From Natural Language Requirements to Runtime Monitors for Resource-Constrained Systems: Integrating FRET and R2U2

Alexis Aurandt[1], Christopher Johannsen[2], Andreas Katis[3], Anastasia Mavridou[3], Kristin Yvonne Rozier[2], and Phillip H. Jones[2]

[1] Collins Aerospace, USA Alexis.Aurandt2@collins.com
[2] Iowa State University, USA {cgjohann, kyrozier, phjones}@iastate.edu
[3] KBR, Inc. at NASA Ames Research Center, Moffett Field, CA
{andreas.katis, anastasia.mavridou}@nasa.gov

Abstract. The Realizable, Responsive, Unobtrusive Unit (R2U2) is a stream-based runtime monitoring framework that verifies a system's adherence to a set of formal system requirements with minimal resource overhead, allowing for real-time, online monitoring on resource-constrained systems. Yet, a persisting challenge for deploying runtime monitors is eliciting formal specifications that accurately capture system requirements commonly expressed in ambiguous natural language; therefore, we employ NASA's Formal Requirements Elicitation Tool (FRET) to configure R2U2 monitors from structured natural language requirements. We extend FRET to formalize requirements in Mission-time Linear Temporal Logic (MLTL) - the native specification logic of R2U2, and we provide 157 MLTL rewrite rules that reduce each of FRET's MLTL formalizations by an average of 15 operators, or 36.05%, decreasing the resources necessary to monitor these requirements with R2U2. We also introduce a novel SMT-based proof technique for automatically proving the correctness of these rewrite rules.

Keywords: Mission-time Linear Temporal Logic (MLTL) · Stream-based runtime monitoring · R2U2 · Rewrite Rules · FRET

1 Introduction

Runtime monitors analyze an input data stream's adherence to formally specified requirements. To this extent, runtime monitors can be deployed online such that requirement violations are detected in real time to enable appropriate mitigation

actions [1,8,25], but these online monitors must frequently fit within *tight* timing and memory constraints of the target system [8,21,41]. Therefore, the resource overhead of these online monitors must be minimized to enable deployment on resource-constrained systems. The Realizable, Responsive, Unobtrusive Unit (R2U2) is a stream-based runtime monitoring framework that is known for its minimal resource overhead [35,41,55,57,59], and the success of R2U2 has been exemplified by its deployment on several resource-constrained systems such as NASA's Robonaut2 [41], NASA's Lunar Gateway VSM [16,17,18,19], NASA's Swift UAS [26,55,61], and more [1,12,32,34,46,47,58,60]. Yet, the resource overhead of R2U2 is contingent on the formal system requirements being monitored, hence introducing the challenge of formalizing requirements that are small enough to satisfy the system's resource constraints. Thus, we focus on configuring R2U2 monitors while simultaneously reducing the resource overhead of the specified requirements themselves.

Another persisting challenge for deploying runtime monitors is the elicitation and formalization of specifications that accurately capture system requirements [29,56]. Often, system requirements are written in ambiguous natural language; therefore, the translation to a formal specification language is also ambiguous. To aid in eliciting and formalizing system requirements for R2U2, the WEST tool provides a visualization of Mission-time Linear Temporal Logic (MLTL) formulas (i.e., R2U2's native specification language) [66], and the R2U2 Playground provides an interactive visualization playground for R2U2 monitors [4]. Nevertheless, both of these tools still require the system requirements to be elicited in the formal specification language of MLTL, and formalizing requirements in temporal logic is known to be challenging [30,52].

To address the challenges of writing specifications in formal languages (such as temporal logics), the usage of *structured natural language* has been extensively studied in requirements engineering [5,10,42,45,64]. A structured natural language is a controlled subset of natural language (plain English) in which sentences follow a prescribed semantic structure. This controlled structure preserves readability for humans but limits ambiguity, enabling each statement to be translated systematically into an unambiguous formal representation (e.g. temporal logic formula, phase event automaton, etc.) amenable to formal analysis such as runtime monitoring. The Formal Requirements Elicitation Tool (FRET) is an open-source framework developed by NASA for the elicitation, formalization, and analysis of system requirements [27,28]. FRET enables users to specify system requirements utilizing an unambiguous structured natural language called FRETish, which is automatically translated to temporal semantics. This translation has also been formally verified with the PVS theorem prover [14]. Furthermore, FRET provides visual semantic diagrams and simulation capabilities to aid the user in validating that the formalization accurately captures the system requirement [27] and includes several additional analysis tools such as realizability checking [39,48], test case generation [40], and CoCoSpec requirement export for Simulink model analysis with CoCoSim [27]. FRET has also been previously extended to produce CoPilot runtime monitors [51], but R2U2 uniquely differs from CoPilot in many ways such as R2U2 can monitor both past-time and future-time properties (i.e., CoPilot can only monitor past-time), provides a unique circular queue architecture to minimize resource overhead, and is realized both in C [35] and safe Embedded Rust [2] (i.e., CoPilot only

generates C-based monitors). While there are other tools that allow for elicitation of formal requirements from natural language (e.g., ASSERT™ [15], SpeAR [24], STIMULUS [33], and Req2Spec [50]), we focus on extending FRET due to its popularity, ease-of-use, and active development. Furthermore, recent efforts to *manually* configure R2U2 monitors from FRETish requirements formalized in FRET (e.g., [23] and [62]) indicate a strong desire for FRET to *automatically* configure R2U2 monitors.

To address the challenge of configuring R2U2 monitors for resource-constrained systems, we introduce 157 rewrite rules that reduce each of FRET's MLTL formalizations by an average of 15 operators, or 36.05%, per monitored requirement. This builds upon the previous work of [36], which investigated reducing the memory requirements of R2U2 monitors by defining parametric rewrites for MLTL formulas. While [36] offered hand-written proofs that each rewrite preserved the semantics of the original formula, we introduce a novel SMT-based proof technique for automatically proving semantic equivalence between the original and rewritten MLTL formulas. Through our SMT-based technique, we were able to automatically prove 152 of these 157 rewrite rules, while the other 5 simply timed out due to limitations of SMT.

Our contributions are an open-source extension of FRET[4] that includes **(1)** a translation from FRETish to MLTL (Section 2), **(2)** a formalization of MLTL available in both WEST (Section 2.1) and R2U2 format (Section 2.2), **(3)** 157 MLTL rewrite rules to reduce the resource overhead of monitoring these MLTL requirements (Section 3), **(4)** a new SMT-based proof technique to prove the correctness of these 157 MLTL rewrite rules (Section 3.2), and **(5)** an export feature to automatically configure R2U2 monitors (Section 4.1 and 4.2).

2 FRETish to MLTL

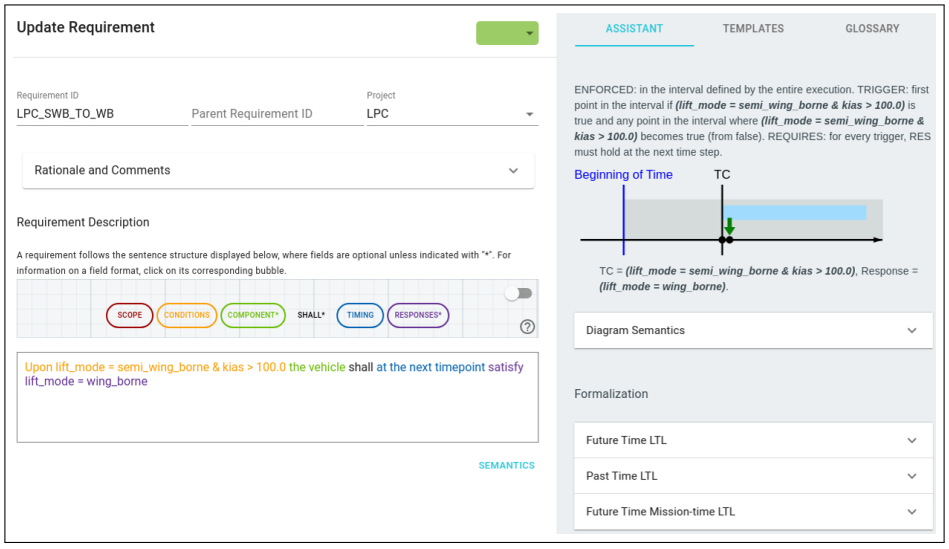

Fig. 1: FRET Edit/Update Requirement Dialog Box

[4] *https://github.com/NASA-SW-VnV/fret/releases/v3.1.0*

Requirements in FRET are composed in a structured natural language called FRETish, which is composed of five fields: `scope`, `condition`, `component`, `timing`, and `response`. Utilizing FRETish, users can specify a requirement for a specific `component` within a certain `scope` of operation under a particular `condition` where the `response` must be satisfied within the defined `timing` constraints. Note that the `scope`, `condition`, and `timing` fields are optional, while the `component` and `response` fields are mandatory. FRETish provides eight possibilities for the `scope` field: *in*/*during mode*, *when*/*if not in mode*, *only in*/*during mode*, *after mode*, *only after mode*, *before mode*, *only before mode*, and none specified (which is equivalent to always) such that *mode* is represented by a Boolean variable. The *mode* can also be represented as a Boolean expression, except in the case of *in*, which must be written with the keyword *while* instead. FRETish also provides three possibilities for the `condition` field: *whenever condition* holds `true`, *upon*/*when*/*where*/*if condition* holds `true` after being `false` (or the *condition* is `true` already at the start of the `scope` interval), and none specified (which is equivalent to always), where *condition* is a Boolean variable/expression. Note the keyword *unless* is also available, which is equivalent to *upon not condition*. Furthermore, FRETish provides twelve possibilities for the `timing` field: *always*, *eventually*, *finally*, *immediately*, *never*, *at the next timepoint*, *after* N time steps, *for* N time steps, *within* N time steps, *before event*, *until event*, and none specified (which is equivalent to *eventually*), where $N \in \mathbb{N}_0$ and *event* is a Boolean variable/expression. Fig. 1 provides an example FRETish requirement LPC_SWB_TO_WB from the Lift Plus Cruise case study [53,54], which specifies that always *upon* the condition of being in semi-wing-borne lift mode and the airspeed (i.e., kias) being greater than 100.0 knots, the vehicle shall *at the next timepoint* be in wing-borne lift mode.

The underlying semantics of FRETish depend on the `scope`, `condition`, and `timing` fields such that there are $8*3*12=288$ total possible combinations of these field types that result in 288 template keys (i.e., a template key is a tuple [`scope`, `condition`, `timing`]).[5] Each of these template keys directly map to Real-Time Graphical Interval Logic (RTGIL) semantics [28], and FRET utilizes the intermediate language called Structured Assertion Language for Temporal logic (SALT) [9] to automatically translate these RTGIL semantics to infinite-trace pure future-time (fmLTL) and pure past-time (pmLTL) metric Linear Temporal Logic [11,28] formulas in nuXmv [13] format.

R2U2 reasons over MLTL and past-time MLTL (ptMLTL) [2,22,43,55] which are bounded variants of LTL and ptLTL, respectively, where each temporal operator has an associated temporal interval constraint that is closed and discrete. In this work, we focus on the future-time logics (i.e., fmLTL and MLTL) as system requirements are more-commonly expressed in future time (e.g., [44,46,65] and even FRETish itself is expressed utilizing mainly future-time natural language).

[5] Recall that the *eventually* and none specified options for the `timing` field are equivalent. Therefore, while we consider all 288 possible template keys to reflect FRET's implementation, there are only $8*3*11=264$ that map to their own distinct semantics.

Definition 1 (fmLTL Syntax). *The syntax of an fmLTL formula φ over a set of atomic propositions $\mathcal{AP}$ is recursively defined as:*

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\psi \mid \psi \wedge \xi \mid \psi \vee \xi \mid \boldsymbol{X}\,\psi \mid \boldsymbol{G}\,\psi \mid \boldsymbol{F}\,\psi \mid \psi\ \boldsymbol{U}\,\xi \mid \psi\ \boldsymbol{V}\,\xi \mid \boldsymbol{G}_I\,\psi \mid \boldsymbol{F}_I\,\psi \mid \psi\ \boldsymbol{U}_I\,\xi \mid \psi\ \boldsymbol{V}_I\,\xi$$

where $p \in \mathcal{AP}$ is an atomic proposition, ψ and ξ are fmLTL formulas, and $I = [lb,ub]$ is a closed interval such that $lb \leq ub$ and $lb,ub \in \mathbb{N}_0$.

Definition 2 (Infinite Trace). *An infinite trace, denoted by π, is an infinite sequence of sets of atomic propositions $\mathcal{AP}$. The i^{th} set is denoted by $\pi(i)$ and contains the atomic propositions that are satisfied at the i^{th} time step. $\pi[lb,ub]$ denotes the trace segment $\pi(lb),\pi(lb+1),...,\pi(ub)$.*

Definition 3 (fmLTL Semantics). *We recursively define $\pi,i \models \varphi$ (infinite trace π starting from time index $i \geq 0$ satisfies, or "models" fmLTL formula φ) as*

- $\pi,i \models \mathsf{true}$
- $\pi,i \models p$ *for* $p \in \mathcal{AP}$ *iff* $p \in \pi(i)$
- $\pi,i \models \neg\psi$ *iff* $\pi,i \not\models \psi$
- $\pi,i \models \psi \wedge \xi$ *iff* $\pi,i \models \psi$ *and* $\pi,i \models \xi$
- $\pi,i \models \boldsymbol{X}\psi$ *iff* $\pi,i+1 \models \psi$
- $\pi,i \models \boldsymbol{G}\ \psi$ *iff* $\forall j$ *where* $j \geq i$ *such that* $\pi,j \models \psi$
- $\pi,i \models \boldsymbol{G}_{[lb,ub]}\ \psi$ *iff* $\forall j \in [i+lb,i+ub]$ *such that* $\pi,j \models \psi$
- $\pi,i \models \psi\ \boldsymbol{U}\,\xi$ *iff* $\exists j$ *such that* $\pi,j \models \xi$ *where* $j \geq i$ *and* $\forall i \leq k < j$, $\pi,k \models \psi$
- $\pi,i \models \psi\ \boldsymbol{U}_{[lb,ub]}\ \xi$ *iff* $\exists j \in [i+lb,i+ub]$ *such that* $\pi,j \models \xi$ *and* $\forall i \leq k < j$, $\pi,k \models \psi$.

Given two fmLTL formulas ψ and ξ, they are semantically equivalent (denoted by $\psi \equiv \xi$) iff $\pi, i \models \psi \leftrightarrow \pi, i \models \xi$ for all infinite traces π and $i \geq 0$. To complete the semantics, $\mathsf{false} \equiv \neg\mathsf{true}$, $\psi \vee \xi \equiv \neg(\neg\psi \wedge \neg\xi)$, $\neg(\psi\ \boldsymbol{U}\ \xi) \equiv (\neg\psi\ \boldsymbol{V}\ \neg\xi)$, $\neg(\psi\ \boldsymbol{U}_I\ \xi) \equiv (\neg\psi\ \boldsymbol{V}_I\ \neg\xi)$, $\neg\boldsymbol{F}\ \psi \equiv \boldsymbol{G}\ \neg\psi$, *and* $\neg\boldsymbol{F}_I\ \psi \equiv \boldsymbol{G}_I\ \neg\psi$. *fmLTL also holds the standard operator equivalences of* $\boldsymbol{F}\ \psi \equiv (\mathsf{true}\ \boldsymbol{U}\ \psi)$, $\boldsymbol{G}\ \psi \equiv (\mathsf{false}\ \boldsymbol{V}\ \psi)$, $\boldsymbol{F}_I\ \psi \equiv (\mathsf{true}\ \boldsymbol{U}_I\ \psi)$, *and* $\boldsymbol{G}_I\ \psi \equiv (\mathsf{false}\ \boldsymbol{V}_I\ \psi)$.

Definition 4 (MLTL Syntax). *The syntax of an MLTL formula φ over a set of atomic propositions $\mathcal{AP}$ is recursively defined as:*

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid p \mid \neg\psi \mid \psi \wedge \xi \mid \psi \vee \xi \mid \boldsymbol{G}_I\ \psi \mid \boldsymbol{F}_I\ \psi \mid \psi\ \boldsymbol{U}_I\ \xi \mid \psi\ \boldsymbol{R}_I\ \xi$$

where $p \in \mathcal{AP}$ is an atomic proposition, ψ and ξ are MLTL formulas, and $I = [lb,ub]$ is a closed interval such that $lb \leq ub$ and $lb,ub \in \mathbb{N}_0$.

Definition 5 (Finite Trace). *A finite trace, denoted by π, is a finite sequence of sets of atomic propositions $\mathcal{AP}$. The i^{th} set is denoted by $\pi(i)$ and contains the atomic propositions that are satisfied at the i^{th} time step. $|\pi|$ denotes the length of π (where $|\pi| < \infty$), and $\pi[lb,ub]$ denotes the trace segment $\pi(lb),\pi(lb+1),...,\pi(ub)$.*

Definition 6 (MLTL Semantics). *We recursively define $\pi,i \models \varphi$ (finite trace π starting from time index $i \geq 0$ satisfies, or "models" MLTL formula φ) as*

- $\pi,i \models \mathsf{true}$
- $\pi,i \models p$ *for* $p \in \mathcal{AP}$ *iff* $p \in \pi(i)$
- $\pi,i \models \neg\psi$ *iff* $\pi,i \not\models \psi$
- $\pi,i \models \psi \wedge \xi$ *iff* $\pi,i \models \psi$ *and* $\pi,i \models \xi$
- $\pi,i \models \boldsymbol{G}_{[lb,ub]}\ \psi$ *iff* $|\pi| \leq i+lb$ *or* $\forall j \in [i+lb,i+ub]$ *such that* $\pi,j \models \psi$

- $\pi,i \models \psi\ \boldsymbol{U}_{[lb,ub]}\ \xi$ *iff* $|\pi| > i+lb$ *and* $\exists j \in [i+lb,i+ub]$ *such that* $\pi,j \models \xi$ *and* $\forall k < j$ *where* $k \in [i+lb,i+ub]$, $\pi,k \models \psi$.

Given two MLTL formulas ψ *and* ξ*, they are semantically equivalent (denoted by* $\psi \equiv \xi$*) if and only if* $\pi,0 \models \psi \leftrightarrow \pi,0 \models \xi$ *for all finite traces* π*. To complete the semantics,* $\mathsf{false} \equiv \neg\mathsf{true}$*,* $\psi \vee \xi \equiv \neg(\neg\psi \wedge \neg\xi)$*,* $\neg(\psi\ \boldsymbol{U}_I\ \xi) \equiv (\neg\psi\ \boldsymbol{R}_I\ \neg\xi)$*, and* $\neg\boldsymbol{F}_I\ \psi \equiv \boldsymbol{G}_I\ \neg\psi$*. MLTL also holds the standard operator equivalences of* $\boldsymbol{F}_I\ \psi \equiv (\mathsf{true}\ \boldsymbol{U}_I\ \psi)$*, and* $\boldsymbol{G}_I\ \psi \equiv (\mathsf{false}\ \boldsymbol{R}_I\ \psi)$.

Definition 7 (Computation Length [22,67]). *Let* $p \in \mathcal{AP}$ *be an atomic proposition and* ψ *and* ξ *be MLTL formulas. The computation length (i.e., the minimum trace length such that no extension of the trace can change the satisfaction of the formula) is recursively defined as follows:*

- $cplen(p) = 1$
- $cplen(\neg\psi) = cplen(\psi)$
- $cplen(\psi \wedge \xi) = max(cplen(\psi), cplen(\xi))$
- $cplen(\boldsymbol{G}_{[lb,ub]}\ \psi) = cplen(\psi)+ub$
- $cplen(\psi\ \boldsymbol{U}_{[lb,ub]}\ \xi) = max(cplen(\psi)-1, cplen(\xi))+ub$

Definition 8 (Mission-time (M) [55]). *All MLTL formulas are mission-time bounded; therefore, let* $M \in \mathbb{N}_0$ *denote the end of mission-time (i.e., the time when the mission ends) such that all trace lengths are restricted by* M *such that* $|\pi| \leq M+1$.

For FRET to configure R2U2 monitors, FRET must produce formalizations in MLTL; hence, we provide a translation from fmLTL to MLTL. fmLTL has *finite*-trace semantics available that capture end-of-trace behavior (e.g., $\mathbf{F}\ \varphi \equiv \neg LAST\ \mathbf{U}\ \varphi$) [28], similar to MLTL. However, R2U2 monitors never consider end-of-trace behavior because R2U2 evaluates $\pi,i \models \varphi$ assuming there will always be an extension to the trace π (and evaluates as soon as sufficient information is available such that $i+lb \leq |\pi| \leq i+cplen(\varphi)$) [2]. Therefore, we translate fmLTL *infinite*-trace semantics (Definition 3) to MLTL's native *finite*-trace semantics (Definition 6).

One significant difference between fmLTL and MLTL is that fmLTL allows for unbounded temporal operators (i.e., temporal operators do not require an associated temporal interval constraint), while MLTL only supports bounded temporal operators. In fmLTL, the unbounded temporal operators signify that the temporal interval is over the entire infinite trace π starting at time index i. Therefore, we translate fmLTL's unbounded temporal operators to MLTL bounded temporal operators with an interval constraint from $[0,M]$ (see Definition 8) as previously defined in [55]. Notably, there is a possibility for semantic loss with this translation when fmLTL's unbounded operators are nested. For example, consider the fmLTL formula $\varphi = \mathbf{F}\ (\mathbf{G}\ \psi)$ which will be translated to $\varphi = \mathbf{F}_{[0,M]}\ (\mathbf{G}_{[0,M]}\ \psi)$ such that $cplen(\varphi) = 2M+1$. The fmLTL formula states that ψ will at some undefined point "stabilize" (i.e., hold indefinitely) along a given trace, whereas the MLTL formula states that ψ will hold for at least M consecutive

fmLTL	MLTL
$\mathbf{X}\psi$	$\mathbf{F}_{[1,1]}$
$\mathbf{G}_{[lb,ub]}\ \psi$	$\mathbf{G}_{[lb,ub]}\ \psi$
$\mathbf{F}_{[lb,ub]}\ \psi$	$\mathbf{F}_{[lb,ub]}\ \psi$
$\psi\ \mathbf{U}_{[0,ub]}\ \xi$	$\psi\ \mathbf{U}_{[0,ub]}\ \xi$
$\psi\ \mathbf{V}_{[0,ub]}\ \xi$	$\psi\ \mathbf{R}_{[0,ub]}\ \xi$
$\mathbf{G}\ \psi$	$\mathbf{G}_{[0,M]}\ \psi$
$\mathbf{F}\ \psi$	$\mathbf{F}_{[0,M]}\ \psi$
$\psi\ \mathbf{U}\ \xi$	$\psi\ \mathbf{U}_{[0,M]}\ \xi$
$\psi\ \mathbf{V}\ \xi$	$\psi\ \mathbf{R}_{[0,M]}\ \xi$

Table 1: fmLTL and MLTL Temporal Operator Equivalents

time steps within the first M time steps of the trace. The MLTL formula is more restrictive and therefore accepts fewer traces than the fmLTL formula; in other words, the translation loses some semantic meaning. Additionally, since the trace length is restricted to $|\pi| \leq M+1$ (Definition 8) but $cplen(\varphi)=2M+1$, it's possible there will be insufficient information for R2U2 to evaluate if $\pi,i \models \varphi$ within the mission. These are limitations of translating from unbounded to bounded operators.

The bounded temporal operators of $\mathbf{F}_I$ and $\mathbf{G}_I$ are equivalent in both fmLTL and MLTL, but $\mathbf{U}_I$ and $\mathbf{V}_I/\mathbf{R}_I$ are semantically distinct such that in fmLTL the satisfaction of $\pi,i \models \psi\ \mathbf{U}_{[lb,ub]}\ \xi$ requires ψ to hold from i to the position before ξ holds in $[i+lb,i+ub]$, while in MLTL, ψ is only required to hold within the interval $[i+lb,i+ub]$ before ξ holds. However, these operators are equivalent when $lb=0$. When FRET produces formalized fmLTL requirements containing $\mathbf{U}_I$ and $\mathbf{V}_I$, they only express $lb=0$; thus, the translation from FRET's bounded fmLTL formalizations to MLTL is direct. Notably, MLTL also discards the next ($\mathbf{X}$) operator since $\mathbf{X}\psi \equiv \mathbf{F}_{[1,1]}\psi$. Table 1 lists all the fmLTL and MLTL temporal operator translations.

2.1 MLTL in WEST Format

WEST is an interactive visualization tool designed to aid developers in validating that their MLTL formula matches the intended system requirement [66]. Similar to R2U2, WEST also does not consider end-of-trace behavior as it only validates an MLTL formula φ over a finite trace π such that $|\pi| = cplen(\varphi)$ [22,67]. Therefore, we provide FRET's MLTL formalizations in WEST format for further requirement validation in WEST. Fig. 2 displays the LPC_SWB_TO_WB requirement in WEST format. Notably, WEST can only reason over atomic propositions and requires all atomic propositions to be in the format "p⟨*number*⟩", and when applicable, M will need to be manually replaced before input into the WEST tool. Fig. 3 exhibits the WEST tool displaying an example satisfying trace for the LPC_SWB_TO_WB requirement from Fig. 2 with M set to 5.

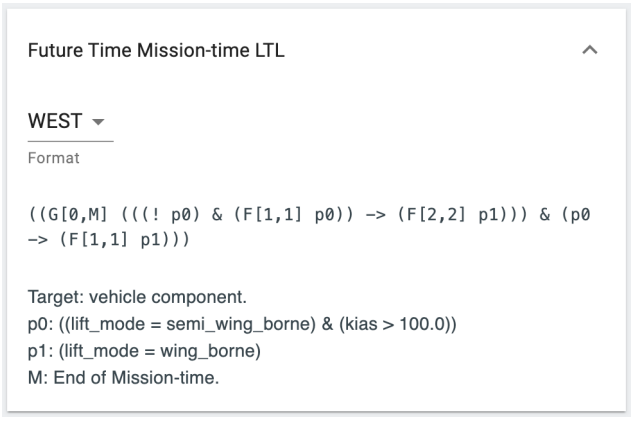

Fig. 2: WEST Format for LPC_SWB_TO_WB

2.2 MLTL in R2U2 Format

When FRET translates FRETISH to a fmLTL or MLTL formula φ, the formalization is designed to be evaluated at the beginning of time when the time index $i = 0$ (i.e., designed to only determine the satisfiability of $\pi,0 \models \varphi$) [28]. Since R2U2 is a

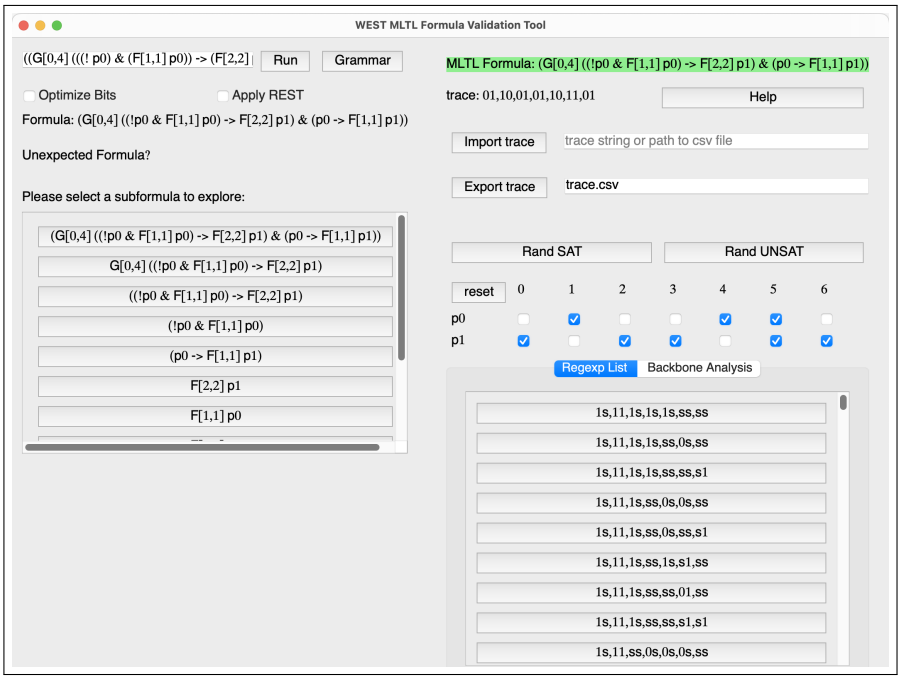

Fig. 3: WEST Tool for LPC_SWB_TO_WB with $M=5$

stream-based runtime monitoring framework, R2U2 re-evaluates MLTL formulas for each time index i; therefore, we need to translate MLTL for stream-based monitoring with R2U2. There are two translations required: **(1)** Since R2U2 re-evaluates for each time index i, R2U2 creates an implicit $\mathbf{G}_{[0,M]}$ operator on the formula φ; therefore, if there is an external $\mathbf{G}_{[0,M]}$ operator (i.e., $\mathbf{G}_{[0,M]}\ \varphi$), we simply remove the global operator (similar to [1], [3], and [12]). **(2)** When there is no external $\mathbf{G}_{[0,M]}$ operator, we cannot apply **(1)** and have to indicate that φ must only be evaluated for $i=0$ by adding an implication, i.e., $(i=0)\rightarrow\varphi$. Fig. 4 demonstrates the LPC_SWB_TO_WB requirement after this translation in the applicable R2U2 format.

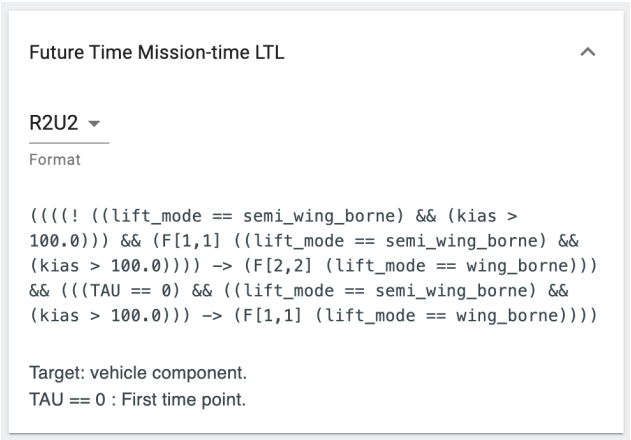

Fig. 4: R2U2 Format for LPC_SWB_TO_WB

3 MLTL Rewrite Rules

R2U2 achieves minimal resource overhead by decomposing MLTL specifications into subformula nodes represented in an Abstract Syntax Tree (AST). Each node computes and stores verdict-timestamp tuple(s) $T_\psi = (v,\tau)$ for its subformula ψ, where $v \in \{\mathsf{true},\mathsf{false}\}$ and $\tau \in \mathbb{N}_0$. A node's verdict-timestamp tuple(s) are stored in a shared connection queue (SCQ), where the SCQ is a circular buffer that overwrites verdict-timestamp tuple(s) in a circular manner. Fig. 5 provides the AST for the LPC_SWB_TO_WB requirement, where each node has an assigned SCQ. R2U2 reasons over the AST by evaluating each subformula node from the bottom up and propagating verdict-timestamp tuples to the parent node(s), computing each verdict according to the semantics of MLTL (Definition 6) as the input stream progresses [2,41].

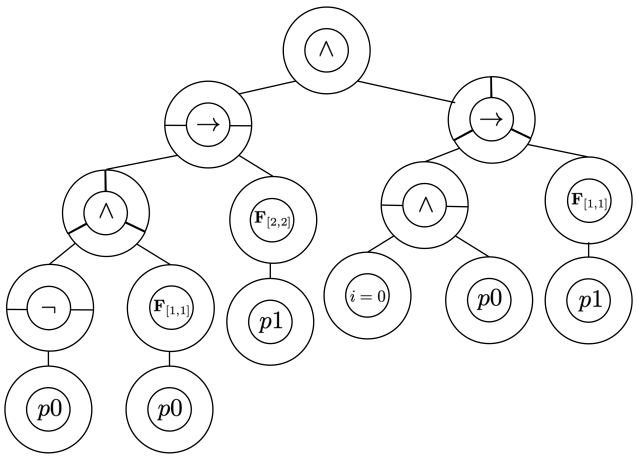

Fig. 5: AST for LPC_SWB_TO_WB where
$p0 \coloneqq (\text{lift\_mode} == \text{semi\_wing\_borne}) \wedge (\text{kias} > 100.0)$ and
$p1 \coloneqq (\text{lift\_mode} == \text{wing\_borne})$

These SCQs are minimally sized to ensure information is never overwritten before its parent node in the tree consumes it. The *propagation delay* of an MLTL formula determines when a verdict-timestamp tuple is safe to overwrite, which represents the time delay between when a set of propositions $\pi(i)$ arrives and when the satisfaction of $\pi,i \models \varphi$ can be computed (Definition 9 below). The minimum size for an AST node φ's SCQ is determined by the worst-case propagation delay of its sibling nodes and its own best-case propagation delay; in the worst case, a node φ must store verdict-timestamp tuple(s) in its SCQ until all of φ's siblings have the same timestamp for these tuples to be consumed by their parent node. Therefore, the size of node φ's SCQ corresponds to the maximum timestamp mismatch between node φ and φ's siblings (Definition 10 below). Consequently, the number of operators in the formula determines the number of SCQs and the interval size of temporal operators determines the size of each node's SCQ. Thus, the number of operators and interval lengths directly impact memory requirements. Similarly, the number of operators directly impacts latency requirements, i.e., the more operators to evaluate, the greater the latency required to evaluate the entire specification [2].

Definition 9 (MLTL Propagation Delay Semantics [41]). *Let ψ and ξ be subformulas of an MLTL formula φ. The worst-case propagation delay ($\varphi.wpd$) and the best-case propagation delay ($\varphi.bpd$) are recursively defined as follows:*

- *If* $\varphi \in \mathcal{AP}$: $\begin{cases} \varphi.wpd = 0 \\ \varphi.bpd = 0 \end{cases}$
- *If* $\varphi = \neg\psi$: $\begin{cases} \varphi.wpd = \psi.wpd \\ \varphi.bpd = \psi.bpd \end{cases}$
- *If* $\varphi = \psi \vee \xi$ *or* $\varphi = \psi \wedge \xi$: $\begin{cases} \varphi.wpd = max(\psi.wpd,\, \xi.wpd) \\ \varphi.bpd = min(\psi.bpd,\, \xi.bpd) \end{cases}$
- *If* $\varphi = \boldsymbol{G}_{[lb,ub]}\psi$ *or* $\varphi = \boldsymbol{F}_{[lb,ub]}\psi$: $\begin{cases} \varphi.wpd = \psi.wpd + ub \\ \varphi.bpd = \psi.bpd + lb \end{cases}$
- *If* $\varphi = \psi\ \boldsymbol{U}_{[lb,ub]}\ \xi$ *or* $\varphi = \psi\ \boldsymbol{R}_{[lb,ub]}\ \xi$: $\begin{cases} \varphi.wpd = max(\psi.wpd,\, \xi.wpd) + ub \\ \varphi.bpd = min(\psi.bpd,\, \xi.bpd) + lb \end{cases}$

Definition 10 (SCQ Memory Size [41,68]). *Let $\mathbb{S}_\varphi$ be the set of all of φ's sibling nodes, then the size of φ's SCQ is the difference between the maximum worst-case propagation delay in $\mathbb{S}_\varphi$ and φ's best-case propagation delay, or zero, whichever is greater (plus one):*

$$SCQ_{size}(\varphi) = max(max\{s.wpd \mid s \in \mathbb{S}_\varphi\} - \varphi.bpd,\, 0) + 1$$

After manually inspecting the MLTL formalizations derived from the 288 possible template keys, we compose a set of 157 rewrite rules to decrease the memory requirements and latency for each formula by lowering the number of operators and size of temporal intervals in a given MLTL formula. The rules include: **(1)** Boolean operator simplifications, **(2)** next operator simplifications, **(3)** temporal operator and interval simplifications, and **(4)** reducing redundant conditions based on when in/not in `scope`. Some of the rewrite rules are variants of previously-explored parametric rules [36]. For example, the rewrite rule $\mathbf{F}_{[1,1]}(\mathbf{F}_{[0,ub]}\ \varphi) \mapsto \mathbf{F}_{[1,ub+1]}\ \varphi$ in Table 3 below is an instantiation of rule R1 from [36]: $\mathbf{F}_{[lb_1,ub_1]}(\mathbf{F}_{[lb_2,ub_2]}\ \varphi) \mapsto \mathbf{F}_{[lb_1+lb_2,ub_1+ub_2]}\ \varphi$. However, many of the 157 rewrite rules are entirely novel (e.g., all those in Tables 5 and 6). The rewrites can therefore be viewed as a major FRET-specific extension to those in [36]. Furthermore, all rewrites are directly implemented into FRET, where FRET applies these rewrites by performing a single bottom-up traversal of the AST for the given MLTL formula, searches for any pattern matches in any of the rewrites, and then applies a rewrite if it finds one. Investigating more complete approaches (i.e., addressing confluence [20]) is left to future work.

These rewrite rules also require some way to reason about intervals with symbolic values (i.e., not just concrete values like [0,5], [1,1], etc.). Therefore, we introduce the notion of a symbolic MLTL formula (Definition 11 below). Example symbolic MLTL formulas include $\mathbf{G}_{[lb,lb+3]}\ \psi$ and $\psi\ \mathbf{U}_{[0,ub]}\ \xi$, where $lb,ub \in Var$ are symbolic values. We can obtain a concrete MLTL formula from a symbolic one by assigning a concrete value to each variable in Var, keeping in mind that for each interval $[\mathcal{I}_1,\mathcal{I}_2]$ it must hold that $\mathcal{I}_1 \leq \mathcal{I}_2$ and $\mathcal{I}_1 \geq 0$.

Definition 11 (Symbolic MLTL Formulas). *Let Var be a set of symbolic interval variable names. A symbolic MLTL formula is defined the same as an MLTL formula*

(Definition 4) except that an interval $I=[\mathcal{I}_1,\mathcal{I}_2]$ is defined using symbolic arithmetic expressions:

$$\mathcal{I} ::= n \mid v \mid -\mathcal{I}_1 \mid \mathcal{I}_1+\mathcal{I}_2 \mid \mathcal{I}_1-\mathcal{I}_2$$

where $n\in\mathbb{N}_0$, $v\in Var$, and $\mathcal{I}$, $\mathcal{I}_1$, and $\mathcal{I}_2$ are symbolic arithmetic expressions.

Boolean Rewrite Rules. FRET already provides a set of Boolean rewrite rules that are applied to the fmLTL formalizations. After manually reviewing all of FRET's MLTL formalizations derived from the 288 possible template keys, we identified an additional 15 Boolean rewrite rules that further decrease the number of operators in these formalizations. Table 2 provides the additional Boolean rewrite rules. Additionally, the number of times each rewrite rule is applied over all 288 template keys is also included to indicate the prevalence of each rewrite rule.

Table 2: Boolean Operator Rewrite Rules

Original	Rewrite	# Times Applied
$\varphi \rightarrow true$	$true$	18
$\varphi \rightarrow false$	$\neg\varphi$	8
$\neg\varphi_1 \vee \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	32
$\neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2)$	$\varphi_1 \rightarrow \varphi_2$	103
$(\varphi_1 \rightarrow \varphi_2) \vee \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	12
$\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)$	$(\varphi_1 \wedge \varphi_2) \rightarrow \varphi_3$	111
$(\varphi_1 \wedge \varphi_2) \rightarrow ((\varphi_2 \wedge \varphi_3) \vee \varphi_4)$	$(\varphi_1 \wedge \varphi_2) \rightarrow (\varphi_3 \vee \varphi_4)$	3
$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_1 \wedge \varphi_2) \vee \varphi_4)$	$(\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee \varphi_4)$	3
$(\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \rightarrow (\varphi_4 \wedge \varphi_3)$	$(\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \rightarrow \varphi_4$	96
$(\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \rightarrow ((\varphi_3 \wedge \varphi_4) \vee \varphi_5)$	$(\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \rightarrow (\varphi_4 \vee \varphi_5)$	6
$\varphi_1 \rightarrow ((\varphi_1 \wedge \varphi_2) \vee \varphi_3)$	$\varphi_1 \rightarrow (\varphi_2 \vee \varphi_3)$	3
$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\varphi_1 \wedge \varphi_4) \vee \varphi_5)))$	$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow (\varphi_4 \vee \varphi_5)))$	3
$(\varphi_1 \wedge (\varphi_2 \vee \varphi_3)) \vee \varphi_3$	$(\varphi_1 \wedge \varphi_2) \vee \varphi_3$	20
$\varphi_1 \wedge (\varphi_1 \wedge \varphi_2)$	$\varphi_1 \wedge \varphi_2$	3
$\varphi_1 \vee (\varphi_2 \wedge \neg\varphi_1)$	$\varphi_1 \vee \varphi_2$	32

Next Operator Rewrite Rules. In MLTL, when there is an external next operator (i.e., $\mathbf{F}_{[1,1]}$) applied to a temporal operator, it's applicable to simply slide the temporal interval constraint by +1, e.g., $\mathbf{F}_{[1,1]}(\psi\ \mathbf{U}_{[lb,ub]}\ \xi) \equiv \psi\ \mathbf{U}_{[lb+1,ub+1]}\ \xi$. Notably, this rewrite rule cannot be applied to fmLTL, i.e., $\mathbf{X}(\psi\ \mathbf{U}_{[lb,ub]}\ \xi) \not\equiv \psi\ \mathbf{U}_{[lb+1,ub+1]}\ \xi$. Therefore, we identified 14 additional rewrite rules that apply this behavior as shown in Table 3.

It is important to note that the first two rewrites in Table 3 actually increase the number of operators; however, both φ_1 and φ_2 are assumed to be subformulas that include temporal operators themselves, which would then efficiently apply the subsequent rewrite rules. To the same extent, we apply an additional 12 rewrite rules that utilize distributive law to move $\mathbf{F}_{[1,1]}$ as an external operator if the internal subformula is temporal as shown in Table 4. For example, $(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}\ ((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{F}_{[1,1]}\ \varphi_3)) \mapsto \mathbf{F}_{[1,1]}\ (\varphi_1\ \mathbf{U}_{[0,M]}\ (\varphi_2 \wedge \varphi_3)))$, which then allows the straightforward application of the rewrite rule $\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}\ \varphi_2) \mapsto \varphi_1\ \mathbf{U}_{[1,M]}\ \varphi_2$ from Table 3 to produce $\varphi_1\ \mathbf{U}_{[1,M]}\ (\varphi_2 \wedge \varphi_3)$ as the final MLTL formula.

Temporal Operator and Interval Rewrite Rules. We also examined the other temporal operators (i.e., $\mathbf{G}_I$, $\mathbf{F}_I$, $\mathbf{U}_I$, and $\mathbf{R}_I$) and composed an additional 10 rewrite

Table 3: Next Operator Rewrite Rules

Original	Rewrite	# Times Applied
$\mathbf{F}_{[1,1]}(\varphi_1 \vee \varphi_2)$	$(\mathbf{F}_{[1,1]}\ \varphi_1) \vee (\mathbf{F}_{[1,1]}\ \varphi_2)$	65
$\mathbf{F}_{[1,1]}(\varphi_1 \wedge \varphi_2)$	$(\mathbf{F}_{[1,1]}\ \varphi_1) \wedge (\mathbf{F}_{[1,1]}\ \varphi_2)$	79
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[1,1]}\ \varphi)$	$\mathbf{F}_{[2,2]}\ \varphi$	28
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[0,M]}\ \varphi)$	$\mathbf{F}_{[1,M]}\ \varphi$	12
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[0,ub-1]}\ \varphi)$	$\mathbf{F}_{[1,ub]}\ \varphi$	46
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[0,ub]}\ \varphi)$	$\mathbf{F}_{[1,ub+1]}\ \varphi$	24
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[ub,ub]}\ \varphi)$	$\mathbf{F}_{[ub+1,ub+1]}\ \varphi$	10
$\mathbf{F}_{[1,1]}(\mathbf{F}_{[1,ub]}\ \varphi)$	$\mathbf{F}_{[2,ub+1]}\ \varphi$	12
$\mathbf{F}_{[1,1]}(\mathbf{G}_{[0,M]}\ \varphi)$	$\mathbf{G}_{[1,M]}\ \varphi$	92
$\mathbf{F}_{[1,1]}(\mathbf{G}_{[0,ub]}\ \varphi)$	$\mathbf{G}_{[1,ub+1]}\ \varphi$	31
$\mathbf{F}_{[1,1]}(\mathbf{G}_{[ub,ub]}\ \varphi)$	$\mathbf{G}_{[ub+1,ub+1]}\ \varphi$	3
$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}\ \varphi_2)$	$\varphi_1\ \mathbf{U}_{[1,M]}\ \varphi_2$	55
$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{R}_{[0,M]}\ \varphi_2)$	$\varphi_1\ \mathbf{R}_{[1,M]}\ \varphi_2$	134
$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{R}_{[0,ub]}\ \varphi_2)$	$\varphi_1\ \mathbf{R}_{[1,ub+1]}\ \varphi_2$	22

Table 4: Next Operator Rewrite Rules via Distributive Law

Original	Rewrite	# Times Applied
$\mathbf{G}_{[0,M]}(\mathbf{F}_{[1,1]}\ \varphi)$	$\mathbf{F}_{[1,1]}(\mathbf{G}_{[0,M]}\ \varphi)$	33
$\mathbf{F}_{[lb,ub]}(\mathbf{F}_{[1,1]}\ \varphi)$	$\mathbf{F}_{[1,1]}(\mathbf{F}_{[lb,ub]}\ \varphi)$	36
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{F}_{[1,1]}\ \varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge \varphi_3))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{F}_{[1,M]}\varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\mathbf{F}_{[0,M]}\ \varphi_3)))$	2
$(\mathbf{F}_{[1,1]}\varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{F}_{[2,2]}\ \varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\mathbf{F}_{[1,1]}\ \varphi_3)))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{F}_{[1,ub+1]}\ \varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\mathbf{F}_{[0,ub]}\ \varphi_3)))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{G}_{[1,M]}\ \varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\mathbf{G}_{[0,M]}\ \varphi_3)))$	13
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\mathbf{G}_{[1,ub+1]}\ \varphi_3))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\mathbf{G}_{[0,ub]}\ \varphi_3)))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}\ ((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge$ $((\mathbf{G}_{[1,M]}\ \varphi_3) \wedge (\mathbf{F}_{[1,1]}\ \varphi_4)))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}\ (\varphi_2 \wedge$ $((\mathbf{G}_{[0,M]}\ \varphi_3) \wedge \varphi_4)))$	11
$(\mathbf{F}_{[1,1]}\ \varphi_1)\mathbf{U}_{[0,M]}$ $((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge (\varphi_3\ \mathbf{R}_{[1,M]}\ \varphi_4))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}$ $(\varphi_2 \wedge (\varphi_3\ \mathbf{R}_{[0,M]}\ \varphi_4)))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{U}_{[0,M]}\ ((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge$ $((\mathbf{F}_{[1,ub]}\ \varphi_3) \vee (\mathbf{G}_{[ub+1,ub+1]}\ \varphi_4)))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}(\varphi_2 \wedge$ $((\mathbf{F}_{[0,ub]}\ \varphi_3) \vee (\mathbf{G}_{[ub,ub]}\ \varphi_4))))$	1
$(\mathbf{F}_{[1,1]}\ \varphi_1)\mathbf{U}_{[0,M]}\ ((\mathbf{F}_{[1,1]}\ \varphi_2) \wedge$ $((\varphi_3\ \mathbf{R}_{[1,M]}\ \varphi_4) \vee \mathbf{G}_{[1,M]}\ \varphi_5))$	$\mathbf{F}_{[1,1]}(\varphi_1\ \mathbf{U}_{[0,M]}(\varphi_2 \wedge$ $((\varphi_3\ \mathbf{R}_{[0,M]}\ \varphi_4) \vee \mathbf{G}_{[0,M]}\ \varphi_5)))$	1

rules that reduced the number of operators (Table 5) and 5 rewrite rules that reduced the temporal interval constraint of these temporal operators (Table 6). In Table 5, the first two rules do not reduce the number of operators, but, similar to the first two rewrite rules in Table 3, allow subsequent rewrite rules to be more easily applied by transforming to negation normal form.

Table 5: Temporal Operator Rewrite Rules

Original	Rewrite	# Times Applied
$\neg(\mathbf{F}_{[0,M]}\ \varphi)$	$\mathbf{G}_{[0,M]}\ \neg\varphi$	8
$\neg(\varphi_1\ \mathbf{U}_{[0,M]}\ \varphi_2)$	$\neg\varphi_1\ \mathbf{R}_{[0,M]}\ (\neg\varphi_2)$	210
$(\neg\varphi_1\ \mathbf{U}_{[0,M]}\ \varphi_1)\vee(\mathbf{G}_{[0,M]}\neg\varphi_1)$	$true$	6
$((\varphi_1\ \wedge\ \varphi_2)\ \vee\ \varphi_3)\ \mathbf{R}_{[0,M]}\ \varphi_1$	$(\varphi_2\ \vee\ \varphi_3)\ \mathbf{R}_{[0,M]}\ \varphi_1$	64
$((\varphi_1\ \vee\ \varphi_2)\ \mathbf{R}_{[0,M]}\ (\varphi_3\ \vee\ \varphi_4))\ \vee\ (\varphi_2\ \mathbf{R}_{[0,M]}\ \varphi_3)$	$(\varphi_1\ \vee\ \varphi_2)\mathbf{R}_{[0,M]}(\varphi_3\ \vee\ \varphi_4)$	20
$((\varphi_1\ \vee\ \varphi_2)\ \mathbf{R}_{[0,M]}\ \neg(\varphi_3\ \wedge\ \varphi_4))\ \vee\ (\varphi_2\ \mathbf{R}_{[0,M]}\ \neg\varphi_3)$	$(\varphi_1\ \vee\ \varphi_2)\ \mathbf{R}_{[0,M]}\ \neg(\varphi_3\ \wedge\ \varphi_4)$	12
$(\mathbf{G}_{[0,M]}\ ((\neg\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \varphi_1))\rightarrow((\mathbf{F}_{[1,1]}\ \neg\varphi_1)\ \mathbf{R}_{[1,M]}\ \varphi_2)))\ \wedge\ (\varphi_1\rightarrow((\mathbf{F}_{[1,1]}\ \neg\varphi_1)\ \mathbf{R}_{[0,M]}\ \varphi_2))$	$\mathbf{G}_{[0,M]}\ (\varphi_1\rightarrow\varphi_2)$	13
$(\mathbf{G}_{[0,M]}\ ((\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \neg\varphi_1))\rightarrow((\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{R}_{[1,M]}\ \varphi_2)))\ \wedge\ (\neg\varphi_1\rightarrow((\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{R}_{[0,M]}\ \varphi_2))$	$\mathbf{G}_{[0,M]}\ (\neg\varphi_1\rightarrow\varphi_2)$	27
$(\mathbf{G}_{[0,M]}\ ((\neg\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \varphi_1))\rightarrow(((\mathbf{F}_{[1,1]}\ \neg\varphi_1)\ \mathbf{R}_{[1,M]}\ \varphi_2)\ \wedge\ (\mathbf{F}_{[1,1]}\ \varphi_3))))\ \wedge\ (\varphi_1\rightarrow(((\mathbf{F}_{[1,1]}\ \neg\varphi_1)\ \mathbf{R}_{[0,M]}\ \varphi_2)\ \wedge\ \varphi_3))$	$(\mathbf{G}_{[0,M]}\ ((\varphi_1\rightarrow\varphi_2)\ \wedge\ ((\neg\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \varphi_1)\rightarrow(\mathbf{F}_{[1,1]}\ \varphi_3)))))\ \wedge\ (\varphi_1\rightarrow\varphi_3)$	12
$(\mathbf{G}_{[0,M]}\ ((\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \neg\varphi_1))\rightarrow(((\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{R}_{[1,M]}\ \varphi_2)\ \wedge\ (\mathbf{F}_{[1,1]}\varphi_3))))\ \wedge\ (\neg\varphi_1\rightarrow(((\mathbf{F}_{[1,1]}\ \varphi_1)\ \mathbf{R}_{[0,M]}\ \varphi_2)\ \wedge\ \varphi_3))$	$(\mathbf{G}_{[0,M]}\ ((\neg\varphi_1\rightarrow\varphi_2)\ \wedge\ ((\varphi_1\ \wedge\ (\mathbf{F}_{[1,1]}\ \neg\varphi_1)\rightarrow(\mathbf{F}_{[1,1]}\ \varphi_3)))))\ \wedge\ (\neg\varphi_1\rightarrow\varphi_3)$	24

Table 6: Temporal Interval Rewrite Rules

Original	Rewrite	# Times Applied
$(\mathbf{G}_{[lb,ub]}\ \varphi)\ \vee\ (\mathbf{G}_{[0,M]}\ \varphi)$	$\mathbf{G}_{[lb,ub]}\ \varphi$	32
$(\mathbf{G}_{[0,ub]}\ \varphi_1)\ \vee\ (\varphi_2\ \mathbf{R}_{[0,M]}\ \varphi_1)$	$\varphi_2\ \mathbf{R}_{[0,ub]}\ \varphi_1$	64
$(\mathbf{G}_{[0,ub]}\ \neg\varphi)\ \wedge\ (\mathbf{F}_{[0,ub+1]}\ \varphi)$	$(\mathbf{G}_{[0,ub]}\ \neg\varphi)\ \wedge\ (\mathbf{F}_{[ub+1,ub+1]}\ \varphi)$	8
$(\mathbf{F}_{[0,ub]}\ \varphi)\ \vee\ (\mathbf{G}_{[0,ub+1]}\ \neg\varphi)$	$(\mathbf{F}_{[0,ub]}\ \varphi)\ \vee\ (\mathbf{G}_{[ub+1,ub+1]}\ \neg\varphi)$	8
$(\varphi_2\ \mathbf{R}_{[0,ub]}\ \neg\varphi_1)\ \wedge\ ((\mathbf{F}_{[0,ub+1]}\ \varphi_1)\ \vee(\mathbf{F}_{[0,ub]}\ \varphi_2))$	$(\varphi_2\ \mathbf{R}_{[0,ub]}\ \neg\varphi_1)\ \wedge\ ((\mathbf{F}_{[ub+1,ub+1]}\ \varphi_1)\ \vee\ (\mathbf{F}_{[0,ub]}\ \varphi_2))$	20

When in/not in `scope` Rewrite Rules. Within the FRETish semantics, many of the `scope` field options will require the formalization to watch for a rising or falling edge of being in a *mode* (i.e., $\neg mode\ \wedge\ \mathbf{F}_{[1,1]}\ mode$ and $mode\ \wedge\ \mathbf{F}_{[1,1]}\ \neg mode$, respectively). Consider a formalization such as $mode\rightarrow(\varphi\ \vee\ \mathbf{F}_{[0,ub]}\ (mode\ \wedge\ (\mathbf{F}_{[1,1]}\ \neg mode)))$. The right-hand side of the implication is only considered when the system is in *mode* at time $i{+}0$, and the system will continue to be in *mode* up to and including when $F_{[1,1]}\ \neg mode$ holds for the first time since $i{+}0$. Thus, this formalization can be rewritten as $mode\rightarrow(\varphi\ \vee\ \mathbf{F}_{[0,ub]}\ (\mathbf{F}_{[1,1]}\neg mode))$. Taking advantage of this

behavior, we composed 101 rewrite rules that reduce these redundant conditions based on when in/not in `scope` (available in Appendix A).

3.1 Resource Overhead Analysis

These 157 rewrite rules resulted in an average reduction of 15 operators, or 36.05% less operators, over all 288 possible template keys; hence, reducing the resource overhead to monitor FRET's MLTL formalizations. Fig. 6 demonstrates the distribution of the percentage decrease of operators across all 288 templates. The reduction in resource overhead can be directly observed in Fig. 7, where the memory requirements to monitor each of FRET's 288 template keys within R2U2 are plotted. The memory requirements plotted are the memory requirements in kilobytes to store the AST in R2U2's Rust realization, given the SCQ sizing in Definition 10. Overall, a decrease in the number of operators and, thus, a decrease in memory requirements are observed for all 288 possible template keys.

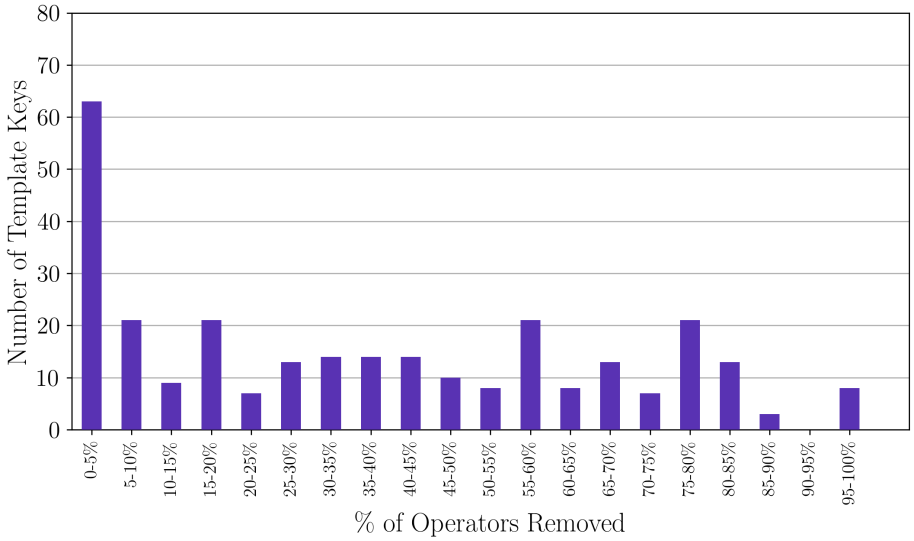

Fig. 6: Percentage of Operators Removed: The x-axis represents the percentage of operators removed from the original template key after applying the 157 rewrite rules. The y-axis represents the number of template keys with that percentage of operators removed.

3.2 Correctness of Rewrite Rules

To prove that each of the rewrites is correct, we present a new SMT-based proof technique for showing the equivalence of MLTL formulas with symbolic intervals (Definition 11). Given two MLTL formulas φ and ψ, a rewrite rule $\varphi \mapsto \psi$ is defined as correct if $\varphi \equiv \psi$. Definition 6 defines equivalence such that $\varphi \equiv \psi$ if and only if $(\pi,0 \models \varphi) \leftrightarrow (\pi,0 \models \psi)$ for all finite traces π. However, the rewrite rules target a monitoring context where we always assume an extension to the trace π up to a maximum length of $M+1$. We therefore modify the notion of equivalence so that $\varphi \equiv \psi$ if and only if $(\pi,0 \models \varphi) \leftrightarrow (\pi,0 \models \psi)$ for all finite traces π where $|\pi| = M+1$ if M is present in either φ or ψ, or $|\pi| = max(cplen(\varphi), cplen(\psi))$ otherwise.

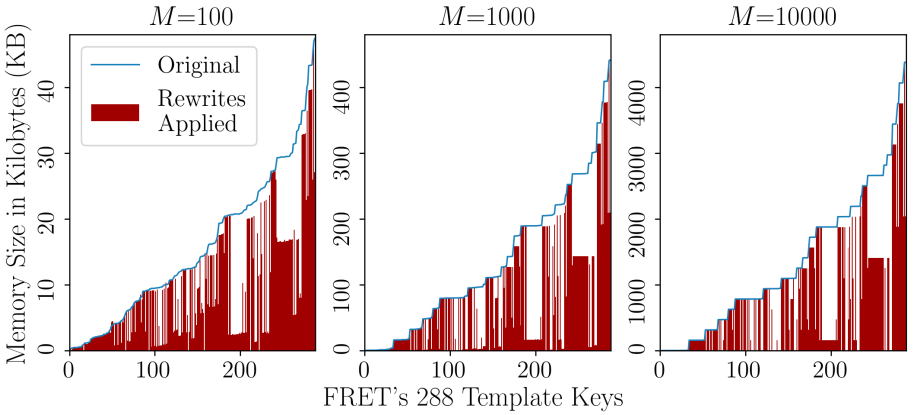

Fig. 7: Memory in Kilobytes Removed: The x-axis represents FRET's 288 template keys in increasing order of memory size. The y-axis demonstrates the memory required for monitoring a single template key in R2U2 before (blue) and after (red) rewrites were applied with increasing values of M (i.e., $M=100$, $M=1000$, and $M=10000$), assuming that all `scope`, `condition`, and `response` fields are over Boolean variables (i.e., not Boolean expressions which would require extra varied memory from R2U2's Booleanizer module [35]) and $N=20$ for any applicable `timing` fields.

One way to prove equivalence is via MLTL satisfiability checking, an NEXPTIME-complete problem [43]. For an MLTL formula φ, satisfiability checking is the problem of computing whether $\exists\pi.(\pi,0\models\varphi)$. If there does not exist a trace π that satisfies φ and does not satisfy ψ (or vice-versa), then we can conclude $\varphi\equiv\psi$.

Lemma 1 (MLTL Equivalence Checking via Satisfiability [38]). *Given two MLTL formulas φ and ψ, the formula $\neg(\varphi\leftrightarrow\psi)$ is unsatisfiable if and only if $\varphi\equiv\psi$.*

Proof. If $\neg(\varphi\leftrightarrow\psi)$ is satisfiable, then there exists a trace π such that $\pi,0\models\varphi$ and $\pi,0\not\models\psi$ (or vice-versa). Conversely, if $\neg(\varphi\leftrightarrow\psi)$ is unsatisfiable, then no such trace exists, implying that the two formulas agree on every trace. □

Existing techniques for MLTL satisfiability checking include reductions to LTL and LTLf model checking [43], Boolean satisfiability [31], and SMT, specifically UFLIA [43] and QF_BV [37] encodings. Any of these techniques can prove that two MLTL formulas are equivalent; however, we are interested in proving that two *symbolic* MLTL formulas are equivalent (i.e., are equivalent for all valid assignments to the interval variables). The UFLIA encoding admits quantifiers, uninterpreted functions, and (crucially) linear integer arithmetic [7]; therefore, we extend this encoding to check if a symbolic MLTL formula φ is satisfiable. Let $Intvl(\varphi)$ be the set of all temporal intervals I that appear in φ, then our extension is as follows:

- For each $p\in\mathcal{AP}$, declare an uninterpreted function $f_p:\mathbb{N}_0\to Bool$, representing whether p holds at a given index of a trace.
- For each $v\in Var$, declare an uninterpreted constant $c_v\in\mathbb{N}_0$.
- For each interval $I=[\mathcal{I}_1,\mathcal{I}_2]\in Intvl(\varphi)$, assert that $\mathcal{I}_1\leq\mathcal{I}_2$ and $\mathcal{I}_1\geq 0$.

- Recursively define the function $\mathsf{SAT}(\varphi, i, len)$ for an MLTL formula φ, time index i, and length of the trace len, where $i, len \in \mathbb{N}_0$, as follows:
  - $\mathsf{SAT}(\mathsf{true}, i, len) = \mathsf{true}$
  - $\mathsf{SAT}(\mathsf{false}, i, len) = \mathsf{false}$
  - $\mathsf{SAT}(p, i, len) = f_p(i)$
  - $\mathsf{SAT}(\neg\psi, i, len) = \neg\mathsf{SAT}(\psi, i, len)$
  - $\mathsf{SAT}(\varphi \wedge \psi, i, len) = \mathsf{SAT}(\varphi, i, len) \wedge \mathsf{SAT}(\psi, i, len)$
  - $\mathsf{SAT}(\varphi \vee \psi, i, len) = \mathsf{SAT}(\varphi, i, len) \vee \mathsf{SAT}(\psi, i, len)$
  - $\mathsf{SAT}(\mathbf{F}_{[\mathcal{I}_1,\mathcal{I}_2]}\ \psi, i, len) = (len \geq i+\mathcal{I}_1) \wedge \exists j.(\mathcal{I}_1 \leq j \leq \mathcal{I}_2) \wedge \mathsf{SAT}(\psi, j, len-j)$
  - $\mathsf{SAT}(\mathbf{G}_{[\mathcal{I}_1,\mathcal{I}_2]}\ \psi, i, len) = (len < i+\mathcal{I}_1) \vee \forall j.(\mathcal{I}_1 \leq j \leq \mathcal{I}_2) \rightarrow \mathsf{SAT}(\psi, j, len-j)$
  - $\mathsf{SAT}(\varphi\ \mathbf{U}_{[\mathcal{I}_1,\mathcal{I}_2]}\ \psi, i, len) = (len \geq i+\mathcal{I}_1) \wedge \exists j.(\mathcal{I}_1 \leq j \leq \mathcal{I}_2) \wedge \mathsf{SAT}(\psi, j, len-j) \wedge \forall k.(\mathcal{I}_1 \leq k \leq j) \rightarrow \mathsf{SAT}(\varphi, k, len-k)$
  - $\mathsf{SAT}(\varphi\ \mathbf{R}_{[\mathcal{I}_1,\mathcal{I}_2]}\ \psi, i, len) = (len < i+\mathcal{I}_1) \vee \forall j.(\mathcal{I}_1 \leq j \leq \mathcal{I}_2) \rightarrow \mathsf{SAT}(\psi, j, len-j) \vee \exists k.(\mathcal{I}_1 \leq k \leq j) \wedge \mathsf{SAT}(\varphi, k, len-k)$

Theorem 1. *Let φ and ψ be two symbolic MLTL formulas with symbolic interval variables over Var. If M appears in φ or ψ, then let $L = M + 1$, otherwise let $L = max(cplen(\varphi), cplen(\psi))$. Then $\varphi \equiv \psi$ for all valid assignments to the variables in Var if and only if $\mathsf{SAT}(\neg(\varphi \leftrightarrow \psi), 0, L)$ is unsatisfiable.*

Proof. The SAT encoding is the same as in Section 4.3 of [43] except that symbolic intervals are in place of concrete ones. The encoding also checks for a valid assignment to each variable in Var (i.e., that cause $\mathcal{I}_1 \leq \mathcal{I}_2$ and $\mathcal{I}_1 \geq 0$ for each interval in $Intvl(\varphi) \cup Intvl(\psi)$). We can therefore extend Theorem 6 of [43] to state that given a symbolic MLTL formula ξ, there exists a valid assignment to the variables in Var that make ξ satisfiable iff $\exists len.\mathsf{SAT}(\xi, 0, len)$ is satisfiable. Conversely, there is no assignment to the variables in Var that make ξ satisfiable iff $\exists len.\mathsf{SAT}(\xi, 0, len)$ is unsatisfiable.

If we let $\xi = \neg(\varphi \leftrightarrow \psi)$, then by Lemma 1, $\varphi \equiv \psi$ iff there is no assignment to the variables in Var that make $\neg(\varphi \leftrightarrow \psi)$ satisfiable iff $\exists len.\mathsf{SAT}(\xi, 0, len)$ is unsatisfiable. Since the re-definition of $\equiv$ only considers traces of length L, then $\varphi \equiv \psi$ iff $\mathsf{SAT}(\neg(\varphi \leftrightarrow \psi), 0, L)$ is unsatisfiable. □

We implemented our new encoding into R2U2's formula compiler, the Configuration Compiler for Property Organization (C2PO) [35], which takes as input a pair of MLTL formulas φ and ψ and outputs an SMTLIB2 file corresponding to $\mathsf{SAT}(\neg(\varphi \leftrightarrow \psi), 0, L)$. [6] We then verify the satisfiability of the SMTLIB2 file using CVC5 [6], which we employ because it specifically supports a parallel portfolio mode. CVC5 proves 152 of the 157 rewrite rules equivalent in under 5 seconds each using 24 parallel processes, meaning this fully-automated approach proved $\sim$96.8% of the rewrites correct without any manual intervention.

The other five rewrite rule proofs timed out after an hour and all feature some form of $\forall/\exists$ quantifier alternation in their SAT encoding, which we suspect is the reason for CVC5 failing to prove them under the timeout. Four of these are the final four rules in Table 5. These are the only rules that have nested mission-time

[6] *https://github.com/R2U2/r2u2/tree/fret-rewrites/compiler/scripts/fret*

bounded temporal intervals (i.e., a $\mathbf{R}_{[0,M]}$ operator nested in a $\mathbf{G}_{[0,M]}$), which has each quantifier range over values from 0 to M. The final unproven rule is $((\varphi_1 \vee \varphi_2) \wedge \varphi_3)\ \mathbf{R}_{[0,M]}\ \varphi_1 \mapsto (\varphi_2 \vee \varphi_3)\ \mathbf{R}_{[0,M]}\ \varphi_1$, which also includes quantifier alternation, but it is unclear exactly why CVC5 struggles to prove it.

4 FRET-R2U2 Integration

The complete workflow of going from FRET to R2U2 monitors is shown in Fig. 8. From FRETish and FRET's fmLTL formalizations, MLTL formalizations (Section 2 above) and important variable data (Section 4.1 below) are employed to produce the correct export files required to configure R2U2 monitors (Section 4.2).

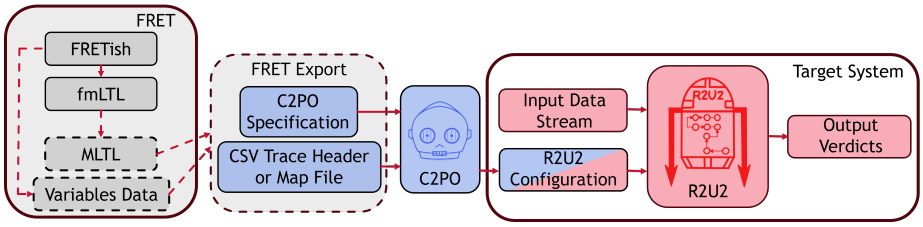

Fig. 8: FRET to R2U2 Workflow: The gray boxes indicate the FRET pipeline, the blue boxes indicate the process required to configure R2U2, the red boxes indicate the pieces required to execute R2U2, and dashed lines indicate the newly added features in the workflow.

4.1 Variables Data Mapping

To configure R2U2 monitors from FRET, users must indicate necessary information about the variables included in their FRETish requirements such as the variable type (i.e., input, output, or internal), the data type (i.e., boolean, integer, or double), and the internal variable assignment (when applicable). Internal variables are expressions over other input/output variables, and an integrated parser will indicate whether the internal variable assignment follows the correct syntax as shown in Fig. 9. With R2U2 selected as the export language, a green check will mark each properly mapped variable as complete and the export button will *only* be enabled once all variables have been marked as complete, as displayed in Fig. 10.

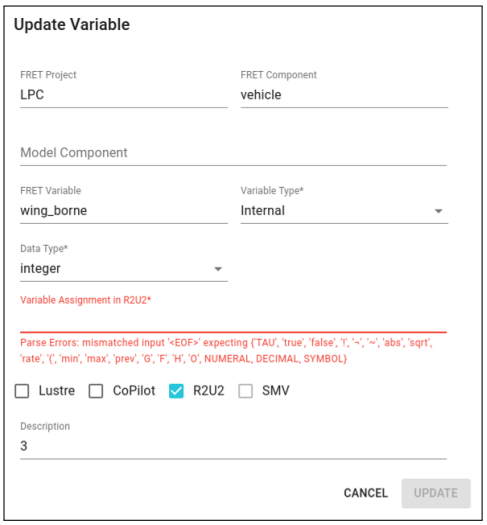

Fig. 9: FRET Update Variable Dialog Box

4.2 FRET Export

C2PO provides a structured specification language with separate sections for defining inputs (i.e., INPUT), variables (i.e., DEFINE), and MLTL formulas (i.e., FTSPEC)

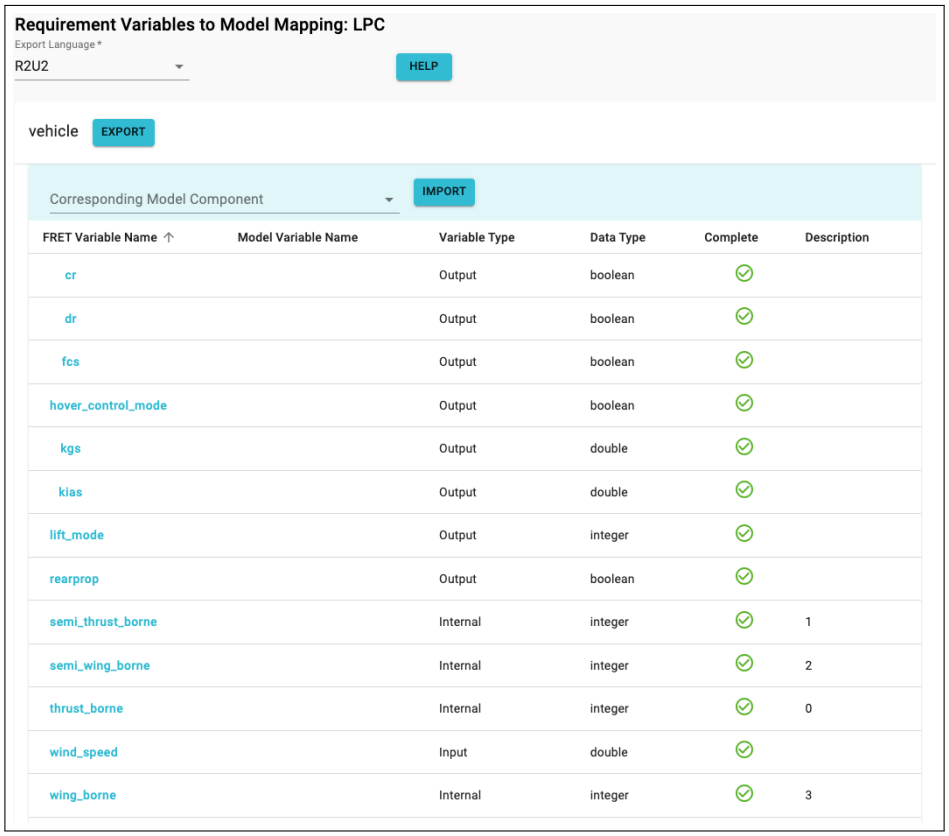

Fig. 10: FRET Variable Mapping Window

[35]. FRET generates a C2PO specification where the INPUT section includes the input/output variables mapped in Section 4.1, the DEFINE section includes the internal variables assigned in Section 4.1, and the FTSPEC section includes the MLTL formalizations in R2U2 format from Section 2.2. The C2PO specification for the LPC_SWB_TO_WB requirement is shown in Fig. 11. Additionally, for each MLTL formula that includes M in the temporal interval constraint of a temporal operator, a comment will appear before the formula in the C2PO specification that alerts the user of two things: **(1)** a large M value can result in high memory usage (refer to Fig. 7 to observe the effect of a large M) and **(2)** M must be manually specified. (Refer to Appendix B for an example of this comment.)

When R2U2 executes, the input data stream is consumed by R2U2 as a vector at each execution step. FRET generates an example CSV trace header (Fig. 12) and map file (Fig. 13) that indicate the mapping of this input data stream vector to C2PO.[7] After FRET exports these files, C2PO can encode the C2PO specification and an input vector mapping (i.e., provided by the CSV trace header or map file) into a R2U2 configuration binary. The R2U2 configuration binary will configure either

[7] Full FRET export for the Lift Plus Cruise case study [53,54] is available in Appendix B.

R2U2's C or Rust realization to produce the corresponding output verdicts for the defined MLTL formulas.

```
-- C2PO Specification Valid with C2PO/R2U2 v4.0 or greater
INPUT
  kias: float;
  lift_mode: int;
DEFINE
  semi_wing_borne := 2;
  wing_borne := 3;
FTSPEC
  LPC_SWB_TO_WB: ((((! ((lift_mode == semi_wing_borne) && (kias >
    100.0))) && (F[1,1] ((lift_mode == semi_wing_borne) && (kias >
    100.0)))) -> (F[2,2] (lift_mode == wing_borne))) && (((TAU == 0)
     && ((lift_mode == semi_wing_borne) && (kias > 100.0))) -> (F
    [1,1] (lift_mode == wing_borne))));
```

Fig. 11: C2PO Specification Exported by FRET

```
# kias, lift_mode
```

Fig. 12: Example CSV Trace Header Exported by FRET.

```
kias:0
lift_mode:1
```

Fig. 13: Example Map File Exported by FRET.

5 Conclusion

We extended FRET to provide a translation from FRETish natural language requirements to MLTL in both WEST and R2U2 format, minimized the resource overhead of these MLTL formalizations with SMT-proved rewrite rules, and provided an export that configures R2U2 runtime monitors for deployment on resource-constrained systems. In the future, we plan to extend FRET to provide formalizations in ptMLTL and, with FRET's recent support of probabilistic FRETish requirements [49], predictive MLTL (PMLTL) [3] to configure R2U2 monitors over past-time and probabilistic properties, respectively. We also seek to improve the rewriting of MLTL formula further by applying techniques such as equality saturation [63].

References

1. Aurandt, A., Jones, P.H., Rozier, K.Y.: Runtime verification triggers real-time, autonomous fault recovery on the CySat-I. In: NASA Formal Methods Symposium. pp. 816–825. Springer (2022). *https://doi.org/10.1007/978-3-031-06773-0_45*
2. Aurandt, A., Jones, P.H., Rozier, K.Y.: Towards a safe, verified runtime monitor for embedded systems: R2U2 in embedded rust. In: NASA Formal Methods Symposium. pp. 31–53. Springer (2025). *https://doi.org/10.1007/978-3-031-93706-4_3*
3. Aurandt, A., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Multimodal model predictive runtime verification for safety of autonomous cyber-physical systems. In: International Conference on Formal Methods for Industrial Critical Systems. pp. 220–244. Springer (2024). *https://doi.org/10.1007/978-3-031-68150-9_13*
4. Aurandt, A., Rozier, K.Y., Jones, P.H.: R2U2 playground: Visualization of a real-time, temporal logic runtime monitor. In: Formal Methods in Computer Aided Design. pp. 126–132. TU Wien Academic Press (2025). *https://doi.org/10.34727/2025/isbn.978-3-85448-084-6_18*
5. Autili, M., Grunske, L., Lumpe, M., Pelliccione, P., Tang, A.: Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar. IEEE Transactions on Software Engineering **41**(7), 620–638 (2015). *https://doi.org/10.1109/TSE.2015.2398877*

6. Barbosa, H., Barrett, C., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., et al.: cvc5: A versatile and industrial-strength smt solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 415–442. Springer (2022)
7. Barrett, C., Stump, A., Tinelli, C., et al.: The smt-lib standard: Version 2.0. In: Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK). vol. 13, p. 14 (2010)
8. Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Ničković, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. Lectures on Runtime Verification: Introductory and Advanced Topics pp. 135–175 (2018). *https://doi.org/10.1007/978-3-319-75632-5_5*
9. Bauer, A., Leucker, M.: The theory and practice of SALT. In: NASA Formal Methods Symposium. pp. 13–40. Springer (2011). *https://doi.org/10.1007/978-3-642-20398-5_3*
10. Becker, S., Dietsch, D., Hauff, N., Henkel, E., Langenfeld, V., Podelski, A., Westphal, B.: Hanfor: Semantic requirements review at scale. In: REFSQ Workshops. vol. 2857, p. 5 (2021)
11. Bozzano, M., Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: nuXmv 2.1.0 user manual. Fondazione Bruno Kessler, Tech. Rept., Trento, Italy (2024)
12. Cauwels, M., Hammer, A., Hertz, B., Jones, P., Rozier, K.Y.: Integrating runtime verification into an automated UAS traffic management system. In: DETECT. Springer, L'Aquila, Italy (September 2020). *https://doi.org/10.1007/978-3-030-59155-7_26*
13. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv symbolic model checker. In: International Conference on Computer Aided Verification. pp. 334–342. Springer (2014). *https://doi.org/10.1007/978-3-319-08867-9_22*
14. Conrad, E., Titolo, L., Giannakopoulou, D., Pressburger, T., Dutle, A.: A compositional proof framework for FRETish requirements. In: Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs. pp. 68–81 (2022). *https://doi.org/10.1145/3497775.3503685*
15. Crapo, A., Moitra, A., McMillan, C., Russell, D.: Requirements capture and analysis in ASSERT (TM). In: 2017 IEEE 25th International Requirements Engineering Conference (RE). pp. 283–291. IEEE (2017). *https://doi.org/10.1109/RE.2017.54*
16. Dabney, J.B., Badger, J.M., Rajagopal, P.: Adding a verification view for an autonomous real-time system architecture. In: Proceedings of SciTech Forum. p. Online. 2021-0566, AIAA (January 2021). *https://doi.org/10.2514/6.2021-0566*
17. Dabney, J.B., Badger, J.M., Rajagopal, P.: Trustworthy autonomy for gateway vehicle system manager. In: 2023 IEEE Space Computing Conference (SCC). pp. 57–62. IEEE (2023). *https://doi.org/10.1109/SCC57168.2023.00018*
18. Dabney, J.B.: Using assume-guarantee contracts in autonomous spacecraft. Flight Software Workshop (FSW) Online: *https://www.youtube.com/watch?v=zrtyiyNf674* (February 2021)
19. Dabney, J.B., Rajagopal, P., Badger, J.M.: Using assume-guarantee contracts for developmental verification of autonomous spacecraft. Flight Software Workshop (FSW) Online: *https://www.youtube.com/watch?v=HFnn6TzblPg* (February 2022)
20. Duffy, D.A.: Principles of automated theorem proving. John Wiley & Sons, Inc. (1991)
21. Ehlers, R.: Efficient temporal logic runtime monitoring for tiny systems. In: International Conference on Tests and Proofs. pp. 3–21. Springer (2024), *https://www.ruediger-ehlers.de/papers/tap2024.pdf*

22. Elwing, J., Gamboa-Guzman, L., Sorkin, J., Travesset, C., Wang, Z., Rozier, K.Y.: Mission-time LTL (MLTL) formula validation via regular expressions. In: International Conference on Integrated Formal Methods. pp. 279–301. Springer (2023). *https://doi.org/10.1007/978-3-031-47705-8_15*
23. Ferreira, R.R.: Runtime verification of low-level software in a vtol uav (2025), *https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/263713/Runtime_Verification_of_Low-Level_Software_in_a_VTOL_UAV.pdf*
24. Fifarek, A.W., Wagner, L.G., Hoffman, J.A., Rodes, B.D., Aiello, M.A., Davis, J.A.: SpeAR v2.0: Formalized past LTL specification and analysis of requirements. In: NASA Formal Methods Symposium. pp. 420–426. Springer (2017). *https://doi.org/10.1007/978-3-319-57288-8_30*
25. Fisher, M., Mascardi, V., Rozier, K.Y., Schlingloff, B.H., Winikoff, M., Yorke-Smith, N.: Towards a framework for certification of reliable autonomous systems. Autonomous Agents and Multi-Agent Systems **35**, 1–65 (2021). *https://doi.org/10.1007/s10458-020-09487-2*
26. Geist, J., Rozier, K.Y., Schumann, J.: Runtime observer pairs and bayesian network reasoners on-board fpgas: flight-certifiable system health management for embedded systems. In: International Conference on Runtime Verification. pp. 215–230. Springer (2014). *https://doi.org/10.1007/978-3-319-11164-3_18*
27. Giannakopoulou, D., Mavridou, A., Rhein, J., Pressburger, T., Schumann, J., Shi, N.: Formal requirements elicitation with FRET. In: International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ-2020) (2020)
28. Giannakopoulou, D., Pressburger, T., Mavridou, A., Schumann, J.: Automated formalization of structured natural language requirements. Information and Software Technology **137**, 106590 (2021). *https://doi.org/10.1016/j.infsof.2021.106590*
29. Goodloe, A.: Challenges in high-assurance runtime verification. In: International Symposium on Leveraging Applications of Formal Methods. pp. 446–460. Springer (2016). *https://doi.org/10.1007/978-3-319-47166-2_31*
30. Greenman, B., Saarinen, S., Nelson, T., Krishnamurthi, S.: Little tricky logic: misconceptions in the understanding of ltl. arXiv preprint arXiv:2211.01677 (2022)
31. Hariharan, G., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Maximum satisfiability of mission-time linear temporal logic. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 86–104. Springer (2023). *https://doi.org/10.1007/978-3-031-42626-1_6*
32. Hertz, B., Luppen, Z., Rozier, K.Y.: Integrating runtime verification into a sounding rocket control system. In: NASA Formal Methods Symposium. pp. 151–159. Springer (2021). *https://doi.org/10.1007/978-3-030-76384-8_10*
33. Jeannet, B., Gaucher, F.: Debugging embedded systems requirements with STIMULUS: an automotive case-study. In: 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016) (2016)
34. Johannsen, C., Anderson, M., Burken, W., Diersen, E., Edgren, J., Glick, C., Jou, S., Kumar, A., Levandowski, J., Moyer, E., et al.: Openuas version 1.0. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1449–1458. IEEE (2021). *https://doi.org/10.1109/ICUAS51884.2021.9476814*
35. Johannsen, C., Jones, P., Kempa, B., Rozier, K.Y., Zhang, P.: R2U2 Version 3.0: Reimagining a toolchain for specification, resource estimation, and optimized observer generation for runtime verification in hardware and software. In: International Conference on Computer Aided Verification. pp. 483–497. Springer (2023). *https://doi.org/10.1007/978-3-031-37709-9_23*

36. Johannsen, C., Kempa, B., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Impossible made possible: Encoding intractable specifications via implied domain constraints. In: International Conference on Formal Methods for Industrial Critical Systems. pp. 151–169. Springer (2023). *https://doi.org/10.1007/978-3-031-43681-9_9*
37. Johannsen, C., Jones, P.H., Rozier, K.Y., Wongpiromsarn, T.: Scalable MLTL Runtime Monitoring and Satisfiability via Bit-Vector Encoding. In: Formal Methods in Computer-Aided Design (FMCAD). TU Wien Academic Press (2025). *https://doi.org/10.34727/2025/isbn.978-3-85448-084-6_9*
38. Johannsen, C.G.: Dynamic set reasoning: Specifying and optimizing monitor encodings. Master's thesis, Iowa State University (2024)
39. Katis, A., Mavridou, A., Giannakopoulou, D., Pressburger, T., Schumann, J.: Capture, analyze, diagnose: Realizability checking of requirements in FRET. In: International Conference on Computer Aided Verification. pp. 490–504. Springer (2022). *https://doi.org/10.1007/978-3-031-13188-2_24*
40. Katis, A., Mavridou, A., Pressburger, T.: A streamlined, formal approach to requirements-based testing. In: NASA Formal Methods Symposium. pp. 159–179. Springer (2025). *https://doi.org/10.1007/978-3-031-93706-4_10*
41. Kempa, B., Zhang, P., Jones, P.H., Zambreno, J., Rozier, K.Y.: Embedding online runtime verification for fault disambiguation on Robonaut2. In: Proceedings of the 18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 196–214. Lecture Notes in Computer Science (LNCS), Springer, Vienna, Austria (September 2020). *https://doi.org/10.1007/978-3-030-57628-8_12*
42. Konrad, S., Cheng, B.H.: Automated analysis of natural language properties for uml models. In: International Conference on Model Driven Engineering Languages and Systems. pp. 48–57. Springer (2005). *https://doi.org/10.1007/11663430_6*
43. Li, J., Vardi, M.Y., Rozier, K.Y.: Satisfiability checking for mission-time LTL. In: International Conference on Computer Aided Verification. pp. 3–22. Springer (2019). *https://doi.org/10.1007/978-3-030-25543-5_1*
44. Li, R., Gurushankar, K., Heule, M.J., Rozier, K.Y.: What's in a name? linear temporal logic literally represents time lines. In: 2023 IEEE Working Conference on Software Visualization (VISSOFT). pp. 73–83. IEEE (2023). *https://doi.org/10.1109/VISSOFT60811.2023.00018*
45. Lorch, R., Meng, B., Siu, K., Moitra, A., Durling, M., Paul, S., Varanasi, S.C., McMillan, C.: Formal methods in requirements engineering: Survey and future directions. In: Proceedings of the 2024 IEEE/ACM 12th International Conference on Formal Methods in Software Engineering (FormaliSE). pp. 88–99 (2024). *https://doi.org/10.1145/3644033.3644373*
46. Luppen, Z., Jacks, M., Baughman, N., Stilic, M., Nasers, R., Hertz, B., Cutler, J., Lee, D.Y., Rozier, K.Y.: Elucidation and analysis of specification patterns in aerospace system telemetry. In: NASA Formal Methods Symposium. pp. 527–537. Springer (2022). *https://doi.org/10.1007/978-3-031-06773-0_28*
47. Luppen, Z.A., Lee, D.Y., Rozier, K.Y.: A case study in formal specification and runtime verification of a cubesat communications system. In: AIAA Scitech 2021 forum (2021). *https://doi.org/10.2514/6.2021-0997*
48. Mavridou, A., Katis, A., Giannakopoulou, D., Kooi, D., Pressburger, T., Whalen, M.W.: From partial to global assume-guarantee contracts: compositional realizability analysis in FRET. In: International Symposium on Formal Methods. pp. 503–523. Springer (2021). *https://doi.org/10.1007/978-3-030-90870-6_27*
49. Mavridou, A., Katis, A., Schumann, J., Pressburger, T., Flores, G.V.: FRET: 3.0. *https://github.com/NASA-SW-VnV/fret/releases/tag/v3.0.0* (2025), GitHub release

50. Nayak, A., Timmapathini, H.P., Murali, V., Ponnalagu, K., Venkoparao, V.G., Post, A.: Req2Spec: Transforming software requirements into formal specifications using natural language processing. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 87–95. Springer (2022). *https://doi.org/10.1007/978-3-030-98464-9_8*
51. Perez, I., Mavridou, A., Pressburger, T., Goodloe, A., Giannakopoulou, D.: Automated translation of natural language requirements to runtime monitors. In: International conference on tools and algorithms for the construction and analysis of systems. pp. 387–395. Springer (2022). *https://doi.org/10.1007/978-3-030-99524-9_21*
52. Prasad, S., Greenman, B., Nelson, T., Krishnamurthi, S.: A misconception-driven adaptive tutor for linear temporal logic. In: International Conference on Computer Aided Verification. pp. 185–200. Springer (2025). *https://doi.org/10.1007/978-3-031-98685-7_9*
53. Pressburger, T., Katis, A., Dutle, A., Mavridou, A.: Using fret to create, analyze and monitor requirements for a lift plus cruise case study. Tech. Rep. TM–20220017032, NASA (2023)
54. Pressburger, T., Katis, A., Dutle, A., Mavridou, A.: Authoring, analyzing, and monitoring requirements for a lift-plus-cruise aircraft. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 295–308. Springer (2023). *https://doi.org/10.1007/978-3-031-29786-1_21*
55. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science (LNCS), vol. 8413, pp. 357–372. Springer-Verlag (April 2014). *https://doi.org/10.1007/978-3-642-54862-8_24*
56. Rozier, K.Y.: Specification: The biggest bottleneck in formal methods and autonomy. In: Verified Software. Theories, Tools, and Experiments: 8th International Conference, VSTTE 2016, Toronto, ON, Canada, July 17–18, 2016. pp. 8–26. Springer (2016). *https://doi.org/10.1007/978-3-319-48869-1_2*
57. Rozier, K.Y., Schumann, J.: R2U2: Tool overview. In: Proceedings of International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES). vol. 3, pp. 138–156. Kalpa Publications, Seattle, WA, USA (September 2017), *https://easychair.org/publications/paper/Vncw*
58. Schumann, J., Moosbrugger, P., Rozier, K.Y.: R2U2: monitoring and diagnosis of security threats for unmanned aerial systems. In: Runtime Verification: 6th International Conference, RV 2015, Vienna, Austria, September 22-25, 2015. Proceedings. pp. 233–249. Springer (2015). *https://doi.org/10.1007/978-3-319-23820-3_15*
59. Schumann, J., Moosbrugger, P., Rozier, K.Y.: Runtime analysis with R2U2: A tool exhibition report. In: Proceedings of the 16th International Conference on Runtime Verification (RV16). Springer-Verlag, Madrid, Spain (September 2016). *https://doi.org/10.1007/978-3-319-46982-9_35*
60. Schumann, J., Roychoudhury, I., Kulkarni, C.: Diagnostic reasoning using prognostic information for unmanned aerial systems. In: Annual Conference of the PHM Society. vol. 7 (2015). *https://doi.org/10.36001/phmconf.2015.v7i1.2548*
61. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. In: International Journal of Prognostics and Health Management (2015). *https://doi.org/10.36001/ijphm.2015.v6i1.2243*
62. Sljivo, I., Mavridou, A., Schumann, J., Perez, I., Vlastos, P.G., Carter, C.: Dynamic assurance of autonomous systems through ground control software. In: AIAA SCITECH 2024 Forum. p. 1208 (2024). *https://doi.org/10.2514/6.2024-1208*

63. Tate, R., Stepp, M., Tatlock, Z., Lerner, S.: Equality saturation: a new approach to optimization. In: Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 264–276 (2009)
64. Varanasi, S.C., Meng, B., Lorch, R., Moitra, A., Siu, K., Paul, S., Durling, M., Beniwal, N., Visnevski, N.: TRACE: Toolkit for requirements analysis, capture, and elicitation. In: NASA Formal Methods Symposium. pp. 380–399. Springer (2025). *https://doi.org/10.1007/978-3-031-93706-4_22*
65. Vázquez, G., Mavridou, A., Farrell, M., Pressburger, T., Calinescu, R.: Robotics: A new mission for FRET requirements. In: NASA Formal Methods Symposium. pp. 359–376. Springer (2024). *https://doi.org/10.1007/978-3-031-60698-4_22*
66. Wang, Z., Guzman, L.P.G., Rozier, K.Y.: WEST: Interactive validation of mission-time linear temporal logic (MLTL). Science of Computer Programming **248** (2025). *https://doi.org/10.1016/j.scico.2025.103365*
67. Wang, Z., Kosaian, K., Rozier, K.Y.: Formally verifying a transformation from MLTL formulas to regular expressions. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–275. Springer (2025). *https://doi.org/10.1007/978-3-031-90643-5_13*
68. Zhang, P., Aurandt, A., Dureja, R., Jones, P.H., Rozier, K.Y.: Model predictive runtime verification for cyber-physical systems with real-time deadlines. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 158–180. Springer (2023). *https://doi.org/10.1007/978-3-031-42626-1_10*

A Additional Rewrite Rules

Table 7: Rewrite Rules to Reduce When In/Not in `scope` (Part I)

Original	Rewrite	# Times Applied
$\varphi_1 \rightarrow (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg \varphi_1)))$	$\varphi_1 \rightarrow (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg \varphi_1))$	1
$\neg\varphi_1 \rightarrow (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)))$	$\neg\varphi_1 \rightarrow (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1))$	2
$(\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg \varphi_1)))$	$(\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg \varphi_1))$	2
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)))$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1))$	4
$\neg\varphi_1 \rightarrow$ $((\varphi_2 \vee (\mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)))) \vee \varphi_3)$	$\neg\varphi_1 \rightarrow$ $((\varphi_2 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1))) \vee \varphi_3)$	1
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $((\varphi_2 \vee (\mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)))) \vee \varphi_4)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $((\varphi_2 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1))) \vee \varphi_4)$	2
$\varphi_1 \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg \varphi_1))))$	$\varphi_1 \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg \varphi_1)))$	1
$(\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg \varphi_1))))$	$(\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg \varphi_1)))$	2
$\neg\varphi_1 \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))))$	$\neg\varphi_1 \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1)))$	1
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))))$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow$ $(\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \varphi_1)))$	2
$\neg\varphi_1 \rightarrow (\varphi_2 \vee (\mathbf{G}_{[0,M]}(\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)))))$	$\neg\varphi_1 \rightarrow (\varphi_2 \vee (\mathbf{G}_{[0,M]}(\mathbf{F}_{[1,1]} \neg \varphi_1)))$	33

Table 8: Rewrite Rules to Reduce When In/Not in `scope` (Part II)

Original	Rewrite	# Times Applied
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]}\varphi_1)))))$	2
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg\varphi_1)))))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_4 \wedge (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))))))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_4 \wedge (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]}\varphi_1))))))$	1
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_4 \wedge (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))))))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_4 \wedge (\varphi_3 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]} \neg\varphi_1))))))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[0,ub-1]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \vee \varphi_2)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[0,ub-1]}(\mathbf{F}_{[1,1]}\varphi_1))) \vee \varphi_2)))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1))$	2
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))))$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1)))$	2
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]} \neg\varphi_1))$	1
$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))))$	$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]} \neg\varphi_1)))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1)))$	1
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow (\varphi_2 \wedge (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))))$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow (\varphi_2 \wedge (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1))))$	1
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_4 \wedge (\varphi_2 \vee \mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]} \neg\varphi_1)))$	1
$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow (\varphi_2 \wedge (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))))$	$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow (\varphi_2 \wedge (\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]} \neg\varphi_1))))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \vee \varphi_3)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1))) \vee \varphi_3)$	1
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \vee \varphi_2)$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[1,ub]}(\mathbf{F}_{[1,1]}\varphi_1))) \vee \varphi_2)$	1
$\varphi_1 \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))\mathbf{U}_{[0,M]}\varphi_2)$	$\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[0,M]}\varphi_2)$	2
$(\varphi_1 \wedge \varphi_3) \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))\mathbf{U}_{[0,M]}\varphi_2)$	$(\varphi_1 \wedge \varphi_3) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[0,M]}\varphi_2)$	2
$\neg\varphi_1 \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[0,M]}\varphi_2)$	$\neg\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{U}_{[0,M]}\varphi_2)$	4
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[0,M]}\varphi_2)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{U}_{[0,M]}\varphi_2)$	4
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[1,ub]}\varphi_2)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{U}_{[1,ub]}\varphi_2)$	4
$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))) \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[1,ub]}\varphi_4)$	$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))) \rightarrow ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{U}_{[1,ub]}\varphi_4)$	8

Table 9: Rewrite Rules to Reduce When In/Not in `scope` (Part III)

Original	Rewrite	# Times Applied
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{U}_{[1,ub]}\varphi_2)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[1,ub]}\varphi_2)$	2
$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\varphi_1))) \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{U}_{[1,ub]}\varphi_4)$	$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\varphi_1))) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[1,ub]}\varphi_4)$	4
$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{U}_{[0,M]}\varphi_4)))$	$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[0,M]}\varphi_4)))$	2
$\neg\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[0,M]}\varphi_4)))$	$\neg\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{U}_{[0,M]}\varphi_4)))$	4
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[0,M]}\varphi_4))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{U}_{[0,M]}\varphi_4))))$	4
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{U}_{[0,M]}\varphi_4))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{U}_{[0,M]}\varphi_4))))$	2
$\neg\varphi_1 \rightarrow (((\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{U}_{[0,M]} ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \wedge \varphi_3)) \vee \varphi_2)$	$\neg\varphi_1 \rightarrow (((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{U}_{[0,M]} ((\mathbf{F}_{[1,1]}\varphi_1) \wedge \varphi_3)) \vee \varphi_2)$	33
$\varphi_1 \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_2)$	$\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,M]}\varphi_2)$	13
$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_2)$	$(\varphi_1 \wedge \varphi_3) \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,M]}\varphi_2)$	3
$\neg\varphi_1 \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_2)$	$\neg\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,M]}\varphi_2)$	27
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_2)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,M]}\varphi_2)$	6
$\varphi_1 \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_3) \wedge \varphi_4)$	$\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,M]}\varphi_3) \wedge \varphi_4)$	12
$\neg\varphi_1 \rightarrow (((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_3) \wedge \varphi_4)$	$\neg\varphi_1 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,M]}\varphi_3) \wedge \varphi_4)$	24
$\varphi_1 \rightarrow ((\varphi_2 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{R}_{[0,M]}\varphi_3)$	$\varphi_1 \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_3)$	2
$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{R}_{[0,M]}\varphi_4)$	$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_4)$	4
$\neg\varphi_1 \rightarrow ((\varphi_2 \vee (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[0,M]}\varphi_3)$	$\neg\varphi_1 \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_3)$	3
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[0,M]}\varphi_4)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_4)$	6
$\neg\varphi_1 \rightarrow ((\varphi_2 \rightarrow (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[0,M]}\varphi_3)$	$\neg\varphi_1 \rightarrow ((\varphi_2 \rightarrow (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_3)$	1
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \rightarrow (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[0,M]}\varphi_4)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \rightarrow (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_4)$	2
$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,M]}\varphi_4)))$	$\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,M]}\varphi_4)))$	3
$\neg\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,M]}\varphi_4)))$	$\neg\varphi_1 \rightarrow (\varphi_2 \wedge (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,M]}\varphi_4)))$	6

Table 10: Rewrite Rules to Reduce When In/Not in `scope` (Part IV)

Original	Rewrite	# Times Applied
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,M]} \varphi_4))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,M]} \varphi_4))))$	6
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,M]} \varphi_4))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \rightarrow (\varphi_2 \wedge (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,M]} \varphi_4))))$	3
$\varphi_1 \rightarrow (((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,M]} ((\varphi_2 \wedge (\varphi_3 \wedge (\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))))) \rightarrow \varphi_4)) \wedge \varphi_5)$	$\varphi_1 \rightarrow (((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,M]} ((\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]} \varphi_1))) \rightarrow \varphi_4)) \wedge \varphi_5)$	12
$\neg\varphi_1 \rightarrow (((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,M]} ((\varphi_2 \wedge (\varphi_3 \wedge (\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))))) \rightarrow \varphi_4)) \wedge \varphi_5)$	$\neg\varphi_1 \rightarrow (((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,M]} ((\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))) \rightarrow \varphi_4)) \wedge \varphi_5)$	24
$\varphi_1 \rightarrow (\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,ub]} \varphi_3))$	$\varphi_1 \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,ub]} \varphi_3))$	1
$(\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,ub]} \varphi_4))$	$(\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,ub]} \varphi_4))$	2
$\neg\varphi_1 \rightarrow (\varphi_2 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,ub]} \varphi_3))$	$\neg\varphi_1 \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_3))$	2
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,ub]} \varphi_4))$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_4))$	4
$\neg\varphi_1 \rightarrow (\varphi_2 \vee (\varphi_3 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,ub]} \varphi_4)))$	$\neg\varphi_1 \rightarrow (\varphi_2 \vee (\varphi_3 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_4)))$	1
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee (\varphi_4 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,ub]} \varphi_5)))$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow (\varphi_2 \vee (\varphi_4 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_5)))$	2
$\varphi_1 \rightarrow ((\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,ub]} \varphi_3)) \wedge \varphi_4)$	$\varphi_1 \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,ub]} \varphi_3)) \wedge \varphi_4)$	1
$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,ub]} \varphi_4)) \wedge \varphi_5)$	$(\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1) \mathbf{R}_{[0,ub]} \varphi_4)) \wedge \varphi_5)$	2
$\neg\varphi_1 \rightarrow ((\varphi_2 \vee (((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))) \mathbf{R}_{[0,ub]} \varphi_3)) \wedge \varphi_4)$	$\neg\varphi_1 \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_3)) \wedge \varphi_4)$	1
$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee (((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))) \mathbf{R}_{[0,ub]} \varphi_4)) \wedge \varphi_5)$	$(\neg\varphi_1 \wedge \varphi_3) \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_4)) \wedge \varphi_5)$	2
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\varphi_2 \rightarrow (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1))) \mathbf{R}_{[1,ub]} \varphi_3)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow ((\varphi_2 \rightarrow (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[1,ub]} \varphi_3)$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \vee \neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,M]} \varphi_4)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,M]} \varphi_4)))$	3
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))) \mathbf{R}_{[0,M]} \varphi_4)))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]} \neg\varphi_1)) \mathbf{R}_{[0,M]} \varphi_4)))$	2
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \rightarrow \neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,M]} \varphi_4)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_3 \rightarrow ((\varphi_2 \rightarrow (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,M]} \varphi_4)))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_2 \rightarrow (\varphi_3 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]} \varphi_1)) \mathbf{R}_{[0,ub]} \varphi_4))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]} (\varphi_2 \rightarrow (\varphi_3 \vee ((\mathbf{F}_{[1,1]} \varphi_1) \mathbf{R}_{[0,ub]} \varphi_4))))$	2

Table 11: Rewrite Rules to Reduce When In/Not in `scope` (Part V)

Original	Rewrite	# Times Applied
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,ub]}\varphi_4)))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,ub]}\varphi_4))))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\varphi_5 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,ub]}\varphi_4)))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow (\varphi_3 \vee (\varphi_5 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,ub]}\varphi_4)))))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow ((\varphi_3 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[0,ub]}\varphi_4)) \wedge \varphi_6)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow ((\varphi_3 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[0,ub]}\varphi_4)) \wedge \varphi_6)))$	1
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow ((\varphi_3 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[0,ub]}\varphi_4)) \wedge \varphi_6)))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\mathbf{F}_{[1,1]}(\varphi_2 \rightarrow ((\varphi_3 \vee ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[0,ub]}\varphi_4)) \wedge \varphi_6)))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2)$	27
$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))) \rightarrow ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_4)$	$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_4)$	12
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2)$	13
$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\varphi_1))) \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_4)$	$(\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\varphi_1))) \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_4)$	6
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3))$	2
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (\varphi_2 \vee ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3) \wedge \varphi_4)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3) \wedge \varphi_4)$	24
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3) \wedge \varphi_4)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3) \wedge \varphi_4)$	12
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}((\varphi_2 \wedge (\varphi_3 \wedge (\neg(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))))) \rightarrow \varphi_4)) \wedge \varphi_5)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow (((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}((\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\varphi_1))) \rightarrow \varphi_4)) \wedge \varphi_5)$	12
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}((\varphi_2 \wedge (\varphi_3 \wedge (\neg(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))))) \rightarrow \varphi_4)) \wedge \varphi_5)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}((\varphi_2 \wedge (\varphi_3 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))) \rightarrow \varphi_4)) \wedge \varphi_5)$	24
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\varphi_2 \vee (((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_3)) \wedge \varphi_4)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3)) \wedge \varphi_4)$	1
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_2 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3)) \wedge \varphi_4)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_2 \vee ((\mathbf{F}_{[1,1]}\neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_3)) \wedge \varphi_4)$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \vee (\varphi_3 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_4)))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow (\varphi_2 \vee (\varphi_3 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_4)))$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\varphi_2 \vee (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_3)$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3)$	3
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_2 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_3)$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow ((\varphi_2 \vee (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_3)$	2
$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow ((\varphi_5 \vee (\varphi_1 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_2)$	$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[1,1]}\neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)$	2
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))) \rightarrow ((\varphi_5 \vee (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_2)$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\neg\varphi_1)))) \rightarrow ((\varphi_5 \vee (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)$	3

Table 12: Rewrite Rules to Reduce When In/Not in `scope` (Part VI)

Original	Rewrite	# Times Applied
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $(\varphi_5 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2))$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $(\varphi_5 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2))$	2
$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow$ $(\varphi_5 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2))$	$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow$ $(\varphi_5 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2))$	1
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $((\varphi_5 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)) \wedge \varphi_6)$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $((\varphi_5 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2)) \wedge \varphi_6)$	1
$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow$ $((\varphi_5 \vee ((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)) \wedge \varphi_6)$	$(\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))) \rightarrow$ $((\varphi_5 \vee ((\mathbf{F}_{[1,1]} \neg\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2)) \wedge \varphi_6)$	1
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $(\varphi_5 \vee (\varphi_6 \vee ((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)))$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $(\varphi_5 \vee (\varphi_6 \vee ((\mathbf{F}_{[1,1]}\varphi_1)\mathbf{R}_{[1,ub]}\varphi_2)))$	1
$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $((\varphi_5 \rightarrow (\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)))\mathbf{R}_{[1,ub]}\varphi_2)$	$(\neg\varphi_1 \wedge (\varphi_3 \wedge (\varphi_4 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)))) \rightarrow$ $((\varphi_5 \rightarrow (\mathbf{F}_{[1,1]}\varphi_1))\mathbf{R}_{[1,ub]}\varphi_2)$	1
$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow$ $(\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow$ $((\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \vee \varphi_4))))$	$(\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \rightarrow$ $(\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow$ $((\mathbf{F}_{[1,1]}\varphi_1) \vee \varphi_4))))$	2
$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow$ $(\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow$ $((\varphi_1 \wedge (\mathbf{F}_{[1,1]} \neg\varphi_1)) \vee \varphi_4))))$	$(\neg\varphi_1 \wedge (\mathbf{F}_{[1,1]}\varphi_1)) \rightarrow$ $(\varphi_2 \wedge (\mathbf{F}_{[1,1]}(\varphi_3 \rightarrow$ $((\mathbf{F}_{[1,1]} \neg\varphi_1) \vee \varphi_4))))$	1

B Lift Plus Cruise (LPC) Case Study Output

The following figures include the files generated from FRET after the R2U2 export button is selected for the Lift Plus Cruise case study [53,54].

```
-- C2PO Specification Valid with C2PO/R2U2 v4.0 or greater
INPUT
  cr: bool;
  dr: bool;
  fcs: bool;
  hover_control_mode: bool;
  kgs: float;
  kias: float;
  lift_mode: int;
  rearprop: bool;
  wind_speed: float;
DEFINE
  semi_thrust_borne := 1;
  semi_wing_borne := 2;
  thrust_borne := 0;
  wing_borne := 3;
FTSPEC
  LPC_TB_STAY_ON_NEXT: ((((lift_mode == thrust_borne) && (kgs <=
    20.0)) && hover_control_mode) -> (F[1,1] (lift_mode ==
    thrust_borne)));
  LPC_STB_STAY_ON_NEXT: ((((lift_mode == semi_thrust_borne) && (kias
     <= 40.0)) && ((kgs > 20.0) || (! hover_control_mode))) -> (F
    [1,1] (lift_mode == semi_thrust_borne)));
  LPC_REACH_HOVER_13: ((TAU == 0) -> (F[0,13] (lift_mode ==
    thrust_borne)));
  LPC_WB_STAY_ON_pre: (prev(false,((lift_mode == wing_borne) && (
    kias > 90.0))) -> (lift_mode == wing_borne));
  LPC_REACH_NOT_FCS_10: ((TAU == 0) -> (F[0,10] (! fcs)));
  LPC_REACH_HOVER_14: ((TAU == 0) -> (F[0,14] (lift_mode ==
    thrust_borne)));
  LPC_CR_STAY_OFF: (prev(false,((! cr) && (kias >= 90.0))) -> (! cr)
    );
  LPC_SWB_STAY_ON_pre: (prev(false,(((lift_mode == semi_wing_borne)
    && (kias <= 100.0)) && (kias > 30.0))) -> (lift_mode ==
    semi_wing_borne));
  LPC_REACH_HOVER_15: ((TAU == 0) -> (F[0,15] (lift_mode ==
    thrust_borne)));
  LPC_FCS_TURN_OFF: ((((! (fcs && (kias < 30.0))) && (F[1,1] (fcs &&
     (kias < 30.0)))) -> (F[2,2] (! fcs))) && (((TAU == 0) && (fcs
    && (kias < 30.0))) -> (F[1,1] (! fcs))));
  LPC_DR_STAY_OFF: (prev(false,((! dr) && (kias >= 60.0))) -> (! dr)
    );
  LPC_REACH_HOVER_10: ((TAU == 0) -> (F[0,10] (lift_mode ==
    thrust_borne)));
  LPC_REACH_HOVER_12: ((TAU == 0) -> (F[0,12] (lift_mode ==
    thrust_borne)));
  LPC_REACH_HOVER_11: ((TAU == 0) -> (F[0,11] (lift_mode ==
    thrust_borne)));
  LPC_REARPROP: (rearprop xor hover_control_mode);
```

Fig. 14: C2PO Specification Exported by FRET (Part I)

```
33  LPC_SWB_TO_STB: ((((! ((lift_mode == semi_wing_borne) && (kias <=
      30.0))) && (F[1,1] ((lift_mode == semi_wing_borne) && (kias <=
      30.0)))) -> (F[2,2] (lift_mode == semi_thrust_borne))) && (((TAU
       == 0) && ((lift_mode == semi_wing_borne) && (kias <= 30.0))) ->
       (F[1,1] (lift_mode == semi_thrust_borne))));
34  LPC_INIT_FCS: ((TAU == 0) -> (fcs <-> (kias > 40.0)));
35  -- WARNING: The following spec (LPC_SWB_STAY_ON_until) includes M
      (i.e., end of mission-time) in a temporal interval.
36  -- If M is large, this specification may result in high memory
      usage.
37  -- Please specify M manually in this file or with the --mission-
      time flag in C2PO.
38  LPC_SWB_STAY_ON_until: ((((! (((lift_mode == semi_wing_borne) &&
      (30.0 < kias)) && (kias <= 100.0))) && (F[1,1] (((lift_mode ==
      semi_wing_borne) && (30.0 < kias)) && (kias <= 100.0)))) -> (((
      lift_mode == semi_wing_borne) && ((kias <= 30.0) || (kias >
      100.0))) R[1,M] ((lift_mode == semi_wing_borne) || ((lift_mode
      == semi_wing_borne) && ((kias <= 30.0) || (kias > 100.0)))))) &&
       (((TAU == 0) && (((lift_mode == semi_wing_borne) && (30.0 <
      kias)) && (kias <= 100.0))) -> (((lift_mode == semi_wing_borne)
      && ((kias <= 30.0) || (kias > 100.0))) R[0,M] ((lift_mode ==
      semi_wing_borne) || ((lift_mode == semi_wing_borne) && ((kias <=
       30.0) || (kias > 100.0)))))));
39  LPC_STB_TO_SWB: ((((! ((lift_mode == semi_thrust_borne) && (kias >
       40.0))) && (F[1,1] ((lift_mode == semi_thrust_borne) && (kias >
       40.0)))) -> (F[2,2] (lift_mode == semi_wing_borne))) && (((TAU
      == 0) && ((lift_mode == semi_thrust_borne) && (kias > 40.0))) ->
       (F[1,1] (lift_mode == semi_wing_borne))));
40  -- WARNING: The following spec (LPC_STB_STAY_ON_until) includes M
      (i.e., end of mission-time) in a temporal interval.
41  -- If M is large, this specification may result in high memory
      usage.
42  -- Please specify M manually in this file or with the --mission-
      time flag in C2PO.
43  LPC_STB_STAY_ON_until: ((((! (((lift_mode == semi_thrust_borne) &&
       (kias <= 40.0)) && (hover_control_mode -> (kgs > 20.0)))) && (F
      [1,1] (((lift_mode == semi_thrust_borne) && (kias <= 40.0)) && (
      hover_control_mode -> (kgs > 20.0))))) -> (((lift_mode ==
      semi_thrust_borne) && ((kias > 40.0) || ((kgs <= 20.0) &&
      hover_control_mode))) R[1,M] ((lift_mode == semi_thrust_borne)
      || ((lift_mode == semi_thrust_borne) && ((kias > 40.0) || ((kgs
      <= 20.0) && hover_control_mode)))))) && (((TAU == 0) && (((
      lift_mode == semi_thrust_borne) && (kias <= 40.0)) && (
      hover_control_mode -> (kgs > 20.0)))) -> (((lift_mode ==
      semi_thrust_borne) && ((kias > 40.0) || ((kgs <= 20.0) &&
      hover_control_mode))) R[0,M] ((lift_mode == semi_thrust_borne)
      || ((lift_mode == semi_thrust_borne) && ((kias > 40.0) || ((kgs
      <= 20.0) && hover_control_mode)))))));
```

Fig. 15: C2PO Specification Exported by FRET (Part II)

```
44  LPC_STB_TO_TB: ((((! (((lift_mode == semi_thrust_borne) &&
      hover_control_mode) && (kgs <= 20.0))) && (F[1,1] (((lift_mode
      == semi_thrust_borne) && hover_control_mode) && (kgs <= 20.0))))
       -> (F[2,2] (lift_mode == thrust_borne))) && (((TAU == 0) && (((
      lift_mode == semi_thrust_borne) && hover_control_mode) && (kgs
      <= 20.0))) -> (F[1,1] (lift_mode == thrust_borne))));
45  LPC_SWB_TO_WB: ((((! ((lift_mode == semi_wing_borne) && (kias >
      100.0))) && (F[1,1] ((lift_mode == semi_wing_borne) && (kias >
      100.0)))) -> (F[2,2] (lift_mode == wing_borne))) && (((TAU == 0)
       && ((lift_mode == semi_wing_borne) && (kias > 100.0))) -> (F
      [1,1] (lift_mode == wing_borne))));
46  LPC_INIT_DR: ((TAU == 0) -> (dr <-> (kias < 60.0)));
47  LPC_TB_TO_STB: ((((! ((lift_mode == thrust_borne) && (
      hover_control_mode -> (kgs > 20.0)))) && (F[1,1] ((lift_mode ==
      thrust_borne) && (hover_control_mode -> (kgs > 20.0))))) -> (F
      [2,2] (lift_mode == semi_thrust_borne))) && (((TAU == 0) && ((
      lift_mode == thrust_borne) && (hover_control_mode -> (kgs >
      20.0)))) -> (F[1,1] (lift_mode == semi_thrust_borne))));
48  LPC_INIT_LIFT_MODE: ((TAU == 0) -> ((lift_mode == wing_borne) <->
      (kias >= 90.0)));
49  LPC_LIFT_MODE: ((((lift_mode == thrust_borne) || (lift_mode ==
      semi_thrust_borne)) || (lift_mode == semi_wing_borne)) || (
      lift_mode == wing_borne));
50  LPC_INIT_KIAS: ((TAU == 0) -> (kias == 120.0));
51  -- WARNING: The following spec (LPC_TB_STAY_ON_until) includes M (
      i.e., end of mission-time) in a temporal interval.
52  -- If M is large, this specification may result in high memory
      usage.
53  -- Please specify M manually in this file or with the --mission-
      time flag in C2PO.
54  LPC_TB_STAY_ON_until: ((((! (((lift_mode == thrust_borne) &&
      hover_control_mode) && (kgs <= 20.0))) && (F[1,1] (((lift_mode
      == thrust_borne) && hover_control_mode) && (kgs <= 20.0)))) ->
      (((lift_mode == thrust_borne) && ((kgs > 20.0) || (!
      hover_control_mode))) R[1,M] ((lift_mode == thrust_borne) || ((
      lift_mode == thrust_borne) && ((kgs > 20.0) || (!
      hover_control_mode)))))) && (((TAU == 0) && (((lift_mode ==
      thrust_borne) && hover_control_mode) && (kgs <= 20.0))) -> (((
      lift_mode == thrust_borne) && ((kgs > 20.0) || (!
      hover_control_mode))) R[0,M] ((lift_mode == thrust_borne) || ((
      lift_mode == thrust_borne) && ((kgs > 20.0) || (!
      hover_control_mode)))))));
```

Fig. 16: C2PO Specification Exported by FRET (Part III)

```
55  -- WARNING: The following spec (LPC_WB_STAY_ON_until) includes M (
      i.e., end of mission-time) in a temporal interval.
56  -- If M is large, this specification may result in high memory
      usage.
57  -- Please specify M manually in this file or with the --mission-
      time flag in C2PO.
58  LPC_WB_STAY_ON_until: ((((! ((lift_mode == wing_borne) && (kias >
      90.0))) && (F[1,1] ((lift_mode == wing_borne) && (kias > 90.0)))
      ) -> (((lift_mode == wing_borne) && (kias <= 90.0)) R[1,M] ((
      lift_mode == wing_borne) || ((lift_mode == wing_borne) && (kias
      <= 90.0))))) && (((TAU == 0) && ((lift_mode == wing_borne) && (
      kias > 90.0))) -> (((lift_mode == wing_borne) && (kias <= 90.0))
       R[0,M] ((lift_mode == wing_borne) || ((lift_mode == wing_borne)
       && (kias <= 90.0))))));
59  LPC_KIAS_KGS: (kias == kgs);
60  LPC_INIT_CR: ((TAU == 0) -> (cr <-> (kias < 90.0)));
61  LPC_KIAS_DERIVATIVE: ((TAU == 0) || (abs((prev(0.0,kias) - kias))
      <= 10.0));
62  LPC_WB_TO_SWB: ((((! ((lift_mode == wing_borne) && (kias <= 90.0))
      ) && (F[1,1] ((lift_mode == wing_borne) && (kias <= 90.0)))) ->
      (F[2,2] (lift_mode == semi_wing_borne))) && (((TAU == 0) && ((
      lift_mode == wing_borne) && (kias <= 90.0))) -> (F[1,1] (
      lift_mode == semi_wing_borne))));
63  LPC_REACH_HOVER_06: ((TAU == 0) -> (F[0,6] (lift_mode ==
      thrust_borne)));
64  LPC_INIT_HOVER_MODE: ((TAU == 0) -> (hover_control_mode <-> (kgs <
       20.0)));
65  LPC_FCS_STAY_OFF: (prev(false,((! fcs) && (kias <= 40.0))) -> (!
      fcs));
66  LPC_CR_STAY_ON: (prev(false,(cr && (kias <= 100.0))) -> cr);
67  LPC_INIT_WIND_SPEED_assumption: ((TAU == 0) -> (wind_speed ==
      10.0));
68  LPC_DR_STAY_ON: (prev(false,(dr && (kias <= 70.0))) -> dr);
69  LPC_REACH_HOVER_16: ((TAU == 0) -> (F[0,16] (lift_mode ==
      thrust_borne)));
70  LPC_WIND_SPEED_30_assumption: ((-30.0 <= wind_speed) && (
      wind_speed <= 30.0));
71  LPC_FCS_TURN_ON: ((((! ((! fcs) && (kias > 40.0))) && (F[1,1] ((!
      fcs) && (kias > 40.0)))) -> (F[2,2] fcs)) && (((TAU == 0) && ((!
       fcs) && (kias > 40.0))) -> (F[1,1] fcs)));
72  LPC_CR_TURN_ON: ((((! ((! cr) && (kias < 90.0))) && (F[1,1] ((! cr
      ) && (kias < 90.0)))) -> (F[2,2] cr)) && (((TAU == 0) && ((! cr)
       && (kias < 90.0))) -> (F[1,1] cr)));
73  LPC_KIAS_KGS_WIND_SPEED: (kgs == (kias + wind_speed));
74  LPC_FCS_STAY_ON: (prev(false,(fcs && (kias >= 30.0))) -> fcs);
75  LPC_CR_TURN_OFF: ((((! (cr && (kias > 100.0))) && (F[1,1] (cr && (
      kias > 100.0)))) -> (F[2,2] (! cr))) && (((TAU == 0) && (cr && (
      kias > 100.0))) -> (F[1,1] (! cr))));
76  LPC_WIND_SPEED_DERIV_assumption: ((TAU == 0) || (abs((prev(0.0,
      wind_speed) - wind_speed)) <= 10.0));
77  LPC_KIAS_0: (kias >= 0.0);
78  LPC_REACH_NOT_FCS_11: ((TAU == 0) -> (F[0,11] (! fcs)));
```

Fig. 17: C2PO Specification Exported by FRET (Part IV)

```
79   LPC_STB_STAY_ON_pre: (prev(false,(((lift_mode == semi_thrust_borne
       ) && (kias <= 40.0)) && ((kgs > 20.0) || (! hover_control_mode))
       )) -> (lift_mode == semi_thrust_borne));
80   LPC_TB_STAY_ON_pre: (prev(false,(((lift_mode == thrust_borne) && (
       kgs <= 20.0)) && hover_control_mode)) -> (lift_mode ==
       thrust_borne));
81   LPC_WIND_SPEED_20_assumption: (abs(wind_speed) <= 20.0);
82   LPC_DR_TURN_ON: ((((! ((! dr) && (kias < 60.0))) && (F[1,1] ((! dr
       ) && (kias < 60.0)))) -> (F[2,2] dr)) && (((TAU == 0) && ((! dr)
        && (kias < 60.0))) -> (F[1,1] dr)));
83   LPC_DR_TURN_OFF: ((((! (dr && (kias > 70.0))) && (F[1,1] (dr && (
       kias > 70.0)))) -> (F[2,2] (! dr))) && (((TAU == 0) && (dr && (
       kias > 70.0))) -> (F[1,1] (! dr))));
84   LPC_REACH_NOT_FCS_09: ((TAU == 0) -> (F[0,9] (! fcs)));
85   LPC_SWB_STAY_ON_NEXT: ((((lift_mode == semi_wing_borne) && (kias
       <= 100.0)) && (kias > 30.0)) -> (F[1,1] (lift_mode ==
       semi_wing_borne)));
86   LPC_WB_STAY_ON_NEXT: (((lift_mode == wing_borne) && (kias > 90.0))
        -> (F[1,1] (lift_mode == wing_borne)));
```

Fig. 18: C2PO Specification Exported by FRET (Part V)

```
1 # cr, dr, fcs, hover_control_mode, kgs, kias, lift_mode, rearprop,
      wind_speed
```

Fig. 19: Example CSV Header File Exported by FRET

```
1 cr:0
2 dr:1
3 fcs:2
4 hover_control_mode:3
5 kgs:4
6 kias:5
7 lift_mode:6
8 rearprop:7
9 wind_speed:8
```

Fig. 20: Example Map File Exported by FRET