Clock Discipline Algorithm in Coq

Benjamin Lion[*], Inria Rennes

September 25, 2024

Context

"Real-time model checking is really simple." As advertised by Leslie Lamport in [3], the real passage of time can be encapsulated explicitely in TLA+ using a *now* variable that increments with a *Tick* action. As a consequence, the model checking of real-time properties is easier, without necessarily requiring all the complex background details.

Yet, a model differs from the code that executes on a machine, unless proven otherwise. This is the main result of CompCert [2], a verified compiler for the C language, that certifies that the semantics of an executable coincides with the semantics of the source code. However, the semantic preservation theorem does not yet include real time guarrantees.

As CompCert explicitly says in its documentation, the semantic preservation theorem ensures that observable behaviours of the source and target programs are the same, and define observable behaviour as "everything the user of the program, or the physical world in which it executes, can *see* about the actions of the program, with the notable exception of execution time and memory consumption.".

Challenge

In this work, we will focus on a specific time sensitive algorithm: the Clock Discipline Algorithm [7]. The algorithm synchronizes a high frequency clock that may accumulate drifting, with a low frequency clock that is reliable. Typically, this algorithm is employed to secure local time on an operating system from the source time given by an external reliable source (i.e., via NTP).

The Clock Discipline algorithm estimates the difference between the two clocks, so that a reading of the local clock can be reliably converted as a value on the source reliable clock.

We will implement this algorithm in software (in a fragment of C), and use a certified compiler to generate code. Moreover, we will prove that the bound

[*]benjamin.lion@inria.fr, cert-t website

estimations given by the algorithm, under assumption verified by the hardware, are realistic and preserved through compilation. The challenge will therefore be to port some of the existing proving technics for memory-safe programs [1], to deal with time-dependent registers (such as [6]).

Goals

The verification of the clock discipline algorithm decomposes into three main goals:

1. Identification of a suitable formalism to formalize the Clock discipline algorithm in Coq.
   The clock discipline algorithm has time-dependent instruction. We will use a simple time model, similar to the time-stamp counter register in modern processor, to reason about correctness of the algorithm. For instance, in [7], it says that "It is possible to convert the discretized time reports of the clocks to continuous time readings by assuming that the associated counter increments once in one period, and by ignoring residual time error within one period."

2. Implementation of the model in a low level language, such as Clight.
   This step requires to get familiar with the Clight syntax and its semantics. Several extensions will need to be considered, to express time syntactically and semantically, such as in [4].

3. Practical generation of certified code on a specific plateform.
   The third objective is to setup a toolchain to generate executable and testable code on some specific architecture. Then, the performance of the generated code could be evaluated with respect to the model prediction.

The set of goals is ambitious, which implies that some of the goals might not be fully completed given the duration of the internship.

Practical details

- supervised by Benjamin Lion;
- funded by the cert-t project (https://blion.gitlabpages.inria.fr/cert-t/);
- gratification;
- integration in the Epicure team at Inria Rennes.

Future work

Time plays a critical role on embedded and systems with energy constraints. One project to benefit from the work in the RIOT [5] operating system for IoT, where its timer library makes use of similar clock discipline algorithm.

References

[1] Sandrine Blazy and Xavier Leroy. "Mechanized semantics for the Clight subset of the C language". In: *CoRR* abs/0901.3619 (2009). arXiv: 0901.3619. URL: http://arxiv.org/abs/0901.3619.

[2] CompCert. *CompCert website*. URL: https://compcert.org/.

[3] Leslie Lamport. "Real-Time Model Checking is Really Simple". In: *13th Advanced Research Working Conference on Correct Hardware Design and Verification Methods*. June 2005. URL: https://www.microsoft.com/en-us/research/publication/real-time-model-checking-is-really-simple/.

[4] Saranya Natarajan and David Broman. "Timed C: An Extension to the C Programming Language for Real-Time Systems". In: *IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2018, 11-13 April 2018, Porto, Portugal*. Ed. by Rodolfo Pellizzoni. IEEE Computer Society, 2018, pp. 227–239. DOI: 10.1109/RTAS.2018.00031. URL: https://doi.org/10.1109/RTAS.2018.00031.

[5] RIOT-OS. *Ztimer in RIOT-OS*. URL: https://doc.riot-os.org/group__sys__ztimer.html.

[6] Wikipedia. *Time Stamp Counter*. URL: https://en.wikipedia.org/wiki/Time_Stamp_Counter.

[7] Hüseyin Yigitler, Behnam Badihi, and Riku Jäntti. "Overview of Time Synchronization for IoT Deployments: Clock Discipline Algorithms and Protocols". In: *Sensors* 20.20 (2020), p. 5928. DOI: 10.3390/S20205928. URL: https://doi.org/10.3390/s20205928.