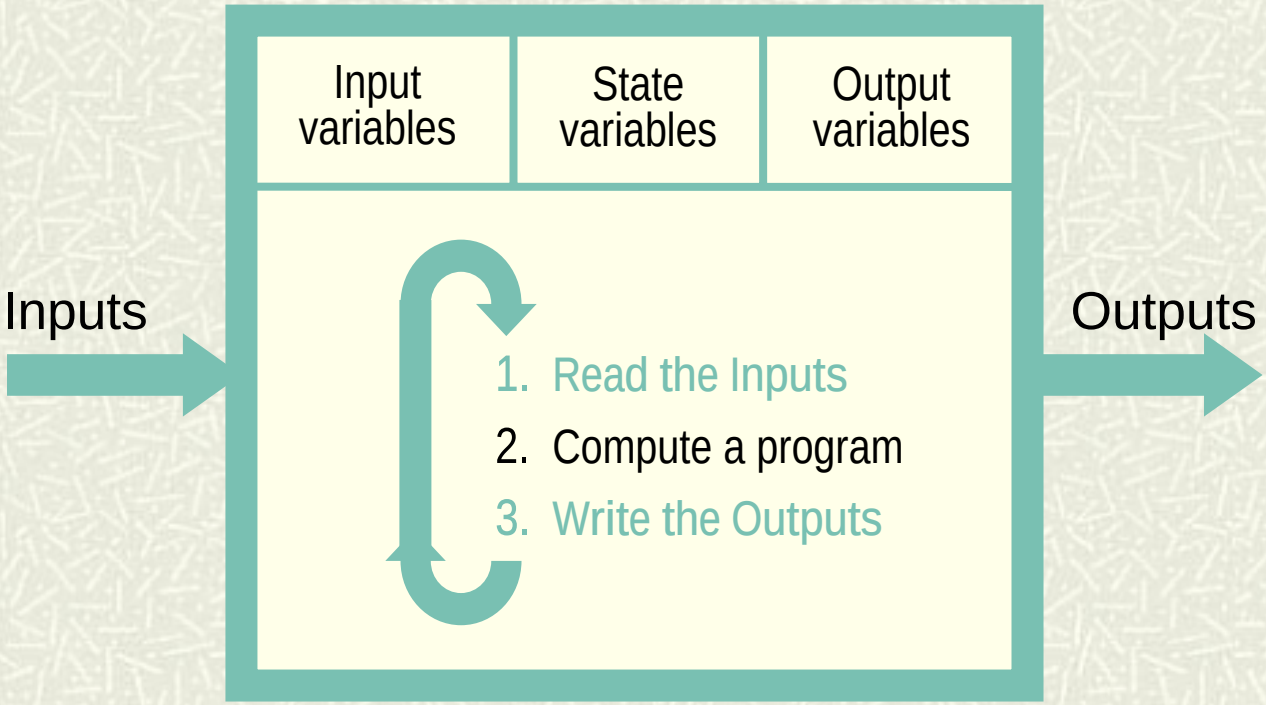
Translatable Finite State Time Machine

Krzysztof Sacha, Warsaw University of Technology

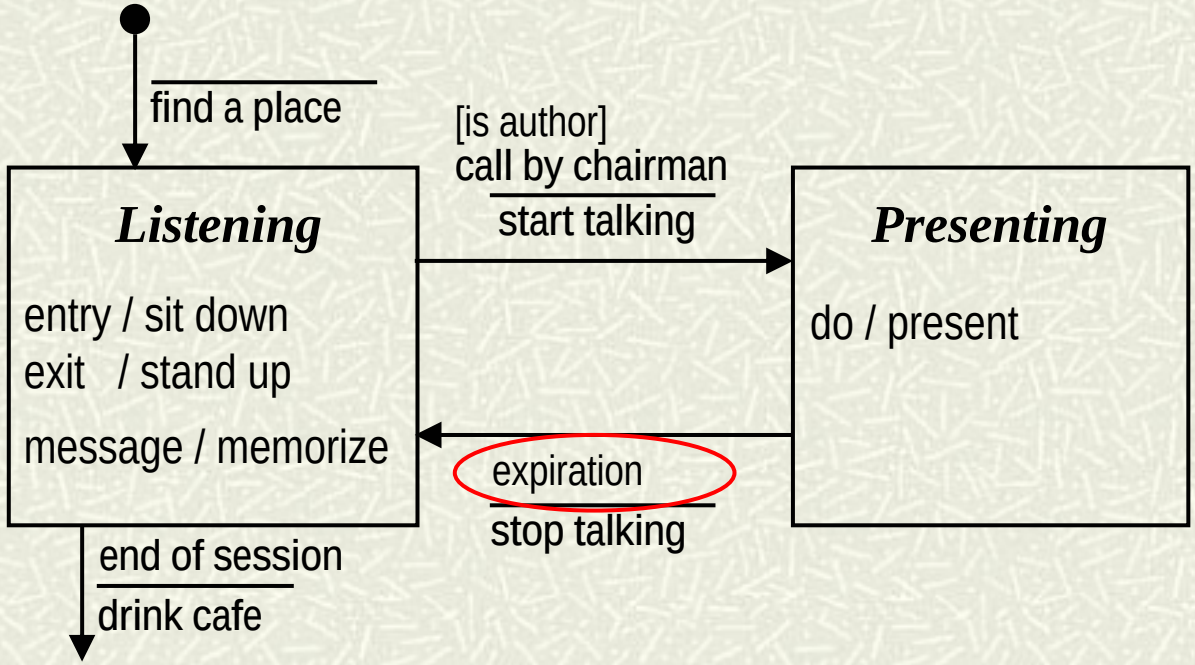
Contents

- Introduction
- UML Statechart
- Finite state time machine
- Program generation
- Conclusions

Introduction – PLC controller

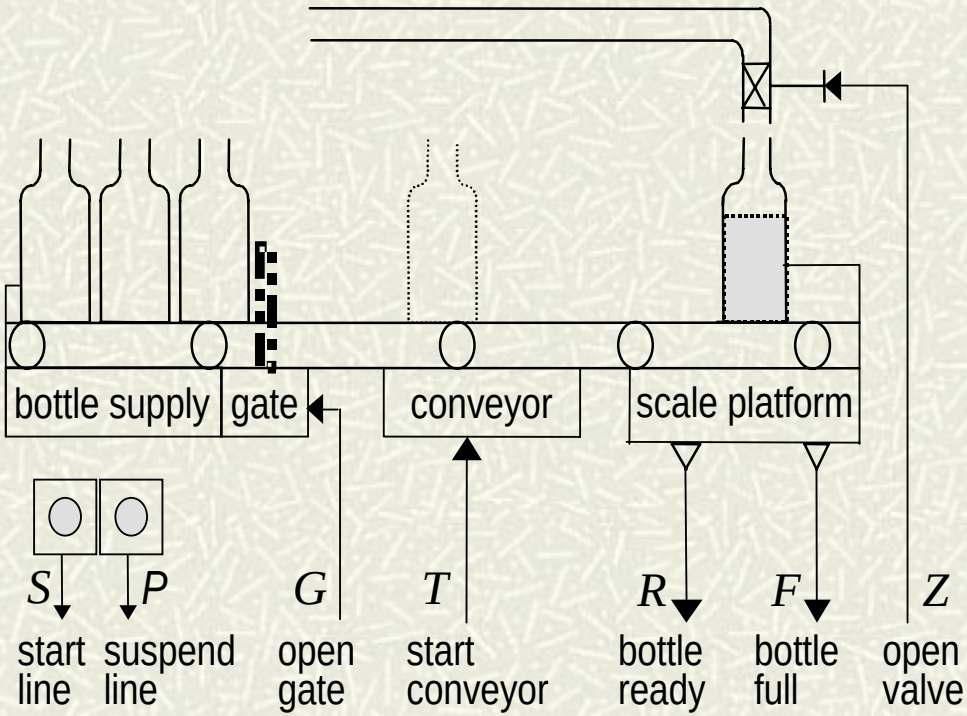


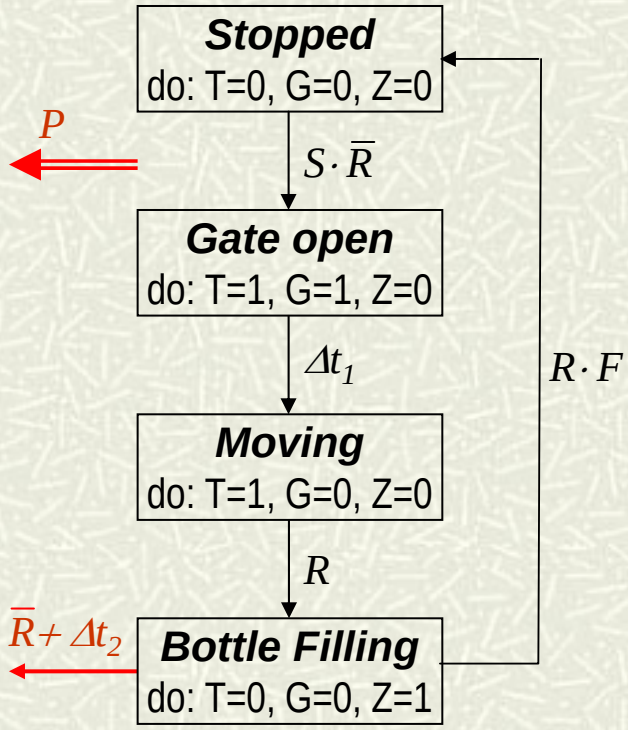
UML Statechart



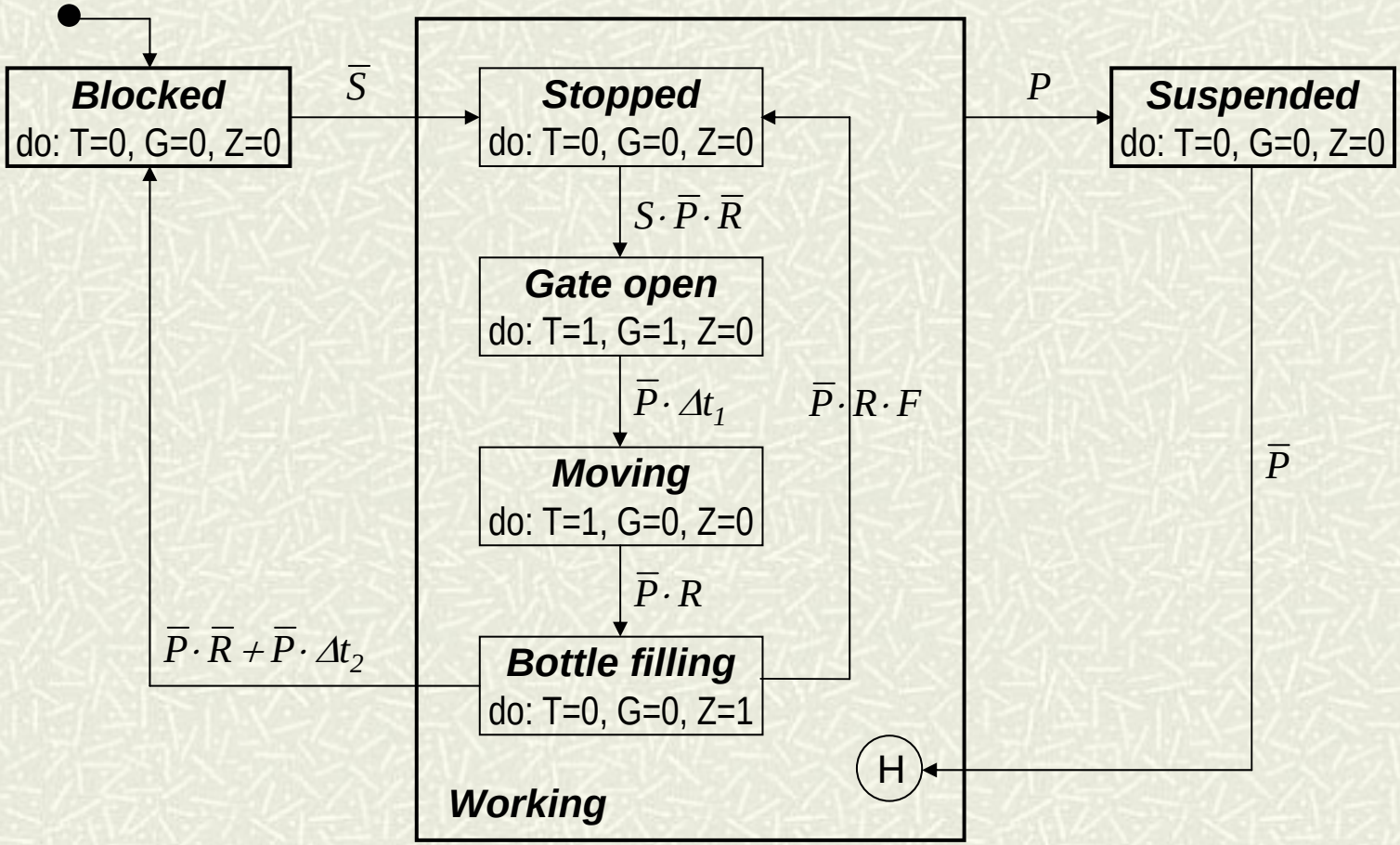
+ hierarchy of states

Example (1): A bottling line





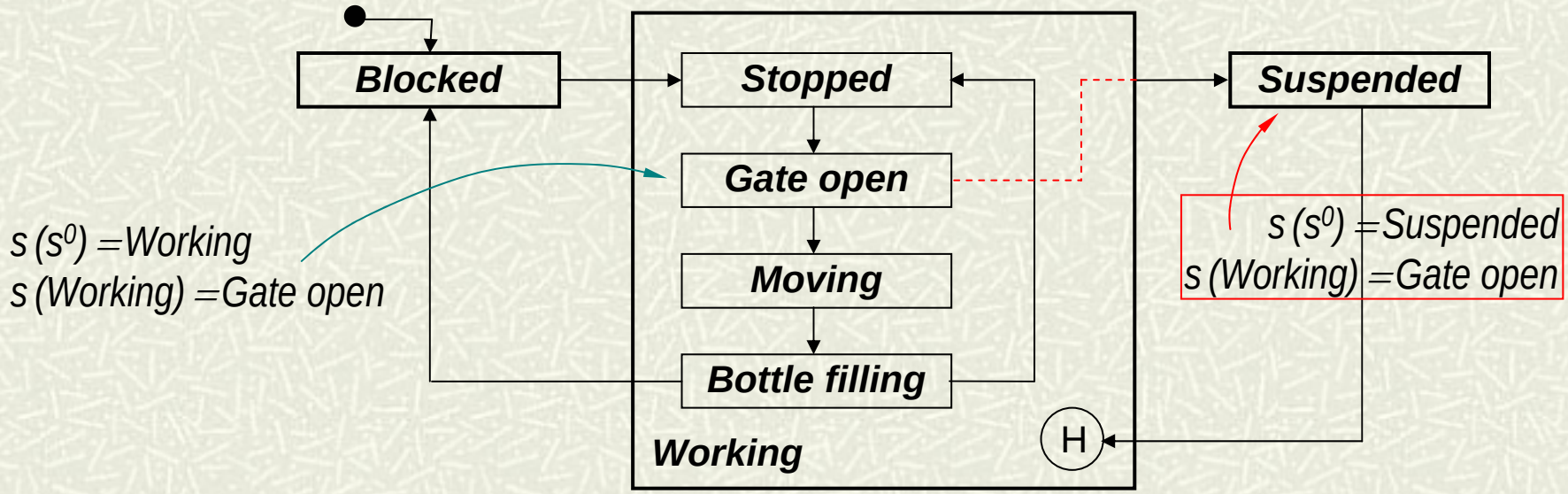
Example (2): State diagram



Hierarchy of states

$H = (Sc, h)$	hierarchy of states
Sc	set of states of the statechart
$h : Sc \rightarrow 2^{Sc}$	hierarchy function
$s : Sc \rightarrow Sc$	compound state of the hierarchy

Example (3): Hierarchy of states



$Sc = \{ s^0, Blocked, Working, Suspended, Stopped, Gate\ open, Moving, Bottle\ filling \}$

$h(s^0) = \{ Blocked, Working, Suspended \}$

$h(Working) = \{ Stopped, Gate\ open, Moving, Bottle\ filling \}$

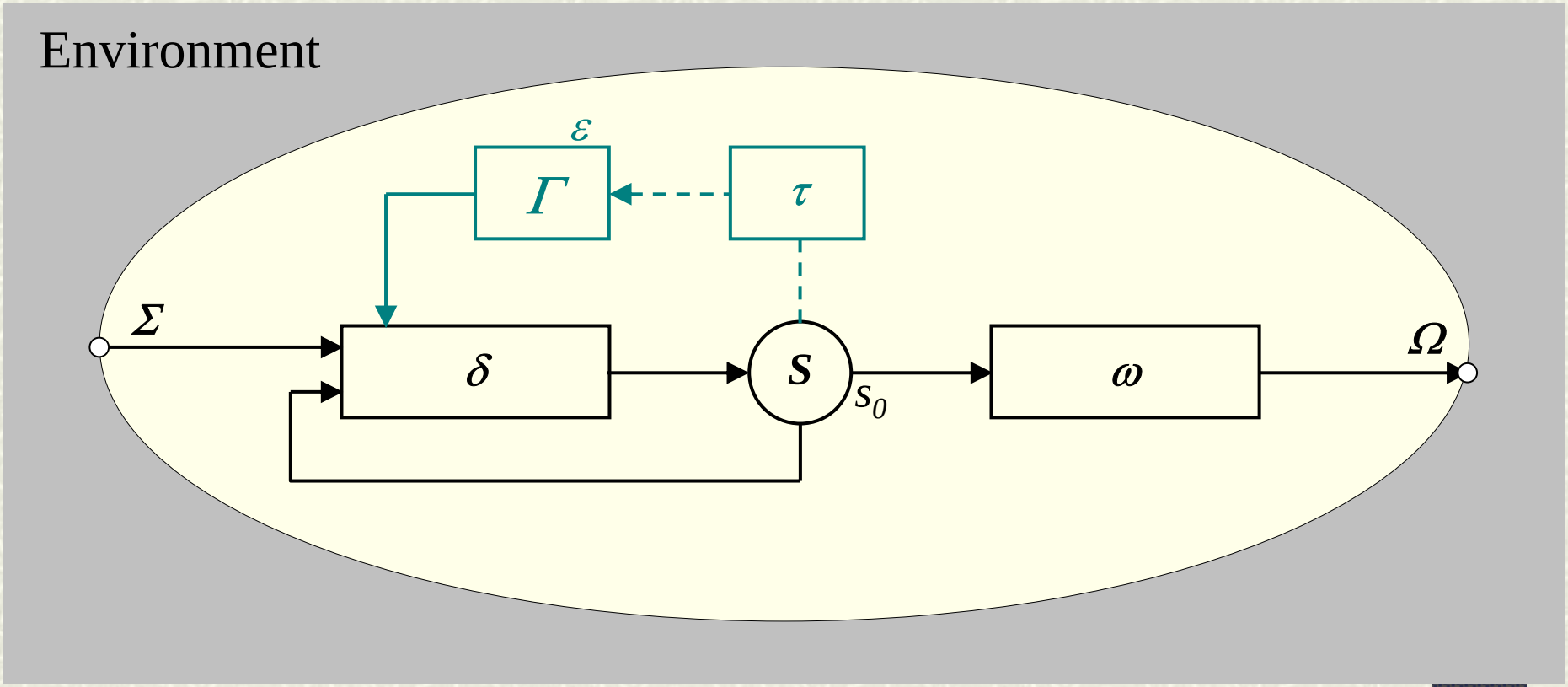
$h(\ldots) = \phi$

Example (4): Coding of states

M1	*M2*	*M3*	*M4*	Line states
0	0			*Blocked*
1	0	0	0	*Stopped*
1	0	0	1	*Gate open*
1	0	1	1	*Moving*
1	0	1	0	*Bottle filling*
1	1	0	0	*Suspended–Stopped*
1	1	0	1	*Suspended–Gate open*
1	1	1	1	*Suspended–Moving*
1	1	1	0	*Suspended–Bottle filling*

Finite state time machine (FSTM)

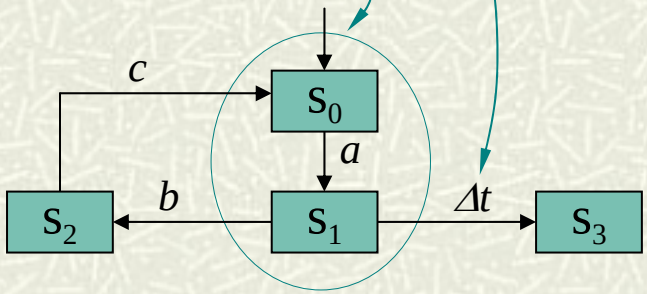
$A = (S, \Sigma, \Omega, \delta, \omega, s_0 , \Gamma, \tau, \varepsilon)$



FSTM: Time extension

Γ set of timer symbols

$\tau : \Gamma \rightarrow 2^S \times R^+$ timer function



$\delta: S \times \Sigma \times 2^\Gamma \rightarrow S$ transition function

$\varepsilon \in R^+$ granularity of time

FSTM: Execution of a machine

$\Gamma = \{ t^1, t^2, \ldots, t^n \}$ the set of timers

$\tilde{t} = \{ \tilde{t}^1, \tilde{t}^2, \ldots, \tilde{t}^n \}$ valuation of timers

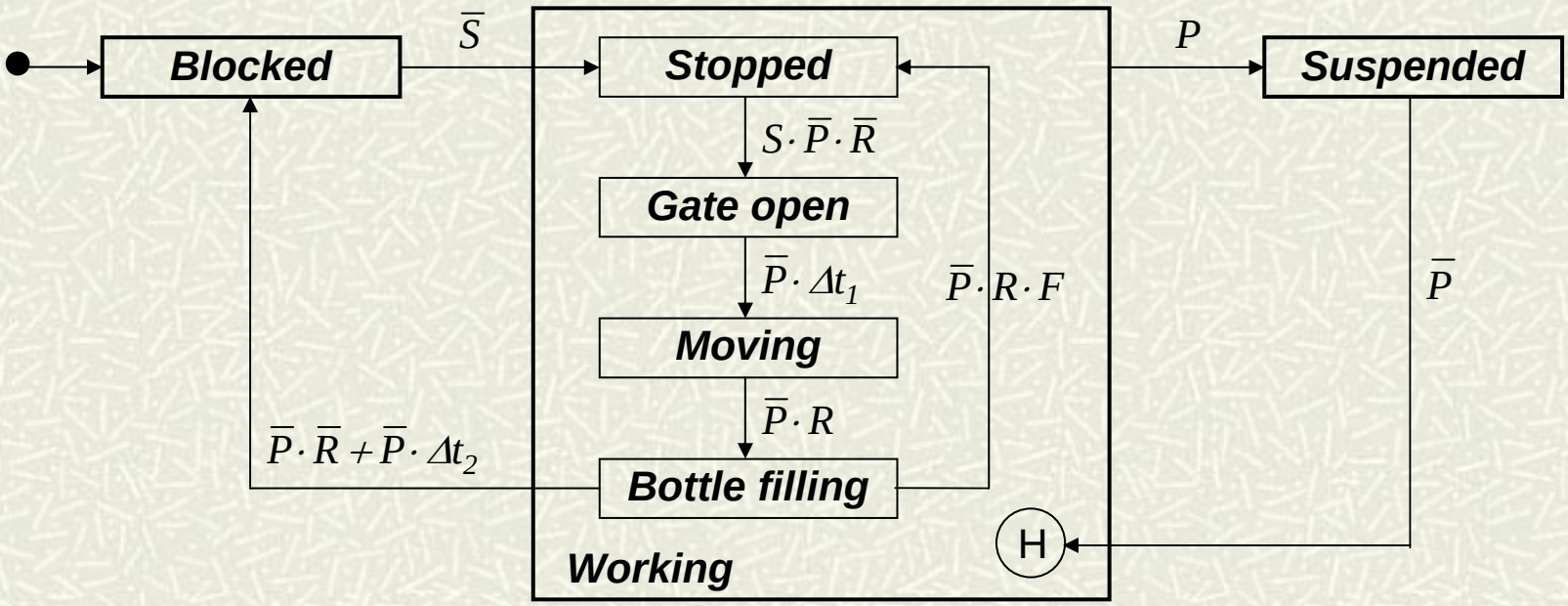
$\Theta(s_k, \tilde{t}_k) = \{ t^i \in \Gamma : s_k \in \tau_S(t^i) \ \& \ \tilde{t}^i_k \geq \tau_R(t^i) \}$ expired timers

$s_{k+1} = \delta(s_k, a_k, \Theta(s_k, t_k))$ the next state

$$t^i_{k+1} = \begin{cases} \tilde{t}^i_k + \varepsilon & \text{if } s_{k+1} \in \tau_S(t^i) \ \& \ s_k \in \tau_S(t^i) \\ 0 & \text{otherwise} \end{cases}$$

the next timer value

Example (5): Transition function



$\delta(\text{Blocked}, \overline{S}, \phi) = \text{Stopped}$
$\delta(\text{Stopped}, S \cdot \overline{P} \cdot \overline{R}, \phi) = \text{Gate open}$
$\delta(\text{Stopped}, P, \phi) = \text{Suspended-Stopped}$
$\delta(\text{Gate open}, \overline{P}, \phi) = \text{Suspended-Gate open}$
$\delta(\text{Gate open}, P, \{t^1\}) = \text{Moving}$

$\delta(\text{Suspended-Stopped}, \overline{P}, \phi) = \text{Stopped}$
$\delta(\text{Suspended-Gate open}, \overline{P}, \phi) = \text{Gate open}$

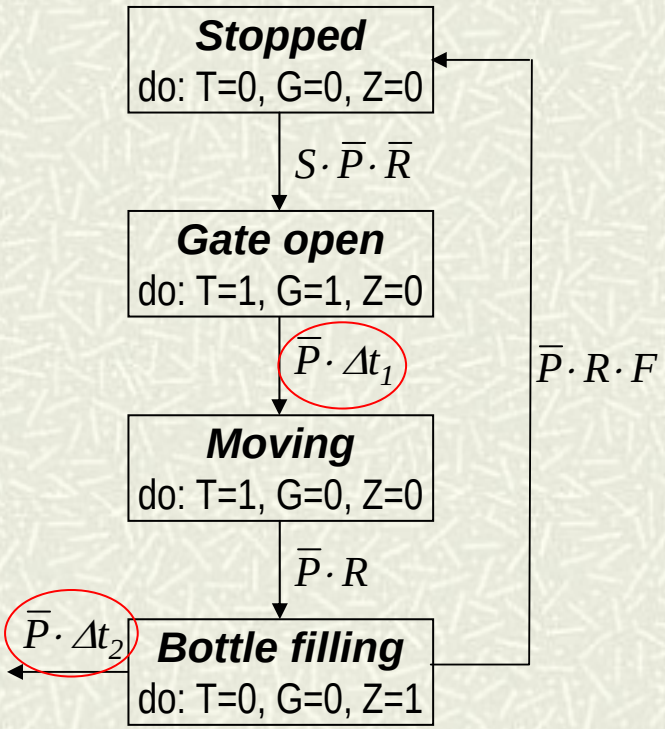
Program generation: Reference model

```
state  = initial_state();
loop_forever {
  input  = poll_the_input();
  timers = set_timers(state,active_timers());
  state  = next_state(state,timers,input);
  output = count_output(state);
  set_the_output(output);
}
```

$$FSTM = (S, \Sigma, \Gamma, \tau, \delta, s_0, \varepsilon, \Omega, \omega),$$

flip-flops , input symbols , *timer devices* , output symbols

Program generation: Timer symbols

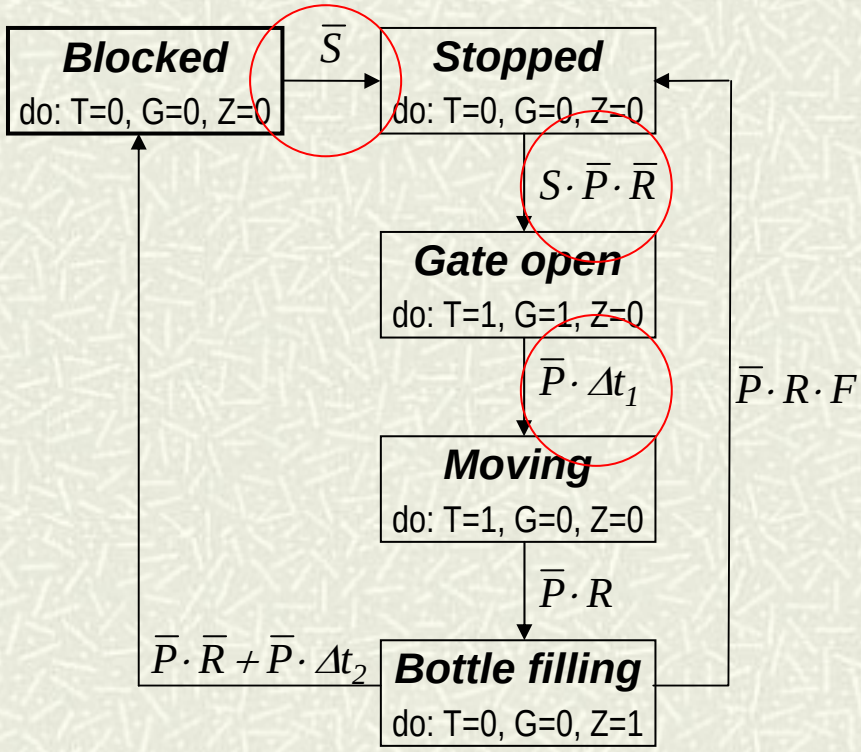


M1	*M2*	*M3*	*M4*	Line states
0	0			*Blocked*
1	0	0	0	*Stopped*
1	0	0	1	*Gate open*
1	0	1	1	*Moving*
1	0	1	0	*Bottle filling*
1	1	*	*	*Suspended*

Set $t1 = M1 \cdot \overline{M2} \cdot \overline{M3} \cdot M4$

Set $t2 = M1 \cdot \overline{M2} \cdot M3 \cdot \overline{M4}$

Program generation: Transition function



M1	*M2*	*M3*	*M4*	Line states
0	0			*Blocked*
1	0	0	0	*Stopped*
1	0	0	1	*Gate open*
1	0	1	1	*Moving*
1	0	1	0	*Bottle filling*
1	1	*	*	*Suspended*

Set $M4 = S \cdot \overline{P} \cdot \overline{R} \cdot M1 \cdot \overline{M2} \cdot \overline{M3} \cdot \overline{M4}$

Set $M3 = \overline{P} \cdot t1 \cdot M1 \cdot \overline{M2} \cdot \overline{M3} \cdot M4$

Set $M1 = \overline{S} \cdot \overline{M1} \cdot \overline{M2}$

Res $M3 = \overline{S} \cdot \overline{M1} \cdot \overline{M2}$

Res $M4 = \overline{S} \cdot \overline{M1} \cdot \overline{M2}$

Program generation: Ladder diagram

(a1) Set $t1 = M1 \cdot \overline{M2} \cdot \overline{M3} \cdot M4$

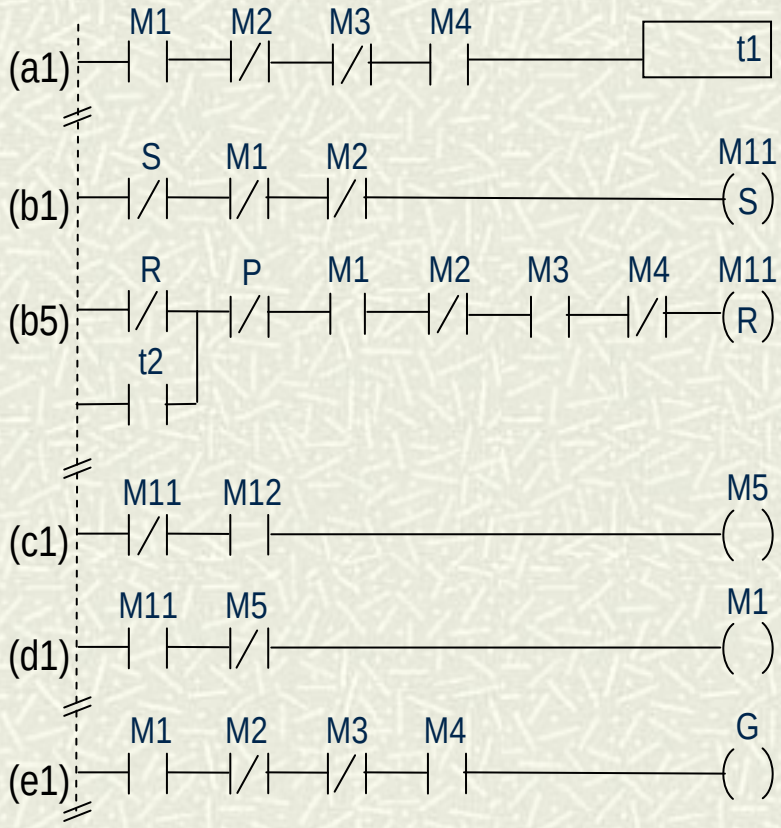
(b1) Set $M11 = \overline{S} \cdot \overline{M1} \cdot \overline{M2}$

(b5) Res $M11 = (\overline{R} + t2) \cdot \overline{P} \cdot M1 \cdot \overline{M2} \cdot M3 \cdot \overline{M4}$

(c1) $M5 = \overline{M11} \cdot M12$

(d1) $M1 = M11 \cdot \overline{M5}$

(e1) $G = M1 \cdot \overline{M2} \cdot \overline{M3} \cdot M4$



Conclusions

- Finite state time machine defines formal semantics of:
  - time extensions to UML statecharts
  - hierarchy of states
- The model enables automatic program generation for PLC controller
- The method has been tested manually
  - automatic compiler is tested
- Future plans:
  - formal verification of the model (UPPAAL)