Demonstration of an Automated Control Synthesis Tool for Manufacturing

Lawrence Holloway, Andy Callahan, Xiaoyi Guan, Praveen Yasarapu, John O'Rear

Center for Manufacturing Systems
and Dept. of Electrical Engineering
University of Kentucky
Lexington, KY USA

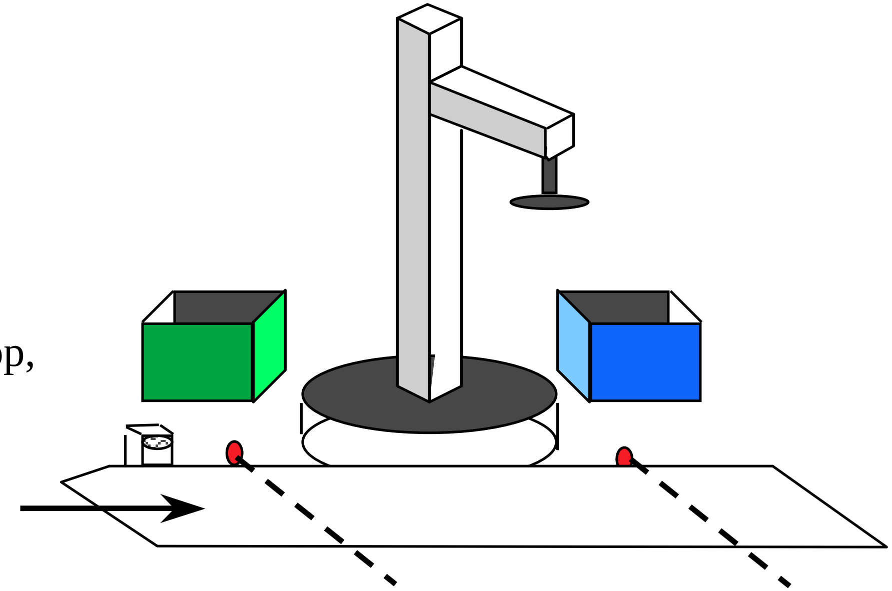
Example

Robot Assembly Cell

Actuators:

- Conveyor: on, off
- Robot-turn: on-left, on-right, off
- Arm: on-up, on-down, off
- Electromagnet: on, off

Sensors: S1, S2, L-bin, R-bin, home, upstop, downstop, block-type



Current Practice in Control Design

User must:

- understand system and interactions
- write control code
- debug control code to confirm spec's are met

Problems:

- different users/designers must each learn system
- unexpected interactions within code
- difficulty determining if controlled system satisfies all specs.
- repeated writing and debug $\Rightarrow$ $$$$

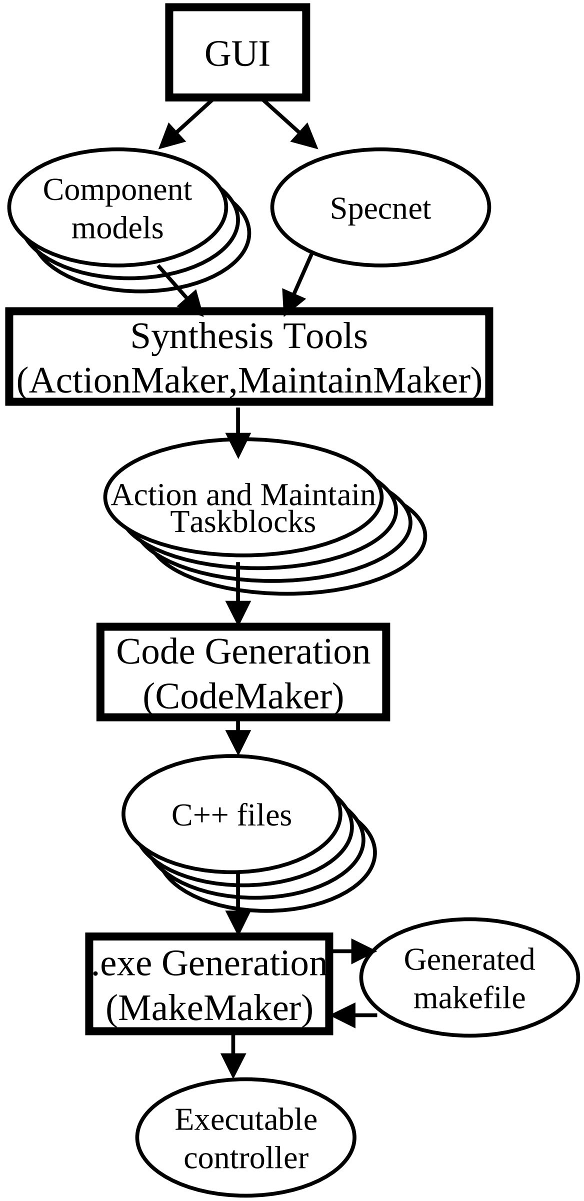
Current practice is inadequate when frequent modifications required.

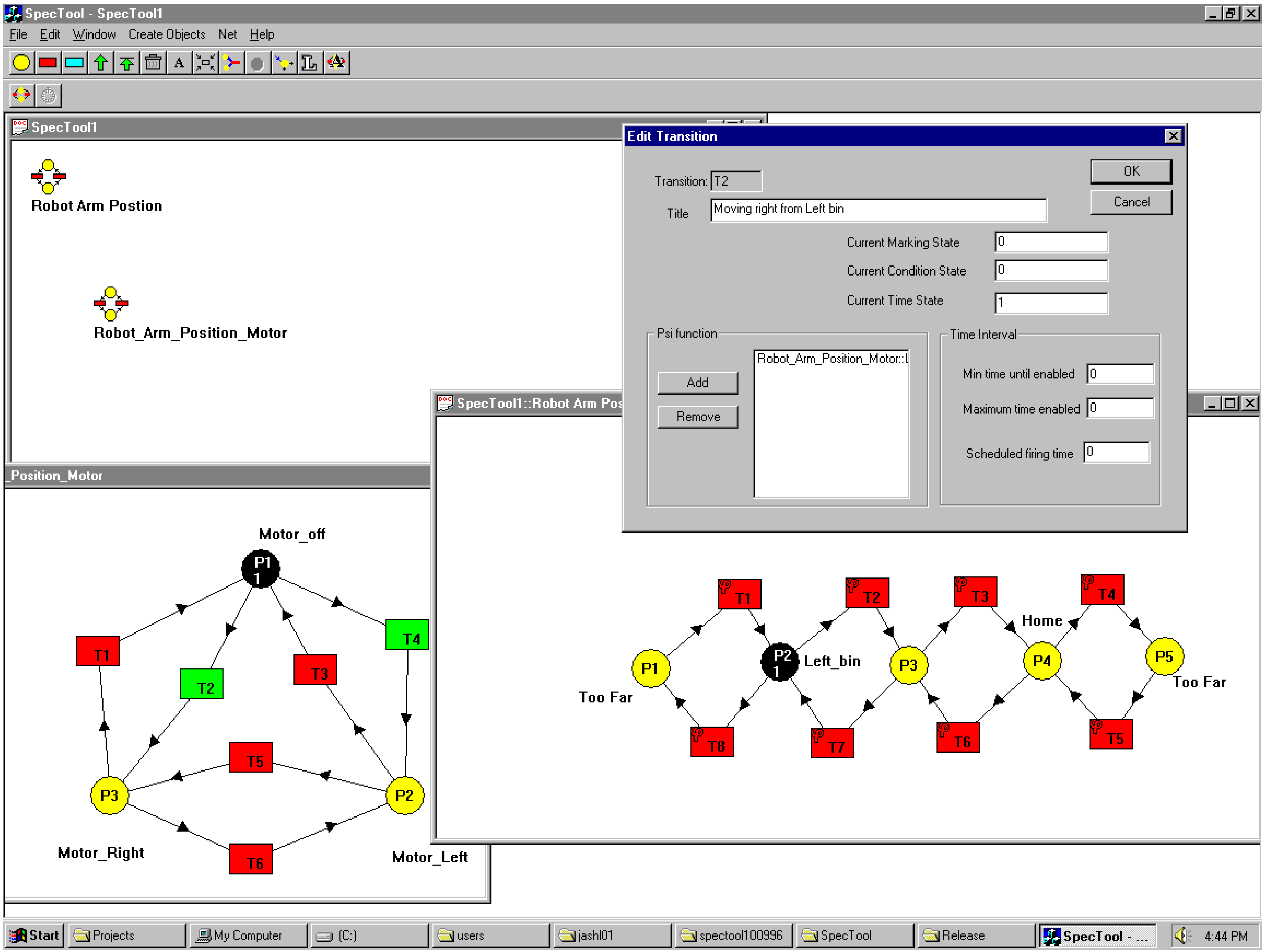
Spectool

Spectool: Software tool for automatic synthesis of control code for manufacturing systems

Inputs models of system components and high-level description of desired state behavior.

Outputs executable control program.





SpecTool - SpecTool1
File Edit Window Create Objects Net Help
SpecTool1
Robot Arm Postion
Robot_Arm_Position_Motor
Position_Motor
Motor_off
P1
1
T1
T2
T3
T4
T5
T6
P2
P3
Motor_Right
Motor_Left
SpecTool1::Robot Arm Pos
Edit Transition
Transition: T2
Title
Moving right from Left bin
OK
Cancel
Current Marking State
0
Current Condition State
0
Current Time State
1
Psi function
Add
Remove
Robot_Arm_Position_Motor::l
Time Interval
Min time until enabled
0
Maximum time enabled
0
Scheduled firing time
0
P1
Too Far
P2
1
Left_bin
P3
P4
Home
P5
Too Far
T1
T2
T3
T4
T5
T6
T7
T8
Start
Projects
My Computer
(C:)
users
jashl01
spectool100996
SpecTool
Release
SpecTool - ...
4:44 PM

Condition Models

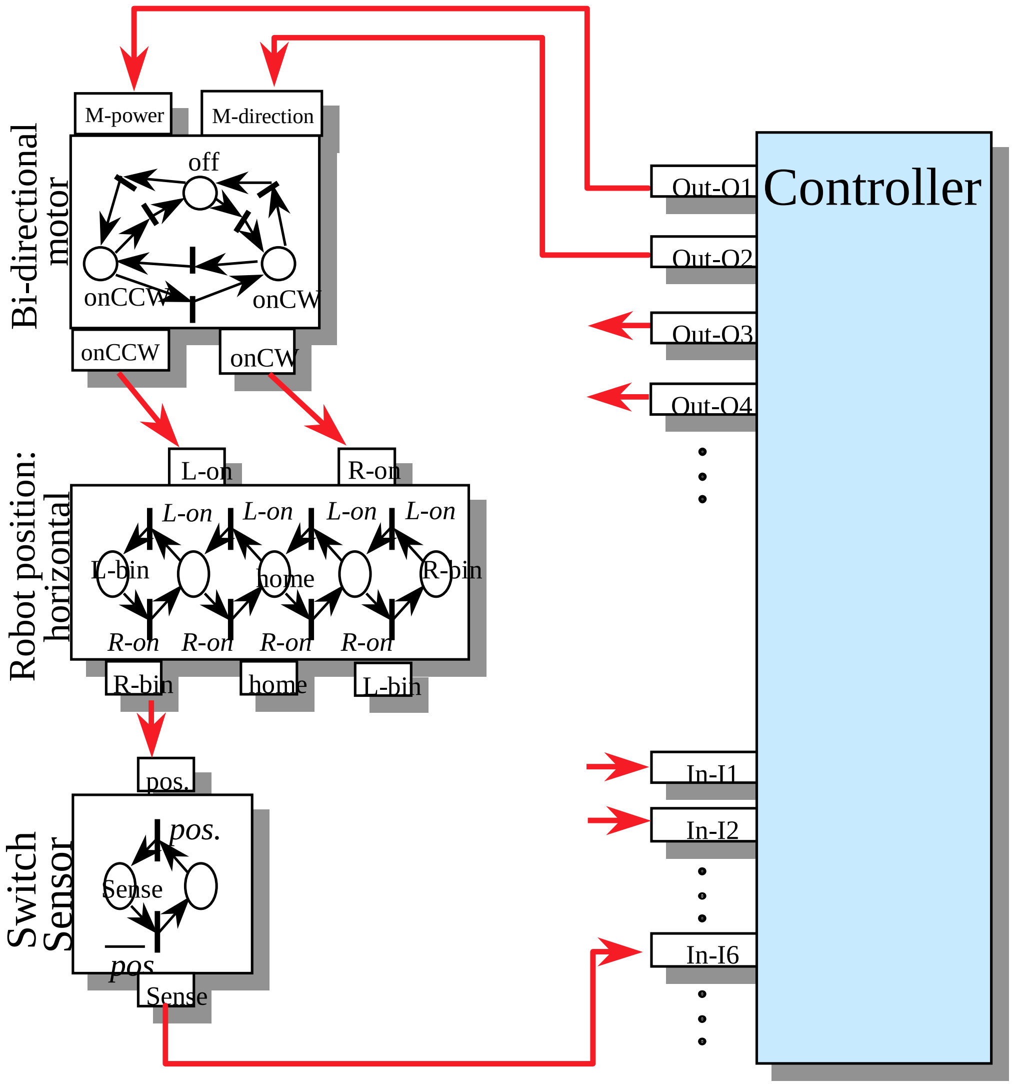
- *Conditions* are signals. Output conditions depend on state. Input conditions influence state change.

- Discrete state represented by condition Petri Net. Places output condition signals, transitions enabled by condition signals

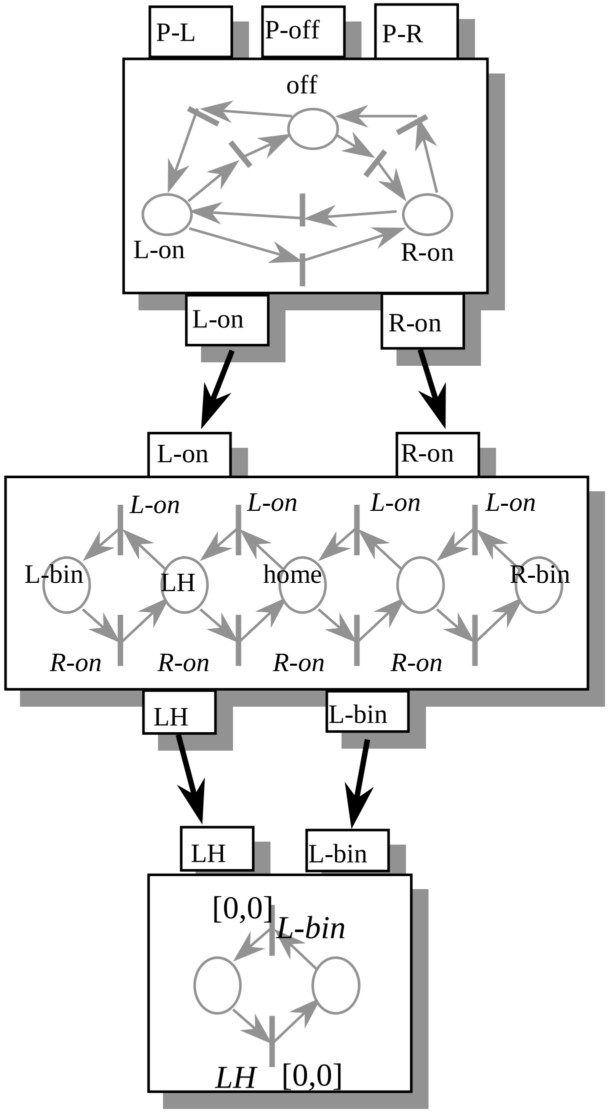
- Modular

- Concurrent

- Avoids state explosion of “remembering” past events



Condition System Model



AllC : **Universe of Condition Labels:**

includes negations, c, ¬c

Net: $G = (P_G, T_G, A_G, C_G)$

P_G, T_G, A_G: defines Petri Net

C_G maps conditions to places,transitions

Output conditions depend on state:

$g(m) = \{c \mid c \in C_G(p)$ for some marked $p\}$

$C_{out.G}$ is set of all output conditions

Next-state set $f(m,C)$ depends on input conditions:

Transition set T fires only if:

- state enabled (input places of T marked)
- conditions in $C_G(t)$ are true ($\forall t \in T$)

Resulting marking: (*standard PN rule)*

$$m'(p) = m(p) + |{}^{(t)}p \cap T| - |p^{(t)} \cap T|$$

Control Synthesis Theory

Requires model of system capabilities in order to synthesize control.

Synthesis Goal: Given specifications of desired closed loop operation and model of plant capabilities, *automatically* determine a feedback controller to achieve the desired operation specifications.

- control is *synthesized to be correct*.
- changes in specifications give automatic revision of control
- plant model also useful for simulation and analysis

Synthesis goal requires theory of techniques and tools that can provably guarantee satisfaction of operation specifications.

Control problem

- Control goal: Develop control to achieve high-level sequencing specifications
  - Control synthesis problem is determining the low-level interactions necessary to achieve the high-level spec.
  - Control issue *is* filling in the details
- Key point:

Control as *navigator*

vs.

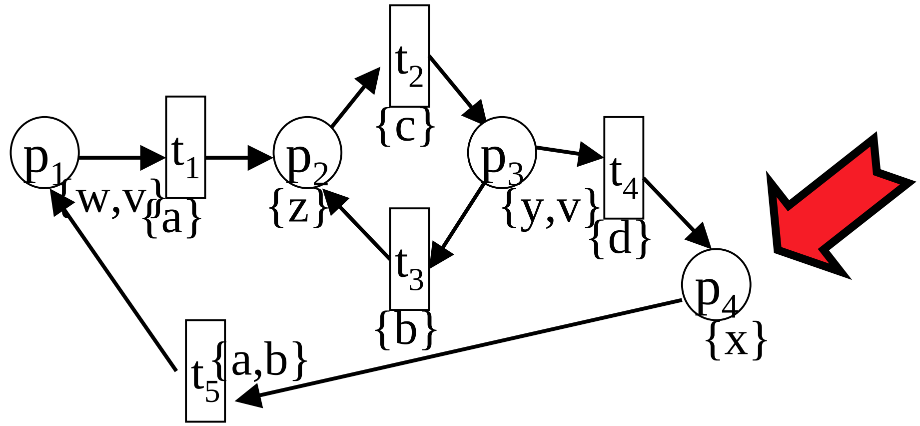
control as *traffic-cop*

Synthesized Control Blocks

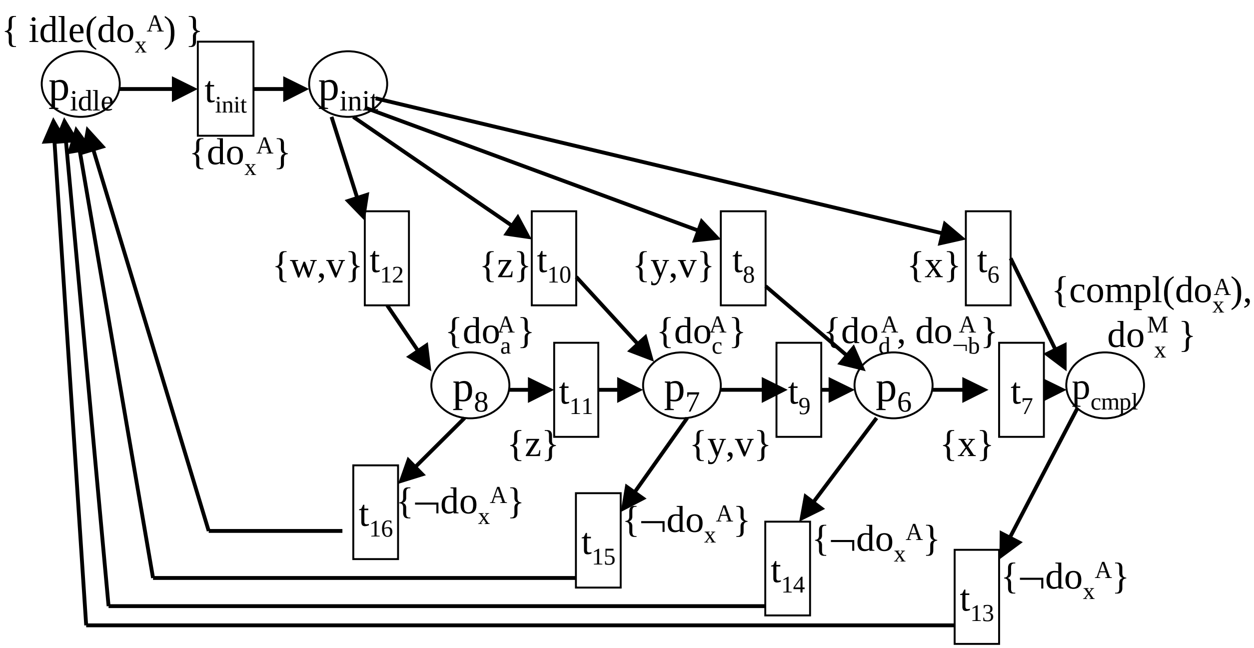
- **Control synthesis problem is determining the low-level interactions necessary to achieve the high-level spec.**
  - **Control issue *is* filling in the details**
- Control synthesized for individual components. Two types of blocks:
  - ActionBlock: drive component to target condition
  - MaintainBlock: keep component in target condition
- Blocks combined
  - Sequentially -- for sequential control specifications
  - Hierarchically -- for dependent components

ActionBlock Synthesis

Plant model:

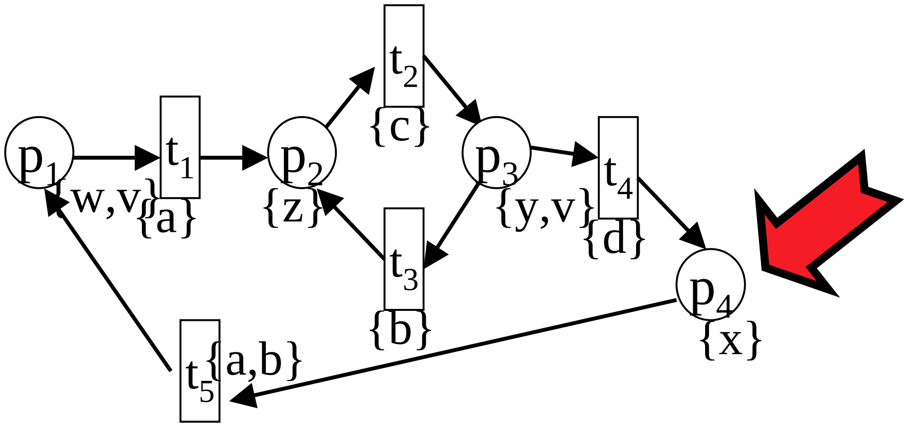


Synthesized ActionBlock for target "x"

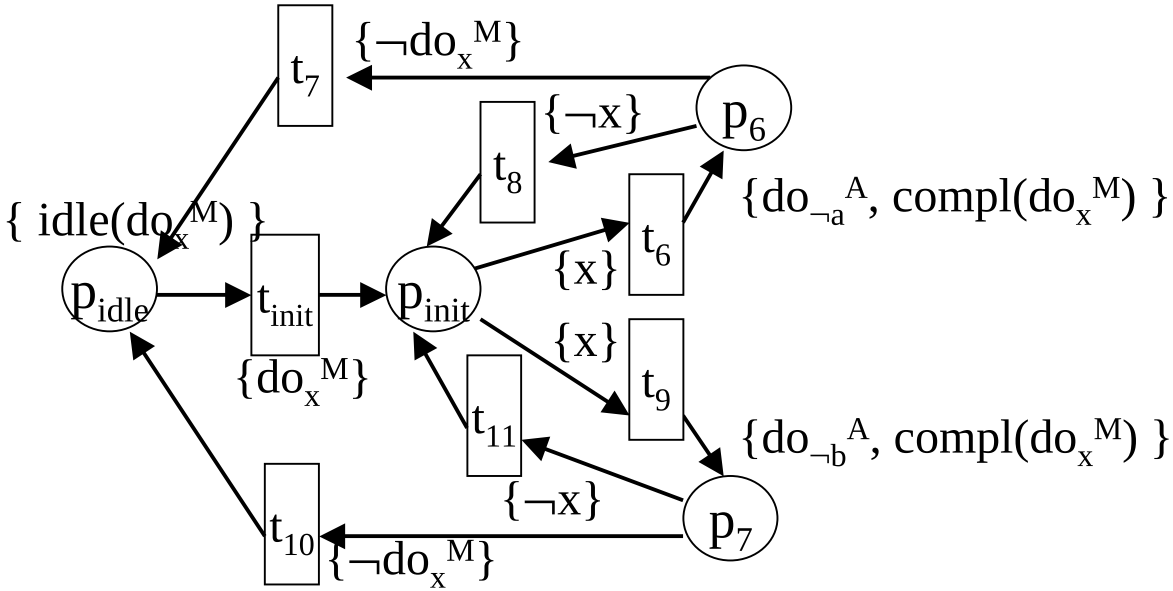


MaintainBlock Synthesis

Plant model:

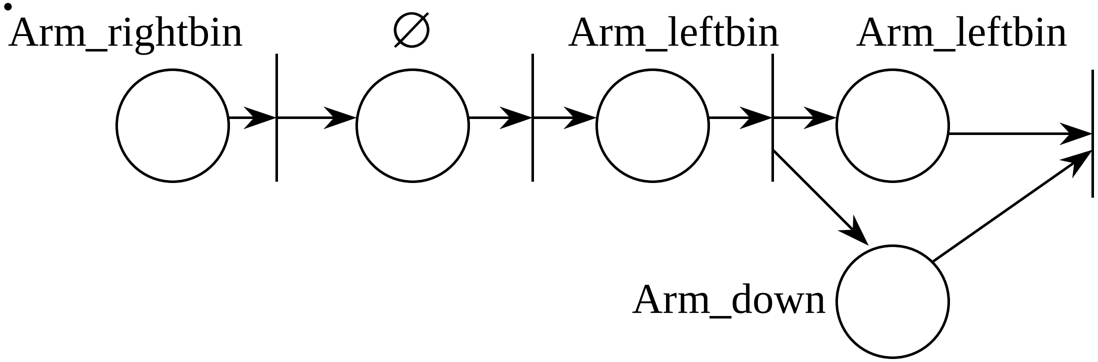


Synthesized MaintainBlock for target “x”

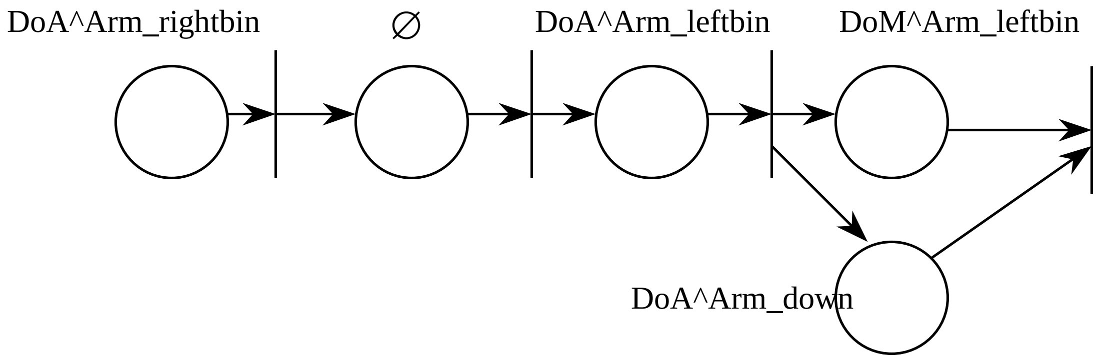


Control Specification

- Specification is a condition system describing desired plant behavior.



- Outputs of places in specs are used as targets in developing action and maintain blocks.



Code Synthesis

CodeMaker: Individual Actionblock and Maintainblock net files are converted into C++ code. Top level spec also converted into C++ code.

MakeMaker: automatically compiles project into executable

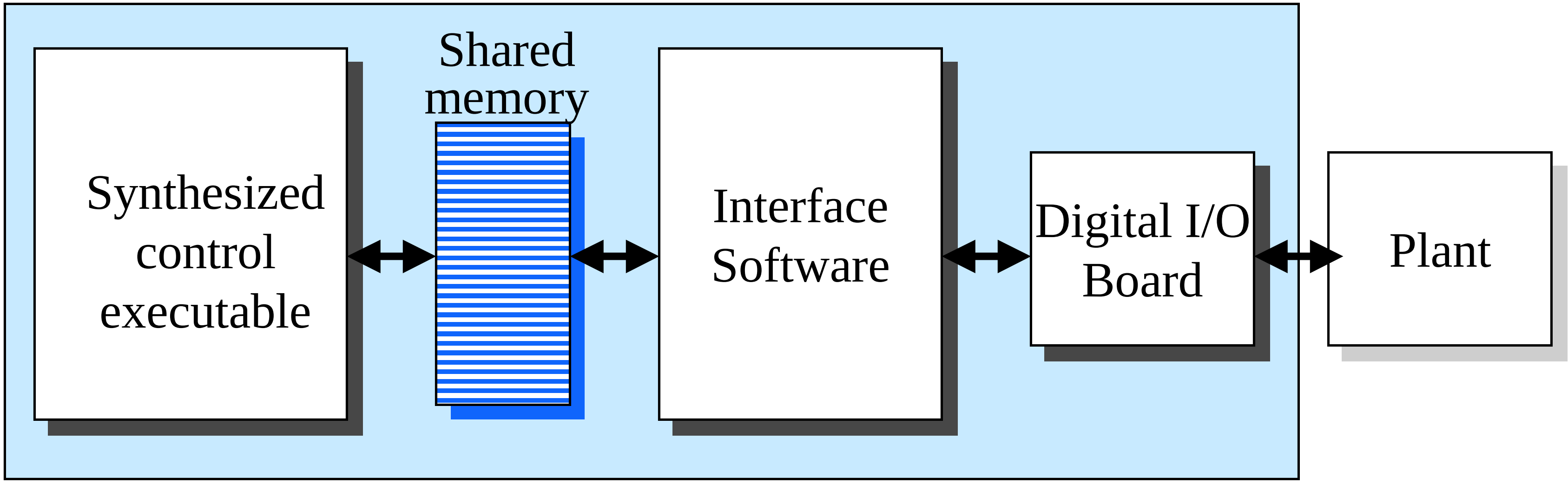
```
void SCO_doA_CHomeH_Class::StateEval()
{

...

      if ((SCO_doA_CHomeH_List[0]) )
      {
         if ((pCondTable->GetConditionValue("doA_CHomeH") > 0))
         {
            SCO_doA_CHomeH_List[0] = false;
            pCondTable->UpdateConditionValue("Idle_CHomeH", -1);
            SCO_doA_CHomeH_List[1] = true;
         }
      }
      if ((SCO_doA_CHomeH_List[1]) )
      {
         if ((pCondTable->GetConditionValue("CHomeH") > 0))
         {
            SCO_doA_CHomeH_List[1] = false;
            SCO_doA_CHomeH_List[2] = true;
            pCondTable->UpdateConditionValue("Cmpl_CHomeH", 1);
            pCondTable->UpdateConditionValue("doM_CHomeH", 1);
         }
      }
      if ((SCO_doA_CHomeH_List[3])  && (pCondTable->GetConditionValue("Cmpl___doA_MotorRight") > 0) &&
(pCondTable->GetConditionValue("Cmpl___doA__qNOTq__MotorLeft") > 0))
      {
         if ((pCondTable->GetConditionValue("CHomeH") > 0))
         {
            SCO_doA_CHomeH_List[2] = true;
            pCondTable->UpdateConditionValue("Cmpl_CHomeH", 1);
            pCondTable->UpdateConditionValue("doM_CHomeH", 1);
            SCO_doA_CHomeH_List[3] = false;
            pCondTable->UpdateConditionValue("doA_MotorRight", -1);
            pCondTable->UpdateConditionValue("doA__qNOTq__MotorLeft", -1);
         }
      }
...
```

Executable

- Executable interacts with hardware through shared memory interface
  - device independent synthesis



Issues and Current Work

- Specification Complexity:
  - current software version: literal vs. “implied” specs
- Model Delays:
  - solution: condition signals that refer to timing objects
- Unobserved states:
  - solution: automated synthesis of state observers (WODES2000)
- Coordination of synthesized control blocks
  - current work on synthesis of supervisor over control

- Further information on this project can be found in <u>IEEE Transactions on Systems, Man, and Cybernetics</u>, Part B, Volume 30(5), Oct. 2000. (to appear).