# ARTiMon Monitoring Tool
# The Hybrid Engine Cycles Benchmark

Nicolas RAPIN

CEA LIST
Point Courrier 174, Gif sur Yvette, F-91191 France
nicolas.rapin@cea.fr

## 1 The Hybrid Engine Case Study

First we would like thank the *Sherpa Engineering* company, provider of this case study, which allowed us to communicate over its treatment with our monitoring tool called ARTiMon [1,2]. The system under monitoring is an hybrid engine of an individual car. The trace provided is in csv format. The first column is for time stamps, next columns are for the state variables values. Notice that times stamps are not regularly spaced (most are separated by a time delta of 0.1 but we voluntary inserted some other time spaces in order to increase the interest of this benchmark). The state variables of the system under monitoring are the followings:

| State Variables | type | Commentary |
|---|---|---|
| cycleid | int | Is an integer identifying the current cycle |
| distance | double | The distance accomplished by the vehicle |
| elecpower | double | Electric Power |
| socelec | double | Charge of the battery (between 0 and 1) |
| fuelpower | double | Power from the fuel engine |
| socfuel | double | Fuel tank level (between 0 and 1) |
| speed | double | Vehicle Speed |

It is also characterized by some constants (typed double):

| Constant Name | Value | Commentary |
|---|---|---|
| vFuel | 45.0 | Fuel Tank Capacity |
| cBat | 1602180.0 | Battery Recharging Rate |
| aFC | 2.5 | Desired Average Fuel Consumption |
| ms | 131.3 | Max Speed |
| br | $-1.7$ | Battery Recharge |

We note $s(t)$ the value of the state variable $s$ at time $t$. Given a cycle $C \in \mathbb{N}$, $end(C)$ (resp. $start(C)$) denotes the last (resp. first) time stamp where $cycleid == C$ holds. With respect to those notations, the variation of $s$ over the cycle $C$ is:

$$\Delta_C(s) = s(end(C)) - s(start(C))$$

We aim the monitoring of three properties which should be evaluated by cycle (this is a difficulty inherent to this benchmark).

### 1.1 Property a: max speed

For any cycle $C$, $max\ \{speed(t)\ /\ t \in [start(C), end(C)]\ \}\ <=\ ms$

### 1.2 Property b: average fuel consumption

For any cycle $C$, $\frac{\Delta_C(socfuel)*vFuel*100}{\Delta_C(distance)} < aFC$

### 1.3 Property c: battery recharge rate

For any cycle $C$:

$$\frac{1}{cBat}\int_{start(C)}^{end(C)} neg(elecpower).dt \leq br$$

where $neg$ is the function satisfying: $neg(x) = x$ if $x < 0$ else $neg(x) = 0$.

## 2 Monitoring of the Properties with the ARTiMon tool

We now present a formalization of the properties with the ARTiMon language. The properties are not presented in order because comments for property $c$ are useful for the other properties. Most of operators used have been presented in [1,2].

### 2.1 Property c: battery recharge rate

| Terms | Type |
|---|---|
| $br = ('double\_cst'\ -1.7)$ | double |
| $zero = ('double\_cst'\ 0)$ | double |
| $cycle\_end = Bot\ cycleid$ | contextual |
| $cycle\_stable = !\ cycle\_end$ | boolean |
| $ep\_zero\_vec[2]\ elecpower\ zero$ | (double, double) |
| $ep\_neg = \#\ (('<'\ ep\_zero\_vec)\ \&\ elecpower)$ | double |
| $ep\_neg\_integ = \#\ ('*'(LIc\ ep\_neg))$ | double |
| $ep\_neg\_integ\_reset = \#r\ (cycle\_stable\ \&\ ep\_neg\_integ)$ | double |
| $integral = (A\{'sum'\}_{[<,0]}\ ep\_neg\_integ\_reset)$ | double |
| $integral\_trans = (cycle\_end\ \&\ integral)$ | double |
| $div\_cbat\_vec[2]\ integral\_trans\ ('double\_cst'1602180)$ | (double, double) |
| $itgl\_div\_cbat = 'div'\ div\_cbat\_vec$ | double |
| $prop\_c\_vec[2]\ itgl\_div\_cbat\ br$ | boolean |
| $prop\_c = cycle\_end \rightarrow ('<'\ prop\_c\_vec)$ | boolean |

This property requires the creation of a signal which keep only negative values of *elecpower*. For this we cut the atomic term *elecpower* with the boolean property $elecpower < 0$. This is done with operator & (overloading of the conjunction) which is interpreted as a cut when its right term is not boolean. Let us recall that a cut term is equal to the right term when the left one is true, else has no defined value. Converting this undefined value to constant 0 with the casting operator # we obtain a term which is equal to *elecpower* if negative else equal to 0. We have to integrate this term by cycle. Concerning the integration, it can be done using an aggregation term of the form $A\{sum\}_{[<,0]}$ (formally $A\{sum\}_{]-\infty,0[}$) which computes the sum of the the strictly past values of its sub-term, combined with one operator of the $LI$ family ($LI$ for Incremental Length) presented in [2]. Here we use $LIc$ (for centered Incremental Length, not published before) which is a variant of $LI$. Suppose $c^i$ (constant function $c$ over interval $i$) is an extension of $(\phi)$ (semantic time function associated to term $\phi$) then $((i.ub - i.lb), c)^{[m,m]}$ with $m = (i.lb + i.ub)/2$ is in $(LIc\ \phi)$. Thus $((i.ub - i.lb) * c)^{[m,m]}$ is in $('*'(LIc\ \phi))$ and represents the integral of this extension. It follows that except at the middle time point of all extensions such a term has no defined value. The casting operator # is used again to convert the undefined value to 0. We obtain a term whose value is 0 everywhere except at some punctual times where it is the integral of a negative extension of *elecpower*. Forming the *sum* aggregation of this term would give us the integral from the origin. But not by cycle as we want to. To obtain the sum by cycle we insert a reset value at each cycle transition time in the latter term. To do this we cut it with *cycle_stable*. This amounts to overwrite at each cycle transition the current value with the undefined value. Then by applying operator $\#r$ (variant of #) undefined punctual values are converted to $RESET\_VAL$ (a value satisfying $sum(RESET\_VAL, a) = 0$ whatever is the aggregate $a$). Now one can form an aggregation term with *sum* as its value is set to 0 at each cycle transition time. Finally we check, at each cycle transition, that the sum is lower that $-1.7$ (i.e. that the battery recharge rate is greater that 1.7).

### 2.2 Property a: max speed

We only give few comments for this property as comments for the property $c$ provide the main ideas to understand the terms below. The aggregation function *greatest* is reset by $RESET\_VAL$ as follows: $greatest(RESET\_VAL, a) = DBL\_MIN$ (the minimum representable double) whatever is the aggregate $a$.

| Terms | Type |
|---|---|
| $speed\_by\_cycle = \#r\ (cycle\_stable\ \&\ speed)$ | double |
| $greatest\_speed = A\{'greatest'\}_{[<,0]} speed\_by\_cycle$ | double |
| $greatest\_end\_cycle = cycle\_end\ \&\ greatest\_speed$ | double |
| $prop\_a\_vec[2]\ greatest\_end\_cycle\ ms$ | (double, double) |
| $greatest\_exceed\_ms = '>'\ prop\_a\_vec$ | boolean |
| $prop\_a = cycle\_end \rightarrow greatest\_exceed\_ms$ | boolean |

### 2.3 Property b: average fuel consumption

For this property we need to compute $\Delta_C(socfuel)$ and $\Delta_C(distance)$. This is the hardest aspect of this property. Due to lack of space the fraction and the comparison with the threshold, already presented for the other properties are not detailed (one creates the suited vectors and applies function $'div'$ and the predicate $' >'$ ). We give below the pattern for computing $\Delta_C(s)$ given a variable $s$ (whose type is double). *Bot* is the falling edge operator. $Top0$ is an operator of the rising edge family. This one consider that its sub-term has an initial defined value (false). Thus we have a rising edge at the first time stamp. We use the overwriting aggregation function := which satisfies $:= (x, a) = x$ whatever is the aggregate $a$. Intuitively a term of the form $A\{':='\}_{[<,0]}\psi$ denotes the *last of the strict past values* of $\psi$. Here & is the cut operator (see 2.1).

| Terms | Type |
|---|---|
| $cycle\_end = Bot\ cycleid$ | contextual |
| $s\_start = (Top0\ cycleid)\ \&\ s$ | double |
| $s\_start\_at\_end = A\{':='\}_{[<,0]}\ s\_start$ | double |
| $s\_at\_end = cycle\_end\ \&\ (A\{':='\}_{[<,0]} s)$ | double |
| $s\_delta\_vec[2]\ s\_at\_end\ s\_start\_at\_end$ | (double,double) |
| $delta\_s = '-'\ s\_delta\_vec$ | double |

## 3 Experimental Results

ARTiMon checks the trace in one seconds (core I5-3230M @ 2.6Ghz ) and allocates 85612 bytes (around 84 kB). Verdicts are provided in an html file that can be open with any standard web browser.

## References

1. Nicolas Rapin. Reactive property monitoring of hybrid systems with aggregation. In Yliès Falcone and César Sánchez, editors, *Runtime Verification: 16th International Conference, RV 2016, Madrid, Spain, September 23–30, 2016, Proceedings*, pages 447–453. Springer International Publishing, Cham, 2016.
2. Nicolas Rapin. Artimon monitoring tool, the time domains. In *RVCubes 2017*. 2017.