TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Institut für Computertechnik
Institute of Computer Technology

Towards a Formal Model of Recursive Self-Reflection

Axel Jantsch

TU Wien, Vienna, Austria

SAINT Workshop on
Self-Aware and Autonomous Systems

Shenzhen, China, May 2019

Self-Awareness

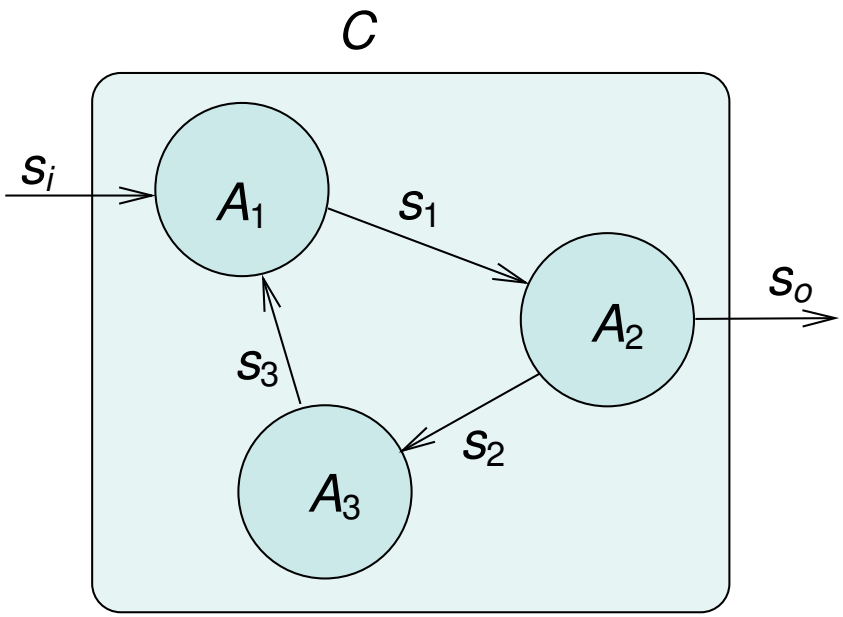
Self-awareness, in this context, is defined by the combination of three properties that IT systems and services should possess:

1. **Self-reflective**: *i) aware of their software architecture, execution environment and the hardware infrastructure on*
   goals in
   s
   vare of
   e
   nt
3. **Self-adaptive**: *proactively adapting as the environment evolves in order to ensure that their QoS requirements and respective SLAs are continuously satisfied while at the same time operating costs and energy-efficiency are optimized.*

- Recursive use of the term “awareness”;
- Static, not dynamic;
- The system cannot be aware of its own self-reflection.

Samuel Kounev et al. “Model-driven Algorithms and Architectures for Self-Aware Computing Systems (Dagstuhl Seminar 15041)”. In: **Dagstuhl Reports** 5.1 (2015). Ed. by Samuel Kounev et al., pp. 164–196

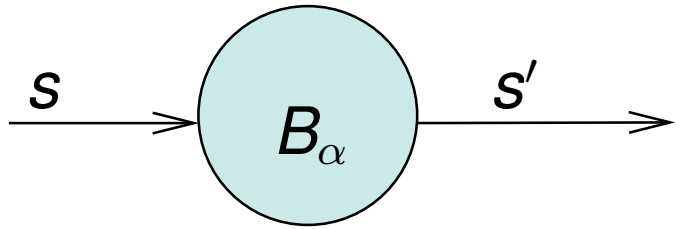
Actors in a Dynamic Dataflow Model



$$A = \langle \mathfrak{T}, I, O, z_0, f, g, \nu, \vec{\mathfrak{m}} \rangle$$

$\mathfrak{T} \subseteq \mathfrak{S}$	... set of states
$I \subseteq \mathcal{P}(S)$	... input signals
$O \subseteq \mathcal{P}(S)$	... output signals
$z_0 \in \mathfrak{T}$	... initial state
$\nu : \mathbb{N} \to \mathcal{P}(\mathbb{N})$	... input partitioning
$f : \mathcal{P}(S) \times \mathfrak{S} \to \mathcal{P}(S)$	... output encoding
$g : \mathcal{P}(S) \times \mathfrak{S} \to \mathfrak{S}$	... next state
$\vec{\mathfrak{m}} : \mathfrak{S} \to \texttt{Action}$	... a meta operator

Example Actor



$$B_\alpha = \langle \{\}, \{s\}, \{s'\}, \cdot, \alpha, \cdot, \nu, \cdot \rangle$$

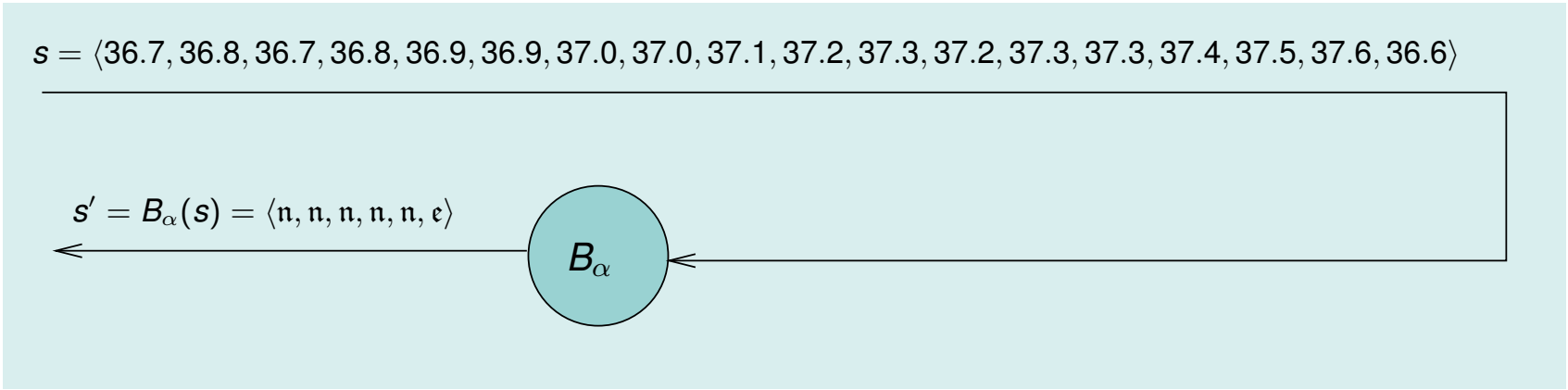
$$\nu(.) = \{3\}$$

$$\alpha(\langle t_1, t_2, t_3 \rangle) = \begin{cases} \mathfrak{l} & \text{if } (t_1 + t_2 + t_3)/3 < 35.5 \\ \mathfrak{n} & \text{if } 35.5 \leq (t_1 + t_2 + t_3)/3 < 37.5 \\ \mathfrak{e} & \text{if } 37.5 \leq (t_1 + t_2 + t_3)/3 < 38.5 \\ \mathfrak{h} & \text{if } 38.5 \leq (t_1 + t_2 + t_3)/3 \end{cases}$$

Abstraction

- Using symbols for many detailed data
- Reduces amount of information
- Keeps the essence
- Key for self-awareness
- General type of learning

Signal Abstraction



$$\alpha(\langle t_1, t_2, t_3 \rangle) = \begin{cases} \mathfrak{l} & \text{if } (t_1 + t_2 + t_3)/3 < 35.5 \\ \mathfrak{n} & \text{if } 35.5 \leq (t_1 + t_2 + t_3)/3 < 37.5 \\ \mathfrak{e} & \text{if } 37.5 \leq (t_1 + t_2 + t_3)/3 < 38.5 \\ \mathfrak{h} & \text{if } 38.5 \leq (t_1 + t_2 + t_3)/3 \end{cases}$$

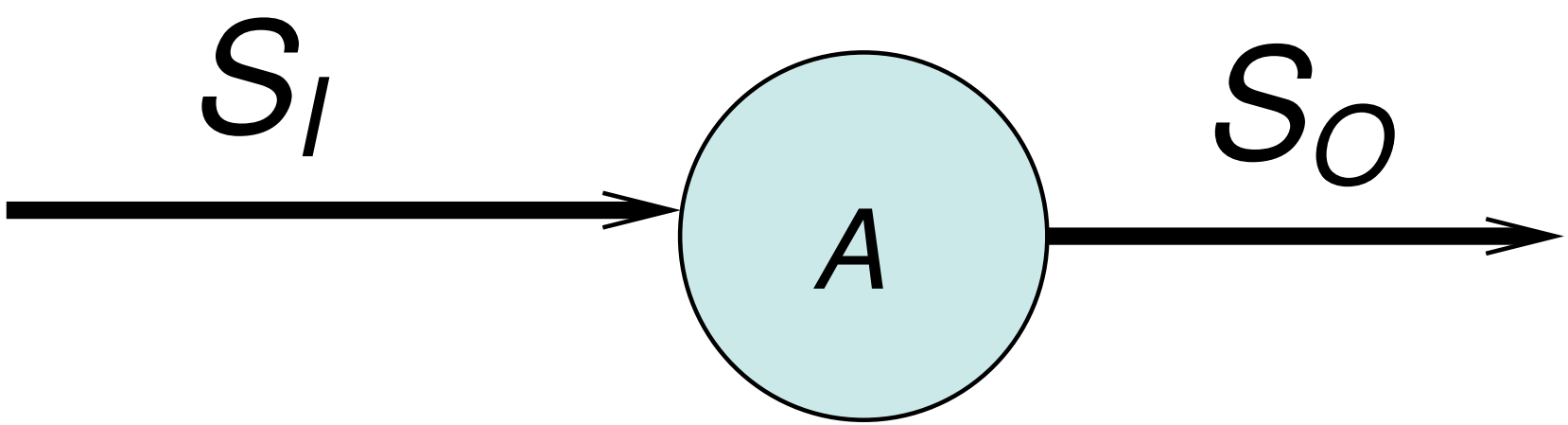
Value Abstraction

$$\alpha_V(\langle x \rangle) = \begin{cases} \mathfrak{A} & \text{if } x = \mathfrak{a} \text{ or } x = \mathfrak{b} \\ x & \text{otherwise} \end{cases}$$

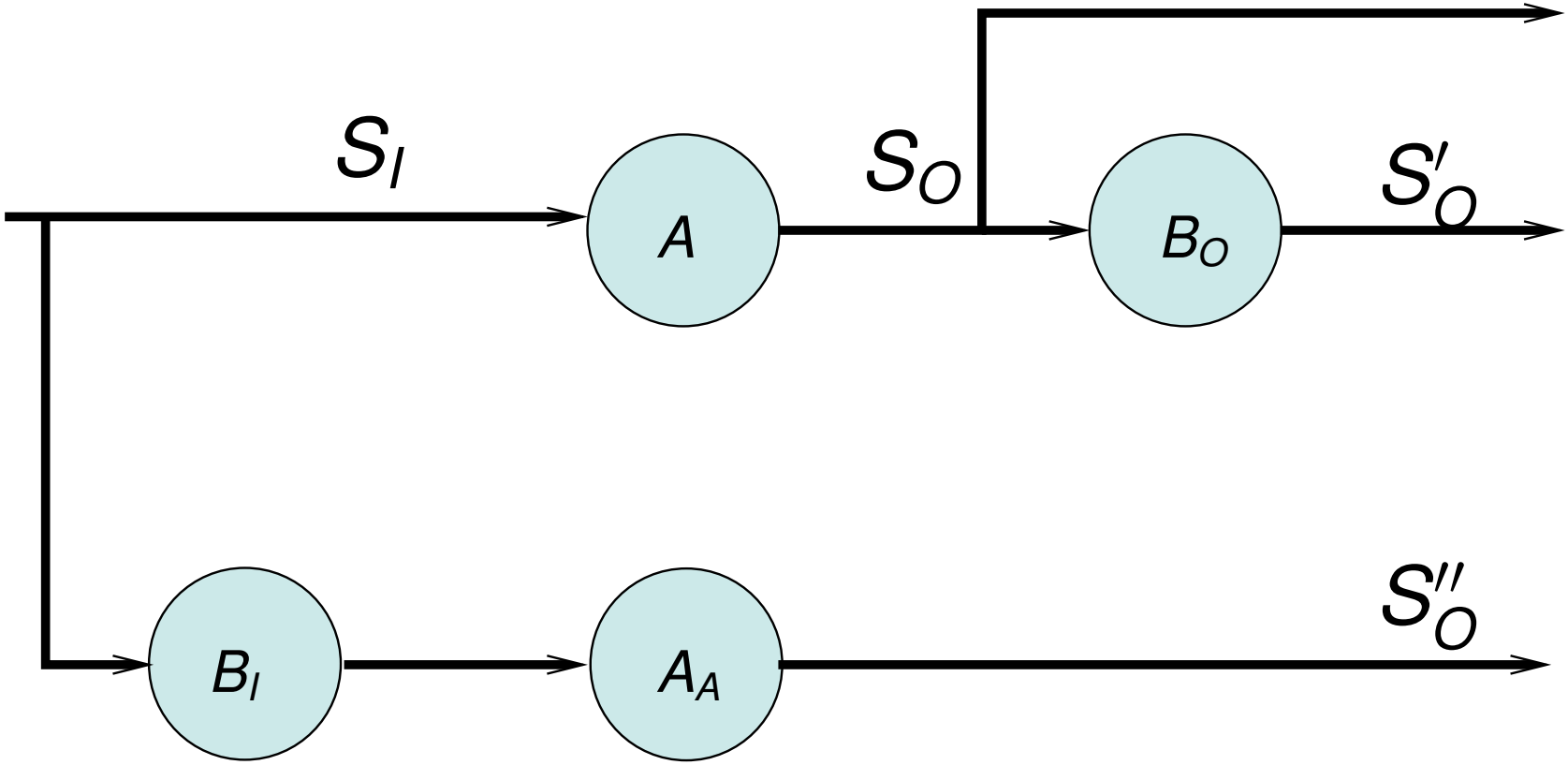
Time Abstraction

$$\alpha_t(\langle x_1, x_2 \rangle) = \begin{cases} \mathfrak{A} & \text{if } x_1 = \mathfrak{a} \text{ and } x_2 = \mathfrak{a} \\ \langle x_1, x_2 \rangle & \text{otherwise} \end{cases}$$

Actor Abstraction

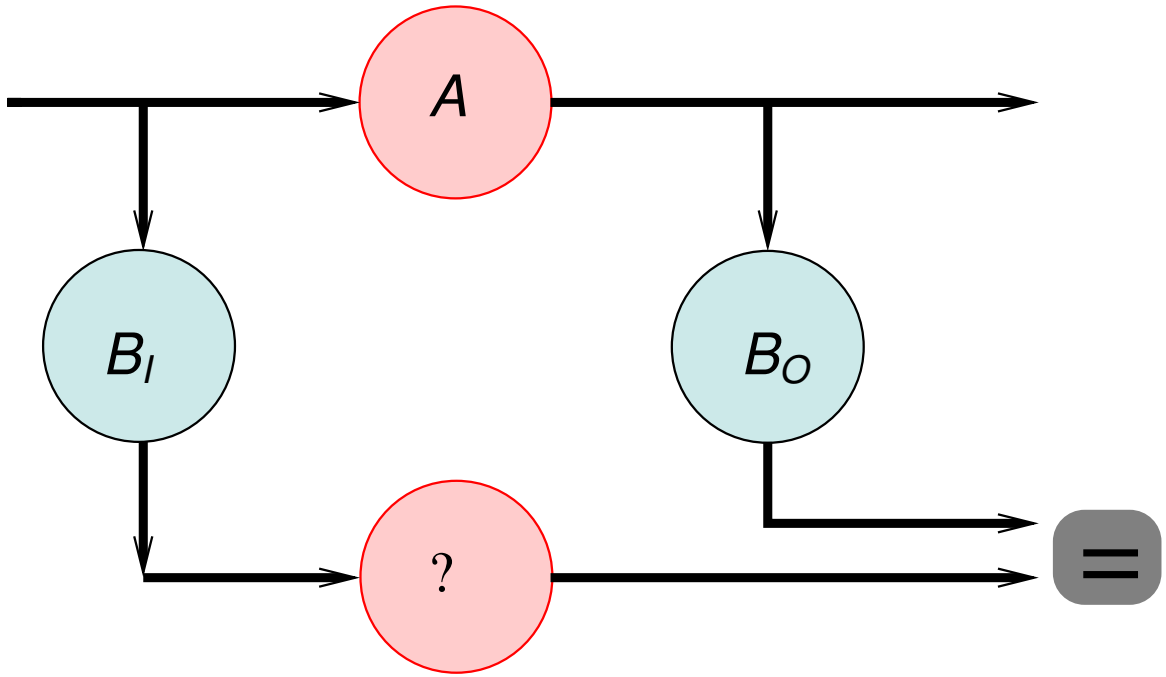


Actor Abstraction



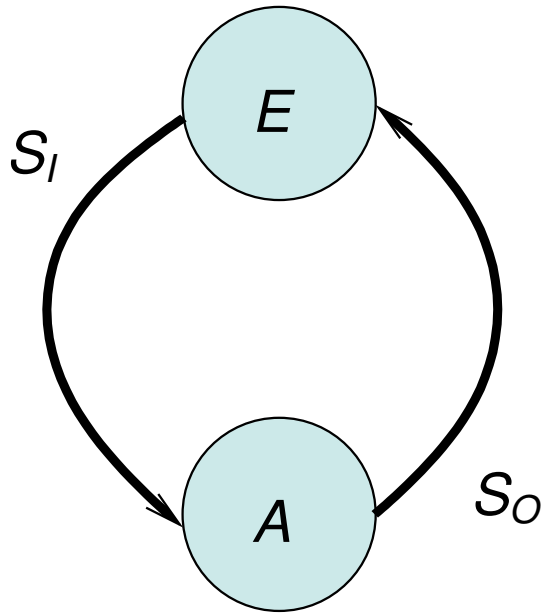
A_A is an actor abstraction of A iff $S'_O = S''_O$.

Actor Abstraction

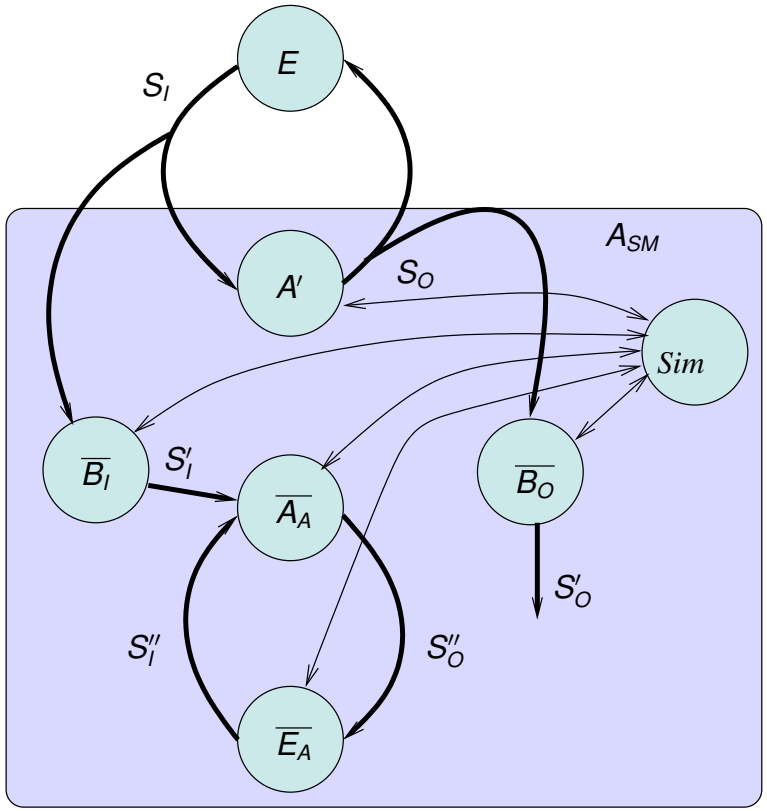


Function Approximator

Self Mdoel



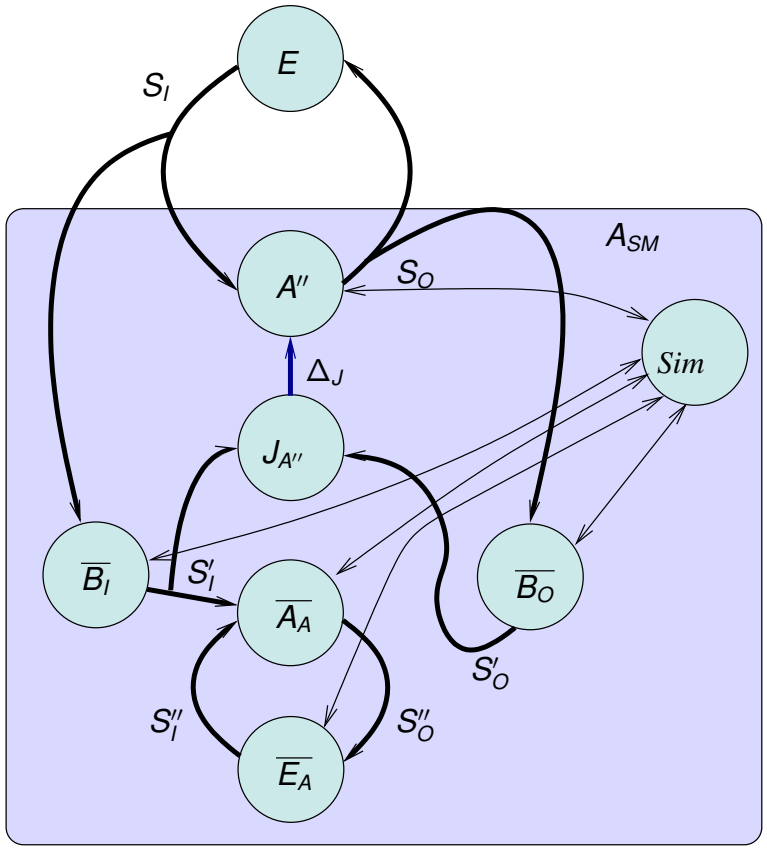
Self Model



$\bar{A}_A$ is a **simulatable actor** of A:

- $\bar{A}_A$ is an actor abstraction of A.
- It has an additional input signal denoted as **control signal**.
- It can be stopped and resumed through the control signal.
- Input signals are duplicated and controlled by the control input.
- It has an additional output signal: **status signal**.

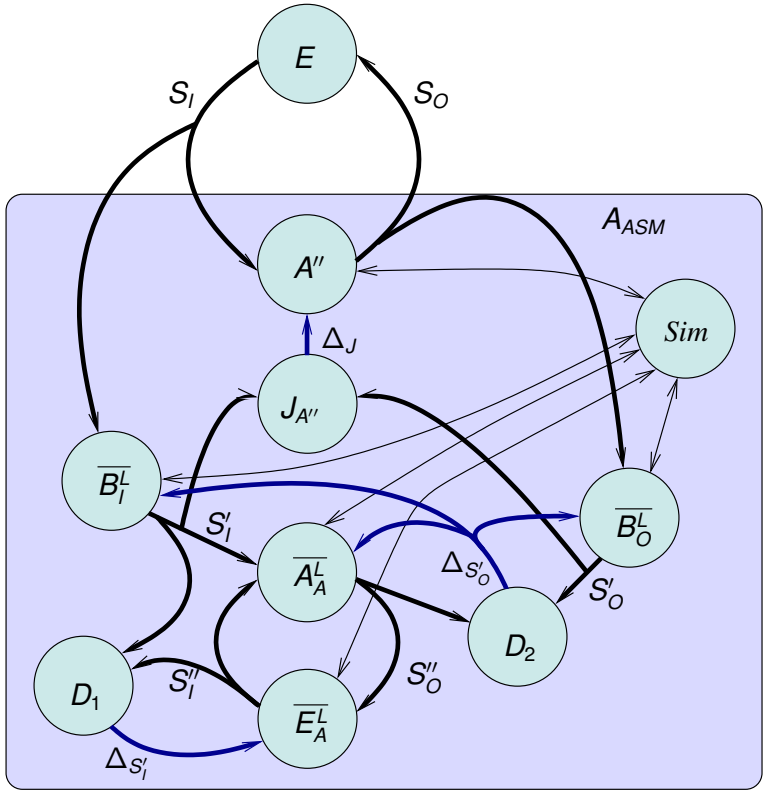
Self Assessment



$J_{A''}$ assesses the behavior and performance of actor A'':

- $J_{A''}$ monitors abstractions of inputs and outputs of A''.
- It compares the observed behavior with expected behavior.
- It reports observed differences in Δ_J.
- It maintains an assessment history.

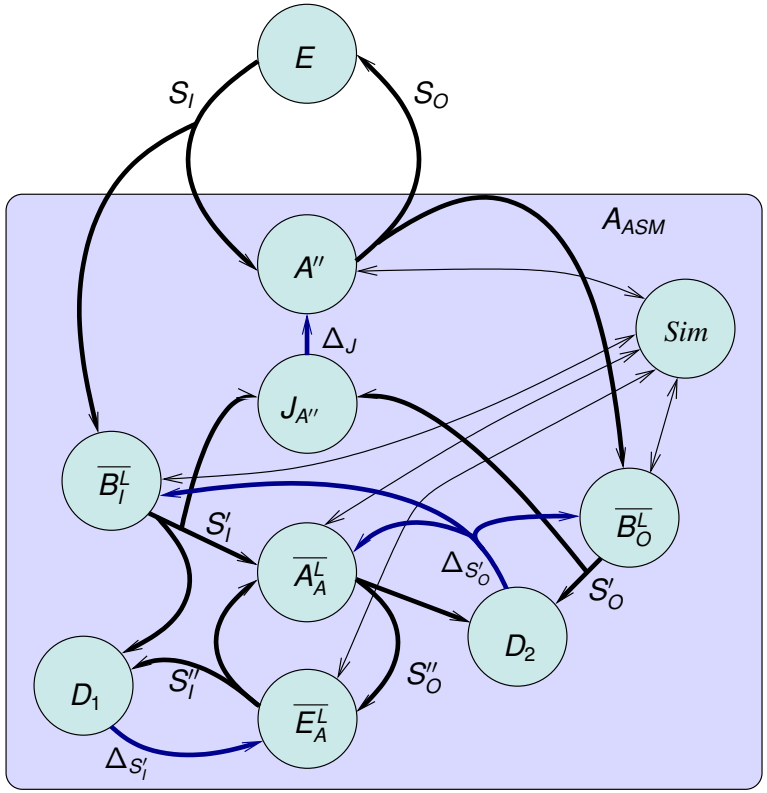
Adaptive Self-Model



A **learning actor** A^L modifies its behavior to minimize an error signal.

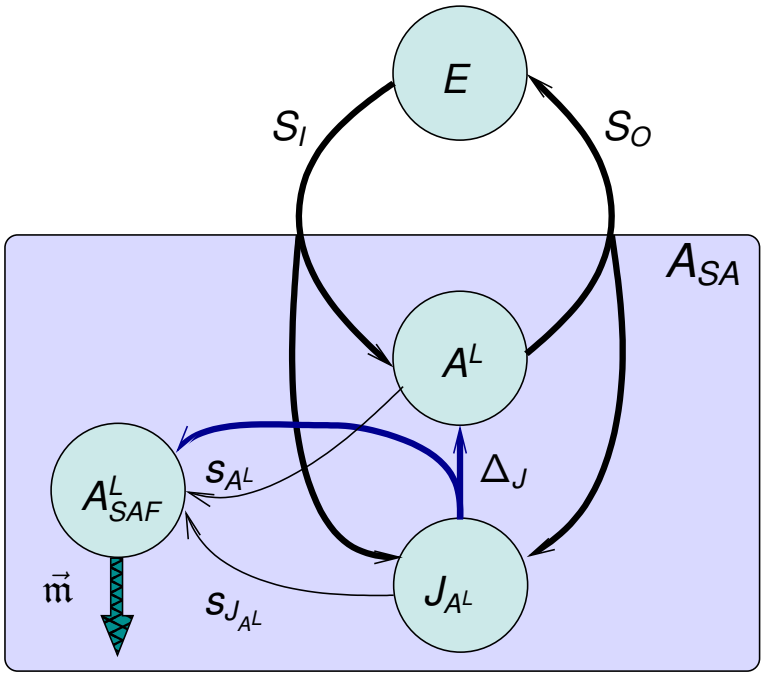
- D_1 analyses the differences between S'_I abd S''_I.
- D_2 analyses the differences between S'_O and S''_O.
- D_1 is used to improve the environment model.
- D_2 is used to improve the actor model and the signal abstractions.

Not Quite Self-Aware



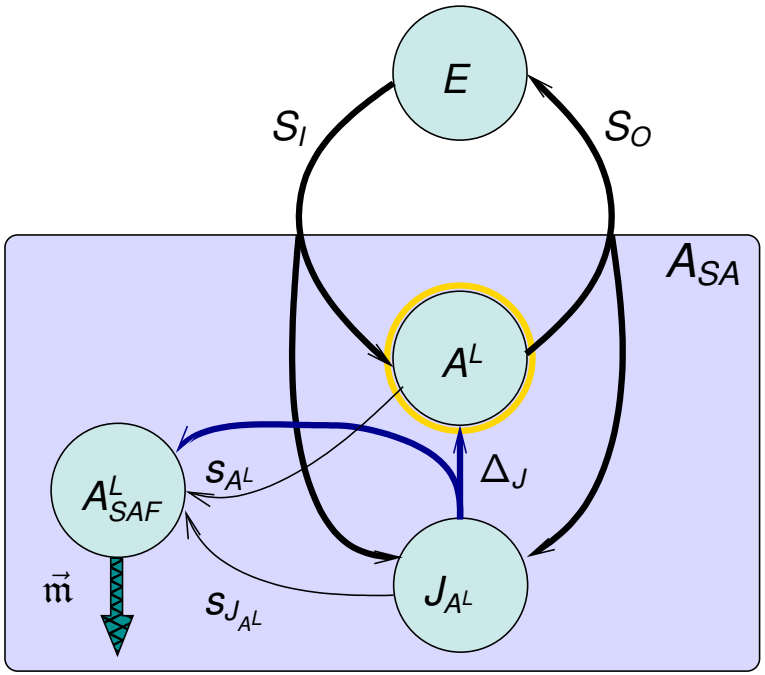
- A_{ASM} uses abstraction, simulation, learning, and a self-model.
- Self-awareness is a **process** that should be, dynamically and flexibly, applicable to a range of actors, including itself.

An Actor Facilitating Self-Awareness



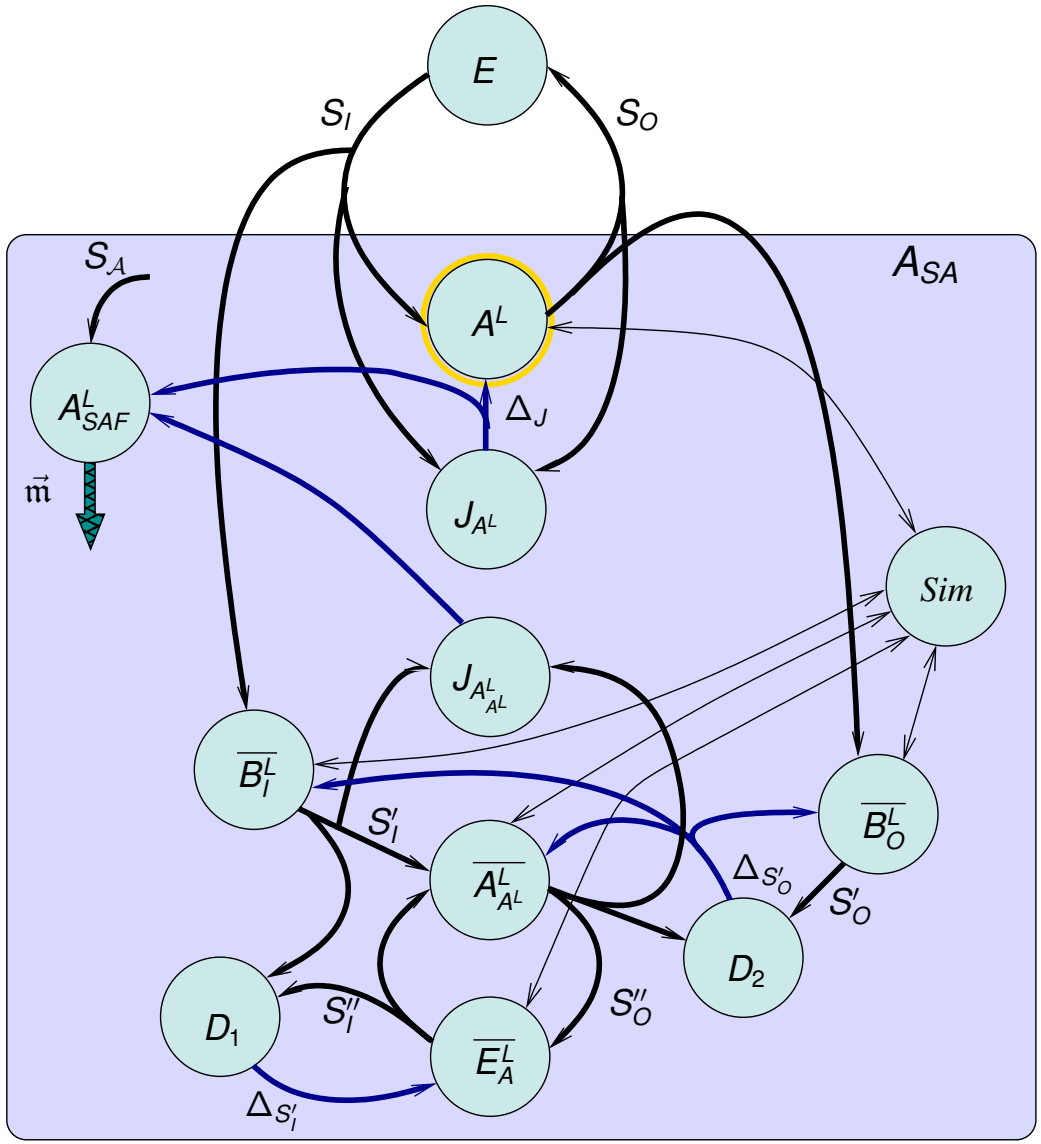
- A^L_{SAF} tracks behavior and expectations.
- It can trigger an in-depth investigation of an actor.

An Actor Facilitating Self-Awareness

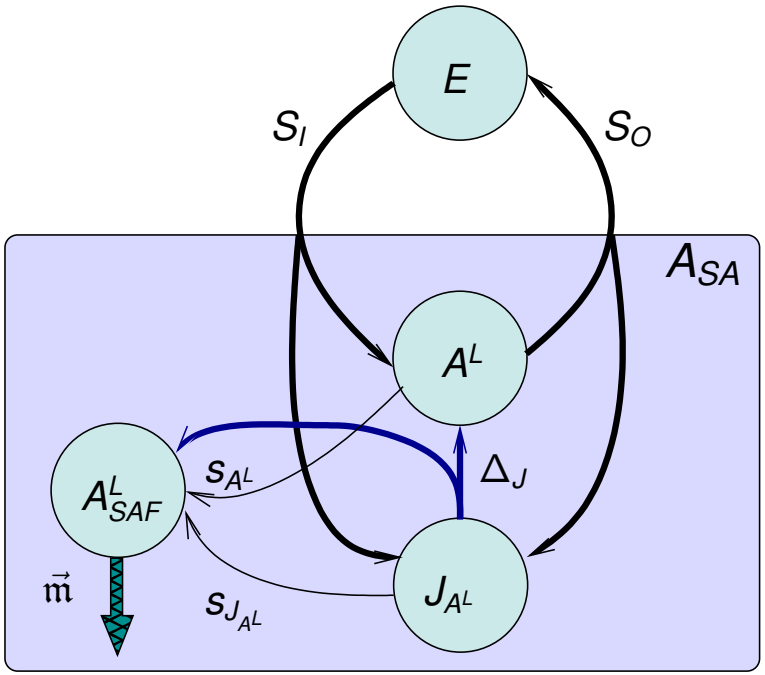


A^L_{SAF} targets A^L

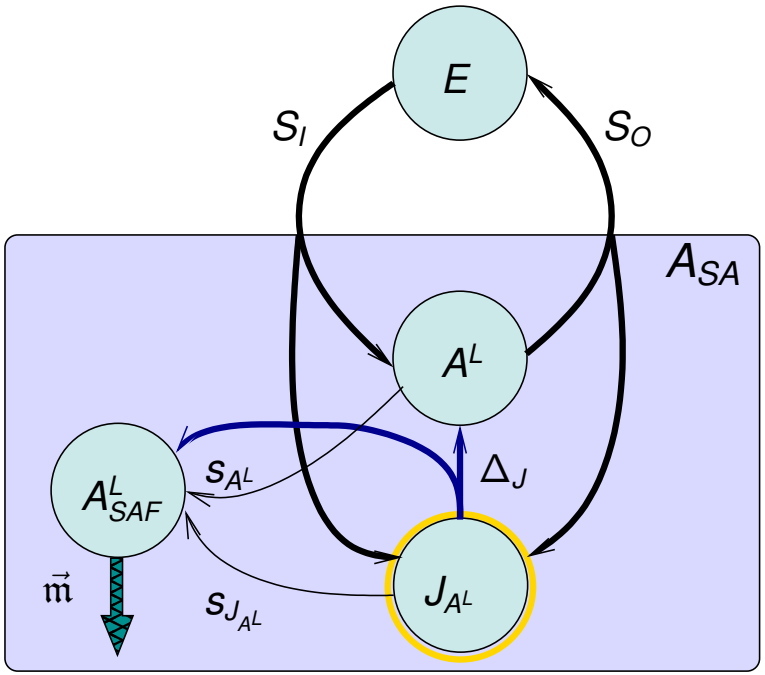
Self-Awareness Target: A^L



An Actor Facilitating Self-Awareness

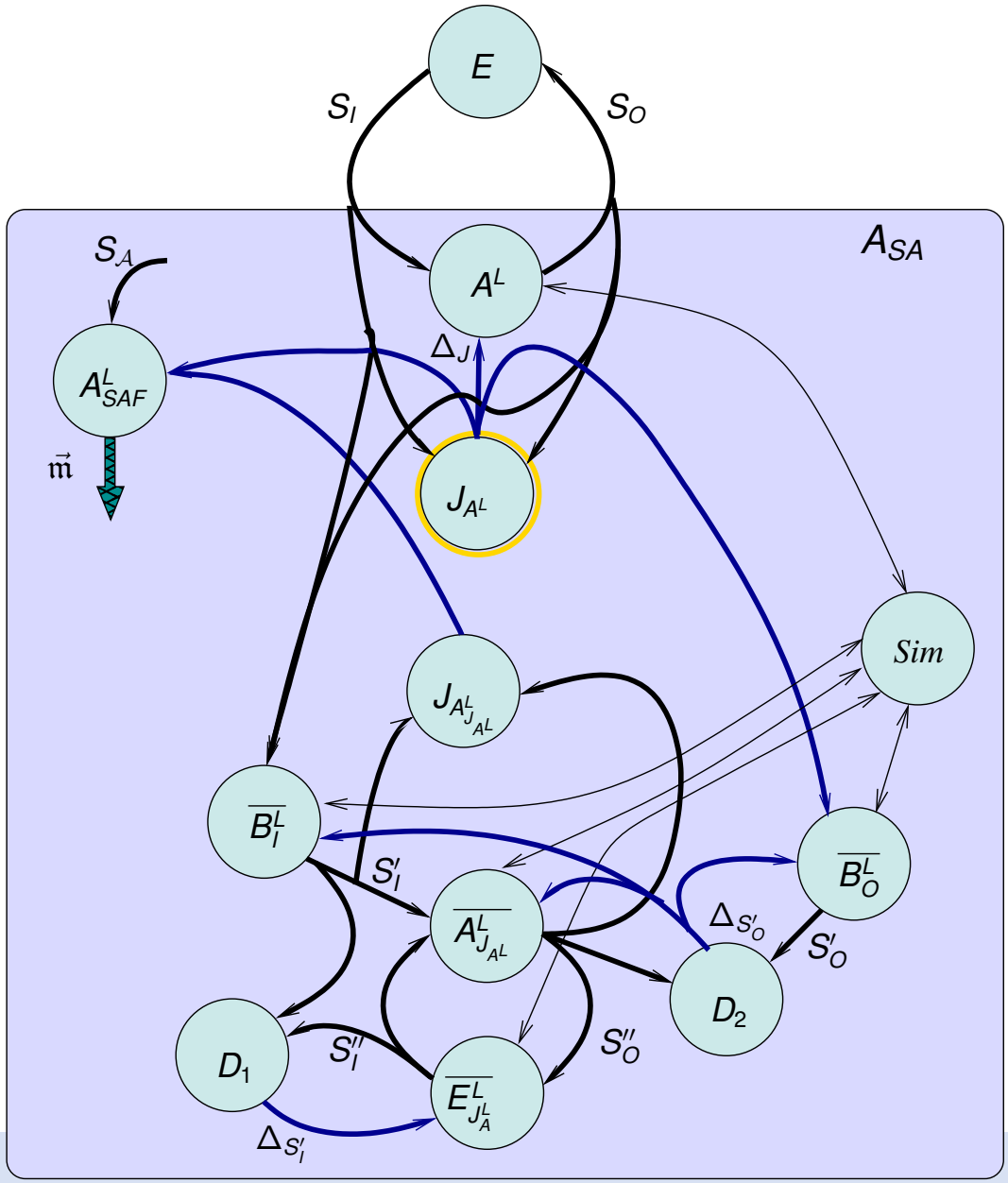


An Actor Facilitating Self-Awareness

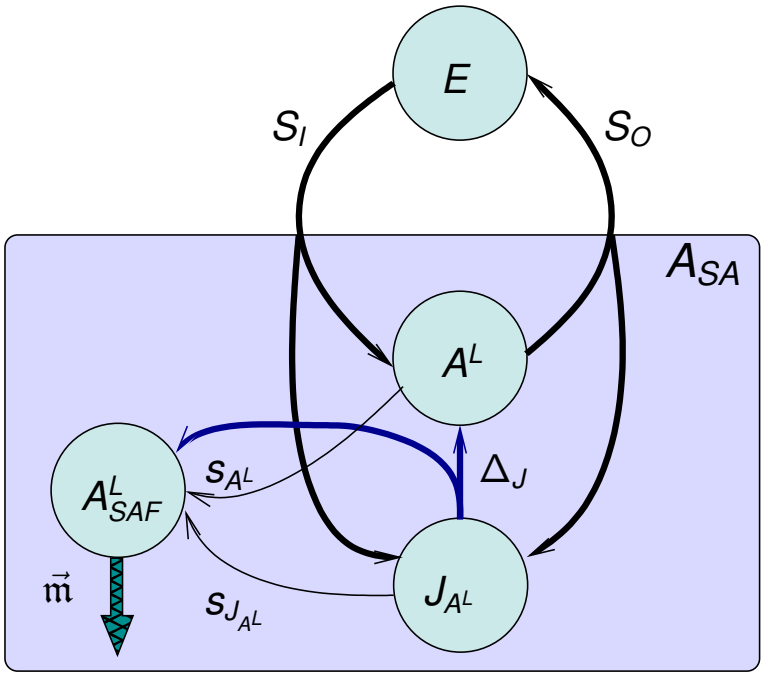


A^L_{SAF} targets J_{A^L}

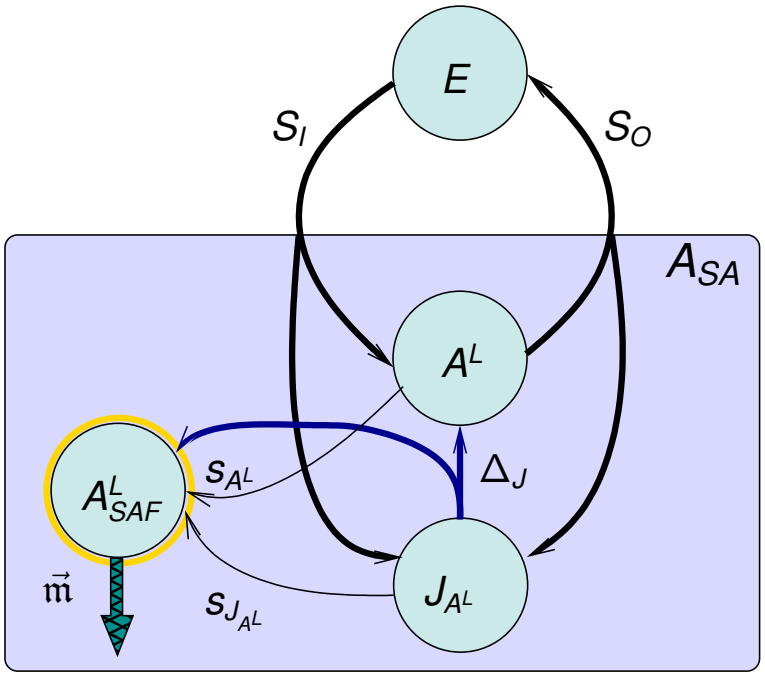
Self-Awareness Target: J_{A^L}



An Actor Facilitating Self-Awareness

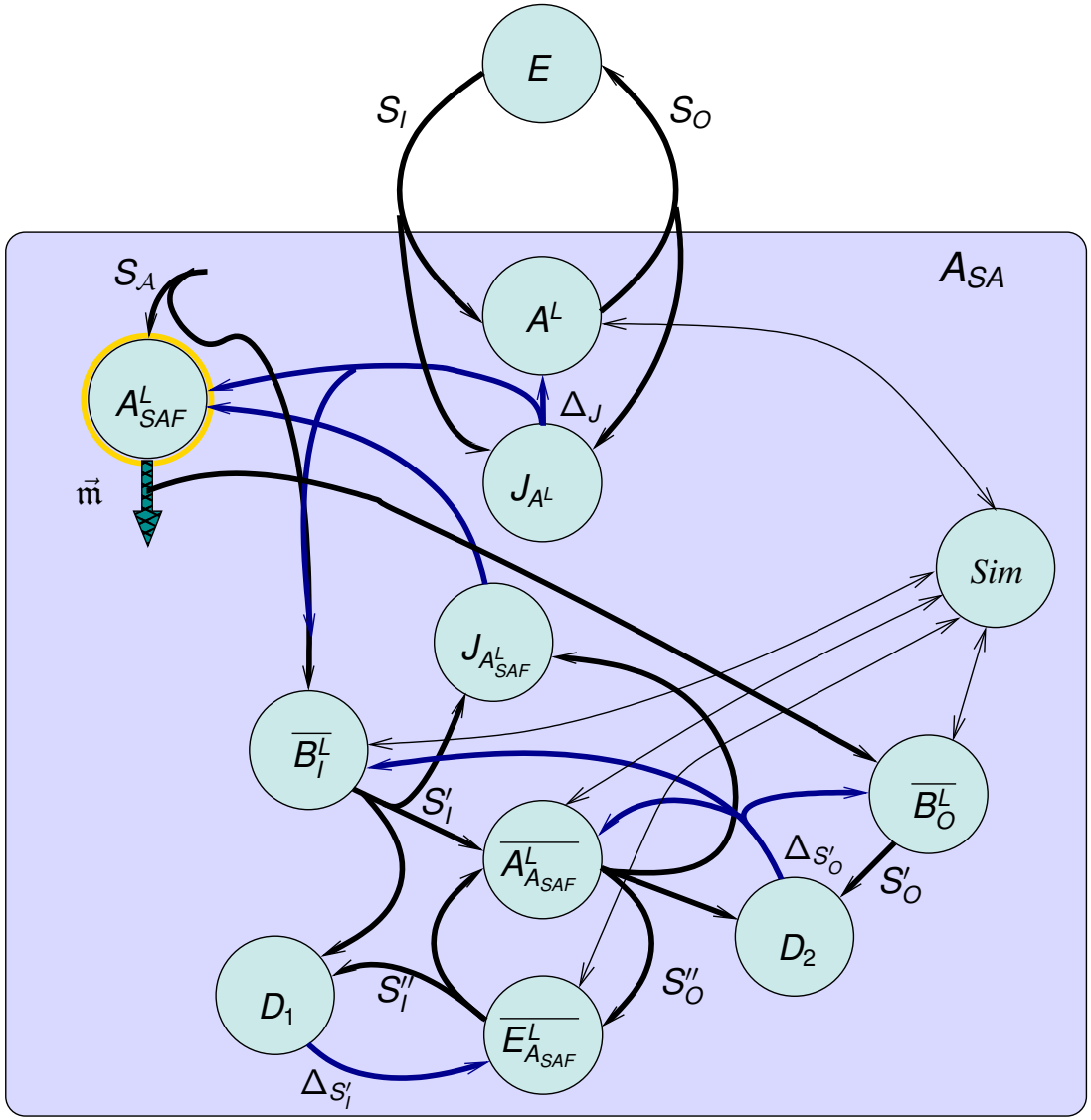


An Actor Facilitating Self-Awareness



A^L_{SAF} targets A^L_{SAF}

Self-Awareness Target: A^L_{SAF}



Summary

- Self-awareness is a machinery to study any of the system's actors;
- It uses abstraction, simulation, assessment, and other techniques;
- It can improve behavior and performance.

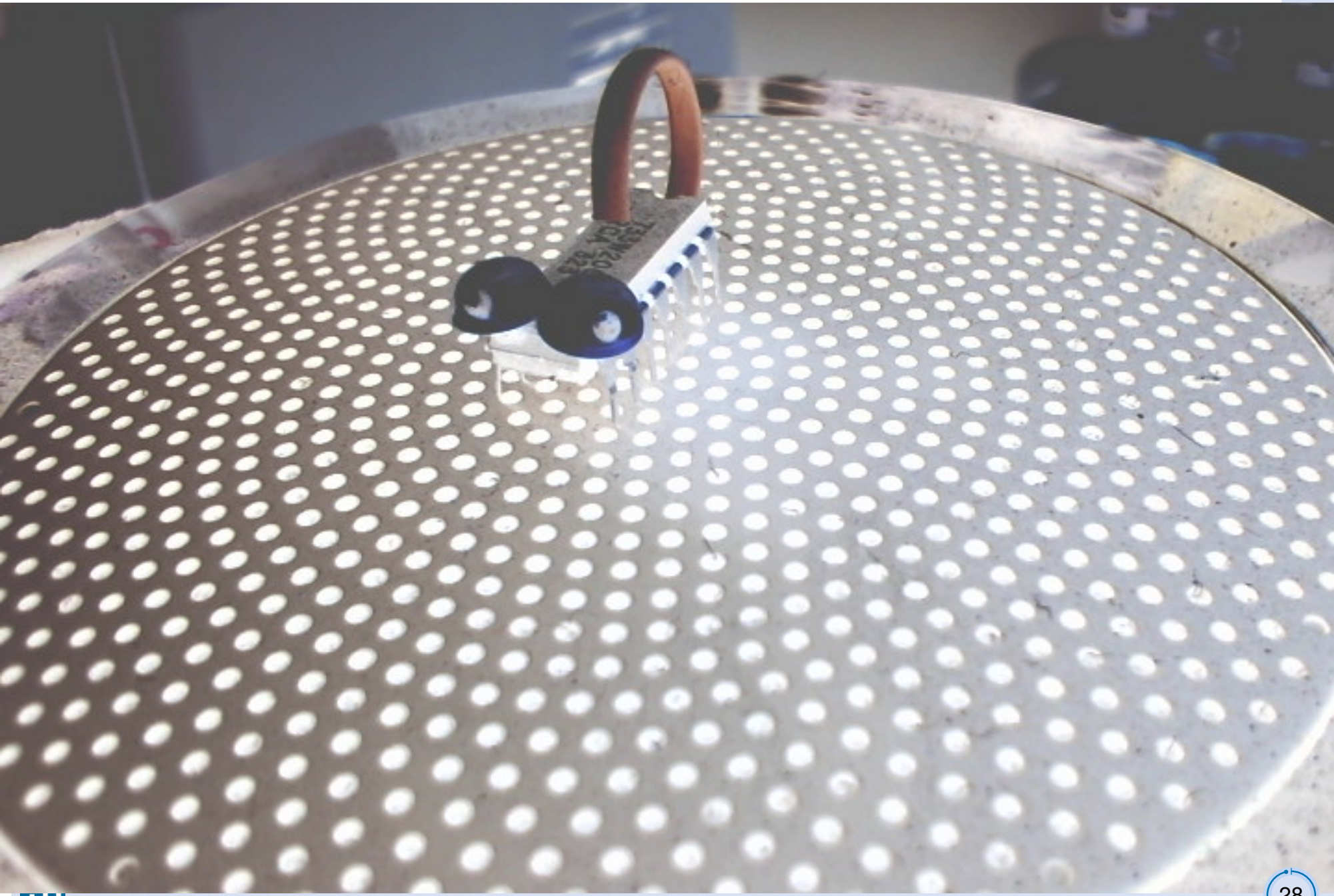
Conclusion

Issues:

- Automatic, efficient abstraction techniques
- Assessment techniques
- Goal management
- Learning
- Simulation

Promise:

- Any actor can be abstracted any number of times $\rightarrow$ no situation is too complex to analyze;
- Any actor can be subject to the scrutiny of self-awareness;
- General machinery applicable to a wide range of application domains;
- Due to continuous learning it is as efficient and effective as a custom design.



References I

Nikil Dutt, Amir M. Rahmani, and Axel Jantsch. “Empowering Autonomy through Self-awareness in MPSoCs”. In: **Proceedings of the IEEE NEWCAS Conference**. Strasbourg, France, June 2017.

Maximilian Götzinger et al. “Model-free Condition Monitoring with Confidence”. In: **International Journal of Computer Integrated Manufacturing** (2019).

Axel Jantsch. “Towards a Formal Model of Recursive Self-Reflection”. In: **International Workshop on Autonomous Systems Design (ASD)**. Mar. 2019, 6:1–6:16.

Axel Jantsch, Nikil Dutt, and Amir M. Rahmani. “Self-Awareness in Systems on Chip – A Survey”. In: **IEEE Design Test** 34.6 (Dec. 2017), pp. 1–19.

Samuel Kounev et al. “Model-driven Algorithms and Architectures for Self-Aware Computing Systems (Dagstuhl Seminar 15041)”. In: **Dagstuhl Reports** 5.1 (2015). Ed. by Samuel Kounev et al., pp. 164–196.