A Compositional Transformation to Bridge the Gap between the Technical System and the Computational System

Dieter Zöbel
Institut für Softwaretechnik
Fachbereich Informatik
Universität Koblenz-Landau
Email: zoebel@uni-koblenz.de

Abstract

The majority of embedded applications with real-time constraints monitor and control a technical system. The correct behavior of such a system typically is described in terms of the technical system. In contrast the embedded hard- and software operates on an image of the technical system which is prone to deviations and delays. Therefore a compositional transformation is proposed which maps assertions specifying the behavior of the technical system to the program level conditions which guarantee for those assertions.

1 Introduction

Like any other human being a scientist is shaped by the community which he or she belongs to. The view on a problem scope and the way to find a solution is deeply inspired by the paradigms which are the common tenet to the respective communities. This phenomenon can be observed when scientists cooperate on a common subject area, e.g. electrical engineers and computer scientists. But even within scientific communities there are disparities of views, e.g. the time-triggered community and the event-triggered community inside the real-time community.

The subject area of embedded applications with real-time constraints is prone to this phenomenon. On one hand there is the hard- and software which builds up the computational system. On the other hand there is the physicality of the technical system which has to be monitored or controlled. The former is more in the focus of computer scientists, the latter more in the focus of engineers. They all have the common motivation to design and implement safe embedded applications by using mature engineering techniques.

Reflecting the state of the art in developing time-critical embedded applications there exists a myriad of mature but isolated techniques for certain questions which are relevant in the design process. One technique to be cited in this context is the worst case execution time analysis (WCET) which supplies indispensable parameters to enforce real-time properties. The execution times of processes are input to any real-time scheduling algorithm which itself builds upon a process oriented paradigm of programming. As a consequence of the variety of isolated techniques, various authors state that there is a strong need for holistic approaches, integrating the diversity by the establishment of a few essential paradigms. Such an approach cannot be a new level of abstraction on top of the existing techniques [3]. Instead it requires more or less a start from scratch.

As desirable as such a holistic approach may be, it is a long term option at the moment. In contrast, short term options have to be far more modest in that they should bridge the gaps which still exist between mature but isolated techniques. Furthermore, they should identify chains of techniques and tools which are able to support certain development processes for embedded applications. The structuring elements of this approach are the interface definitions between the joints of the chain. In its modesty this approach reveals where there are versatile techniques and tools, where they are weak, or where they are missing at all. Additionally it has to be noticed that high level techniques and tools which use certain assertions pretend that in turn these assertions can be easily propagated to lower levels of abstraction. Shortcomings of this kind can be observed for modelling languages and adjacent verification tools which apply to the technical system. So, it may be that the correctness of a system is proved using basic assertions about the technical system. However, the common verification techniques neglect that for completeness a profound subchain of techniques and tools which are needed to derive these basic assertions from lower level abstractions. Often those assertions have to be derived arduously from the level program code [7].
The following two sections show the basic ideas of a transformation technique which is able to bridge the gap between verification techniques applied to the technical system and the techniques of real-time programming applied to the computational system. The next sections presents a case study applying this bridging technique to a standard real-time application. Finally, there is a conclusion assessing the technique introduced and an outlook to further research efforts on this topic.

2 Bridging the gap

In the scope of real-time scheduling basic techniques towards the formulation of real-time conditions have been adopted from modelling techniques originally applied to database systems. This centers around the term consistency which in addition to a value based definition in the scope of database systems requires certain extensions referring to the time this data was created and the aging of this data when being used by real-time processes (see [1] and [5]). The two decisive definitions – absolute and relative temporal consistency – bound the absolute and relative time since the data has been taken from the technical system.

A generalization of this approach to determine real-time conditions distinguishes between the technical system, represented in terms of real-time entities, and its observation, namely real-time images [4]. A relation, called temporal accuracy, is defined for assigning the real-time image to some real-time entity within a bound history. Based on this knowledge the worst case error when utilizing this real-time image is estimated and can be taken into account for decisions which have to be made by the real-time process.

This paper wants to give a brief sketch, how the consequent extension of these approaches cited above results in a surplus value which consists in bridging the gap between a certain assertion $I$ necessary for the correct operation of the technical system and the coded control action $CA$ corresponding to the following program fragment:

```c
if (Condition) Action;
```

To explain the approach in more detail the viewpoint of a programmer developing a time-critical embedded application is adopted here. This viewpoint is program-centric in that the values of variables are processed and evaluated for decision making. Particulary in the scope of embedded systems several questions emerge from this view and unsettle the programmer:

- How precise is the value of a variable in correlation to the technical system?
- From which instant of time with respect to the technical system does the value of a variable stem?
- At which instant of time a decision will be made by the program in execution?
- At which instant of time will the decision made by the program take effect in the technical system.

Program code written under these circumstances is aggregated to processes which build up the computational part of the embedded system. These processes are executed concurrently following some real-time scheduling policy. Even though there is a profound theory of scheduling behind, the question remains what is the right control scheduling policy. Even though there is a profound theory of scheduling behind, the question remains what is the right control scheduling policy.

Figure 1. The reference architecture of an embedded system consisting of computational system which monitors and controls the technical system.

3 Transformation of value domains

The computational system monitors and controls the technical system. Let $x$ and $y$ be physical entities of the technical system. Later in the case study $x$ will be the fuel level of a tank and $y$ a pump which can be switched to refuel the tank. The entities $x$ and $y$ have corresponding value domains $V_x$ and $V_y$. Typically sensors and actuators as in figure 1 introduce deviations in value and cause time delays. Additionally the infrastructure and the application processes of the computational system are responsible for further delays. Therefore an invariant property $I$ cannot be directly used as Condition in the respective program fragment. Instead a transformation has to be applied which takes into account all deviations and delays:

$$I \bowtie \triangleright V_C$$

The operator $\bowtie$ correlates value domains on one hand belonging to the technical system on the other hand to the computational system. E.g. in the following case study the
values $V'_x$ satisfying invariant $I$ are correlated to these observed values $OV'_x$ of the computational system which guarantee for the validity of the invariant.

This correlation can be computed in a compositional way, which step by step takes into account all deviations and delays. E.g. the first step is build up by the sensor relation which models the falsifying behavior of the sensor: $$SR_x \subseteq V_x \times OV_x$$

For some set of observed values $OV'_x \subseteq OV_x$ it should be known which physical values may have caused them via the sensor: $$DOM(SR_x, OV'_x) = \{ v_x \in \{ ov_x \in SR_x \wedge ov_x \in OV'_x \} \}$$

Analogously there is a respective relation $AR_y$ on the actuator side.

A further operator, which is needed, has to predict what may happen in the technical system. This is captured by mapping $TS$: $$TS_x : 2^{V_x} \times \Delta T \rightarrow 2^{V_x}$$

Let $ov_x$ be some image value of the fuel level processed in the computational system. Applying the following mapping it is possible to derive all values $V'_x \subseteq V_x$ which may have been read once before by the sensor system and after some delay are processed by some process as image value $ov_x$. To compute what is put in the sentence above as after some delay includes the possible time interval starting from the earliest to the latest time this value may stem from. This time interval $[t_{early}, t_{late}]$ must include the processing time and henceforth depends on the policy of real-time scheduling.

$$\bigcup_{\Delta \tau \in [t_{early}, t_{late}]} TS_x(DOM(SR_x, \{ ov_x \}), \Delta \tau)$$

Unfortunately this is not the operational structure which is needed from the viewpoint of program development. In typical applications the requirements in terms of $I$ are given and the control action $CA$, particularly the Condition has to be coded. So, the inversion of the formula above is needed which is explained in detail in [7].

4 A case study:
Controlling the fill-level of a tank

To illustrate the transformation to find the correct Condition we refer an example of the fuel tank mounted near the jet engine in an airplane [2]. Let us assume that the fill-level of this tank should by guarantee never be less than some value: $$I \equiv v_x \geq 50 l$$

Because the fuel of this tank is steadily consumed by engines there is a pump to refill this tank from other tanks. The status of the pump is determined by: $$pump_{on} \equiv v_y = 1$$

As any other technical system our fill-level control system suffers from a lot of time- and value-dependant imprecisions. Control is possible only if some knowledge is available about the lower and upper bounds of these imprecisions. Let us assume to have the following knowledge:

- The vendor of the fuel-level measurement system guarantees that the value $ov_x$ never deviates more than $\pm 10\%$ from the value $v_x$.
- The fill-level sensor is an independent device. When read by the process which executes $CA$ the age of the fill-level value is somewhere between $10 ms$ and $50 ms$.
- Fuel is steadily consumed from the tank, with minimum consumption of $0.1 l/s$ and in peak situations up to $20 l/s$. So $v_x$ is perishable between these largely differing rates.
- Process $i$ responsible for the fulfillment of $I$ is preemptive and periodic within the interval $\Delta p_i = 150 ms$.
- Finally the reaction by the actuation system has to be modelled. Here the assumption is that from setting $OV_y$ until the instant of time that the pump is running lasts up to $350 ms$. Conversely, there is no reaction at all of the pump before $70 ms$.

This allows to calculate the lower and upper bounds for:

$$t_{early} = 10 ms + 70 ms = 100 ms$$
$$t_{late} = 50 ms + 2 \times 150 ms + 350 ms = 700 ms$$

Now we can derive Condition reversing the formula mentioned in the end of the last section.

1. We determine the set $V'_x \subseteq V_x$ for which $I$ holds: $$V'_x = \{ v'_x \in V_x | v'_x \geq 50 l \}$$

2. Next we determine those values $V''_x$ which have been sensed in some past $t - \tau$, $\tau \in [t_{early}, t_{late}]$ and still satisfy $I$ at time $t$. From the deliberations above we know that any decision effecting the pump is based on sensed fill-levels $v_x$ in the interval: $$100 ms \leq \tau \leq 700 ms$$

So, the fill level minimally shrinks by

$$100 ms \times 0.1 l/s = 0.1 l$$
and maximally increases by

\[ 700 \text{ms} \times 20l/s = 14l \]

Taking into account the highest decrease we find \( V''_x \) that contains scalar values and the imprecision is proportional to \( v''_x \) it suffices to concentrate on border values. So, we look for the smallest \( ov''_x \), such that the corresponding values \( v''_x \) are in \( V''_x \) and find this border value by multiplying the border value from above with the highest deviation:

\[ 64l \times 1.1 = 70.4l \]

In terms of relation \( SR_x \) we can assert that whenever \( ov''_x \geq 70.4l \) then all \( v''_x \) for which \( (v''_x, ov''_x) \in SR_x \) are elements of \( V''_x \).

4. We code the condition \((OVx >= 70.4)\) which finally guarantees that \( I \) holds under all value- and time-dependent imprecisions. Hence, the resulting transformation reads:

\[ (v_x \leq 50l) \supseteq (OVx >= 70.4) \]

The control application presented in this case study, though it is still rather simple, demonstrates the essential steps to gain the right Condition to fulfill the specification \( I \). Different from our demonstration we have often the case that the periods are not known. Instead, there we may be interested in those functional dependencies which determine the period (which is also a deadline here). So it may be that we code \((OVx >= 80)\) and ask for the consequences with respect to the periods. This situation can in turn be solved by using the period as parameter in the equations above, e.g. instead of \( 14l \) we compute the maximum decrease as a function of \( \Delta_p \):

\[ (400 \text{ms} + 2 \times \Delta_p) \times 20l/s \]

Continuing this calculation we obtain that the period should be less than \( 368\text{ms} \). This demonstrates the degree of freedom available with this canonical approach.

5 Conclusion and outlook

First of all this approach wants to be understood as a consequent and sophisticated enhancement of those papers which already have modelled value- and time-dependent imprecisions of real-time systems (e.g. [1], [4] or [6]). Even at the lowest level of abstraction – the coding of statements which interfere with the technical system to be controlled – the topics of scheduling and verification can be combined.

So, on one hand there is the verification at the level of programs. Here a property e.g. that Condition is evaluated in any period can be proved. This asserts that the correct Action is executed if necessary. On the other hand there is the verification at the level of the technical system. Here the basic assertion \( I \) regarding the fuel level is input for the deduction of higher level properties like the aeronautical stability of the plane. In this context the transformation \( I \supseteq C \) bridges a gap between two important sub-chains of techniques and tools. At the same time the transformation has a upper interface in a value domain \( V \) and a lower level interface in the value domain \( OV \) which makes it independent of the lower and higher level verification tools.

The essential disadvantage so far is that the transformation has to be performed manually. Consequently, those relations and mapping that compose the transformation have to be identified and elaborated to generic building blocks which permit the automated derivation of correlations of the value domains. This would enhance both the top down design of embedded applications and the bottom up adaption and tuning of system parameters as it is needed in the scope of sensitivity analysis.

References


