Formal Refinement and Partitioning of a Fuel Pump System for Small Aircraft in Hybrid Event-B

Richard Banach
School of Computer Science,
University of Manchester, Manchester, U.K.
E-mail: banach@cs.man.ac.uk

***Abstract*—A case study centred on a fuel supply system for a small aircraft is presented in Hybrid Event-B, an extension of conventional Event-B that allows for the modelling and verification of hybrid and cyberphysical systems exhibiting nontrivial continuous behaviour. In contrast to many such case studies, which concentrate predominantly on timing issues, the focus in the present work is on nontrivial physical behaviour, and on the effect that this has on various refinement and partition strategies.**

I. Introduction

In today's ever-increasing interaction between digital devices and the physical world, formalisms are needed to express the more complex behaviours that this allows. Furthermore, these days, it is no longer sufficient to focus on isolated systems, as it is more and more the case that families of such systems are coupled together using communication networks, and can thus influence each others' working. Today, the concept of *Cyber-Physical Systems* [1], [2], [3], [4], [5] has risen to prominence. These new kinds of system throw up novel challenges in terms of design technique, and it is proving more and more difficult to ignore the continuous characteristics in their behaviours, especially if designers want to engineer close to optimal values of system parameters.

The B-Method has long been well extablished as a methodology for modelling and verification of discrete event systems. The standard reference for the classical B-Method is [6]. The classical method emphasised accumulation of submodels into a reference abstract model, to be followed by relatively monolithic refinement of this towards implementation, ending in machine generated compilable and runnable source code (in a language such as C, for example).

In the last decade or so, the B-Method evolved into a more flexible modelling and verification framework, Event-B [7]. In Event-B, action refinement [8], [9], [10] is the main underlying mechanism for using refinement to accumulate design detail. The Event-B approach, and its supporting tool Rodin [11], [12] has proved to be popular in the model based development world [13].

However, despite this, the purely discrete event foundation of Event-B makes it poorly adapted to the needs of continuously evolving behaviour such as that found in cyberphysical systems. Therefore, Hybrid Event-B [14], [15] has been introduced to bring continuous capabilities to the traditionally based discrete Event-B, in order to address some of the challenges referred to. Earlier applications of this formalism include [16], [17], [18], [19]. As described below, traditional discrete Event-B events serve as the 'mode events' that interleave the 'pliant events' of Hybrid Event-B. The latter express the continuously varying behaviour of a hybrid formalism that includes both kinds of event. In this manner, a rigorous link can be made between continuous and discrete update, as needed in contemporary applications.

In this paper, we present a case study based on a fuel pumping system in a small aircraft. Unlike many case studies of cyberphysical systems targetted at the verification domain, where there is an emphasis on timing considerations, there is a preponderance of focus on physical behabviour in this case study, which brings the physical modelling capabilities of Hybrid Event-B to the fore. Besides this, we explore the ramifications of different partition and refinement strategies in the given context. As we explain below, there are non-trivial consequences of different choices regarding these aspects when we have continuous state update, compared with the situation for pure isolated instantaneous state update. These considerations form the main contribution of this case study.

The rest of the paper is as follows. Section II gives the background on the fuel pumping system we study here. Section III briefly outlines the main elements of Hybrid Event-B. Section IV gives the top level model of the fuel system in Hybrid Event-B and covers the first refinement. Section V covers the introduction of non-trivial continuous behaviour. Section VI considers the impact of different strategies for partition and further refinement of the system model. Section VII concludes.

II. A Simplified Aircraft Fuel System

Fig. 1 outlines some elements of a simplified engine fuel delivery system for a light aircraft. The aircraft engine itself, not shown in Fig. 1, receives fuel via a high pressure pump from the relatively small Collector tank. This high pressure system is beyond the scope of our study. The collector tank in turn is fed from the main left and right fuel tanks, contained in the wings. An arrangement of pipework and valves is in place to enable fuel to move from the main tanks to the collector, and between the two main tanks. In addition to these components, there is often also a reserve tank to provide additional fuel supplies for emergency situations. This too is beyond the scope of this study. Many variations on this scheme are possible, and found in practice on various types of aircraft.

Each of the two main tanks has a low pressure pump; these are P_L and P_R in Fig. 1. The pumps have bypass mechanisms so that if the relatively low pump pressure is not sufficient

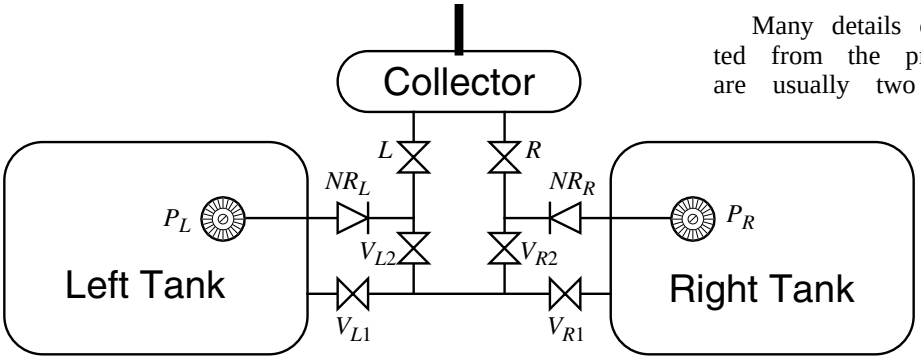

Fig. 1. A schematic of a small aircraft fuel delivery system.

to cause the flow of fuel out of the tank then the fuel is simply returned to the tank without damage to any part of the apparatus (for instance if the needed valves are not open, or if there is no more room in the fuel system downstream of the pump, or if there is a blockage in the pipe system in some inopportune place). This also protects against hydraulic hammer.[1]

Immediately beyond the pumps are non-return valves NR_L and NR_R. Beyond the non-return valves there are various pipes and valves to allow various flow arrangements as described. L and R are the (two way) valves that allow fuel to move into the collector tank from the left and right main tanks respectively. There are also further two way valves $V_{L1}, V_{L2}, V_{R1}, V_{R2}$. Two fuel gauges, G_L and G_R, inform the cockpit of the current amount of fuel in the tanks.

It is clear that if (say) all the right valves are closed, and all the left valves are open, then at least part of any fuel pumped from the left tank will return to the tank via V_{L1} and V_{L2}, depending on the relative hydraulic resistence in the various pipes, decreasing the flow into the collector tank, even though L is open. So it is important that in order to achieve a desired movement of fuel, not only must certain valves be open, but others must also be closed.

Two controls are provided within the framework we work with in this paper. The **fuel pump control** may be $OFF, BOTH, LEFT, RIGHT$. Also the **fuel rebalance control** may be $OFF, L2R, R2L$. These controls are independent, aside from the constraint that it is forbidden that when the engine is being fed by a single pump, P_L or P_R, that that same pump, P_L or P_R respectively, is simultaneously rebalancing fuel to the other tank.

In the framework of this paper, we treat the output of the fuel gauges as information for the pilot. This information can obviously influence the pilot's decisions on the use of the fuel pump and fuel rebalance controls, but for this paper, the gauges remain outside the control loop. In a realistic system, there will be various signals in the cockpit when the current state of the fuel system enters an undesirable regime, but we do not include such considerations in this paper.

[1]Hydraulic hammer is the phenomenon of shock waves propagating round a hydraulic circuit following sudden movements in parts of the circuit, such as when valves are switched on or off in a high pressure circuit. Hydraulic hammer can cause severe damage to equipment if not defended against properly.

Many details of a practical system have been omitted from the preceding account. For example, there are usually two pumps per tank, one mechanically driven from the engine for normal operation, and the other electrically driven, for engine startup, and as a fallback in case the other pump fails.

Aside from the features noted above, the fuel system of an aircraft must have a large number of additional capabilities. It must function properly, keeping the engine fed with fuel, if (even a considerable amount of) water gets into the fuel system (which must also be prevented from freezing). It must not allow an excessive amount of air into the system (which could cause engine failure) regardless of the altitude that the aircraft reaches. Along with the preceding, the fuel tanks must be properly vented to the outside air so that depletion of fuel does not cause negative relative pressure in the tanks, (which would cause potential starvation of the fuel supply to the engine). Venting notwithstanding, the fuel system must keep working properly even when the aircraft is flying upside down (if it is licensed to do so). The fuel system must prevent ignition of fuel vapour when the aircraft is hit by lightning. The list goes on. A good idea of the true complexity of the fuel supply system problem may be gained from Chapter 14 of [20].

III. A Brief Outline of Hybrid Event-B

In this section we briefly outline Hybrid Event-B. The bulk of this material refers to a single machine. However, we will need to consider multiple machines, so we include what we need for multiple machines below. For lack of space, we just indicate the essentials. More detail is included in the context of the machines of our case study.

A. Single Hybrid Event-B Machines

Unlike Event-B (and most other discrete event systems), in Hybrid Event-B, all variables are functions of time (which is read-only) explicitly or implicitly. So time is specially declared, as are clocks, which behave like time but can be reset. Variables are of two kinds. There are *mode variables* which change their values via discontinuous assignment in *mode events*. There are also *pliant variables* which are allowed to change continuously, such change being specified via *pliant events*, which have a non-zero duration.

Invariants are an important element. These are state properties that must hold *at all moments of time* during a run, and proof that the invariants indeed hold in this way constitutes the most important handle on correctness offered by the Hybrid Event-B methodology.

The remainder of a machine consists of events. Mode events are analogues of events in discrete Event-B and define instantaneous updates to the state. Pliant events are new to Hybrid Event-B. They specify the continuous evolution of the pliant variables over an interval of time, by various permitted means: one way is by direct assignment to a time dependent expression; another way is to specify an ordinary differential equation that the variable has to satisfy; a third

way is to demand that the continuous evolution satisfies a given predicate on states for as long as it lasts. Any sensible combination of these is permitted for the family of pliant variables of a machine.

Briefly, the semantics of a Hybrid Event-B machine consists of a set of *system traces*, each of which is a collection of functions of time, expressing the value of each machine variable over the duration of a system run. A run starts at some initial moment of time t_0, and lasts either for a finite time, or indefinitely. The duration of the run, $\mathcal{T}$, an interval of the reals, breaks up into a succession of left-closed right-open subintervals: $\mathcal{T} = [t_0 \ldots t_1), [t_1 \ldots t_2), [t_2 \ldots t_3), \ldots$. Mode events take place at the isolated times corresponding to the common endpoints of these subintervals t_i. In between, the mode variables are constant, and the pliant events stipulate continuous change in the pliant variables.

We insist that on every subinterval $[t_i \ldots t_{i+1})$ the behaviour is governed by a well posed initial value problem [21]. Time t_{i+1} is defined as the earliest time at which a mode event becomes enabled, at which point the continuous behaviour is preempted, the mode event executes, and a further pliant event is executed after its completion. A system run is *well formed*, and thus belongs to the semantics of the machine, provided that at runtime:

(1) Every enabled mode event is feasible, i.e. has an after-state, and on its completion enables a pliant event (but does not enable any mode event).[2]

(2) Every enabled pliant event is feasible, i.e. has a time-indexed family of after-states, and EITHER:
   (i) During the run of the pliant event a mode event becomes enabled. It preempts the pliant event, defining its end. ORELSE
   (ii) During the run of the pliant event it becomes infeasible: finite termination. ORELSE
   (iii) The pliant event continues indefinitely: nontermination.

Thus, in a well formed run mode events alternate with pliant events. We refer to [14] for a more detailed presentation (and to [15] for the extension to multiple machines). The presentation just given is quite close to typical modern formulations of hybrid systems, e.g. [22], [23], or [24] for a perspective stretching further back.

B. Multiple Hybrid Event-B Machines

To model large systems, multi-machine configurations are certainly desirable. At minimum, they partition the functionality, allowing limited focus and independent working. In a framework like Hybrid Event-B this throws up two classes if issues, one structural and the other conceptual.

On the structural side are mundane issues about how to organise multiple machines syntactically. In Hybrid Event-B, the PROJECT construct does this job. It names the machines constituting the system, also the INTERFACE constructs that hold the shared variables and invariants concerning them. It handles any instantiation issues that may arise (in a component based approach), and most importantly, any *synchronisations* needed to ensure various mode events in different machines execute simultaneously.

[2]If a mode event has an input, the semantics assumes that its value only arrives at a time strictly later than the previous mode event, ensuring part of (2) automatically.

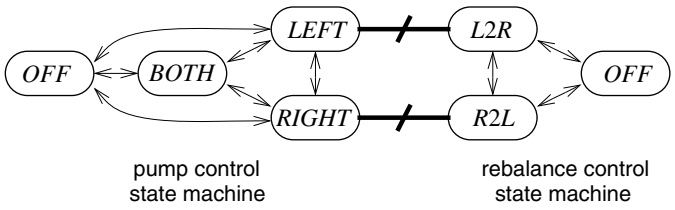

Fig. 2. The top level fuel delivery system transition diagram. The pump state diagram is on the left and the rebalance control state diagram is on the right. The heavy crossed lines connect the only forbidden pairs of states. Otherwise, every pair of states, and every transition involving one or other of the pump or rebalance controls, is permitted.

On the conceptual side lies the observation that because of the inclusion of continuous behaviour, *all* components are always executing *some* event (always a pliant event, except for a discrete set of times). An integrated representation risks hitting the combinatorial explosion of needing to represent each possible combination of concurrent activities within a separate event — whereas a properly partitioned decomposition into multiple machines can delegate the combinatorial explosion to the concurrent semantics of the overall system, leading to economy in the description. We encounter this in the case study below. See [15] for a detailed discussion of all these issues.

IV. Top Level Fuel System Models

We now embark on the modelling of the fuel system in Hybrid Event-B, and on uncovering the insights this can offer.

The state machine view of the fuel supply system is shown in Fig. 2. This consists of two state machines, corresponding to the pump control and the rebalance control. The overall state machine is the product of these two, aside from the two forbidden states indicated by the heavy struck through lines. For the sake of simplicity, we will assume that these state machines are implemented in the cockpit by a set of four press-and-latch buttons for the pump control, and a set of three press-and-latch buttons for the rebalance control, with, in addition, a mechanical interlock to prevent the engagement of the forbidden states. We assume that pressing-and-latching any button of either set causes the release of the previously depressed button from the set.

The state level view merely reflects the changes of configuration of the fuel system that can be effected by the pilot. And although we have described it in purely mechanical terms, there is, of course, no reason that such state control should not be implemented digitally in a modern light aircraft.

The mode level view is captured in the Hybrid Event-B machine *FuelPump_0* of Fig. 3. The machine has two variables, *pumpctrl* and *rebalctrl*, with the obvious meanings, and the values each can take are described in the first two lines of the INVARIANTS clause. The remaining invariants describe the forbidden configurations. The remainder of the machine describes the EVENTS that are available. There are events to manipulate the fuel control: *PumpOFF*, *PumpBOTH*,

```
MACHINE FuelPump_0
VARIABLES pumpctrl, rebalctrl
INVARIANTS
  pumpctrl ∈ {OFF, BOTH, LEFT, RIGHT}
  rebalctrl ∈ {OFF, L2R, R2L}
  pumpctrl = LEFT ⇒ rebalctrl ≠ L2R
  pumpctrl = RIGHT ⇒ rebalctrl ≠ R2L
  rebalctrl = L2R ⇒ pumpctrl ≠ LEFT
  rebalctrl = R2L ⇒ pumpctrl ≠ RIGHT
EVENTS
  INITIALISATION
    BEGIN pmpctrl, rebalctrl := OFF, OFF END
  PumpOFF
    BEGIN pumpctrl := OFF END
  PumpBOTH
    BEGIN pumpctrl := BOTH END
...  ...
```

```
...  ...
  PumpLEFT
    WHEN rebalctrl ≠ L2R
    THEN pumpctrl := LEFT END
  PumpRIGHT
    WHEN rebalctrl ≠ R2L
    THEN pumpctrl := RIGHT END
  RebalOFF
    BEGIN rebalctrl := OFF END
  RebalL2R
    WHEN pumpctrl ≠ LEFT
    THEN rebalctrl := L2R END
  RebalR2L
    WHEN pumpctrl ≠ RIGHT
    THEN rebalctrl := R2L END
  PliTrue
    STATUS pliant
    COMPLY INVARIANTS END
END
```

Fig. 3. The top level Hybrid Event-B model of the fuel delivery system. The pilot's view.

PumpLEFT, *PumpRIGHT*; and events to manipulate the rebalance control: *RebalOFF*, *RebalL2R*, *RebalR2L*. These are all *mode events* in Hybrid Event-B parlance, i.e. they specify instantaneous changes of state at isolated moments of time. To this extent they are just like (conventional) Event-B events, and the notation is deliberately kept the same. However, since Event-B describes the behaviour at *all* moments of time, each Event-B machine must have at least one *pliant event*, to cover continuous behaviour between the isolated mode events. In *FuelPump_0* this duty is covered by the *PliTrue* event, which simply stipulates default compliance with the *INVARIANTS* any time a mode event is not executing.

A certain amount of previous experience [16], [17], [18], [19], has shown that focusing first on a mode description of a desired design is highly beneficial in organising the refinement based development of a complex hybrid system in a perspicuous manner. In the present case we follow the same strategy, but notice first that the mode level description we gave is not yet at the pumps and valves level of the description in Section II, so is not yet good to interface with the more physical behaviour we wish to capture in this case study. Accordingly, we refine the pilot's command level view of the fuel supply system in *FuelPump_0*, to a version concerned with the pumps and valves in machine *FuelPump_1*, given in Fig. 4.

FuelPump_1 is a formal refinement of *FuelPump_0*, as stated in the REFINES clause. In this paper we save space by not declaring which events are refinements of others when name identity suggests it. Likewise we omit events' STATUS declarations except for pliant events. STATUS declarations distinguish mode events from pliant events, and record other pragmatic properties of events. The most important of these omitted from the mode events of this paper is the async property, which allows mode events to execute lazily (i.e. *not* as soon as they are enabled, the latter (eager execution) being the default according to the semantics of Hybrid Event-B [14]).

The variables of *FuelPump_1* are named by analogy with the description in Section II. Pumps are either *OFF* or *ON*, while valves are either *CL*osed or *OP*en. After declaring the variables and their allowed values, the remainder of the INVARIANTS of *FuelPump_1* are *joint invariants*, concerned with expressing the relationship between the *FuelPump_0* variables and the *FuelPump_1* variables (in traditional Event-B manner). Our assumptions about how the controls work result in a relatively simple correspondence between pilot controls and settings of the pumps and valves. The joint invariants make clear that the fuel control can be implemented using just the pumps and their valves to the collector (L for pump P_L), while the rebalancing control can be implemented using the various $V_{\_}$ valves. This makes for a particularly clean design. It is easy to imagine that if the pipework depicted in Fig. 1 were controlled in a different way, then the correspondence between the two levels could come out more complicated. The events of *FuelPump_0* can now be translated to corresponding *FuelPump_1* events straighforwardly, if a little more verbosely.

It has to be admitted that the clean design is partly a consequence of deliberate oversimplification. Thus the only practical way of achieving fuel rebalancing is if the pumps are *BOTH* on. That way part (but only part) of the flow of one pump is diverted to refilling the other tank. But we might wish to rebalance on the ground, without the other pump running. Or we might wish (in the air) to feed the engine using one pump and use the other pump exclusively for rebalancing. Both scenarios are impossible in our setup since they couple the state of the L and R valves to the state of *rebalctrl*. Representing such things would be entirely possible, at the cost of a more detailed, longer description. We avoid doing so to save space in this relatively short paper.

V. Physical Behaviour of the Pumps

In Fig. 5 we find the next level of detail in our development. For the first time we introduce some pliant variables to represent the continuous behaviour. We focus exclusively on the flow rates in the pipework of the model of Fig. 1. The fuel flow rate to the engine is flr_E. The flow rate generated by the left pump is flr_L, while the right pump generates flr_R. The flow rates actually delivered to the engine by the two pumps are $flrd_L$ and $flrd_R$ respectively. The rebalancing flow rates are flr_{L2R} and flr_{R2L}.

The new invariants describe some of the properties of these flow rates. The engine flow rate is a real number, and

MACHINE *FuelPump_1*
REFINES *FuelPump_0*
VARIABLES
$pumpP_L, pumpP_R, valveL, valveR$
$valveV_{L1}, valveV_{L2}, valveV_{R1}, valveV_{R2}$
INVARIANTS
$pumpP_L, pumpP_R \in \{OFF, ON\}$
$valveL, valveR, valveV_{L1}, valveV_{L2}, valveV_{R1}, valveV_{R2} \in \{CL, OP\}$
$pumpctrl = OFF \Leftrightarrow pumpP_L = OFF \wedge pumpP_R = OFF \wedge valveL = CL \wedge valveR = CL$
$pumpctrl = LEFT \Leftrightarrow pumpP_L = ON \wedge pumpP_R = OFF \wedge valveL = OP \wedge valveR = CL$
$pumpctrl = RIGHT \Leftrightarrow pumpP_L = OFF \wedge pumpP_R = ON \wedge valveL = CL \wedge valveR = OP$
$pumpctrl = BOTH \Leftrightarrow pumpP_L = ON \wedge pumpP_R = ON \wedge valveL = OP \wedge valveR = OP$
$rebalctrl = OFF \Leftrightarrow valveV_{L1} = CL \wedge valveV_{R1} = CL \wedge valveV_{L2} = CL \wedge valveV_{R2} = CL$
$rebalctrl = L2R \Leftrightarrow valveV_{L1} = CL \wedge valveV_{R1} = OP \wedge valveV_{L2} = OP \wedge valveV_{R2} = CL$
$rebalctrl = R2L \Leftrightarrow valveV_{L1} = OP \wedge valveV_{R1} = CL \wedge valveV_{L2} = CL \wedge valveV_{R2} = OP$
EVENTS
INITIALISATION
BEGIN
$pumpP_L, pumpP_R, valveL, valveR := OFF, OFF, CL, CL$
$valveV_{L1}, valveV_{L2}, valveV_{R1}, valveV_{R2}, := CL, CL, CL, CL$
END
PumpOFF
BEGIN
$pumpP_L, pumpP_R, valveL, valveR := OFF, OFF, CL, CL$
END
... ...

... ...
PumpBOTH
BEGIN
$pumpP_L, pumpP_R, valveL, valveR := ON, ON, OP, OP$
END
PumpLEFT
WHEN $\neg(valveV_{L1} = CL \wedge valveV_{R1} = OP)$
THEN
$pumpP_L, pumpP_R, valveL, valveR := ON, OFF, OP, CL$
END
PumpRIGHT
WHEN $\neg(valveV_{L1} = OP \wedge valveV_{R1} = CL)$
THEN
$pumpP_L, pumpP_R, valveL, valveR := OFF, ON, CL, OP$
END
RebalOFF
BEGIN
$valveV_{L1}, valveV_{R1}, valveV_{L2}, valveV_{R2} := CL, CL, CL, CL$
END
RebalL2R
WHEN $\neg(pumpP_L = ON \wedge pumpP_R = OFF)$
THEN
$valveV_{L1}, valveV_{R1}, valveV_{L2}, valveV_{R2} := CL, OP, OP, CL$
END
RebalR2L
WHEN $\neg(pumpP_L = OFF \wedge pumpP_R = ON)$
THEN
$valveV_{L1}, valveV_{R1}, valveV_{L2}, valveV_{R2} := OP, CL, CL, OP$
END
PliTrue
STATUS pliant
COMPLY *INVARIANTS* END
END

Fig. 4. Level 1 Hybrid Event-B model of the fuel delivery system. Introduction of the valves and pumps.

is 0 if the engine is switched off, but lies between ERT_{MIN} and ERT_{MAX} otherwise. All the other flow rates are also real and lie between 0 and PRT_{MAX}, which is the maximum rate that can be delivered by either pump. All these variables are initialised to 0. Note that we do not write e.g. $flr_E(t)$ — the time dependence is an automatic consequence of the PLIANT declaration. (N. B. Mode variables are also functions of time, albeit piecewise constant ones.)

The previously introduced mode events remain unchanged, so we turn attention to the pliant event *FlyAircraft*, which refines the earlier *PliTrue*. To understand this pliant event we observe that the fuel is an *incompressible* fluid. On this basis, fuel entering the pipework is instantaneously balanced by fuel leaving the pipework. Thus, the semantics of the pipework system is overwhelmingly one of equality between various quantities. However, the relative dependencies between the various quantities are less clear. The engine demands as much fuel as it needs to function at the power the pilot requests. The pumps, when switched on, wish to pump as hard as they can. The extent they are able to do so depends also on which flows though the pipework are available.

The ANY clause of *FlyAircraft* introduces a number of quantities. All are implicitly time dependent. flr_E^{CH} is the chosen fuel rate corresponding to the pilot's request; it is constrained to the same values as flr_E in the WHERE clause. The other quantities, in pairs, describe how pairs of flows which meet at a single place must be constrained. Thus they are all values in the open interval $(0 \ldots 1)$ (hence all are nonzero), and pairwise, they sum to 1 (reflecting incompressibility), with an additional constraint concerning their relative magnitude. Thus c_L and c_R describe how the raw outputs of the left and right pumps are scaled back when both are feeding the engine (with remaining pump output returned to the relevant tank). They sum to 1, and do not differ by much $|c_L - c_R| < H$, reflecting our expectation that the two pumps are similar. The quantities $c_{L2R,E}$ and $c_{L2R,R}$ describe how the output of the left pump is divided between feeding the engine ($c_{L2R,E}$) and filling the right tank ($c_{L2R,R}$), when rebalancing is set to $L2R$ and the left pump is working. Here we expect the rebalancing to outweigh feeding the engine, reflected in the constraint $c_{L2R,E}/c_{L2R,R} < H$. Similarly for $c_{R2L,E}$ and $c_{R2L,L}$. The same constant H is used for all these constraints, for simplicity.

At the present level of modelling, the dynamics of the fuel system is still very nondeterminstic. The COMPLY clause stipulates what is defined. The first line stipulates that the fuel rate to the engine, flr_E, must be the rate chosen by the pilot, flr_E^{CH}, according to how the aircraft is being flown. The next line says that flr_E is the sum of the delivered fuel rates from the two pumps, $flrd_L + flrd_R$.

The lines after that treat the case when both pumps are switched off. Then, there is no raw output from either pump: flr_L and flr_R both 0. Therefore there is no delivered output

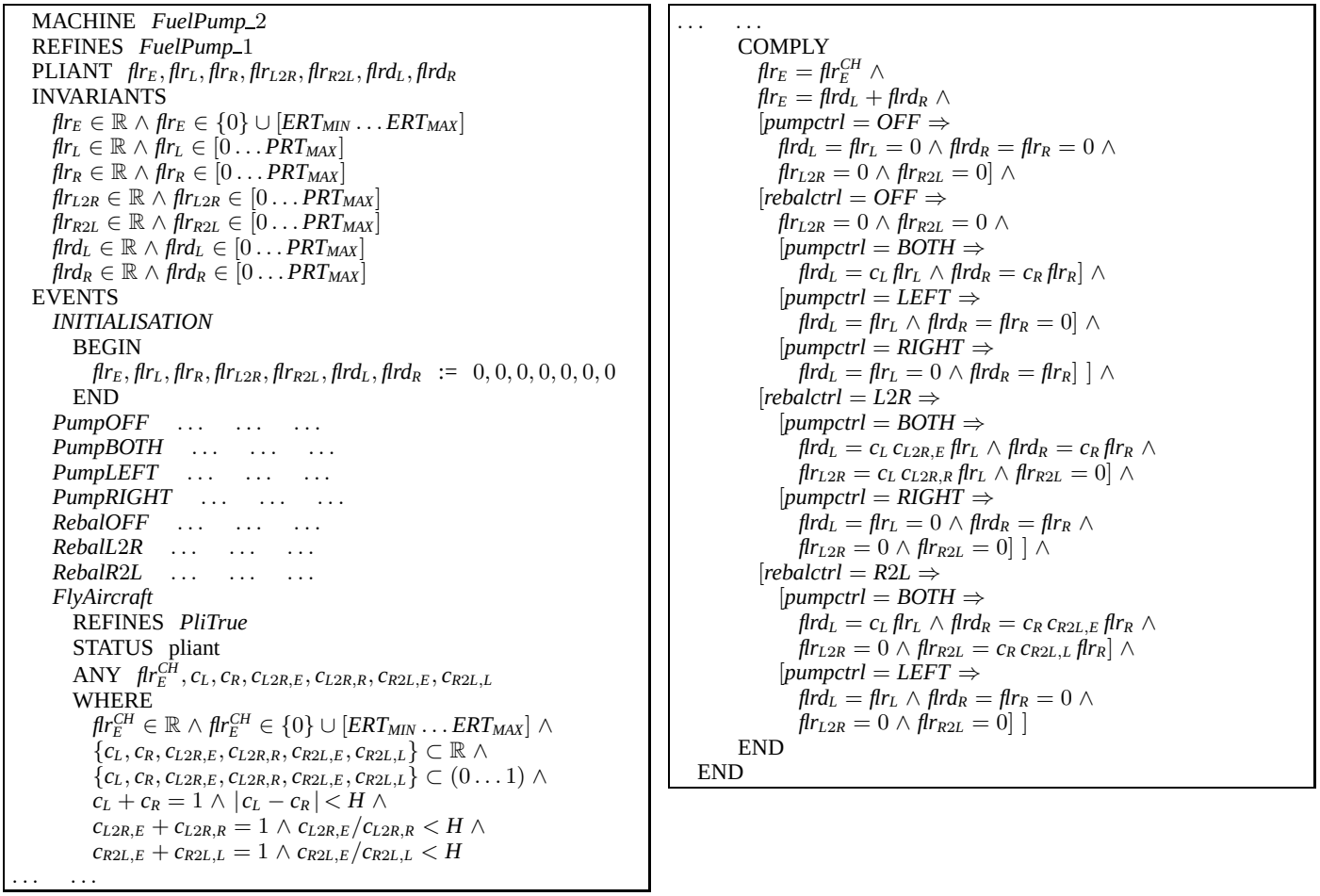

Fig. 5. Level 2 Hybrid Event-B model of the fuel delivery system. Introduction of the fuel flow rates to the engine.

to the engine either. Neither can there be any rebalancing going on: flr_{L2R} and flr_{R2L} both 0, regardless of the setting of the rebalance control. When flr_E^{CH} is nonzero, this case is interesting, since then, the collection of constraints $0 < flr_E^{CH} = flr_E = flrd_L + flrd_R = 0 + 0$ is unsatisfiable. The semantics of Hybrid Event-B stipulates that if the specification of a pliant event becomes infeasible (as is the case here), and there is no mode event enabled at that moment, then the execution stops. Here, it corresponds to the case of the pilot switching the pumps off while the aircraft is flying. This causes engine failure and the aircraft crashes (unless the pilot is able to restart the engine). So this is well represented in our model.

Equally interesting is the case of the engine catching fire. Now, the pilot isolates the fuel supply from the engine to allow the fire to go out: $flr_E^{CH} = 0$. Because the fuel is incompressible, the delivered fuel rates become 0 too, and so any raw pumping outputs drop to zero too (according to cases discussed below), with all pumping output returned to its tank. The pilot can now switch the pumps off while the fire is going out. Once the fire is extinguished, the pilot can switch the pumps on again, and then restart the engine ($flr_E^{CH} > 0$). This sequence of events does not cause infeasibility, so is also well represented in our model.

The remaining cases are easiest to understand according to the setting of *rebalctrl*, starting with the $rebalctrl = OFF$ case. Then, when $pumpctrl = BOTH$, the left and right pumps' delivered output to the engine are respectively proportional to c_L and c_R times their raw output (for that engine demand, the rest going to bypass). When *pumpctrl* is *LEFT* or *RIGHT* then the relevant pump is solely responsible for its delivered output matching the engine demand.

We examine the case when $rebalctrl = L2R$, noting that the *R2L* case is symmetrical. When $pumpctrl = BOTH$, not only are the raw outputs scaled by c_L and c_R, but the left pump's c_L-scaled output is further scaled by $c_{L2R,E}$, a relatively small number, to reflect the small contribution that the left pump makes to feeding the engine in this case, since most of its output (the $c_{L2R,R}$ proportion) will be refilling the right tank. In the $pumpctrl = RIGHT$ case no refilling takes place since the left pump is inactive — it is like the *OFF*/*LEFT* case above. Finally, the $pumpctrl = LEFT$ case is excluded by the invariants (since it is assumed that $c_L\ c_{L2R,E}\ flr_L$ is not enough to feed the engine).

We comment further on the nondeterminism of this specification. Consider the $pumpctrl = BOTH$ case without rebalancing. At any moment, the demand $flr_E^{CH} = flr_E$ is fulfilled by $c_L\, flr_L + c_R\, flr_R$. This is a combination of four nondeterministic quantities, so may supply the needed value in many ways. In reality, what governs the actual flows of fuel in those parts of the system made accessible by the valve settings, is

a combination of: the power of the pumps, the hydrostatic resistances needed to activate the bypass mechanisms in each pump, the relative hydrostatic resistances of the connecting pipework, the hydrostatic resistances in the collector/engine assembly, and the requirement of maintaining a single value of hydrostatic pressure throughout the considered system owing to the incompressible nature of the fuel. Since we do not model these things explicitly, our approach is but an approximation to the reality of such a system, and the nondeterministic (and time dependent) nature of the contributing values makes up for our ignorance of the details. Still, the proportionality factors we use give a reasonable indication of the portion of the pumps' outputs being used in the various cases.

If *rebalctrl* $= L2R$ this aspect is exacerbated. The fuel demand $flr_E^{CH} = flr_E$ is now fulfilled by $c_L\ c_{L2R,E}\ flr_L + c_R\, flr_R$, a more complicated combination of five nondeterministic quantities, with a further expression $c_L\ c_{L2R,R}\ flr_L$ describing the flow to the right tank. Whether this is, in reality, credible as given, with a multiplicative rescaling taking account of the flow distribution, depends again on the factors mentioned. However, the remainder of the paper is not critically affected by this, so we retain this style of description for the sake of simplicity.

VI. Further Refinement and Partitioning

As the development process progresses, the models created get bigger. We circumvented some of this by including only the new material introduced in each step in Figs. 3-5. Eventually though, the detail gets too much for a single syntactic construct. In [15] a number of approaches are described for combining a number of Hybrid Event-B machines and their supporting INTERFACE constructs into a single system with well defined semantics. Some of these are concerned with decomposing a large machine into a number of smaller ones. We can call this *partition in space*.

As well as the preceding though, in this section, we want also to discuss *partition in time*. This is a phenomenon akin to decomposing a sequential program into its individual steps, but for pliant events. The possibility arises rather naturally, since the behaviour of physical equipment is usually governed by local laws, which act largely independently of the context. For example the behaviour of a resistor is given by Ohm's Law, which can be stated independently of the circuit in which the resistor is located. Such a 'global' description can be contrasted with a description of behaviour in particular episodes of time, when the current has some particular value, etc. The latter can be seen as a refinement of the former.

In this section we discuss the tradeoffs between partition in space and partition in time in the context of our case study. We do not have the space to write out our models in full, but the preceding figures do contain enough detail to make the discussion clear. In one respect this is because we had to define the behaviour of the fuel flow in the pipework via an explicit case analysis rather than a single universally applicable physical law.

Regarding partition in space, at the opposite extreme of combining everything into a single machine, is to have every single physical component (which approximately amounts to each individual variable in our description) in a machine of its own. However, a more reasonable partition would split the system into a collector machine (focusing on engine quantities), and left and right tank machines (focusing on the respective pumps and valves). An INTERFACE construct would hold the variables shared by more than one machine.

In such an arrangement, the mode events of Fig. 4 would be split into *synchronised* pairs of events, one for each tank machine, and each holding the assignments from the parent event that updated the variables relevant to that tank. The specification of the synchronisation would be held in the text of the interface mentioned.

Regarding the pliant event of Fig. 5, this would have a portion in each tank machine. Each of these portions would have a case analysis, but it would be simpler than that of *FlyAircraft* in Fig. 5 since each pump can only output one of four cases: $0, flr_X, c_X\, flr_L, c_X\, c_{Y,Z}\, flr_X$. When the machines ran, the relevant cases from each would be selected, moment by moment, according to the external demand and control settings. In the case of Fig. 5, the *FlyAircraft* event is always enabled, and is restarted according to the correct case, after every mode event occurrence at runtime. In the partitioned case, the same thing happens, but the mode event occurrences are synchronised across the component machines, and the selection of the correct cases in the pliant events takes place independently in each component machine.

The observant reader will ask at this point how the constraints like $c_L + c_R = 1$ from the WHERE clause of *FlyAircraft* are handled in this distributed framework, since they involve a coupling between values that are now held in separate machines. For such situations, multi-machine Hybrid Event-B provides I/O capabilities between synchronised events which have the semantics of local bound values. So the left tank machine can choose a suitable $c_L!$ value that it outputs. The right tank machine inputs this value as $c_L?$ and it can then choose a value of c_R that ensures that $c_L? + c_R = 1$ as needed. The semantics of this is instantaneous, so it is identical to the single machine Fig. 5 case.

If we now follow the partition in space by a partition in time, all that happens is that the single pliant event in each component machine gets split into its four cases, each now becoming an event in its own right, this time with a nontrivial entry condition that ensures that only the correct event is enabled after any mode event occurrence.

We can compare the space-then-time approach to its converse: time-then-space. For this approach, the pliant event of Fig. 5 would first be split into all its cases for the partition in time phase. This would create a considerable proliferation of cases that would not be needed later. Then, in the partition in space phase, the mode events would be decomposed as described earlier. For the pliant events, the splitting would also take place as before, but it would generate redundant events. To see this consider a given machine, say the left tank machine for the sake of argument. Then, the pliant events from two parent cases that differed in their right tank provisions but were identical regarding their left tank assignments could generate essentially the same event for the left tank. Clearly this is symptomatic of a combinatorial explosion that is wasteful and should be avoided. A first recommendation to emerge from this discussion would thus be to do partitioning in space before partitioning in time.

However, we can learn more from this discussion. Examining *FlyAircraft* in Fig. 5 we see that it is not a simple statement like Ohm's Law. Rather, it contains a fairly complex case analysis. In the case of a suite of equipment with nontrivial functionality this characteristic is likely to be found often. Thus an improvement on the strategy recommended so far would be to only introduce the pliant behaviours when the system has been decomposed to the extent that each self-contained piece of physical equipment can reside in its own machine. So the recommended Hybrid Event-B development strategy we end up with reads as follows.

- Start by developing the mode view of the system. Use a default pliant event such a *PliTrue* to ensure correct semantics. Restrict refinements to mode events only, until it becomes appropriate to introduce nontrivial pliant behaviour.
- Introduce additional decomposition and synchronisation into the system model as needed, to permit each self-contained piece of physical apparatus to be contained in a machine of its own.
- Introduce the required nontrivial pliant behaviours into the various machines of the system, profiting from the decomposition to avoid the risk of combinatorial explosion in the pliant events.
- Continue to refine until the desired level of detail has been achieved.

The above represents the culmination of the considerations of this section.

VII. Conclusions

In this paper we started by outlining a simplified fuel supply system for a small aircraft. This is a system which contains a preponderance of physical apparatus — which was useful in that it provided a good vehicle for the issues that we wanted discuss. After a very brief overview of the essential elements of Hybrid Event-B, we started to develop the system according to the strategy of attending to the mode event structure first, a strategy that has already proved useful previously. We developed the system to the point where the pliant behaviour of the pump system needed to be brought into the models. We did this in a manner that allowed for fairly straightforward modelling, albeit that the correspondence with physical reality was simplified to an extent, a point we justified on the basis of the extent to which it, in turn, simplified the work we needed to do.

After that, a number of different directions for further development presented themselves. We referred to these as partition in space and partition in time. The detail presented until that stage enabled us to point out the main characteristics of how subsequent development would go along these directions. After some discussion of the alternatives, we came up with a standardised strategy for doing such complex developments in Hybrid Event-B, which we described in the preceding section. This offers concrete recommendations for making best use of the technical devices made available in the Hybrid Event-B formalism, particularly in the multi-machine case.

References

[1] E. Geisberger and M. Broy (eds.), "Living in a Networked World. Integrated Research Agenda Cyber-Physical Systems (agendaCPS)," pp. 1–293, 2015, http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Projektberichte/acaetch_STUDIE_agendaCPS_eng_WEB.pdf.

[2] J. Sztipanovits, "Model Integration and Cyber Physical Systems: A Semantics Perspective," in *Proc. FM-11*, Butler and Schulte, Eds. Springer, LNCS 6664, p.1, http://sites.lero.ie/download.aspx?f=Sztipanovits-Keynote.pdf, 2011, Invited talk, FM 2011, Limerick, Ireland.

[3] J. Willems, "Open Dynamical Systems: Their Aims and their Origins. Ruberti Lecture, Rome," 2007, http://homes.esat.kuleuven.be/~jwillems/Lectures/2007/Rubertilecture.pdf.

[4] "Summit Report: Cyber-Physical Systems," 2008, http://iccps2012.cse.wustl.edu/_doc/CPS_Summit_Report.pdf.

[5] L. Barolli, M. Takizawa, and F. Hussain, "Special Issue on Emerging Trends in Cyber-Physical Systems," *J. Amb. Intel. Hum. Comp.*, vol. 2, pp. 249–250, 2011.

[6] J.-R. Abrial, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[7] ——, *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.

[8] R. Back and K. Sere, "Stepwise Refinement of Action Systems," in *Proc. MPC-89*, vol. 376. Springer, LNCS, 1989, pp. 115–138.

[9] R. Back and J. von Wright, "Trace Refinement of Action Systems," in *Proc. CONCUR-94*, vol. 836. Springer, LNCS, 1994, pp. 367–384.

[10] R. J. R. Back and J. von Wright, *Refinement Calculus: A Systematic Introduction*. Springer, 1998.

[11] J.-R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in Event-B," *Int. J. Soft. Tools for Tech. Trans.*, vol. 12, no. 6, pp. 447–466, 2010.

[12] Rodin Tool, http://www.event-b.org/ http://www.rodintools.org/ http://sourceforge.net/projects/rodin-b-sharp/.

[13] L. Voisin and J.-R. Abrial, "The Rodin Platform has Turned Ten," in *Proc. ABZ-14*, ser. LNCS, Ait Ameur and Schewe, Eds., vol. 8477. Springer, 2014, pp. 1–8.

[14] R. Banach, M. Butler, S. Qin, N. Verma, and H. Zhu, "Core Hybrid Event-B I: Single Hybrid Event-B Machines," *Sci. Comp. Prog.*, vol. 105, pp. 92–123, 2015.

[15] R. Banach, M. Butler, S. Qin, and H. Zhu, "Core Hybrid Event-B II: Multiple Cooperating Hybrid Event-B Machines," 2014, submitted.

[16] R. Banach and M. Butler, "Cruise Control in Hybrid Event-B," in *Proc. ICTAC-13*, ser. LNCS, Liu, Woodcock, and Zhu, Eds., vol. 8049. Springer, 2013, pp. 76–93.

[17] ——, "A Hybrid Event-B Study of Lane Centering," in *Proc. CSDM-13*, Aiguier, Boulanger, Krob, and Marchal, Eds. Springer, 2013, pp. 97–111.

[18] R. Banach, "Pliant Modalities in Hybrid Event-B," in *Proc. Jifeng He Festschrift 2013*, ser. LNCS, Liu, Woodcock, and Zhu, Eds., vol. 8051. Springer, 2013, pp. 37–53.

[19] ——, "The Landing Gear System in Multi-Machine Hybrid Event-B," *Int. J. Soft. Tools for Tech. Trans.*, 2015, to appear.

[20] U.S. Department of Transportation, Federal Aviation Administration, Flight Standards Service, "Aviation Maintenance Technician Handbook — Airframe," 2012, http://www.faa.gov/regulations_policies/handbooks_manuals/aircraft/amt_ airframe_handbook/.

[21] W. Walter, *Ordinary Differential Equations*. Springer, 1998.

[22] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[23] A. Platzer, *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

[24] L. Carloni, R. Passerone, A. Pinto, and A. Sangiovanni-Vincentelli, "Languages and Tools for Hybrid Systems Design," *Foundations and Trends in Electronic Design Automation*, vol. 1, pp. 1–193, 2006.