



University of Illinois at Chicago  
Department of **Computer  
Science**

Automatic Generation of Control  
Supervisors for Discrete Event Systems

*Ugo Buy*

14 May 2007





# Acknowledgements

- Professor Houshang Darabi
  - Director, Discrete Event System Control (DESC) Lab,  
Dept. of Mechanical and Industrial Engineering, UIC
- NIST/ATP program
- Davide Bonicelli, Liviu Grigore, Sambhavi Jayavelan, Jing Liu, Mihai Lehene, Haritha Siddabathuni, Haisheng Wang, Zheng Zhang





# Concurrent and real-time systems

- What is a concurrent software system? One that contains multiple processes (or threads, tasks, etc.) that can be executed in parallel
- Real-time system also subject to timing constraints (e.g., deadlines on completion of computations)
- Special issues:
  - Possibility of concurrency errors and real-time errors (e.g., mutex violations, deadlocks, missed deadlines)
  - Difficult errors to detect through traditional testing
  - Automatic verification also notoriously difficult





## Analysis challenge

- State explosion problem: Arbitrary interleaving of computations performed by different processes.
- An example: Given  $n$  processes, each performing one computation, how many system states are there?



- Answer: Exponential growth in  $n$ , or  $2^n$ 
  - Quickly gets quite large





# Supervisory control of DES

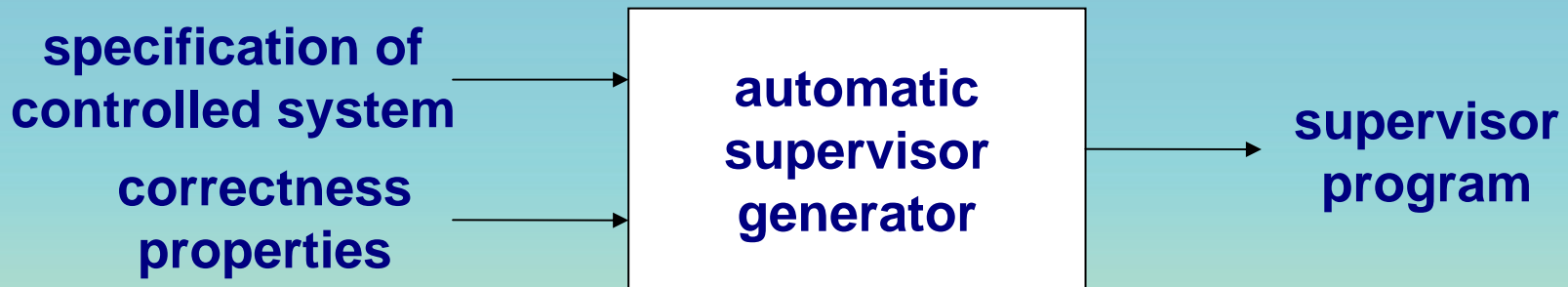
- Given system model, add controller enforcing desired concurrency and timing properties to model
  - Sometimes much more tractable than verification
  - Support dynamic reconfiguration through automatic supervisor synthesis
  - Similar mathematical models to verification (e.g., finite state automata and Petri nets)





# Supervisor generation

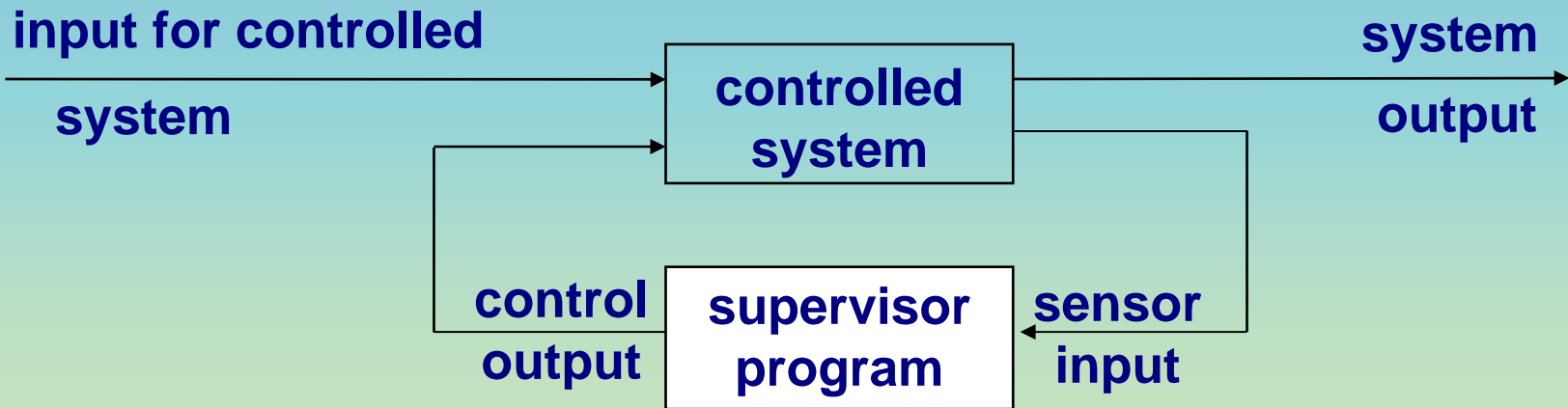
- Given system model, add controller enforcing desired concurrency and timing properties to model





# Execution of controlled system

- Supervisor program prevents undesired behaviors from occurring





# Manufacturing application

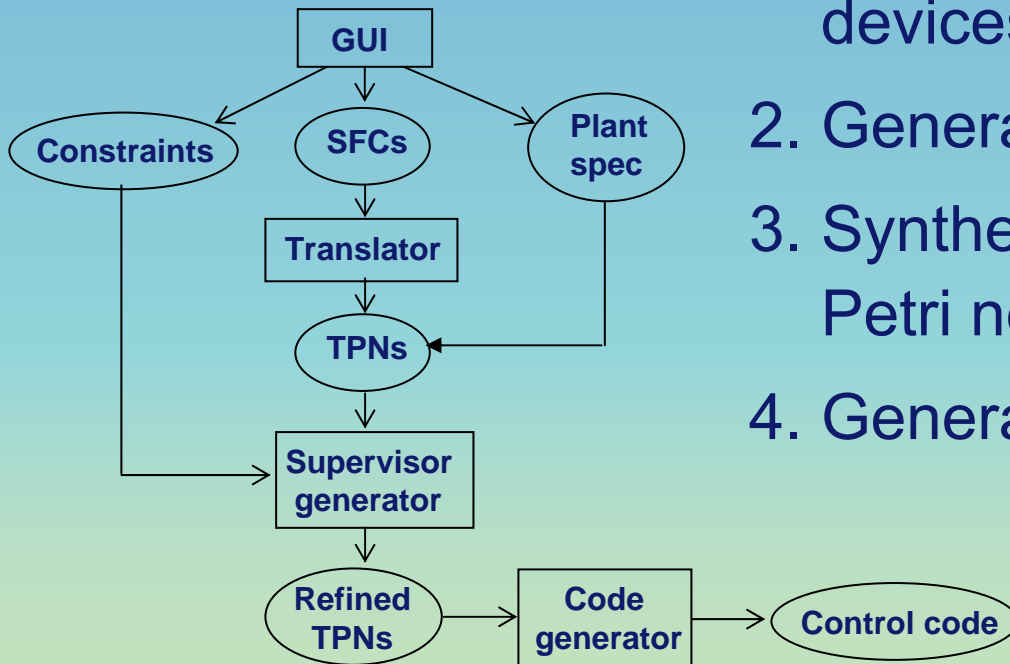
- Dynamic reconfiguration of discrete manufacturing systems
- Funding from NIST in collaboration with Starthis, Inc. of Arlington Heights, Illinois
- Rationale:
  - Control programs are hard to write and to maintain
  - Flexible manufacturing demands rapid reconfiguration
  - Possibility of deadlock, safety property violations, deadline violations in manufacturing plants
  - Disastrous consequences sometimes possible





# Our flow of supervisor synthesis

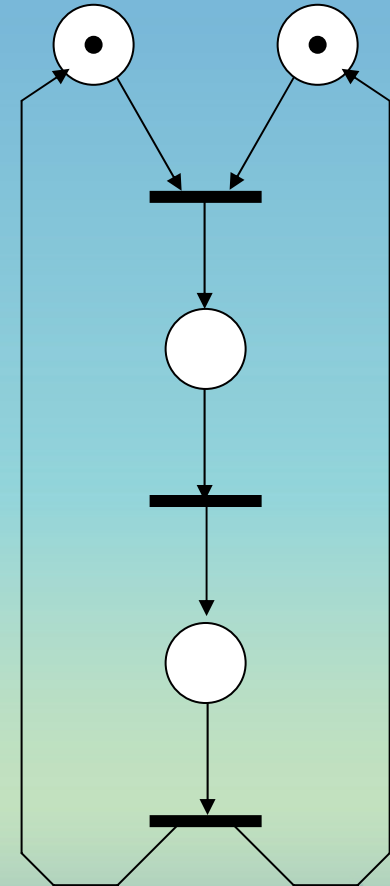
1. Specify behavior of manufacturing devices and correctness properties
2. Generate Petri net
3. Synthesize control supervisor for Petri net
4. Generate target code





# Petri nets

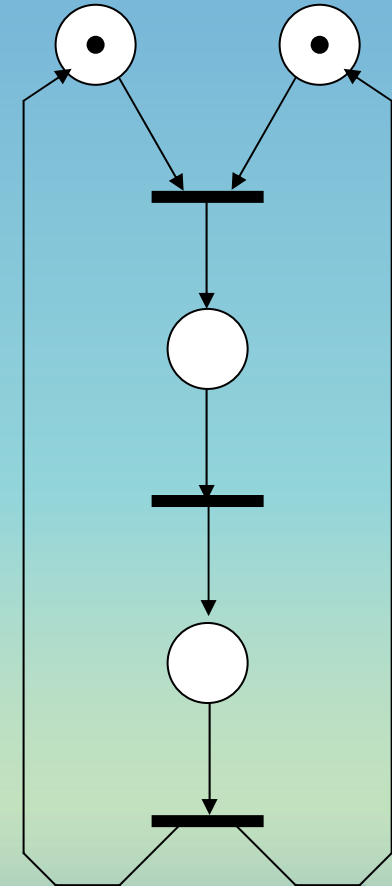
- Special graphs with two node kinds, *places* and *transitions*
  - Places shown as circles, transitions as bars
- Places can have tokens, shown as dark circles
- Directed arcs connect transition to places and vice versa





# Petri nets: Transition firing rule

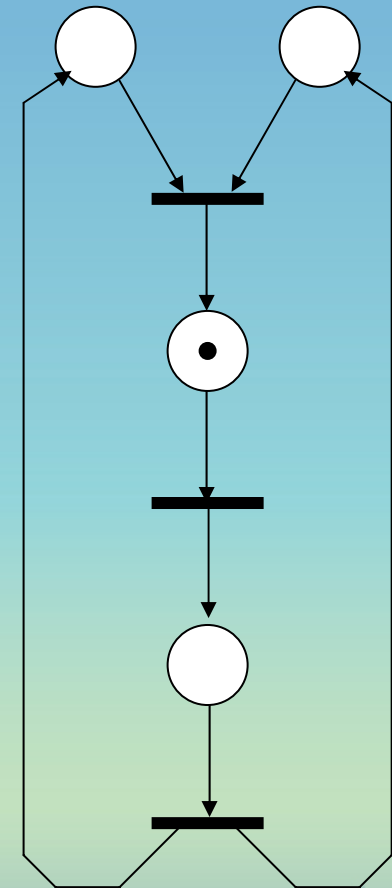
- Transition is *enabled* if all places feeding into it have a token
- Enabled transition can *fire*
- Transition firing causes:
  - Token to disappear from places feeding into transition
  - Token to appear in places that transition feeds





# Petri nets: Transition firing rule

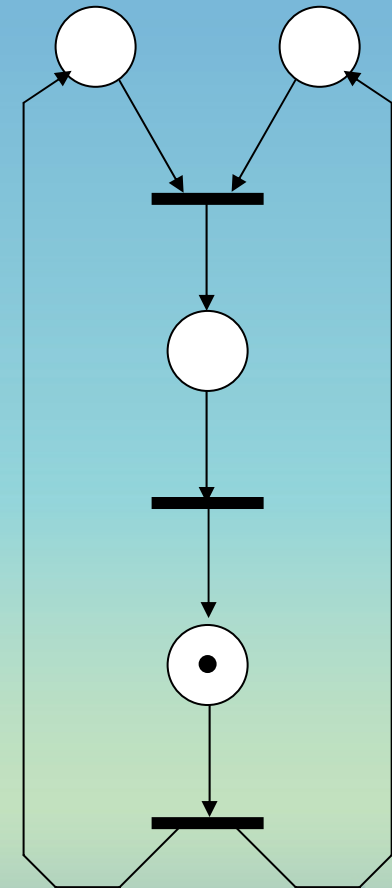
- Transition is *enabled* if all places feeding into it have a token
- Enabled transition can *fire*
- Transition firing causes:
  - Token to disappear from places feeding into transition
  - Token to appear in places that transition feeds





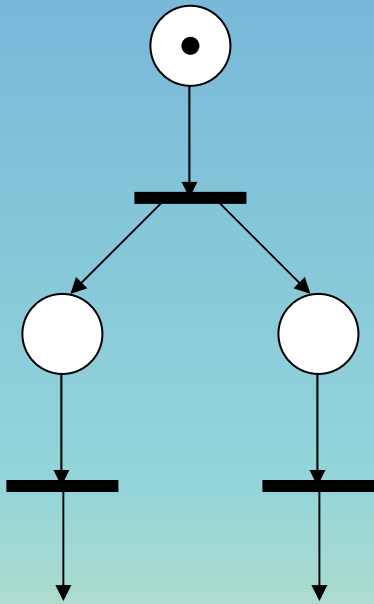
# Petri nets: Transition firing rule

- Transition is *enabled* if all places feeding into it have a token
- Enabled transition can *fire*
- Transition firing causes:
  - Token to disappear from places feeding into transition
  - Token to appear in places that transition feeds

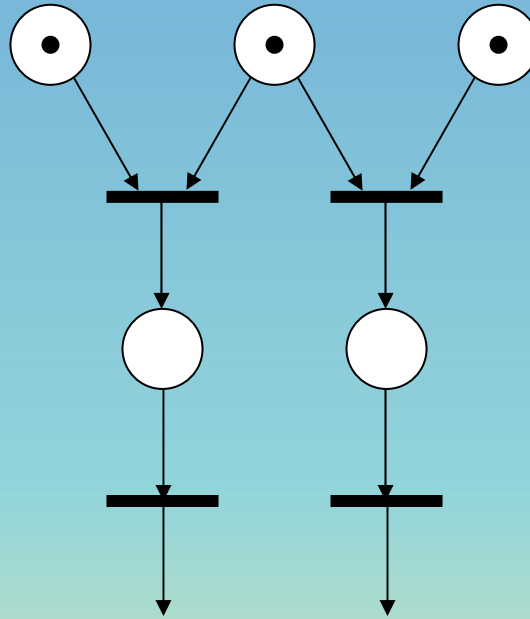




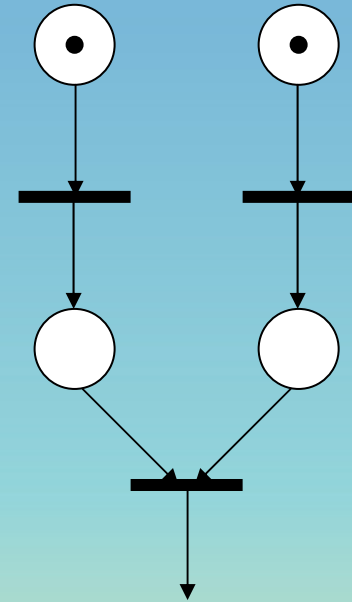
# Examples of Petri net models



Parallel computation



Choice



Synchronization





# Approach for supervisor generation

- Different methods depending on properties being enforced
  - Mutual exclusion properties and freedom from deadlock
    - Linear algebra based on matrix representation of net
  - Real-time properties
    - Technique based on *net unfoldings*





## Current projects

- Enforcing mutex and liveness in concurrent Java programs
- Coordination of medical, transportation, and communication resources in response to mass-casualty situations





Thank you very much!

