12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Bridging the Gap between Supervisory Control and Reactive Synthesis: Case of Full Observation and Centralized Control ⋆

Rüdiger Ehlers * **Stéphane Lafortune** ** **Stavros Tripakis** ***
Moshe Vardi ****

* *University of Bremen and DFKI GmbH*
** *University of Michigan*
*** *University of California, Berkeley and Aalto University*
**** *Rice University*

Abstract: We present a formal connection between supervisory control theory in the field of control engineering and reactive synthesis in the field of formal methods. We focus on the case of fully-observed discrete-event systems that are controlled by a single controller/supervisor in order to achieve a safety specification and a non-blocking specification. The connection is shown by a reduction of the corresponding supervisory control problem to a problem of reactive synthesis with plants and maximal permissiveness, subject to a CTL temporal logic specification. In order to establish the desired reduction, we prove two new results regarding (i) a simplified version of the standard supervisory control problem and (ii) a class of reactive synthesis problems that admit unique maximally permissive solutions. The reduction complements prior work at the boundary of supervisory control and reactive synthesis.

1. INTRODUCTION

We present a formal connection between synthesis problems that have been considered, largely separately, in the two communities of control science in engineering and formal methods in computer science. By making this connection mathematically precise, we hope to "bridge the gap" between two research areas that aim at tackling similar synthesis problems for discrete event systems, but from different angles, and by emphasizing different, and often complementary, aspects. Such a formal bridge should be a source of inspiration for new lines of investigation that will leverage the power of the synthesis techniques that have been developed in these two areas.

Supervisory Control: The control science and engineering community has been investigating feedback control of Discrete Event Systems (DES) using models from computer science, such as automata and Petri nets. The body of control theory developed in DES has been for specifications that are expressible as regular languages, in the case of DES modeled by automata, or in terms of constraints on the state (marking vector), in the case of DES modeled by Petri nets. Control-theoretic frameworks have been developed for both of these modeling formalisms; cf. Seatzu et al. [2013]. In this paper, we focus on the supervisory control theory for DES modeled by finite-state automata and subject to regular language specifications. Both the "plant" (i.e., uncontrolled system) and the specification are represented as finite-state automata over a common event set. The foundations for this framework were developed in the seminal work of Ramadge and Wonham [1987]. Since then, a whole body of theory has been developed that covers a wide variety of control architectures and information structures, with vertical and horizontal modularity. The reader is referred to Cassandras and Lafortune [2008] and Wonham [2013] for textbook expositions of this theory. The focus of this theory is on the synthesis of provably safe and non-blocking controllers for a given plant, despite limited actuation and limited sensing capabilities.

Reactive Synthesis: The design of *reactive systems*, i.e., systems that engage in an ongoing interaction with their environment, is one of the most challenging problems in computer science. In reactive systems, a correct system should satisfy the specification with respect to *all* environment behaviors. The specification is usually expressed in a temporal logic. Pnueli and Rosner [1989b], Abadi et al. [1989], and Dill [1989] argued that the right way to approach synthesis of reactive systems is to use the model of a, possibly infinite, game between the environment and the system. A correct system can be then viewed as a winning strategy in this game. It turns out that satisfiability of the specification is not sufficient to guarantee the existence of such a strategy. Abadi et al. [1989] called specifications for which a winning strategy exists *realizable*. Since then, the subject of *reactive synthesis* has been an active area of research, attracting considerable attention; see, e.g., Pnueli and Rosner [1989a], Vardi [1995] and Kupferman and Vardi [2000].

⋆ This work was supported by the University of California at Berkeley, the University of Michigan, Rice University, Aalto University, the Academy of Finland, and the US National Science Foundation, via projects *ExCAPE: Expeditions in Computer Augmented Program Engineering* and *COSMOI: Compositional System Modeling with Interfaces*. This work was also supported in part by the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies), and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley (supported by NSF, NRL, and the following companies: Bosch, National Instruments, and Toyota).

Related Works: This paper is not the first to explore connections between supervisory control and reactive synthesis. On the supervisory control side, several authors have considered control of DES subject to temporal logic specifications; see, e.g., Thistle and Wonham [1986], Lin [1993], and Jiang and Kumar [2006]. Supervisory control of DES with *infinite* behavior has also been considered by many researchers; see, e.g., Ramadge [1989], Kumar et al. [1992], Thistle and Wonham [1994a], and Thistle and Wonham [1994b]. On the other hand, several researchers in the formal methods community have investigated supervisory control of fully- and partially-observed DES; see, e.g., Hoffmann and Wong Toi [1992], Asarin et al. [1995], Madhusudan [2001], Kupferman et al. [2000], Arnold et al. [2003], and Riedweg and Pinchinat [2003].

Contribution: In the present paper, we restrict attention to the classical version of centralized supervisory control for fully-observed systems modeled by languages of finite strings. Our goal is to establish a precise connection of supervisory control problems with problems of reactive synthesis, by showing how specific problem instances reduce to each other. To our knowledge, such reductions have not been published elsewhere. Our results therefore complement the existing work. We start in Section 2 by briefly presenting necessary background material from supervisory control and reactive synthesis. The main results of this paper are contained in Section 3. First, we present in Section 3.1 a simplification of the basic supervisory control problem, non-blocking version, to one where the safety specification has been absorbed into the plant model. We then show that the resulting Simple Supervisory Control Problem (SSCP) has a state-based solution. Second, for bridging reactive synthesis with supervisory control, we need two technical steps: the first step is to consider reactive synthesis with plants; the second step is to bring in the issue of maximal permissiveness into this reactive synthesis setting. These two steps are covered in Section 3.2. We then establish the formal reduction from SSCP to a reactive synthesis problem with plants and maximal permissiveness in Section 3.3. A discussion and some concluding comments follow in Sections 3.4 and 4.

Due to space limitations, we do not review temporal logics such as LTL, CTL, and CTL*. Moreover, our presentation excludes proofs and has few examples. Our focus is on presenting the necessary concepts for our main results, Theorem 4 and Corollary 2. We refer the reader to Ehlers et al. [2013] for a detailed treatment of our results.

2. BACKGROUND

Supervisory Control: In supervisory control theory, plants are typically modeled as deterministic finite-state automata. A deterministic finite-state automaton (DFA) is a 5-tuple $G = (X, x_0, X_m, E, \delta)$ where

- X is a finite set of states, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is the set of *marked* states;
- E is a finite set of events. E is (implicitly) partitioned into two disjoint subsets: $E = E_c \cup E_{uc}$ where E_c models the set of *controllable* events and E_{uc} the set of *uncontrollable* events.
- $\delta : X \times E \to X$ is the transition function, which in general will be partial.

The transition function is partial because G models the physically possible behavior of the uncontrolled plant, as a generator of events. Selection of the states to "mark," i.e., to be included in X_m, is a modeling consideration to capture strings that represent that the system has completed some task. It is customary to extend δ to strings. The DES G defines the following languages: $\mathcal{L}(G) = \{\sigma \in E^* \mid \delta(x_0, \sigma) \text{ is defined}\}$ and $\mathcal{L}_m(G) = \{\sigma \in E^* \mid \delta(x_0, \sigma) \in X_m\}$.

A *supervisor* for G is a function $S : E^* \to 2^E$. To ensure that S never disables an uncontrollable event, we require that $E_{uc} \subseteq S(\sigma)$ for all $\sigma \in E^*$. Given $G = (X, x_0, X_m, E, \delta)$ and $S : E^* \to 2^E$ for G, the *closed-loop system* S/G is formally defined as follows: $S/G = (X', x_0', X_m', E, \delta')$ where

- $X' = X \times \mathcal{L}(G)$
- $x_0' = (x_0, \varepsilon)$
- $X_m' = X_m \times \mathcal{L}(G)$
- $\delta'\big((x,\sigma), e\big) = \begin{cases} (\delta(x,e), \sigma e) & \text{if } \delta(x,e) \text{ is defined} \\ & \text{and } e \in S(\sigma) \\ \text{undefined} & \text{otherwise.} \end{cases}$

S/G is an automaton, therefore, languages $\mathcal{L}(S/G)$ and $\mathcal{L}_m(S/G)$ are well defined. It is easy to verify that $\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$ since a marking in S/G is completely determined by a marking in G. S is said to be *non-blocking for* G iff $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$ where the overline notation denotes prefix-closure.

Consider a plant G and two supervisors S_1, S_2 for G. We say that S_1 *is no more permissive than* S_2 iff $S_1(\sigma) \subseteq S_2(\sigma)$ for any σ. We say that S_2 *is strictly more permissive than* S_1 iff S_1 is no more permissive than S_2 and $S_1 \neq S_2$.

S is needed in order to enforce the *safety specification* imposed on G. In supervisory control, the safety specification is modeled by a prefix-closed regular language, denoted by L_a, over the event set E of G. L_a is prefix-closed since for a string to be safe, all of its prefixes should also be safe. In this paper, we define the *admissible marked language* L_{am} for plant G as $L_{am} := L_a \cap \mathcal{L}_m(G)$. S is said to be *safe for* G *with respect to* L_{am} if $\mathcal{L}(S/G) \subseteq \overline{L_{am}}$. A supervisor S which is non-blocking for G and safe w.r.t. L_{am} is said to be *maximally-permissive with respect to* G *and* L_{am} if there is no supervisor S' which is non-blocking for G, safe w.r.t. L_{am}, and strictly more permissive than S. The theorem below shows that, for non-blockingness and safety, a *unique* maximally-permissive supervisor exists, provided that a supervisor exists at all. This well-known result from Ramadge and Wonham [1987] motivates the definition of BSCP-NB that follows.

Theorem 1. Consider G and L_{am} as defined above. If there exists a supervisor which is non-blocking for G and safe w.r.t. L_{am}, then there exists a unique maximally-permissive supervisor S_{mpnb} which is non-blocking for G and safe w.r.t. L_{am}.

Definition 1. (BSCP-NB). Given G and L_{am} as defined above, find if it exists, or state that there does not exist, a supervisor for G which is non-blocking for G, safe w.r.t. L_{am}, and maximally-permissive.

Reactive Synthesis: In reactive synthesis, we build correct-by-construction "controllers" (or system imple-