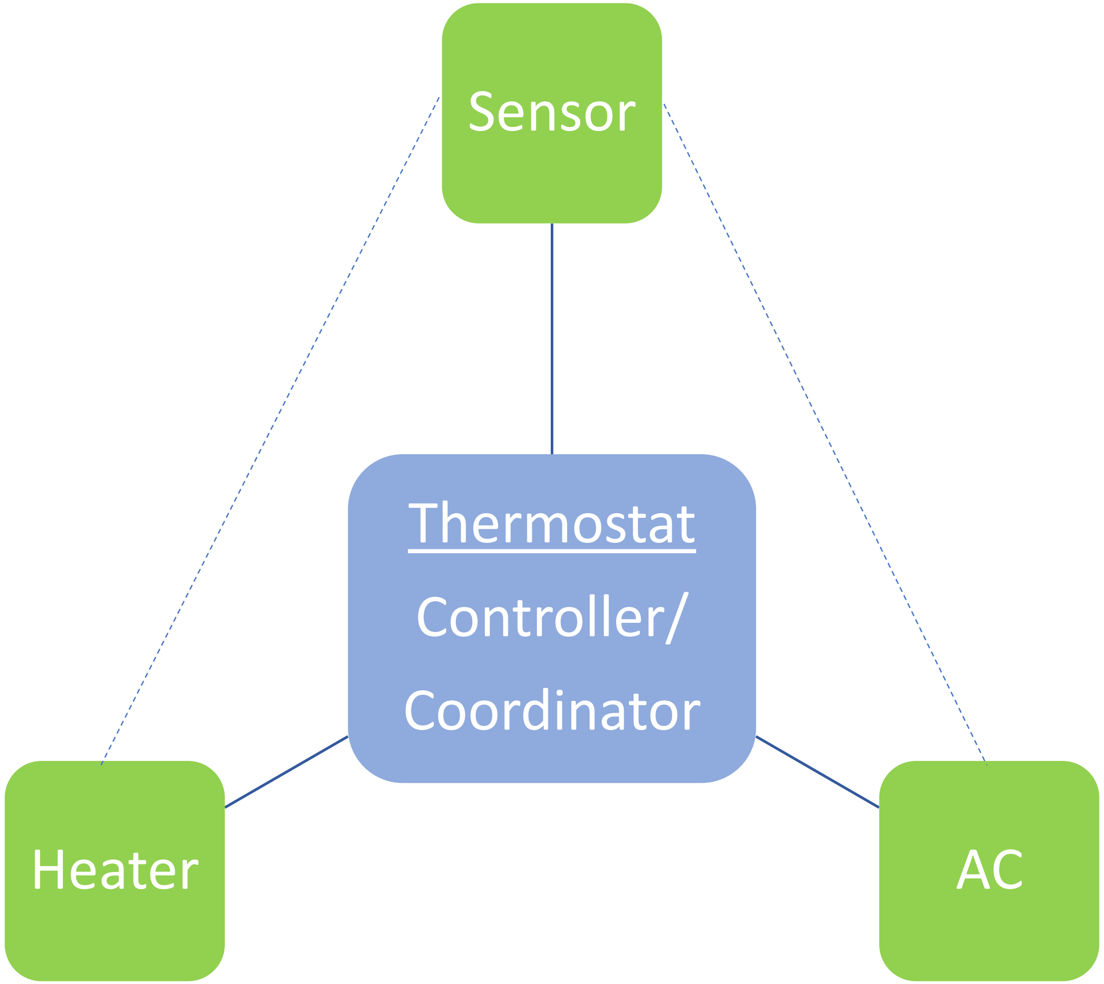
Synthesis
of
coordination programs
from
temporal specifications

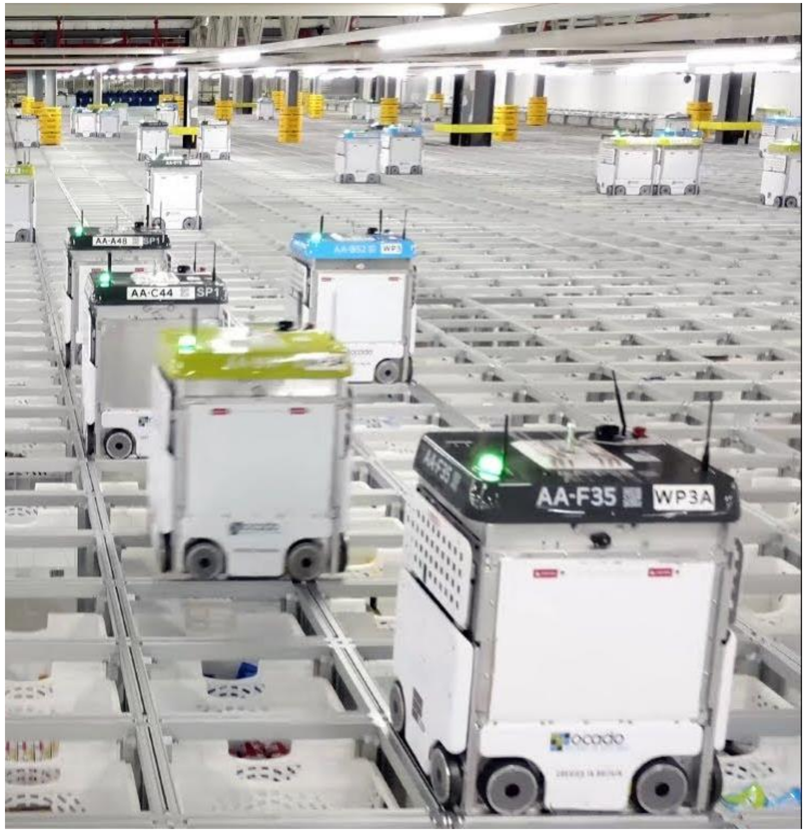
Suguman Bansal, Rice University
Kedar Namjoshi, Nokia Bell Labs, Murray Hill
Yaniv Sa’ar, Nokia Bell Labs, Kfar Saba

Coordination programs

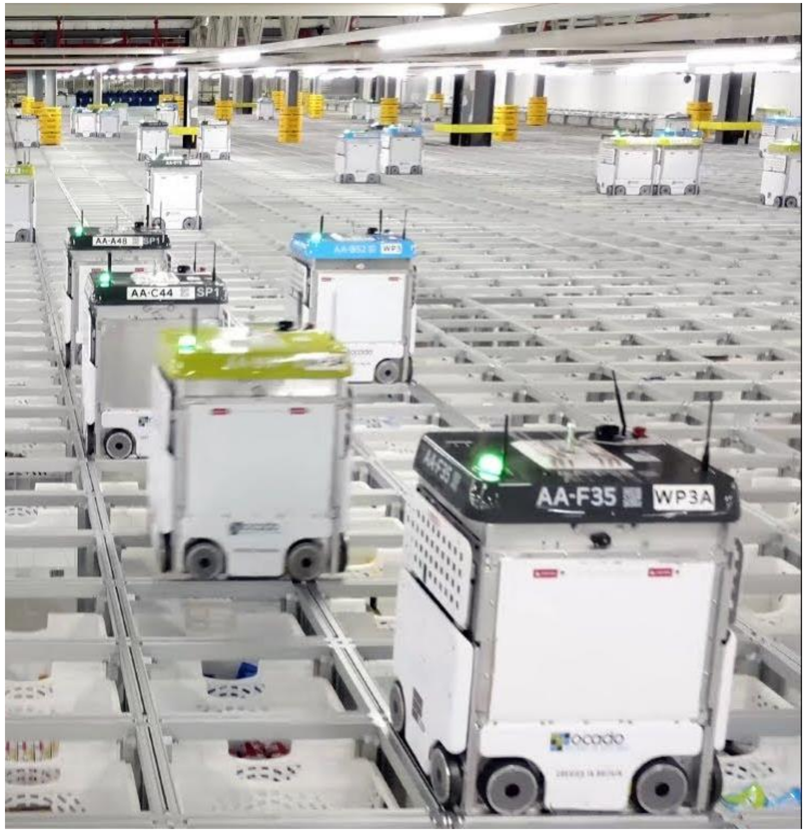
Coordination programs

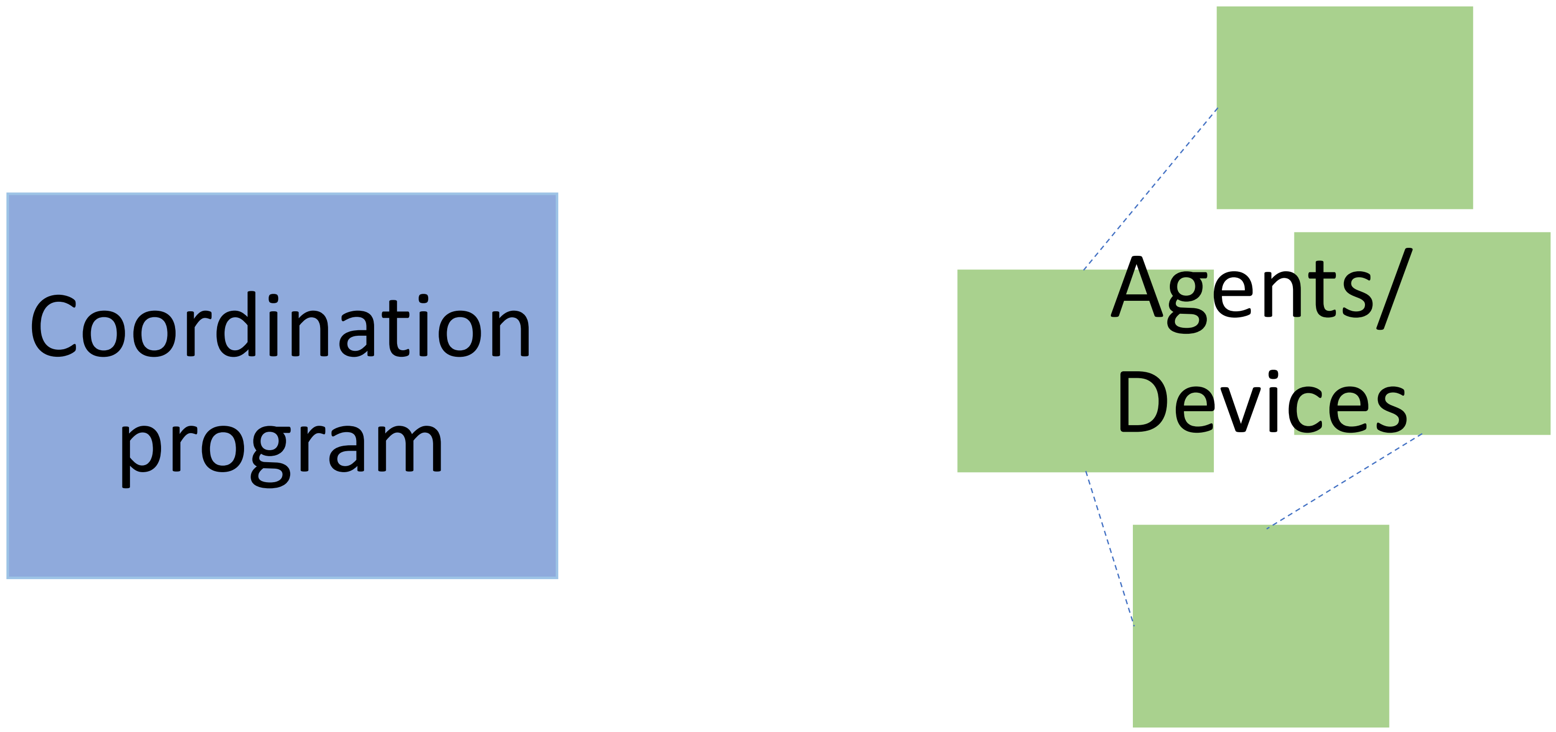


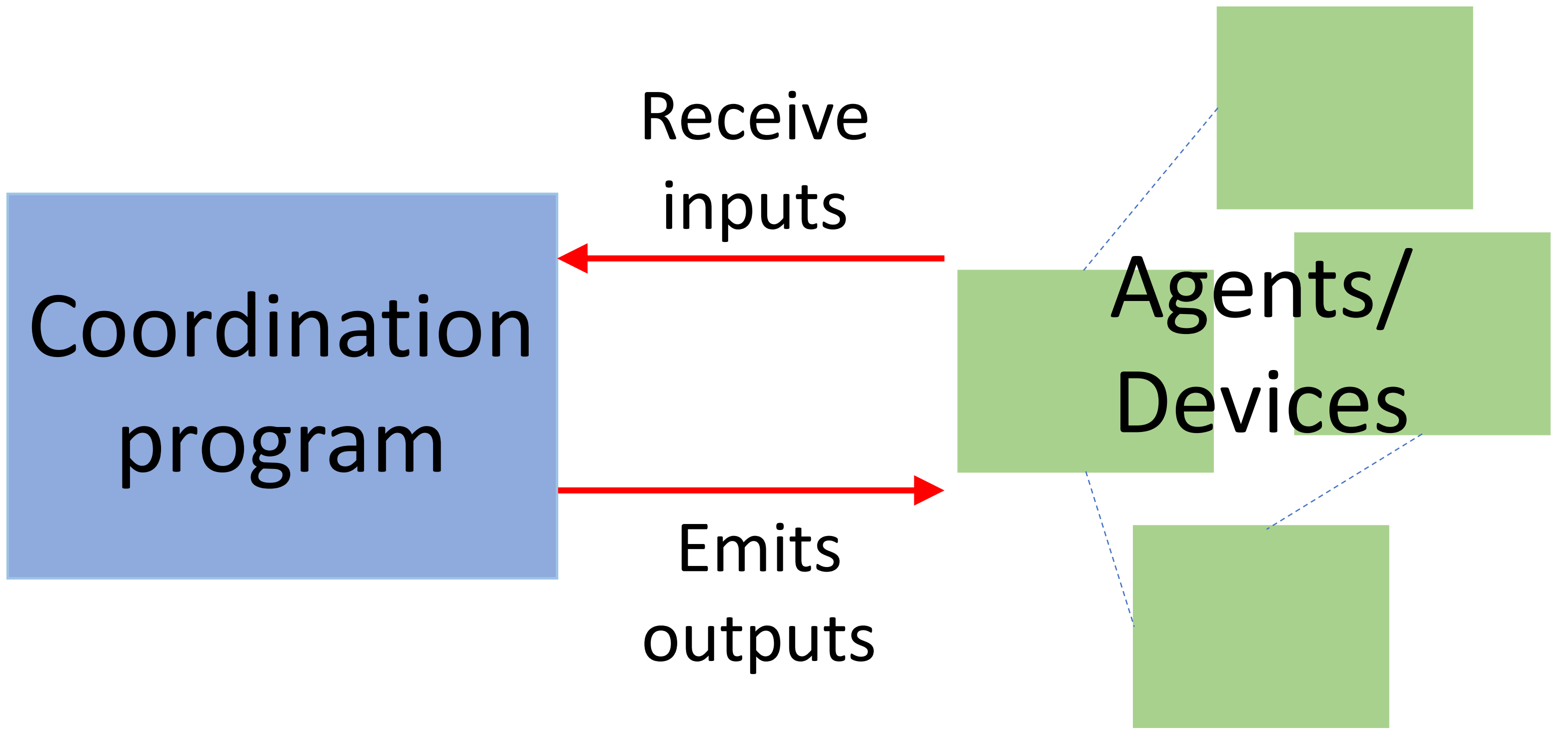
Coordination programs

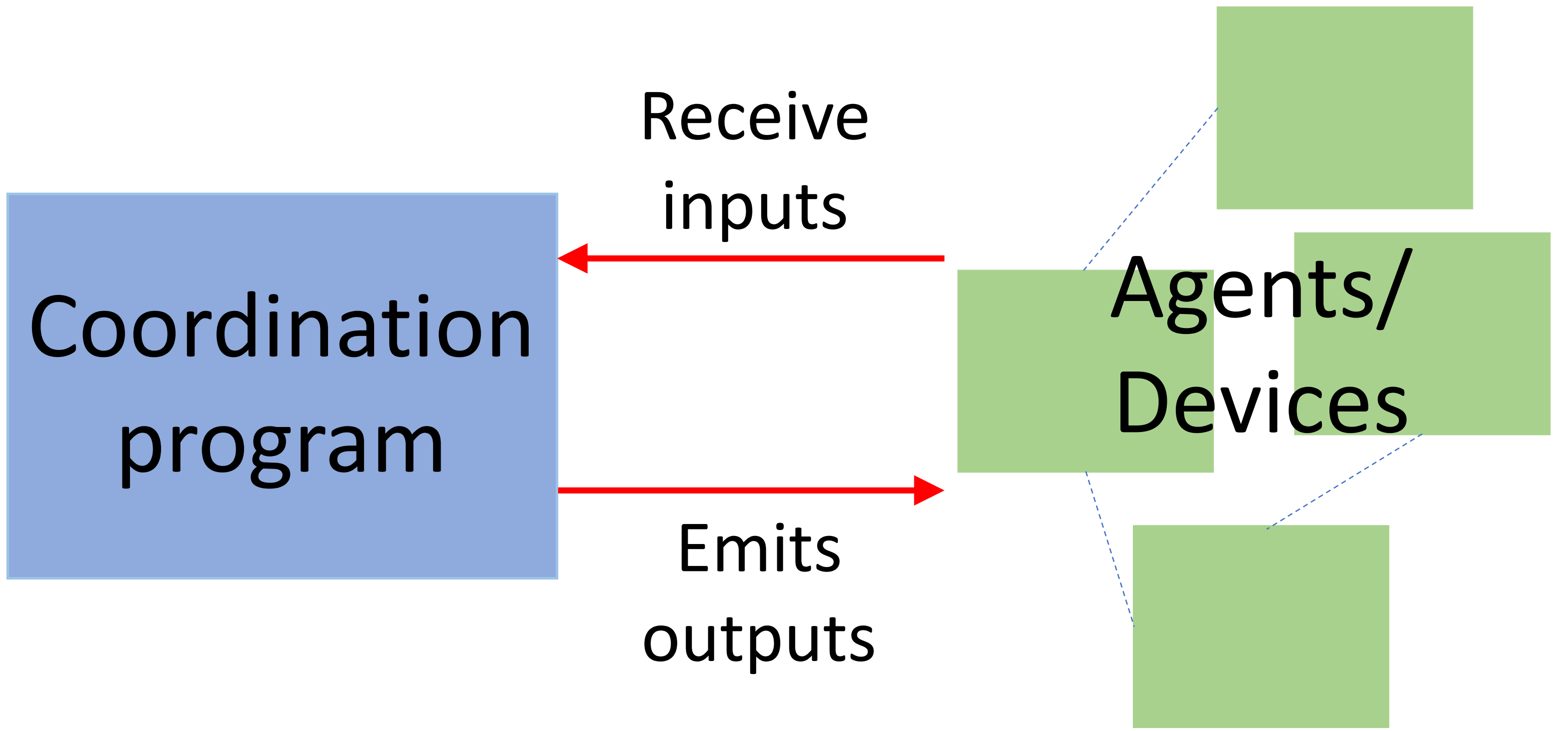


Coordination programs

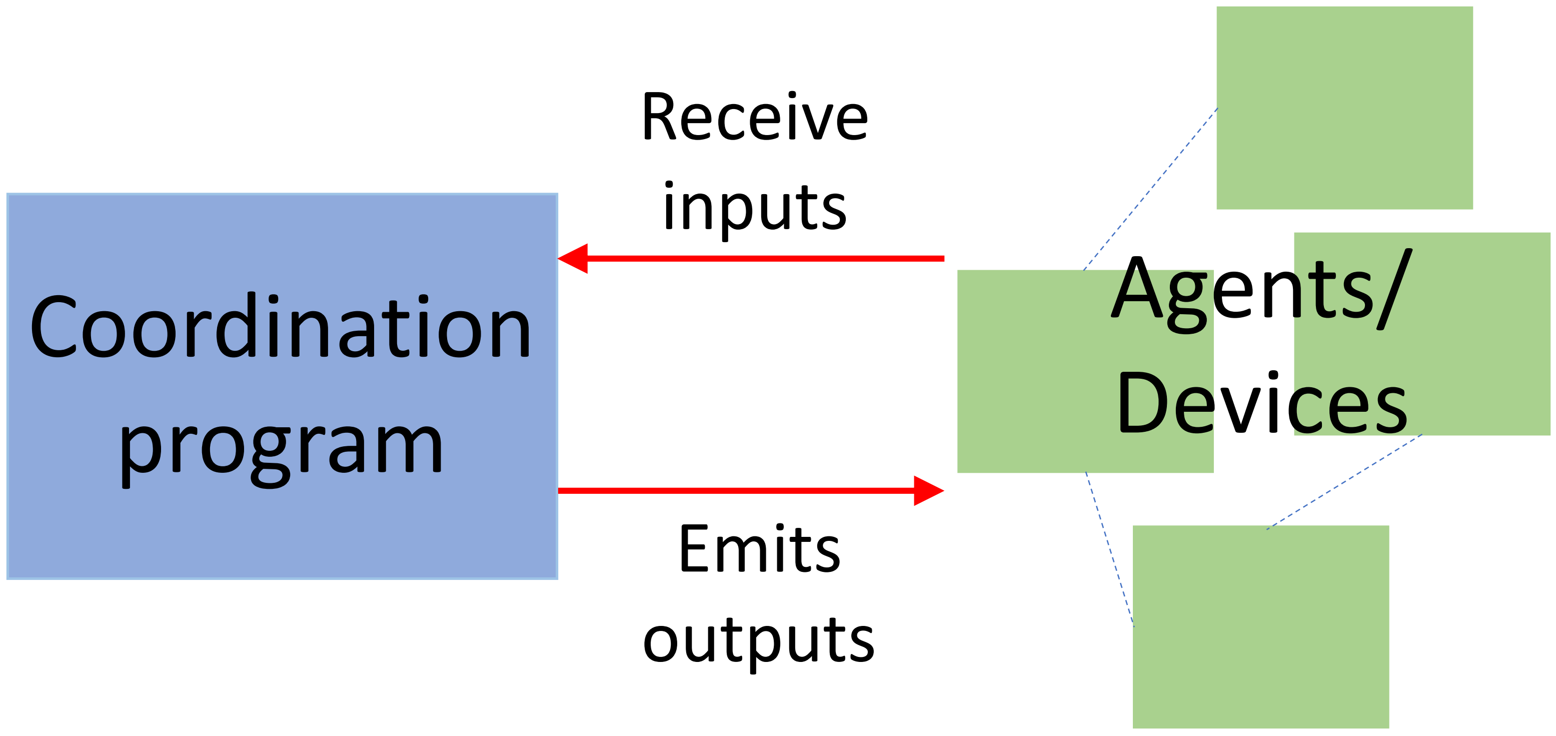



Coordination program
Agents/ Devices


Receive
inputs
Coordination
program
Agents/
Devices
Emits
outputs



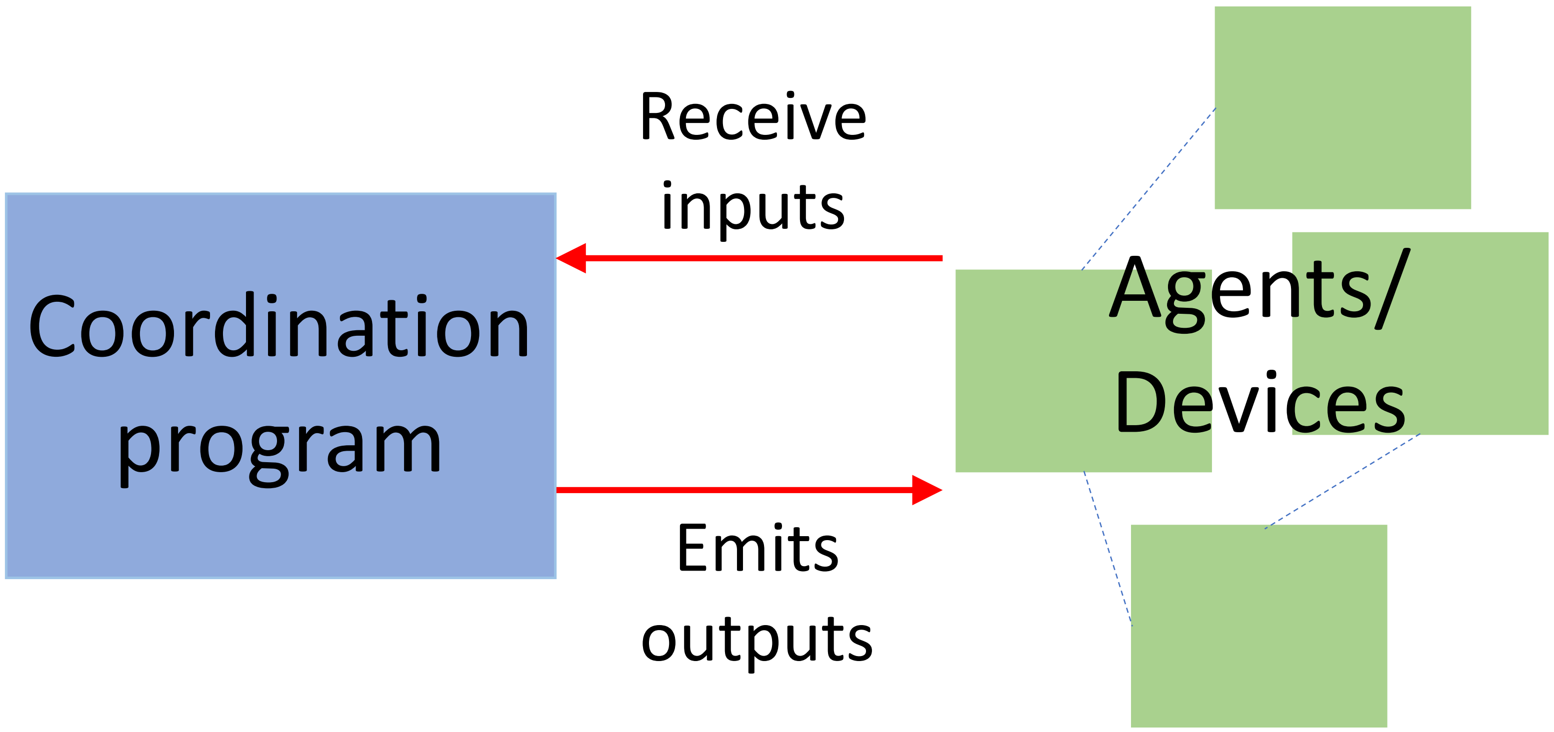
Designing coordination programs is hard



Designing coordination programs is hard

1. Reactive system

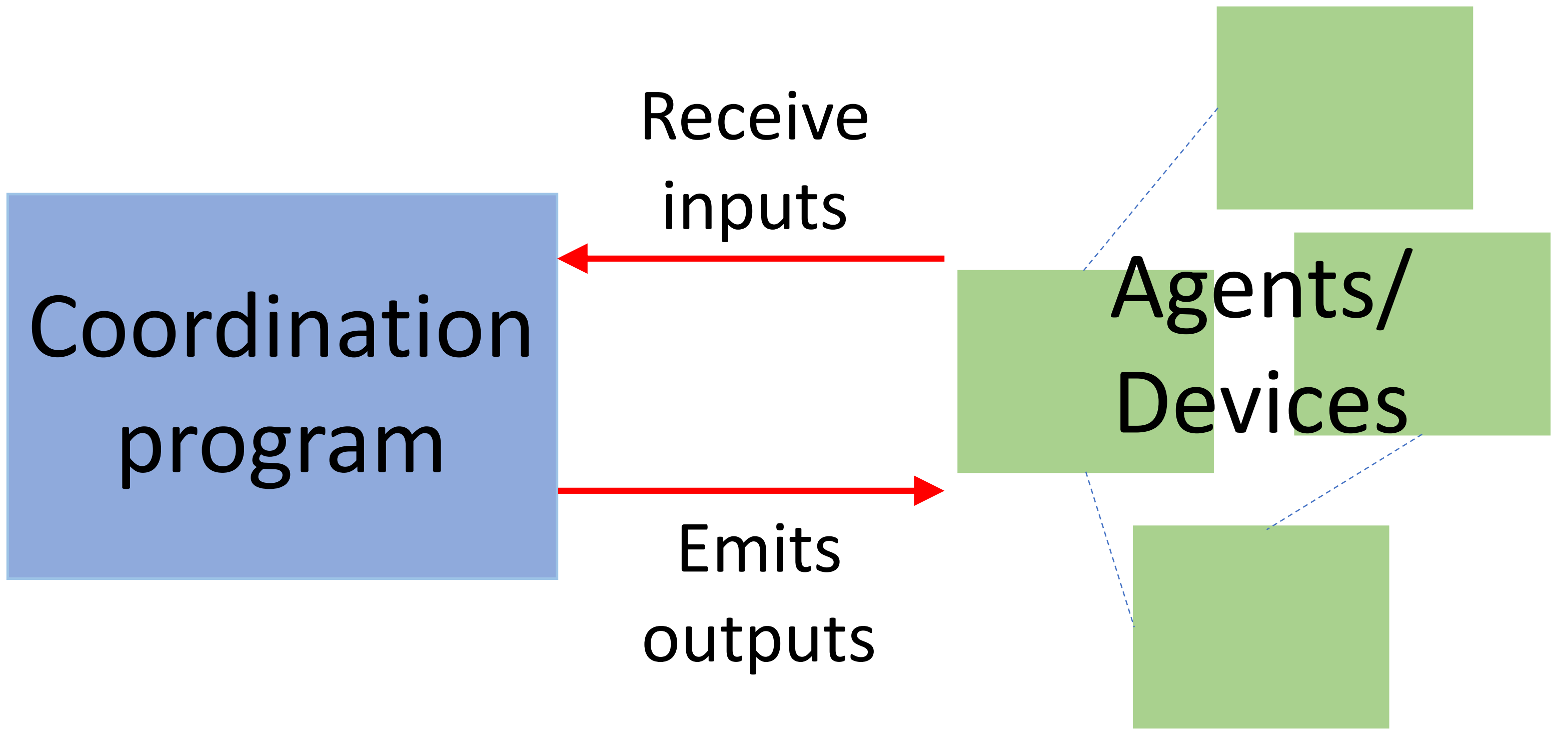
Repeated interaction with agents/devices



Designing coordination programs is hard

1. Reactive system
2. Asynchrony

Each device operates at its own clock



Coordination program has limited visibility:
Sees the interface only

Designing coordination programs is hard

1. Reactive system
2. Asynchrony
3. Partial information

Specifying intent of a coordination program is easier

E.g. “Thermostat should maintain ambient temperature”

Specifying intent of a coordination program is easier

E.g. “Thermostat should maintain ambient temperature”

Can we automatically generate a coordination program from a high-level specification?

Specifying intent of a coordination program is easier

E.g. “Thermostat should maintain ambient temperature”

Can we automatically generate a coordination program from a high-level specification?

Coordination synthesis

Contributions

Formulate, solve, and demonstrate coordination synthesis

Contributions

Formulate, solve, and demonstrate coordination synthesis

Formalize coordination synthesis

"Easier to specify" formalization

Contributions

Formulate, solve, and demonstrate coordination synthesis

Formalize coordination synthesis

"Easier to specify" formalization

Design efficient automata-based synthesis algorithm

Accounts for all three challenges – Reactive, asynchrony, partial information

Prior work accounts for at most two

Contributions

Formulate, solve, and demonstrate coordination synthesis

Formalize coordination synthesis

"Easier to specify" formalization

Design efficient automata-based synthesis algorithm

Accounts for all three challenges – Reactive, asynchrony, partial information

Prior work accounts for at most two

Conduct case-studies on prototype implementation

High-level specification

Intent of coordination program

Linear Temporal Logic (LTL) [Pnueli, FOCS 1977]

High-level specification

Intent of coordination program

Linear Temporal Logic (LTL) [Pnueli, FOCS 1977]

Device description

Communicating Sequential Processes (CSP) [Hoare, CACM 1978]

- Rich structure: asynchrony, non-determinism …
- Communication model: Message passing
- Interface: Visible and hidden actions

CSP processes [Hoare, , CACM 1978]

Processes: $P, Q, R, \ldots$

- Public actions $(a_0, a_1, \ldots a_n)$ and Private actions $(b_0, b_1, \ldots b_n)$

$$P = action_0 \rightarrow Q_0 \mid action_1 \rightarrow Q_1 \mid \ \ldots \ \mid action_n \rightarrow Q_n$$

"Process P evolves to process Q_i on $action_i$"

CSP processes [Hoare, , CACM 1978]

Processes: $P, Q, R, \ldots$

- Public actions $(a_0, a_1, \ldots a_n)$ and Private actions $(b_0, b_1, \ldots b_n)$

$$P = action_0 \rightarrow Q_0 \mid action_1 \rightarrow Q_1 \mid \ \ldots \ \mid action_n \rightarrow Q_n$$

"Process P evolves to process Q_i on $action_i$"

- Allows structural non-determinism: $P = a \rightarrow Q_0 \mid a \rightarrow Q_1$

CSP processes [Hoare, , CACM 1978]

Processes: $P, Q, R, \ldots$

- Public actions $(a_0, a_1, \ldots a_n)$ and Private actions $(b_0, b_1, \ldots b_n)$

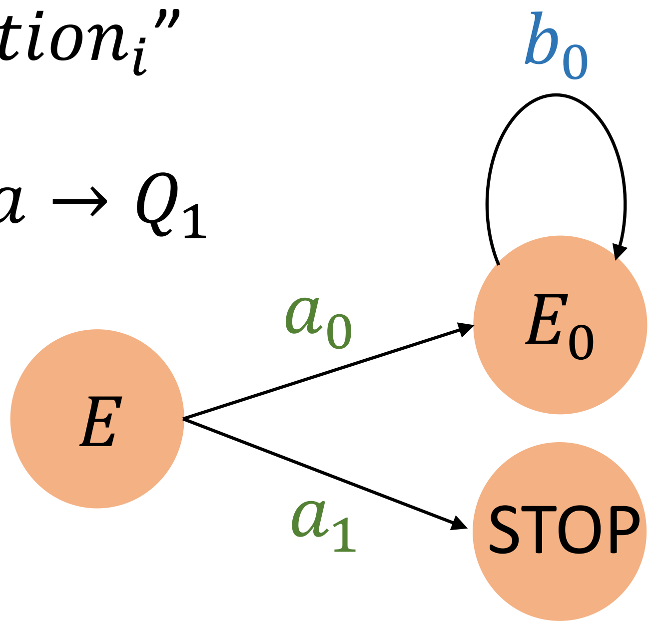
$$P = action_0 \rightarrow Q_0 \mid action_1 \rightarrow Q_1 \mid \ \ldots \ \mid action_n \rightarrow Q_n$$

"Process P evolves to process Q_i on $action_i$"

- Allows structural non-determinism: $P = a \rightarrow Q_0 \mid a \rightarrow Q_1$

Example: $E = a_0 \rightarrow E_0 \mid a_1 \rightarrow STOP$

$E_0 = b_0 \rightarrow E_0$



CSP interactions [Hoare, 1978]

Synchronized public actions

- Processes evolve together on public actions

  Let, $P_0 = a \rightarrow Q_0, \quad P_1 = a \rightarrow Q_1$ then $(P_0 \mid\mid P_1) = a \rightarrow (Q_0 \mid\mid Q_1)$

CSP interactions [Hoare, 1978]

Synchronized public actions

- Processes evolve together on public actions

  Let, $P_0 = a \to Q_0, \quad P_1 = a \to Q_1$ then $(P_0 \,||\, P_1) = a \to (Q_0 \,||\, Q_1)$

  Let, $P_0 = a_0 \to Q_0, \quad P_1 = a_1 \to Q_1$ then $(P_0 \,||\, P_1)$ no evolution

CSP interactions [Hoare, 1978]

Synchronized public actions

- Processes evolve together on public actions

  Let, $P_0 = a \rightarrow Q_0, \quad P_1 = a \rightarrow Q_1$ then $(P_0 \mid\mid P_1) = a \rightarrow (Q_0 \mid\mid Q_1)$

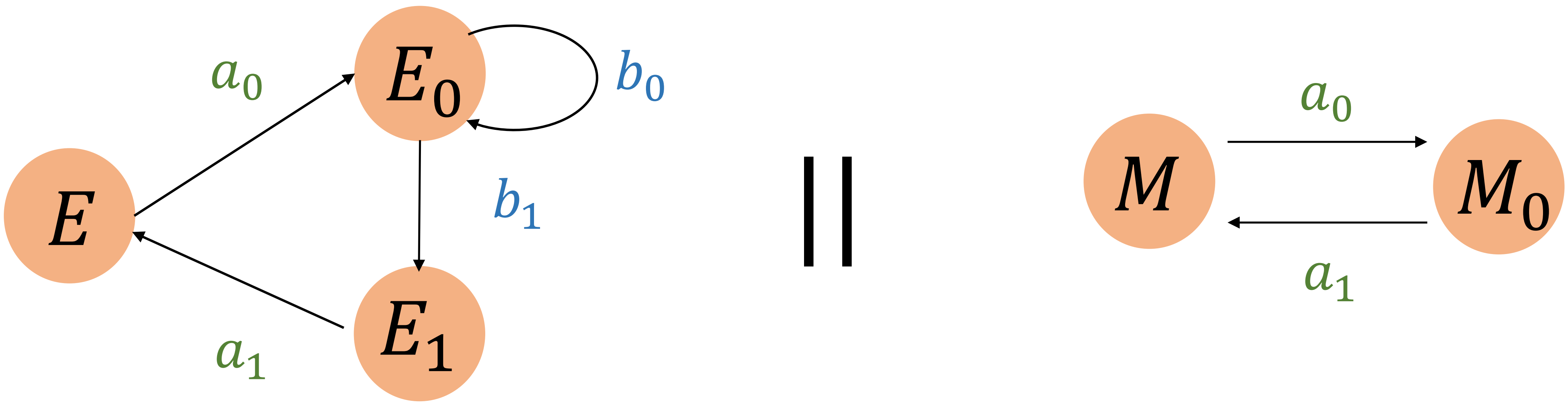
  Let, $P_0 = a_0 \rightarrow Q_0, \quad P_1 = a_1 \rightarrow Q_1$ then $(P_0 \mid\mid P_1)$ no evolution

Internal private actions

- Process with private action evolves by itself

  Let, $P_0 = b \rightarrow Q_0, \quad P_1 = a \rightarrow Q_1$ then $(P_0 \mid\mid P_1) = b \rightarrow (Q_0 \mid\mid P_1)$

Example



$a_0 b_0 b_0 b_0 \ldots$

$a_0 b_1 a_1 \ldots$

$\ldots$

Coordination synthesis

Given a CSP environment description

E = E1 || E2 || … || En

and an LTL specification **S**,

Generate a coordinator (CSP) **M** s.t.
E || M satisfies **S**

Challenge I: Partial information

Coordinator may not know the current state of **E**

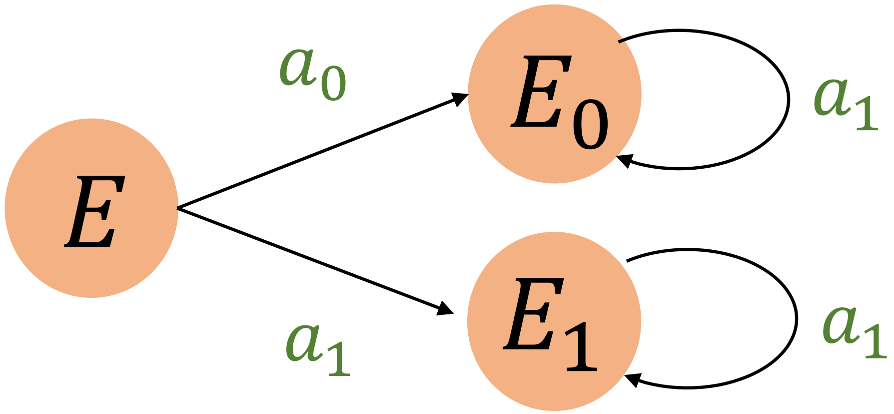
- Structural non-determinism:
  Evolution to multiple states

- Internal actions:
  Evolution due to internal actions is unknown to coordinator

Challenge II: Deadlock freedom

Coordinator must guarantee no deadlock despite partial information

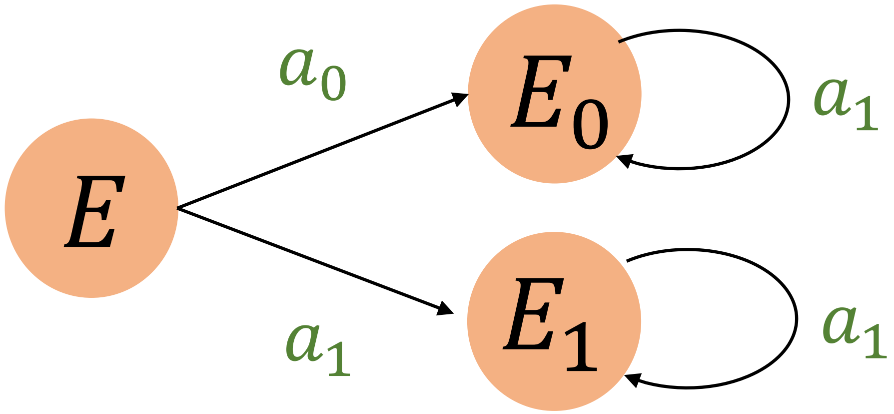
$$
\begin{aligned}
E &= a_0 \to E_0 \mid a_1 \to E_1 \\
E_0 &= a_1 \to E_0 \\
E_1 &= a_1 \to E_1
\end{aligned}
$$



Challenge II: Deadlock freedom

Coordinator must guarantee no deadlock despite partial information

$$E = a_0 \to E_0 \mid a_1 \to E_1$$
$$E_0 = a_1 \to E_0$$
$$E_1 = a_1 \to E_1$$



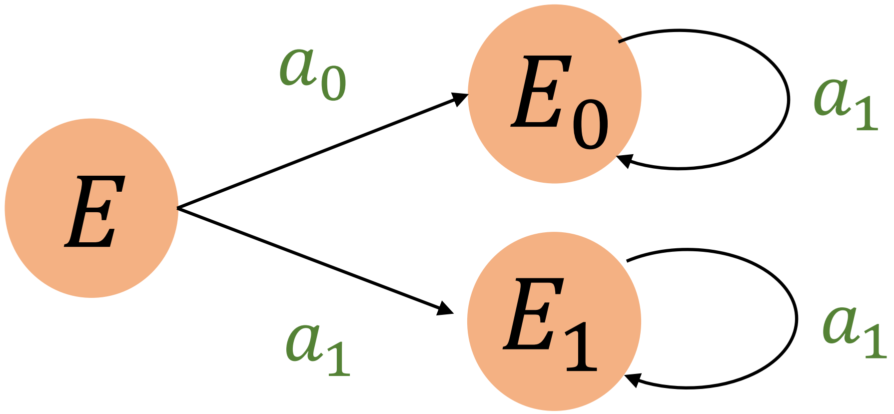
$$M = a_0 \to M$$

$E||M$ deadlocks

Challenge II: Deadlock freedom

Coordinator must guarantee no deadlock despite partial information

$$E \;= a_0 \to E_0 \mid a_1 \to E_1$$
$$E_0 = a_1 \to E_0$$
$$E_1 = a_1 \to E_1$$



$$M = a_0 \to M$$

$E||M$ deadlocks

$$M = a_1 \to M$$

$E||M$ does not deadlock

Challenge III: Asynchrony

Coordinator doesn't know how many or which actions have taken place

a_0 a_1 a_2 a_3 ....

Expansion

a_0 b_0 a_1 b_1 a_2 b_2 b_3 a_3 ....

Every valid expansion satisfies **S**

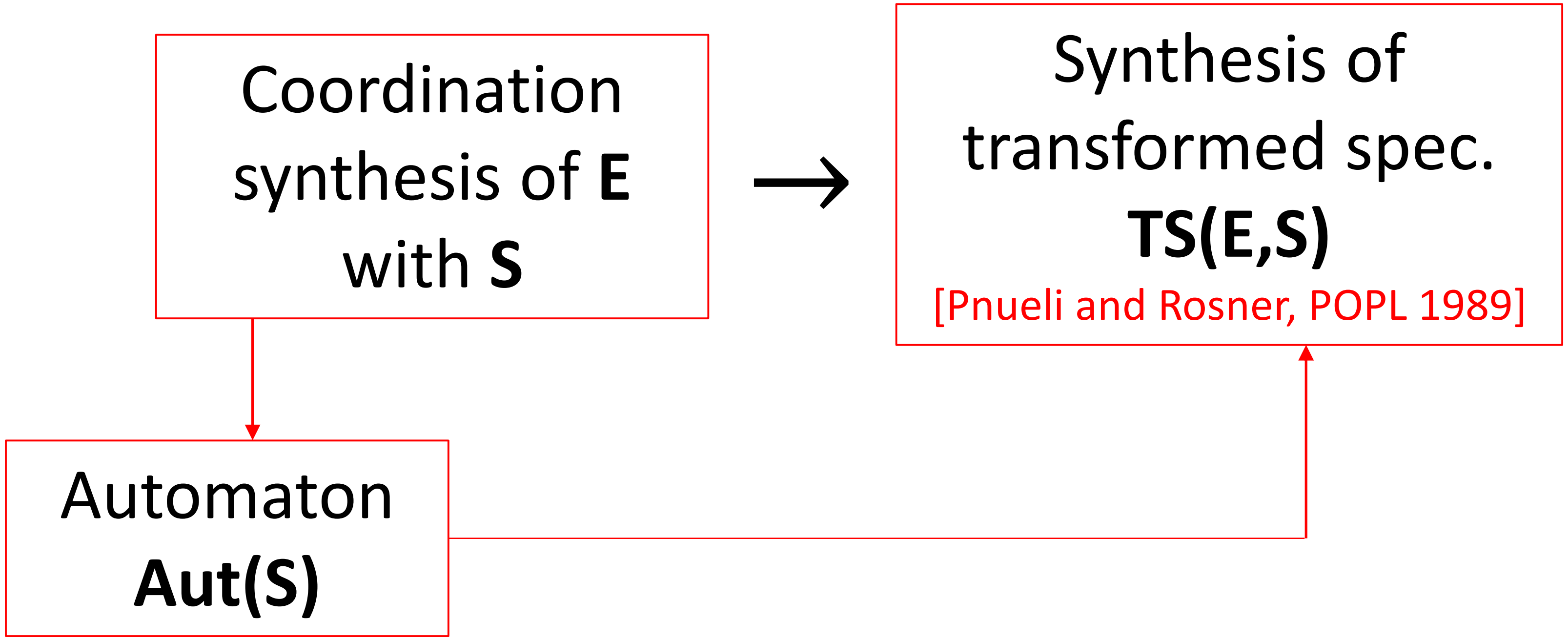
Core technique

Coordination synthesis with **E,S** reduces to Synch. synthesis with **TS(E,S)**

Coordination synthesis of **E** with **S**	→	Synthesis of transformed spec. **TS(E,S)** [Pnueli and Rosner, POPL 1989]

Core technique

Coordination synthesis with **E,S** reduces to Synch. synthesis with **TS(E,S)**



Challenge III: Asynchrony

Coordinator doesn't know how many or which actions have taken place

a_0 a_1 a_2 a_3 ....

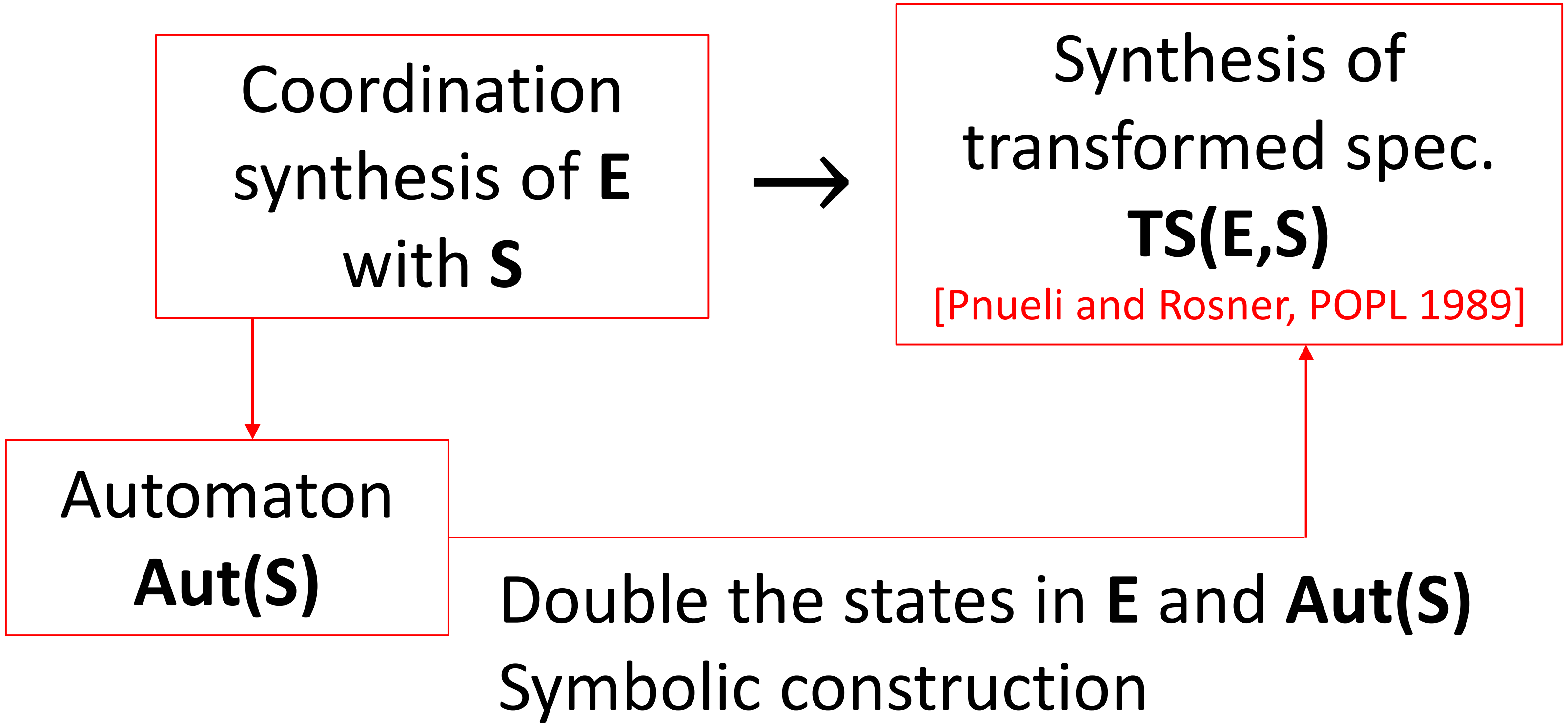
Expansion

a_0 b_0 a_1 b_1 a_2 b_2 b_3 a_3 ....

Every valid expansion satisfies **S**

Core technique

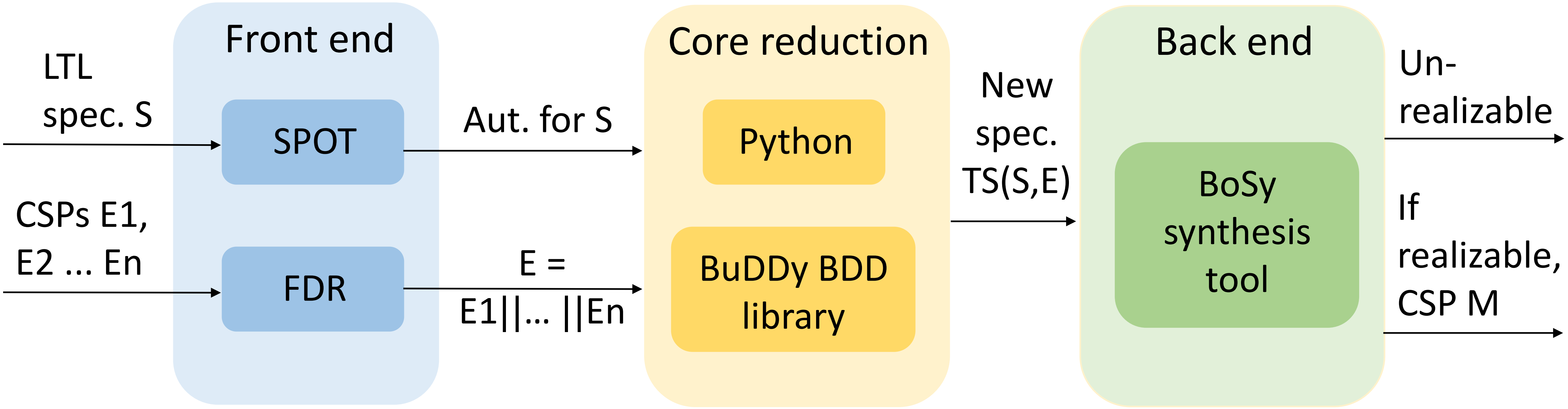
Coordination synthesis with **E,S** reduces to Synch. synthesis with **TS(E,S)**



Implementation

Prototype **CoSy** (**Co**ordination **Sy**nthesis)

- Core reduction in Python
- BDD-based symbolic reduction



Thermostat

How will coordination synthesis help?

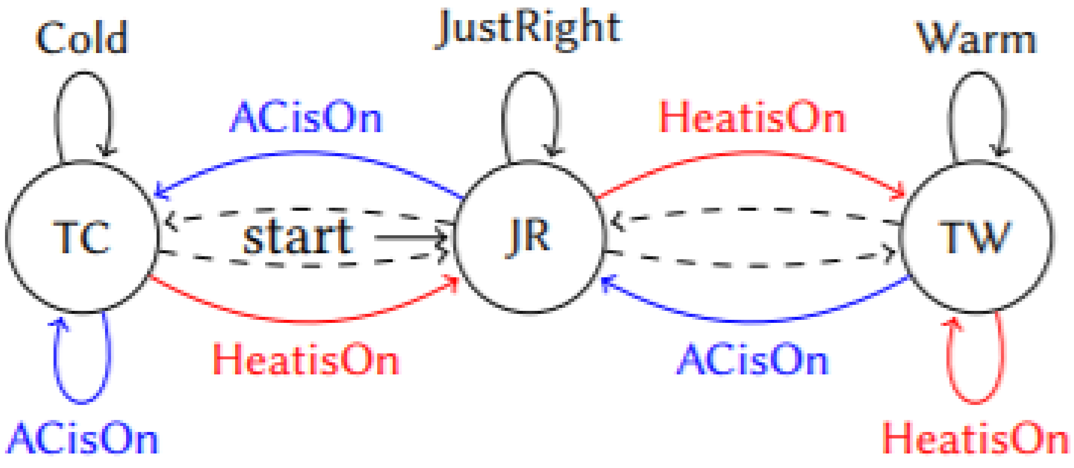


Fig. 6. Room temperature Sensor process (JR: Just right, TW: Too warm, TC: Too cold).

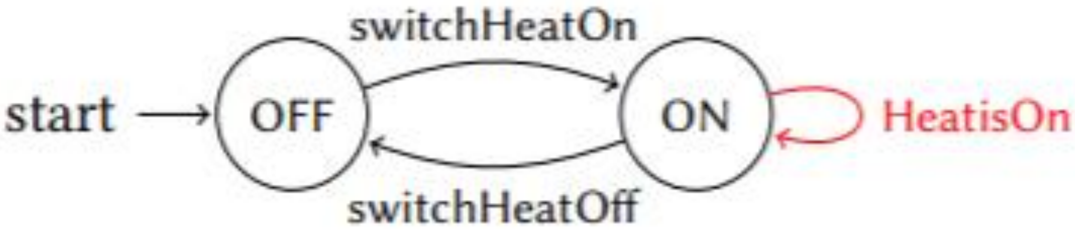


Fig. 7. Heater process.

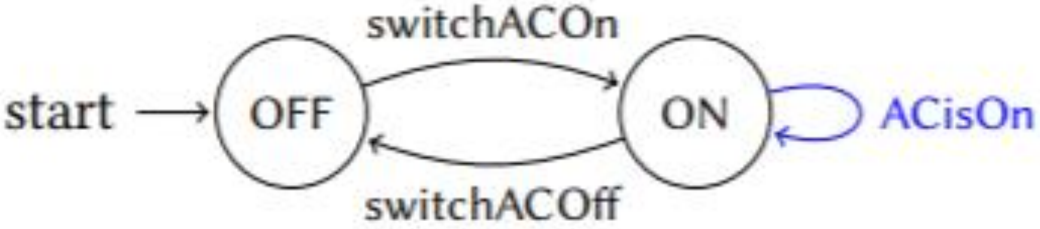


Fig. 8. AC process.

Thermostat case study

Phase I

Maintain ambient temperature

Thermostat doesn’t interact with all devices

+

Phase II

Thermostat must interact with all devices (Fairness)

AC and Heater are switched on at the same time

+

Phase III

Energy efficient

SATISFIED!

Coordination synthesis, in a nutshell

- Modelling
  - CSP environments – Interface, non-determinism, private/public actions
  - LTL specifications – Expressive
- Algorithm
  - Efficient automata-based reduction to synchronous synthesis
  - Prototype + Case studies demonstrate utility
- Complexity analysis
  - PSPACE-hard in size of E
  - Algorithm is exponential in E, number of devices