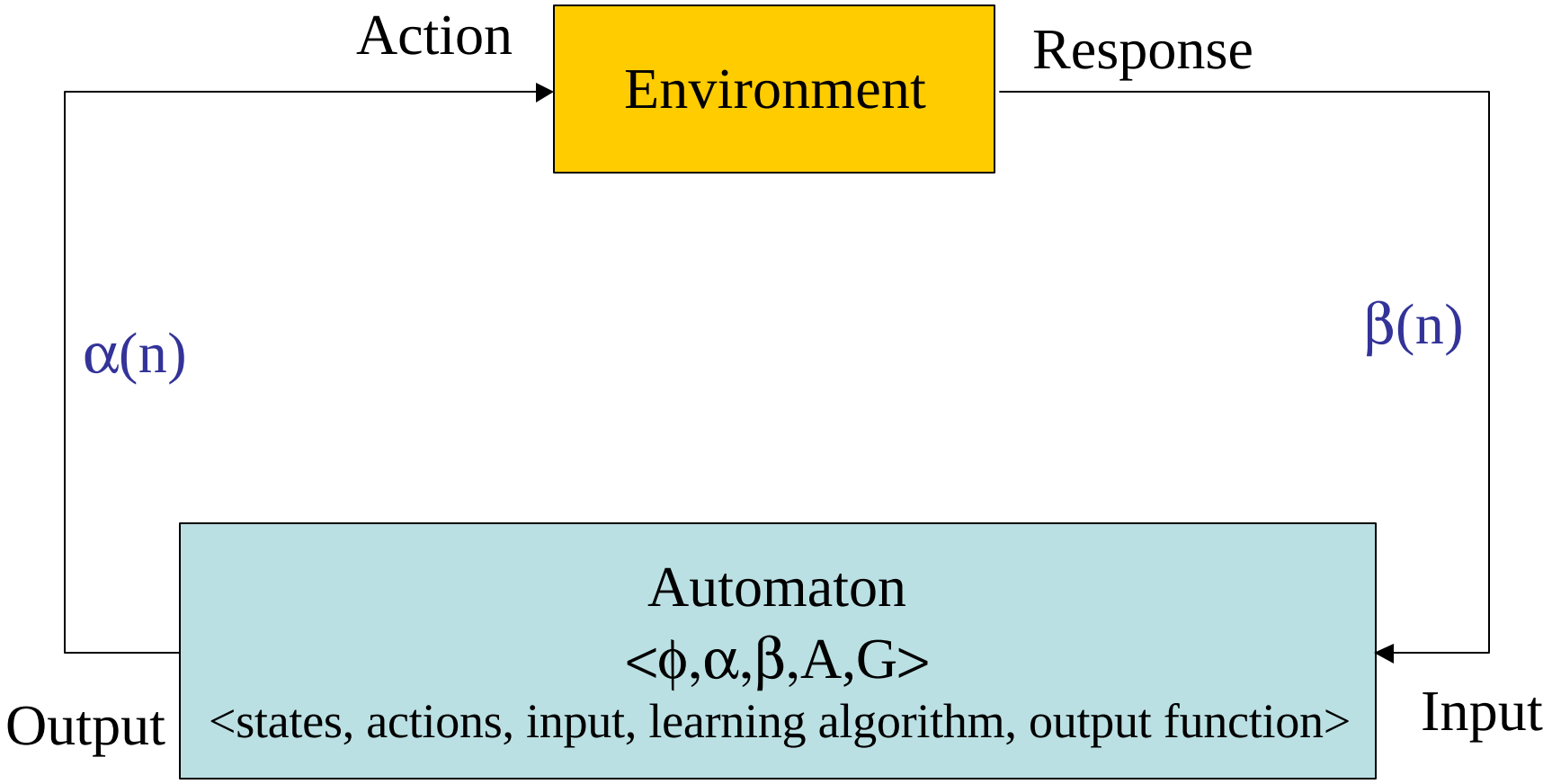
Learning Automata

- Learns the unknown nature of an environment
- Variable structure stochastic learning automaton is a quintuple $\{\varphi,\alpha,\beta,A,G\}$ where:
  - $\varphi(n)$, state of automaton; $\varphi=\{\varphi_1,\ldots,\varphi_s\}$
  - $\alpha(n)$, output of automaton; $\alpha=\{\alpha_1,\ldots,\alpha_r\}$
  - $\beta(n)$, input to automaton; $\beta=\{\beta_1,\ldots,\beta_m\}$
  - A, is the learning algorithm;
  - G[.], is the output function; $\alpha(n)=G[\varphi(n)]$

  n indicates the iteration number.

Learning Automaton Schematic



Probability Vector

- $p_j(n)$, action probability; the probability that automaton is in state j at iteration n.
- Reinforcement scheme

  If $\alpha(n) = \alpha_i$ and for j <> i; (j=1 to r)

  $p_j(n+1) = p_j(n) - g[p_j(n)]$ when $\beta(n) = 0$.

  $p_j(n+1) = p_j(n) + h[p_j(n)]$ when $\beta(n) = 1$.
- In order to preserve probability measure,

  $\Sigma\ p_j(n) = 1$, for j = 1 to r.

contd ...

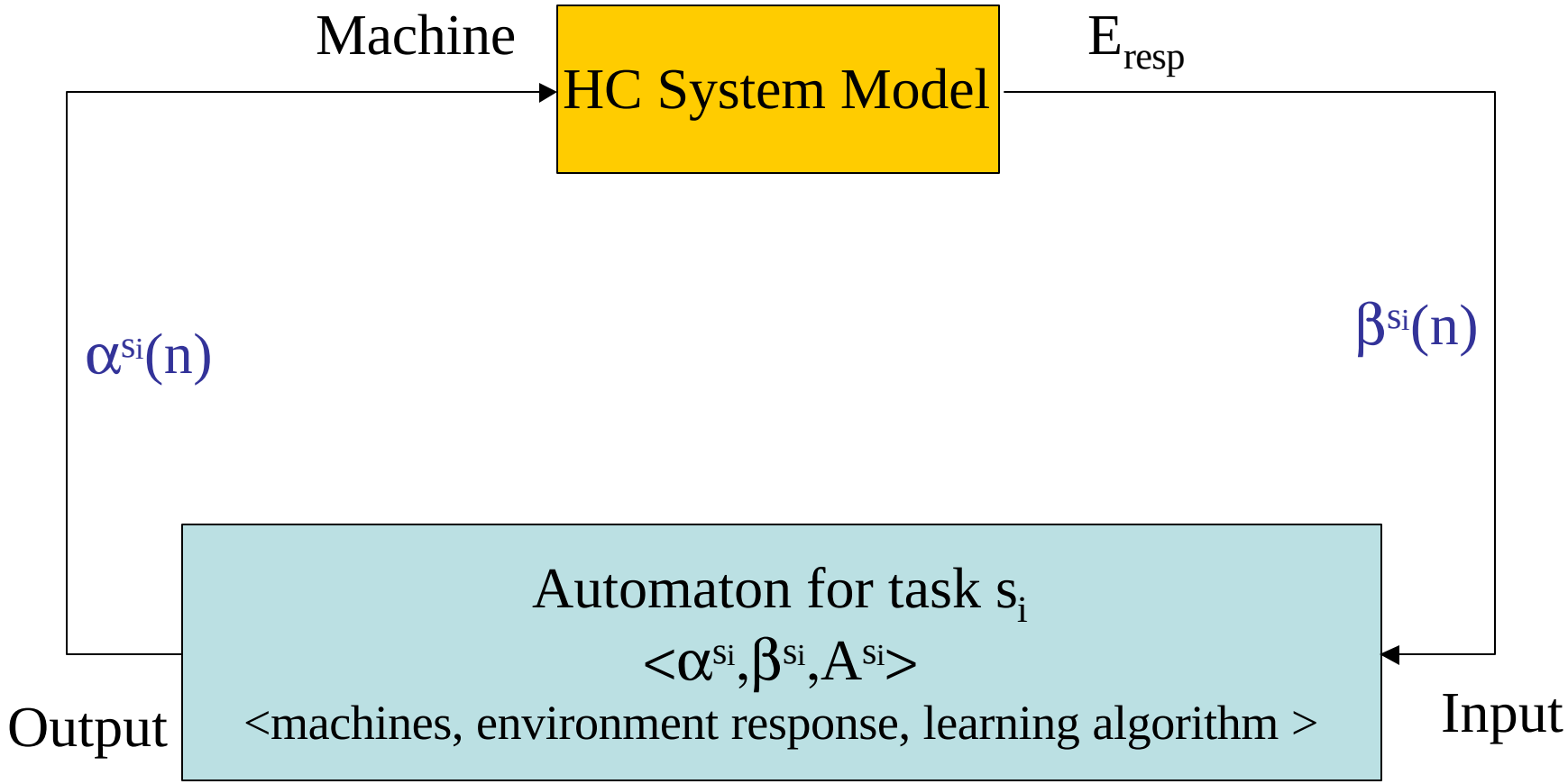
- If $\alpha(n) = \alpha_i$

$$p_i(n+1) = p_i(n) + \sum_{j=1,\, j<>i}^{r} g(p_j(n)) \quad \text{when } \beta(n) = 0$$

$$p_i(n+1) = p_i(n) - \sum_{j=1,\, j<>i}^{r} h(p_j(n)) \quad \text{when } \beta(n) = 1$$

- *g(.)* is the reward function
- *h(.)* is the penalty function

Schematic of Proposed Automata Model for Mapping/Scheduling



Model Construction

- Every task s_i associated with an S-model automaton (VSSA).
- VSSA represented as $\{\alpha^{si}, \beta^{si}, A^{si}\}$, since r = s
  - α^{si} is set of actions $\alpha^{si} = m_0, m_1, \ldots, m_{|M|-1}$
  - β^{si} is input to the automaton, $\beta^{si} \in [0, 1]$

    closer to 0 – action favorable to system;

    closer to 1 – action unfavorable to system
  - A^{si} is reinforcement scheme
- $p_{ij}(n)$ - action probability vector
  - probability of assigning task s_i to machine m_j

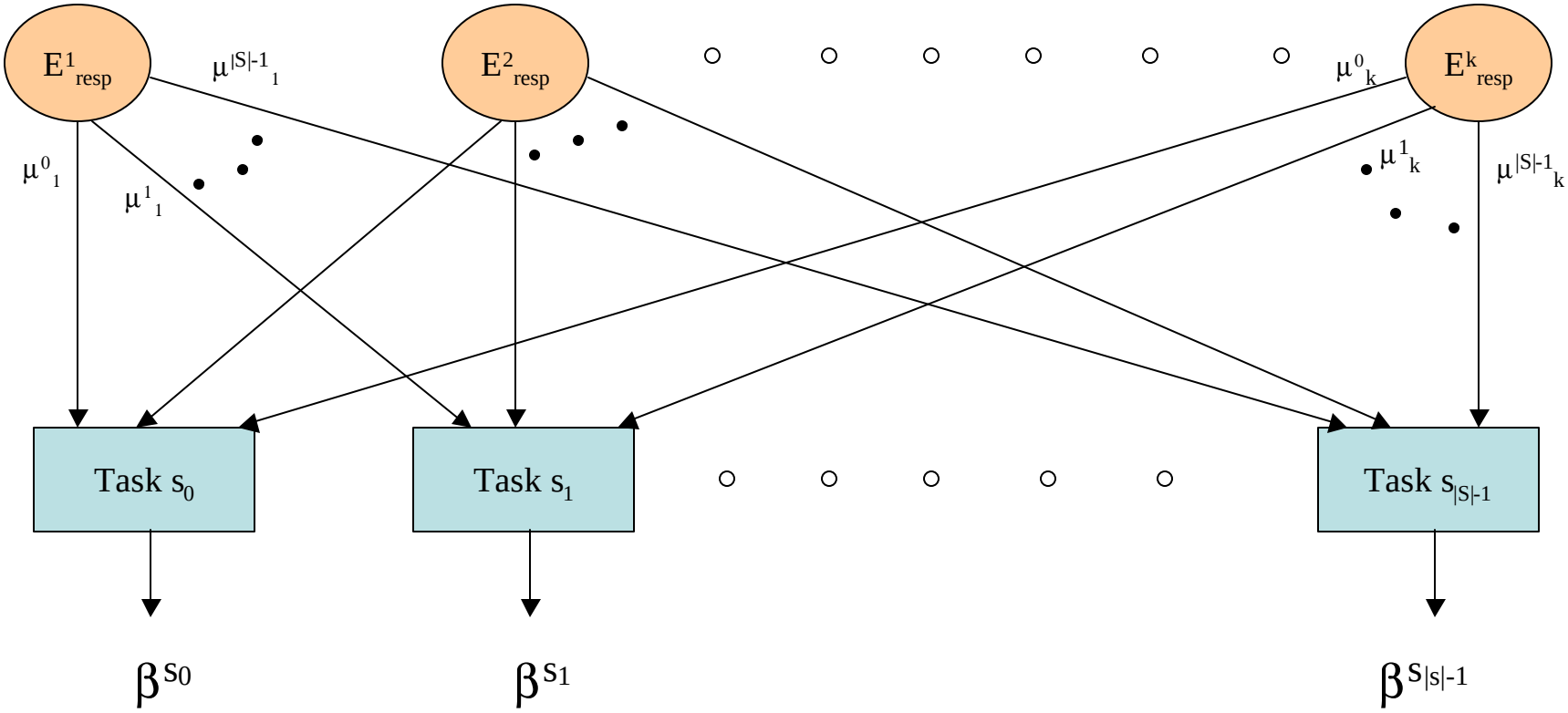
contd …

- Automata model for Mapping/Scheduling
  - S-model VSSA is used
  - Each automaton is represented as a tuple $\{\alpha^{si}, \beta^{si}, A^{si}\}$
  - $\alpha^{si} = m_0, m_1, \ldots, m_{|M|-1}$
  - $\beta^{si} \in [0, 1]$

    (closer to 0 - favorable, 1 - unfavorable)
  - If $c_k(n)$ is better than $c_k(n-1)$

    $E^k_{resp} = 0$ else $E^k_{resp} = 1$
  - Translating E^k_{resp} to $\beta^{si}(n)$ requires two steps

Translating E^k_{resp} to β^{s_i}



contd …

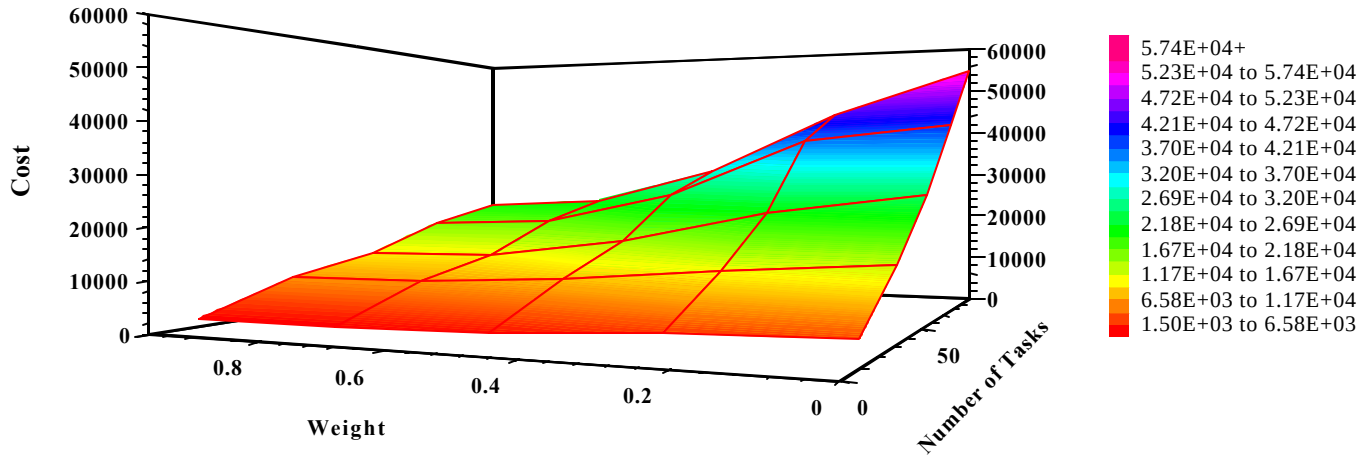
- Step 1: Translate E^k_{resp} to $\mu^{si}_k(n)$, where
  - $\mu^{si}_k(n)$ - input to automaton s_i with respect to cost metric c_k
  - achieved by the heuristics

- Step 2: Achieved be means of Lagrange's multiplier

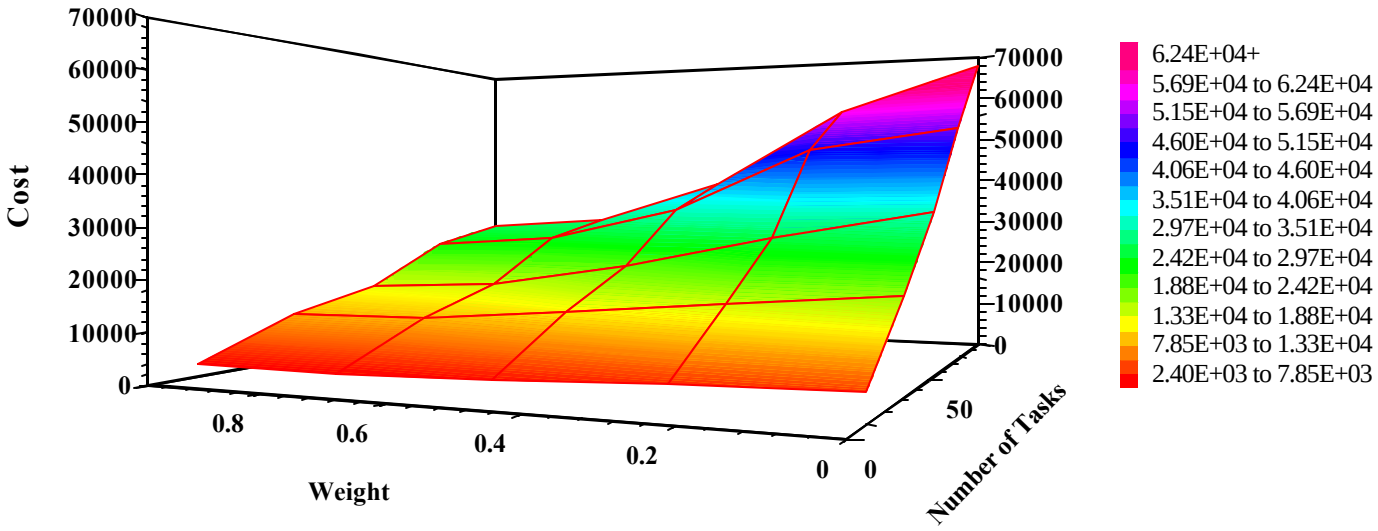
$$\beta^{si}(n) = \sum_{j=1}^{|C|} \lambda_k * \mu^{si}_k(n), \; i=1 \text{ to } |S|-1; \quad \sum_{j=1}^{|C|} \lambda_k = 1, \; \lambda_k > 0$$

where λ_k is the weight of metric c_k

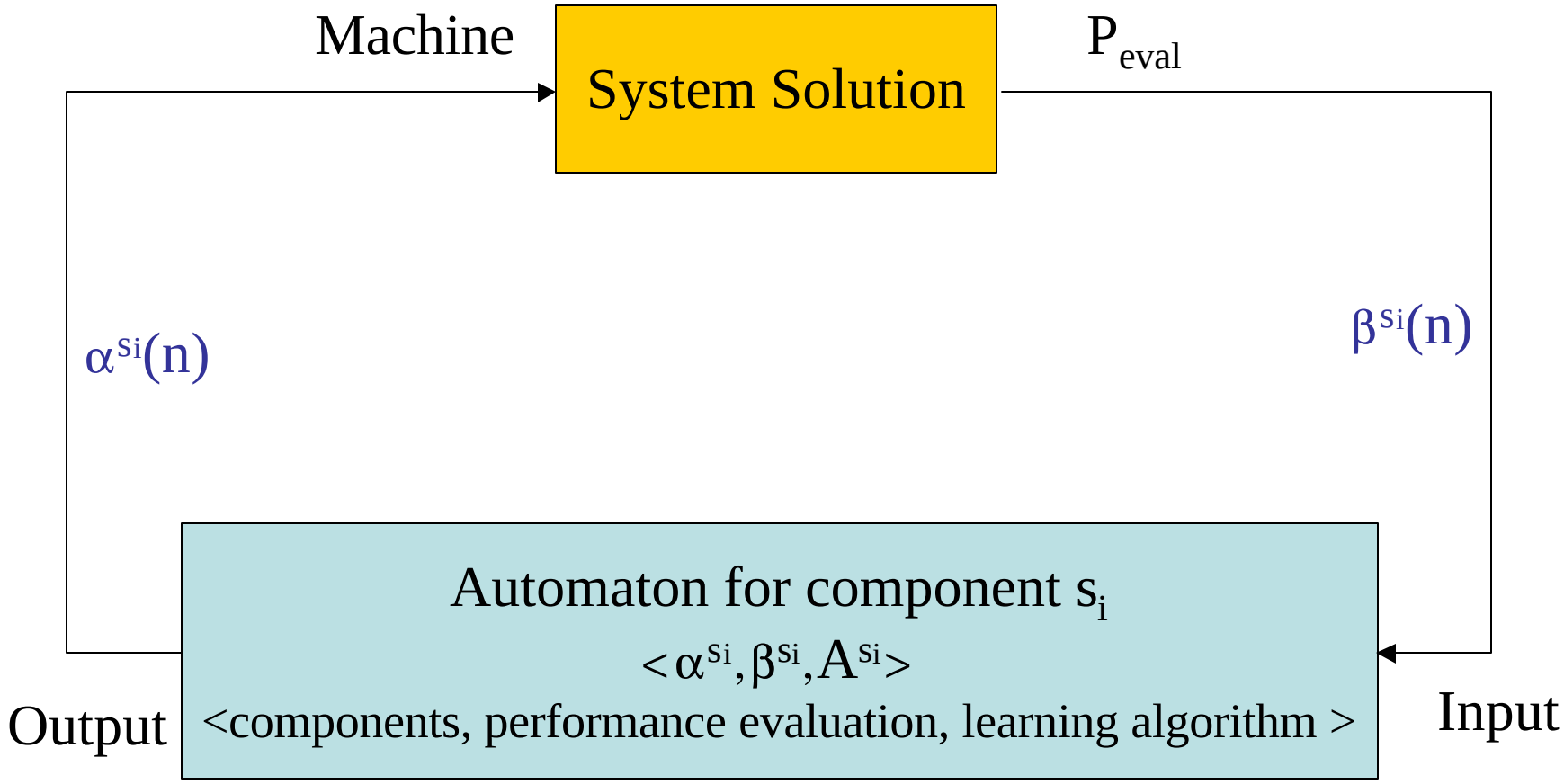
Low Communication Complexity, Machines = 5



Medium Communication Complexity, Machines = 5



Schematic of Proposed Automata Model for Architecture Trades



Model Construction

- Every component of the HW system s_i associated with a P-model automaton (VSSA).
- VSSA represented as $\{\alpha^{si}, \beta^{si}, A^{si}\}$, since r = s
  - α^{si} is set of component types $\alpha^{si} = c_0, c_1, \ldots, c_{|M|-1}$
  - β^{si} is input to the automaton, $\beta^{si} = 0, 1$

    0 – performance favorable to system; 1 – unfavorable to system
  - A^{si} is reinforcement scheme
- $p_{ij}(n)$ - action probability vector
  - probability of choosing component s_i from component type c_j

contd …

- Automata model for Architecture Trades
  - P-model VSSA is used
  - Each automaton is represented as a tuple $\{\alpha^{si}, \beta^{si}, A^{si}\}$
  - $\alpha^{si} = c_0, c_1, \ldots, c_{|M|-1}$
  - $\beta^{si} \in 0, 1$

    (0 - favorable, 1 - unfavorable)
  - If $c_k(n)$ is better than $c_k(n-1)$

    $P_{eval} = 0$ else $P_{eval} = 1$

Conclusions

- Adaptive Framework for Mapping and Architecture trades
- Automata models allow optimization of multiple criteria
- Efficient / gracefully degradable solutions
- Framework construction suitable for tool integration
  - Mapping algorithm integrated with **SAGE™**
- Provides a basis for systems design from application to the embedded HW