Modular compilation of real-time programs

Ph.D. thesis proposal, October 2025, Lille

Supervision: Julien Forget, Associate Professor (HDR), University of Lille

1 Context

An Embedded Real-Time System (ERTS) controls a physical device in its environment, at a rate adapted to the device and its environment. This requires not only to compute correct values, but also to compute values at the right time. For instance, in the control-command system of an aircraft, computations must be performed quickly enough to react to external perturbations, such as a gust of wind. Real-time systems can be found in several industrial domains, including automotive, avionics, nuclear plants or automated production lines.

Many ERTS are of critical nature; a malfunction may damage properties or even put human lives in danger. So as to ensure the safety of ERTS, a wide range of formal techniques have been proposed to support their design, implementation and validation. Unfortunately, one of the main drawbacks of formal methods is their poor scalability, thus limiting their applicability to complex modern ERTS. A classical approach to reduce complexity is the component-based approach. First, the system is decomposed into smaller components, each developed and analyzed in isolation. Then, the components are integrated together, and a simpler analysis is performed at the integration level.

In the current industrial practice for the development of ERTS, timing requirements are specified and analyzed only at the global level, when assembling all components. Thus, timing analysis suffers from the scalability issues mentioned previously.

2 Thesis objective

The objective of the thesis is to contribute to a framework for the component-based development of real-time systems, focusing on *modular compilation*.

The system will be programmed with the synchronous data-flow language Prelude [2], which was specifically designed for the development of critical ERTS. The language already contains constructs to structure a program as a hierarchical set of components. However, the Prelude compiler flattens the program before translating it into a non-hierarchical set of concurrent threads implemented in C.

In this thesis, we will study the compilation of a Prelude program into C modules structured with a hierarchy similar to that of the source program. The candidate will define a notion of real-time interface for modules, which will expose the temporal characteristics necessary to compose modules together. The compiler will target execution with a Real-Time Operating System equipped with a hierarchical scheduler [1]. In this scheduling model, each module is assigned a *partition* of the system resources and its sub-modules are executed by a *local scheduler*. Top-level modules are executed by a *global scheduler*. Particular care will go into ensuring that the semantics of the Prelude program is preserved by the compilation, especially regarding the temporal characteristics of the module interfaces.

3 Applicant skills

Ideally, the applicant will have experience in some of the following areas:

- Compilation;
- Synchronous languages;
- Real-time scheduling;
- Operating Systems;
- Programming: C, OCaml.

4 Research team: SyCoMoRES

The Ph.D. student will work within the SyCoMoRES team, a joint CRIStAL-Inria research team, located in Lille, France. One of the objectives of the SyCoMoRES project-team is to develop frameworks for the design and the analysis of embedded real-time systems. SyCoMoRES aims at reducing the complexity of large scale embedded real-time software projects by using modularity: component-based design and implementation, parametric model specification, and symbolic analysis. We use formal models and tools at all levels of abstraction.

References

[1] Giuseppe Lipari, Paolo Gai, Michael Trimarchi, Giacomo Guidi, and Paolo Ancilotti. A hierarchical framework for component-based real-time systems. *Electronic Notes in Theoretical Computer Science*, 116:253–266, 2005.

[2] Claire Pagetti, Julien Forget, Frédéric Boniol, Mikel Cordovilla, and David Lesens. Multi-task implementation of multi-periodic synchronous programs. *Discrete event dynamic systems*, 21:307–338, 2011.