{;} \lceil S_2 \rceil \mathbin{;} \lceil S_3 \rceil \mathbin{;} \mathit{true} \mathbin{;} \lceil S_4 \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \mathbin{;} \\
 & (\lceil P \wedge C \rceil \wedge \lceil \neg(x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7, Chop-false}\} \\
 & \mathit{false}
\end{array}
$$

Next we show the case where S_4 holds at the beginning of the first subinterval.
Case 4:

$$
\begin{array}{ll}
& (\lceil P \wedge C \rceil \wedge (\lceil S_4 \rceil \,;\, true) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 6}\} \\
& (\lceil P \wedge C \rceil \wedge (true \,;\, \lceil S_4 \rceil) \wedge \lceil r > 0 \rceil \wedge \ell = T) \,; \\
& (\lceil P \wedge C \rceil \wedge \lceil \neg (x_g \leq 1 \wedge x_d \leq 0) \rceil) \\
\Rightarrow & \{\text{Lemma 7, Chop-false}\} \\
& false
\end{array}
$$

Based on the above case analysis and through two further applications of **Chop-false** we can conclude the proof for

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

■

Notes

1. Our notion of hardware includes mechanical, electrical and hydraulic components. The paper will make some of these differences clear.
2. Note that once the guard is true, the jump is immediate in physically modelled plants, but may be subject to explicit time constraints for timed controllers.
3. In general, we need formal rewrite rules and tools to obtain the simplest form for each application, but here the model is presented in simplified form
4. 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.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Journal of Theoretical Computer Science*, 138:3–34, 1995.
2. J.J. Beaman and R.C. Rosenberg. Constitutive and modulation structure in bond graph modeling. *Journal of Dynamic Systems, Measurement and Control*, 110(4):395–402, December 1988.
3. G. Berry. The Esterel v5 Language Primer. Technical report, Ecole des Mines and INRIA, Sophia-Antipolis, `http://zenon.inria.fr/meije/esterel`, April 1997.
4. G. Berry. The Foundations of Esterel. In *Proofs, Languages and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1998. To appear.
5. R. Boyer and Y. Yu. Automated Proofs of Object Code for a Widely used Microprocessor. *Journal of the ACM*, 43(1):166–192, 1996.
6. Z. Chaochen, A. P. Ravn, and M. R. Hansen. An Extended Duration Calculus for Hybrid Real-Time Systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Proc. Workshop on Theory of Hybrid Systems, October 1992, LNCS 736*, pages 36–59, Lyngby, Denmark, 1993. Springer Verlag.
7. M. Croxford and J. Sutton. Breaking through the V and V bottleneck. In *Proceedings of Ada in Europe*. Springer Verlag, 1995.
8. B. Dutertre and V. Stavridou. Formal Requirements Analysis of an Avionics Control System. *IEEE Transactions on Software Engineering*, 25(5):267–278, May 1997.

9. N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.
10. A. Hall. Using Formal Methods to develop an ATC Information System. *IEEE Software*, 12(6):66–76, 1996.
11. M. R. Hansen and Z. Chaochen. Duration Calculus: Logical Foundations. *Formal Aspects of Computing*, pages 283–330, 1997.
12. D. Harel. STATECHARTS: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8:231–274, 1987.
13. M. Heimdahl and N. Leveson. Completeness and Consistency in Heirarchical State-based Requirements. *IEEE transactions on Software Engineering*, 22(6):363–377, June 1996.
14. T.A. Henzinger and P-H. Ho. Model Checking Strategies for Linear Hybrid Systems. In *proc. of Workshop on Formalisms for Representing and Reasoning about Time, as part of the Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, May 1994.
15. D.C. Karnopp, R.C. Rosenberg, and D. Margolis. *System dynamics - A unified approach (2nd edition).* John Wiley & Sons, New York, 1990.
16. 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.
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: Compositional Modelling and Framework for Verification.* PhD thesis, Dept. of Computer and Information Science, Linköping University, March 1994. Dissertation No. 338.
19. S. Nadjm-Tehrani. Time-Deterministic Hybrid Transition Systems. In *Hybrid Systems V, Proceedings of the fifth international workshop on hybrid systems, LNCS, To appear.* Springer Verlag, 1998.
20. S. Nadjm-Tehrani and J-E. Strömberg. From Physical modelling to Compositional models of Hybrid Systems. In Langmaack et al. [16], pages 583–604.
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. S. Nadjm-Tehrani and J-E. Strömberg. JAS-95 Lite: Modelling and Formal Analysis of Dynamic Properties. Technical Report LITH-IDA-R-96-41, Dept. of Computer and Information Science, Linköping University, December 1996. Currently appears on `http://www.ida.liu.se/~snt/activities.html`.
23. M. R. Nielsen. Support for Duration Calculus Verification. Master's thesis, Dept. of Information Technology, Technical University of Denmark, April 1997.
24. N. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Proc. 11th International Conference on Automated Deduction, LNCS 607.* Springer Verlag, 1992.
25. Henry M. Paynter. *Analysis and design of engineering systems.* MIT Press, Cambridge, M.A., 1961.
26. P.J.G. Ramadge and W. M. Wonham. The Control of Discrete Event Systems. *Proceedings of the IEEE*, (77):81–97, March 1989.
27. 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.
28. H. Ruess, N. Shankar, and M. Srivas. Modular Verification of SRT division. In *In proceedings of the International Conference on Computer Aided Verification, CAV'96, LNCS 1102*, pages 123–134. Springer Verlag, 1996.
29. J. U. Skakkebaek and N. Shankar. Towards a Duration Calculus Proof Assistant in PVS. In Langmaack et al. [16], pages 660–679.
30. U. Söderman. *Conceptual modelling of mode switching physical systems.* PhD thesis, Linköping University, Linköping, 1995. Dissertation no. 375.
31. J.-E. Strömberg and S. Nadjm-Tehrani. Hybrid Systems Verification Combining Duration Calculus and Bond Graphs. In *the invited session on Hybrid Dynamic Systems, Proc. IFAC-IFIP-IMACS Conference on Control of Industrial Systems*, pages 481–486. IFAC, 1997.

32. J.-E. Strömberg, S. Nadjm-Tehrani, and J. Top. Switched Bond Graphs as Front-end to Formal Verification of Hybrid Systems. In R. Alur, T.A. Henzinger, and E. Sontag, editors, *Proc. of the DIMACS International Workshop on Verification and Control of Hybrid Systems, LNCS 1066*, pages 282–293. Springer Verlag, 1996.
33. J.E. Strömberg. *A mode switching modelling philosophy.* PhD thesis, Linköping University, Linköping, 1994. Dissertation no. 353.
34. 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.
35. J.L. Top. *Conceptual modelling of physical systems.* PhD thesis, University of Twente, Enschede, 1993.
36. J. van Dijk. *On the role of bond graph causality in modelling mechatronic systems.* PhD thesis, University of Twente, Enschede, 1994.