Rüdiger Ehlers (Berkeley, Cornell), Stéphane Lafortune (Michigan), Stavros Tripakis (Berkeley), and Moshe Vardi (Rice)
“Classic” Synthesis Frameworks

- **Reactive synthesis:**
  - From declarative specifications (e.g., LTL formulas) to implementations (e.g., Mealy or Moore state machines).
  - *On the Synthesis of a Reactive Module* [Pnueli-Rosner, POPL’89], but also earlier, e.g., [Church ’63].
  - See, e.g., Moshe Vardi’s summer school tutorial for details.

- **Supervisory control:**
  - Feedback control for discrete-event systems (DES).
  - *Supervisory control of a class of discrete event processes* and *On the supremal controllable sublanguage of a given language* [Ramadge-Wonham, SIAM J. Control Optim. ’87].
  - See, e.g., [Cassandras & Lafortune ’08].
This Work

- Bridge the gap: how are the two frameworks related
  - in theory?
  - in practice?

- Bridge the communities.

- Pedagogical, although results are new to our knowledge.

- Work in progress.
SUPERVISORY CONTROL
Supervisory Control: General Framework

- Plant generally modeled as deterministic finite-state automaton ($G'$): regular language

ExCAPE Review Meeting ()
Bridging the Gap
20 August 2013 5 / 20
Supervisory Control: General Framework

Plant generally modeled as deterministic finite-state automaton ($G'$): regular language

- Supervisor ($S$) can disable **controllable** events.

Specifications vary, but typically:
- **Safety**: all behaviors of the closed-loop system must be in some set of “good” behaviors (regular sublanguage of that of $G'$).
- **Non-blockingness**: supervisor must always allow system to reach an accepting (aka marked) state.
- **Maximal permissiveness**: supervisor must not disable more events than strictly necessary.
Supervisory Control: General Framework

closed-loop system $S/G$:

- Plant generally modeled as deterministic finite-state automaton ($G$): regular language
- Supervisor ($S$) can disable **controllable** events.
- Specifications vary, but typically:
  - **Safety**: all behaviors of the closed-loop system must be in some set of “good” behaviors (regular sublanguage of that of $G$).
  - **Non-blockingness**: supervisor must always allow system to reach an accepting (aka *marked*) state.
  - **Maximal permissiveness**: supervisor must not disable more events than strictly necessary.
Supervisor Synthesis: a basic problem

Simple Supervisory Control Problem (SSCP)

Given plant $G$, synthesize (if possible) supervisor $S$ such that:

- $S$ is non-blocking.
- $S$ is **maximally-permissive**, that is, for any other non-blocking supervisor $S'$:

$$L_m(S''/G) \subseteq L_m(S/G)$$
Supervisor Synthesis: a basic problem

Simple Supervisory Control Problem (SSCP)

Given plant $G$, synthesize (if possible) supervisor $S$ such that:

- $S$ is non-blocking.
- $S$ is maximally-permissive, that is, for any other non-blocking supervisor $S'$:
  $$L_m(S'/G) \subseteq L_m(S/G)$$

- We proved: Can reduce the standard supervisory control problem (safety and non-blocking) to SSCP (non-blocking only).
Supervisor Synthesis: a basic problem

Simple Supervisory Control Problem (SSCP)

Given plant $G$, synthesize (if possible) supervisor $S$ such that:

- $S$ is non-blocking.
- $S$ is maximally-permissive, that is, for any other non-blocking supervisor $S'$:

$$L_m(S'/G) \subseteq L_m(S/G)$$

- We proved: Can reduce the standard supervisory control problem (safety and non-blocking) to SSCP (non-blocking only).
- Can show that if a non-blocking supervisor exists, then the maximally-permissive non-blocking supervisor is **unique** and **state-based** ("memoryless").
REACTIVE SYNTHESIS
Reactive Synthesis

Reactive Synthesis Problem (RSP)

Given LTL formula $\phi$ with input/output atomic propositions, synthesize (if possible) a controller $M$ (Moore or Mealy machine) such that all behaviors of $M$ (inputs are uncontrollable) satisfy $\phi$.

This is the implementability problem [Pnueli-Rosner POPL 1989].
BRIDGING THE GAP
Summary: Main Differences

- Supervisory control has explicit plants – reactive synthesis does not.
- Supervisors are parents – controllers are ... controllers.
- Supervisory control asks for maximally-permissive controllers – these generally don’t exist in reactive synthesis.
- (Most of) supervisory control theory done in a finite-string setting – reactive synthesis is about infinite strings.
Reactive Synthesis with Plants

Inspired from [Kupferman et al CONCUR 2000]:

Reactive Synthesis Control Problem (RSCP)

Given plant $P$ and temporal logic formula $\phi$ synthesize (if possible) a strategy $f$ such that the closed-loop system satisfies $\phi$. 
Reactive Synthesis with Plants

Inspired from [Kupferman et al CONCUR 2000]:

Reactive Synthesis Control Problem (RSCP)

Given plant $P$ and temporal logic formula $\phi$ synthesize (if possible) a strategy $f$ such that the closed-loop system satisfies $\phi$.

- Plant modeled as a transition system with system states and environment states.
Reactive Synthesis with Plants

Inspired from [Kupferman et al CONCUR 2000]:

Reactive Synthesis Control Problem (RSCP)

Given plant $P$ and temporal logic formula $\phi$ synthesize (if possible) a strategy $f$ such that the closed-loop system satisfies $\phi$.

- Plant modeled as a transition system with system states and environment states.
- Strategy disables some successors of system states.
Inspired from [Kupferman et al CONCUR 2000]:

**Reactive Synthesis Control Problem (RSCP)**

Given plant $P$ and temporal logic formula $\phi$ synthesize (if possible) a strategy $f$ such that the closed-loop system satisfies $\phi$.

- Plant modeled as a **transition system** with **system** states and **environment** states.
- Strategy disables some successors of system states.
- Different versions of the problem depending on the temporal logic used: RSCP-LTL, RSCP-CTL, RSCP-CTL*, ...
For some formulas maximally-permissive strategies always exist:

**Theorem**

For any CTL formula $\phi := \text{AG EF } p$, where $p$ is a state formula, RSCP admits a unique maximally-permissive state-based strategy enforcing $\phi$ (if such a strategy exists).
Maximal Permissiveness in RSCP-CTL

For some formulas maximally-permissive strategies always exist:

**Theorem**

For any CTL formula $\phi :\equiv \text{AG EF } p$, where $p$ is a state formula, RSCP admits a unique maximally-permissive state-based strategy enforcing $\phi$ (if such a strategy exists).

We therefore define a variant of RSCP-CTL:

**RSCP-CTL_{max}**

Given plant $P$ and CTL $\phi :\equiv \text{AG EF } p$ compute (if it exists) the unique maximally-permissive state-based strategy enforcing $\phi$. 
Results

Relations between different synthesis problems:

Corollary 1: BSCP-NB ⬇️ SSCP ⬆️ special case

Theorem 5: SSCP → RSCP-LTL \( \max \) → RSP

\[ \text{supervisory control problems} \]

\[ \text{reactive synthesis problems} \]

Cf. technical report under preparation.

\( \text{\rightarrow\rightarrow\rightarrow\rightarrow: work in progress} \)
Results

Relations between different synthesis problems:

- BSCP-NB
- SSCP
- RSCP-CTL
- RSCP-LTL
- RSP

Section 3.4

Corollary 1

special case

Theorem 5

-section 3.5

supervisory control problems

reactive synthesis problems

Cf. technical report under preparation.

----- : work in progress
Reducing SSCP to RSCP-CTL\textsuperscript{max}

Main idea:

▶ DES can be transformed to a transition system.
  ▶ Marked states labeled with atomic proposition $acc$.

▶ Non-blockingness can be expressed in CTL:

$$\phi_{nb} := \text{AG EF } acc$$

i.e., from any reachable state, there exists a path to an accepting state.
Reducing SSCP to RSCP-CTL\textsubscript{max}

**Theorem**

Let $G$ be a DES plant and $P_G$ its transformation.

1. A non-blocking supervisor exists for $G$ iff a strategy enforcing $\phi_{nb} := AG EF acc$ exists for $P_G$.

2. Assuming supervisor/strategy exist, there is a 1-1 computable mapping between the unique non-blocking maximally-permissive state-based supervisor for $G$, and the unique maximally-permissive state-based strategy enforcing $\phi_{nb}$ on $P_G$. 
Discussion

First (to our knowledge) bridge between the reactive synthesis and DES/supervisory control problems and communities.
First (to our knowledge) bridge between the reactive synthesis and DES/supervisory control problems and communities.

This work would not have happened without ExCAPE!

- Partial observability.
- Modular, decentralized, hierarchical control architectures.
- Algorithmic procedures.
- $\omega$-regular supervisory control theory (cf. [Thistle '96]).
- Supervisory control of Petri nets.
Discussion

First (to our knowledge) bridge between the reactive synthesis and DES/supervisory control problems and communities.

This work would not have happened without ExCAPE!

Merely scratched the surface; expand bridge to:

- Partial observability.
- Modular, decentralized, hierarchical control architectures.
- Algorithmic procedures.
- $\omega$-regular supervisory control theory (cf. [Thistle ’96]).
- Supervisory control of Petri nets.
Case Studies

Two applications of supervisory control being considered:

- Collision avoidance in vehicular/robotic systems
  - New discrete-event approach based on discretization in time and space and on applying supervisory control techniques (disturbances, measurement errors, etc.)
  - Efficient algorithmic techniques
  - Initiating a collaboration with Hadas Kress-Gazit at Cornell

- Avoidance of concurrency bugs
  - Building on the Gadara Project (Michigan - HP Labs - Georgia Tech)
  - Petri net models are suitable to reduced control logic overhead
  - From deadlock avoidance to regular language specifications
  - Will collaborate with Stavros Tripakis at Berkeley
“Gadara” Methodology

C program
source code
 compile
control flow graph
translation
Petri net
 compile
instrumentation
control logic
 synthesis
control logic
 offline
compile
 Instrumented binary
 online
observe
control
observe
control
observe
control
ExCAPE and the Control Systems Community

Reaching out to control community: discrete-event, hybrid, cyber-physical systems

- Invited session on control problems in software systems at IEEE Conference on Decision and Control (CDC), December 2012
- Special session on ExCAPE at American Control Conference (ACC), June 2013
- Planned invited session on ExCAPE at International Workshop on Discrete Event Systems (WODES), May 2014
- More to come...
C. Cassandras and S. Lafortune.
*Introduction to Discrete Event Systems.*

A. Church.
Logic, arithmetic and automata.

Open systems in reactive environments: Control and synthesis.

A. Pnueli and R. Rosner.
On the synthesis of a reactive module.

P. Ramadge and W. Wonham.
Supervisory control of a class of discrete event processes.

J.G. Thistle.
Supervisory control of discrete event systems.

W. Wonham and P. Ramadge.
On the supremal controllable sublanguage of a given language.