Core Hybrid Event-B III: Fundamentals of a Reasoning Framework

Richard Banach[a]

[a]*Department of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.*

Abstract

The Hybrid Event-B framework was introduced to add continuously varying behaviour to the discrete changes of state characteristic of the well established Event-B method. This is made necessary by the needs of verifying the hybrid and cyber-physical systems that are increasingly prevalent today. The semantic foundation of Hybrid Event-B rests on piecewise absolutely continuous functions of time. This enables unproblematic modelling of all classical physical phenomena, as well as the specification of conventional discrete changes of state, regardless of whether these arise in the physical arena or as abstractions of computational behaviour. In this paper, the large gap between arbitrary piecewise absolutely continuous functions, and what can be reasoned about mechanically/symbolically, is addressed. First, piecewise absolutely continuous real functions are restricted to piecewise complex analytic functions, real and without singularities on a semi-infinite portion of the real axis. This class has good properties with respect to symbolic manipulation and thus provides a good foundation for an approach to system verification that avoids dealing with the interleaved quantifiers of mathematical analysis, thus reducing the verification of the proof obligations of Hybrid Event-B to calculational checks. The individual proof obligations, whose discharge assures the correctness of a Hybrid Event-B machine, are examined, and results establishing sufficient conditions for their successful discharge via calculation are given. A small scale case study illustrates the verification process in this setting.

1. Introduction

Event-B [3] has been a well established formalism for discrete event modelling, specification and refinement for a long time. Tool support for Event-B via the extensible Rodin toolset [110] has been available for over ten years, and has been downloaded more than 10,000 times by people in most developed countries around the world [129]. However, today's system developments are increasingly targeted at producing embedded and cyber-physical systems, and a purely discrete event perspective proves wanting in the face of the requirements of such developments. For this reason, Hybrid Event-B [24, 25], has been developed as a clean extension of the original discrete event formalism, to permit continuously varying state change alongside the discrete events.

The approach to developing this extension was semantics-first. Piecewise absolutely continuous real functions of time provide a good universe of behaviours in which to ground the foundations of the Hybrid Event-B formalism. Ordinary differential equation (ODE) systems, interrupted by isolated discrete changes of state, yield such functions as solutions, via the Carathéodory interpretation [130]. Other forms of specification, such as direct assignment, or conforming to a constraint, can be defined so as to only yield time dependent behaviours in the same class. This gives a wide range of possibilities for specifying behaviour at the syntactic level.

Email address: `richard.banach@manchester.ac.uk` (Richard Banach)

In [24], henceforth referred to as PaperI, we introduced Hybrid Event-B for single machines. In that paper, we explored extensively the background and motivations for doing the design in the way it was done, and we gave both an informal account of Hybrid Event-B, and a formal semantics, along with a discussion of refinement. We introduced a collection of proof obligations (POs), which, suitably intepreted, enabled us to prove correctness of a Hybrid Event-B machine (corresponding to absence of any 'runtime ABORT'). These were heavily patterned after the discrete Event-B POs, a strategy that works provided time is viewed as a *parameter* and not as a dynamical variable.

Of course, single machines are not enough. The cyber-physical systems we mentioned, are, these days, highly interconnected interacting multi-agent systems, coupling together a typically very heterogeneous collection of subsystems. A Hybrid Event-B design that caters for such situations should be able to model separate subsystems as separate machines, independent and yet interconnected in ways that look convincing from the application domain perspective. In [25], henceforth referred to as PaperII, we introduced a multi-machine extension of the single machine framework. In that paper, we extended the discussion of the issues explored in PaperI to the multi-machine case — specifically showing how we could adapt the single machine POs to the multi-machine case using some relatively lightweight extra syntactic restrictions.

Of course, it is important that a framework such as the one introduced in PaperI and PaperII is shown to be fit for purpose. So, alongside the theoretical investigation, a number of case studies, small and large, and in a variety of application areas, have been performed to confirm the effectiveness of Hybrid Event-B for modelling and reasoning in the embedded and cyber-physical systems arena. Among these, and without derailing ourselves from our goals in this paper with more detailed discussion, we can point to [22, 21, 16, 19, 27, 18, 17, 23, 20].

As far as deployment in practice is concerned, the theoretical framework developed in PaperI and PaperII is of limited use without mechanical support, and so in the present paper we address the question of the mechanised reasoning that could be built into a tool supporting the Hybrid Event-B framework (such as a suitable adaptation of the Rodin toolset [110]). This requires relating the implications of the mathematical basis of the semantics, ultimately to the kind of symbolic manipulations carried out by mechanised reasoners. This is a long journey, though fortunately the literature contains a great deal that can profitably be brought to bear on it. The aim is to trace the ground from the syntactic form of a Hybrid Event-B machine's POs to the symbolic manipulations that can evidence their truth for the given machine. Taking into account the size of this task, the purpose of this paper is thus *strategic*, rather than tactical and detailed, and, since much of the material needed for it is relatively well-known, much low level detail is frequently alluded to rather than covered fully. References are routinely included where further detail can be found.

Given that unrestricted hybrid dynamics is undecidable, a fact that goes almost without saying, to get to a point where mechanisation can be helpful, some restrictions must be imposed in order to be certain that one is dealing with cases with identifiable solutions. Therefore, in the detailed sections of the paper, various constraints on event components, invariants, etc., are introduced. The intention in doing this is not to be prohibitive, or to discourage consideration of additional cases, but to indicate the most readily mechanisable possibilities. Besides, in single piece of work, one has to stop somewhere.

The mathematical storehouse that one can draw on for the kind of mechanisation pursued here is vast, and, along with focusing on specific restrictions and the results that they lead to, from time to time there are digressions on how the specific conditions treated may be extended to yield additional solvable cases. An ideally designed tool for supporting Hybrid Event-B would make it unproblematic to incorporate extensions covering such additional cases.

In dealing with the selected tractable cases, and considering the breadth of material covered, as well as omitting low level detail, the author considered that an element of pedagogical style might be helpful, so the account refrains from being as terse as it might be for specialists in the various topics treated.

The rest of this rather long paper consists of eight parts. Part 1 overviews Hybrid Event-B. Thus, Section 2 introduces the languages of Hybrid Event-B, though it glosses over a number of low level technical details. Section 3 presents a rather minimal cruise control case study for purposes of orientation, and for future reexamination in Section 24. Section 4 briefly recalls the proof obligations (POs) from PaperI that are used to verify a formal development such as the cruise control one, again for purposes of orientation, prior to detailed examination later.

Part 2 focuses on mathematical foundations. Section 5 defines the piecewise absolutely continuous functions that underpin the semantics of Hybrid Event-B, while Section 6 restricts these to those functions which arise as piecewise holomorphic functions, these being exclusively the functions that occur in the overwhelming majority of engineering applications. Section 7 specifically considers the deadlock freeness and well-formedness POs. These raise technical questions that can only be answered by reference to the more precise material in Sections 5 and 6.

After this preparatory material, Part 3 is concerned with the logical foundations that a mechanised prover would need to be based on, in order to connect the task of verifying the POs, to the semantic foundations of Hybrid Event-B. The main aim of the process is to circumvent the many interleavings of quantifiers inherent in the definitions of mathematical analysis, and to replace them with calculational checks, requiring no quantifier handling. Section 8 fixes the lexical and syntactic conventions assumed in the sequel. Section 9 covers the base logic, a natural deduction style framework within which more detailed calculations are expected to take place. Section 10 comments further on this.

Part 4, which is rather short, consisting of Section 11 alone, discusses mode events. There is not much to say about those, since discharging a mode event PO amounts to doing a specific calculation (with whatever calculational tools may be available), there being no quantifiers to be concerned about.

The bulk of the rest of the paper is concerned with pliant event POs, and how to deal with them via calculation. Part 5 looks at general properties of pliant events. Section 12 gathers together a disparate collection of technical notations and definitions that are needed at various points in the treatment of pliant event POs later. Section 13 discusses pliant event basics — their wide generality is now restricted to a range of possibilites that will allow for subsequent definitive reasoning. Section 14 covers the so-called pliant modalities. These are properties such as continuity, that were introduced into Hybrid Event-B as syntactic sugar in [15]. They typically involve reasoning about more than one event in sequence, somewhat like deadlock freeness and well-formedness.

Part 6 embarks on the detailed examination of pliant event POs. Section 15 is a lead-in to the main results of the rest of the paper, indicating the pliant event POs that are covered individually in separate sections later. Section 16 covers the pliant event feasibility PO in detail. Given the expressivity of Hybrid Event-B, there is a lot of case analysis, to clearly separate easy cases from more difficult ones.

Part 7 deals exclusively with machine invariants. Section 17 focuses on the structure of machine invariants, restricting their generality to a limited range of possibilities, deemed to be reasonably tractable, just as Section 13 restricted the permitted form of pliant events. Section 18 covers invariant preservation for pliant events, considering how the restricted pliant events might preserve the restricted machine invariants. Again there is a large amount of case analysis, for the same reasons as before.

Part 8 is concerned with refinement. Section 19 briefly reviews how refinement is proved on the basis of the POs given earlier. Section 20 looks at pliant event feasibility under refinement. This is very similar to feasibility for machine events, so does not require much discussion. Then, Section 21 considers joint invariants, restricting them to a specific, regular relation form, just as other components were restricted earlier. Section 22 covers joint invariant preservation under pliant event refinement, applying the restricted pliant event forms to the restricted joint invariant forms. Again, there is a lot of case analysis — given the importance of the topic, it is vital to identify cases that can be dealt with efficiently. Section 23 briefly considers the refinement POs that contain witness relations.

Part 9 covers remaining issues. Section 24 revisits the earlier cruise control case study in the light

```
 MACHINE $HyEvBMch$
 TIME $t$
 CLOCK $clk$
 PLIANT $x, y$
 VARIABLES $u$
 INVARIANTS
   $x, y, u \in \mathbb{R}, \mathbb{R}, \mathbb{N}$
 EVENTS
   INITIALISATION
     STATUS ordinary
     WHEN
       $t = 0$
     THEN
       $clk, x, y, u := 1, x_0, y_0, u_0$
     END
... ...
```

```
... ...
     MoEv
       STATUS ordinary
       ANY $i?, l, o!$
       WHERE
         $grd(x, y, u, i?, l, t, clk)$
       THEN
         $x, y, u, clk, o! :|$
           $BApred(x, y, u, i?, l, o!,$
           $t, clk, x', y', u', clk')$
       END
... ...
```

```
... ...
     PliEv
       STATUS pliant
       INIT $iv(x, y, t, clk)$
       WHERE $grd(u)$
       ANY $i?, l, o!$
       COMPLY
         $BDApred(x, y, u,$
           $i?, l, o!, t, clk)$
       SOLVE
         $\mathcal{D}x =$
           $\phi(x, y, u, i?, l, o!, t, clk)$
         $y, o! :=$
           $E(x, u, i?, l, t, clk)$
       END
   END
```

Figure 1: A schematic Hybrid Event-B machine.

of the material now covered. Despite the simplicity of the example, a number of surprises emerge when it is examined from the point of view of the POs. Section 25 discusses how the approach described earlier may throw up non-engineering cases, and what to do about that. Section 26 looks at related work, extending the account of PaperI. Finally, Section 27 concludes. In contrast to the focus of the present paper, a more informal and wideranging discussion of issues relevant to the material here appears in [26].

Notation 1.1. *In the sequel, for brevity, we seldom make any distinction between logical notations for entities of interest, and their interpretation in standard set theory. This is, because, in the context of the kind of engineering applications that our investigation is useful for, only the standard set theoretic interpretation is ever of interest.*

Part 1 — Hybrid Event-B Overview

2. Hybrid Event-B, a Sketch

In this section, we briefly review the Hybrid Event-B formalism. This is discussed more carefully in PaperI and PaperII. Inevitably we gloss over many points covered more carefully there, though we include those details needed for the present paper.

In Fig. 1 we see a basic Hybrid Event-B machine, *HyEvBMch*. It starts with declarations of time and of a clock. In Hybrid Event-B, time is a first class citizen in that all variables are functions of time, whether explicitly or implicitly. Time is read-only. Clocks allow more flexibility, since they increase like time, but may be set during mode events. Variables are of two kinds. There are mode variables (like u) which typically take their values in discrete sets and change their values discontinuously during mode events. There are also pliant variables (such as x, y), declared in the PLIANT clause, which typically take their values in topologically dense sets (normally $\mathbb{R}$) and which may change continuously, such change being specified via pliant events.

Next come the invariants. The types of the variables are asserted to be the sets from which the variables' values *at any given moment of time* are drawn. More complex invariants are similarly predicates that are required to hold *at all moments of time* during a run.

Then, the events. The *INITIALISATION* has a guard that synchronises time with the start of any run, while all other variables are assigned their initial values as usual.

Mode events are direct analogues of events in discrete Event-B. They can assign all machine variables (except time itself). In the schematic *MoEv* of Fig. 1, we see three parameters $i?, l, o!$, (an input, a local parameter, and an output respectively), and a guard *grd* which can depend on all the machine variables. We also see the generic after-value assignment specified by the before-after predicate *BApred*, which can specify how the after-values of all variables (except time, inputs and locals) are to be determined.

Pliant events constitute the most obvious distinction between Hybrid Event-B and discrete Event-B. They specify the continuous evolution of the pliant variables over an interval of time. The schematic pliant event *PliEv* of Fig. 1 shows the structure. There are two guards: there is *iv*, for specifying enabling conditions on the pliant variables, enabling conditions that mix mode and pliant variables, and clocks and time; and there is *grd*, for specifying enabling conditions purely involving the mode variables. The separation between the two is motivated by considerations connected with refinement.

The body of a pliant event contains three parameters $i?, l, o!$, (again an input, a local parameter, and an output) which are functions of time, defined over the duration of the pliant event. The behaviour of the event is defined by the COMPLY and SOLVE clauses. The SOLVE clause specifies behaviour fairly directly. For example the behaviour of pliant variable y and output $o!$ is given by a direct assignment to the (time dependent) value of the expression $E(\ldots)$. Alternatively, the behaviour of pliant variable x is given by the solution to the first order ODE $\mathcal{D}x = \phi(\ldots)$, where $\mathcal{D}$ indicates differentiation with respect to time. The COMPLY clause can be used to express any additional constraints that are required to hold during the pliant event via its before-during-and-after predicate *BDApred*. Typically, constraints on the permitted range of values for the pliant variables, and similar restrictions, can be placed here. Of all time dependent behaviours that satisfy these clauses, only piecewise absolutely continuous functions of time which have both left and right limits at all times, and which are right continuous at all times are considered.

The COMPLY clause has another purpose. When specifying at an abstract level, we do not necessarily want to be concerned with all the details of the dynamics — it is often sufficient to require some constraints to hold which express the needed safety properties of the system during the pliant event. The COMPLY clauses of the pliant events can house such constraints directly, leaving it to lower level refinements to add the necessary details of the dynamics.

Briefly, the semantics of a Hybrid Event-B machine consists of a set of *system traces*, each being a set of functions of time, expressing the value of each machine variable over the duration of a run.

Time is modelled as an interval $\mathcal{T}$ of the reals. A run starts at some initial moment of time, t_0 say, and lasts either for a finite time, or indefinitely. The duration of the run $\mathcal{T}$, breaks up into a succession of left-closed right-open subintervals: $\mathcal{T} = \langle [t_0 \ldots t_1), [t_1 \ldots t_2), [t_2 \ldots t_3), \ldots \rangle$. The idea is that mode events (with their discontinuous updates) take place at the isolated times corresponding to the common endpoints of these subintervals t_i, and in between, the mode variables are constant and the pliant events stipulate piecewise absolutely continuous change in the pliant variables. Individual subintervals in such a partition of $\mathcal{T}$, and also typical left-closed right-open intervals that are of interest in the discourse, are generically referred to using the notation $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$.

Insisting on piecewise absolute continuity is equivalent to demanding that the behaviour over time of each variable is a succession of solutions to well posed initial value problems, e.g. $\mathcal{D}xs = \phi(xs \ldots)$, where xs is a relevant tuple of pliant variables and $\mathcal{D}$ is the time derivative [130, 115, 134, 113, 87]. ('Well posed' means that $\phi(xs \ldots)$ has Lipschitz constants which are uniformly bounded over a period of absolutely continuous behaviour, bounding its variation with respect to xs, and that $\phi(xs \ldots)$ is measurable in t.) In between the absolutely continuous pieces, isolated discontinuities are acceptable, modelling the actions of mode events, and of discontinuities coming from the environment.

Pursuing this view, a mode transition becomes a before-/after- pair of variable valuations (the change taking place at a single instant, and corresponding to the (different) one-sided limits at that instant), while a pliant transition becomes a time-parameterised set of pairs of such variable valuations (the common before-valuation being at the beginning of the piecewise absolutely continuous behaviour, and the various after-valuations being indexed by the time interval, the changes no longer taking place at a single instant). From this perspective, many questions regarding mode and pliant events and transitions have very similar answers.

Within any interval in which the variables' behaviour is specified by a pliant event, we seek the earliest time at which a mode event becomes enabled, and this time becomes the preemption point beyond which the pliant behaviour is abandoned, and the next pliant behaviour is scheduled after the completion of the mode event.

In this manner, assuming that the *INITIALISATION* event has achieved a suitable initial assignment to variables, a system run is *well formed*, and thus belongs to the semantics of the machine, provided that at runtime:

[A] 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).[1]

[B] 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. The last event (if there is one) is a pliant event whose duration may be finite or infinite. The relatively simple view of Hybrid Event-B just expounded, persists in suitable form, in a multi-machine system.

3. A Small Case Study: Simplified Cruise Control

In this section we look at a small case study centred on cruise control, to better illustrate the technical details which follow later. It is a reduced version of a study carried out in [22], where a more detailed account may be found. A broader range of case studies has been mentioned in the Introduction.

In Fig. 2 we see the state transition diagram for a simplified cruise control system (CCS). The CCS starts in the *OFF* state, from where it can be *Sw*itched*On* to put it into the *ON* state, which sets the speed of the car that should be maintained automatically. In this state only three things can happen in our reduced version. In one, the driver can *Sw*itch*Off* the CCS. In another, the speed can be *Tip*ped*Up* by the driver to increase it a little. Finally, the speed can be *Tip*ped*Down* to decrease it a little.

The main advantage of using Hybrid Event-B for developments like this is the possibility of incorporating closed loop control directly into the formal development, an option which is much less practicable

[1] If a mode event has an input, the semantics assumes that its value only arrives at a time *strictly later* than the previous mode transition, ensuring part of **[A]** automatically. By this means we can ensure a mode event executes asynchronously. Thus we have *eager mode events* which obey all of **[A]**, and *lazy mode events* which are delayed after a previous mode event completes, regardless of other details of the event's definition. The case study in Section 3 introduces a convenient syntactic sugar for this, and precise technical details are covered in Section 7.

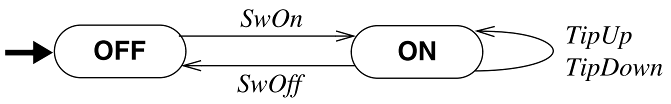

Figure 2: The state transition diagram for a simplified cruise control system.

```
MACHINE CruiseControl_0
VARIABLES mode
INVARIANTS
  $mode \in \{OFF, ON\}$
EVENTS
  INITIALISATION
    STATUS ordinary
    BEGIN mode := OFF END
  PliTrue
    STATUS pliant
    COMPLY INVARIANTS
    END
... ...
```

```
... ...
  SwOn
    STATUS ordinary
    ANY in?
    WHERE $in? = swOn \wedge$
      $mode = OFF$
    THEN mode := ON END
  SwOff
    STATUS async
    WHEN $mode = ON$
    THEN mode := OFF END
... ...
```

```
... ...
  TipUp
    STATUS async
    WHEN $mode = ON$
    THEN skip END
  TipDown
    STATUS async
    WHEN $mode = ON$
    THEN skip END
END
```

Figure 3: Mode level description of cruise control operation.

for a purely discrete formalism like (conventional) Event-B. While most closed loop controller design takes place within the frequency domain [102, 48, 52, 9, 11], the increasingly popular state based formulation of rigorous control theory [122, 7, 71] —especially when supported by tools such as SIMULINK [96]— enables a direct connection with the conceptual framework of Hybrid Event-B to be made.

The development starts with Fig. 3. This is an almost verbatim transcription of the state machine of Fig. 2 into the Hybrid Event-B machine *CruiseControl_0*. The current state of the state machine is represented by the variable *mode*, which changes discontinuously during mode events. In particular, the only pliant behaviour is the default *PliTrue*, which simply demands that the invariants are maintained in between mode transitions.

Note the difference between mode event *SwOn* and the others. *SwOn* has 'STATUS ordinary' and an input *in*? (whose value is never used). According to footnote 1, the input arrives at some moment in time strictly later than the most recent preceding mode transition (but additionally satisfying any other constraints that may be present in the event's guard). Thus, *SwOn* is deemed *not* enabled *by definition* at the completion of any other mode event. The 'STATUS async' of the other mode events is simply syntactic sugar to enable the verbosity of the unused input to be avoided, while still stipulating that they are scheduled at moments strictly later than the previous mode transition.

Machine *CruiseControl_0* is refined to *CruiseControl_1* in Fig. 4. Variables for the cruise control set velocity *setv* and for the actual velocity *v* are introduced, and in the state where cruise control is active (i.e. when $mode = ON$), the pliant behaviour is more specific. Namely, it is stipulated in event *Cruise* that *v* and *setv* do not differ by too great a margin Δ_{Cruise}, that that margin is not exceeded, and that the acceleration is not ever permitted to get bigger than Δ_{MCA}. Since on entry to the $mode = ON$ state, *v* and *setv* are set equal, *Cruise* is clearly feasible. Note though, that the COMPLY clause is only a safety property (presumably addressing a high level requirement), and not a prescription of specific dynamical behaviour. Events *TipUp* and *TipDown* alter the set velocity *setv* discontinuously, by a constant amount, either *TUD*, or as little as is needed to reach $VCC_{\max}$ or $VCC_{\min}$ respectively. Of course, it will not be provable that once one of *TipUp* or *TipDown* has completed, then *Cruise* will have a feasible trajectory with suitable right limit at its start. In such a case, since none of the scheduling possibilities at the end of Section 2 holds, the execution would abort according to the semantics of PaperI.

Machine *CruiseControl_1* is refined to *CruiseControl_2* in Fig. 5. The only difference from the earlier *CruiseControl_1* machine is that the safety properties in *Cruise* are refined to an actual control law. In *Cruise*, we have simple negative feedback control, giving negative exponential drift of the velocity towards the set velocity.

This small development captures the essence of the Hybrid Event-B development philosophy, of introducing more detailed functionality into a system model step by step, policed by refinement, as in

```
MACHINE CruiseControl_1
REFINES CruiseControl_0
PLIANT $v$
VARIABLES $mode, setv$
INVARIANTS
  $v \in [0 \ldots V_{\max}]$
  CONTINUOUS($v$)
  $mode \in \{OFF, ON\}$
  $setv \in [VCC_{\min} \ldots VCC_{\max}]$
EVENTS
  INITIALISATION
    STATUS ordinary
    REFINES INITIALISATION
    BEGIN
      $v :\in [0 \ldots V_{\max}]$
      $setv :\in [VCC_{\min} \ldots VCC_{\max}]$
      $mode := OFF$
    END
  PliDefault
    STATUS pliant
    REFINES PliTrue
    WHEN $mode = OFF$
    COMPLY INVARIANTS
    END
... ...
```

```
... ...
  SwOn
    STATUS async
    REFINES SwOn
    WHEN $mode = OFF \wedge$
      $v \in [VCC_{\min} \ldots VCC_{\max}]$
    THEN
      $mode := ON$
      $setv := v$
    END
  Cruise
    STATUS pliant
    REFINES PliTrue
    WHEN $mode = ON$
    COMPLY
      $|v - setv| \leq \Delta_{\mathrm{Cruise}} \wedge$
      $|\mathcal{D}\, v| \leq \Delta_{\mathrm{MCA}}$
    END
  SwOff
    STATUS async
    REFINES SwOff
    WHEN $mode = ON$
    THEN $mode := OFF$
    END
... ...
```

```
... ...
  TipUp
    STATUS async
    REFINES TipUp
    WHEN $mode = ON$
    THEN
      $setv := \min\{VCC_{\max},$
          $setv + TUD\}$
    END
  TipDown
    STATUS async
    REFINES TipDown
    WHEN $mode = ON$
    THEN
      $setv := \max\{VCC_{\min},$
          $setv - TUD\}$
    END
END
```

Figure 4: Cruise control operation with abstract continuous behaviour.

Event-B. The subtleties built into this simple case study are explored in more detail in Section 24.

4. Hybrid Event-B Proof Obligations

In this section we briefly examine the proof obligations (POs) of Hybrid Event-B. In PaperI it is proved that if these are satistfied for some user written machine, then it is both well defined, and also satisfies the properties claimed for it (e.g. that the invariants declared within it are true at all times), i.e. the machine is semantically correct. Below, we consider how the automated reasoning framework described in Sections 8 onwards, is capable of discharging them.

For simplicity, we focus on the POs relevant to a single machine system. For multi-machine systems, discussed in PaperII, it turns out that the main new issues for POs concern static scope rather than new reasoning facilities; thus they do not impinge on the issues of interest here (typically there are questions of visibility, concerning which variables are needed for which POs in which contexts, and where they might be located in a multi-construct project).

Our overview of the POs uses the following conventions. State variables are u (decorated as needed),

```
MACHINE CruiseControl_2
REFINES CruiseControl_1
PLIANT $v$
VARIABLES $mode, setv$
INVARIANTS
    ... ...
EVENTS
... ...
```

```
... ...
  INITIALISATION ... ...
  PliDefault ... ...
  SwOn ... ...
  SwOff ... ...
  TipUp ... ...
  TipDown ... ...
... ...
```

```
... ...
  Cruise
    STATUS pliant
    REFINES Cruise
    WHEN $mode = ON$
    SOLVE $\mathcal{D}\, v = -C(v - setv)$
    END
END
```

Figure 5: Cruise control operation with continuous control.

Init/FIS $\exists u' \bullet \mathit{Init}_A(u')$

Init/INV $\mathit{Init}_A(u') \Rightarrow I(u')$

MoEv/FIS $I(u) \land \mathit{grd}_{\mathit{MoEvA}}(u,i?,l) \Rightarrow (\exists u',o! \bullet \mathit{BApred}_{\mathit{MoEvA}}(u,i?,l,o!,u'))$

PliEv/FIS
$$\begin{aligned}&I(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{iv}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{grd}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}}))\\ &\quad\Rightarrow (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet [\ (\mathbb{t}_{\mathrm{R}} - \mathbb{t}_{\mathrm{L}} \geq \delta_{\mathsf{Zeno}\mathit{PliEvA}}) \land\ ]\ (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet\\ &\qquad\quad \mathit{BDApred}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t) \land \mathit{SOL}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t))))\end{aligned}$$

MoEv/INV $I(u) \land \mathit{grd}_{\mathit{MoEvA}}(u,i?,l) \land \mathit{BApred}_{\mathit{MoEvA}}(u,i?,l,o!,u') \Rightarrow I(u')$

PliEv/INV
$$\begin{aligned}&I(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{iv}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{grd}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}})) \land (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet \mathrm{TRM}(\mathbb{t}_{\mathrm{R}}) \land (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}},\\ &u(t),i?(t),l(t),o!(t) \bullet \mathit{BDApred}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t) \land \mathit{SOL}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t)))\\ &\quad\Rightarrow (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}} \bullet I(u(t)))\end{aligned}$$

MoPli/WFor
$$\begin{aligned}&\exists u_0, i_0?, l_0, o_0! \bullet I(u_0) \land \mathit{grd}_{\mathit{MoEvA}}(u_0,i_0?,l_0) \land \mathit{BApred}_{\mathit{MoEvA}}(u_0,i_0?,l_0,o_0!,u) \land I(u)\\ &\quad\Rightarrow \neg\,[\ \exists l \bullet \mathit{grd}_{\mathit{MoEvA1}}(u,l) \lor \mathit{grd}_{\mathit{MoEvA2}}(u,l) \ldots \mathit{grd}_{\mathit{MoEvAN}}(u,l)\ ] \land\\ &\qquad [\ (\mathit{iv}_{\mathit{PliEvA1}}(u) \land \mathit{grd}_{\mathit{PliEvA1}}(u)) \lor (\mathit{iv}_{\mathit{PliEvA2}}(u) \land \mathit{grd}_{\mathit{PliEvA2}}(u)) \lor \ldots \lor\\ &\qquad\ (\mathit{iv}_{\mathit{PliEvAM}}(u) \land \mathit{grd}_{\mathit{PliEvAM}}(u))\ ]\end{aligned}$$

PliMo/WFor
$$\begin{aligned}&I(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{iv}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}})) \land \mathit{grd}_{\mathit{PliEvA}}(u(\mathbb{t}_{\mathrm{L}})) \land (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}}, u(t), i?(t), l(t), o!(t) \bullet\\ &\mathit{BDApred}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t) \land \mathit{SOL}_{\mathit{PliEvA}}(u(t),i?(t),l(t),o!(t),t) \land \mathsf{MAXIMAL}(\mathbb{t}_{\mathrm{R}}) \land\\ &\neg\,[\ \exists i?, l \bullet \mathit{grd}_{\mathit{MoEvA1}}(u(t),i?,l) \lor \mathit{grd}_{\mathit{MoEvA2}}(u(t),i?,l) \lor \ldots \lor \mathit{grd}_{\mathit{MoEvAN}}(u(t),i?,l)\ ]))\\ &\quad\Rightarrow \mathsf{WELLDEF}(\mathbb{t}_{\mathrm{R}}) \land [\ \exists i?, l\ \bullet \mathit{grd}_{\mathit{MoEvA1}}(\overrightarrow{u(\mathbb{t}_{\mathrm{R}})}, i?, l) \lor \mathit{grd}_{\mathit{MoEvA2}}(\overrightarrow{u(\mathbb{t}_{\mathrm{R}})}, i?, l) \lor \ldots \lor\\ &\qquad\qquad \mathit{grd}_{\mathit{MoEvAN}}(\overrightarrow{u(\mathbb{t}_{\mathrm{R}})}, i?, l)\ ]\end{aligned}$$

Figure 6: Single machine correctness POs.

inputs are $i?$, outputs are $o!$. The first order predicates used in the POs are those that appear in the syntactic outline of Fig. 1, described in more detail in Section 2, e.g. $\mathit{grd}(\ldots)$, $\mathit{BApred}(\ldots)$, $\mathit{BDApred}(\ldots)$. When a machine is refined, the state variables, inputs, outputs of the refining machine are called $w, j?, p!$. Additional decorations A/C (for abstract/concrete) are used in refinement POs for clarity. The POs themselves are in Figs. 6-8.

Fig. 6 contains the POs that establish correctness of an individual machine. Init/FIS (PaperI (11)) proves the feasibility of initialisation, i.e. an initial state satisfying the definition exists. Init/INV (PaperI (12)) shows that initialisation establishes the invariants $I(u')$. Here u' is the after-state of the initialisation. MoEv/FIS (PaperI (13)) is traditional mode event feasibility, i.e. there is an after-state and output reachable from the before-state and input via *BApred*, when the invariants and event guard hold in the before-state. PliEv/FIS (PaperI (14)) is pliant event feasibility, i.e. there is a time-indexed family of after-states given by *SOL*, satisfying the SOLVE clause and the *BDApred* clause — optionally it asks that the duration of the solution is greater than a Zeno constant $\delta_{\mathsf{Zeno}\mathit{PliEvA}}$ (for cases where it is practicable to prove this).

MoEv/INV (PaperI (15)) is traditional mode event invariant preservation — $I(u')$ is reestablished in the after-state. PliEv/INV (PaperI (16)) is invariant preservation for pliant events, demanding that $I(u(t))$ is established by the solution for all times until the preemption time, which is defined using the TRM predicate.[2] TRM itself is calculated via the disjunction of the guards of all eager mode events; precise details are given in Section 7. MoPli/WFor (PaperI (17)) controls the handover from mode to pliant events: if a mode event makes a valid step, it disables all (eager) mode events and enables some pliant event. PliMo/WFor (PaperI (18)) does the converse: if a pliant event runs for a time which is MAXIMAL

[2]N. B. TRM is omitted in the case of pliant events declared as FINAL, i.e. events permitted to run forever.

Init/FISR	$\exists w' \bullet Init_C(w')$
Init/INVR	$Init_C(w') \Rightarrow (\exists u' \bullet Init_A(u') \wedge K(u',w'))$
MoEv/FISR	$\exists u \bullet K(u,w) \wedge grd_{MoEvC}(w,j?,k) \Rightarrow (\exists w', p! \bullet BApred_{MoEvC}(w,j?,k,p!,w'))$
MoEv/GRDR	$I(u) \wedge K(u,w) \wedge grd_{MoEvC}(w,j?,k) \Rightarrow (\exists i?, l \bullet grd_{MoEvA}(u,i?,l))$
MoEv/INVR	$I(u) \wedge K(u,w) \wedge grd_{MoEvC}(w,j?,k) \wedge BApred_{MoEvC}(w,j?,k,p!,w')$
	$\Rightarrow (\exists i?, l, o!, u' \bullet BApred_{MoEvA}(u,i?,l,o!,u') \wedge K(u',w'))$
MoEv/FISRW	$I(u) \wedge K(u,w) \wedge grd_{MoEvC}(w,j?,k) \wedge BApred_{MoEvC}(w,j?,k,p!,w')$
	$\Rightarrow (\exists i?, l, o!, u' \bullet W(u,i?,l,o!,u',w,j?,k,p!,w'))$
MoEv/GRDRW	$I(u) \wedge K(u,w) \wedge grd_{MoEvC}(w,j?,k) \wedge W(u,i?,l,o!,u',w,j?,k,p!,w')$
	$\Rightarrow grd_{MoEvA}(u,i?,l)$
MoEv/INVRW	$I(u) \wedge K(u,w) \wedge grd_{MoEvC}(w,j?,k) \wedge BApred_{MoEvC}(w,j?,k,p!,w') \wedge W(u,i?,l,o!,u',w,j?,k,p!,w')$
	$\Rightarrow BApred_{MoEvA}(u,i?,l,o!,u') \wedge K(u',w')$
MoEv/NewR	$I(u) \wedge K(u,w) \wedge grd_{NewEvC}(w,j?,k) \wedge BApred_{NewEvC}(w,j?,k,p!,w') \Rightarrow K(u,w')$
MoEv/NewRV	$BApred_{NewEvC}(w,j?,k,p!,w') \Rightarrow V(w') < V(w)$
MoEv/RelDLF	$I(u) \wedge K(u,w) \wedge (\exists o!, p!, u', w' \bullet W(u,i?,l,o!,u',w,j?,k,p!,w')) \wedge$
	$[\, grd_{MoEvA1}(u,i?,l) \vee grd_{MoEvA2}(u,i?,l) \vee \ldots \vee grd_{MoEvAN}(u,i?,l) \,]$
	$\Rightarrow grd_{MoEvC1}(w,j?,k) \vee grd_{MoEvC2}(w,j?,k) \vee \ldots \vee grd_{MoEvCM}(w,j?,k)$

Figure 7: Single machine refinement POs: initialisation and mode events.

during which no mode event is enabled, then (assuming it is not a FINAL event), the termination time must be WELLDEFined and the state values then must enable a mode event. MAXIMAL and WELLDEF are also discussed in Section 7.

The correctness of a refinement is confirmed by POs shown in Figs. 7 and 8. Mode events are dealt with in Fig. 7, starting with Init/FISR (PaperI (20)), which is is feasibility of refined intialisation. Then Init/INVR (PaperI (21)) demands that the after-state w' of initalisation establishes the joint invariant $K(u',w')$ — for which there must now exist a suitably chosen counterpart abstract initial value u'.[3] MoEv/FISR (PaperI (22)) checks feasibility of a refining mode event, in the presence of a suitably chosen abstract value. MoEv/GRDR (PaperI (23)) is mode guard strengthening — if the refining event *MoEvC* is enabled, then its abstract counterpart *MoEvA* must be enabled too. MoEv/INVR (PaperI (24)) is the conventional simulation relationship for mode events — the abstract event must be able to simulate (via the joint invariant) anything the refining event is able to do. The next three POs MoEv/FISRW, MoEv/GRDRW and MoEv/INVRW (PaperI (25)-(27)), redo the work of the previous three —but this time with the help of a witness relation W that is able to supply the existential witnesses required by the original versions— provided the witness relation W itself is feasible (MoEv/FISRW). PO MoEv/NewR (PaperI (28)) is the simplified invariant preservation PO when a 'new' refining mode event —one newly introduced during the refinement and not having an abstract counterpart— refines 'skip', keeping the abstract state unchanged. PO MoEv/NewRV (PaperI (29)) demands that such new mode events (strictly) decrease a VARIANT function V (where V must take values in a well founded set, normally $\mathbb{N}$). MoEv/RelDLF (PaperI (35)) is mode event relative deadlock freeness: if one or more of the abstract mode events is enabled, then at least one refining mode event is enabled, enforcing a kind of synchronisation.

Then come the pliant events in Fig. 8. PliEv/FISR (PaperI (30)) is feasibility of refining pliant

[3]In Event-B, the joint invariant, also known as the gluing invariant or retrieve relation, defines the relationship between the abstract and concrete state spaces. Likewise for Hybrid Event-B.

PliEv/FISR $(\exists u(\mathbb{t}_{\mathrm{L}}) \bullet I(u(\mathbb{t}_{\mathrm{L}})) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}}))$
$\Rightarrow (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet [\ (\mathbb{t}_{\mathrm{R}} - \mathbb{t}_{\mathrm{L}} \geq \delta_{\mathsf{Zeno}PliEvC}) \wedge]\ (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}} \bullet (\exists w(t), j?(t), k(t), p!(t) \bullet$
$BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t)))))$

PliEv/GRDR $I(u(\mathbb{t}_{\mathrm{L}})) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}}))$
$\Rightarrow [\ iv_{PliEvA}(u(\mathbb{t}_{\mathrm{L}})) \wedge]\ grd_{PliEvA}(u(\mathbb{t}_{\mathrm{L}}))$

PliEv/INVR $I(u(\mathbb{t}_{\mathrm{L}})) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \ \Rightarrow$
$\big(\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet \mathrm{TRM}(\mathbb{t}_{\mathrm{R}}) \wedge (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}}, w(t), j?(t), k(t), p!(t) \bullet$
$BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t))$
$\Rightarrow (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet$
$BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge$
$K(u(t), w(t)))))\big)$

PliEv/FISRW $I(u(\mathbb{t}_{\mathrm{L}})) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \ \Rightarrow$
$\big(\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet \mathrm{TRM}(\mathbb{t}_{\mathrm{R}}) \wedge (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}}, w(t), j?(t), k(t), p!(t) \bullet$
$BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t))$
$\Rightarrow (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet W(u(t), i?(t), l(t), o!(t), w(t), j?(t), k(t), p!(t)))))\big)$

PliEv/INVRW $I(u(\mathbb{t}_{\mathrm{L}})) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \ \Rightarrow$
$\big(\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet \mathrm{TRM}(\mathbb{t}_{\mathrm{R}}) \wedge (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}}, w(t), j?(t), k(t), p!(t) \bullet$
$BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge$
$W(u(t), i?(t), l(t), o!(t), w(t), j?(t), k(t), p!(t)))$
$\Rightarrow (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}} \bullet$
$BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge$
$K(u(t), w(t))\big)$

PliEv/RelDLF $I(u) \wedge K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \wedge [\ (iv_{PliEvA1}(u(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvA1}(u(\mathbb{t}_{\mathrm{L}}))) \vee$
$(iv_{PliEvA2}(u(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvA2}(u(\mathbb{t}_{\mathrm{L}})) \vee \ldots \vee (iv_{PliEvAM}(u(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvAM}(u(\mathbb{t}_{\mathrm{L}}))\ ]$
$\Rightarrow [\ (iv_{PliEvC1}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC1}(w(\mathbb{t}_{\mathrm{L}}))) \vee (iv_{PliEvC2}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvC2}(w(\mathbb{t}_{\mathrm{L}})) \vee \ldots \vee$
$(iv_{PliEvCN}(w(\mathbb{t}_{\mathrm{L}})) \wedge grd_{PliEvCN}(w(\mathbb{t}_{\mathrm{L}}))\ ]$

Figure 8: Single machine refinement POs: pliant event POs.

events, in the presence of suitable abstract corresponding values — a time-parameterised family of refining after-states must exist. PliEv/GRDR (PaperI (31)) is pliant guard strengthening, including the optional removal of checking abstract INIT guards (this is connected with details of refinement beyond the scope of this paper; see PaperI). PliEv/INVR (PaperI (32)) is the pliant simulation condition, ensuring a time-parameterised family of before-state/after-state simulations at all the individual moments of time in $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$. The POs PliEv/FISRW (PaperI (33)) and PliEv/INVRW (PaperI (34)) do the same job as PliEv/FISR and PliEv/INVR, but with the help of a time-dependent witness relation W, as for the mode events. Note that there is no PliEv/GRDRW, since the guards of pliant events cannot depend on inputs (so that there is no existential quantification in PliEv/GRDR). Finally there is PliEv/RelDLF (PaperI (36)), essentially, the same as the relative deadlock freeness PO for mode events.

Part 2 — Mathematical Foundations, Deadlock Freeness, Wellformedness

5. Piecewise Absolutely Continuous Functions

The semantics of Hybrid Event-B, as described in PaperI, is cast in terms of (piecewise) absolutely continuous functions of time for each variable. We recall the basic definition.

Definition 5.1 (Absolute Continuity). *Let I be a real interval and $f : I \to \mathbb{R}$ be a function. Then f is absolutely continuous on I iff for every positive ε there is a positive δ such that whenever a finite family of pairwise disjoint subintervals $\{(x_k, y_k)\}_k$ satisfies $\sum_k (y_k - x_k) < \delta$, then $\sum_k |f(y_k) - f(x_k)| < \varepsilon$.*

The key property of absolutely continuous functions is the following [134, 87, 113, 115].

Theorem 5.2. *The following are equivalent.*

1. *f is absolutely continuous on an interval.*
2. *f has a derivative almost everywhere, and increments of f over subintervals of the interval are given by the Lebesgue integral of that derivative over those subintervals.*

Definition 5.3 (*PACcl*). *Let $\mathbb{t}_\mathrm{L}$ be a real number, and let $\mathbb{t}_\mathrm{R}$ be either real or ∞. Let $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ be the left closed right open interval they define, empty if $\mathbb{t}_\mathrm{R} \le \mathbb{t}_\mathrm{L}$ and semi-infinite if $\mathbb{t}_\mathrm{R} = \infty$. Let S be either $\mathbb{R}$ or a set endowed with the discrete topology. Let PACcl$(S)[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ be the set of functions $f : [\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R}) \to S$ which are characterised by the properties:*

1. *$\mathbb{t}_\mathrm{L} < \mathbb{t}_\mathrm{R}$;*
2. *f is **P**iecewise **Ab**solutely[4] **C**ontinuous on $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$;*
3. *all discontinuities of f in $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ are isolated (i.e. are not accumulation points of discontinuities);*
4. *f is right continuous everywhere in $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$;*
5. *f has left limits everywhere in $(\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$, including at $\mathbb{t}_\mathrm{R}$ — implying f has a well defined piecewise continuous extension to the closed interval $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R}]$.*

When any of S or $\mathbb{t}_\mathrm{L}$ and $\mathbb{t}_\mathrm{R}$ are clear from context, or their precise values are not crucial to the discourse, we simplify, writing e.g., PACcl.

Remark 5.4. *The above makes it clear that elements of PACcl$(S)[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ can always be integrated, but can only be differentiated when they have no discontinuities in $(\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$.*

PACcl is the basic building block of the universe within which the dynamics of Hybrid Event-B plays out. The semantics of each state variable v, taking values in a set S as described, during an execution instance of some pliant event *PliEv*, is an element of *PACcl*$(S)[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$. In this, $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ defines the interval of time during which the dynamics of that execution instance of *PliEv* takes place, with $\mathbb{t}_\mathrm{L}$ finite and $\mathbb{t}_\mathrm{R}$ finite or infinite.

The property identified in Theorem 5.2 makes it essentially immaterial whether the behaviour of a state variable is specified via an ODE (in the sense of Carathéodory [130], to allow for the Lebesgue theory), or via a direct assignment to a *PACcl* function, or via a predicate expression (provided the predicate expression is interpreted exclusively in *PACcl* functions). What we said about state variables applies equally to inputs and outputs: *all* functions that figure in a Hybrid Event-B system are required to be in *PACcl*.

[4]If S is not $\mathbb{R}$, then the qualifier ‘absolutely’ is obviously not needed.

We consider the discontinuities permitted by the 'piecewise' of Definition 5.3. Since [73] it has been known that when discontinuities are isolated and have well defined limits on both sides, there can be only countably many of them. In a finite interval $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ the worst that can happen is a Zeno accumulation of smaller and smaller discontinuities occurring closer and closer together towards a limit (which can happen countably many times).

In Hybrid Event-B, discontinuities can enter in various ways. The most obvious is via the occurrence of mode events that impose a finite difference between the left limit and right limit of a state variable's value at some time point. Other possibilities arise via pliant events, for instance: when a direct assignment to a function that is continuous in its variables consumes a discontinuity in one of its inputs; when a direct assignment is to a function that is *prima facie* discontinuous; or combinations of such things. Note that a discontinuity in the RHS of an ODE integrates out the discontinuity, defining a function that has a kink at that point, but is continuous, due to the Carathéodory interpretation.

If we lift the 'absolutely' qualifier, then we get a set of functions that are useful as semantic vehicles for a stochastic version of Hybrid Event-B in which conventional ODEs are generalised to stochastic DEs, whose trajectories are Wiener paths (again piecewise). The semantic space for such behaviours is typically referred to as càdlàg (*continue à droite, limite à gauche*), and the 'cl' in *PACcl* is homage to this. The stochastic generalisation of Hybrid Event-B will be pursued in detail elsewhere.

6. Pieces of Holomorphic Functions

While *PACcl* is in many ways a natural conceptual vehicle for the semantics of Hybrid Event-B, it nevertheless contains (uncountably) many functions which, from an engineering viewpoint, are more general or pathological than needed, in that they are complicated to define and to work with (not to mention that they never arise in engineering problems). In this section we focus on a subset that will form the basis of a practical framework for reasoning about Hybrid Event-B. This subset is based on complex analytic functions [43, 128, 57, 84].

Restricting to complex analytic functions (as opposed to merely real analytic ones) avoids pathologies latent in the real analytic case (such as having a Taylor series that does not yield the original function). At each point where it is defined, a complex analytic function has a Taylor series which yields its value in a non-empty disk of convergence, and generally speaking, the 'routine procedures of engineering mathematics', especially ones based on purely symbolic manipulation, are rendered valid. This is especially important when contemplating automated tools, for which the use of symbolic/algebraic manipulations offers the only feasible approach.

Definition 6.1 (*PH*). *The set of functions $PH[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ is the set of functions f such that:*

1. *$f \in PACcl(\mathbb{R})[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$;*
2. *f is the restriction to the real line interval $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R})$ of a complex function $\hat{f}$ which is holomorphic (i.e. complex analytic) in an open tube surrounding $[\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R}]$ — which is to say that for every $t \in [\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R}]$, there is an open disk $D_t \subseteq \mathbb{C}$, centred on t, such that $\hat{f}$ is analytic in D_t.*

Again, when $\mathbb{t}_\mathrm{L}, \mathbb{t}_\mathrm{R}$ are not crucial, we write e.g., PH.

As noted, the real functions used in 'engineering mathematics' are invariably ***P***ieces of ***H***olomorphic functions, i.e. members of *PH*, and satisfy strong calculational properties. The most heavily utilised ones have names, universally recognised without further qualification, it being understood in both mathematical and applications discourse that those names are never to be overridden.

Proposition 6.2 (*PH* examples). *The following are examples of functions in* $PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$.

1. *constant, polynomial and multinomial functions, that are real on* $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*;*
2. *exponential, trigonometric and hyperbolic functions;*
3. *rational functions with poles away from* $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*, that are real on* $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*;*
4. *(most) 'special functions' (arising from application-inspired ODEs) on suitably chosen* $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*;*
5. *Fourier sine and cosine transforms of square integrable real functions of compact support;*
6. *Laplace transforms of exponentially bounded real functions.*

There are many similar examples (e.g. in case 5, the Fourier transform itself follows in an evident complex extension of PACcl and PH).

Proof: Mostly these are obvious; 5 follows from e.g. Theorem 7.22 in [114]; 6 follows by differentiating under the integral. $\square$

The presence of cases 1 and 2 in Proposition 6.2 already puts the vast majority of applications in the 'classical engineering' sphere within reach of any system based on *PH*, since such applications rely very heavily —in fact almost exclusively— on this class of functions. The other cases are also of help.

The focus on *PH* and its properties enables the design of a reasoning framework which is both *specific* and *extensible*. It is specific in the sense of providing detailed support for selected members of *PH* such as those mentioned in the early cases of Proposition 6.2. It is extensible in providing a framework aligned with the generic properties of *PH* that allows for the addition of user generated information, in order to include provision for fresh, application-specific reasoning facilities when required.

The reason for doing both comes from varying demands for scrutiny in applications demanding varying degrees of dependability. For applications requiring moderate to high levels of dependability, the use of tools like *Mathematica* [95], *Maple* [94], *Scilab* [118], *Matlab* [96], etc. to support *PH* may be sufficient, especially when multiple tools are used to provide a measure of redundancy. These tools typically do not expose the detailed reasoning they do to outside inspection, mainly because the algorithms employed constitute valuable commercial secrets.

However, in applications requiring the very highest levels of dependability, scrutiny of the detailed arguments employed at all points of the development can be a requirement for certification. In such cases, the possibility of doing the reasoning in-house as part of the development can open the relevant proof trees to review, albeit at the cost of needing to provide the relevant axiomatic and tactical foundations as part of the development itself.

Proposition 6.3 (*PH* properties). *Let* $f \in PH[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$ *and* $g \in PH[\mathbb{t}^g_{\mathrm{L}}, \mathbb{t}^g_{\mathrm{R}})$*, and let* $[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}}) \cap [\mathbb{t}^g_{\mathrm{L}}, \mathbb{t}^g_{\mathrm{R}}) = [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ *be nonempty. Then:*

1. *Let* $\mathbb{t}^f_{\mathrm{L}} \le \mathbb{t}'_{\mathrm{L}} < \mathbb{t}'_{\mathrm{R}} \le \mathbb{t}^f_{\mathrm{R}}$*, then* $f|_{[\mathbb{t}'_{\mathrm{L}}, \mathbb{t}'_{\mathrm{R}})} \in PH[\mathbb{t}'_{\mathrm{L}}, \mathbb{t}'_{\mathrm{R}})$.
2. *Let* $\diamond \in \{+, -, \times, {}^\wedge\}$*. Then* $f \diamond g \in PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$.
3. *If g is nonzero in* $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ *and* $\lim_{t \to \mathbb{t}_{\mathrm{R}}} g(t) \neq 0$*, then* $f/g \in PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$.
4. *If* $t \in (\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$*, then* $\frac{df}{dt} \in PH[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$.
5. *If* $t \in (\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$*, then* $\int_{\mathbb{t}_{\mathrm{L}}}^t f \, ds \in PH[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$.
6. *Let* $\mathrm{ran}(f) \subseteq [\mathbb{t}^g_{\mathrm{L}}, \mathbb{t}^g_{\mathrm{R}})$*, then* $g \circ f \in PH[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}})$.
7. *Let* $[\mathbb{t}^f_{\mathrm{L}}, \mathbb{t}^f_{\mathrm{R}}) = [\mathbb{t}^g_{\mathrm{L}}, \mathbb{t}^g_{\mathrm{R}}) = [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*, and let* $\star$ *be convolution, namely* $f \star g(t) = \int_{\mathbb{t}_{\mathrm{L}}}^t f(t-s) g(s) \, ds$*. If* $\mathbb{t}_{\mathrm{R}} < \infty$*, then* $f \star g \in PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$*. If* $\mathbb{t}_{\mathrm{R}} = \infty$ *and* $f \in L^1([\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}))$ *and* $g \in L^\infty([\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}))$*, then* $f \star g \in PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$.

Proof: Basically, this is routine; see [128, 43, 115, 75, 84, 57]. For 7, if $\mathbb{t}_{\mathrm{R}} < \infty$, the result follows from 2 and 5. If $\mathbb{t}_{\mathrm{R}} = \infty$, definedness of $f \star g$ follows by essentially bounding g by a constant (since $g \in L^{\infty}$) and using integrability of f (since $f \in L^1$); and local analyticity follows by differentiating under the integral. □

It is clear that whereas an $f \in PACcl(S)[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ can have isolated discontinuities in $(\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$, an $f \in PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ cannot — analyticity forbids it. The members of $PACcl(S)[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ that are of most interest practically, are those that can be assembled from a finite collection of members of *PH* whose domains partition $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$. If the collection is not a singleton, this requires case analysis, to which we pay attention below.

Notation 6.4. *For the sake of definiteness, when needed, we refer to specific functions in PH using their Mathematica names, e.g. Exp, Log, Sin, Cos, ArcTan, ArcCsch, Gamma, BesselJ, etc. Where the function is multiple valued (e.g. Log, ArcTan, ArcCsch), the principal value returned by Mathematica is understood. Where the function is polymorphic —e.g. for Gamma, where there is: the gamma function Gamma(z); or the incomplete gamma function Gamma(a,z), etc.— the number of arguments supplied in actual use disambiguates the possibilities. A generic element of PH is referred to using a name such as PHf or PHg etc., decorated as needed.*

For formal manipulation, the name itself is insufficient, and formally, an element of $PH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ is a triple: $\langle \mathit{PHf}, \phi_L, \phi_R \rangle$, where PHf names the function, and ϕ_L, ϕ_R are expressions specifying $\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}$ respectively. (The expressions specifying ϕ_L and ϕ_R are required also to satisfy $\phi_L < \phi_R$, of course.)

Definition 6.5 (*FPH*). *The set of functions $FPH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ consists of functions f which are **F**inite sequences of abutting elements of PH, e.g.: $\langle \mathit{PHf}_1, \phi_{L,1}, \phi_{R,1} \rangle, \langle \mathit{PHf}_2, \phi_{L,2}, \phi_{R,2} \rangle, \ldots, \langle \mathit{PHf}_n, \phi_{L,n}, \phi_{R,n} \rangle$, where their domains partition $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$. Thus, the following must all hold:*

$$\mathbb{t}_{\mathrm{L}} = \phi_{L,1} < \phi_{R,1} = \phi_{L,2} < \phi_{R,2} = \phi_{L,3} < \ldots < \phi_{R,n-1} = \phi_{L,n} < \phi_{R,n} = \mathbb{t}_{\mathrm{R}} \tag{1}$$

Thus, elements of $FPH[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ can be written:

$$\langle f, \phi_L, \phi_R \rangle \equiv \langle \mathit{PHf}_1, \phi_{L,1}, \phi_{R,1}; \mathit{PHf}_2, \phi_{L,2}, \phi_{R,2}; \ldots; \mathit{PHf}_n, \phi_{L,n}, \phi_{R,n} \rangle \tag{2}$$

We call $\{\mathbb{t}_{\mathrm{L}}, \phi_{L,1}, \phi_{R,1}, \ldots, \phi_{L,n}, \phi_{R,n}, \mathbb{t}_{\mathrm{R}}\} \equiv JP$ the join points of $\langle f, \phi_L, \phi_R \rangle$, and $\{\phi_{R,1}, \ldots, \phi_{L,n}\} \equiv IJP$ the internal join points of $\langle f, \phi_L, \phi_R \rangle$. As usual, when $\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}$ are not crucial, we write e.g., FPH.

6.1. Holomorphic Functions of Several Variables

The functions in *FPH* on various intervals will capture the time dependence of variables in systems specified via Hybrid Event-B. However, the specifications themselves typically require functions of (several) variables, as well as of time. For this purpose, we will use the evident extension of the preceding ideas to ***S***everal variables.

Definition 6.6 (*PHS*). *The set of functions $PHS^n[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ is the set of functions f such that:*

1. *f is real on $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}) \times \mathbb{R}^n$;*
2. *f is the restriction to the real submanifold $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}) \times \mathbb{R}^n$ of a complex function $\hat{f}$ which is holomorphic (i.e. complex analytic) in an open region surrounding $[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}) \times \mathbb{R}^n$ — which is to say that for every $(t,z) \in [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}] \times \mathbb{R}^n$, there is an open polydisk $D_{(t,z)} \subseteq \mathbb{C}^{n+1}$, centred on (t,z), such that $\hat{f}$ is analytic in $D_{(t,z)}$.*

At the cost of additional complexity we could also have partitioned the last n coordinates in the definition of $PHS^n[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ in various ways, but we can manage without this.

Definition 6.7 (*FPHS*). *The set of functions $FPHS^n[\mathbb{t}_L, \mathbb{t}_R)$ consists of functions which are **F**inite sequences of abutting elements of PHS, in the manner of Definition 6.5 and Definition 6.6. Again, when n and $\mathbb{t}_L, \mathbb{t}_R$ are not crucial, we can write PHS and FPHS.*

Even more so than in the case of a single variable, restricting to complex analytic functions of several variables pays dividends in terms of guaranteeing good behaviour (see e.g. [89, 61, 55, 74, 80, 86]).

7. Deadlock Freeness and Wellformedness POs

Regarding the POs introduced in Section 4, mostly they concern a single event, or a pair of events related through refinement. However, a number involve collections of events from the machine, selected according to specific criteria. In this section we examine this more carefully. We recall that mode events are either eager or lazy.

Definition 7.1 (Lazy, Eager Mode Events). *A mode event is **lazy** iff at least one of the following holds:*

1. *It has an input from the environment.*
2. *It has a guard that constrains the value of the time variable or the value of a clock variable.*

*If a mode event is not lazy it is **eager**.*

Eager mode events must be disabled upon the completion of any mode event (i.e. their *grd* is unsatisfiable in the new after-state). Lazy mode events may be enabled upon the completion of a mode event but are deemed to not execute immediately. They are designated as such to obviate the need for verbosity or circumlocution to ensure that events do not fall foul of clause **[A]** of the runtime semantics when written in the most intuitive way. Thus, inputs are deemed not to arrive at the moment of execution of the previous mode event. Likewise, events with a time specification are deemed not to execute again immediately upon completion. This is despite the fact that, due to the instantaneous execution of mode events, the time at which their after-state is established is the same as the time that satisfied their guard, potentially re-enabling them. It is not excluded that mode events satisfying further syntactic criteria may in future also be designated lazy, in order to enhance expressivity. *INITIALISATION*, although written in mode event syntax, is not regarded as in scope for the remarks that follow.

Pliant events are categorised as either final or non-final.

Definition 7.2 (Final, Non-Final Pliant Events). *A pliant event is **final** iff its STATUS field contains the word 'FINAL'. Otherwise it is **non-final**.*

Semantically, a non-final pliant event should enable a mode event at some point during its execution (unless it merely becomes infeasible, signalling finite termination). A final pliant event need not do so.

We now discuss the POs where the remarks above could imply the need for more precision.

MoEv/RelDLF In this PO, the disjunction over abstract mode events, *MoEvA1*,*MoEvA2*,...,*MoEvAN*, includes all abstract mode events. Also, the disjunction over concrete mode events, *MoEvC1*,*MoEvC2*, ...,*MoEvCM*, covers all concrete mode events.

PliEv/RelDLF In this PO, the disjunction over abstract pliant events, *PliEvA1*,*PliEvA2*,...,*PliEvAM*, includes all abstract pliant events. Also, the disjunction over concrete pliant events, *PliEvC1*,*PliEvC2*, ...,*PliEvCN*, covers all concrete pliant events.

Before we consider the wellformedness POs, MoPli/WFor and PliMo/WFor, we consider the auxiliary predicate $\mathsf{TRM}(\mathbb{t}_R)$, the termination predicate for a pliant event, needed for PliEv/INV. We recall that for a pliant event, $\mathbb{t}_L$ will refer to the start time of an execution instance of it.

$\mathsf{TRM}(\mathbb{t}_\mathrm{R})$ This predicate specifies that $\mathbb{t}_\mathrm{R}$ is a suitable preemption point for a pliant behaviour that is specified as in the hypotheses of PliEv/INV. Thus we assume that $I(u(\mathbb{t}_\mathrm{L})) \wedge iv_{PliEvA}(u(\mathbb{t}_\mathrm{L})) \wedge grd_{PliEvA}(u(\mathbb{t}_\mathrm{L})) \wedge BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t)$ holds, for t in a maximal left-closed right-open interval $[\mathbb{t}_\mathrm{L}, t_{\max})$ starting at $\mathbb{t}_\mathrm{L}$. Let

$$\mathbb{t}_{\mathrm{R}\,\mathbf{eager}} \equiv \inf\{s \mid (\exists l \bullet grd_{MoEv}(u(s), l)) \wedge MoEv \in \mathbf{eager} \wedge s \in [\mathbb{t}_\mathrm{L}, t_{\max})\} \tag{3}$$

$$\mathbb{T}_{\mathrm{R}\,\mathbf{lazy}} \equiv \{s \mid (\exists i, l \bullet grd_{MoEv}(u(s), i, l, \tau)) \wedge MoEv \in \mathbf{lazy} \wedge s \in (\mathbb{t}_\mathrm{L}, \mathbb{t}_{\mathrm{R}\,\mathbf{eager}}]\} \tag{4}$$

where, in (4), τ refers to any explicit time or clock dependence. Then:

$$\mathsf{TRM}(\mathbb{t}_\mathrm{R}) \equiv (\mathbb{t}_\mathrm{R} = \mathbb{t}_{\mathrm{R}\,\mathbf{eager}} \vee \mathbb{t}_\mathrm{R} \in \mathbb{T}_{\mathrm{R}\,\mathbf{lazy}}) \tag{5}$$

The apparent complexity of the definition in (5) is motivated by modelling convenience. Thus the infimum in (3) permits $\mathbb{t}_\mathrm{R}$ to define a time at which none of the eager mode events is actually enabled (because the enabled region happens to be non-left-closed). This holds even though a precise preemption moment is required by the semantics (the preemption thus being defined to take place at the relevant boundary point). The same laxity is not permitted for lazy mode events; the input is required to arrive at a moment when the event is in all other respects actually enabled. (If lazy events can be enabled at $\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$, the choice of event to execute next is nondeterministic, but note that a lazy event cannot be scheduled later than $\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$; see PaperI.) Of course, $\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$ may be infinite, and $\mathbb{T}_{\mathrm{R}\,\mathbf{lazy}}$ may be empty, so (5) may specify an unbounded behaviour if both of these hold.

Of more use in the wellformedness (and other) POs are the sets of states obtained by removing the detailed dependence on the trajectory $u(s)$ in (3)-(5). Thus, letting overline denote topological closure, we define:

$$\text{pre-}\mathbb{t}_{\mathrm{R}\,\mathbf{eager}} \equiv \overline{\{u \mid (\exists l \bullet grd_{MoEv}(u, l)) \wedge MoEv \in \mathbf{eager}\}} \tag{6}$$

$$\text{pre-}\mathbb{T}_{\mathrm{R}\,\mathbf{lazy}} \equiv \{u \mid (\exists i, l \bullet grd_{MoEv}(u, i, l, \tau)) \wedge MoEv \in \mathbf{lazy}\} \tag{7}$$

$$\text{pre-}\mathbb{t}_\mathrm{R} \equiv \text{pre-}\mathbb{t}_{\mathrm{R}\,\mathbf{eager}} \cup \text{pre-}\mathbb{t}_{\mathrm{R}\,\mathbf{lazy}} \tag{8}$$

These yield more easily calculable sufficient conditions to check for in a variety of POs.

MoPli/WFor This PO treats the handover from mode events to pliant events. In this PO, the negated disjunction over mode events $MoEvA1, MoEvA2, \ldots, MoEvAN$ in the conclusion ranges over all **eager** mode events of the machine. Moreover, the set of states that the disjunction ranges over must be as in (6), i.e. the relevant closure. This is needed so that the after-state u established by the $BApred_{MoEvA}$ in the hypothesis, which becomes the initial state of the ensuing enabled pliant event, does not fall outside the guard of every eager mode event as required, yet *does* fall on an eager mode event guard boundary. This would imply a zero duration for the ensuing pliant event, since a preemption would be enabled at $\mathbb{t}_\mathrm{L}$, contradicting **[B]** of Section 2. (Note that this consideration does not apply to lazy mode events since they are *axiomatically* not permitted to execute immediately after another mode event.) Besides this, the disjunction over pliant events $PliEvA1, PliEvA2, \ldots, PLiEvAM$ in the conclusion, covers all pliant events.

PliMo/WFor This PO treats the handover from pliant events to mode events. Note that only non-final pliant events are subject to it. Final pliant events *may* be preempted (though they usually aren't); non-final pliant events *must* be preempted. The handover involves the complexities of the termination mechanism, and of the role of the environment in supplying inputs at nondeterministically chosen moments, as discussed above. The latter aspect takes certain considerations beyond the formal sphere of state based formalisms such as the present one, thus making some of the notations used rather imprecise without suitable meta level amplification. The intention of the $\mathsf{MAXIMAL}(\mathbb{t}_\mathrm{R})$ predicate is to express that the pliant behaviour is to continue as long as possible while no eager mode event forces preemption — and besides, the environment refuses to offer an input that would satisfy a lazy mode event's guard. (It is in this manner that the negated mode event guards in the hypotheses $\neg [\, \exists i?, l \bullet grd_{MoEvA1}(u(t), i?, l) \vee \ldots]$ need to be read.) With this understanding, we can take:

$$\mathsf{MAXIMAL}(\mathbb{t}_\mathrm{R}) \;\equiv\; \mathsf{TRM}(\mathbb{t}_\mathrm{R}) \wedge \mathbb{t}_\mathrm{R} < \infty \tag{9}$$

A consequence of (9) is that if $\mathbb{t}_\mathrm{R} \in \mathbb{T}_{\mathrm{R}\,\mathbf{lazy}} - \{\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}\}$, then the environment is committed to supplying the appropriate input at that moment. (This, of course, concerns the appropriateness of the model for the application domain, rather than its mathematical content.)

In the conclusion of PliMo/WFor, we have to cater for the following complication. The pliant behaviour whose preemption we are contemplating, at some moment t may undergo a discontinuity where the after-value (of the state variables) $u(t)$ enables a mode event. But the before-value (i.e. left limit) $\overrightarrow{u(t)}$, may also enable a mode event. In order that models involving edge-triggered phenomena may be handled in Hybrid Event-B, the design of the formalism determines the following.

If the pliant behaviour at $\mathbb{t}_\mathrm{R}$ is continuous, then no complication arises, and the preceding discussion is sufficient since the left and right limits of the ongoing pliant behaviour are the same at the preemption point. If the pliant behaviour at $\mathbb{t}_\mathrm{R}$ is discontinuous though, then the left limit value $\overrightarrow{u(t)}$ is disregarded, and only the right limit value $u(t)$ is considered in determining the enabledness of preempting mode events. The term $\mathsf{WELLDEF}(\mathbb{t}_\mathrm{R})$ and the presence of the optional overarrows in the conclusion of PliMo/WFor are intended to indicate this operational interpretation.

7.1. Consequences of Wellformedness and Termination

One consequence of the above is that if a preempting mode event *MoEv*, enabled by a discontinuity at time $\mathbb{t}_\mathrm{R}$ as just discussed, is selected for execution, it may itself reassign one or more variables used in determining its enabledness. This will override their values at time $\mathbb{t}_\mathrm{R}$, which are discarded. This is the only occasion in which the operational semantics of variables in Hybrid Event-B does not record within the system trace some value that they take during execution. The detailed operational semantics are described in PaperI.

A further consequence of the determination of the termination predicate concerns refinement. When an abstract pliant event is refined to a concrete one, and at runtime the concrete pliant event is not preempted by 'new' events, freshly introduced at the concrete level of abstraction, then the durations of abstract and concrete pliant transitions are the same. This follows from the guard strengthening and relative deadlock freeness POs. This means that the $\mathbb{t}_\mathrm{R}$ values at the two levels must be consistent for a pair of simulating runs. However this does not hold when 'new' concrete transitions occur between the abstract pliant transition start time and end time. In the latter case, only a detailed exploration of the two systems' reachability relations can reveal the necessary consistency.

Part 3 — Logical Basics

8. Formal Reasoning about Hybrid Event-B: Lexical and Syntactic Issues

In this section we embark on the construction of a formal reasoning system suited to the automated manipulation of Hybrid Event-B machines, and aiming for the mechanical proof of the POs described before. We start with lexical considerations.

```
   CONTEXT HEB_Ctx
   EXTENDS A_Ctx
   SETS S,T
   CONSTANTS A,B,a,b
   MATHEMATICA
     Cos, Gamma, BesselJ
... ...
```

```
... ...
   AXIOMS
     A,B,a,b ∈ ...
     S = {A}
     T = {A,B}
   THEOREMS
     S ⊆ T
   END
```

Figure 9: A Hybrid Event-B CONTEXT.

8.1. *Contexts*

Fig. 9 shows a Hybrid Event-B CONTEXT *HEB_Ctx*, which is the container for all the static mathematics needed by any machine which SEES it (and which is not built-in to a practical Hybrid Event-B tool). *HEB_Ctx* EXTENDS another context *A_Ctx*, making *A_Ctx*'s contents freely usable in *HEB_Ctx*. It introduces some (names of) SETS. These are the sets with discrete topology spoken of in Definition 5.3.[5] It introduces some CONSTANTS. These typically name elements of the sets just mentioned, or name pieces of mathematics constructed from the built-in facilities of a supporting tool. The MATHEMATICA section declares *Mathematica* names, to be used by the machine that SEES *HEB_Ctx*.[6]

The AXIOMS section contains facts to be assumed about the preceding constituents. Its most important role is to declare the type (i.e. enclosing basic set) of each constant introduced in the CONSTANTS section. The THEOREMS section also contains facts about the preceding constituents, but unlike AXIOMS (which are assumed to be true) these must be proved (to the satisfaction of the supporting tool).

8.2. *Names and Scopes*

A Hybrid Event-B project consisting of a context and a machine defines a number of classes of names. From the context there are: names of sets, *SetN*; names of constants, *ConN*; *Mathematica* names, *MatN*. From the machine there are: the name of time, *TimN*; names of clocks, *CloN*; names of mode variables, *ModN*; names of pliant variables, *PliN*; names of events, *EveN*; names of inputs, *InpN*; names of local variables, *LocN*; names of outputs, *OutN*; names of locally scoped variables, *LsvN*.

Definition 8.1 (Free and Bound Names). *In a single machine Hybrid Event-B project, the set of free names is given by the union (assumed disjoint):*

$$FreN \;\equiv\; SetN \uplus ConN \uplus MatN \uplus TimN \uplus CloN \uplus ModN \uplus PliN \tag{10}$$

The set of names bound by their enclosing machine is:

$$BoMN \;\equiv\; EveN \tag{11}$$

The set of names bound by their enclosing event is given by:

$$BoEN \;\equiv\; InpN \uplus LocN \uplus OutN \tag{12}$$

The set of names bound by their local scope (lambda abstractions, universal and existential quantifiers, etc.) is:

$$BoLN \;\equiv\; LsvN \tag{13}$$

[5]Some freedom of choice exists regarding how modelling and arithmetic using $\mathbb{N}$ and $\mathbb{Z}$ are treated, since, for purposes of verification their topologies are effectively discrete too. For simplicity in this paper though, we will subsume all matters involving $\mathbb{N}$ and $\mathbb{Z}$ using their counterparts in $\mathbb{R}$, even if a practical tool need not be constrained to do the same.

[6]Evidently, if other systems such as *Maple*, *Scilab*, *Matlab* were used instead, there would be analogous sections here.

The usual syntactic rules apply [28, 70, 45, 39, 38]. Thus, free names do not clash, and may be replaced using a non-clashing global substitution on free names.[7] If a (newly introduced) free name clashes with some bound name, the bound name must be alpha converted to avoid the clash. Bound names may be alpha converted at will, provided they do not clash with any name bound in the bound name's own scope, or with any name bound in an (enclosing) outer scope, or with any free name. If alpha converting a bound name threatens a clash with a name bound in an (enclosed) inner scope, the inner scoped name must also be alpha converted in a manner that avoids any clash.

By introducing a fresh binary lexical constructor, e.g. '.', all names aside from those in *BoLN* may be disambiguated using 'named Bohm chains', e.g. an input *Inputname*? of an event *Eventname* in machine *Machinename* may be renamed *Machinename.Eventname.Inputname*?. In a single machine scenario such as this paper, the top level of this structure is evidently redundant, but it is useful in the multi-machine arena. The named Bohm chain technique does not work for *BoLN* since the scopes for those names are anonymous — the Bohm chains may be extended using numerical indices as in the original Bohm chain concept [28].

The syntactic form of a Hybrid Event-B project is **rectified** iff there are no free name clashes among the subsets of *FreN* in (10), all free names are distinct from all bound names, and for all bound names *bn*, there is exactly one binder such that all occurrences of *bn* appear in the scope of that binder.

Assumption 8.2 (Rectification). *It is assumed that all (single machine) Hybrid Event-B projects are given in a rectified form.*

8.3. Abstract Syntax

In the material below, we will employ a more compact notation for the abstract syntax of events than the user centred notation of Fig. 1. A mode event will be written:

$$MoEv : \boldsymbol{[}\; ll \mid grd \blacksquare BA \;\boldsymbol{]} \tag{14}$$

In (14), *MoEv* is the event name, *ll*, which we write more explicitly as $i?, l, o!$ when needed, are the locals (from *InpN*, *LocN*, *OutN*), *grd* is the guard, and *BA* is (notation specifying) the before-after predicate. Likewise, a pliant event will be written:

$$PliEv : \boldsymbol{\{}\; iv \,;\, grd \mid ll \mid BDA \blacksquare DE \,;\, DA \;\boldsymbol{\}} \tag{15}$$

In (15), *PliEv* is again the event name and *ll* are the locals. The init and guard predicates are *iv* and *grd*, while *BDA* is the before-during-and-after predicate; *DE* and *DA* denote the ODEs and direct assignments respectively of the event. The cute syntax is to enable easy readability of the various elements.

9. Formal Reasoning about Hybrid Event-B: Base Logic

Correctness of Hybrid Event-B machines rests on the POs. But before examining these in more detail, we briefly cover the basic logical apparatus within which the later discussions reside.

There are, of course, many ways of setting up the basic machinery of mathematical logic in a formal way. For consistency with earlier work, we follow the path that has been used in the B-Method since its inception. Specifically, for the conventional (purely discrete) B-Method, the original formulation of the reasoning process was based on a Rasiowa-Sikorski (RS) style natural deduction framework [108]. These days, formal reasoning for (purely discrete) Event-B in the Rodin tool [4, 110] is presented in a sequent calculus [56, 116] style variant of this [1, 3] (though behind the scenes the system is typically

Propositional Rules:

$$\frac{}{\mathsf{H},P\vdash P}\,\text{HYP}\qquad \frac{\mathsf{H}\vdash Q}{\mathsf{H},P\vdash Q}\,\text{MON}\qquad \frac{\mathsf{H},P\vdash Q}{\mathsf{H}\vdash P\Rightarrow Q}\,\text{IMP-R}\qquad \frac{\mathsf{H},P,Q\vdash R}{\mathsf{H},P,P\Rightarrow Q\vdash R}\,\text{IMP-L}\qquad \frac{\mathsf{H}\vdash P\quad \mathsf{H},P\vdash Q}{\mathsf{H}\vdash Q}\,\text{CUT}$$

$$\frac{\mathsf{H},\neg Q\vdash P}{\mathsf{H},\neg P\vdash Q}\,\text{NOT-L}\qquad \frac{\mathsf{H},P\vdash\bot}{\mathsf{H}\vdash\neg P}\,\text{NOT-R}\qquad \frac{}{\mathsf{H},\bot\vdash P}\,\text{FALSE-L}\qquad \frac{\mathsf{H}\vdash P\quad \mathsf{H}\vdash\neg P}{\mathsf{H}\vdash\bot}\,\text{FALSE-R}$$

$$\frac{\mathsf{H},P,Q\vdash R}{\mathsf{H},P\wedge Q\vdash R}\,\text{AND-L}\qquad \frac{\mathsf{H}\vdash P\quad \mathsf{H}\vdash Q}{\mathsf{H}\vdash P\wedge Q}\,\text{AND-R}\qquad \frac{\mathsf{H},\neg P\vdash Q}{\mathsf{H}\vdash P\vee Q}\,\text{OR-R}\qquad \frac{\mathsf{H},P\vdash R\quad \mathsf{H},Q\vdash R}{\mathsf{H},P\vee Q\vdash R}\,\text{OR-L}$$

Quantifier and Equality Rules:

$$\frac{\mathsf{H},\forall x\bullet P,[x:=E]P\vdash Q}{\mathsf{H},\forall x\bullet P\vdash Q}\,\text{ALL-L}\qquad \frac{\mathsf{H}\vdash P\quad x\ \mathsf{nfin}\ \mathsf{H}}{\mathsf{H}\vdash\forall x\bullet P}\,\text{ALL-R}\qquad \frac{\mathsf{H},P\vdash Q\quad x\ \mathsf{nfin}\ \mathsf{H},Q}{\mathsf{H},\exists x\bullet P\vdash Q}\,\text{XST-L}\qquad \frac{\mathsf{H}\vdash[x:=E]P}{\mathsf{H}\vdash\exists x\bullet P}\,\text{XST-R}$$

$$\frac{\mathsf{H}\vdash\exists x\bullet P\quad \mathsf{H},P\vdash Q\quad x\ \mathsf{nfin}\ \mathsf{H}}{\mathsf{H}\vdash\exists x\bullet Q}\,\text{CUT-XST}\qquad \frac{[x:=F]\mathsf{H},E=F\vdash[x:=F]P}{[x:=E]\mathsf{H},E=F\vdash[x:=E]P}\,\text{EQ-LR}\qquad \frac{[x:=E]\mathsf{H},E=F\vdash[x:=E]P}{[x:=F]\mathsf{H},E=F\vdash[x:=F]P}\,\text{EQ-RL}$$

Calculational Base:

Sets: Enumerated, BOOL, $\mathbb{N},\mathbb{N}_1,\mathbb{Z}$. Reals: $\mathbb{R}$. Rationals and real algebraic numbers: $\mathbb{Q},\mathbb{A}$.
Operators: set theoretic $(\varnothing,=,\mapsto,\times,\mathbb{P},\cup,\cap,\in,\ldots)$, boolean $(\mathrm{FALSE},\mathrm{TRUE},\bot,\top,\wedge,\vee,\neg,\Rightarrow,\Leftrightarrow,=,\ldots)$,
arithmetic $(0,1,+,-,*,/,\mathrm{mod},^\wedge,\ldots,=,<,\leq,>,\geq,\ldots)$, dense types $(0,1,+,-,*,/,\ldots,=,<,\leq,>,\geq,\ldots)$.

External call:

$$\frac{\mathsf{H},q\in\mathbb{Q},x\in\mathbb{R}\vdash\mathsf{K}\quad \mathsf{K},q=q_{\mathsf{X}},x=x_{\mathsf{X}}\vdash\mathsf{K}_{\mathsf{X}}\quad \langle\!\langle\,\mathsf{K}_{\mathsf{X}},q_{\mathsf{X}}\in\mathbb{Q},x_{\mathsf{X}}\in\mathbb{R}\vdash P(\ldots E(q_{\mathsf{X}},x_{\mathsf{X}})\ldots)\,\rangle\!\rangle\quad q_{\mathsf{X}},x_{\mathsf{X}}\ \mathsf{new}}{\mathsf{H},q\in\mathbb{Q},x\in\mathbb{R}\vdash P(\ldots E(q,x)\ldots)}\,\text{EXTERN}$$

Figure 10: The base logic used in the B-Method, with the calculational base extended for Hybrid Event-B.

translated into a more deterministic RS system for efficiency, see [1]). Also, nowadays, Rodin includes various alternative provers in a configurable way [111].

Fig. 10 gives the rules of the base logic, reproduced from [3]. Each rule describes a generic step of goal oriented formal reasoning, stating that to prove the consequent (the sequent below the line), it is sufficient to prove the antecedent (the sequent above the line). Each sequent separates the hypotheses to the left of the turnstyle from the goal to its right. Needed hypotheses are written explicitly, with H denoting a set of additional hypotheses.

The propositional rules are routine, with HYP being the most used to close a branch of the proof tree, and CUT being the only 'creative' rule, requiring the invention of P during a backwards step. In [2, 3] there are further derived rules for additional connectives, which we omit here.

The quantifier rules merit a little more comment. In ALL-L, if we are assuming $P(x)$ for all values of x anyway, then we can additionally assume $P(E)$ if it helps in proving Q. Notation $[x:=E]P$ denotes the substitution of all free occurrences of x in P by E. Note that in $[x:=E]P$ the variable x is bound (just as in $\forall x\bullet P$).

In ALL-R, nfin means 'not free (but may occur bound) in'. We need x to have no free occurrences in H when stripping the $\forall$ quantifier, so that its meaning is not inadvertently constrained by any other properties of free occurrences of x in H. Of course, working in a rectified framework as defined earlier, which we can extend to apply to sequents too, the nfin property will hold automatically. Similar comments hold for XST-L and CUT-XST. Rules EQ-LR and EQ-RL simply describe the uniform substitution of equals by equals.

[7]This could be used, for example, to replace *Mathematica* names by (say) *Maple* names, if needed.

The calculational capabilities of the B-Method are rooted in set theory. The contexts discussed in Section 8 allow the introduction of user defined enumerated (finite) sets, usable as types for variables. Additionally, BOOL is regarded as a predefined set consisting of elements TRUE and FALSE (these being *terms*, and having no connection with the truth or falsehood of any logical property, the latter being written $\top$ and $\bot$). Besides these, [3] introduces the naturals $\mathbb{N}$ and positive naturals $\mathbb{N}_1$.

Calculationally, the set theoretic emphasis leads to a rich algebra of constructors and operations on sets. As well as the basics indicated in Fig. 10, there are constructors and operations for relations, and for the many useful operations that one might want to apply to them; likewise when relations are restricted to being functional. This quickly leads to sequences, trees, graphs, and all the other paraphernalia that is so useful in the modelling of discrete applications. Fixpoints, and fixpoint operators, emerge naturally; see [2, 3]. Provided the cardinality of all sets in the discourse remains finite (i.e. there is a finite universal set for each type in the discourse), the truth of any well formed predicate can be decided in principle, at worst by exhaustive enumeration.

The calculational properties of the naturals are introduced via Peano arithmetic. Immediately, Gödel incompleteness leads to undecidability because of the presence of multiplication [121, 120, 93]. In general, the unrestrained combination of logic, set theory and arithmetic, while convenient for modelling, has poor decidability properties, and different provers fare differently in the face of the difficulties. See e.g. [111]. Also, various calculational rules need *well-definedness conditions* to prevent e.g. division by zero. These are discussed in [3]. Most calculations take place in a directed way, reflecting how the same results are arrived at by hand. Thus they are implemented via a suite of (oriented) rewrite rules. And, although Peano arithmetic suffices to highlight the foundational issues, it is not really adequate for manipulating many practical system models, so more direct techniques are often used.

Assumption 9.1 (Base Logic Arithmetic). *It is assumed that the base logic contains facilities for doing arithmetic on naturals and integers $\mathbb{N}$ and $\mathbb{Z}$, to a degree adequate for practical calculation in the conventional (discrete) Event-B world.*

For Hybrid Event-B, the preceding is not enough, as time is real, and smooth state evolution takes place within the reals. Thus:

Assumption 9.2 (Base Logic Dense Types). *It is assumed that the input language of Hybrid Event-B allows variable declarations of real type $\mathbb{R}$. It is assumed that the base logic recognises expressions of other dense types: rationals $\mathbb{Q}$, and real arithmetic numbers $\mathbb{A}$.*

While users will write system models using reals, internally, the verification system will usefully distinguish arbitrary reals from the more manipulable rationals and real arithmetic numbers — both will require internal representations different from integers and their subsets (e.g. pairs for $\mathbb{Q}$; see [100] for a discussion of representations for $\mathbb{A}$). We refer to them as the dense (numeric) types.

Assumption 9.2 is phrased carefully. Reals, rationals and arithmetic numbers are recognised, but it is not claimed that the base logic will evaluate predicates or expressions containing them. It assumed that one or more external systems (e.g. *Mathematica*) will provide an **oracle** to evaluate and reason about these. The schematic rule EXTERN captures this process. It says that in order to prove a predicate P involving an expression E depending on a rational q and a real x under hypotheses H (which include the facts $q \in \mathbb{Q}, x \in \mathbb{R}$), it is sufficient to make new copies[8] q_X and x_X of q and x, to derive a suitable context K for this goal from H and to make a copy K_X of it, and to delegate the required reasoning to an external system, use of which is indicated using heavy angle brackets $\langle\!\langle\, \mathsf{K}_\mathsf{X}, q_\mathsf{X} \in \mathbb{Q}, x_\mathsf{X} \in \mathbb{R} \vdash P(\ldots E(q_\mathsf{X}, x_\mathsf{X}) \ldots) \,\rangle\!\rangle$.

For this to work effectively, we need good separation between the base logic and the logic implemented by the external system. This is a syntactic issue which can be handled in the following way.

[8]The new keyword introduces previously unused identifiers, so preserves the rectified property.

Firstly, there are no operators that combine a numerical value with an element of an enumerated set, so the only possibility for interference between the base logic and calculational facilities for the dense types is in set forming operations, when these involve numeric elements. If we order our numeric types as $\varnothing \prec \mathbb{N}_1 \prec \mathbb{N} \prec \mathbb{Z} \prec \mathbb{Q} \prec \mathbb{A} \prec \mathbb{R}$, then we can cast any numerical expression to the l.u.b. in this order of any of the variables or constants occurring in it. We can then forbid, syntactically, the creation of any set containing elements of a type which in this order is too complex for the base logic to handle. Things are helped by the observation that in hybrid and cyber-physical systems, there is little call for set-theoretically complex constructions involving continuous quantities. Of course, the restriction can be relaxed by degrees, depending on how much effort one is prepared to expend on the interworking of base and external reasoning systems. And while the order $\prec$ is conveniently viewed as straightforward inclusion in informal mathematics, internally, a good deal of care is needed to relate the different representations appropriate to the different subsets of $\mathbb{R}$, akin to the use of explicit inclusion maps in a categorical setting.

10. Formal Reasoning about Hybrid Event-B: Beyond Base Logic

Beyond simple arithmetic, hybrid and cyber-physical systems require, at the very least, working with elements of *PH* such as *Exp*, *Log*, *Sin*, *Cos*, etc. Beyond these, more complex elements such as *Gamma*, *BesselJ*, etc., may, at times, be required. Our approach to these is two-fold.

Firstly, mature readily available tools such as *Mathematica* contain large amounts of knowledge about the kind of *PH* functions mentioned. This includes many facts expressed, both via equalities (such as $Sin^2(t)+Cos^2(t)=1$ for all t), and via inequalities (such as $Exp(-t) \leq 1$ for $t \in [0\ldots+\infty)$). Making use of this knowledge directly is the first choice for supplementing the reasoning power of the base logic. This requires the correct handling of the dialogue between base and external systems.

Secondly, more complex applications may require more specialised human intervention. This can involve both exact and approximate techniques. The NIST Handbook [103] gives a good indication of what can be aimed for regarding exact analytical approaches; such approaches could be incorporated by hand in an interactive extensible tool. Approximate techniques may be based on the observation that if a desired property can be safely approximated using polynomials (e.g. over a finite interval), then quantifier elimination via cylindrical algebraic decomposition (CAD, reviewed below) provides a very powerful decision strategy which can be incorporated into an interactive extensible tool.

Part 4 — Mode Event POs

11. Formal Reasoning about Hybrid Event-B: Mode Events

Correctness of Hybrid Event-B machines rests on the POs. The POs that deal exclusively with mode events are in Figs. 6 and 7, covering both individual machine correctness and the correctness of mode event refinement. These are Init/FIS, Init/INV, MoEv/FIS, MoEv/INV, Init/FISR, Init/INVR, MoEv/FISR, MoEv/GRDR, MoEv/INVR, MoEv/FISRW, MoEv/GRDRW, MoEv/INVRW, MoEv/NewR, MoEv/NewRV, MoEv/RelDLF.

These POs all concern relationships between individual before- and after- values of the machine variables involved, these being the before- and after- values arising in a putative mode transition of the machine. However, since the POs are actually PO schemas, a large number of instances is typically covered by the same PO, and the symbolic reasoning involved in proving such a PO covers all of them.

The maturity of the B-Method, in both its classical and Event-B incarnations, provides a wealth of experience in proving mode event POs and there is little to add here. Suitably instantiated using the ingredients of a mode event such as $\mathit{MoEv} : [\![\, \mathit{ll} \mid \mathit{grd} \blacksquare \mathit{BA} \,]\!]$, the POs translate directly into formulas that can form the consequent of rules such as those that appear in Fig. 10, constituting the starting point for backwards reasoning. Beyond the logical structure catered for in Fig. 10, lie the demands of the calculations needed in each case, which we overviewed in the previous sections.

The main challenge in the mode event POs is the discovery of suitable witnesses for the existentially quantified variables in PO schemas Init/FIS, MoEv/FIS, Init/FISR, Init/INVR, MoEv/FISR, MoEv/GRDR, MoEv/INVR. In this regard, the refinement PO versions which use a witness relation (which has to be supplied, of course), namely MoEv/GRDRW, MoEv/INVRW simplify the problem, at the price of needing to prove the feasibility of the witness relation itself, MoEv/FISRW.

The techniques that can be used to prove the mode event POs just mentioned can also be applied to the pliant event guard strengthening PO PliEv/GRDR, the mode-to-pliant wellformedness PO MoPli/WFor, and the pliant event relative deadlock freeness PO PliEv/RelDLF. This is because these POs refer only to values at a single moment, i.e the initial time point of a pair of pliant transitions, the $\mathbb{t}_{\mathrm{L}}$ of the pliant event(s) that will ensue. The same perspective can also be applied to the pliant-to-mode wellformedness PO PliMo/WFor, if we bear in mind the more subtle details regarding this PO discussed in Section 7.

Part 5 — Pliant POs: Technicalities, Basics, Modalities

12. Technical Definitions and Notations

In this section, we gather a range of technical material and notational conventions that are needed in dealing with pliant events and their transitions at various points in the remainder of the paper.

12.1. Cylindrical Algebraic Decomposition (CAD)

We recall that CAD [100, 29, 36] deals with a boolean ($\wedge$, $\vee$, $\neg$) combination P of equality ($=$) and inequality ($<$, $\leq$, $>$, $\geq$) predicates of multinomial expresssions in n variables (e.g. $57x^3y^2zw$), i.e., a semi-algebraic predicate (when quantification is added). It systematically partitions $\mathbb{R}^n$ into a finite collection of regions, in each of which the truth or falsehood of each individual predicate is provably constant (and thus computable by evaluating at a single point in the region). This leads to the decidability of P as a whole (and to the decidability of quantified variants of P by combining the truth values of suitable subcollections of the regions).

Simple examples of semi-algebraic predicates (some needing reduction to a more standard form for further processing) are:

$$x^3y^2zw \leq 15 \tag{16}$$

$$(\forall w \bullet (\exists y \bullet x^3y^2zw > w^2x)) \tag{17}$$

$$x \leq 1 \wedge y \leq 2 \wedge z \leq 2 \wedge w \leq 2 \Rightarrow x^3y^2zw \leq 15 \tag{18}$$

Nowadays, the CAD technique is supported by readily available tools such as *QEPCAD* [107], *REDLOG* [109] and also *Mathematica* [95]. The *MetiTarski* tool [97, 8] applies these ideas to reasoning about non-polynomial functions. In this paper, we merely assume the existence of these possibilities, without delving deeper into the details. We assume the availability of the requisite calculational oracle, and of its potential for deciding such issues, when necessary.

12.2. Standard, and Multidimensional Calculus notions

For precision later, we state some standard conventions used below.

1. Basic notions from logic, such as conjunctive or disjunctive normal form, are used without comment.
2. When we mention topological notions on $\mathbb{R}^n$ (open set, interior, closure etc.), the Euclidean topology on $\mathbb{R}^n$ is always to be understood. Inner products, e.g. $\langle x, y \rangle \equiv \Sigma_{1 \leq i \leq n} x_i.y_i$, and norms, e.g. $||x||^2 = \langle x, x \rangle$, are also to be understood in the Euclidean sense.
3. Where needed below, we use basic notions from geometry, multivariable calculus, etc., [90, 131, 33]. These include the directional derivative on $\mathbb{R}^n$, written $\mathrm{D}(g;\theta)$, where, in some open region (containing a point of interest x), g is a scalar field and θ is a vector field, and:
$$\mathrm{D}(g;\theta)(x) \equiv \sum_{1 \leq i \leq n} \theta_i(x) \frac{\partial g(x)}{\partial x_i} \tag{19}$$
We also need the Jacobian matrix, written $J_x^y(g)$, where, in some open region (containing a point of interest x), g is a vector field, and:
$$J_x^y(g) \quad \text{is the matrix} \quad \left[\frac{\partial g_i(x)}{\partial x_j} \right] \tag{20}$$
where y_i is a typical coordinate in the range of g.
When the dimensions of domain and range are equal, the determinant of J_x^y is referred to as the Jacobian of g, and if it is non-zero, the inverse function theorem can be applied.
We also use elementary properties of foliations in Section 21.

12.3. Multidimensional Inequalities

Below, several of the pliant modalities that we consider in Section 14 and later use the usual notations for inequalities, $<$, $\leq$, $\geq$, $>$. When applied to single real valued variables, this is uncontroversial. However, when the arguments belong to the multidimensional *FPHS* world, there is less clarity. We deal with this by using a generalisation of the notion of distribution function from probability theory [62, 31, 88].

Definition 12.1 (Generalised Distribution Function). *A function $\eta : \mathsf{X} \to \mathbb{R}$ is a generalised distribution function (GDF) iff:*

1. $\forall z \in \mathbb{R} \bullet \eta^{-1}(z) \neq \varnothing$
2. $a < b \Rightarrow \{x \in \mathsf{X} \mid \eta(x) \leq a\} \subset \{x \in \mathsf{X} \mid \eta(x) \leq b\}$

Definition 12.2 (Generalised Inequality). *Given a GDF η on a multidimensional space X (e.g. $\mathbb{R}^n$), for $x, y \in \mathsf{X}$, we interpret $x \leq y$ to hold iff $\eta(x) \leq \eta(y)$. Similarly for other kinds of inequality and on all other properties that normally depend on the order on $\mathbb{R}$, such as monotonicity, convexity, etc.*

Obviously, a GDF on X generates a partial order on X, but in a way that can be used conveniently in calculations. Natural examples of GDFs η on $\mathbb{R}^n$ are:

1. $\eta((x_1 \ldots x_n)) = \max_j \{x_j\}$
2. $\eta((x_1 \ldots x_n)) = \min_j \{x_j\}$
3. $\eta((x_1 \ldots x_n)) = \Sigma_{1 \leq j \leq n} \mu_j.x_j$ where $\{0\} \neq \{\mu_j\} \subseteq \mathbb{R}$
4. Given fixed $\phi : \mathbb{R}^n \to \mathbb{R}$ and $0 \neq r \in \mathbb{R}^n$, suppose for all $x \in \mathbb{R}^n$, that $\phi(x - \lambda r) = 0$ has a unique solution λ^*; define $\eta((x_1 \ldots x_n))$ to be that λ^*

Many useful GDFs arise by specialising case 4, which works by foliating $\mathbb{R}^n$ with copies of the ϕ hypersurface. For example, choosing $\phi(z) = \langle z, \mu \rangle$ with $r = \mu/||\mu||^2$ recovers case 3, while choosing $\phi(z) = \prod_{1 \leq j \leq n} \diamond_j |z_j|$ where, for each j, $\diamond_j$ is a choice of $+$ or $-$, and with $r = (\diamond_1 1, \diamond_2 1 \ldots \diamond_n 1)$ gives 2^n possibilities corresponding to all the 2^n-ants of $\mathbb{R}^n$. The 'all $-$' choice corresponds to case 1, while the 'all $+$' choice corresponds to case 2. Choosing different nonzero magnitudes for the components of r leads to maxi/minimising the corresponding collections of differently weighted components of x.

Assumption 12.3. *We assume below, whenever needed, that all uses of inequalities in a multidimensional context are interpreted with respect to a specific (even if unstated) GDF. This allows use of the usual notations without proliferating the terminology.*

12.4. Left and Right Limits

Notation 12.4. *We write $\overleftarrow{e(t)}$ for the limiting value of e from the right at time t (i.e. from greater t), and $\overrightarrow{e(t)}$ for the limiting value from the left, where e is an expression. Invariably t is one of $\{\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}\}$. Since all functions in the semantics of any Hybrid Event-B system must be continuous from the right, $\overleftarrow{v(t)} = v(t)$ must always hold. Nevertheless, we retain the left arrow for emphasis.*

12.5. Syntactic and Semantic Quantifiers

Notation 12.5 (Heavy and Light Quantifiers). *We use heavy quantifiers $\boldsymbol{\forall}, \boldsymbol{\exists}$ to range over purely syntactic quantitites e.g. event names or variable names, when the quantified element is syntactic. We use normal quantifiers $\forall, \exists$ to range over semantic values of individual variables (as usual). We write $\boldsymbol{\forall} x$ (when x is a variable and the quantified property is semantic) to signify a conjunction of instances of the semantic property conjoined over the relevant equivalence class of the finest partition of the relevant variables that respects the variable couplings forced by the terms occurring in the property.*

12.6. Proposition Notation

The propositions in the remainder of the paper typically hold true only on the basis of a large number of hypotheses. For economy of presentation, these are given in list form, with list items grouped using *and* and *or*, and with nesting precedence indicated by indentation. If this form were to be rearranged into disjunctive normal form, each disjunct would contain the assumptions needed for a sound inference rule of the more familiar $\frac{\mathit{ANTECEDENTS}}{\mathit{CONSEQUENT}}$ form, as in Fig. 10.

13. Formal Reasoning about Hybrid Event-B: Pliant Event Basics

Unlike the mode event PO schemas, which only concern properties of pairs of before- and after-variable valuations (albeit, typically in a generic manner, covering parameterised families of instances), the pliant event PO schemas concern properties of ranges of variable valuations directly indexed by intervals of time, in which the time dependence may enter non-trivially (and where the PO schemas may possibly be further parameterised in various ways). In Sections 5 and 6 we restricted the kinds of time dependent behaviours we allowed for variables, ending with the family *FPH*. The aim of this was to give scope (insofar as it is possible) for the reasoning needed about pliant event POs involving such functions, to be reduced to the kind of discrete symbolic manipulation that works for mode event POs. That this aim is not unrealistic is a reflection of the fact that it would just amount to a systematisation and formalisation of a subset of the kind of calculations that could successfully be done by hand in more traditional approaches. Then again, achieving the aim is more complicated than for mode events.

In PaperI and PaperII the runtime semantics of Hybrid Event-B systems presumed that any enabled pliant event selected to execute would produce a suitable time dependent behaviour; if no such event was available then the relevant system trace simply ABORTed. Likewise, when proving that the truth of the POs yielded well defined system traces, their truth was simply assumed. None of this required overly precise restriction of the syntactic content of pliant events, beyond some generic properties. In the present paper, we want to focus more closely on the provability of the POs, so the syntactic content of pliant events must be scrutinised with greater precision.

13.1. Differential Equations and Direct Assignments

Let $\mathit{PliEv} : \{\ \mathit{iv}\ ; \mathit{grd}\ |\ \mathit{ll}\ |\ \mathit{BDA} \blacksquare \mathit{DE}\ ; \mathit{DA}\ \}$ be a pliant event. The presence of both differential equations *DE* and direct assignments *DA* pitches *PliEv* into the differential-algebraic equation systems (DAEs) sphere — beyond which, the *BDA* clause in *PliEv*, in principle permits even the most general possible specification $\mathit{BDA} \equiv P(\mathcal{D}xs, xs, t)$, where P is an arbitrary predicate.

Nowadays, DAEs are an area of active research. Basic motivating information on DAEs appears in [65] and in the early part of [60]. A more comprehensive survey is [81], while more specialised material can be found in [82, 78]. The cited works initiate their accounts in the analytic sphere before discussing numerical approaches, the latter being the focus of an even more copious literature. All of these make clear the evident fact that the unrestricted form $P(\mathcal{D}xs, xs, t)$ is not a useful basis for a solution strategy; so we put aside the *BDA* clause for the moment.

Turning to the *DE* and *DA* clauses, most progress is made for linear systems; [81] gives a good overview. Unlike the case of pure ODE systems, where for non-pathological systems a unique solution over some interval is guaranteed for arbitrary initial values (unless contradictory ODEs for the same variable are given, or none), for DAE systems there are many more possibilities, even in the constant coefficients case. We may have: (1) inconsistency (e.g. $x(t) := y(t)\ ||\ y(t) := x(t)+1$); (2) a unique solution with forced initial value (e.g. $x(t) := 1$); (3) a unique solution with arbitrary initial value (e.g. $\mathcal{D}x(t) = 1$); (4) a solution with constrained initial value depending on an arbitrary function (e.g. $x(t) := y(t) + 1$, where y is another state variable rather than an inhomogeneous term); (5) a solution with partly arbitrary initial value depending on an arbitrary function (e.g. $\mathcal{D}x(t) = y(t)$, where y is a state variable again); (6) a solution with forced initial value depending on arbitrary but constrained inhomogeneous terms (e.g. $\mathcal{D}x(t) = f_1(t)\ ||\ x(t) := f_2(t)$). The possibilities proliferate if we permit more general linear combinations of derivatives, and non-constant coefficients.

The above indicates some of the phenomena expressible via DEs and DAs. While overspecification (i.e. inconsistency) is clearly to be avoided, underspecification is acceptable in Hybrid Event-B models, on the understanding that it is needed to capture generic desired properties which can be refined to something more precise at a lower level of abstraction. What we should avoid though, is achieving underspecification in an obscure way. Thus cases like (4) and (5) above are acceptably underspecified, since it is evident that $y(t)$ is unconstrained, whereas the flexibility offered by case (6) should be avoided, since the constraint between $f_1(t)$ and $f_2(t)$ arises much more indirectly.

Linear constant coefficients (LCC) DAE systems are well suported by tools like *Mathematica*. The general theory for linear variable coefficients systems is significantly more complex than the LCC case [81, 82], and naturally enough, they are less comprehensively supported. Accordingly, constant coefficients linear systems are uppermost in our mind in the subsequent discussion of pliant events, though we make as little use as possible of their specific properties in order to maximise the applicability of what we say.

The general LCC case is fully analysed via the classical Kronecker canonical form of the matrix pair containing the coefficients of the variables and of their first order derivatives of the DAE system [10, 85]. This contains blocks capturing overconstrained and underconstrained cases as above, as well as more conventional ODE Jordan and nilpotent blocks. However, by exploiting the syntactic structure of pliant events in Fig. 1 we can go somewhat beyond the LCC case.

Definition 13.1 (EV graph). *Given a pliant event PliEv, its EV graph is a directed graph whose nodes are either: (a) variables occurring in the DE or DA of PliEv (initially tagged 'V'), or (b) known functions*

```
SOLVE
  $z := x + y$
  $\mathcal{D}x = y$
  $\mathcal{D}y = -x + w$
  $w := 3$
END
```

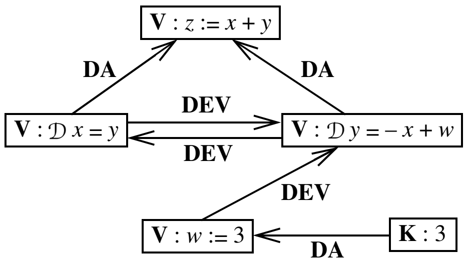

Figure 11: The EV graph of a simple SOLVE clause. The EV tags are shown in bold. The variables and constants in the graph are in the LHSs of the DEs and DAs in the V- and K-tagged boxes. The DEs and DAs are included for illustration.

or constants (tagged 'K'). There is a directed edge (DEV-edge, tagged 'DEV') from w to v precisely when v is the LHS of a differential equation in DE, and variable w (including derivatives of w) occurs in the RHS of that differential equation. There is a directed edge (DEK-edge, tagged 'DEK') from w to v precisely when v is the LHS of a differential equation in DE, and K-tagged node w occurs in the RHS of that differential equation. There is a directed edge (DA-edge, tagged 'DA') from w to v precisely when v is the LHS of a direct assignment in DA, and w (whether tagged V or tagged K, and including derivatives of variables) occurs in the RHS of that direct assignment.

The softened EV graph of PliEv replaces all DEV-edges in its EV graph by undirected edges. PliEv has an admissible EV graph iff:

1. *no variable occurs as the LHS of both a DE differential equation and a DA direct assignment (and so we can refer to DE nodes and DA nodes, according to how the variable in question is assigned), and*
2. *every cycle in the softened EV graph of PliEv (formed by traversing either way along the DEV-edges, and in the direction of other edges) contains no DA-edge.*

The idea behind this definition of the EV graph is that while families of ODEs whose variables are interdependent in a fairly arbitrary manner are guaranteed solutions by the standard existence theorems [130, 64], families of arbitrarily tangled DA systems are not, so we want the graph minor of EV formed by contracting the DEV-edges to be acyclic, allowing for a directed solution strategy flowing up from known functions at the leaves.

Fig. 11 shows a simple example. The directed solution strategy would first substitute the value 3 for w in the ODE for y. Then the (x, y) DE system could be solved (assuming initial values were available). Finally, the derived functional forms of (x, y) could be substituted into the DA for z.

Proposition 13.2 (EV strategy). *Let PliEv have an admissible EV graph. Then, given suitable initial values (see Definition 13.4, and remarks immediately following it), there is a sound directed strategy for solving the DE/DA clauses of PliEv.*

Proof: Since *PliEv* is finite, its EV graph is finite. The solution strategy repeats the following two steps until no more can be done.

Step 1: For every V-tagged node v, all of whose incoming DA-edges have a K-tagged node as their source (so v is the LHS of a *DA*, depending on only known data), evaluate v as a function of time and known information, and retag v with tag K. Repeat until no more can be done.

Step 2: Collect together every V-tagged node v, which is not reachable from any DA-edge which has a V-tagged node as its target (via a path of DA (and DEK) directed edges and DEV undirected edges[9]),

[9]In fact, there will be no DEK edges in such a path

solve the ODE system determined by *DE* for all of these v, and retag these v nodes with tag K.

Soundness of this strategy can be argued as follows. By admissibility, every V-tagged node has either incoming DE-edges exclusively, or incoming DA-edges exclusively. We claim that while there is at least one V-tagged node in the EV graph, then at least one V-tagged node will satisfy the criteria of Step 1 or Step 2.

For suppose not. Choose a V-tagged node v and mark it. If v has incoming DA-edges, one of them will have a V-tagged node w as its source. If w is marked, we have a contradiction as a cycle containing a DA-edge has been constructed. Otherwise w is unmarked, so mark it and move to it. Alternatively, v has incoming DE-edges, at least one of them being a DEV-edge that is reachable (via a path of DA/DEK directed edges and DEV undirected edges) from a DA-edge which has a V-tagged node as its target. Choose w as the source of such a DEV-edge. Mark it and move to it.

Repeat the preceding from the most recently moved to node. Since the path thereby constructed always has a DA-edge in its sights, and the enclosing EV graph is finite, a cycle containing a DA-edge is inevitable. This is a contradiction.

Thus, while the system contains some as-yet-unsolved-for variables, we can apply one or other of the steps, reducing the number of these. So, by finiteness of the EV graph, all variables will be solved for eventually, and thus the proposed solution strategy is sound. $\square$

Regarding the solution strategy, Step 1 concerns substitutions of expressions for the LHS variables, which we regard as doable in principle. Step 2 concerns an ODE system whose RHSs are regarded as either known in principle or dependent on LHS variables of the same ODE system, and which we may thus also regard as solvable in principle. If all the RHSs of the ODE system are linear in the variables and their derivatives then, at worst, we can transform it into a variable coefficients standard form linear ODE system; usually each choice of variable coefficients generates new special functions which may well require *ad hoc* processing. If the coefficients are constant, then the system can be solved using the usual variation of parameters formula [130, 64]. If there are no derivatives in the RHSs, then the system is in standard form already and no precursor transformation is needed.

We remark that the approach just sketched, will produce for the state variables, functions of time which are in *FPHS*, provided any functions occurring in the RHSs of the clauses in *DE* and *DA* are themselves in *FPHS*, and they are combined using the algebraic and function composition operators in Proposition 6.3. For assignments in *DA* this is immediate, while for the differential equations in *DE* it follows from results in [130].

13.2. The COMPLY Clause

If any state variable does not figure in the *DE* or *DA* clauses the definition of Hybrid Event-B stipulates that it should implicitly obey COMPLY *INVARIANTS*. It is timely therefore, to revisit the COMPLY clause. As noted before, the most general form of COMPLY clause, COMPLY $P(\mathcal{D}xs, xs, t)$, is not suitable for systematic processing, so we will be more specific. The principal idea is that, aside from specific additional forms of constraint given below, the COMPLY clause should be a semi-algebraic predicate in the variables. Cylindrical algebraic decomposition, mentioned earlier, then gives a partition of the variables' state space on which each term of the semi-algebraic predicate is uniformly true, or uniformly false. Any earlier derived solution to DE/DA needs to fall in the true regions only, and in cases where the DE/DA behaviour is polynomial, this can be achieved by an extension of the CAD analysis. However, most solutions to DE/DA will not be polynomial, so suitable polynomial bounds may have to be derived, either in an *ad hoc* manner, or by using a tool like MetiTarski [97, 8], designed for the mechanical proof of inequalities of this kind.

Arbitrary semi-algebraic predicates are still too general. For technical reasons we restrict to semi-algebraic predicates that denote sets which are 'sufficiently open', described as follows.

Definition 13.3. *A semi-algebraic set is sufficiently open iff each connected component is contained in the closure of its interior, and each connected component has full Euclidean dimension, where all topological notions are understood in terms of the relative Euclidean topology.*

For example, if the variables' state space is three dimensional, a 3D ball, including the upper hemisphere of its boundary but excluding the boundary's lower hemisphere, could form a suitable sufficiently open constraint set for the dynamics, and would be described by a straightforward semi-algebraic predicate. But the corresponding semi-algebraic predicate could be extended to include various two dimensional sheets or one dimensional hairs, extruding from the ball, that would be unsuitable as dynamical constraints. The closure of the interior of such an extension would include the original 3D ball and its part boundary, but excise the sheets and hairs. Determining whether semi-algebraic predicates are sufficiently open is computable, since every CAD description consists of a finite number of pieces, each of which is homeomorphic to an open cube of appropriate dimension [29]. We call constraints satisfying the preceding considerations the **CAD constraints of** ***BDA***.

While the preceding time independent formulation will be the focus in this paper, we observe that the formulation could be extended by allowing the parameters of the CAD to vary over time (e.g. a region $x^2(t)+y^2(t) < R_0^2$ could become $x^2(t)+y^2(t) < R^2(t)$). In such a case, we would want the CAD state space partitions at different times in the execution of the pliant event to be homeomorphic in a smoothly time-dependent way, in particular, to prevent change in the global structure of the CAD state space partition as a result of change in the discriminants of its semi-algebraic description due to the time-dependence of the parameters. One way of assuring this is to insist on the provision of a witness to the homeomorphism via a time-dependent strictly positive rescaling of the variables (e.g. the substitutions $x(t) \to x(t)\frac{R(t)}{R_0}$ and $y(t) \to y(t)\frac{R(t)}{R_0}$ rescale the time varying earlier example to its static version, provided $R(t)$ is strictly positive for all relevant t).

We can enlarge the possibilities considered, by permitting the occurrence of further types of constraint in the COMPLY clause. For example, we will include the pliant modalities, discussed in Section 14 below. We call these the **MOD constraints of** ***BDA***.

Additionally, we could add semi-algebraic conditions that constrain variable values at specific times during the execution of the pliant event, thereby further constraining the pliant behaviours defined. Most likely such constraints would be demanded at the pliant event termination time $\mathbb{t}_R$, or at chosen durations relative to the start time, e.g. at $t - \mathbb{t}_L = h$ with $h > 0$ a constant. (We require $h > 0$ in all such cases since $h = 0$ corresponds to an INIT constraint. Also, if a pliant transition is preempted before the time specified in such a constraint has been reached, it is not regarded as an error.) We call these the **TIME constraints of** ***BDA***.

In all of these cases, the aim would be to evaluate the consequences of the constraints by including them in the CAD analysis. Even though derivations of closed form solutions would frequently be impossible, existence of solutions in the algebraic number field can be discerned if the constraints can be reduced to multinomial constraints having rational coefficients, regardless of whether these emerge directly, or are generated by using safe approximations.

Definition 13.4 (Pliant Event Properties). *Pliant event* $PliEv : \{\!\!\{\ iv\ ;\ grd \mid ll \mid BDA \blacksquare DE\ ;\ DA\ \}\!\!\}$, *having name and components as shown, will normally be assumed given as part of the discourse in the sequel. We write* $\circledast$ *(or PliEv-*$\circledast$ *where there is possible ambiguity about the name) to denote that:*

1. *BDA consists of CAD and MOD and TIME constraints; BDA may also contain any typing declarations required for local variables;*
2. *all other functions occurring in the components of PliEv are FPHS functions of all their variables;*
3. *DE is Lipschitz bounded.*

We say the components of PliEv, namely $iv, grd, \mathit{ll}, BDA, DE, DA$*, are trivial iff they are respectively* $\mathsf{true}, \mathsf{true}$*, empty,* true*, empty, empty. Indicating trivial components by a hyphen, we write for example, "Suppose PliEv-*$\circledast$: $\pmb{\{}$ -;-|-| $BDA \blacksquare DE$; DA $\pmb{\}}$*" to hypothesise a pliant event having trivial* iv, grd, ll*, and with all nontrivial components satisfying the* $\circledast$ *property.*

If ll *in PliEv contains no input variables, we say PliEv is input-free.*

13.3. Other Possibilities

The above gives one way of making the input language of pliant events precise, together with a rationale for processing it. But this should not be seen as excluding other possibilities, particularly when these are supported by calculational oracles such as *Mathematica*. For instance, one area which we have neglected in this paper is systems featuring delay ODEs [64, 66, 46], e.g. $\mathcal{D}x(t) = Ax(t-1) + f(t)$. These are becoming increasingly important in the study of cyber-physical systems, since the implied delay in the arrival of the information needed to determine the derivative term $\mathcal{D}x(t)$ due to equipment latency, can have a marked effect on issues such as system stability. Limited families of such systems are amenable to analytic techniques; see [119, 53]. Most realistic examples are approached via numerical simulation. Although delay ODEs are important, we do not consider them further here.

Although we do not consider delay ODEs in this paper, we *do* permit, in pliant events, inhomogeneous terms (i.e. terms that do not involve the state variables, such as $R(t)$ above, or $f_1(t), f_2(t)$ from the earlier examples) to depend on relative time, i.e. the time since the start of the pliant event execution, e.g. $R(t - \mathbb{t}_L), f_1(t - \mathbb{t}_L), f_2(t - \mathbb{t}_L)$. This decouples their behaviour from the start times of their execution instances.

14. Formal Reasoning about Hybrid Event-B: Pliant Event Modalities

The easiest part of the programme announced at the start of Section 13 concerns the pliant modalities [15]. These allow familiar properties of functions and variables to be referred to using familiar names, rather than their usual, but more opaque logical definitions, thus aiding readability. Such properties can occur in two places in Hybrid Event-B machines: in the INVARIANTS of a machine, and in pliant events' COMPLY clauses.

When such a modality occurs in a pliant event's COMPLY clause, its role is specificational, and the relevant property may be assumed during any pliant transition specified by the event. Thus, only the consistency of the property with the rest of the event's specification (if any) is needed for the event's feasibility.

When the modality occurs among the INVARIANTS, it describes a property that must be proved to hold in all runs of the machine. This generates conditions that must hold for (reachable) pliant events and for the mode events that interleave them. One snag that now arises is that many of the modalities considered are not properties of an individual state, but are properties that depend on a (time) neighbourhood of a state in a run, and/or on families of these (leading to the kind of quantifications we are trying to eliminate). Overwhelmingly, the POs of Hybrid Event-B are individual state or event properties, so alone, may be insufficient to establish the modalities. Accordingly, in this section, we focus on the properties of *runs* that are sufficient to establish the truth of the modalities. Since all runs consist of alternating mode and pliant transtions, establishing properties of runs reduces to induction over this alternating structure. In each case, where needed, we follow up the properties of runs with some remarks about what would be required to reduce the properties of interest to properties concerning individual events, or concerning a small number of events. Later, in Section 18, we look at the POs for establishing invariants in pliant transitions, which also contribute to what is needed here.

Given what has been stated, one easy way of dealing with an invariant modality in a pliant event is to demand the same modality in the event's COMPLY clause. Normally however, in this situation, more precisely defined pliant behaviour is needed, and a number of evident sufficient conditions can usually be identified that guarantee the property. Since the same conditions can be used to check consistency in the COMPLY case, we focus on the INVARIANTS in the results below.

The modalities dealt with here are invariably introduced in the context of the properties of a single real variable. In our context, when we say 'variable' we intend not only state variables, but to include the inputs, locals, and outputs of a pliant event, as appropriate. A single real variable's values range over a single copy of $\mathbb{R}$. Therefore the results in this section are stated in those terms too.[10]

The results here can be extended to trajectories of variables whose values range over several real dimensions fairly straightforwardly. In those case where inequalities are needed to frame the required concepts, we assume there is a given GDF η which is used to intepret the inequality relations. In particular lower or upper bounds, where needed, are assumed to be given by values of η.

14.1. The CONTINUOUS Modality and its Relatives

The CONTINUOUS(x) modality declares that a pliant variable's behaviour is absolutely continuous during any run of the system. Since we restrict to *PH* functions calculationally, the 'absolutely' can be taken for granted.

Proposition 14.1 (CONTINUOUS Modality). *Let x be an individual pliant variable. Then CONTINUOUS(x) holds throughout every run if the following hold.*

1. *($\forall$ MoEv $\bullet$ x is not assigned in MoEv), and*
2. *($\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   (a) *CONTINUOUS(x) is a top level conjunct in BDA, or*
   (b) *x is assigned via an ODE in DE, or*
   (c) *x is assigned via DA, and*
      i. *PliEv is input-free, and the RHS of DA is a (vector-valued) PHS function of its arguments, or*
      ii. *PliEv has inputs, and the RHS of DA is a (vector-valued) PHS function of its arguments, but the RHS of the equation for x does not depend on any input, or*
      iii. *PliEv has inputs, and the RHS of DA is a (vector-valued) FPHS function of its arguments, but the RHS of the equation for x does not depend on any input and is continuous at all its internal join points, or*
      iv. *PliEv has inputs, and the RHS of DA is a (vector-valued) FPHS function of its arguments, but the RHS of the equation for x depends only on inputs that may be assumed to be globally continuous, and the RHS is continuous at all its internal join points).*

Proof: Let x be a pliant variable and suppose the stated assumptions hold. Consider a run of the machine. It is right continuous by definition. The *INITIALSATION* endows x with an initial value x_0. Next, a pliant event *PliEv* executes. If CONTINUOUS(x) is a top level conjunct in *BDA*, then x is continuous during the execution of *PliEv*. If x is assigned via *DE*, then x is continuous during the execution of *PliEv* (since any discontinuities in the RHS of the ODE for x are integrated into kinks).

Otherwise, x is assigned via *DA*. Cases i-iii in the assumptions are strengthenings of case iv, so it is sufficient to consider case iv alone. Unlike the *DE* case, any discontinuity in the RHS of a *DA* equation for x is immediately reflected in the value of x, so case iv simply states that no such discontinuity arises. So x is continuous during the execution of *PliEv*.

Upon completion of *PliEv* a mode event *MoEv* executes. By 1, it does not assign x, so x retains its value and is continuous at the execution point of *MoEv*. The remainder of the run consists of alternating

[10]We use the phrase 'individual pliant variable' to emphasise this

pliant and mode event executions, and the preceding arguments apply in each case, giving an inductive proof that CONTINUOUS(x) is true throughout the run. □

Although cases i-iii are subsumed by case iv, their conditions are easier to check statically; so it is worth mentioning them. Moreover, further similar variations could be imagined. A technicality arises with case iv concerning inputs, since Hybrid Event-B allows different events to have different input spaces. The use of the phrase 'globally continuous' is intended to signify that despite this, continuity across event boundaries must be ensured. Simple restrictions, such as only allowing a single input space for all pliant events and then checking all potentially enabled pliant/mode/pliant boundary cases, would be sufficient for this, although more complex cases can easily be imagined. We do not pursue this in full generality.

It is clear that where the checks involved in the above do not just concern a single event, they can be decomposed into a series of checks, each involving a set of events of small, predetermined cardinality. As such, a verification tool would encounter no additional difficulty in generating the relevant verification conditions from a machine definition, compared with doing the same where only a single event was needed.

We consider the n-DIFFERENTIABLE(x) modality. The n-DIFFERENTIABLE(x) modality extrapolates the continuous modality, and declares that a pliant variable's behaviour is n times differentiable during any run of the system. The restriction to *PHS* functions calculationally, means that provided mode events do not assign x, it is sufficient to check that at all join points between two *PHS* functions, either the requisite degree of differentiability exists, or that the two functions' Taylor series exist and agree on their first several terms. We treat this modality more briefly, in particular, omitting the three special cases of case (c) below.

Proposition 14.2 (n-DIFFERENTIABLE Modality). *Let x be an individual pliant variable. Then n-DIFFERENTIABLE(x) holds throughout every run if the following hold.*

1. *$(\forall$ MoEv $\bullet$ x is not assigned in MoEv$)$, and*
2. *$(\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   - (a) *n-DIFFERENTIABLE(x) is a top level conjunct in BDA, or*
   - (b) *x is assigned via an ODE in DE, whose RHS is a FPHS function of its arguments which is $n-1$ times differentiable at all its internal join points, or*
   - (c) *x is assigned via DA, and*
     - i. *PliEv has inputs, and the RHS of DA is a (vector-valued) FPHS function of its arguments, but the RHS of the equation for x depends only on inputs that may be assumed to be globally n times differentiable, and the RHS is n times differentiable at all its internal join points$)$, and*
3. *$(\forall$ $PliEv_1, MoEv, PliEv_2$ $\bullet$ whenever a transition of $PliEv_1$ enables MoEv and the after-state of MoEv enables $PliEv_2$, then the Taylor series of the transitions of $PliEv_1$ and $PliEv_2$ exist at the common point, and agree on their first $n+1$ terms$)$.*

Proof Sketch: The overall structure is as for Proposition 14.1, so we just point out the differences. At join points in the RHS we need n times differentiability for the *DA* case, and $n-1$ times differentiability for the *DE* case, to ensure n times differentiability of x. Additionally, we need n times differentiability for the join points between abutting pliant event executions (assuming the intervening mode event does not modify the variable). For that, a Taylor series common to $n+1$ terms (i.e. up to the n'th derivatives) is sufficient. □

We see in this Proposition that the 'global continuity' of the previous Proposition has to be more explictly catered for in case 3. Aside from that, the analogous remarks made there are adequate here too.

Note also that clause 3 of Proposition 14.2 goes beyond event-by-event verification, and strays a little into what would usually be regarded as model checking territory, in that sequences of successively enabled events are considered. Such possibilities could obviously be contemplated more widely.

We consider briefly the $\mathsf{CONST}(x)$ modality. The $\mathsf{CONST}(x)$ modality (and $\mathsf{CONST}(x,E)$), which declare the constancy of x (and its value as E) is only useful in the COMPLY context, since outside that context, we can declare a constant, or a mode variable and an invariant equating its value to its initial value, etc. Thus the role of these modalities is specificational, and does not require further discussion here.

14.2. The PLENV Modalities

The PLENV modalities constrain their argument to remain within a PL*iant* ENV*elope* specified by a lower bound function and/or an upper bound function. Thus $\mathsf{PLENVL}(x,lb)$ ensures $lb \leq x$, while $\mathsf{PLENVU}(x,ub)$ ensures $x \leq ub$. Furthermore, $\mathsf{PLENV}(x,lb,ub)$ is defined to be the conjunction of $\mathsf{PLENVL}(x,lb)$ and $\mathsf{PLENVU}(x,ub)$. For the multidimensional generalisation, here and for the remainder of the section, we refer to remarks above.

Proposition 14.3 (PLENV Modalities). *Let x be an individual pliant variable. Then $\mathsf{PLENVL}(x,lb)$ holds throughout every run if the following hold.*

1. *($\forall MoEv \bullet x - lb \geq 0$ in the after-state of every MoEv transition), and*
2. *($\forall PliEv \bullet PliEv$-$\circledast$, and*
   - (a) *$\mathsf{PLENVL}(x,lb)$ is a top level conjunct in BDA, or*
   - (b) i. *$x - lb \geq 0$ at every stationary point of $x - lb$ in every transition of PliEv, and*
     - ii. *$x - lb \geq 0$ at the left and right limits of every internal join point of $x - lb$ in every transition of PliEv, and*
     - iii. *$x - lb \geq 0$ at the left limit or final value at $\mathbb{t}_R$ in every transition of PliEv).*

Also, $\mathsf{PLENVU}(x,ub)$ holds throughout every run if the following hold.

1. *($\forall MoEv \bullet ub - x \geq 0$ in the after-state of every MoEv transition), and*
2. *($\forall PliEv \bullet PliEv$-$\circledast$, and*
   - (a) *$\mathsf{PLENVU}(x,ub)$ is a top level conjunct in BDA, or*
   - (b) i. *$ub - x \geq 0$ at every stationary point of $ub - x$ in every transition of PliEv, and*
     - ii. *$ub - x \geq 0$ at the left and right limits of every internal join point of $ub - x$ in every transition of PliEv, and*
     - iii. *$ub - x \geq 0$ at the left limit or final value at $\mathbb{t}_R$ in every transition of PliEv).*

Moreover, $\mathsf{PLENV}(x,lb,ub)$ is true throughout every run if both of the preceding hold.

Proof: For $\mathsf{PLENVL}(x,lb)$, let x be a pliant variable and suppose the stated assumptions hold. Consider a run of the machine. The *INITIALSATION*, considered as a mode event executed at t_0, endows x with an initial value $x_0(t_0) \geq lb(t_0)$. Next, a pliant event *PliEv* executes, and its initial value for u satisfies $x \geq lb$ since these are the values at t_0. If $\mathsf{PLENVL}(x,lb)$ is a top level conjunct in *BDA*, we are done. Otherwise, since x and lb are both assumed to be *FPHS* functions, the extrema of $x - lb$ in each *PH* piece are at its stationary points and boundary points. These can be located by differentiating $x - lb$ and checking the join points. Assumptions (b).i,ii,iii confirm that the whole *PliEv* execution respects $\mathsf{PLENVL}(x,lb)$.

Upon completion of *PliEv* a mode event *MoEv* executes. Its before-state, reached by the preceding *PliEv*, respects $\mathsf{PLENVL}(x,lb)$, as does its after-state by 1. After this, the arguments repeat in the usual way. For $\mathsf{PLENVU}(x,ub)$ the argument is analogous, and for $\mathsf{PLENV}(x,lb,ub)$ we need both $\mathsf{PLENVL}(x,lb)$ and $\mathsf{PLENVU}(x,ub)$. □

When lb/ub is/are constants E_L/E_U, we can optimise the preceding results, yielding modalities $\mathsf{LBND}(x,E_L)$, $\mathsf{UBND}(x,E_U)$ and $\mathsf{BND}(x,E_L,E_U)$. These constant constraint versions are typically very important practically.

Proposition 14.4 (BND Modalities). *Let x be an individual pliant variable. Then $\mathsf{LBND}(x,E_L)$ holds throughout every run if the following hold.*

1. *($\forall$ MoEv $\bullet$ $x \geq E_L$ in the after-state of every MoEv transition), and*
2. *($\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   (a) *$\mathsf{LBND}(x,E_L)$ is a top level conjunct in BDA, or*
   (b) i. *$x \geq E_L$ at every stationary point of x in every transition of PliEv, and*
       ii. *$x \geq E_L$ at the left and right limits of every internal join point of x in every transition of PliEv, and*
       iii. *$x \geq E_L$ at the left limit or final value at $\mathbb{t}_R$ in every transition of PliEv).*

Also, $\mathsf{UBND}(x,E_U)$ holds throughout every run if the following hold.

1. *($\forall$ MoEv $\bullet$ $x \leq E_U$ in the after-state of every MoEv transition), and*
2. *($\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   (a) *$\mathsf{UBND}(x,E_U)$ is a top level conjunct in BDA, or*
   (b) i. *$x \leq E_U$ at every stationary point of x in every transition of PliEv, and*
       ii. *$x \leq E_U$ at the left and right limits of every internal join point of x in every transition of PliEv, and*
       iii. *$x \leq E_U$ at the left limit or final value at $\mathbb{t}_R$ in every transition of PliEv).*

Moreover, $\mathsf{BND}(x,E_L,E_U)$ is true throughout every run if both of the preceding hold.

Proof: A straightforward adaptation of Proposition 14.3 to the case where the bounds, E_L and E_U, are constant. $\square$

We see that in all the cases of this subsection, the checks needed can be carried out on a per-event basis. This is no surprise since static bounds occur very often as invariants.

14.3. The MONINC and MONDEC Modalities

The $\mathsf{MONINC}(x)$ and $\mathsf{MONDEC}(x)$ modalities constrain their argument to be monotonically increasing or decreasing respectively. There is an evident connection with $\mathsf{LBND}(\mathcal{D}x,0)$ and $\mathsf{UBND}(\mathcal{D}x,0)$ at points where the derivative exists. The following is rather straightforward.

Proposition 14.5 (MONINC and MONDEC Modalities). *Let x be an individual pliant variable. Then $\mathsf{MONINC}(x)$ holds throughout every run if the following hold.*

1. *($\forall$ MoEv $\bullet$ the after-value of x is no less than its before-value in every MoEv transition), and*
2. *($\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   (a) *$\mathsf{MONINC}(x)$ is a top level conjunct in BDA, or*
   (b) i. *$\mathcal{D}x \geq 0$ at every non-join point of x in every transition of PliEv, and*
       ii. *the after-value of x is no less than its before-value at the left and right limits of every internal join point of x in every transition of PliEv).*

Also, $\mathsf{MONDEC}(x)$ holds throughout every run if the following hold.

1. *($\forall$ MoEv $\bullet$ the after-value of x is no greater than its before-value in every MoEv transition), and*
2. *($\forall$ PliEv $\bullet$ PliEv-$\circledast$, and*
   (a) *$\mathsf{MONDEC}(x)$ is a top level conjunct in BDA, or*

(b) i. $\mathcal{D}x \leq 0$ *at every non-join point of x in every transition of PliEv, and*
   ii. *the after-value of x is no greater than its before-value at the left and right limits of every internal join point of x in every transition of PliEv).*

Proof Sketch: A function is monotonically increasing at a discontinuity if its after-value is no less than its before-value. It is monotonically increasing in a neighbourhood where its derivative exists if that derivative is nonnegative. On this basis, since the behaviours of variables are all piecewise holomorphic functions, the conditions stated are sufficient to construct an inductive proof of monotonic increase of x throughout a run, in the usual way. The monotonically decreasing case is similar. □

14.4. The CVEX and CCAVE Modalities

The $\mathsf{CVEX}(x)$ and $\mathsf{CCAVE}(x)$ modalities constrain their argument to be convex and concave respectively. There is an evident connection with $\mathsf{LBND}(\mathcal{D}^2x, 0)$ and $\mathsf{UBND}(\mathcal{D}^2x, 0)$ at points where the second derivative exists. The following is rather straightforward.

Proposition 14.6 (CVEX and CCAVE Modalities). *Let x be an individual pliant variable. Then CVEX(x) holds throughout every run if the following hold.*

1. $(\forall$ *MoEv* $\bullet$ *x is not assigned in MoEv), and*
2. $(\forall$ *PliEv* $\bullet$ *PliEv-*$\circledast$*, and*
   (a) *CVEX(x) is a top level conjunct in BDA, or*
   (b) i. $\mathcal{D}^2x \geq 0$ *at every non-join point of x in every transition of PliEv, and*
      ii. *the after-value of* $\mathcal{D}x$ *is no less than its before-value at the left and right limits of every internal join point of x in every transition of PliEv), and*
3. $(\forall$ *PliEv*$_1$*, MoEv, PliEv*$_2$ $\bullet$ *whenever a transition of PliEv*$_1$ *enables MoEv and the after-state of MoEv enables PliEv*$_2$*, then the after-value of* $\mathcal{D}x$ *is no less than its before-value at the MoEv transition).*

Also, CCAVE(x) holds throughout every run if the following hold.

1. $(\forall$ *MoEv* $\bullet$ *x is not assigned in MoEv), and*
2. $(\forall$ *PliEv* $\bullet$ *PliEv-*$\circledast$*, and*
   (a) *CCAVE(x) is a top level conjunct in BDA, or*
   (b) i. $\mathcal{D}^2x \leq 0$ *at every non-join point of x in every transition of PliEv, and*
      ii. *the after-value of* $\mathcal{D}x$ *is no greater than its before-value at the left and right limits of every internal join point of x in every transition of PliEv).*
3. $(\forall$ *PliEv*$_1$*, MoEv, PliEv*$_2$ $\bullet$ *whenever a transition of PliEv*$_1$ *enables MoEv and the after-state of MoEv enables PliEv*$_2$*, then the after-value of* $\mathcal{D}x$ *is no greater than its before-value at the MoEv transition).*

Proof Sketch: If a function is convex, it must be continuous (except, perhaps, at the extremes of its domain, when the domain is a closed interval). It is convex at a discontinuity of its derivative if the derivative's after-value is no less than its before-value. It is convex in a neighbourhood where its second derivative exists if that second derivative is nonnegative. On this basis, since the behaviours of variables are all piecewise holomorphic functions, the conditions stated are sufficient to construct an inductive proof of convexity of x throughout a run, in the usual way. The concave case is similar. □

In this and the preceding subsection, it is clear that we have replaced conditions involving two or three time points of a function by conditions involving derivatives at a single time point, relying on piecewise holomorphicity for justification. This changes (and arguably simplifies) the task faced by an automated verifier.

Part 6 — Pliant POs: Event Feasibility

15. Formal Reasoning about Hybrid Event-B: Pliant Event POs

The POs that deal exclusively with pliant events are in Figs. 6 and 8, and cover both individual machine correctness, and the correctness of pliant event refinement. They are PliEv/FIS, PliEv/INV, PliEv/FISR, PliEv/INVR, PliEv/FISRW, PliEv/INVRW. Pliant event guard strengthening and relative deadlock freeness, PliEv/GRDR and PliEv/RelDLF, have been discussed earlier and do not need further consideration here. The same applies for the wellformedness POs MoPli/WFor and PliMo/WFor.

The main issue around the pliant POs comes from the expressiveness of pliant events. To increase their expressivity, several convenient clauses for specifying pliant behaviour are permitted as described in Section 2, and various possibilities for them that lead to tractable decision procedures are discussed in Section 13. This entails a holistic approach for pliant event verification, as the possible interactions between the various clauses have to be taken into account when proving correctness. This was already evident to a degree in the pliant modality propositions of Section 14.

In most practical cases, only a portion of full pliant event expressivity is used for any particular event, leading to economies in the verification task. This opens the way for exploiting the generic properties that are assumed about pliant events and using these to reduce pliant event verification to simple syntactic checks to the greatest degree possible. This is the aim of the next few sections. We take the POs one by one, and build up from structurally simple cases. We use the same notations as in Figs. 6-8 for variables etc.

16. Pliant Event Feasibility, PliEv/FIS

The feasibility PO for pliant events, reproduced from Fig. 6 is:

PliEv/FIS:

$$\begin{aligned}&I(u(\mathbb{t}_{\mathrm{L}})) \land iv_{PliEvA}(u(\mathbb{t}_{\mathrm{L}})) \land grd_{PliEvA}(u(\mathbb{t}_{\mathrm{L}}))\\ &\quad \Rightarrow (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet [\ (\mathbb{t}_{\mathrm{R}} - \mathbb{t}_{\mathrm{L}} \geq \delta_{\mathsf{Zeno}PliEvA}) \land]\ (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet\\ &\qquad\quad BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \land SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t))))\end{aligned} \tag{21}$$

Feasibility asserts that the event in question is viable, in that its intial conditions do not rule out a pliant event transition *a priori*. As noted before, ensuring statically that the Zeno bound $\mathbb{t}_{\mathrm{R}} - \mathbb{t}_{\mathrm{L}} \geq \delta_{\mathsf{Zeno}PliEvA}$ holds, is usually impossible without determining the entire system dynamics, so we will not attempt to do so. Also, $\mathbb{t}_{\mathrm{R}}$ is existentially quantified in (21), but without the clauses that fix its value as *the* termination time (of any particular execution) of the event, so detailed termination considerations are not needed.

In contrast to mode events, where feasibility depends on the before-values of the variables and inputs alone, pliant transitions extend over time, and the *BDApred* predicate may impose global constraints, which raises the question of whether feasibility should be interpreted globally or locally.

A couple of examples illustrate the point: (1) A behaviour $u(t) = \sin(t)$ for $t \geq 0$ satisfies $\mathsf{CCAVE}(u)$ if t does not exceed π, but fails to satisfy it if t exceeds π. If other constraints in a pliant event force this behaviour for u and its *BDApred* stipulates $\mathsf{CCAVE}(u)$, is the event feasible if its duration, starting at $t = 0$, cannot be predicted in advance (for instance its duration may depend on inputs from the environment)? (2) The same behaviour for u may satisfy $u(\mathbb{t}_{\mathrm{R}}) \in [\frac{1}{2} \dots \frac{3}{4}]$ if $\mathbb{t}_{\mathrm{R}}$ takes a value near $\pi/4$, but not if $\mathbb{t}_{\mathrm{R}}$ takes a value near π. Is the event feasible in these cases? In this paper we interpret feasibility *locally*, which is to say that a pliant event is feasible iff there is a neighbourhood of the initial time $\mathbb{t}_{\mathrm{L}}$, in which all the constraints in the event which apply within that neighbourhood are true. According to this interpretation, all the examples just discussed are feasible, since we 'need not look too far into the future' to check feasibility.[11]

Although feasibility is interpreted locally, it still asserts a non-empty left-closed right-open duration for the event execution, and right continuous behaviour overall for the execution. For duration, this entails the prevention of *immediate termination*, i.e. termination at $\mathbb{t}_{\mathrm{L}}$. However, we need not be concerned with this in PliEv/FIS since the PO MoPli/WFor covers it. For right continuous behaviour, while the presumption of *FPH* behaviour for solutions covers most of what is required, we need to check that consecutive pieces of behaviour join appropriately, and, for a pliant event execution, at $\mathbb{t}_{\mathrm{L}}$ in particular.

This precludes $iv \wedge grd$ being more generous than the initial values implicit in a nontrivial *DA* or a nontrivial *BDA*, since that might be interpreted as implying a possible jump in the value of some variable(s) at the start of some execution of the pliant event (that has not been achieved by the immediately preceding mode transition).[12] Therefore, for a pliant event $\mathit{PliEv} : \pmb{\{}\ iv\ ;\ grd \mid ll \mid \mathit{BDA} \blacksquare \mathit{DE}\ ;\ \mathit{DA}\ \pmb{\}}$, we capture the initial state values permitted by $\mathit{BDA} \wedge \mathit{DA}$ thus:

$$iv_{B/DA} \equiv \{u \mid (\ \exists ll\ \bullet \overleftarrow{\mathit{BDA}(u,ll,\mathbb{t}_{\mathrm{L}})} \wedge \overleftarrow{\mathit{DA}(u,ll,\mathbb{t}_{\mathrm{L}})}\ )\} \tag{22}$$

If $iv \wedge grd$ does not entail $iv_{B/DA}$ then there are two ways that a mechanised verification system can avoid the implied jump mentioned above. One way is to conjoin $iv_{B/DA}$ to $iv \wedge grd$ during verification. The guard, thus strengthened, may result in a failure of the MoPli/WFor PO, if the disjunction of pliant event guards in the conclusion of the PO becomes too strong.

An alternative way is to demand that $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ is proved during verification. If this holds, then it is clear that at runtime, any pliant event that is enabled according to the user's definition can execute in a right continuous manner. This is a cleaner way of handling the issue, which is therefore adopted below.

Proposition 16.1 (PliEv/FIS-BDA). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. *PliEv-$\circledast : \pmb{\{}\ iv\ ;\ grd \mid ll \mid \mathit{BDA} \blacksquare$-;-$\pmb{\}}$, and*
2. *$(\forall x \bullet$ I and grd and $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$, and*
3. *$(\forall x \bullet$ the CAD constraints, if any, of BDA hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$, and*
4. *$(\forall x \bullet$ the MOD constraints, if any, of BDA hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$, and additionally:*
   (a) *if $\mathsf{PLENVL}(x,lb)$ and/or $\mathsf{PLENVU}(x,ub)$ (or $\mathsf{PLENV}(x,lb,ub)$) is/are present in the MOD constraints, then $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})}$ and/or $\overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$ is/are in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$, and if both are present, then $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})} < \overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$, and $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})} \le \overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ and/or $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})} \le \overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$, as applicable, and*
   (b) *as for case* (a) *for $\mathsf{LBND}(x,E_L)$, $\mathsf{UBND}(x,E_U)$, $\mathsf{BND}(x,E_L,E_U)$, and*
   (c) *if $\mathsf{MONINC}(x)$ or $\mathsf{MONDEC}(x)$ is present in the MOD constraints (both together are not permitted), then $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$, and*
   (d) *if $\mathsf{CVEX}(x)$ or $\mathsf{CCAVE}(x)$ is present in the MOD constraints (both together are not permitted), then $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$).*

[11]It is worth noting that in the correctness proofs of PaperI and PaperII (i.e. proofs that when the POs hold, an execution does not encounter a 'runtime error'), feasibility was implicitly assumed to hold *globally*, i.e. for as long as needed. Little is lost in the distinction between these approaches, since loss of feasibility is defined as finite termination, a correct outcome (see PaperI). In any case, in straightforward cases, local feasibility can be extended to global feasibility, though more complex cases will require a full solution of the pliant behaviour.

[12]Note that *DE* does not create any consistency issue since the general theory of Lipschitz bounded ODEs guarantees a right continuous solution from any initial value.

Proof: Let *PliEv* be as in 1 and suppose that 2 holds. Suppose there are only CAD constraints (or none), satisfying 3, in *BDA*. Then, since all functions appearing in *PliEv* are *FPHS* and the CAD constraints describe a region contained in the closure of its nonempty interior, there will be a *PH* function for any variable x, which remains inside the CAD constraints within some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$. Such a function will discharge the PliEv/FIS PO locally.

If any of the MOD constraints mentioned in 4 are present, then we have to ensure that we are appropriately 'away from any boundary point' of the CAD constraints to ensure that there will be a *PH* function for x that satisfies them in a right-open interval starting at $\mathbb{t}_{\mathrm{L}}$ based on the $\mathbb{t}_{\mathrm{L}}$ values alone — hence the particular conditions in 4.(a)-4.(d). Since any *PH* function is smooth, any $\mathsf{CONTINUOUS}(x)$, $\mathsf{CONST}(x)$, $\mathsf{CONST}(x,E)$ or n-$\mathsf{DIFFERENTIABLE}(x)$ modality is automatically satisfied locally.

Each of these MOD constraints generates a candidate right-open interval starting at $\mathbb{t}_{\mathrm{L}}$ inside which they are satisfied. Since there are only a finite number of them due to the finite syntax of any *PliEv*, the intersection of all of them gives a right-open interval starting at $\mathbb{t}_{\mathrm{L}}$ which is sufficient to discharge the PliEv/FIS PO locally.

Finally, if there are TIME constraints then since each refers to a time strictly greater than $\mathbb{t}_{\mathrm{L}}$, we merely need to ensure the duration of the interval just discussed is curtailed to before the earliest relevant moment that might force infeasibility, to ensure that none of them are required to hold in order to discharge PliEv/FIS locally. □

We note that in avoiding certain 'boundary situations' above we have derived results which are suboptimal, in that 'boundary versions' could be derived for some of them, provided appropriate additional conditions were imposed. However, such additional conditions would typically have to extend beyond just mentioning the values at $\mathbb{t}_{\mathrm{L}}$, so we did not pursue these in the interests of simplicity (and also because such cases are of little engineering interest; see Section 25).

Proposition 16.2 (PliEv/FIS-DE). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. $PliEv$-$\circledast : \{\ iv\ ;\ grd \mid ll\ |\text{-}\blacksquare\ DE \equiv \mathcal{D}x = \phi\ ;\text{-}\}$*, and*
2. $(\forall x \bullet$ *I and grd and iv hold for* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$.

Proof: Let *PliEv* be as in 1 and suppose that 2 holds. Since *BDA* is vacuously true, in order to satisfy the PO, we just need to check that *DE* has a solution SOL_{PliEv} in some non-empty right-open interval starting at $\mathbb{t}_{\mathrm{L}}$. Since ϕ, the RHS of *DE*, is Lipschitz continuous by assumption, and is given by a(n in general vector-valued) *FPHS* function of its variables making it trivally measurable in time, standard theory [130] says that there is a *FPHS* solution in some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$, as required. □

Proposition 16.3 (PliEv/FIS-DA). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. $PliEv$-$\circledast : \{\ iv\ ;\ grd \mid ll\ |\text{-}\blacksquare\text{-};\ DA\ \}$*, and*
2. $(\forall x \bullet$ *I and grd and* $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ *hold for* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$*, and*
3. *the EV graph is acyclic, and its leaves refer to known FPHS functions.*

Proof: Let *PliEv* be as in 1 and suppose that 2 and 3 hold. Since there are no DE nodes in the EV graph, the EV graph is identical to the graph minor formed by contracting the DEV-edges, which is thus acyclic, therefore admissible. Proposition 13.2 thus guarantees that the obvious evaluation strategy, of solving by substitution from known information at the leaves, towards the roots, will be successful. Then, since the RHS of all the DA assignments are *FPHS* functions, the outcome ensures that there is a solution to the DA system, in some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$, discharging PliEv/FIS PO locally. □

Proposition 16.4 (PliEv/FIS-BDA/DE). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. *PliEv-$\circledast$:* **{** *iv* **;** *grd* | *ll* | *BDA* $\blacksquare$ *DE* $\equiv \mathcal{D}x = \phi$ **;-}**, *and*
2. *($\forall x \bullet$ I and grd and $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$), and*
3. (a) *($\forall x \bullet$ $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of the region specified by the CAD constraints, if any, of BDA), or*
   (b) *($\forall x \bullet$ $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the boundary of the region specified by the CAD constraints of BDA, and the vector field $\phi(x)$ points into the interior of the CAD constraints region), or*
   (c) *($\forall x \bullet$ $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the boundary of the region specified by the CAD constraints of BDA, and, letting $d(p)$ denote the Euclidean distance from a point p to the boundary of the region specified by the CAD constraints (positive for the interior, negative for the exterior), then $\mathrm{D}(d; \phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})) > 0$, where $\mathrm{D}(f; v)$ is the directional derivative of f in the direction v), or*
   (d) *($\forall x \bullet$ $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the boundary of the region specified by the CAD constraints of BDA, and, assuming the CAD constraints satisfy: (i) they are in disjunctive normal form, CAD $\equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} Q_{i,j}$, (ii) each $Q_{i,j}$ is of the form $Q_{i,j} \equiv P_{i,j} \diamond 0$, where $\diamond \in \{=, >\}$; then, letting $\hat{\imath}$ indicate a disjunct such that $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ satisfies $\bigwedge_{j=1}^{m_{\hat{\imath}}} Q_{\hat{\imath},j}$, and letting $\hat{\jmath}$ range over all $j \in \{1 \ldots m_{\hat{\imath}}\}$ such that $Q_{\hat{\imath},\hat{\jmath}}$ is of the form $P_{\hat{\imath},\hat{\jmath}} > 0$ and it holds that $P_{\hat{\imath},\hat{\jmath}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}) = 0$, we have, for all $\hat{\jmath}$, that $\langle \phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}), \nabla P_{\hat{\imath},\hat{\jmath}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}) \rangle > 0$, where $\langle ., . \rangle$ is the Euclidean inner product),*

   and
4. *($\forall x \bullet$ the MOD constraints, if any, of BDA hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$, and additionally:*
   (a) *if PLENVL(x, lb) and/or PLENVU(x, ub) (or PLENV(x, lb, ub)) is/are present in the MOD constraints, then $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})}$ and/or $\overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$ is/are in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$, and if both are present, then $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})} < \overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$, and $\overleftarrow{lb(\mathbb{t}_{\mathrm{L}})} < \overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ and/or $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})} < \overleftarrow{ub(\mathbb{t}_{\mathrm{L}})}$, as applicable, and*
   (b) *as for case* (a) *for LBND(x, E_L), UBND(x, E_U), BND(x, E_L, E_U), and*
   (c) *if MONINC(x) or MONDEC(x) is present in the MOD constraints (both together are not permitted), then $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$, and*
   (d) *if CVEX(x) or CCAVE(x) is present in the MOD constraints (both together are not permitted), then $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of the CAD constraints, if any, at $\mathbb{t}_{\mathrm{L}}$).*

Proof: Let *PliEv* be as in 1 and suppose that 2 holds. Then the arguments used in Proposition 16.2 show that there is a solution SOL_{PliEv} in some non-empty right-open interval starting at $\mathbb{t}_{\mathrm{L}}$. We need to confirm that some initial portion of SOL_{PliEv} lies within the constraints defined by *BDA*.

If $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the interior of any CAD constraints in *BDA*, then there will be an open ball centred on $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ also in the interior of the CAD constraints. Inside this ball, SOL_{PliEv} will satisfy the CAD constraints. This covers case 3.(a).

For cases 3.(b)-3.(d), $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the boundary of the CAD constraints in *BDA*, so to ensure that SOL_{PliEv} has an initial portion that lies within the CAD constraints, it is sufficient that $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ points into the interior of the CAD constraints. (If not, if there were points on SOL_{PliEv}, arbitrarily close to $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ but outside the CAD constraints, then $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ would not point into the constraints' interior.[13]) Case 3.(b) states this directly. In case 3.(c), since the CAD constraints specify a region of full dimension,

[13]We avoid engagement with the weaker but more subtle conditions that would allow $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ to be tangential to the CAD constraints while yet keeping the DE flow contained within them [130, 41, 42].

points in its interior will lie at a positive distance from the boundary. Thus, provided the directional derivative of the distance function at $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ in the direction $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ is positive, $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ will point strictly into the interior of the CAD constraints, reducing to case 3.(b).

Case 3.(d) exploits the assumed CAD structure to derive more easily computable conditions. Since the CAD constraints are written in a first order language, the constraints can be manipulated into DNF, and with all constraints written as $P = 0$ or $P > 0$. Since $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the region specified by the CAD constraints, there is a disjunct, labelled $\hat{\imath}$ say, such that all its $P_{\hat{\imath},j} \diamond 0$ constraints are satisfied. Moreover, since the CAD constraints specify a region with full dimension contained in the closure of its interior and $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ is in the boundary of this region, $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ borders one or more open sets specified via one or more $P_{\hat{\imath},\hat{\jmath}} > 0$ constraints. The fact that P is multinomial ensures that the gradient ∇P exists, and the form $P > 0$ ensures that the gradient points strictly into the interior of the region specified by the relevant $P_{\hat{\imath},\hat{\jmath}} > 0$. So provided the inner product $\langle \phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}), \nabla P_{\hat{\imath},\hat{\jmath}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}) \rangle$ is strictly positive, $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ will point into the interior of the $P_{\hat{\imath},\hat{\jmath}} > 0$ region. Therefore, if this holds for all relevant $\hat{\jmath}$, then $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ will point into the intersection of these regions, i.e. into the interior of the CAD constraints, as required.[14]

Turning to the MOD constraints, we can largely rerun the arguments of Proposition 16.1. The principal difference is the need to ensure that there is an open ball centred on $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ in which all required constraints hold, so that some initial portion of SOL_{PliEv} lies within the constraints, regardless of the direction in which $\phi(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ points. The only tangible effect of this is the strengthening of the nonstrict inequalities in 4.(a) of Proposition 16.1 to strict inequalities in 4.(a) here. Otherwise the arguments are appropriate.

We deal with any TIME constraints as in Proposition 16.1; and again, we take the intersection of any intervals generated under separate cases above, to get an interval satisfying all relevant conditions for discharging the PliEv/FIS PO locally. $\square$

As previously, we have avoided various 'boundary situations' in the proposition above, most prominently the ones in point 4.(a). The techniques used in 3.(d) could be used to enlarge the remit of the proposition, if desired.

Proposition 16.5 (PliEv/FIS-BDA/DA). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. *$PliEv$-$\circledast$: $\{$ iv ; grd | ll | BDA ■-; DA $\}$, and*
2. *$(\forall x \bullet I$ and grd and $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ hold for $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$, and*
3. *The EV graph is acyclic, and its leaves refer to known FPHS functions, and*
4. *condition 3.(a) in Proposition 16.4 holds, and*
5. *condition 4 in Proposition 16.4 holds.*

Proof: Let *PliEv* be as in 1 and suppose that 2 and 3 hold. Then we can replay the proof of Proposition 16.3 to deduce that there is a solution, in some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$, to the DA system of *PliEv*. We need to check that we have enough conditions on *BDA* to ensure that the solution says within the *BDA* constraints for some nontrivial initial subinterval. But here, we can replay the relevant parts of the proof of Proposition 16.4, since there, we showed that conditions 3 and 4 were sufficient to ensure that

[14]The applicability of case 3.(d) can depend on the representation of the CAD region. Thus, consider the ODE system $\mathcal{D}u_1 = 1, \mathcal{D}u_2 = 0$ with $(u_1(\mathbb{t}_{\mathrm{L}}), u_2(\mathbb{t}_{\mathrm{L}})) = (0,0)$ in the region $(0 \leq u_1 < 1) \wedge (-1 < u_2 < 1)$ with the region represented in the obvious way as a single rectangle plus appropriate boundary line. Then 4.(d) succeeds. But if the (u_1, u_2) region was, perversely, described as $(u_1 > 0 \wedge -u_1 + 1 > 0 \wedge u_2 > 0 \wedge -u_2 + 1 > 0) \vee (u_1 > 0 \wedge -u_1 + 1 > 0 \wedge u_2 + 1 > 0 \wedge -u_2 > 0) \vee (u_1 = 0 \wedge u_2 + 1 > 0 \wedge -u_2 + 1 > 0) \vee (u_1 > 0 \wedge -u_1 + 1 > 0 \wedge u_2 = 0)$; then 3.(d) would fail.

a suitable function given in advance —by a DE system there, by a DA system here, and in contrast to the situation in Proposition 16.1 where the function could be chosen while considering the *BDA* constraints themselves— remained within the *BDA* constraints to the extent needed. We are done. □

Proposition 16.6 (PliEv/FIS-DE/DA). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. $PliEv\text{-}\circledast : \{\ iv\ ;\ grd \mid ll \mid\text{-}\blacksquare\ DE\ ;\ DA\ \}$*, and*
2. $(\forall x \bullet I$ *and* grd *and* $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ *hold for* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$*, and*
3. *the EV graph is admissible, and any leaves of the EV graph refer to known FPHS functions, and*
4. *condition 3.(a) in Proposition 16.4 holds, and*
5. *condition 4 in Proposition 16.4 holds.*

Proof: This is very much like Proposition 16.5. The main difference is that we need to consider a general EV graph with both DE and DA nodes. Admissibility then guarantees that the DE/DA system is solvable in principle in some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$, by Proposition 13.2. □

Proposition 16.7 (PliEv/FIS-BDA/DE/DA). *Letting x stand for any of $u, i?, l, o!$ as appropriate, a pliant event PliEv satisfies PO PliEv/FIS if:*

1. $PliEv\text{-}\circledast : \{\ iv\ ;\ grd \mid ll \mid BDA\ \blacksquare\ DE\ ;\ DA\ \}$*, and*
2. $(\forall x \bullet I$ *and* grd *and* $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$ *hold for* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$*, and*
3. *the EV graph is admissible, and any leaves of the EV graph refer to known FPHS functions, and*
4. *condition 3 in Proposition 16.4 holds, and*
5. *condition 4 in Proposition 16.4 holds.*

Proof: This is very much like Proposition 16.5 and Proposition 16.6 too. The argument splits into two. The first part deals with the existence of a solution to the DE+DA system of *PliEv* in some right-open interval starting at $\mathbb{t}_{\mathrm{L}}$, dealt with as in Proposition 16.6. Then the second part deals with the satisfaction of the *BDA* constraints in some nontrivial initial subinterval. But this is exactly as in Propositions 16.4 and 16.5. We are done. □

Part 7 — Pliant POs: Machine Invariants

17. Formal Reasoning about Hybrid Event-B: Machine Invariants

The interplay of mode and pliant variables and transitions permitted by a formalism like Hybrid Event-B permits the writing of potentially quite exotic invariants. However, engineering concerns are typically more focused on the simpler cases. This also enhances the possibilities for automating the reasoning. Regarding the trajectories of a pliant variable, we focus on what we call *proto-invariants*. These are not required to be invariant themselves, but to constitute basic units regarding pliant variables, that, in combination with properties concerning mode variables, make up machine and refinement invariants. The properties assumed for the valuations of pliant variables enable the derivation of some generic results about their trajectories, and checking such properties might well be entrusted to external reasoners, using the EXTERN rule of Fig. 10. The remaining reasoning would then be delegated to more conventional reasoners, SMT-style. Below, when discussing pliant event invariant preservation, we focus on the proto-invariants, since their constituents are the only ones that can change during pliant transitions.

We admit two types of proto-invariant. In the first, the trajectory of a pliant variable x is contained in a sufficiently open CAD set. We refer to such a proto-invariant as a CAD.PI. In the second, the trajectory of a pliant variable x satisfies one of the pliant modalities discussed in Section 14: CONTINUOUS, n-DIFFERENTIABLE, PLENVL, PLENVU, PLENV, LBND, UBND, BND, MONINC, MONDEC, CVEX, CCAVE. We refer to such a proto-invariant as a MOD.PI.

In a CAD.PI the trajectory of x is confined to a suitable safety region, a reasonable engineering requirement. In a MOD.PI a particular characteristic of the pliant trajectory is demanded. Of the pliant modalities mentioned above, those from CONTINUOUS to BND are straighforwardly compatible with a variable remaining within a finite region indefinitely, in line with the safety region requirement just stated. The remainder, from MONINC to CCAVE, are not —unless relevant properties are forced to approach zero asymptotically— which would be unusual in an engineering context. Thus, an invariant in which any of these others could occur might be conditional on circumstances (e.g. the values of mode variables) that could be expected to hold only intermittently, at best.

Connected with the preceding point is the question of whether time itself ought to be permitted to occur in invariants. The intuitive notion of an invariant, as a property that holds at all times, suggests not. On the other hand, time's essence as a read-only variable that is synchronised once at the beginning of an execution and is thereafter not updated by any event, suggests that invariants that assert state properties that are conditional on the current time may be contemplated. The present author would discourage the latter kind of invariant, in case it prompts inappropriate intuitions at user level, but we do not forbid them here.[15]

Similar remarks apply, with different emphasis, to clocks. Clocks may be reset by mode events. Therefore, an invariant that says 'provided clock *casio_clk* says it is 3 o'clock, P holds' depends on what mode events reset *casio_clk* previously. One could thus say that such an invariant is very incomplete, in that a lot of the context that makes precise under what conditions *casio_clk* would say '3 o'clock' has been omitted from the invariant. One may even argue that an invariant that says 'provided 3 hours have elapsed since the start of the execution, P holds' has a more universally understood meaning. At any rate, we would urge the maximum caution if, despite the misgivings just described, such invariants were to be used.

18. Pliant Event Invariant Preservation, PliEv/INV

The invariant preservation PO for pliant events, reproduced from Fig. 6 is:

PliEv/INV:

$$\begin{aligned}
&I(u(\mathbb{t}_\mathrm{L})) \land iv_{PliEvA}(u(\mathbb{t}_\mathrm{L})) \land grd_{PliEvA}(u(\mathbb{t}_\mathrm{L})) \land (\exists \mathbb{t}_\mathrm{R} > \mathbb{t}_\mathrm{L} \bullet \mathrm{TRM}(\mathbb{t}_\mathrm{R}) \land (\forall \mathbb{t}_\mathrm{L} \le t < \mathbb{t}_\mathrm{R},\\
&u(t), i?(t), l(t), o!(t) \bullet BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \land SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t)))\\
&\qquad \Rightarrow (\forall \mathbb{t}_\mathrm{L} \le t < \mathbb{t}_\mathrm{R} \bullet I(u(t)))
\end{aligned} \tag{23}$$

As indicated in Section 17, since invariants can exhibit fairly arbitrary dependencies between mode and pliant variables that are too general to permit worthwhile generic analysis, in this section we focus on the proto-invariants described there, assuming that it is the truth of any such proto-invariant that is needed to discharge the invariant that contains it. More so than with feasibility, we focus on cases that allow (proto-)invariant preservation to be deduced rather easily, since small departures from the simplest cases tend to make the complexity of the task comparable to that of the general case.

[15]The analogue, in a purely discrete event framework, of having an invariant that depended on time, would be to have a read-only variable that counted all the event occurrences in an execution, and to have an invariant that depended on it. Described thus, it comes across as a rather unnatural thing to do.

Proto-invariants come in two kinds: CAD.PI and MOD.PI, as already stated. Along with the CAD.PIs, we introduce *internal* CAD proto-invariants CAD.IPIs, which we focus on where appropriate. A CAD.IPI is defined like a CAD.PI, and is intended to be contained in some enclosing CAD.PI. Just as when in proving algorithms correct we sometimes strengthen the invariant for technical convenience, a CAD.IPI can strengthen the invariant given by its enclosing CAD.PI for technical convenience. Thus, whereas a CAD.PI originates from the original invariants, and thus reflects the concerns of the application being modelled, a CAD.IPI may be chosen with the concerns of formal verification uppermost. Provided the containment is as stated, both sets of concerns may be optimally addressed.

Below, we seek to identify conditions that enable us to deduce that a proto-invariant (PI) is safely maintained by a pliant transition. This can happen in a number of ways. Firstly, we may be able to deduce that the PI holds indefinitely. In such a case, whether or not any particular transition is preempted by a mode transition becomes irrelevant to the correctness of the pliant transition. Secondly, we may be able to deduce that the PI holds until a time at which the pliant transition becomes infeasible. In that case, if the state reached at the limit of feasibility enables a mode transition, then the pliant transition is preempted and the execution continues, or if it does not, then we have finite termination. Both are regarded as correct according to the semantics of PaperI. Thirdly, we may be able to deduce that the PI holds for a (potentially undetermined) period, but that the transition may subsequently violate the PI. This is incorrect, and in such a case, we must identify additional conditions that prevent the violation in order to deduce safe execution.

Definition 18.1 (Pliant Event Maintains Property). *We say that a pliant event PliEv in machine M maintains a property Φ iff, in every execution of M every pliant transition Π specified by PliEv either: (a) is preempted by a mode transition, or (b) becomes infeasible (thus ending the execution of Π), or (c) continues indefinitely; and in each of these three cases, Φ holds throughout Π if it holds at the start, $\mathbb{t}_{\mathrm{L}}$.*

Proposition 18.2 (PliEv/INV/CAD.PI-BDA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-⊛ : $\{\ iv \wedge iv_{B/DA}\ ; grd \mid ll \mid BDA$ ▪-;-$\}$, and*
2. *PliEv-⊛ satisfies the feasibility criteria of Proposition 16.1, and*
3. (a) *if BDA contains CAD constraints, then CAD $\Rightarrow$ CAD.PI, and*
   (b) *if BDA contains MOD constraints, then*
   ($\forall x \bullet$ CAD.PI can be written as CAD.PIx $\wedge$ CAD.PI$\overline{x}$ where CAD.PIx contains all occurrences of x in CAD.PI and CAD.PI$\overline{x}$ contains no occurrences of x, and
      i. *if CAD.PIx bounds x above, then the MOD constraints contain PLENV(U) or (U)BND constraints which are adequate to prove the CAD.PIx bound on x, and*
      ii. *if CAD.PIx bounds x below, then the MOD constraints contain PLENV(L) or (L)BND constraints which are adequate to prove the CAD.PIx bound on x, and*
      iii. *if there is a PLENV(U) or PLENV(L) or MOD constraint for which there exists t_{INF}, where t_{INF} is the infimum of all times at which the MOD constraint becomes inconsistent with CAD.PIx and $t_{INF} > \mathbb{t}_{\mathrm{L}}$, then there is a TIME constraint that forces infeasibility at $t_{TIME} \leq t_{INF}$).*

Additionally, for each of cases 3.(a), 3.(b).i, 3.(b).ii, if BDA contains no constraints other than the ones mentioned, then any pliant transition of PliEv specified by that case continues indefinitely, or until preempted.

Proof: Let Π be a pliant transition satisfying *PliEv*. We examine the available alternatives. Suppose that case 3.(a) holds. Then *CAD.PI* is maintained as long as Π continues, which will be a non-empty interval of time, by feasibility. The time-independent nature of both *CAD* and *CAD.PI* ensures that if there are no other constraints in *BDA*, any behaviour of the variables respecting *BDA* can continue indefinitely, or until it is preempted by a mode transition. Alternatively, if there are additional constraints in *BDA*, they may become inconsistent with *CAD* at some future time, resulting in infeasibility of Π.

Suppose that case 3.(b) holds. Then for each relevant state variable x we can identify a top level conjunct of *CAD.PI* that contains all the occurrences of x in *CAD.PI*. If this conjunct demands upper or lower bounds on x, then cases 3.(b).i and 3.(b).ii ensure that the MOD constraints in *BDA* contain constraints strong enough to satisfy the demanded bounds. In case 3.(b).iii, the time dependent nature of the bound in the MOD constraint is prevented from allowing behaviour that violates *CAD.PI* by the hypothesised TIME constraint. The time-independent nature of constraint satisfaction in cases 3.(a), 3.(b).i, 3.(b).ii, ensures that if there are no other constraints in *BDA*, any behaviour of the variables respecting *BDA* can continue indefinitely, or until it is preempted by a mode transition. Alternatively, if there are additional constraints in *BDA*, they may become inconsistent with *CAD* at some future time, resulting in infeasibility of Π. $\square$

Evidently, the above could be strengthened to extend the scope of the $\forall x$ quantification to include case 3.(a) too, allowing some variables to be deduced safe on the basis of the CAD constraints of *BDA* and others to be handled via the MOD constraints. However, in such a framing of the proposition care has to be taken so that e.g., variables other than the x in question are *not* existentially quantified or otherwise inappropriately coupled to other variables, as that would be unsound. We do not pursue the complications that can arise in this way.

Proposition 18.3 (PliEv/INV/MOD.PI-BDA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-⊛* : $\pmb{\{}$ *iv* $\wedge$ $iv_{B/DA}$; *grd* | *ll* | *BDA* ■-;-$\pmb{\}}$, *and*
2. *PliEv-⊛ satisfies the feasibility criteria of Proposition 16.1, and*
3. *BDA contains the same MOD constraints as are in MOD.PI.*

Additionally, if BDA contains no constraints other than the ones mentioned in MOD.PI, then any pliant transition of PliEv continues indefinitely, or until preempted.

Proof: Let Π be a pliant transition satisfying *PliEv*. Feasibility ensures that Π has a non-empty duration, and it is obvious that MOD.PI can be maintained indefinitely provided the MOD constraints in *BDA* specify the same constraints as are demanded in MOD.PI. If there are additional MOD constraints, the argument is as in Proposition 18.2. $\square$

Evidently, in condition 3 of Proposition 18.3, we can also allow strengthenings of the MOD.PI constraints in MOD (e.g. by tightening an upper or lower bound), provided these do not lead to infeasibility. Similar observations apply below. For simplicity, we do not elaborate the details.

The next result concerns constraining the behaviour of an ODE system within a region, which is a well studied problem, so we just touch on the basics. The first few cases in clause 5 mimic the structure of Proposition 16.4 to the extent possible. After that, further cases consider mode event enabledness (and thus preemption) preempting escape from the region of constraint, and the availability of safe estimates on the behaviour to imply remaining within the desired region. Further comments follow the proof.

Proposition 18.4 (PliEv/INV/CAD.PI-DE). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Then, letting overline denote topological closure, PliEv maintains CAD.PI if:*

1. *PliEv*-$\circledast$: $\pmb{\{}$ *iv* ; *grd* | *ll* |-$\blacksquare$ $DE \equiv \mathcal{D}x = \phi$;-$\pmb{\}}$, *and*
2. *PliEv*-$\circledast$ *satisfies the feasibility criteria of Proposition 16.2, and*
3. *there exists an internal CAD PI, CAD.IPI, such that* $\overline{\mathit{CAD.IPI}} \subseteq \overline{\mathit{CAD.PI}}$, *and*
4. $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})} \in \mathit{CAD.IPI}$, *and*
5. (a) ($\forall x \bullet$ *if x is in the boundary of CAD.IPI, then the vector field* ϕ *points into the interior of CAD.IPI*), *or*
   (b) ($\forall x \bullet$ *if x is in the boundary of CAD.IPI, letting* $d(p)$ *denote the Euclidean distance from a point p to the boundary of CAD.IPI (positive for the interior, negative for the exterior), then* $\mathrm{D}(d;\phi(x)) > 0$, *where* $\mathrm{D}(f;v)$ *is the directional derivative of f in the direction v*), *or*
   (c) ($\forall x \bullet$ *if x is in the boundary of CAD.IPI, assuming CAD.IPI satisfies: (i) it is in disjunctive normal form,* $\mathit{CAD.IPI} \equiv \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} Q_{i,j}$, *(ii) each* $Q_{i,j}$ *is of the form* $Q_{i,j} \equiv P_{i,j} \diamond 0$, *where* $\diamond \in \{=,>\}$; *then, letting* $\hat{\imath}$ *indicate a disjunct such that x satisfies* $\bigwedge_{j=1}^{m_{\hat{\imath}}} Q_{\hat{\imath},j}$, *and letting* $\hat{\jmath}$ *range over all* $j \in \{1 \ldots m_{\hat{\imath}}\}$ *such that* $Q_{\hat{\imath},\hat{\jmath}}$ *is of the form* $P_{\hat{\imath},\hat{\jmath}} > 0$, *and it holds that* $P_{\hat{\imath},\hat{\jmath}}(x) = 0$, *we have, for all* $\hat{\jmath}$, *that* $\langle \phi(x), \nabla P_{\hat{\imath},\hat{\jmath}}(x) \rangle > 0$, *where* $\langle .,. \rangle$ *is the Euclidean inner product*), *or*
   (d) co-$\overline{\mathit{CAD.IPI}} \subseteq$ pre-$\mathbb{t}_{\mathrm{R}}$, *or* co-$\overline{\mathit{CAD.IPI}} \subseteq$ pre-$\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$, *as applicable, where* co- *denotes set complement, or*
   (e) ($\forall x \bullet \exists K_x \bullet$ *defining* $\tilde{t} = (t - \mathbb{t}_{\mathrm{L}})$, *if the solution to* $DE \equiv \mathcal{D}x = \phi$ *is given for* $\tilde{t} \leq K_x$ *by an absolutely convergent power series in* $\tilde{t}$, $x(\tilde{t}) = \sum_{k=0}^{\infty} a_k \tilde{t}^k$, *and is given for* $\tilde{t} \geq K_x$ *by an inverse power series* $x(\tilde{t}) = \sum_{k=0}^{\infty} b_k \tilde{t}^{-k}$, *and*
      i. *if CAD.IPI specifies an upper bound* X_{DU} *for x then, regarding the power series, there exist* N_{PU} *and* $T_{PU} \geq K_x$ *such that for* $k > N_{PU}$ *and* $\tilde{t} \geq T_{PU}$ *the terms in the series beyond index* N_{PU} *can be partitioned into groups such that the value of each group is negative, and the truncated series (i.e. polynomial)* $x_{\mathrm{trnPU}}(\tilde{t}) = \sum_{k=0}^{N_{PU}} a_k \tilde{t}^k$ *satisfies* $\tilde{t} \in [0 \ldots T_{PU}] \Rightarrow x_{\mathrm{trnPU}}(\tilde{t}) \leq X_{DU}$, *and regarding the inverse power series, there exist* N_{AU} *and* $T_{AU} \leq K_x$ *such that for* $k \leq N_{AU}$ *and* $\tilde{t} \geq T_{AU}$ *the truncated series (i.e. rational function)* $x_{\mathrm{trnAU}}(\tilde{t}) = \sum_{k=0}^{N_{AU}} b_k \tilde{t}^{-k}$ *satisfies* $\tilde{t} \in [T_{AU} \ldots \infty] \Rightarrow x_{\mathrm{trnAU}}(\tilde{t}) + \varepsilon_{\mathrm{AU}} \leq X_{DU}$, *where* $\varepsilon_{\mathrm{AU}}$ *is a safe bound for the terms in the series beyond index* N_{AU}, *and*
      ii. *if CAD.IPI specifies a lower bound* X_{DL} *for x then, regarding the power series, there exist* N_{PL} *and* $T_{PL} \geq K_x$ *such that for* $k > N_{PL}$ *and* $\tilde{t} \geq T_{PL}$ *the terms in the series beyond index* N_{PU} *can be partitioned into groups such that the value of each group is positive, and the truncated series (i.e. polynomial)* $x_{\mathrm{trnPU}}(\tilde{t}) = \sum_{k=0}^{N_{PL}} a_k \tilde{t}^k$ *satisfies* $\tilde{t} \in [0 \ldots T_{PL}] \Rightarrow x_{\mathrm{trnPL}}(\tilde{t}) \geq X_{DL}$, *and regarding the inverse power series, there exist* N_{AL} *and* $T_{AL} \leq K_x$ *such that for* $k \leq N_{AL}$ *and* $\tilde{t} \geq T_{AL}$ *the truncated series (i.e. rational function)* $x_{\mathrm{trnAL}}(\tilde{t}) = \sum_{k=0}^{N_{AL}} b_k \tilde{t}^{-k}$ *satisfies* $\tilde{t} \in [T_{AL} \ldots \infty] \Rightarrow x_{\mathrm{trnAL}}(\tilde{t}) - \varepsilon_{\mathrm{AL}} \geq X_{DL}$, *where* $\varepsilon_{\mathrm{AL}}$ *is a safe bound for the terms in the series beyond index* N_{AU},)

      or
   (f) ($\forall x \bullet$ *the characteristics of the solution to* $DE \equiv \mathcal{D}x = \phi$ *can be ascertained with enough precision that the number and value of any turning points and join points can be determined sufficiently accurately to show that they fall within any upper or lower bounds set by CAD.IPI — e.g. if DE is a linear system, and especially if it has constant coefficients*).

Let pre-$\mathbb{t}_{\mathrm{R}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ *denote the set of (path) connected components of* pre-$\mathbb{t}_{\mathrm{R}}$ *that have nonempty intersection with the component of CAD.IPI containing* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$. *Then, in addition to the preceding:*

6. *If* pre-$\mathbb{t}_{\mathrm{R}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ *is non-empty, and there is a* $0 < B \in \mathbb{R}$ *such that, for all* $x \in \mathit{CAD.IPI}$ *we have that* $\mathrm{D}(b;\phi) \geq B$, *where* $b(p)$ *is the Euclidean distance from a point p to the boundary of the nearest component of* pre-$\mathbb{t}_{\mathrm{R}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ *(positive for the interior, negative for the exterior), then the trajectory of PliEv from* $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$ *has finite duration, and is preempted at a time* $\mathbb{t}_{\mathrm{R}}$ *satisfying* $\mathbb{t}_{\mathrm{R}} \leq \mathbb{t}_{\mathrm{L}} + b(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})/B$.

Proof: Let Π be a pliant transition satisfying *PliEv*, for which the stated conditions hold. We permit checking the required conditions against *CAD.IPI* rather than *CAD.PI* in case they fail for *CAD.PI* but hold for a suitable subset. Since *CAD.IPI* may be smaller than *CAD.PI*, we need condition 4 to ensure that the initial value of x is in *CAD.IPI*. After this, the argument follows the proof of Proposition 16.4 broadly speaking.

Thus, Π will continue for a non-empty interval of time by feasibility, and if the region we need it to be confined to, hypothesised to be the internal proto-invariant *CAD.IPI*, has the property that at all times the vector field ϕ at its boundary points strictly into its interior, then the dynamics never escapes from *CAD.IPI*, which becomes an attractor for the dynamics. The alternatives in 5.(a)-(c) patterned after their counterparts in Proposition 16.4, give sufficient conditions for this; and if ϕ is independent of time (as *CAD.IPI* is assumed to be), then it is sufficient to ignore time in checking the conditions, whereas if ϕ depends on time, then the checks need to be established for all (relevant) times.

Thus, 5.(a) demands directly that ϕ points into the interior of *CAD.IPI* at its boundary, while 5.(b) relies on the distance function to the boundary, which will be non-zero in the non-empty interior of *CAD.IPI*. 5.(c) presumes a DNF form for the CAD structure of *CAD.IPI*, and the argument is as in Proposition 16.4.

For 5.(d), feasibility guarantees that Π will continue for a non-empty interval of time. Condition 5.(d) guarantees that if Π attempts to escape from *CAD.IPI*, it will encounter pre-$\mathbb{t}_{\mathrm{R}}$ first, and thus be preempted. (To preclude the possibility that a guard may be disabled (by conditions depending on time) at the moment the state trajectory enters pre-$\mathbb{t}_{\mathrm{R}}$, we assume that lazy mode events cooperate sufficiently to provide any preemption that depends on them. If not, we use pre-$\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$ instead.) If the encounter takes place at a boundary point of pre-$\mathbb{t}_{\mathrm{R}}$, the semantics of PaperI stipulates that preemption takes priority over invariant failure (should it be the case that invariant failure is a possibility), thus ensuring maintenance of *CAD.PI*.

For 5.(e), we assume that we can divide the trajectory of *DE* into two, a prefix for $\tilde{t} = t - \mathbb{t}_{\mathrm{L}}$ not lasting beyond $\tilde{t} = K_x$ and a suffix for $\tilde{t}$ extending from $\tilde{t} = K_x$ to infinity. Assuming that an upper bound, demanded by *CAD.IPI*, is required, for the former, we truncate the power series in a way that the discarded part contributes negatively, thus allowing us to assert that if the retained part satisfies the *CAD.IPI* upper bound (using CAD techniques for example), then so does the exact solution. For the latter, we truncate the inverse power series, assuming that there is a safe estimate for the retained part, that enables the deduction that the exact solution also satisfies the *CAD.IPI* upper bound. The argument for lower bounds is similar.

For 5.(f), we observe that the RHS of $DE \equiv \mathcal{D}x = \phi$ is a *FPHS* function, which we can write as $\langle \phi_1, \tilde{t}_{L,1}, \tilde{t}_{R,1}; \phi_2, \tilde{t}_{L,2}, \tilde{t}_{R,2}; \ldots ; \phi_n, \tilde{t}_{L,n}, \tilde{t}_{R,n} \rangle$. On each piece $(\phi_k, \tilde{t}_{L,k}, \tilde{t}_{R,k})$, the solution to *DE* will be analytic, and so its extremal values will occur among the turning points of ϕ_k and its values at $\tilde{t}_{L,k}, \tilde{t}_{R,k}$. If these can be ascertained with enough precision to determine whether any upper or lower bounds demanded by *CAD.IPI* are satisfied, then the preservation of *CAD.IPI* can be reduced to a finite number of checks. Moreover, if *DE* is linear, then its solution may be accessed via the standard variation of parameters form; and if *DE* is linear with constant coefficients, then the solution is available in closed form.

For 6, we firstly note that the preoccupation with connected components is motivated by the observation that Hybrid Event-B dynamics depend not only on pliant variables, but on mode variables too. These do not have any intrinsic notion of Euclidean distance, so we have to restrict to specific components, agreeing on the values of all the mode variables (which remain constant during a pliant transition), in order to have a meaningful result.[16]

[16]The same comments apply earlier, to the feasibility results. There though, the fact that the point of interest $\overleftarrow{x(\mathbb{t}_{\mathrm{L}})}$, lay on the boundary, effectively resolved the issue.

Now, the preceding conditions ensure that the trajectory of x stays within *CAD.IPI*, and assumption 6 states that the rate of approach of the trajectory of x to pre-$\mathbb{t}_{\mathrm{R}}(\overleftarrow{x(\mathbb{t}_{\mathrm{L}})})$ is bounded below by B, and since $b(x)$ is assumed always finite (otherwise it would be undefined), the expression given is the latest time at which preemption can occur, assuming that the environment will supply any needed existentially quantified values, and that any explicit time/clock constraints and dependencies are consistent. □

A number of comments on the preceding result are merited. In particular, the various subcases of condition 5 point the way to a much larger range of detailed approaches to deducing something about the region within which the solution to a *DE* system is confined. For example, the early subcases point the way to Lyapunov and related techniques; see for example [63, 135, 122, 71] not to mention many other sources. The use of polynomials and CAD techniques has been discussed earlier, and the use of asymptotic techniques can yield good information about large $\tilde{t}$ behaviour; see for example [132, 44, 47, 30, 68]. Global techniques can also be brought to bear; see for example [76, 77, 104]. Linear systems are familiar and are covered in many places, among which we can mention [130, 37, 9, 7, 102]. All of the topics just indicated are ancient, consequently the literature around them is extensive and we have just indicated essential entry points.

Beyond the preceding, and beyond the time invariant restrictions for CAD invariants adopted in this paper, lie more dynamic behaviours. For instance we may contemplate cases in which an initial consistency between CAD.PI and CAD.IPI (one or both of which may be time dependent), becomes an inconsistency, but where nevertheless, the correctness of the dynamics is rescued by a provably timely preemption. Some of these cases impinge on the results that follow.

Proposition 18.5 (PliEv/INV/MOD.PI-DE). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-*$\circledast$: $\pmb{\{}$ *iv* ; *grd* | *ll* |-■ $DE \equiv \mathcal{D}x = \phi$;-$\pmb{\}}$, *and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.2, and*
3. ($\forall x \bullet$ *if MOD.PI is any of the following forms, then the associated conditions apply (with substitution of* ϕ *for* $\mathcal{D}x$ *where relevant):*
   (a) CONTINUOUS(x)*: Proposition 14.1.2.(b),*
   (b) n-DIFFERENTIABLE(x)*: Proposition 14.2.2.(b),*
   (c) PLENVL(x, lb) *or* PLENVU(x, ub) *or* PLENV(x, lb, ub)*: Proposition 14.3.2.(b), as applicable,*
   (d) LBND(x, E_L) *or* UBND(x, E_U) *or* BND(x, E_L, E_U)*: Proposition 14.4.2.(b), as applicable,*
   (e) MONINC(x) *or* MONDEC(x)*: Proposition 14.5.2.(b), as applicable,*
   (f) CVEX(x) *or* CCAVE(x)*: Proposition 14.6.2.(b), as applicable*)

Proof: This result simply repackages results from the various propositions mentioned, and is included to record completeness of coverage. □

Proposition 18.6 (PliEv/INV/CAD.PI-DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-*$\circledast$: $\pmb{\{}$ $iv \wedge iv_{B/DA}$; *grd* | *ll* |-■-; $DA \equiv x := E$ $\pmb{\}}$, *and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.3, and*
3. ($\forall x \bullet x = E \Rightarrow$ *CAD.PI*)

Proof: Let *PliEv* be as in 1 and suppose that 2 holds. Feasibility ensures that any pliant transition Π satisfying *PliEv* starts out by maintaining *CAD.PI*. The fact that 3 holds independently of time then ensures that *CAD.PI* is maintained indefinitely. □

This result evidently generalises to the time dependent *CAD.PI* case, simply by interpreting condition 3 in a fully time dependent way.

Proposition 18.7 (PliEv/INV/MOD.PI-DA). *Let x stand for any of* $u, i?, l, o!$ *as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-*$\circledast$: $\{\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid$-■-; $DA \equiv x := E\ \}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.3, and*
3. $(\forall x \bullet$ *if MOD.PI is any of the following forms, then the associated conditions apply (with substitution of E for x where relevant):*
   (a) *CONTINUOUS*(x)*: Proposition 14.1.2.(c),*
   (b) *n-DIFFERENTIABLE*(x)*: Proposition 14.2.2.(c),*
   (c) *PLENVL*(x, lb) *or PLENVU*(x, ub) *or PLENV*(x, lb, ub)*: Proposition 14.3.2.(b), as applicable,*
   (d) *LBND*(x, E_L) *or UBND*(x, E_U) *or BND*(x, E_L, E_U)*: Proposition 14.4.2.(b), as applicable,*
   (e) *MONINC*(x) *or MONDEC*(x)*: Proposition 14.5.2.(b), as applicable,*
   (f) *CVEX*(x) *or CCAVE*(x)*: Proposition 14.6.2.(b), as applicable*$)$

Proof: This result simply repackages results from the various propositions mentioned, and is included to record completeness of coverage. □

Proposition 18.8 (PliEv/INV/CAD.PI-BDA/DE). *Let x stand for any of* $u, i?, l, o!$ *as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Let CAD.IPI* $\subseteq$ *CAD.PI be an internal proto-invariant for CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-*$\circledast$: $\{\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA$ ■ DE ;-$\}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.4, and*
3. *condition 3 of Proposition 18.2 holds, and*
4. *conditions 4-6 of Proposition 18.4 hold.*

Proof: The requirements of feasibility are dealt with in Proposition 16.4. Beyond that, we simply combine the conditions demended in Proposition 18.2 and Proposition 18.4 and conclude that if all hold as required, the proto-invariant will be maintained. (If inconsistency between the conditions demanded in the two propositions arises during the course of the execution, this is infeasibility, and causes finite termination — unless a mode event is enabled at the same moment, in which case preemption takes place.) □

For the remaining results, the arguments are, as in the previous result, just combinations of earlier arguments so the sketch proofs are omitted.

Proposition 18.9 (PliEv/INV/MOD.PI-BDA/DE). *Let x stand for any of* $u, i?, l, o!$ *as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-*$\circledast$: $\{\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA$ ■ DE ;-$\}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.4, and*
3. *condition 3 of Proposition 18.3 holds, and*
4. *condition 3 of Proposition 18.5 holds.*

Proposition 18.10 (PliEv/INV/CAD.PI-BDA/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Let CAD.IPI $\subseteq$ CAD.PI be an internal proto-invariant for CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-*$\circledast : \pmb{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA \blacksquare\text{-};\ DA\ \pmb{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.5, and*
3. *condition 3 of Proposition 18.2 holds, and*
4. *condition 3 of Proposition 18.6 holds.*

Proposition 18.11 (PliEv/INV/MOD.PI-BDA/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-*$\circledast : \pmb{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA \blacksquare\text{-};\ DA\ \pmb{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.5, and*
3. *condition 3 of Proposition 18.3 holds, and*
4. *condition 3 of Proposition 18.7 holds.*

Proposition 18.12 (PliEv/INV/CAD.PI-DE/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Let CAD.IPI $\subseteq$ CAD.PI be an internal proto-invariant for CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-*$\circledast : \pmb{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid\text{-}\blacksquare\ DE\ ;\ DA\ \pmb{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.6, and*
3. *conditions 4-6 of Proposition 18.4 hold, and*
4. *condition 3 of Proposition 18.6 holds.*

Proposition 18.13 (PliEv/INV/MOD.PI-DE/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

1. *PliEv-*$\circledast : \pmb{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid\text{-}\blacksquare\ DE\ ;\ DA\ \pmb{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.6, and*
3. *condition 3 of Proposition 18.5 holds, and*
4. *condition 3 of Proposition 18.7 holds.*

Proposition 18.14 (PliEv/INV/CAD.PI-BDA/DE/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant CAD.PI. Let CAD.IPI $\subseteq$ CAD.PI be an internal proto-invariant for CAD.PI. Then PliEv maintains CAD.PI if:*

1. *PliEv-*$\circledast : \pmb{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA \blacksquare\ DE\ ;\ DA\ \pmb{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.7, and*
3. *condition 3 of Proposition 18.2 holds, and*
4. *conditions 4-6 of Proposition 18.4 hold, and*
5. *condition 3 of Proposition 18.6 holds.*

Proposition 18.15 (PliEv/INV/MOD.PI-BDA/DE/DA). *Let x stand for any of $u, i?, l, o!$ as appropriate. Let the invariants I of a machine with pliant event PliEv contain a proto-invariant MOD.PI. Then PliEv maintains MOD.PI if:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathit{PliEvC}}{\text{PliEv/FIS}_C}}{\text{PliEv/INV}_C}}{\mathit{PliEvC}\ \boxed{\text{SOUND}}} \qquad \dfrac{\dfrac{\dfrac{\mathit{PliEvA}}{\text{PliEv/FIS}_A}}{\text{PliEv/INV}_A}}{\mathit{PliEvA}\ \boxed{\text{SOUND}}} \qquad \dfrac{\dfrac{\mathit{PliEvC}}{\textbf{PliEv/FISR}} \qquad \dfrac{\mathit{PliEvC} \quad \mathit{PliEvA}}{\text{PliEv/GRDR}}}{\textbf{PliEv/INVR}}}{\mathit{PliEvC}\ \boxed{\text{REFINES}}\ \mathit{PliEvA}}$$

Figure 12: Schematic pliant event refinement verification.

1. *PliEv-*$\circledast$: $\boldsymbol{\{}\ iv \wedge iv_{B/DA}\ ;\ grd \mid ll \mid BDA \blacksquare DE\ ;\ DA\ \boldsymbol{\}}$*, and*
2. *PliEv-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.7, and*
3. *condition 3 of Proposition 18.3 holds, and*
4. *condition 3 of Proposition 18.5 holds, and*
5. *condition 3 of Proposition 18.7 holds.*

Part 8 — Pliant POs: Refinement

19. Pliant Event Refinement Verification

While the procedure for proving that an event of an individual machine is correct is relatively straightforward in principle —one proves that the event is feasible, and then proves that its possible behaviours preserve the invariants— the procedure for proving that an event refinement is correct is a little more complicated. Fig. 12 indicates what is involved in a sequent style, the bold items highlighting steps involving pliant behaviour that are yet to be discussed in detail.

Suppose we wish to prove that *PliEvC* refines *PliEvA*. On the left of Fig. 12 are two branches that prove the soundness of *PliEvC* and *PliEvA* on their own terms. Presuming the soundness of *PliEvC* and *PliEvA*, the refinement is the concern of the right subtree. First, we strengthen PliEv/FIS$_C$ to get PliEv/FISR, as insisting that the intial state $w(\mathbb{t}_{\text{L}})$ of *PliEvC* is suitably connected to an abstract state (over and above what is demanded in PliEv/FIS$_C$) could conceivably render it infeasible. Alongside, we rely on the soundness of *PliEvC* and *PliEvA* to confirm that the guards of *PliEvC*, when mapped through the joint invariant K, always fall within the guards of *PliEvA*, giving PliEv/GRDR. Once all this is known, the simulation PO, PliEv/INVR, can be addressed, showing that each transition of *PliEvC* can be simulated by one of *PliEvA*. In particular, the truth of PliEv/FISR and PliEv/GRDR ensure that there is no need to address feasibility or enabledness of the desired *PliEvA* transition within PliEv/INVR itself.

20. Pliant Event Feasibility Under Refinement, PliEv/FISR

The feasibility PO for pliant events under refinement, reproduced from Fig. 8 is:

PliEv/FISR:

$$\begin{aligned}&(\exists u(\mathbb{t}_{\text{L}}) \bullet I(u(\mathbb{t}_{\text{L}})) \wedge K(u(\mathbb{t}_{\text{L}}), w(\mathbb{t}_{\text{L}})) \wedge iv_{\mathit{PliEvC}}(w(\mathbb{t}_{\text{L}})) \wedge grd_{\mathit{PliEvC}}(w(\mathbb{t}_{\text{L}}))\\ &\quad \Rightarrow (\exists \mathbb{t}_{\text{R}} > \mathbb{t}_{\text{L}} \bullet [\ (\mathbb{t}_{\text{R}} - \mathbb{t}_{\text{L}} \geq \delta_{\mathsf{Zeno}\mathit{PliEvC}}) \wedge\ ]\ (\forall \mathbb{t}_{\text{L}} < t < \mathbb{t}_{\text{R}} \bullet (\exists w(t), j?(t), k(t), p!(t) \bullet\\ &\qquad\qquad BDApred_{\mathit{PliEvC}}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{\mathit{PliEvC}}(w(t), j?(t), k(t), p!(t), t))))) \end{aligned} \tag{24}$$

We see that the only material difference between (21) and (24) is the presence of the extra assumption about the abstract variables at time $\mathbb{t}_{\text{L}}$ in (24). Therefore, aside from modifying hypothesis 2 in the propositions of Section 16 to include I and K suitably, we regard this PO as covered by Section 16.

21. Formal Reasoning about Hybrid Event-B: Joint Invariants

Just as for machine invariants, we focus on proto-invariants as the key ingredients of joint invariants.

Definition 21.1 (Regular Relation). *Let $R \subseteq \mathsf{X} \times \mathsf{Z}$ be a relation. Then R is a regular relation iff any of the following equivalent conditions holds:*

1. $R = R \,{\mathbin{;}}\, R^{\mathrm{T}} \,{\mathbin{;}}\, R$.
2. $\forall x, x' \in \mathsf{X} \bullet R(x,\_) \cap R(x',\_) \neq \varnothing \Rightarrow R(x,\_) = R(x',\_)$
   (and a corresponding result involving $R(\_,z)$ and $R(\_,z')$).
3. $\exists \mathsf{Y}, f : \mathsf{X} \twoheadrightarrow \mathsf{Y}, g : \mathsf{Z} \twoheadrightarrow \mathsf{Y} \bullet R = f \,{\mathbin{;}}\, g^{-1}$.
4. $\exists$ *a bijection $\approx$ between equipollent partitions of* $\mathrm{dom}\, R$ *and* $\mathrm{ran}\, R \bullet R(x,z) \Leftrightarrow [x] \approx [z]$.

where $\_^{\mathrm{T}}$ is relational transpose, $R(x,\_) = \{z \mid R(x,z)\}$, $R(\_,z) = R^{\mathrm{T}}(z,\_)$, $\twoheadrightarrow$ denotes a partial onto function, ${\mathbin{;}}$ is (forward) relational composition, and $[y]$ is the equivalence class of y in the relevant partition.

A regular relation R is primitive iff $R = \mathrm{dom}R \times \mathrm{ran}R$ (i.e. R is universal from its domain to its range).

The claimed equivalences are easily shown. Regular relations are so useful because they subsume functions, partial or not, inverse or not, as well as the more general cases. Especially because of their connection with functions and their inverses, they arise with great frequency in the refinement of specifications [99, 98, 14, 13], though the fact is not hugely appreciated.

Lemma 21.2. *Each regular relation R can be uniquely decomposed as a (disjoint) union of primitive regular relations:*

$$R \equiv \bigcup_{ix \in IX} R_{ix} \tag{25}$$

where IX is a suitable indexing set.

Proof: It is sufficient to identify the R_{ix} subrelations of R with primitive regular relations formed from the cartesian products of the equivalence classes of $\mathrm{dom}R$ and $\mathrm{ran}R$ that are paired by the bijection $\approx$ in Definition 21.1.4. Disjointness and uniqueness are relatively straightforward. □

A proto-invariant which is a regular relation between appropriate components of the abstract and concrete spaces X and Z of a Hybrid Event-B refinement will be referred to as an RR.PI, provided it satisfies suitable regularity conditions, similar to the definitions in Section 6. Specifically, we have the following.

Definition 21.3 (RR.PI). *Let $R : \mathsf{X} \leftrightarrow \mathsf{Z}$ be a regular relation. Let $X\!R \equiv \mathrm{dom}R$ and $Z\!R \equiv \mathrm{ran}R$. Then R is an RR.PI iff, for suitable n, m, k:*

1. *$X\!R$ is a CAD subset of the real subspace of $\mathbb{C}^n$;*
2. *$Z\!R$ is a CAD subset of the real subspace of $\mathbb{C}^m$;*
3. *$\exists \tilde{f} : \mathbb{C}^n \nrightarrow \mathbb{C}^k$ which**:** is holomorphic and of constant (complex) rank $k \leq n$ in an open neighbourhood of $\overline{X\!R}$**,** is real on the real subspace of $\mathbb{C}^n$**,** the graph of $\tilde{f}|_{\overline{X\!R}}$ is a CAD subset of $\mathbb{R}^n \times \mathbb{R}^k$**,** and $f \equiv \tilde{f}|_{X\!R}$;*
4. *$\exists \tilde{g} : \mathbb{C}^m \nrightarrow \mathbb{C}^k$ which**:** is holomorphic and of constant (complex) rank $k \leq m$ in an open neighbourhood of $\overline{Z\!R}$**,** is real on the real subspace of $\mathbb{C}^m$**,** the graph of $\tilde{g}|_{\overline{Z\!R}}$ is a CAD subset of $\mathbb{R}^m \times \mathbb{R}^k$**,** and $g \equiv \tilde{g}|_{Z\!R}$;*
5. *$\tilde{f}[\mathsf{X}] = \tilde{g}[\mathsf{Z}]$, and R arises as $f \,{\mathbin{;}}\, g^{-1}$, as in Definition 21.1.3.*

We note some redundancy in Definition 21.3. Since the domain and range of a function are existentially quantified over its graph, the facts that $\mathcal{XR}$ and $\mathcal{ZR}$ are CAD, follow from points 3 and 4. We will similarly assume below that any further properties required of an RR.PI R are available in computable form via the CAD nature of the graphs of f and g, including the representation of R using $R(x,z) \equiv (\exists y \bullet f_{\mathrm{gr}}(x,y) \wedge g_{\mathrm{gr}}(z,y))$, where f_{gr} and g_{gr} are the graphs of f and g. In the sequel, we will assume that the notations and conventions of Definitions 21.1 and 21.3 (in particular, the dimensionalities of the domains and ranges of f and g) are adopted by default.

RR.PIs are the only PIs we consider for the role of joint invariant in this paper. Although other kinds of invariant may be justifiably viewed as interesting in the pliant event refinement context —such as ones denoting approximate relationships of various kinds between abstract and concrete states— these often do not behave well *vis à vis* the refinement notion in (26), at least not without significant and detailed system-specific help (for example, regarding stability and/or contracting properties of the dynamics). For the sake of economy, we do not treat them in this paper, but leave their consideration for future work.

Lemma 21.4 (Trajectory Lifting). *Let $R : \mathsf{X} \leftrightarrow \mathsf{Z} \equiv f \,⨾\, g^{-1}$ be an RR.PI. Let $\Pi_C : [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}) \to \mathcal{ZR}$ be a $PHS^m[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ function and let $R(x_0, z_0)$ hold, where $z_0 = \Pi_C(\mathbb{t}_{\mathrm{L}})$. Then there is a $PHS^n[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ function $\Pi_A : [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}}) \to \mathcal{XR}$, such that $x_0 = \Pi_A(\mathbb{t}_{\mathrm{L}})$ and for all $t \in [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$, we have $R(\Pi_A(t), \Pi_C(t))$. If $n = k$ then Π_A is unique. If $n > k$ then there is a family of different functions Π_A satisfying the claimed properties. Similarly, when the roles of Π_C and Π_A are reversed.*

Proof: Consider $\chi = g \circ \Pi_C$. Since it is a composition of holomorphic functions, it is a $PHS^k[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ function. Since f is holomorphic, is of constant rank, and is onto its range, it is a submersion of $\mathcal{XR}$, and by the implicit function theorem, is a projection map, which thus foliates $\mathcal{XR}$ [90, 131, 61, 55]. Let $\mathcal{XR}_{x_0}$ be the leaf containing x_0.

For any $t \in [\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$, consider $f^{-1}(g \circ \Pi_C(t))$. If $n = k$, it is a singleton set (because f is injective), and we can take $\Pi_A(t)$ to be given by the single point it contains. This gives a unique Π_A, and the required holomorphy properties follow easily. If $n > k$, then there will be a local coordinate system for the fiber $f^{-1}(g \circ \Pi_C(t))$, which extends to a local coordinate system for $\mathcal{XR}$ at $\mathcal{XR}_{x_0} \cap f^{-1}(g \circ \Pi_C(t))$, made by adding coordinates in the leaf $\mathcal{XR}_{x_0}$ to the fiber coordinates. We can then choose Π_A so that its leaf coordinates remain within the leaf $\mathcal{XR}_{x_0}$, and its fiber coordinates vary arbitrarily within the fibers, provided the variation remains holomorphic and real on real coordinates. The required holomorphy properties again follow easily and the reverse result follows easily too. $\square$

Corollary 21.5. *Lemma 21.4 applies also to $FPHS^m[\mathbb{t}_{\mathrm{L}}, \mathbb{t}_{\mathrm{R}})$ functions.*

Proof: We need merely to repeat the construction for each succeeding holomorphic piece of Π_C, choosing for each such piece, the initial value in $\mathcal{XR}$ arbitrarily in the fiber above its initial value in $\mathcal{ZR}$, including the option of making Π_A stay in the same leaf. $\square$

The pliant event refinement PO (26) requires us to exhibit an abstract pliant transition corresponding to a given concrete one, satisfying a simulation property based on a joint invariant K. The fact that primitive (i.e. universal) relations are regular, and that products, intersections and disjoint unions of regular relations are regular, helps considerably in ensuring that an analysis based on RR.PIs extends to a useful wider class of joint invariants. The main tool for achieving the analysis will be the trajectory lifting lemma (and the details in its proof). Consistent with Notation 1.1, we do not distinguish between regular relations as (set theoretic) relations, and their definitions using logical expressions — so universal relations are left unstated or are true, and products and intersections are given by conjunctions.

22. Pliant Event Invariant Preservation Under Refinement, PliEv/INVR

The invariant preservation PO for pliant events under refinement, reproduced from Fig. 8 is:

PliEv/INVR:

$$\begin{aligned}
&I(u(\mathbb{t}_{\mathrm{L}})) \land K(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}})) \land iv_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \land grd_{PliEvC}(w(\mathbb{t}_{\mathrm{L}})) \;\Rightarrow \\
&\quad \big(\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet \mathrm{TRM}(\mathbb{t}_{\mathrm{R}}) \land (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}}, w(t), j?(t), k(t), p!(t) \bullet \\
&\quad\quad BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \land SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t)) \\
&\quad\quad\quad \Rightarrow (\forall \mathbb{t}_{\mathrm{L}} < t < \mathbb{t}_{\mathrm{R}} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet \\
&\quad\quad\quad\quad\quad BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \land SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \land \\
&\quad\quad\quad\quad\quad K(u(t), w(t))))\big)
\end{aligned} \tag{26}$$

In Section 16 we considered feasibility in detail, taking the various mechanisms available to define a pliant event, BDA, DE, DA, into account. Examining all non-empty combinations of these led to seven results. In Section 18 we considered invariant preservation in detail. Taking the two kinds of machine proto-invariant plus the pliant event mechanisms into account, doubled the number of results, some being covered quite briefly.

In the case of pliant event refinement, there are two pliant events in play, an abstract one and a concrete one, thus forty nine mechanism combinations to consider, even when there is only one kind of refinement proto-invariant. Since we observed in Section 18 that the more complex mechanism combinations were covered by merely conjoining the conditions pertinent to the individual mechanisms inside them, in this section, we take a similar approach to the complex cases, but without explicitly listing all the results regarding these conjunction-generated cases. Thus, the number of explicit results in this section reduces to a more manageable nine: one for each combination of a single abstract mechanism with a single concrete mechanism. We reduce our work further by noting that certain combinations, ones refining a more precise mechanism to a less precise one, e.g. DA refined to BDA, make little practical sense, so we do not treat them in detail. We also combine the four DE/DA cases into one result, including (because these mechanisms interact closely via the EV graph) the further cases that arise when both mechanisms are present in either machine.

Definition 22.1 (Syntactic and Semantic Refinement). *In the context of an abstract machine M_A containing $PliEv_A$ and a concrete machine M_C containing $PliEv_C$, the phrase '$PliEv_C$ is a syntactic refinement of $PliEv_A$' means that M_C declares that $PliEv_C$ refines $PliEv_A$. In the same context, the phrase '$PliEv_C$ is a semantic refinement of $PliEv_A$ via RR.PI' means that the PO (26), with K instantiated to RR.PI, and with suitable initial values, is, in fact, provable.*[17]

Proposition 22.2 (PliEv/INVR/RR.PI-BDA›BDA). *Let x stand for any of $u, i?, l, o!$ as appropriate, the variables of an abstract pliant event $PliEv_A$. Let z stand for any of $w, j?, k, p!$ as appropriate, the variables of a concrete pliant event $PliEv_C$. Let $PliEv_C$ be a syntactic refinement of $PliEv_A$. Let the joint invariants of the refinement contain an RR.PI proto-invariant $R : \mathsf{X} \leftrightarrow \mathsf{Z}$. Suppose $I(u(\mathbb{t}_{\mathrm{L}})) \land R(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}}))$ holds. Then $PliEv_C$ is a semantic refinement of $PliEv_A$ via RR.PI if:*

1. *$PliEv_A$-$\circledast$: $\pmb{\{}$ iv_A ; $grd_A \mid i?, l, o! \mid BDA_A$ ■-;-$\pmb{\}}$,*
   $PliEv_C$-$\circledast$: $\pmb{\{}$ iv_C ; $grd_C \mid j?, k, p! \mid BDA_C$ ■-;-$\pmb{\}}$, and
2. *$PliEv_A$-$\circledast$ satisfies the feasibility criteria of Proposition 16.1, and*

[17]This implies that if the actual joint invariant K between the two machines is a more complex expression than just $RR.PI$ alone, then, at minimum, all occurrences of $RR.PI$ in K are positive, i.e. they are under an even number of negations.

3. *$PliEv_C$-$\circledast$ satisfies the feasibility criteria of Proposition 16.1 and Section 20, and*
4. $(\forall z \bullet BDA_C \wedge R \Rightarrow BDA_A)$*, and (alternatively)*
5. *if BDA_A contains CAD constraints CAD_A, then*
   (a) *BDA_C contains CAD constraints CAD_C, and $(\forall z \bullet CAD_C \wedge R \Rightarrow CAD_A)$, or*
   (b) *BDA_C contains MOD constraints MOD_C, and*
   $(\forall x \bullet$ if x is needed to satisfy the hypotheses of any of clauses i-iii below, then CAD_A can be written as $CAD_A x \wedge CAD_A \overline{x}$ where $CAD_A x$ contains all occurrences of x in CAD_A and $CAD_A \overline{x}$ contains no occurrences of x, and
      i. *if $CAD_A x$ bounds x above, then the MOD constraints MOD_C contain* PLENV(U) *or* (U)BND *constraints which, conjoined with R (as in 4), are sufficient to prove the $CAD_A x$ bound on x, and*
      ii. *if $CAD_A x$ bounds x below, then the MOD constraints MOD_C contain* PLENV(L) *or* (L)BND *constraints which, conjoined with R (as in 4), are sufficient to prove the $CAD_A x$ bound on x, and*
      iii. *if MOD_C contains a* PLENV(U) *or* PLENV(L) *or other MOD constraint for which there exists t_{INF}, where t_{INF} is the infimum of all times at which the MOD constraint, conjoined with R (as in 4), becomes inconsistent with $CAD_A x$ and $t_{INF} > \mathbb{t}_{\mathrm{L}}$, then there is a TIME constraint $TIME_C$ that forces infeasibility at $t_{TIME_C} \leq t_{INF}$), and*
6. *if BDA_A contains MOD constraints MOD_A, then BDA_C contains MOD constraints MOD_C which, conjoined with R (as in 4), are sufficient to prove MOD_A, and*
7. *if BDA_A contains TIME constraint $TIME_A$, then BDA_C contains TIME constraint $TIME_C$ or MOD constraint MOD_C that forces infeasibility no later than $TIME_A$ does.*

Clause 4 of this proposition states the obvious, in demanding that BDA_C, mapped through the joint invariant R, satisfies BDA_A. The clauses that follow are all special cases of clause 4, structured to follow the form of results in Section 18. Specifically, in Section 18 we had non-trivial invariants, and non-trivial specifications of behaviour that had to conform to them. Here, the earlier non-trivial invariants are replaced by the abstract behaviour specifications, and the earlier non-trivial behaviours are replaced by concrete behaviour specifications that are mapped through R.

Proof: Let Π_C be a feasible concrete pliant transition with its intial state $w(\mathbb{t}_{\mathrm{L}})$ *RR.PI*-related to an abstract state $u(\mathbb{t}_{\mathrm{L}})$ which satisfies the abstract invariant I. We apply the Lifting Lemma 21.4 to get an abstract pliant transition Π_A. We will assume that Π_A stays within the leaf containing $u(\mathbb{t}_{\mathrm{L}})$. It is immediate from the lemma that *RR.PI* is satisfied throughout the duration of Π_A, so it remains to check, across the cases, that if the assumed behavioural specification holds for Π_C, then the corresponding required behavioural specification holds for Π_A.

It is now evident that case 5 supports an argument based on the proof of Proposition 18.2; that case 6 supports an argument based on the proof of Proposition 18.3; and that case 7 discharges the PO correctly too (since the semantics of PaperI permits a concrete execution to terminate sooner than the abstract execution it refines). □

In the rest of this section, the difunctional form $R \equiv f \,_{9}^{\circ}\, g^{-1}$ of an *RR.PI* enables us to use properties involving the middle space Y.

Proposition 22.3 (PliEv/INVR/RR.PI-BDA›DA). *Let x stand for any of $u, i?, l, o!$ as appropriate, the variables of an abstract pliant event $PliEv_A$. Let z stand for any of $w, j?, k, p!$ as appropriate, the variables of a concrete pliant event $PliEv_C$. Let $PliEv_C$ be a syntactic refinement of $PliEv_A$. Let the joint invariants of the refinement contain an RR.PI proto-invariant $R : \mathsf{X} \leftrightarrow \mathsf{Z} \equiv f \,_{9}^{\circ}\, g^{-1}$, where the ranges of f and g are in $\mathsf{Y} \ni y$. Referring to item 1 below, let $E_y \equiv g \circ E$, and let $DA_y \equiv y := E_y$. Suppose $I(u(\mathbb{t}_{\mathrm{L}})) \wedge R(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}}))$ holds. Then $PliEv_C$ is a semantic refinement of $PliEv_A$ via RR.PI if:*

1. *$PliEv_A$-$\circledast$: $\pmb{\{}$ iv_A ; grd_A | $i?, l, o!$ | BDA ■-;-$\pmb{\}}$,*
   $PliEv_C$-$\circledast$: $\pmb{\{}$ iv_C ; grd_C | $j?, k, p!$ |-■-; $DA \equiv z := E$ $\pmb{\}}$, and
2. *$PliEv_A$-$\circledast$ satisfies the feasibility criteria of Proposition 16.1, and*
3. *$PliEv_C$-$\circledast$ satisfies the feasibility criteria of Proposition 16.2 and Section 20, and*
4. *if BDA_A contains CAD constraints CAD, letting $CAD_y \equiv (\exists x \bullet CAD \wedge f)$, then the conditions of Proposition 18.6 hold, with CAD_y, E_y and DA_y playing the roles of CAD.PI, E and DA respectively in the Proposition, and*
5. *if BDA contains MOD constraints MOD, then*
   ($\forall x \bullet$ if MOD is any of the following forms, then the associated conditions apply (with substitution of E_y for E where relevant in the cited Propositions):
   (a) *$\mathsf{CONTINUOUS}(x)$:*
   Proposition 14.1.2.(c) applied to DA, so that it satisfies $\mathsf{CONTINUOUS}(z)$,
   (b) *n-$\mathsf{DIFFERENTIABLE}(x)$:*
   Proposition 14.2.2.(c) applied to DA, so that it satisfies n-$\mathsf{DIFFERENTIABLE}(z)$,
   (c) *$\mathsf{PLENVL}(x, lb)$ or $\mathsf{PLENVU}(x, ub)$ or $\mathsf{PLENV}(x, lb, ub)$:*
   provided f and g are both monotonically increasing, either or both of Proposition 14.3.2.(b), as applicable, applied to DA, so that it satisfies
   $\mathsf{PLENVL}(z, R[lb])$ or $\mathsf{PLENVU}(z, R[lb])$ or $\mathsf{PLENV}(z, R[lb], R[lb])$ respectively,
   (d) *$\mathsf{LBND}(x, E_L)$ or $\mathsf{UBND}(x, E_U)$ or $\mathsf{BND}(x, E_L, E_U)$:*
   provided f and g are both monotonically increasing, either or both of Proposition 14.4.2.(b), as applicable, applied to DA, so that it satisfies
   $\mathsf{LBND}(z, R[E_L])$ or $\mathsf{UBND}(z, R[E_U])$ or $\mathsf{BND}(z, R[E_L], R[E_U])$ respectively,
   (e) *$\mathsf{MONINC}(x)$ or $\mathsf{MONDEC}(x)$:*
   provided f and g are both monotonically increasing, Proposition 14.5.2.(b), as applicable, applied to DA, so that it satisfies $\mathsf{MONINC}(z)$ or $\mathsf{MONDEC}(z)$ respectively,
   (f) *$\mathsf{CVEX}(x)$ or $\mathsf{CCAVE}(x)$:*
   provided g is monotonically increasing,
      i. *for $\mathsf{CVEX}(x)$: [either f^{-1} is convex, or f is decreasing, invertible and convex] and Proposition 14.5.2.(b), as applicable, applied to DA, so that it satisfies $\mathsf{MONINC}(z)$,*
      ii. *for $\mathsf{CCAVE}(x)$: [either f^{-1} is concave, or f is increasing, invertible and convex] and Proposition 14.5.2.(b), as applicable, applied to DA, so that it satisfies $\mathsf{MONINC}(z)$).*

Proof: Let Π_C be a *DA* trajectory. As earlier, let $y = g \circ \Pi_C$ lift Π_C to Y. Let $u(\mathbb{t}_\mathrm{L})$ be the abstract state corresponding to $\Pi_C(\mathbb{t}_\mathrm{L})$ in the hypotheses of (26).

For case 4, adherence to a CAD constraint hinges on containment (of a trajectory within the CAD set). Now, set containment is preserved by both image and inverse image through a function. So, if y remains within CAD_y, then the further lift of y via f^{-1} to a trajectory within the leaf containing $u(\mathbb{t}_\mathrm{L})$ will remain within $f^{-1}(CAD_y) = CAD$, as required.

For case 5, compliance with a MOD constraint hinges on verifying generic properties of trajectories. These are typically preserved through holomorphic maps, with the addition of some further assumptions at times. For 5.(a),(b), n-differentiability (including $n = 0$) follows from the facts that Π_C is holomorphic, g is holomorphic, f^{-1} within the leaf containing $u(\mathbb{t}_\mathrm{L})$ is holomorphic, and the relevant propositions take care of any join points.

For the remaining cases, the caveats and assumptions of Section 14.2 regarding the interpretation of $\leq, \geq$ according to the formulation in Section 12.3 apply.

For 5.(c),(d), If $z \geq z_{lb}$ for all $z_{lb} \in g^{-1}(f(lb))$, then $g(z) \geq f(lb)$ since g is monotonic. Consequently, if $f(x) = g(z)$, then $x \geq lb$ since f is monotonic and inverse image maintains non-strict monotonicity (provided 'equal cases are suitable ordered'). Thus $\mathsf{PLENVL}(x, lb)$ follows for x in the leaf containing

$u(\mathbb{t}_{\mathrm{L}})$ if all z visited in Π_C satisfy $z \geq z_{lb}$ for all $z_{lb} \in g^{-1}(f(lb))$, i.e. $\mathsf{PLENVL}(z, R[lb]))$. The other results are analogous.

For 5.(e), the preceding argument may be readily adapted to the increase (or decrease) over time, rather than the maintenance of a bound.

For 5.(f).i, if f^{-1} is convex and g is monotonically increasing, then $f^{-1} \circ g$ is evidently convex on arguments that increase over time within the leaf containing $u(\mathbb{t}_{\mathrm{L}})$. Alternatively, if f is decreasing, invertible and convex, then f^{-1} is convex [101], so precomposing with monotonically increasing g yields $f^{-1} \circ g$ which is again convex on arguments that increase over time within the leaf containing $u(\mathbb{t}_{\mathrm{L}})$. The proposition adds conditions that take care of join points. For 5.(f).ii, the reasoning is dual. $\square$

We note that 5.(f).i/ii above indicate only a small selection of results that can be obtained by considering combinations of in/dec-rease, (non-)invertibility, convex/concav-ity, etc. for both f and g — we regard such further results as of less interest. We note further that we cannot exploit the conditions in Proposition 14.6.2.(b) directly (i.e. without additional guarantees), since, even if $f^{-1} \circ g$ is increasing, its rate of increase may be too low to maintain the convexity of a Π_C assured convex using the Proposition.

In the results below, the notation $J^y_z = \partial g / \partial z$ denotes the Jacobian of partial derivatives of the components of the range of g (held in vector y) with respect to the components of the domain of g (held in vector z); see Section 12.2.

Proposition 22.4 (PliEv/INVR/RR.PI-BDA›DE). *Let x stand for any of $u, i?, l, o!$ as appropriate, the variables of an abstract pliant event $PliEv_A$. Let z stand for any of $w, j?, k, p!$ as appropriate, the variables of an concrete pliant event $PliEv_C$. Let $PliEv_C$ be a syntactic refinement of $PliEv_A$. Let the joint invariants of the refinement contain an RR.PI proto-invariant $R : \mathsf{X} \leftrightarrow \mathsf{Z} \equiv f \mathbin{⨾} g^{-1}$, where the ranges of f and g are in $\mathsf{Y} \ni y$. Let $J^y_z = \partial g / \partial z$ be the Jacobian matrix of g, and, referring to item 1 below, let $\phi_y \equiv (\exists z \bullet J^y_z \wedge \phi)$, and let $DE_y \equiv \mathcal{D}y = \phi_y$. Suppose $I(u(\mathbb{t}_{\mathrm{L}})) \wedge R(u(\mathbb{t}_{\mathrm{L}}), w(\mathbb{t}_{\mathrm{L}}))$ holds. Then $PliEv_C$ is a semantic refinement of $PliEv_A$ via RR.PI if:*

1. *$PliEv_A$-⊛ : $\pmb{\{}$ iv_A ; grd_A | $i?, l, o!$ | BDA ▪-;-$\pmb{\}}$,*
   $PliEv_C$-⊛ : $\pmb{\{}$ iv_C ; grd_C | $j?, k, p!$ |-▪ $DE \equiv \mathcal{D}z = \phi$;-$\pmb{\}}$, and
2. *$PliEv_A$-⊛ satisfies the feasibility criteria of Proposition 16.1, and*
3. *$PliEv_C$-⊛ satisfies the feasibility criteria of Proposition 16.2 and Section 20, and*
4. *if BDA contains CAD constraints CAD, letting CAD_y be given by $(\exists x \bullet CAD \wedge f)$, then the conditions of Proposition 18.4 hold, with CAD_y and DE_y playing the role of CAD.PI and DE respectively in the Proposition, and*
5. *if BDA contains MOD constraints MOD, then*
   $(\forall x \bullet$ if MOD is any of the following forms, then the associated conditions apply:
   (a) *$\mathsf{CONTINUOUS}(x)$:*
   Proposition 14.1.2.(b), applied to DE, so that its integral curve satisfies $\mathsf{CONTINUOUS}(z)$,
   (b) *n-$\mathsf{DIFFERENTIABLE}(x)$:*
   Proposition 14.2.2.(b), applied to DE, so that its integral curve satisfies n-$\mathsf{DIFFERENTIABLE}(z)$,
   (c) *$\mathsf{PLENVL}(x, lb)$ or $\mathsf{PLENVU}(x, ub)$ or $\mathsf{PLENV}(x, lb, ub)$:*
   provided f and g are both monotonically increasing, either or both of Proposition 14.3.2.(b), as applicable, applied to DE, so that its integral curve satisfies
   $\mathsf{PLENVL}(z, R[lb])$ or $\mathsf{PLENVU}(z, R[ub])$ or $\mathsf{PLENV}(z, R[lb], R[ub])$ respectively,
   (d) *$\mathsf{LBND}(x, E_L)$ or $\mathsf{UBND}(x, E_U)$ or $\mathsf{BND}(x, E_L, E_U)$:*
   provided f and g are both monotonically increasing, either or both of Proposition 14.4.2.(b), as applicable, applied to DE, so that its integral curve satisfies
   $\mathsf{LBND}(z, R[E_L])$ or $\mathsf{UBND}(z, R[E_U])$ or $\mathsf{BND}(z, R[E_L], R[E_U])$ respectively,

(e) *$\mathsf{MONINC}(x)$ or $\mathsf{MONDEC}(x)$: provided f and g are both monotonically increasing, Proposition 14.5.2.(b), as applicable, applied to DE, so that its integral curve satisfies $\mathsf{MONINC}(z)$ or $\mathsf{MONDEC}(z)$ respectively,*

(f) *$\mathsf{CVEX}(x)$ or $\mathsf{CCAVE}(x)$:*
provided g is monotonically increasing,

i. *for $\mathsf{CVEX}(x)$: $[$ either f^{-1} is convex, or f is decreasing, invertible and convex $]$ and Proposition 14.5.2.(b), as applicable, applied to DE, so that its integral curve satisfies $\mathsf{MONINC}(z)$,*

ii. *for $\mathsf{CCAVE}(x)$: $[$ either f^{-1} is concave, or f is increasing, invertible and convex $]$ and Proposition 14.5.2.(b), as applicable, applied to DE, so that its integral curve satisfies $\mathsf{MONINC}(z)$).*

Proof: Let Π_C be an integral curve of *DE*. It satisfies $\mathcal{D}z = \phi$. As earlier, let $g \circ \Pi_C$ lift Π_C to Y. Then $g(\Pi_C)$ will satisfy $\mathcal{D}y = J_z^y \circ \phi = \phi_y$.

For case 4, if $g(\Pi_C)$ satisfies the CAD constraints CAD_y, and $u(\mathbb{t}_L)$ is the abstract state corresponding to $\Pi_C(\mathbb{t}_L)$ in the hypotheses of (26), then the further lifting of $g(\Pi_C)$ via f^{-1} to the leaf containing $u(\mathbb{t}_L)$ will satisfy the CAD constraints *CAD* of *BDA*. The proposition supplies conditions under which $g(\Pi_C)$ satisfies CAD_y.

For case 5, as before, compliance with a MOD constraint hinges on verifying generic properties, with the addition of further assumptions where needed. For 5.(a),(b), re. n-differentiability (including $n = 0$), it follows that if Π_C is n-$\mathsf{DIFFERENTIABLE}$, then the holomorphy of g and f^{-1} within the leaf containing $u(\mathbb{t}_L)$ ensure the required property, and the stated conditions ensure n-differentiability of Π_C as well as taking care of any join points.

For the remaining cases, the caveats and assumptions of Section 14.2 regarding the interpretation of $\leq, \geq$ according to the formulation in Section 12.3 apply.

For 5.(c),(d), the argument is as in Proposition 22.3, with the cited proposition supplying sufficient conditions, and for those, the zeros of $\phi - \mathcal{D}lb$ identify the needed non-join-point stationary points. The other results are analogous.

For 5.(e), the preceding argument may be readily adapted to the increase (or decrease) over time, rather than the maintenance of a bound.

For 5.(f), the reasoning is similar to the corresponding case in Proposition 22.3. $\square$

Proposition 22.5 (PliEv/INVR/RR.PI-DA›BDA). This would refine a behaviour specified by a direct assignment to a behaviour specified by CAD or PM or TIME constraints, which are usually employed for more loosely defined behaviours, so is not of great practical interest. (In any case, aside from the direction of the central implication, it can be viewed as symmetric to Proposition 22.3).

Proposition 22.6 (PliEv/INVR/RR.PI-DE›BDA). This would refine a behaviour specified by a differential equation to a behaviour specified by CAD or PM or TIME constraints, which are usually employed for more loosely defined behaviours, so is not of great practical interest. (In any case, aside from the direction of the central implication, it can be viewed as symmetric to Proposition 22.4).

Proposition 22.7 (PliEv/INVR/RR.PI-DE/DA›DE/DA). *Let x, partitioned into $x_{\mathrm{DE}}, x_{\mathrm{DA}}$, stand for any of $u, i?, l, o!$ as appropriate, the variables of an abstract pliant event $PliEv_A$. Let z, partitioned into $z_{\mathrm{DE}}, z_{\mathrm{DA}}$, stand for any of $w, j?, k, p!$ as appropriate, the variables of a concrete pliant event $PliEv_C$. Let $PliEv_C$ be a syntactic refinement of $PliEv_A$. Let the joint invariants of the refinement contain an RR.PI proto-invariant $R : \mathsf{X} \leftrightarrow \mathsf{Z} \equiv f \mathbin{⨾} g^{-1}$, where the ranges of f and g are in $\mathsf{Y} \ni y$. Let $J_z^y = \partial g / \partial z$ be the Jacobian matrix of g, and, referring to item 1 below, let $E_{C,y} \equiv (\exists z \bullet J_z^y \wedge dE_C/dt)$, let $\phi_{C,y} \equiv (\exists z \bullet J_z^y \wedge \phi_C)$, and let $DE_{C,y} \equiv \mathcal{D}y = E_{C,y} + \phi_{C,y}$. Let $J_x^y = \partial f / \partial x$ be the Jacobian matrix of f, and, referring to item 1* *below, let* $E_{A,y} \equiv (\exists z \bullet J_x^y \wedge dE_A/dt)$*, let* $\phi_{A,y} \equiv (\exists x \bullet J_x^y \wedge \phi_A)$*, and let* $DE_{A,y} \equiv \mathcal{D}y = E_{A,y} + \phi_{A,y}$*. Suppose* $I(u(\mathbb{t}_\mathrm{L})) \wedge R(u(\mathbb{t}_\mathrm{L}), w(\mathbb{t}_\mathrm{L}))$ *holds. Then* $PliEv_C$ *is a semantic refinement of* $PliEv_A$ *via RR.PI if:*

1. $PliEv_A$*-*$\circledast$: $\{$ iv_A ; grd_A | $i?,\text{-},o!$ |-■ $DE_A \equiv \mathcal{D}x_{\mathrm{DE}} = \phi_A$; $DA_A \equiv x_{\mathrm{DA}} := E_A$ $\}$,
   $PliEv_C$*-*$\circledast$: $\{$ iv_C ; grd_C | $j?,\text{-},p!$ |-■ $DE_C \equiv \mathcal{D}z_{\mathrm{DE}} = \phi_C$; $DA_C \equiv z_{\mathrm{DA}} := E_C$ $\}$*, and*
2. $PliEv_A$*-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.1, and*
3. $PliEv_C$*-*$\circledast$ *satisfies the feasibility criteria of Proposition 16.2 and Section 20, and*
4. *for every choice of* $j?$*, there is an* $i?$*, both ranging over* $[\mathbb{t}_\mathrm{L} \ldots \mathbb{t}_\mathrm{R})$*, such that:*
   $DE_{A,y} \equiv \mathcal{D}y = E_{A,y} + \phi_{A,y} = E_{C,y} + \phi_{C,y} = \mathcal{D}y \equiv DE_{C,y}$.

Proof: Let Π_C be a solution to $PliEv_C$-$\circledast$ consisting of a directly assigned trajectory for z_{DA} and an integral curve for z_{DE}. Let Π_A refer to the corresponding solution to $PliEv_A$-$\circledast$, desired according to (26), and already given by DE_A and DA_A. The given Π_C will be a refinement of the given Π_A if f maps Π_A to the same thing to which g maps Π_C. Since both abstract and concrete behaviours are deterministic once the functions $i?$ and $j?$ have been chosen to satisfy the stated criteria, the desired equality can be established by considering either the solution in Y itself, or the differential equation that it satisfies. Since differentiation is more readily done than integration, we opt for the latter. The ODE satisfied by mapping the desired Π_A via f is $DE_{A,y}$, while the ODE satisfied by mapping the given Π_C via g is $DE_{C,y}$. If the right hand sides of these two ODEs are the same, then the fact that $R(u(\mathbb{t}_\mathrm{L}), w(\mathbb{t}_\mathrm{L}))$ holds means that $f(u(\mathbb{t}_\mathrm{L})) = g(w(\mathbb{t}_\mathrm{L}))$, i.e. both ODEs also have the same initial values, from which the result follows by the uniqueness of solutions to ODEs that have the Lipschitz property. □

23. Formal Reasoning about Hybrid Event-B: POs with Witnesses

The POs for feasibility and invariant preservation under refinement, incorporating witness relations, reproduced from Fig. 8 are:

PliEv/FISRW

$$\begin{aligned}
&I(u(\mathbb{t}_\mathrm{L})) \wedge K(u(\mathbb{t}_\mathrm{L}), w(\mathbb{t}_\mathrm{L})) \wedge iv_{PliEvC}(w(\mathbb{t}_\mathrm{L})) \wedge grd_{PliEvC}(w(\mathbb{t}_\mathrm{L})) \;\Rightarrow \\
&\quad \big(\exists \mathbb{t}_\mathrm{R} > \mathbb{t}_\mathrm{L} \bullet \mathrm{TRM}(\mathbb{t}_\mathrm{R}) \wedge (\forall \mathbb{t}_\mathrm{L} < t < \mathbb{t}_\mathrm{R}, w(t), j?(t), k(t), p!(t) \bullet \\
&\quad BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t)) \\
&\qquad \Rightarrow (\forall \mathbb{t}_\mathrm{L} < t < \mathbb{t}_\mathrm{R} \bullet (\exists u(t), i?(t), l(t), o!(t) \bullet W(u(t), i?(t), l(t), o!(t), w(t), j?(t), k(t), p!(t)))) \big)
\end{aligned} \tag{27}$$

PliEv/INVRW

$$\begin{aligned}
&I(u(\mathbb{t}_\mathrm{L})) \wedge K(u(\mathbb{t}_\mathrm{L}), w(\mathbb{t}_\mathrm{L})) \wedge iv_{PliEvC}(w(\mathbb{t}_\mathrm{L})) \wedge grd_{PliEvC}(w(\mathbb{t}_\mathrm{L})) \;\Rightarrow \\
&\quad \big(\exists \mathbb{t}_\mathrm{R} > \mathbb{t}_\mathrm{L} \bullet \mathrm{TRM}(\mathbb{t}_\mathrm{R}) \wedge (\forall \mathbb{t}_\mathrm{L} < t < \mathbb{t}_\mathrm{R}, w(t), j?(t), k(t), p!(t) \bullet \\
&\quad BDApred_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge SOL_{PliEvC}(w(t), j?(t), k(t), p!(t), t) \wedge \\
&\quad W(u(t), i?(t), l(t), o!(t), w(t), j?(t), k(t), p!(t))) \\
&\qquad \Rightarrow (\forall \mathbb{t}_\mathrm{L} < t < \mathbb{t}_\mathrm{R} \bullet \\
&\qquad\quad BDApred_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge SOL_{PliEvA}(u(t), i?(t), l(t), o!(t), t) \wedge \\
&\qquad\quad K(u(t), w(t))) \big)
\end{aligned} \tag{28}$$

For (27), it is clear that if we already have PliEv/FISR (which is presumed for the conventional purpose for which such witnesses are used in Event-B), then we can assume the hypotheses of both implications, after which the PO reduces to showing that W, assumed provided already, (and depending, of course, on the pliant abstract and concrete event pair in question), holds over the required interval. Given the structure of (27), this amounts to checking that the trajectory defined by $BDApred_{PliEvC} \wedge SOL_{PliEvC}$ remains within the range of W (when W is regarded as a relation from abstract to concrete spaces). This resembles an invariant preservation check for the $BDApred_{PliEvC} \wedge SOL_{PliEvC}$ trajectory, except that it can depend on time, and more variables (abstract and concrete ones) are involved. Given that perspective, we regard verification of (27) as subsumed by our earlier work.

If we know that (27) holds, then (28) can be viewed as another analogue of a time dependent invariant preservation check, except that this time W plays the role of trajectory, and $BDApred_{PliEvA} \wedge SOL_{PliEvA} \wedge K$ is the time dependent invariant.

We note that the structure of (28) resembles that of the propositions in Section 22, even if the visual appearance of the two things does not obviously suggest it. In the propositions of Section 22, various properties are hypothesised among the case analyses of the listed propositions. Pulling apart all these case analyses to generate individual results, would reveal implications with the structure of (28), with the various properties mentioned, playing the role of candidate W. We can thus conclude that, since the the properties mentioned are sufficient to prove the propositions, they provide cases that are sufficient to instantiate W (i.e., they are 'witnesses for W witnesses'). The converse does not hold. In (28), W is defined as a generic sufficient condition. In particular, there is no need for W to specify a regular relation. So there is no necessity for an arbitrary W, that is sufficient for (28), to fall into one of the cases in the propositions of Section 22.

The above not withstanding, we note that the above covers *only* witnesses for the joint invariant of a claimed refinement, and the discharge of the existential quantifiers involved. But proof in Hybrid Event-B involves many instances of existential quantification. We observe that in all of those cases, a similar approach could be taken, introducing a 'W-like witness' in the hypotheses of the relevant implication, as happens in (28). This observation has particular force for Hybrid Event-B, much more so than for discrete Event-B, since existential witnesses that have non-empty duration (as opposed to witnesses that just assert the existence of a single value) are typically needed. Were this observation to be pursued more seriously, the approach just discussed for joint invariant witnesses could be replicated *mutatis mutandis*.

Part 9 — Case Study, Boundary Cases, Related Work, Conclusions

24. Revisiting the Example

In this section we revisit the small case study of Section 3 (Figs. 3-5) from the perspective of the preceding material. Since most of the case study uses mode events, and their POs are dealt with in the usual calculational way, we can be relatively brief.

One thing missing from Section 3 is mention of any CONTEXT. In a real development this would be needed to list the attributes of all the constants and similar identifiers that appear in Figs. 3-5.

24.1. *CruiseControl*_0

If we look at machine *CruiseControl*_0, we see that *INITIALISATION* will trivially satisfy Init/FIS and Init/INV. Beyond that, the only pliant event, *PliTrue*, merely states that the invariants are to be complied with (in some unspecified manner), which creates no work for a verifier, so disposes of PliEv/FIS and PliEv/INV. The mode events either skip and are thus trivial, or simply swap one choice of an element from the finite set $\{OFF, ON\}$ for the other, again trivial for a verifier, so we can say that the MoEv/FIS and MoEv/INV instances are easily discharged.

This leaves MoPli/WFor and PliMo/WFor. We refer to Section 7. For MoPli/WFor, since *all* the mode events are asynchronous, whenever one of them completes, the enabledness of any of them is irrelevant, and *PliTrue* is always enabled anyway, thus satisfying the PO. For PliMo/WFor, since there are no eager mode events, $\mathbb{t}_{\mathrm{R}\,\mathbf{eager}}$ in (3) is infinite, and so $\mathbb{T}_{\mathrm{R}\,\mathbf{lazy}}$ is all of time after $\mathbb{t}_{\mathrm{L}}$. Therefore, since *PliTrue* can always execute for as long as needed, a $\mathbb{t}_{\mathrm{R}}$ value chosen to be any time strictly after $\mathbb{t}_{\mathrm{L}}$ will do, and will satisfy $\mathsf{MAXIMAL}(\mathbb{t}_{\mathrm{R}})$. The non-satisfaction of guard disjunction prior to $\mathbb{t}_{\mathrm{R}}$ in the PO hypotheses is irrelevant because lazy mode events do not contribute to the disjunction, and, once time reaches the chosen $\mathbb{t}_{\mathrm{R}}$, the guard disjunction in the PO conclusions ranges over all mode events, and is thus trivially satisfied. So the PO holds.

24.2. *CruiseControl*_1

Machine *CruiseControl*_1 starts by mentioning some real constants: $0, V_{\max}, VCC_{\min}, VCC_{\max}$. As mentioned, these, their types, and the relationships between them, e.g. $0 < VCC_{\min} < VCC_{\max} < V_{\max}$, would be held in a suitable CONTEXT. The same applies to all the other constants needed.

24.2.1. *The CruiseControl*_1 *Dynamics*

Before turning to the POs and their verification, we review the dynamics informally, adding more detail to the brief account in Section 3. The dynamics starts in the *OFF* mode, with the velocity variables set arbitrarily within their ranges. *PliDefault* maintains this state of affairs until v satisfies the stronger $[VCC_{\min} \ldots VCC_{\max}]$ constraint, at which point *SwOn* can execute, setting the mode to *ON*, and setting *setv* to the current value of v. The *Cruise* event then runs, allowing v to vary smoothly (but not too fast, since $|\mathcal{D}\,v|$ is bounded), while maintaining the $|v - setv| \leq \Delta_{\mathrm{Cruise}}$ constraint. Provided $\Delta_{\mathrm{Cruise}} \leq \min\{VCC_{\min}, V_{\max} - VCC_{\max}\}$ (which can be specified in the CONTEXT), this also maintains all the invariants, including $\mathsf{CONTINUOUS}(v)$ (which follows directly from Proposition 14.1.1).

Since the mode is *ON*, from time to time *TipUp* and *TipDown* can execute, after which, *Cruise* must resume (since it is the only pliant event that is enabled when the mode is *ON*). *TipUp* and *TipDown* have been designed so that their assignments to *setv* maintain the $[VCC_{\min} \ldots VCC_{\max}]$ invariant on *setv*. However there is no guarantee that these assignments maintain the $|v - setv| \leq \Delta_{\mathrm{Cruise}}$ constraint of *Cruise* (unless $VCC_{\max} - VCC_{\min}$ itself is less than Δ_{Cruise}). This is required because, as discussed in Section 13.3, according to the semantics of PaperI, when *Cruise* executes, the behaviour it imposes on v must be right continuous at $\mathbb{t}_{\mathrm{L}}$ and must satisfy the *BDApred* constraint. In other words $|v(\mathbb{t}_{\mathrm{L}}) - setv(\mathbb{t}_{\mathrm{L}})| \leq \Delta_{\mathrm{Cruise}}$ must hold, and the immediately preceding mode event must ensure this. If not, the execution ABORTs.

Of course, it is easy enough to avoid the ABORT possibility, and at least four options suggest themselves. Option 1: We can alter the assigned values in *TipUp* and *TipDown* to $\min\{VCC_{\max}, setv + TUD, v + \Delta_{\mathrm{Cruise}}\}$ and $\max\{VCC_{\min}, setv - TUD, v - \Delta_{\mathrm{Cruise}}\}$ respectively. Option 2: We can augment the guards in both *TipUp* and *TipDown* to add $|setv - v| + TUD \leq \Delta_{\mathrm{Cruise}}$, which ensures they only execute when it is safe to do so. Option 3: We can try to incorporate $|setv - v| \leq \Delta_{\mathrm{Cruise}}$ into an additional invariant. Since the value of *setv* is irrelevant in the *OFF* mode, the most suitable such invariant would be $mode = ON \Rightarrow |setv - v| \leq \Delta_{\mathrm{Cruise}}$. However, while making the need for the additional property explicit, this alone would not fix things without altering the mode events according to either of the preceding options, since it is what they do that causes the property to fail. Option 4: We can make *Cruise* itself more forgiving, e.g. by suitably incorporating in the *BDApred* clause a condition such as $|v - setv| > \Delta_{\mathrm{Cruise}} \Rightarrow \mathcal{D}\,v = \mathrm{sgn}(setv - v)\,\Delta_{\mathrm{MCA}}$.[18]

[18]It is to be noted that such a condition in *BDApred* falls outside the limitations we have imposed on *BDApred* clauses in this paper, despite the fact that it does the job required. The situation is easily remedied by splitting the *Cruise* event into two: e.g., *CruiseNormal* and *CruiseConverge*. The former would be like the existing *Cruise*, but with an additional guard $|v - setv| \leq \Delta_{\mathrm{Cruise}}$. The latter would have guard $|v - setv| > \Delta_{\mathrm{Cruise}}$ and would replace the COMPLY *BDApred* clause with SOLVE $\mathcal{D}\,v = \mathrm{sgn}(setv - v)\,\Delta_{\mathrm{MCA}}$. We omit the straightforward details.

Each option has its pros and cons. For Option 1, *TipUp* and *TipDown* might alter the acceleration by less than what the user might be expecting. For Option 2, *TipUp* and *TipDown* might not execute upon user demand, which is almost certain to surprise users. Option 3 has the same effect at user level as the first two, but makes the requirement much clearer in the model. Option 4, with its more forgiving approach to the violation of the stated property, might be seen as offering the best solution from the user perspective. And, once *Cruise* is executing again, the above repeats indefinitely.

*24.2.2. The CruiseControl_*1 *POs, Original Formulation*

We now reexamine the above from the point of view of the POs, looking at the correctness of *CruiseControl_*1, both as a machine in its own right, and —given that it is declared to be a syntactic refinement of *CruiseControl_*0— whether it is also a semantic refinement of *CruiseControl_*0. We start from the original formulation of Figs. 3-5, and discuss the optional enhancements later.

There is no declaration in *CruiseControl_*1 of an explicit joint invariant, but the abstract variables are duplicated there, extended by further concrete variables, so the joint invariant is an inverse projection from the abstract *mode* variable to the concrete $(mode, v, setv)$ tuple: $K \equiv (mode, (mode, v, setv))$.

INITIALISATION easily satisfies Init/FIS and Init/INV, and since it is a simple extension of the *CruiseControl_*0 *INITIALISATION* to additional variables, and the inverse projection K is total and onto, it also trivially satisfies Init/FISR and Init/INVR.

PliDefault is like its abstraction *PliTrue*, hence readily satisfies PliEv/FIS and PliEv/INV. But it has a stronger guard, and so, clearly satisfies PliEv/FISR, PliEv/GRDR and PliEv/INVR too.

There is also the *Cruise* event, a pliant event that gives a different refinement of *PliTrue* when the strengthened guard of *PliDefault* does not hold. As noted above, without modification, we cannot guarantee $|v(\mathbb{t}_{\mathrm{L}}) - setv(\mathbb{t}_{\mathrm{L}})| \leq \Delta_{\mathrm{Cruise}}$ for executions of *Cruise*. This is made visible in the PliEv/FIS PO for *Cruise*:

$$\begin{aligned}
&\underline{v(\mathbb{t}_{\mathrm{L}}) \in [0 \ldots V_{\max}]} \wedge \mathsf{CONTINUOUS}(v(\mathbb{t}_{\mathrm{L}})) \wedge mode(\mathbb{t}_{\mathrm{L}}) \in \{OFF, ON\} \wedge \\
&\underline{setv(\mathbb{t}_{\mathrm{L}}) \in [VCC_{\min} \ldots VCC_{\max}]} \wedge mode(\mathbb{t}_{\mathrm{L}}) = ON \Rightarrow \\
&\quad (\exists \mathbb{t}_{\mathrm{R}} > \mathbb{t}_{\mathrm{L}} \bullet (\forall \mathbb{t}_{\mathrm{L}} \leq t < \mathbb{t}_{\mathrm{R}} \bullet (\exists v(t), setv(t), mode(t) \bullet \\
&\qquad \underline{|v(t) - setv(t)| \leq \Delta_{\mathrm{Cruise}}} \wedge |\mathcal{D}\, v(t)| \leq \Delta_{\mathrm{MCA}})))
\end{aligned} \tag{29}$$

which is not provable, where we have underlined the elements that cause the proof failure. Fortunately, this failure to prove follows immediately from the values at $\mathbb{t}_{\mathrm{L}}$ alone. It thus holds regardless of whether we interpret the semantics as originally formulated in paperI (i.e., using *PACcl* functions) or we restrict to just *FPHS* functions, as we would have to if we were intending to exploit the propositions earlier in the paper. Indeed, if we wished to use Proposition 16.1 for feasibility, we would need to verify the $\mathbb{t}_{\mathrm{L}}$ consistency check $(I \wedge iv \wedge grd \Rightarrow iv_{B/DA})$, and this would fail to prove because the same underlined elements occur in it.[19]

Regarding PliEv/INV, all the terms appearing in (29) occur in the hypotheses, which are therefore inconsistent. Since falsity proves anything, on a purely formal level PliEv/INV discharges, but the inconsistency of its hypotheses raises alarm in a model based formal development, since it leads to the conclusion of some specific properties (*CruiseControl_*1's invariants) for an impossible element of the model. On this basis, we can say that PliEv/INV fails.

[19]However, this would not be the sole reason Proposition 16.1 would not succeed. In the COMPLY clause of *Cruise*, we also have $|\mathcal{D}\, v(t)| \leq \Delta_{\mathrm{MCA}}$, and Proposition 16.1 says nothing about conditions on derivatives, so would not be applicable. Of course, it would be easy enough to enhance the provisions of Proposition 16.1 to include CAD constraints on derivatives, since only existence in a right-open interval is needed for feasibility. This is a good illustration of the open ended and extensible nature of the results presented in this paper — it is always relatively easy to think of sound extensions that are not explicitly catered for in the paper.

The same story repeats for PliEv/FISR and PliEv/INVR, since the only difference in these POs is the appearance of K among the hypotheses, which is trivially satisfied, since as stated before, K is total and onto. Aside from that, PliEv/GRDR is provable for *Cruise* since it just refers to strengthened guard values at $\mathbb{t}_{\mathrm{L}}$.

Regarding the mode events, *SwOff* is identical to its abstraction, so POs MoEv/FIS, MoEv/INV, MoEv/FISR and MoEv/INVR are trivially discharged. And *SwOn* is scarcely more complicated, adding a guard, and an assignment to a fresh variable, so the four POs are easily discharged for it too. *TipUp* and *TipDown* follow the same pattern. Their guards are unchanged, and the assignment to a fresh variable causes no difficulty, so their four POs are easily discharged.

This leaves the POs that control the structure of executions, MoPli/WFor, PliMo/WFor, and the POs that synchronise across refinements, MoEv/RelDLF, PliEv/RelDLF. For the first two POs, MoPli/WFor and PliMo/WFor, some of the reasoning for *CruiseControl*_0 applies once more. Thus, for MoPli/WFor, since all the mode events are asynchronous, whenever one of them completes, the enabledness of any of them is irrelevant, and then, whether *PliDefault* runs or *Cruise* runs depends solely on the after-value of the *mode* variable. If the after-value is *OFF*, then *PliDefault* runs, and the argument for the discharge of the PO is as for *CruiseControl*_0, since the value of the *mode* variable is the only fact relevant to the *PliDefault* guard. If the after-value is *ON*, then *Cruise* runs, and the argument is similar, since the value of the *mode* variable is again the only nontrivial point.[20]

For PliMo/WFor, the argument for *PliTrue* from *CruiseControl*_0 can be applied vebatim to both *PliDefault* and *Cruise*. Additionally, since we have the $\mathsf{CONTINUOUS}(v)$ invariant, the detail of concern in Section 7 for the term $\mathsf{WELLDEF}(\mathbb{t}_{\mathrm{R}})$ also holds without issue, so the PO is easily dealt with. We turn to MoEv/RelDLF, PliEv/RelDLF. For both of these, in the original formulation, it is easy to see that at both abstract and concrete levels, the requisite disjunctions of guards evaluate to true, so the POs are trivially discharged.[21]

Taking stock at this juncture, we see that regardless how the event consistency condition is handled, verification detects the flaw in the *CruiseControl*_1 machine,

*24.2.3. The CruiseControl*_1 *POs, Optional Enhancements*

In this section we briefly examine how the preceding account changes in the light of the enhancements to the model suggested at the end of Section 24.2.1. We do not repeat all the detail that stays the same, focusing on the new phenomena that arise, and taking the options in turn.

For Option 1, *TipUp* and *TipDown* have merely had their assigned values altered, so their POs will go through as before. *Cruise* is insensitive to the changed mode events, so the previous account continues to hold unchanged regarding its various POs. For MoPli/WFor and PliMoWFor, and for MoEv/RelDLF and PliEv/RelDLF, the story is as before since no guards have changed, and these four POs are only sensitive to guards.

For Option 2, *TipUp* and *TipDown* have had their guards strengthened, so their POs will go through as before. The story for *Cruise* is again as previously. For MoPli/WFor, the strengthened guards in *TipUp* and *TipDown* will make the PO proof go through, since it did so before. For PliMo/WFor, the argument is unchanged, as in the previous case. PliEv/RelDLF remains unaffected, and, for MoEv/RelDLF, although the guard strengthening would lead us to expect the PO to fail to prove (because the stronger guards are in the PO conclusions), we see that *SwOff* remains enabled even when *TipUp* and *TipDown* are not, and

[20]We note though, that had we taken the route of conjoining $iv_{B/DA}$ to the guard of *Cruise* to enforce consistency, as discussed in Section 15, then this PO would have failed to prove since assuming the invariants in the hypothesis of the PO would not have been sufficient to prove the stronger $|v(t) - setv(t)| \leq \Delta_{\mathrm{Cruise}}$ — which just replays the arguments above in a different context.

[21]We note, as for MoPli/WFor, that had we strengthened the *Cruise* guard with $iv_{B/DA}$, the proof would have failed as the abstract guard disjunction true would not have implied the stronger concrete one.

so the PO succeeds! Obviously, a more sensitive relative deadlock freeness PO could pick up this detail.

Option 3 is, in practice, largely like the first two, the precise effects depending on which of options 1 or 2 accompanies the adoption of the additional invariant.

Option 4 changes the $iv_{B/DA}$ of *Cruise* to true, so all issues connected with pliant event consistency become trivial.

24.3. CruiseControl_2 POs, Original Formulation

Machine *CruiseControl_2* has similar dynamics to machine *CruiseControl_1*. Only the *Cruise* event has been altered, and has a more precisely defined and deterministic dynamics — in this case it can be solved for completely (in terms of Exp if the external reasoner is *Mathematica*). So the joint invariant between the two machines is an identity. We subscript identically named events to distinguish the *CruiseControl_1* version from the *CruiseControl_2* version, e.g. $Cruise_1$ and $Cruise_2$.

Focusing on the original formulation, all the mode events are identical in the two models, and the identity joint invariant makes their refinement trivial. We note that the pliant event consistency problem has disappeared from machine *CruiseControl_2*, as the DE in *Cruise* has a well defined, holomorphic, right continuous solution from any initial value, and thus does not demand any additional initial value constraint.

We check the POs that need examination. For *Cruise*, PliEv/FIS holds by an easy application of Proposition 16.2. For PliEv/INV, we refer to Proposition 18.4. Clauses 1 and 2 hold trivially, For clause 3 we can identify both the required CAD.PI and its CAD.IPI subset with the interval $[0 \dots V_{\max}]$. Then, clause 4 holds since $v \in [0 \dots V_{\max}]$ is an invariant, and we assume the invariants in the hypotheses of the PO. Finally, noting that the interval $[VCC_{\min} \dots VCC_{\max}]$, which constrains *setv*, is strictly inside $[0 \dots V_{\max}]$, we see that the vector field in the RHS of the DE $\mathcal{D}\, v = -C(v - setv)$ points into the interior of the interval $[0 \dots V_{\max}]$ for all values of *setv* that satisfy the invariants, and for all values of v that satisfy the invariants and lie on the boundary of $[0 \dots V_{\max}]$. Therefore, we can use clause 5.(a) to complete the discharge of the PO. And since there is no stronger constraint such as $|v - setv| \leq \Delta_{\text{Cruise}} \wedge |\mathcal{D}\, v| \leq \Delta_{\text{MCA}}$ to worry about, the PliMo/WFor and MoPli/WFor POs go through without difficulty. So, it is evident that in its own terms, *CruiseControl_2* is unproblematic.

However, regarding a refinement from *CruiseControl_1* to *CruiseControl_2*, it is a different matter, as the previous difficulties reappear due to the involvement of *CruiseControl_1*. We need to reexamine PliEv/FISR, PliEv/GRDR, PliEv/INVR, MoEv/RelDLF, PliEv/RelDLF.

PO PliEv/FISR proves successfully since the PO only asks for an execution of $Cruise_2$, not of $Cruise_1$, and the joint invariant is an identity. PO PliEv/GRDR proves successfully for $Cruise_1$ and $Cruise_2$ since the PO just compares guards, which are identical. PO PliEv/INVR fails to prove for $Cruise_1$ and $Cruise_2$ because, outside the $|v(\mathbb{t}_{\text{L}}) - setv(\mathbb{t}_{\text{L}})| \leq \Delta_{\text{Cruise}}$ region, $Cruise_1$ does not have an execution that satisfies the joint invariant (which demands equality between the two machines' values of v).

For MoEv/RelDLF, we have success as before, since the simple guard of *SwOff*, unchanged between the two models, subsumes the guards of *TipUp* and *TipDown*. Finally, for PliEv/RelDLF, we also have success, since the abstract guards are at least as strong as the concrete ones.

24.3.1. The CruiseControl_2 POs, Optional Enhancements

For the optional modifications, options 1 and 2 are unremarkable, and go as in the preceding paragraphs. More interesting is option 3 enhanced by the option 1 modifications to *TipUp* and *TipDown*. This will be provable as a refinement, as the strengthened *CruiseControl_1* invariants now curtail the reach of the PliEv/FISR, PliEv/GRDR, PliEv/INVR PO conclusions for $Cruise_1$ and $Cruise_2$ so that they are not demanded outside the $|v(\mathbb{t}_{\text{L}}) - setv(\mathbb{t}_{\text{L}})| \leq \Delta_{\text{Cruise}}$ region.

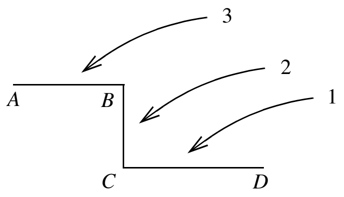

Figure 13: A step, with some trajectories of an incident ball.

Also of interest is option 4. As stated, this proposed refinement will fail, since the $|v - setv| > \Delta_{\text{Cruise}} \Rightarrow \mathcal{D}\,v = \text{sgn}(setv - v)\,\Delta_{\text{MCA}}$ clause in *Cruise*$_1$ stipulates a constant $\mathcal{D}\,v$, whereas the corresponding DE of *CruiseControl_2* cannot satisfy that property. Still, we can contemplate a modification such as $|v - setv| > \Delta_{\text{Cruise}} \Rightarrow \mathcal{D}\,v \geq \text{sgn}(setv - v)\,\Delta_{\text{MCA}}$, which would work better. The modification would evidently be better still if there were also an upper bound on the magnitude of $\mathcal{D}\,v$. This could be done, but would require an upper bound on the magnitude of v. This could be achieved via *TipUp* and *TipDown* and would need to be captured in an additional invariant or included as a result in the THEOREM section of the machine, as it is a global property.

The above completes our reprise of the case study. Although the models contained relatively few pliant events, we saw clearly how the issues they raised impacted mode events too.

25. Dealing with Non-Engineering Cases

Fig. 13 shows an ideal rectangular step A, B, C, D. We can imagine an ideal pointlike ball, moving under gravity, thrown against it along a family of trajectories (indexed using a parameter α) like the three shown in the figure. For definiteness, the value of α can be the distance from D along a straight line running through D and B, with points above D positive, and the three trajectories shown having parameters $\alpha_1 < \alpha_2 < \alpha_3$ respectively. We consider a number of scenarios.

In Scenario 1, we assume ideal bounces on impact with a surface, i.e. no loss of energy. Given appropriate initial energy, trajectory 1 first impacts CD, then bounces up, impacts BC, and bounces out along a trajectory like the reverse of trajectory 2. Given lossless dynamics, with appropriate choice of dynamical parameters, the trajectory can be exactly the reverse of trajectory 2. Thus, there is a similar story for trajectory 2, bouncing down and reversing trajectory 1.

As we increase α from α_1, trajectory 1's impact point with CD drifts towards C, and, keeping the two trajectories synchronised, trajectory 2's impact point with BC also drifts towards C. This synchronised process reaches a unique limit at C, which yields an acceptable definition of an ideal bounce from an ideal corner like C.

If we view the dynamical variables, position and velocity, as functions of α, they will be continuous but not smooth at C. Of course, their derivatives can acquire discontinuities there.

In Scenario 2, we introduce a nontrivial coefficient of restitution $c < 1$. Then, no version of trajectory 1 can be the reverse of trajectory 2, and *vice versa*, because of the loss of energy during a bounce. Nevertheless, as C is approached, the initial segments of the two trajectories can be arranged to converge smoothly to the same limiting segment reaching C, after which, the same post-bounce[22] behaviour ensues.

The dynamical variables will be continuous in α, but not smooth at C once more. Some derived quantities, e.g. the total system energy at C, will be discontinuous where they were continuous before.

[22]The bounce at C has to be viewed as a limit of a pair of bounces, as in Scenario 1.

In Scenario 3, we turn our attention to the other corner, B. Things are quite different here. The post-bounce behaviour of trajectory 2 is always downwards and to the right, whereas post-bounce behaviour of trajectory 3 is always upwards and to the left. Thus there is no common limiting behaviour that we can use to define the bounce at B itself, even when the initial segments of trajectories 2 and 3 have a common limit at B, whose components in the direction of the parameter α are moreover smooth. Of course, we can make an *ad hoc* definition at B, but this is not necessarily helpful.

Situations like the three scenarios just described are not uncommon. Examples include Buridan's Principle [83], and the impossibility of implementing various phenomena on circles and spheres smoothly [117]. Many such situations are handled quite effectively these days via non-smooth analysis [40, 41, 42]. The structure of the exceptional cases can also align well with the CAD descriptions of relevant spaces, whether physical spaces or parameter spaces.

The correct understanding of these situations is a valid mathematical challenge. However it is of much less concern from an engineering perspective, in which, any useful system must have behaviour that is stable in an appropriate neighbourhood of its basic definition. From the point of view of Hybrid Event-B, mainly aimed at the design of practical systems, the engineering viewpoint is thus much more relevant than the mathematical one. Still, a relatively idealised level of modelling will invariably throw up such cases from time to time, so the question arises of how to deal with them efficiently. Regarding this, we make some observations.

Firstly, we notice that inconvenient phenomena such as those noted always arise on a manifold of lower dimension than that of the model as a whole. This is further emphasised by our earlier insistence that the CAD domains we consider must be sufficiently open. This implies that the set of values that characterise the problematic cases will be of Lebesgue measure zero in the set of values that characterise the problem domain as a whole. This helps considerably in isolating the problematic cases.

Secondly, it is possible that different sufficiently open regions that abut in the model space, in fact do so along boundaries that characterise such problematic cases. Presuming that transitions between different sufficiently open regions are likely to be managed using mode events, problematic cases may possibly cause violation of condition **[A]** mentioned in the semantic sketch of Section 2, as the limiting values in one region overlap with limiting values in another region.[23]

For both of these reasons, it is easiest to simply exclude the relevant values from the model, once the system implications of the situation have been properly understood. We call the relevant sets of values, the non-engineering cases. In a backwards oriented formal reasoning process, the relevant sets of values will emerge within unprovable statements, or statements that characterise the violation of some semantic condition. This makes their precise identification easier.

Recommendation 25.1 (Non-Engineering Case Elimination). *Once the non-engineering cases have been identified, they should be excluded from the model. For a non-engineering case characterised by a condition involving only static data, the negation of the condition identifying it can be accommodated in the AXIOMS of the model CONTEXT. For a non-engineering case characterised by a condition also involving dynamic data, the negation of the condition identifying it can be accommodated in the AXIOMS of the model's MACHINE.*

26. Related Work

PaperI included an account of related approaches to hybrid systems, and how they are connected (or not) with the B Method in general and with the design of Hybrid Event-B in particular. In this paper we

[23]Of course, this can happen with non-problematic cases too.

progress this line, with a focus on different approaches to mechanising the reasoning needed for hybrid systems in more contemporary work. For longer established approaches see [35, 127, 59].

The KeYmaera formalisation of hybrid and cyber-physical systems (more recently evolved to KeYmaeraX) [106, 105] features a logic based perspective. Thus, specific formal languages such as hybrid programs for the description of systems are developed, and logics such as differential dynamic logic are elaborated in order to formally reason about them. Support from computer algebra tools is central to the practical handling of mathematics beyond pure arithmetic and logic, as here. A large literature based on this approach can be found at [79].

There is a big overlap between the KeYmaera and Hybrid Event-B approaches. After all, they are targeting broadly the same class of systems. Perhaps the chief differences are that differential dynamic logic admits expressing liveness as well as safety properties, and that KeYmaera is program based, whereas the Hybrid Event-B approach is event based. And KeYmaera has its own tool, which Hybrid Event-B currently lacks.

Also related to Hybrid Event-B is the Hybrid Action Systems approach [12, 112]. This approach is based on the weakest precondition semantics of actions, the actions being of two kinds, conventional and differential. These are rather analogous to our mode and pliant events, except that differential actions can proceed only as long as a guard condition holds. The focus on weakest preconditions makes the connection with safety properties such as (global) invariants more indirect, although these can be built in to the individual actions. On the other hand, the same focus puts a stress on refinement, which is handled in the normal weakest precondition manner.

In the Hybrid CSP approach [67, 138, 91, 137], the process approach of CSP is extended with a differential equation term which is executed while a boolean condition remains true (as for the differential actions of Hybrid Action Systems). In [136] it is evident that the same general class of properties and analysis approaches seen in contemporary hybrid systems verification are translated to this world. That this is possible is inevitable, as the mathematics that underpins these approaches is common to all, and is rather far from the details of their surface syntax, which constitutes the major distinguishing feature between them.

Approaches such as these (and many surveyed in [35]) focus on the applications level, and strive to progress the analysis towards actual implementation, as far as proves possible. Most restrict expressivity in various ways in order to gain decidability. Other approaches focus on automating specific engineering criteria, such as stability via the Lyapunov technique, progressing to code, and based on underpinning the reasoning via PVS [69]. In [58] there is an extensive treatment of such work, targetted towards the code level.

Other relevant work starts at the foundational level, building the continuous mathematics needed for applications work from the ground up, axiomatically. By now, there are several such approaches, and in [32] there is an extensive survey of the differences between them and of some of the consequences of those differences. It is noteworthy that different detailed choices at foundational level lead to different kinds of difficulty later on, as larger results, at higher levels of abstraction, are pursued.

Despite the rigour of such approaches, it seems unlikely that they will lead to systems that will appear attractive to use to engineers, at least not without heavy (tool level, software) engineering. This would be needed to evade the foundational level issues that proving within these frameworks routinely throws up. The approach of Hybrid Event-B is based not on real analysis but on complex analysis, given that it legitimises routine engineering calculation, while excluding the pathological cases that real analysis admits, the exclusion of which complicates the purely real theory.[24]

[24]For a simple illustration, the function $f^{\mathbb{R}}(t) = \exp(t^{-2})$ is infinitely smooth, so has derivatives, hence a Taylor series, for all t. But at $t = 0$ the derivatives are all 0, so the Taylor series represents the zero function there, and not $f^{\mathbb{R}}$. The corresponding complex function $f^{\mathbb{C}}(z) = \exp(z^{-2})$ has an essential singularity at $z = 0$, so is not holomorphic in any region containing $z = 0$.

Closer to Hybrid Event-B is a body of work that attempts to use the existing Rodin tool *as is*, in order to prove correctness of hybrid and cyber-physical systems. Among these we can mention [126, 125, 124, 5, 34, 123]. All of these make use of the *Theory Plugin* of Rodin [92, 72], which enables Rodin to be extended by new types, bearing new properties. (A comparison of modelling the same simple system using Event-B and Hybrid Event-B appears in [23].)

A more recent development of this strategy, of using Rodin *as is*, can be found in [50, 51, 49]. In these works, the theory plugin is used to build a theory of ODEs and their solutions at a fairly high level of abstraction, and refinement is used to progress closer to implementation. In [6], some of the theoretical approach of the KeYmaera framework is adapted into the Event-B/Rodin context.

There are two main things to say about these approaches. The first is that they all depend on building up some continuous mathematics from scratch in the theory plugin. Especially in the earlier works, this only allows very simple examples to be treated, due to the large amount of theory that needs to be built before contact with applications is made. Hence these approaches have limited scalability, although the situation is eased a little in the later works, due to the higher level of abstraction aimed for. The second is that they model an evolving function of time (typically the trajectory of a continuous variable) by appending increments of its graph, whose domains are contiguous time intervals. The restrictions imposed by Event-B, and implemented in Rodin, then force time to be a dynamical variable (whose updates are used to define the domains of the contiguous time intervals), and this decants the responsibility of correctly using the time variable onto the user. Unfortunately, this cannot always be relied on, and models featuring causally invalid invariants can easily be written and proved correct.[25] Hybrid Event-B has been designed to avoid both risks (among others).

27. Conclusions

In the preceding sections we journeyed along the road announced in the Introduction, that started with the syntactic form of a Hybrid Event-B machine and derived sufficient conditions for the discharge of the associated proof obligations, focusing on those that were not already covered within the existing (discrete) Event-B and its tools — i.e. the kind of continuously varying quantities that are found in hybrid and cyber-physical systems. The aim for these is to reduce the required verification conditions to the kind of calculations and logical inferences that could be readily performed in a tool, specifically avoiding having to deal with the many interleavings of quantifiers that would have to be faced if the raw mathematical analysis style definitions of the mathematical entities involved were to be faced head on. How painful the latter approach might get, should something more technically demanding be attempted, can be judged from the small scale case study [34].

We first outlined Hybrid Event-B informally, and give a tiny cruise control case study. We also briefly introduced the POs. After that we focused on the foundations more precisely. We introduced the piecewise absolutely continuous functions, noting that they provided a suitable foundation for the Hybrid Event-B semantics of PaperI. We then focused on the subset of these consisting of piecewise holomorphic functions, noting that these had much more robust properties *vis à vis* calculation. In practical use, all instances of the former class that are actually needed are also members of the latter subclass.

After these foundations, we moved on to the reasoning that could underpin a mechanised approach to the discharge of the proof obligations of Hybrid Event-B. This occupied many sections, as described

In any region excluding $z = 0$, $f^{\mathbb{C}}$ is holomorphic and its Taylor series represents it faithfully. In fact, a significant proportion of 'pathological', or 'peculiar' examples in real analysis are constructed by using pieces of holomorphic functions, but continuing them to points at which holomorphy fails.

[25]For example: 'it is 3 o'clock $\Rightarrow$ property P holds', where the progress of time is (of course) inexorable, but the occurrence of *MaybeEv*, needed to establish P, is not guaranteed in the context of the model.

in the Introduction. The main approach throughout all of these was to focus on instances of the issue at hand that would yield to a general approach founded on familiar results from analysis (specialised, where appropriate, to the complex case), and thereby to reduce the issue in question to the calculational checks that would confirm that that specialisation was applicable. We concluded by revisiting the earlier tiny case study to illustrate how the issues it threw up were dealt with using the approach to verification developed in the bulk of the paper.

The present paper can be viewed as a strategic template for a semi-automated verification system for Hybrid Event-B. The intention was to provide a framework that, as well as providing a basic structure that could be implemented, was also broad enough to be extensible, should the need arise. Such need might indeed arise, if it were driven strongly enough from the application sphere. Thus it is important to emphasise that the account we gave should not be viewed as closing the book on reasoning about continuously varying quantities in Hybrid Event-B. In our drive to construct a mechanisation framework, many generalisations of what we specifically covered were mentioned in passing, and could be pursued in more detail. Moreover, the fields of pure and applied mathematics, not to mention engineering in general and control engineering in particular, have generated an enormous literature over the centuries, of which we have just scratched the surface. Many elements of this could be incorporated into a mechanised reasoning framework for Hybrid Event-B, should there be sufficient incentive to do so, as hinted at in our related work section.

In this paper, we rarely descended to a sufficiently low level to allow direct coding of such a system. Still, the author hopes that what has been done will provide sufficient guidance for a viable implementation. These days, it is recognised that to get a wide variety of reasoning capabilities into a single tool, multiple reasoners need to be combined. Tools such as Why3 [133, 54] have been designed to provide a framework to permit such interoperability effectively. Thus, as well as following the backward reasoning approach that has implicitly underpinned the approach of this paper, it is envisaged that other reasoners with specific capabilities would be brought to bear to more effectively realise the goals pursued here.

Acknowledgements

I would like to thank Shengchao Qin for comments on earlier drafts of this paper. I would also like to thank an anonymous reviewer for suggestions that resulted in worthwhile improvements to the paper.

References

[1] J.R. Abrial, Event-B: Structure and Laws, in: Rodin Project Deliverable D7: Event-B Language. http://rodin.cs.ncl.ac.uk/deliverables/D7.pdf.

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

[3] J.R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.

[4] J.R. Abrial, M. Butler, S. Hallerstede, T. Hoang, F. Mehta, L. Voisin, Rodin: An Open Toolset for Modelling and Reasoning in Event-B, Int. J. Soft. Tools for Tech. Trans. 12 (2010) 447–466.

[5] J.R. Abrial, W. Su, H. Zhu, Formalizing Hybrid Systems with Event-B, in: Derrick, Fitzgerald, Gnesi, Khurshid, Leuschel, Reeves, Riccobene (Eds.), Proc. ABZ-12, volume 7316, Springer, LNCS, 2012, pp. 178–193.

[6] M. Afendi, R. Laleau, A. Mammar, Modelling Hybrid Programs with Event-B, in: Raschke, Mery, Houdek (Eds.), Proc. ABZ-20, volume 12071, Springer, LNCS, 2020, pp. 139–154.

[7] N. Ahmed, Dynamic Systems and Control With Applications, World Scientific, 2006.

[8] B. Akbarpour, L. Paulson, MetiTarski: An Automatic Prover for the Elementary Functions, in: Proc. Calculemus-08, volume 5144, Springer, LNCS, 2008, pp. 217–231.

[9] P. Antsaklis, A. Michel, Linear Systems, Birkhauser, 2006.

[10] J. Aplevich, Implicit Linear Systems, Springer, 1991.

[11] K. Astrom, T. Hagglund, Advanced PID Control, ISA, 2006.

[12] R.J. Back, L. Petre, I. Porres, Continuous Action Systems as a Model for Hybrid Systems, Nordic J. Comp. 8 (2001) 2–21. Extended version of FTRTFT-00, LNCS 1926, 202-213.

[13] R. Banach, Regular Relations and Bicartesian Squares, Theor. Comp. Sci. 129 (1994) 187–192.

[14] R. Banach, On Regularity in Software Design, Sci. Comp. Prog. 24 (1995) 221–248.

[15] R. Banach, Pliant Modalities in Hybrid Event-B, in: Liu, Woodcock, Zhu (Eds.), Proc. Jifeng He Festschrift 2013, volume 8051 of *LNCS*, Springer, 2013, pp. 37–53.

[16] R. Banach, The Landing Gear Case Study in Hybrid Event-B, in: Ait Ameur, Schewe (Eds.), Proc. ABZ-14: Landing Gear Case Study, volume 433 of *CCIS*, Springer, 2014, pp. 126–141.

[17] R. Banach, Formal Refinement and Partitioning of a Fuel Pump System for Small Aircraft in Hybrid Event-B, in: Bonsangue, Deng (Eds.), Proc. IEEE TASE-16, IEEE, 2016, pp. 65–72.

[18] R. Banach, Hemodialysis Machine in Hybrid Event-B, in: Butler, Schewe, Mashkoor, Biro (Eds.), Proc. ABZ-16, volume 9675, Springer, LNCS, 2016, pp. 376–393.

[19] R. Banach, The Landing Gear System in Multi-Machine Hybrid Event-B, Int. J. Soft. Tools for Tech. Trans. 19 (2017) 205–228.

[20] R. Banach, J. Baugh, A Simple Hybrid Event-B Model of an Active Control System for Earthquake Protection, in: Adamatzky, Kendon (Eds.), Proc. Susan Stepney Festschrift 2019, volume 35 of *Emergence, Complexity, Computation*, Springer, 2019, pp. 157–194.

[21] R. Banach, M. Butler, A Hybrid Event-B Study of Lane Centering, in: Aiguier, Boulanger, Krob, Marchal (Eds.), Proc. CSDM-13, Springer, 2013, pp. 97–111.

[22] R. Banach, M. Butler, Cruise Control in Hybrid Event-B, in: Z. Liu, Woodcock (Ed.), Proc. ICTAC-13, volume 8049 of *LNCS*, Springer, 2013, pp. 76–93.

[23] R. Banach, M. Butler, Modelling Hybrid Systems in Event-B and Hybrid Event-B: A Comparison of Water Tanks, in: Ogata, Lawford, Liu (Eds.), Proc. ICFEM-16, volume 10009, Springer, LNCS, 2016, pp. 90–105.

[24] R. Banach, M. Butler, S. Qin, N. Verma, H. Zhu, Core Hybrid Event-B I: Single Hybrid Event-B Machines, Sci. Comp. Prog. 105 (2015) 92–123.

[25] R. Banach, M. Butler, S. Qin, H. Zhu, Core Hybrid Event-B II: Multiple Cooperating Hybrid Event-B Machines, Sci. Comp. Prog. 139 (2017) 1–35.

[26] R. Banach, W. Su, Cyberphysical Systems: A Behind-the-Scenes Foundational View, in: Mashkoor, Wang, Thalheim (Eds.), Models: Concepts, Theory, Logic, Reasoning, and Semantics (Klaus-Dieter Schewe Festschrift), College Publications, Tribute Series, vol. 34, 2018, pp. 177–201.

[27] R. Banach, P. Van Schaik, E. Verhulst, Simulation and Formal Modelling of Yaw Control in a Drive-by-Wire Application, in: Proc. FedCSIS-15 (IWCPS), pp. 731–742.

[28] H. Barendregt, The Lambda Calculus, Its Syntax and Semantics, Elsevier, 1981.

[29] S. Basu, R. Pollack, M.F. Roy, Algorithms in Real Algebraic Geometry, Springer, 2006.

[30] C. Bender, S. Orszag, Advanced Mathematical Methods for Scientists and Engineers I, Springer, 2010.

[31] P. Billingsley, Probability and Measure, Wiley, 1995. 3rd ed.

[32] S. Boldo, C. Lelay, G. Melquiond, Coquelicot: A User-Friendly Library of Real Analysis for Coq, Math. Comput. Sci. 9 (2015) 41–62. Also HAL: hal-00860648, https://hal.inria.fr/hal-00860648v1.

[33] F. Brickell, R. Clark, Differentiable Manifolds, Van Nostrand Reingold, 1970.

[34] M. Butler, J.R. Abrial, R. Banach, Modelling and Refining Hybrid Systems in Event-B and Rodin, in: From Action System to Distributed Systems: The Refinement Approach, CRC Press, 2016, pp. 29–42.

[35] L. Carloni, R. Passerone, A. Pinto, A. Sangiovanni-Vincentelli, Languages and Tools for Hybrid Systems Design, Foundations and Trends in Electronic Design Automation 1 (2006) 1–193.

[36] B. Caviness, J. Johnson (eds.), Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer, 1998.

[37] C. Chicone, Ordinary Differential Equations with Applications, Springer, 2nd edition, 2006.

[38] A. Church, The Calculi of Lambda Conversion, Princeton University Press, 1941.

[39] A. Church, Introduction to Mathematical Logic, Princeton University Press, 1956.

[40] F. Clarke, Optimization and Nonsmooth Analysis, Society for Industrial Mathematics, 1987.

[41] F. Clarke, Functional Analysis, Calculus of Variations and Optimal Control, Springer, 2013.

[42] F. Clarke, Y. Ledyaev, R. Stern, P. Wolenski, Nonsmooth Analysis and Control Theory, Springer, 1997.

[43] E. Copson, An Introduction to the Theory of Functions of a Complex Variable, Oxford University Press, 1935.

[44] E. Copson, Asymptotic Expansions, Cambridge University Press, 1965.

[45] H. Curry, R. Feys, Combinatory Logic, Vol. I, Elsevier, 1958.

[46] O. Diekmann, S. van Gils, S. Verduyn Lunel, H.O. Walther, Delay Equations: Functional-, Complex-, and Nonlinear Analysis, Springer, 1995.

[47] R. Dingle, Asymptotic Expansions: Their Derivation and Interpretation, Academic, 1973.

[48] R. Dorf, R. Bishop, Modern Control Systems, Pearson, 2010.

[49] G. Dupont, Y. Ait-Ameur, M. Pantel, N. Singh, An Event-B Based Generic Framework for Hybrid Systems Formal Modelling, in: Proc. IFM-20, volume 12546, Springer, LNCS, 2020, pp. 82–102.

[50] G. Dupont, Y. Ait-Ameur, N. Singh, F. Ishikawa, T. Kobayashi, M. Pantel, Embedding Approximation in Event-B: Safe Hybrid System Design Using Proof and Refinement, in: Proc. ICFEM-20, volume 12531, Springer, LNCS, 2020, pp. 251–267.

[51] G. Dupont, Y. Ait-Ameur, N. Singh, M. Pantel, Event-B Hybridation: A Proof and Refinement-based Framework for Modelling Hybrid Systems, ACM Trans. Emb. Com. Sys. 20, Article 35 (2021).

[52] K. Dutton, S. Thompson, B. Barraclough, The Art of Control Engineering, Addison Wesley, 1997.

[53] T. Erneux, Applied Delay Differential Equations, Springer, 2009.

[54] J.C. Filliatre, C. Marche, The Why/Krakatoa/Caduceus Platform for Deductive Program Verification, in: Damm, Hermanns (Eds.), Proc. CAV-07, volume 4590, Springer, LNCS, 2007, pp. 173–177.

[55] K. Fritzsche, H. Grauert, From Holomorphic Functions to Complex Manifolds, Springer, 2002.

[56] J. Gallier, Logic for Computer Science, Wiley, 1987.

[57] T. Gamelin, Complex Analysis, Springer, 2001.

[58] P.L. Garoche, Formal Verification of Control System Software, Princeton University Press, 2019.

[59] E. Geisberger, M. Broy (eds.), Living in a Networked World. Integrated Research Agenda Cyber-Physical Systems (agendaCPS), 2015. http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Projektberichte/acaetch_STUDIE_agendaCPS _eng_WEB.pdf.

[60] M. Gerdts, Optimal Control Problems of ODEs and DAEs, De Gruyter, 2012.

[61] H. Grauert, K. Fritzsche, Several Complex Variables, Springer, 1976.

[62] G. Grimmett, D. Stirzaker, Probability and Random Processes, Oxford University Press, 2001. 3rd ed.

[63] W. Haddad, V. Chellaboina, Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach, Princeton University Press, 2008.

[64] E. Hairer, S. Norsett, G. Wanner, Solving Ordinary Differential Equations I Nonstiff Problems, Springer, 1993.

[65] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II Stiff and Differential-Algebraic Problems, Springer, 1996.

[66] J. Hale, S. Verduyn Lunel, Introduction to Functional Differential Equations, Springer, 1993.

[67] J. He, From CSP to Hybrid Systems, in: Roscoe (Ed.), A Classical Mind, Essays in Honour of C.A.R. Hoare, Prentice-Hall, 1994, pp. 171–189.

[68] J. Heading, An Introduction to Phase-Integral Methods, Dover, 2013.

[69] H. Herencia-Zapana, R. Jobredeaux, S. Owre, et al., PVS Linear Algebra Libraries for Verification of Control of Software Algorithms in C/ACSL, in: Goodloe, Person (Eds.), Proc. NASA NFM-12, volume 7226, Springer, LNCS, 2012, pp. 173–177.

[70] R. Hindley, J. Seldin, Introduction to Combinators and λ-Calculus, Cambridge University Press, 1986.

[71] D. Hinrichsen, A. Pritchard, Mathematical Systems Theory I, Springer, 2005.

[72] T.S. Hoang, L. Voisin, A. Salehi, M. Butler, T. Wilkinson, N. Beauger, Theory Plug-in for Rodin 3.x. https://arxiv.org/abs/1701.08625.

[73] E. Hobson, The Theory of Functions of a Real Variable and the Theory of Fourier's Series, Cambridge University Press, 1907. Later enlarged, Vol. 1 and Vol. 2, reprinted Dover, 1957.

[74] L. Hormander, An Introduction to Complex Analysis in Several Variables, Elsevier, 3rd. edition, 1990.

[75] J. Howie, Complex Analysis, Springer, 2003.

[76] J. Hubbard, B. West, Differential Equations: A Dynamical Systems Approach: Ordinary Differential Equations, Springer, 1991.

[77] J. Hubbard, B. West, Differential Equations: A Dynamical Systems Approach: Higher-Dimensional Systems, Springer, 1995.

[78] A. Ilchmann, T. Reis (Eds.), Surveys in Differential-Algebraic Equations (3 vols.), Springer, 2013, 2015.

[79] KeYmaera. http://symbolaris.com.

[80] S. Krantz, Function Theory of Several Complex Variables, John Wiley, 1982.

[81] P. Kunkel, V. Mehrmann, Differential-Algebraic Equations: Analysis and Numerical Solution, European Mathematical Society, 2006.

[82] R. Lamour, R. Marz, C. Tischendorf, Differential-Algebraic Equations: A Projector Based Analysis, Springer, 2013.

[83] L. Lamport, Buridan's Principle, Foundations of Physics 42 (2012) 1056–1066.

[84] S. Lang, Complex Analysis, Springer, 2008.

[85] A. Laub, Matrix Analysis for Scientists and Engineers, SIAM, 2004.

[86] C. Laurent-Thiébaut, Holomorphic Function Theory in Several Variables, Springer, 2011.

[87] R. Leadbetter, S. Cambanis, V. Pipiras, A Basic Course in Measure and Probability, Cambridge University Press, 2014.

[88] R. Leadbetter, S. Cambanis, V. Pipras, A Basic Course in Measure and Probability, Cambridge University Press, 2014.

[89] J. Lebl, Tasty Bits of Several Complex Variables, lulu.com, 2019.

[90] J. Lee, Introduction to Smooth Manifolds, Springer, 2002.

[91] J. Liu, J. Lv, Z. Quan, H. Zhao, C. Zhou, L. Zou, A Calculus for Hybrid CSP, in: Ueda (Ed.), Proc. APLAS-10, volume 6461, Springer, LNCS, 2010, pp. 1–15.

[92] I. Maamria, M. Butler, A. Edmunds, A. Rezazadeh, On an Extensible Rule-Based Prover for Event-B, in: Frappier, Glasser, Khurshid, Laleau, Reeves (Eds.), Proc. ABZ-10, volume 5977, Springer, LNCS, 2010, p. 407.

[93] Y. Manin, A Course in Mathematical Logic, Springer, 1977.

[94] Maple. http://www.maplesoft.com.

[95] Mathematica. http://www.wolfram.com.

[96] MATLAB and SIMULINK. http://www.mathworks.com.

[97] MetiTarski. https://www.cl.cam.ac.uk/ lp15/papers/Arith/.

[98] A. Mili, An Introduction to Program Fault Tolerance, Prentice-Hall, 1990.

[99] A. Mili, W. Xiao-Yang, Y. Quing, Specification Methodology: An Integrated Relational Approach, Soft. Prac. Exp. 16 (1986) 1003–1030.

[100] B. Mishra, Algorithmic Algebra, Springer, 1993.

[101] M. Mrsevic, Convexity of the Inverse Function, The Teaching of Mathematics (2008) 21–24. http://www.teaching.math.rs/.

[102] K. Ogata, Modern Control Engineering, Pearson, 2008.

[103] F. Olver, D. Lozier, R. Boisvert, C. Clark, NIST Handbook of Mathematical Functions, Cambridge University Press, 2010.

[104] L. Perko, Differential Equations and Dynamical Systems, Springer, 4th edition, 2008.

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

[106] A. Platzer, Logical Foundations of Cyber-Physical Systems, Springer, 2018.

[107] QEPCAD. http://www.usna.edu/CS/qepcadweb/B/QEPCAD.html.

[108] H. Rasiowa, R. Sikorski, The Mathematics of Metamathematics, Polish Scientific Publishers, 1963.

[109] REDLOG. http://www.redlog.eu/.

[110] RODIN Tool. http://www.event-b.org/.

[111] RODIN User's Handbook. http://handbook.event-b.org/current/pdf/rodin-doc.pdf.

[112] M. Rönkö, A. Ravn, K. Sere, Hybrid Action Systems, Theor. Comp. Sci. 290 (2003) 937–973.

[113] H. Royden, P. Fitzpatrick, Real Analysis, Pearson, 2010.

[114] W. Rudin, Principles of Mathematical Analysis, McGraw-Hill, 1976.

[115] W. Rudin, Real and Complex Analysis, McGraw-Hill, 1987. 3rd. ed.

[116] Ryan, M. and Sadler, M., Valuation Systems and Consequence Relations, in: Abramsky, Gabbay, Maibaum (Eds.), Handbook of Logic in Computer Science, Vol. 1, Oxford University Press, 1992.

[117] R. Sanfelice, Hybrid Feedback Control, Princeton University Press, 2021.

[118] Scilab. http://www.scilab.org.

[119] H. Smith, An Introduction to Delay Differential Equations with Applications to the Life Sciences, Springer, 2011.

[120] Smorynski, C., The Incompleteness Theorems, in: Barwise (Ed.), Handbook of Mathematical Logic, Elsevier, 1977.

[121] R. Smullyan, Gödel's Incompleteness Theorems, Oxford University Press, 1992.

[122] E. Sontag, Mathematical Control Theory, Springer, 1998.

[123] W. Su, J.R. Abrial, Aircraft Landing System: Approaches with Event-B to the Modelling of an Industrial System, Int. J. Soft. Tools Tech. Trans. 19 (2017) 141–166.

[124] W. Su, J.R. Abrial, H. Zhu, Complementary Methodologies for Developing Hybrid Systems with Event-B, in: Proc. ICFEM-12, volume 7504, Springer, LNCS, 2012, pp. 230–248.

[125] W. Su, J.R. Abrial, H. Zhu, R. Huang, From Requirements to Development: Methodology and Example, in: Proc. ICFEM-11, volume 6991, Springer, LNCS, 2011, pp. 437–455.

[126] W. Su, F. Yang, X. Wu, J. Gou, H. Zhu, Formal Approaches to Mode Conversion and Positioning for Vehicle Systems, in: Proc. 3rd IEEE International Workshop on Security Aspects of Process and Services Engineering (COMPSAC Workshops), IEEE, 2011, pp. 416–421.

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

[128] E. Titchmarsh, The Theory of Functions, Oxford University Press, 2nd. edition, 1939.

[129] L. Voisin, J.R. Abrial, The Rodin Platform has Turned Ten, in: Ait Ameur, Schewe (Eds.), Proc. ABZ-14, volume 8847, Springer, LNCS, 2014, pp. 1–8.

[130] W. Walter, Ordinary Differential Equations, Springer, 1998.

[131] F. Warner, Foundations of Differentiable Manifolds and Lie Groups, Springer, 1983.

[132] E. Whittaker, G. Watson, A Course of Modern Analysis, Cambridge University Press, 4th. edition, 1927.

[133] Why3. http://why3.lri.fr/.

[134] Wikipedia, Absolute continuity.

[135] V. Yakubovich, G. Leonov, Stability of Stationary Sets in Control Systems with Discontinuous Nonlinearities, World Scientific, 2004.

[136] N. Zhan, S. Wang, H. Zhao, Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach, Springer, 2017.

[137] N. Zhan, S. Wang, H. Zhao, Hybrid CSP, in: Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach, Springer, 2017, pp. 71–90.

[138] C. Zhou, J. Wang, A. Ravn, A Formal Description of Hybrid Systems, in: Alur, Sontag, Henzinger (Eds.), Proc. HS-95, volume 1066, Springer, LNCS, 1995, pp. 511–530.