`libFAUDES`
A Software Library for Supervisory Control

Abstract. `libFAUDES` implements data structures and algorithms for finite automata and regular languages. The library takes a control theoretic perspective as originally introduced by P.J. Ramadge and W.M. Wonham in the 1980's and, since then, actively discussed within the scientific community. `libFAUDES` is an open project, contributions to the further development are highly appreciated.

Supervisory Control

For a concise introduction, consider the so called *plant* to be modelled by a prefix-closed language L over Σ. Here, L is interpreted as the set of all *possible strings* the plant can exhibit. Furthermore, assume another prefix-closed language $E \subseteq L$ to represent a formal *specification*. Here, E is interpreted as the set of all *acceptable strings*. In the situation where E is a strict subset of L, the plant may exhibit unacceptable strings, and one asks for a *supervisor* that interactively restricts the plant behaviour accordingly. For this purpose, the alphabet Σ is decomposed as a disjoint union of *controllable events* and *uncontrollable events*, $\Sigma = \Sigma_{\mathrm{c}} \dot{\cup} \Sigma_{\mathrm{uc}}$, respectively. The supervisor may at any instance of time disable any controllable event. Technically, this is expressed by the *controllability condition* $K\Sigma_{\mathrm{uc}} \cap L \subseteq K$ imposed on the closed-loop behaviour $K \subseteq L$. A crucial observation is that, given a plant and a specification, there uniquely exists a supremal controllable sublanguage $K^{\uparrow} \subseteq E \subseteq L$. Moreover, if L and E are regular, then so is $K^{\uparrow}$, and a realisation S can be computed from automata representations of L and E. Here, S is interpreted as an implementation of a supervisor that, when operating the plant, enforces the specification by strategically disabling controllable events.

Since the original work of P.J. Ramadge and W.M. Wonham in the 1980's, many researchers have contributed to supervisory control. This includes further development of the theoretical framework, e.g, to cover hierarchical, modular and decentralized control architectures, as well as discussions on implementation issues, e.g. automatic code generation for Programmable Logic Controllers (PLCs) in the context of industrial applications. However, there are still various open questions and, hence, supervisory control remains an active field of research.

Scope of `libFAUDES`

The main purpose of `libFAUDES` is to promote the framework of supervisory control theory in two regards. First, `libFAUDES` implements a wide range of algorithms on finite automata, including general functions (e.g. boolean operations on the associated languages), solutions to common controller synthesis problems (e.g. supremal controllable and normal sublanguages), simulations to inspect and debug closed-loop dynamics and the execution of controllers on the actual physical plant (so called *hardware-in-the-loop simulation*). Thus, `libFAUDES` can serve as an environment to evaluate a design method in a real-world application context. The second aspect in the design of libFAUDES is to provide a software infrastructure to reduce the coding effort for newly developed methods in supervisory control.

Implementation details:

C++ Library. libFAUDES is written in C++ with use of the Standard Template Library (STL). The generator classes are set-based models that resemble the definition of a quintuple automaton $G = (Q, \Sigma, \delta, Q_o, Q_m)$. Using libFAUDES to implement algorithms originally stated in terms of automata is straightforward and benefits from the general infrastructure, like file I/O and graph visualisation via Graphviz/dot[1].

Non-restrictive Open-Source License. libFAUDES sources are distributed for free under conditions of the GNU Lesser General Public License (LGPL). There are no relevant restrictions to use the library in non-open source and/or commercial projects. libFAUDES comes with a (compile-time-) plug-in mechanism to extend the library in an organised manner. In particular, plug-in developers are invited but not forced to contribute their code to libFAUDES.

Lua Scripting Support. The interpreter luafaudes makes libFAUDES data types and functions available within the scripting language Lua[2]. The wrapper code that integrates libFAUDES with Lua is automatically generated by the build-system. The additional coding effort required to provide access to library extensions from within Lua is minimal.

User Reference. The libFAUDES build system can organize an optional user reference, to complement the C++ API documentation. Thus, developers who wish to illustrate, motivate or explain their algorithms by examples, informative text or formal definitions, can directly address users of libFAUDES based applications, e.g. luafaudes or the graphical front-end DESTool.

Further Information

libFAUDES is currently maintained by the Lehrstuhl für Regelungstechnik, University Erlangen-Nürnberg, Germany. A detailed user reference, examples, literature references, a list of contributing authors, and downloadable packages are available at the project web page www.rt.eei.uni-erlangen.de/FGdes/faudes.

This summary for the presentation of libFAUDES *at* CIAA12 *has been composed by*
Thomas Moor, Lehrstuhl für Regelungstechnik, University Erlangen-Nürnberg, Germany,
Klaus Schmidt, Department of Mechatronics Engineering, Çankaya University, Turkey, and
Tomáš Masopust, Institute of Mathematics, Academy of Sciences, Czech Republic.

[1] See www.graphviz.org
[2] See www.lua.org