UNIVERSITÀ DEGLI STUDI DI UDINE
hic sunt futura

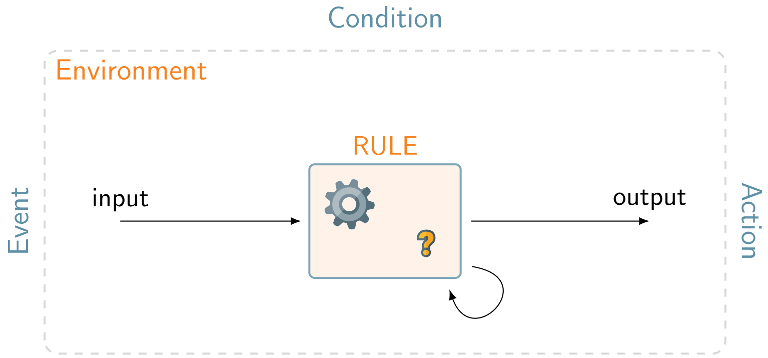
An Attribute-Based Events Model for Collective Adaptive Systems

based on joint work with M. Pasqua (U. Verona), M. Paier (IMT Lucca), and others

Marino Miculan
marino.miculan@uniud.it

February 2, 2024

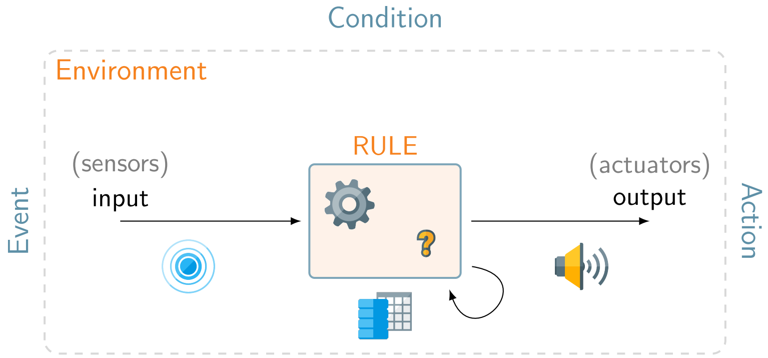
Theory Days at Randivälja



State-based ECA rules: "**on** movement **if** alarm = "active" **then** siren ← on"

variables can be internal, or connected to sensors or to actuators

Event-driven programming of smart systems

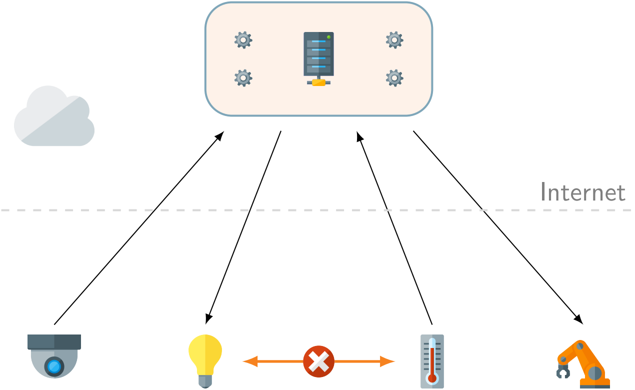


State-based ECA rules: "**on** movement **if** alarm = "active" **then** siren ← on"

variables can be internal, or connected to sensors or to actuators

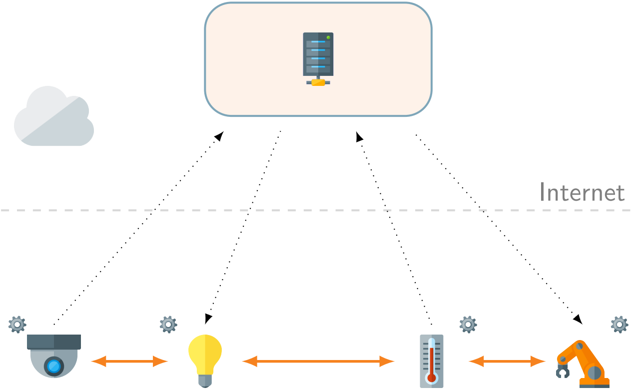
Actual smart (ECA) devices setting

- Centralized
- No intra-nodes communication
- Cloud-dependent
- Very popular as Trigger-Action Platforms (TAP):



Next (ECA) IoT architecture: *edge computing*

- Fully distributed
- Communication between nodes
- Cloud-agnostic
- Identity decoupled, for scalability
- *Collective Adaptive Systems*



Programming model for edge CAS?

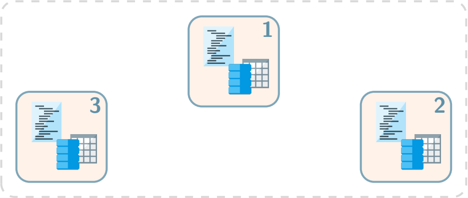
We need programming abstractions and models for edge computing with:

- peer-to-peer, decentralised control
- identity decoupling, for scalability (no point-to-point communication)
- open and flexible (nodes can join and leave dynamically)
- which integrate neatly within the ECA paradigm

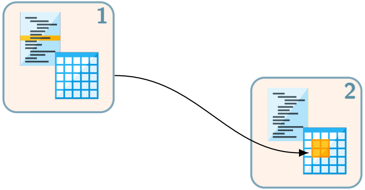
Attribute-based Memory Updates

Nodes behavior: defined by ECA rules like "on z for all Π : $x \leftarrow e$"

Nodes state: local memory



Interaction: remote updates



Attribute-based interaction: on all nodes satisfying Π, update the remote x with e

AbU syntax

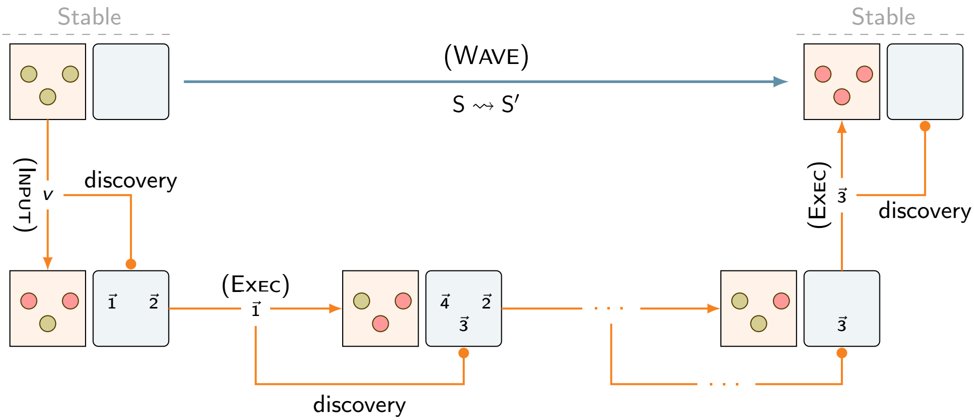
- An **AbU system** S is an **AbU node** $R, \iota\langle\Sigma, \Theta\rangle$ or the parallel of systems $S_1 \,\|\, S_2$
- Each node is equipped with a list R of **AbU rules** and an **invariant** ι specifying *admissible* states

"on all nodes with (remote) x greater than the current (local) x"

for all: $@(x < \bar{x}) : \bar{x} \leftarrow x, \bar{y} \leftarrow \bar{y} + 1$

"update the (remote) x with the current (local) x, and increment remote y"

AbU execution model



AbU operational semantics

[M. Pasqua, MM, TCS 20

LTS semantics, with judgments:

$$R, \iota\langle\Sigma, \Theta\rangle \xrightarrow{\alpha} R, \iota\langle\Sigma', \Theta'\rangle$$

A label α can be:

- an **input** label, upd $\blacktriangleright$ T
- an **execution** label, upd $\triangleright$ T
- a **discovery** label, T

AbU operational semantics: rules

$$(\textsc{Exec})\ \frac{\begin{array}{c}\mathsf{upd} \in \Theta \quad \mathsf{upd} = (x_1, v_1) \dots (x_k, v_k) \quad \Sigma' = \Sigma[v_1/x_1 \dots v_k/x_k] \quad \Sigma' \models \iota \\ \Theta'' = \Theta \setminus \{\mathsf{upd}\} \quad X = \{x_i \mid i \in [1..k] \wedge \Sigma(x_i) \neq \Sigma'(x_i)\} \\ \Theta' = \Theta'' \cup \mathsf{DefUpds}(R, X, \Sigma') \cup \mathsf{LocalUpds}(R, X, \Sigma') \quad T = \mathsf{ExtTasks}(R, X, \Sigma')\end{array}}{R, \iota\langle\Sigma, \Theta\rangle \xrightarrow{\mathsf{upd} \triangleright T} R, \iota\langle\Sigma', \Theta'\rangle}$$

$$(\textsc{Exec-Fail})\ \frac{\mathsf{upd} \in \Theta \quad \mathsf{upd} = (x_1, v_1) \dots (x_k, v_k) \quad \Sigma' = \Sigma[v_1/x_1 \dots v_k/x_k] \quad \Sigma' \not\models \iota \quad \Theta' = \Theta \setminus \{\mathsf{upd}\}}{R, \iota\langle\Sigma, \Theta\rangle \xrightarrow{\mathsf{upd} \triangleright T} R, \iota\langle\Sigma, \Theta'\rangle}$$

$$(\textsc{Input})\ \frac{\begin{array}{c}v_1, \dots, v_k \in \mathbb{V} \quad \Sigma' = \Sigma[v_1/x_1 \dots v_k/x_k] \quad X = \{x_1, \dots, x_k\} \\ \Theta' = \Theta \cup \mathsf{DefUpds}(R, X, \Sigma') \cup \mathsf{LocalUpds}(R, X, \Sigma') \quad T = \mathsf{ExtTasks}(R, X, \Sigma')\end{array}}{R, \iota\langle\Sigma, \Theta\rangle \xrightarrow{(x_1,v_1)\dots(x_k,v_k) \blacktriangleright T} R, \iota\langle\Sigma', \Theta'\rangle}$$

$$(\textsc{Disc})\ \frac{\Theta'' = \{[\![\mathsf{act}]\!]\Sigma \mid \exists i \in [1..n]\,.\,\mathsf{task}_i = \varphi : \mathsf{act} \wedge \Sigma \models \varphi\} \quad \Theta' = \Theta \cup \Theta''}{R, \iota\langle\Sigma, \Theta\rangle \xrightarrow{\mathsf{task}_1 \dots \mathsf{task}_n} R, \iota\langle\Sigma, \Theta'\rangle}$$

$$(\textsc{StepL})\ \frac{\mathsf{S}_1 \xrightarrow{\alpha} \mathsf{S}_1' \quad \mathsf{S}_2 \xrightarrow{T} \mathsf{S}_2'}{\mathsf{S}_1 \parallel \mathsf{S}_2 \xrightarrow{\alpha} \mathsf{S}_1' \parallel \mathsf{S}_2'}\ \alpha \in \{\mathsf{upd} \triangleright T, \mathsf{upd} \blacktriangleright T\} \qquad (\textsc{StepR})\ \frac{\mathsf{S}_1 \xrightarrow{T} \mathsf{S}_1' \quad \mathsf{S}_2 \xrightarrow{\alpha} \mathsf{S}_2'}{\mathsf{S}_1 \parallel \mathsf{S}_2 \xrightarrow{\alpha} \mathsf{S}_1' \parallel \mathsf{S}_2'}\ \alpha \in \{\mathsf{upd} \triangleright T, \mathsf{upd} \blacktriangleright T\}$$

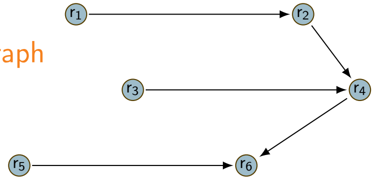
Research questions and problems

1. Stability: after an input, does a wave computation always terminates?
2. Confluence: will different executions end up with the same state(s)?
3. Global invariants: how to guarantee that trajectories will not invalidate a given global property?
4. Security: how to avoid information leakages?
5. Safety: how to avoid unintended interactions?
6. Implementation: how to make it efficient, portable and scalable?
7. ...

Stabilization

The wave semantics may exhibit internal divergence, namely $S \xrightarrow{\alpha_0} S^0 \xrightarrow{\alpha_1} \ldots$

ECA
dependency graph



rule A $r_4\ (\square) : r_6 \leftarrow \square$

rule B $r_3\, r_2\ (\square) : r_4 \leftarrow \square$

rule C $r_5\ (\square) : r_6 \leftarrow \square$

rule D $r_1\ (\square) : r_2 \leftarrow \square$

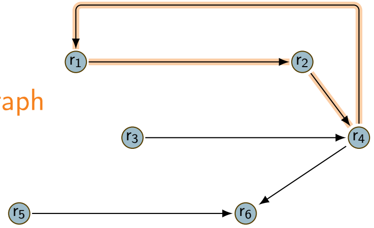
Theorem (AbU stabilization)

If the ECA dependency graph of an AbU system S is acyclic, then S is stabilizing.

Stabilization

The wave semantics may exhibit internal divergence, namely $S \xrightarrow{\alpha_0} S^0 \xrightarrow{\alpha_1} \ldots$

ECA
dependency graph



rule A $r_4\ (\Box) : r_6 \leftarrow \Box\ r_1 \leftarrow \Box$

rule B $r_3\ r_2\ (\Box) : r_4 \leftarrow \Box$

rule C $r_5\ (\Box) : r_6 \leftarrow \Box$

rule D $r_1\ (\Box) : r_2 \leftarrow \Box$

Theorem (AbU stabilization)

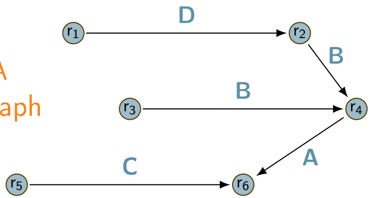
If the ECA dependency graph of an AbU system S is acyclic, then S is stabilizing.

Can we do better? E.g., including (some) loops? (Control theory may be useful here?)

Confluence

The AbU scheduler should not influence the AbU semantics: for all S_1 and S_2 such that $S \rightharpoonup^* S_1$ and $S \rightharpoonup^* S_2$, there exists S' such that $S_1 \rightharpoonup^* S'$ and $S_2 \rightharpoonup^* S'$

labeled ECA dependency graph



rule A $r_4\ (\square) : r_6 \leftarrow \square$

rule B $r_3\ r_2\ (\square) : r_4 \leftarrow \square$

rule C $r_5\ (\square) : r_6 \leftarrow \square$

rule D $r_1\ (\square) : r_2 \leftarrow \square$

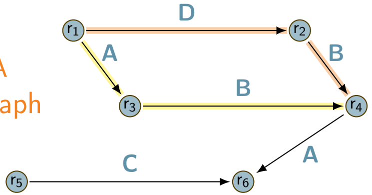
Theorem (AbU confluence)

If for each pair (x, y) of nodes in the labeled ECA dependency graph of an AbU system S we have that $|\text{walks}(x, y)| \leq 1$, then S is confluent.

Confluence

The AbU scheduler should not influence the AbU semantics: for all S_1 and S_2 such that $S \rightharpoonup^* S_1$ and $S \rightharpoonup^* S_2$, there exists S' such that $S_1 \rightharpoonup^* S'$ and $S_2 \rightharpoonup^* S'$

labeled ECA dependency graph



rule A $r_4\ (\square) : r_6 \leftarrow \square\ r_3 \leftarrow \square$

rule B $r_3\ r_2\ (\square) : r_4 \leftarrow \square$

rule C $r_5\ (\square) : r_6 \leftarrow \square$

rule D $r_1\ (\square) : r_2 \leftarrow \square$

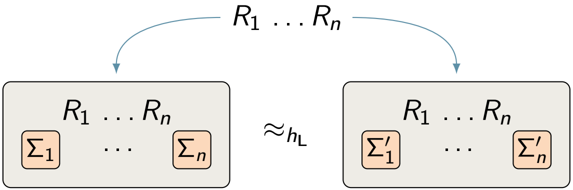
Theorem (AbU confluence)

If for each pair (x, y) of nodes in the labeled ECA dependency graph of an AbU system S we have that $|\text{walks}(x, y)| \leq 1$, then S is confluent.

Security and safety: Behavioral equivalences

Security: protection of confidential data (by means of *noninterference*)

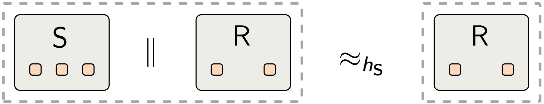
- Assign different *security levels* to resources: e.g. L (public) and H (confidential)
- Security policy: No information flow from H to L allowed
- *Bisimulation* $\approx_{h_L}$ that hides L-level updates



for all L-equivalent states $\Sigma_1 \equiv_L \Sigma'_1 \dots \Sigma_n \equiv_L \Sigma'_n$

Safety: prevention of unintended interactions

- The systems S and R are known to be safe
- Is the ensemble S ∥ R still safe?
- Bisimulation $\approx_{hs}$ that hides the updates of S

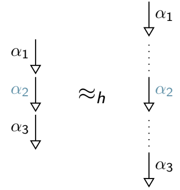


S does not interact with, or is transparent for, R

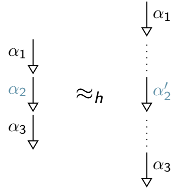
Hiding bisimulation

- Weak bisimulation hiding labels not related to the requirements
- Parametric on a function h making non-observable labels α such that $h(\alpha) = \diamond$

if $h(\alpha_2) \neq \diamond$



if $h(\alpha_2) = \diamond$



with $h(\alpha'_2) = \diamond$

Security h_L hides:

- discovery labels
- execution labels with H resources

Safety h_S hides:

- discovery labels
- execution labels produced by S

Security: verification mechanism

Algorithm `IFRules` for computing information flows:

- Compute a constancy analysis for conditions and expressions
- Check explicit flows for the default action
- Check explicit and implicit flows for the task action

Theorem (Soundness for Security)

If $\texttt{IFRules}(R) = $ *false then* R *is noninterferent, namely* R *is secure.*

Safety: verification mechanism

- Compute sinks: resources that rules may update
- Compute sources: resources that may influence rules behavior

Check that the sinks of S does not overlap with the sources of R

LHS (default), LHS (task):

$$\{y_1, \ldots, y_n\} \cup \{y_{n+1}, \ldots, y_{n+m}\}$$

event, default, task:

$$\boxed{x_1 \ldots x_k} > \boxed{y_1 \leftarrow \varepsilon_1 \ldots y_n \leftarrow \varepsilon_n} \, , \, \boxed{(\mathrm{cnd}) : y_{n+1} \leftarrow \varepsilon_{n+1} \ldots y_{n+m} \leftarrow \varepsilon_{n+m}}$$

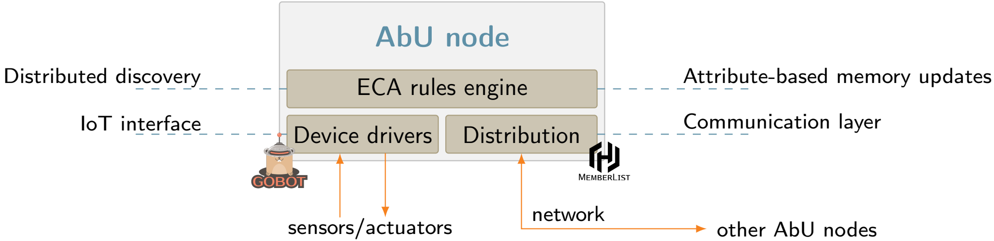
RHS (default), RHS (task):

$$\{x_1, \ldots, x_k\} \cup \mathrm{Vars}(\varepsilon_1) \cup \ldots \cup \mathrm{Vars}(\varepsilon_n) \cup \mathrm{Vars}(\mathrm{cnd}) \cup \mathrm{Vars}(\varepsilon_{n+1}) \cup \ldots \cup \mathrm{Vars}(\varepsilon_{n+m})$$

Theorem (Soundness for Safety)

If $\mathrm{sinks}(S) \cap \mathrm{sources}(R) = \varnothing$ *then* S *is transparent for* R.

A (modular) distributed implementation



- ECA rules engine module: AbU semantics
- Device drivers module: abstraction of physical resources
- Distribution module: abstraction of send/receive and cluster nodes join/leave

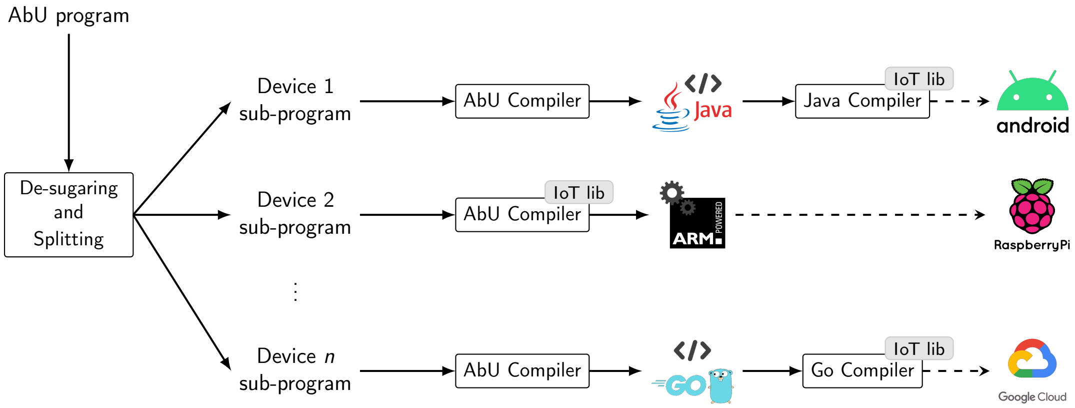
AbU-lang: a Domain Specific Language for the IoT

```
# AbU devices definition.

hvac : "An HVAC control system" {
   physical output boolean heating = false
   physical output boolean condit = false
   logical integer temp = 0
   logical integer humidity = 0
   physical input boolean airButton
   logical string node = "hvac"
   where not (condit and heating == true)
} has cool warm dry stopAir

tempSens : "A temperature sensor" {
   physical input integer temp
   logical string node = "tempSens"
} has notifyTemp

humSens : "A humidity sensor" {
   physical input integer humidity
   logical string node = "humSens"
} has notifyHum
```

```
\%
   AbU (ECA) rules definition.
   Rules can be referenced by multiple devices.
%\

rule cool on temp
   for (this.temp < 18) do this.heating = true

rule warm on temp
   for (this.temp > 27) do this.heating = false

rule dry on humidity; temp
   for (this.temp * 0.14 < this.humidity)
      do this.condit = true

rule stopAir on airButton
   for (this.airButton) do this.condit = false

rule notifyTemp on temp
   for all (ext.node == "hvac")
      do ext.temp = this.temp
```

[M. Pasqua, M. Comuzzo, MM., IEEE Access 2022]

AbU-lang Programs Compilation Cycle



Conclusion

AbU: attribute-based memory updates programming

- Simple to use (ECA paradigm)
- Suitable for the IoT domain
- Strongly decentralized
- Local nodes coordination
- Supports several verification techniques

Thanks for the attention

- M Miculan, M Pasqua, *A Calculus for Attribute-based Memory Updates*, Proc. ICTAC 2021 - LNCS 12819;
- M Pasqua, M Miculan, *On the Security and Safety of AbU Systems*, International Conference on Software Engineering and Formal Methods, LNCS 13085, 2021.
- M Pasqua, M Miculan, *Distributed Programming of Smart Systems with Event-Condition-Action Rules*, ICTCS 2022: 201-206
- M Pasqua, M Comuzzo, M Miculan, *The AbU Language: IoT Distributed Programming Made Easy*, IEEE Access 10: 132763-132776 (2022)
- M Pasqua, M Miculan, *AbU: A calculus for distributed event-driven programming with attribute-based interaction*. TCS 958: 113841 (2023)
- `https://github.com/abu-lang`