# Securing Data and Critical Decisions in IoT Systems: a Control Flow Analysis Approach

**Chiara Bodei**[1], P. Degano[1,2], G-L. Ferrari[1], L. Galletta[2]

[1] Dip. di Informatica, Università di Pisa, Italy and
[2] IMT School for Advanced Studies, Lucca, Italy

# The presentation

- Static Analysis, originally developed to enhance interpreter and compiled code efficiency, has gained attention as it can validate crucial system behaviour aspects
- As far as Concurrency is concerned, it has been used to analyse process algebras for a variety of communication and mobility paradigms
- The current presentation deals with the use of process algebras and static analysis techniques for modelling IoT systems

# Outline

# The Big Picture

To analyse IoT-systems, implemented in appropriate programming languages to evaluate specific properties of interest, we can

- directly analyze the programming language itself, similar to how optimising compilers operate
- extract the system's behaviour from the programming language, in a process calculus setting, and analyse it to determine the desired properties

# CFA and Process Algebras

Control Flow Analysis (CFA) is a static analysis technique that has been applied to several process algebras (PAs):

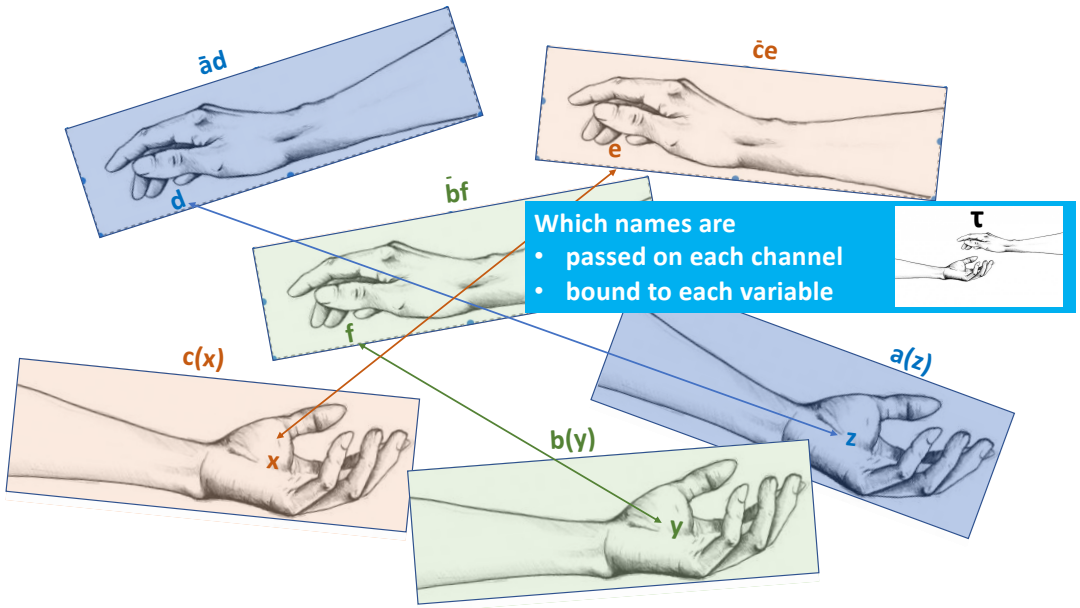
- $\pi$-calculus
- Mobile Ambients
- Calculi for modelling biological systems
- Security-oriented process algebras
- IoT Systems

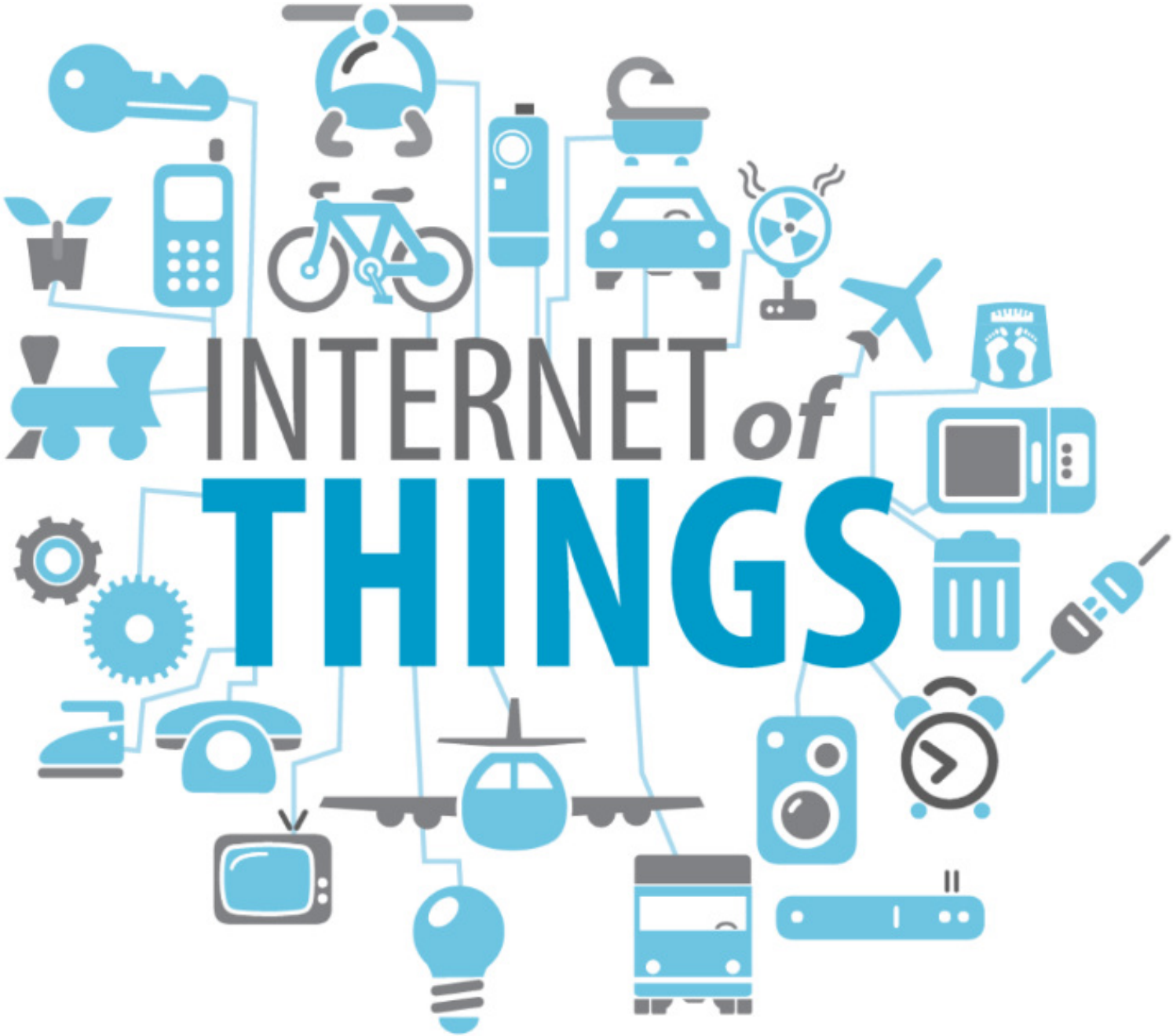
# CFA & PAs: intuition

CFA in PAs over-approximates the possible communications

Example: In the $\pi$-calculus process $\overline{a}d.P \| a(z).Q$, it predicts that the value $d$

- can be sent on channel $a$ and that
- can be bound to the variable $z$




INTERNET of
THINGS

# IoT Challenges

Designing & implementing IoT systems is hard

- Smart objects are heterogeneous
  - Many hw, sw, protocols, etc.
- Cyberphysical and dynamic systems
- Security & privacy
  - Highly critical but difficult to achieve

# Our goal

A unifying framework based on formal methods

- Specify IoT systems
- Abstractly reason on their properties
  - Correctness
  - Security and safety
  - Risks and costs

Our proposal: relying on the process algebra IoT-LySA

# IoT-LySA: a process algebra for IoT

Derived from LySA (a security-oriented process algebra), IoT-LySA is characterised by

- Hierarchical structure
  - Multiple nodes
  - Sensors
  - Actuators
  - Group communication among nodes
  - Local communication à la Linda
  - Cryptography
- Formal Semantics
- Properties of interest here
  - Tracking data analysis

# Tracking data analysis

A Control Flow Analysis (CFA) to safely over-approximate

- Interactions among nodes
- How data spread from sensors to network
- How data are manipulated
- On which data and logical conditions critical actuator decisions depend

We actually track abstract data, by abstracting from their concrete values

# Why static analysis

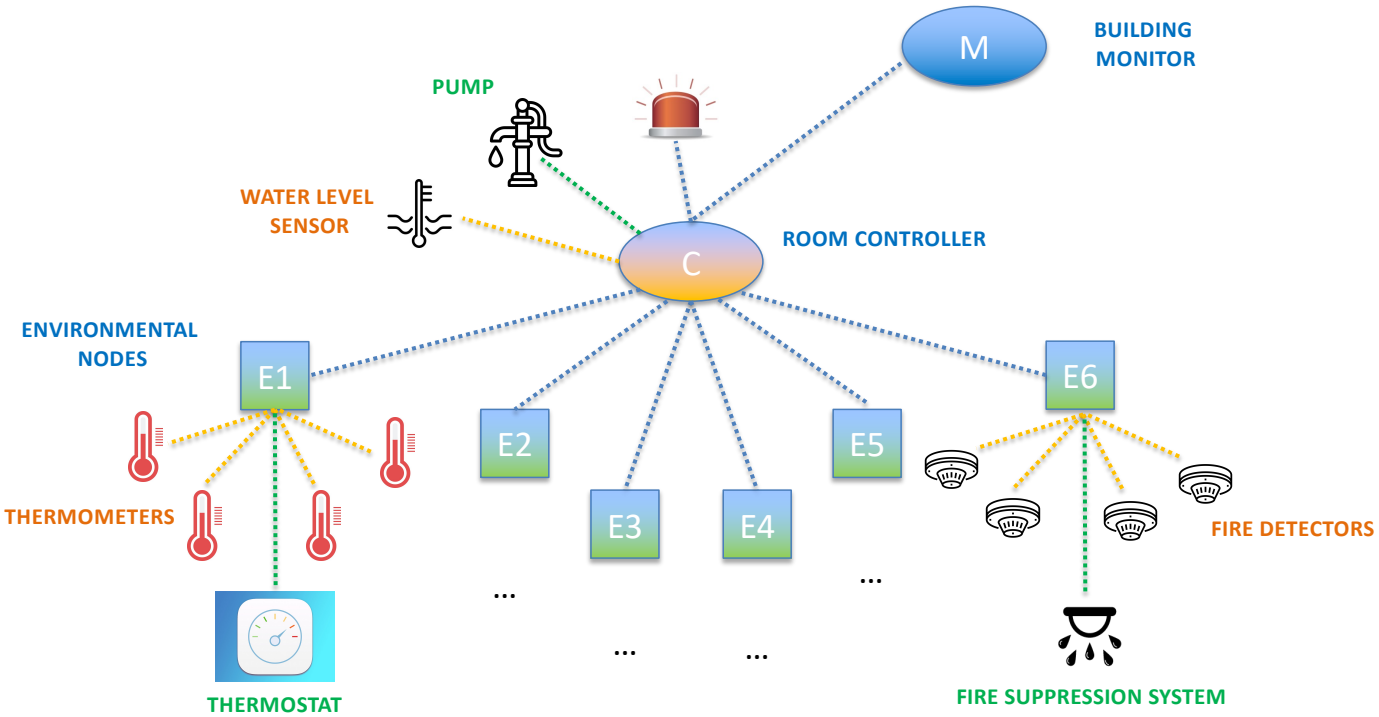
- Help designers to reason about possible flaws in the early stages of development
- Detect possible risky actions
- Give designers hints about possible countermeasures to adopt at runtime

Example: What happens if an attacker can tamper with a sensor? ("WHAT IF" reasoning)

# Our present analysis

- Goal: Determine whether our design is robust against data alteration. Do critical computations depend on possibly altered data?
- How: Over-approximate from which data sources computations depend. Do they depend on possible tamperable (security issue) or frail (robustness issue) data sources?

# Indoor environmental monitor scenario



# IoT-LySA specification

$$
\begin{aligned}
M &= \ell_M : [\Sigma_M \parallel P_M \parallel B_M] \\
C &= \ell_C : [\Sigma_C \parallel P_C \parallel S_{water} \parallel A_{pump} \parallel B_C] \\
E_1 &= \ell_T : [\Sigma_T \parallel P_T \parallel T_1 \parallel \ldots \parallel T_4 \parallel A_{Thermostat} \parallel B_T] \\
E_6 &= \ell_F : [\Sigma_F \parallel P_F \parallel F_1 \parallel \ldots \parallel F_4 \parallel A_{FireSuppression} \parallel B_F]
\end{aligned}
$$

- $\ell_C$ is the node identifier of $C$
- $\Sigma_C$ is the local shared store, $P_C$ is the control process, $S_{water}$ is the sensor, and $A_{pump}$ the actuator, and $B_C$ represents other components

# Syntax of IoT-LySA (1)

## Nodes

| | | |
|---|---|---|
| $N ::=$ | $0$ | nil node |
| | $\ell : [B]$ | single node |
| | $N_1 \mid N_2$ | composition |

## Node body

| | | |
|---|---|---|
| $B ::=$ | $\Sigma_\ell$ | store |
| | $S$ | sensors |
| | $A$ | actuators |
| | $P$ | processes |
| | $B \parallel B$ | composition |

## Sensors

| | | |
|---|---|---|
| $S ::=$ | $0$ | nil |
| | $\tau.S$ | internal action |
| | $i := v.S$ | storing $v$ |
| | $h$ | iteration var. |
| | $\mu h.S$ | iteration |

## Actuators

| | | |
|---|---|---|
| $A ::=$ | $0$ | nil |
| | $\tau.A$ | internal action |
| | $(\!\| j, \Gamma \|\!).A$ | command |
| | $\gamma.A$ | action |
| | $h$ | iteration var. |
| | $\mu h.A$ | iteration |

# Syntax of IoT-LySA (2)

## Processes

| | | |
|---|---|---|
| $P ::=$ | $\langle\langle E_1, \cdots, E_k \rangle\rangle \triangleright L.\, P$ | multi-output |
| | $(E_1, \cdots, E_j;\, x_{j+1}^{a_{j+1}}, \cdots, x_k^{a_k}).\, P$ | input (with matching) |
| | $x^a := E.\, P$ | assignment |
| | $\langle E, j, \gamma \rangle.\, P$ | command to actuator $j$ |
| | $\ldots$ | nil, conditional, iteration |

## Terms

| | | |
|---|---|---|
| $E ::=$ | $M^a$ | annotated terms |
| $M ::=$ | $v$ | values |
| | $i$ | sensor location |
| | $x$ | variables |
| | $f(E_1, \cdots, E_n)$ | function on data |

# IoT-LySA specification (cont.)

## Environmental node for fire detectors

$$
\begin{aligned}
P_F \quad = \quad & \mu h.(z_{f1} := 1_{f1}). \cdots .(z_{f4} := 4_{f4}). \\
& \langle\!\langle E_6, z_{f1}, z_{f2}, z_{f3}, z_{f4} \rangle\!\rangle \triangleright \{\ell_C\}. \\
& \ldots \\
& (C; z_{check_{Alarm}}, z_{alarm}).\langle z_{alarm}, A_{FireSuppression}, \mathrm{Go}_{\mathrm{fs}}(z_{alarm}) \rangle.h
\end{aligned}
$$

The control process $P$:

1. stores the fire detection data collected by sensors of node $E_6$
2. sends them to the Room Controller
3. waits for the actuation decision and produces the suitable trigger command for the Fire Suppression System

# IoT-LySA specification (cont.)

## Room controllers inside: fire detection

$$
\begin{aligned}
P_C \quad = \quad & \ldots check_{TA} := gt(avg_T, T_{anomaly}). \\
& dt_1 = dt(w_{f1}), \ldots \\
& check_{FA} := or(dt_1, \cdots, dt_4). \\
& check_{alarm} := and(check_{water}, and(check_{TA}, check_{FA})) \\
& \langle check_{alarm}, Pump, \mathrm{PumpReg}(alarm) \rangle. \\
& \langle\langle C, check_{alarm}, alarm \rangle\rangle \triangleright \{\ell_F, \ell_M\}.h
\end{aligned}
$$

Once received sensed data by the environmental nodes, $P_C$

1. checks the temperature average and the data sensed by the fire detection sensors
2. IF (the water level is not right) AND ((the average temperature in the room is over the $T_F$) OR (at least a sensor detects fire)) THEN a fire alarm must arise

# Analysis overview

## Questions

- How are sensor data manipulated?
- Where do sensor data flow?
- on which data the trigger of an actuator may depend on?

## The analysis components

The CFA computes an over-approximation to

- data a node may store ($\hat{\Sigma}$)
- messages a node may receive ($\kappa$)
- the values that may be associated to labelled terms ($\Theta$)
- which values may affect the condition triggering an action ($\gamma$)

# Analysis overview (cont.)

The CFA analysis is specified as a Flow Logic with judgements

1. $(\hat{\Sigma}, \Theta) \models_\ell M^a$ for expressions
2. $(\hat{\Sigma}, \kappa, \Theta, \alpha) \models_\ell B$ for node internals
3. $(\hat{\Sigma}, \kappa, \Theta, \alpha) \models N$ for nodes

The CFA enjoys the classical results of correctness and existence of its approximations

# Analysis overview (cont.)

- Our analysis over-approximates all possible data flows, abstracting from their concrete values
- CFA abstract values $\hat{v}$ record the type of data, their origin (not their numerical consistency), the way they are exchanged and the functions used to aggregate and process them

## Abstract values $\hat{v}$

| | | |
|---|---|---|
| $\hat{v} ::=$ | $(i, a)$ | data from sensor $i$ |
| | $(v, a)$ | value for clear data |
| | $(f(\hat{v}_1, \cdots, \hat{v}_n), a)$ | value for aggregated data |
| | $(\top, a)$ | term of depth greater than $d$ |

- Abstract values that correspond to critical conditions, "symbolically" encode the logical structure and the supply chain of the data used for decision-making

# Anatomy of critical decision

- Our CFA supports us in identifying the functional and logic components of the abstract values, on which critical decisions are made
- We analyse CFA abstract values related to critical decisions in terms of both security and robustness of the sensors and the resulting reliability of their values
- Something can go wrong and alter the correct behaviour of an actuator, because of an accidental failure or of an intentional malicious attack
- Several analyses are possible, in a sort of WHAT-IF fashion

# Analysis in our scenario

## Focus on the actuation decision

The actuation decision is based here on the evaluation of $check_{alarm}$ and therefore on the sensed values, their aggregations, as well as on the reliability of the nodes where data are collected, processed and propagated

## Abstract value

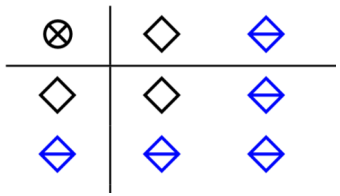
CFA can provide the corresponding composite abstract value

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_t, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$$

# Ex-post Taint analysis

Sensors can be tampered and data can be unreliable
Starting from the CFA results, we adopt an ex-post taint analysis

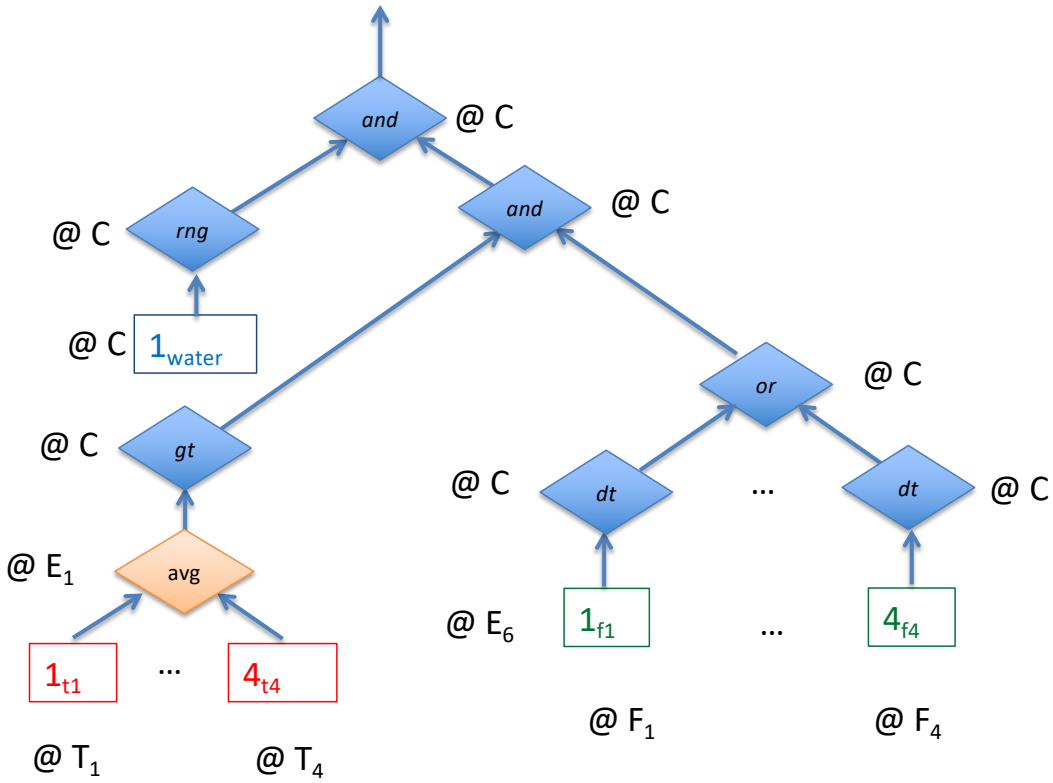
- partitioning abstract sensor data (our sources) in tainted/frail and untainted/unfrail: the ones coming from possibly compromised sensors, and
- applying suitable propagation policies, directly on the abstract values



This allows us to understand whether the abstract values of interest (our sinks) are tainted or not and to intervene with possible sanitisations

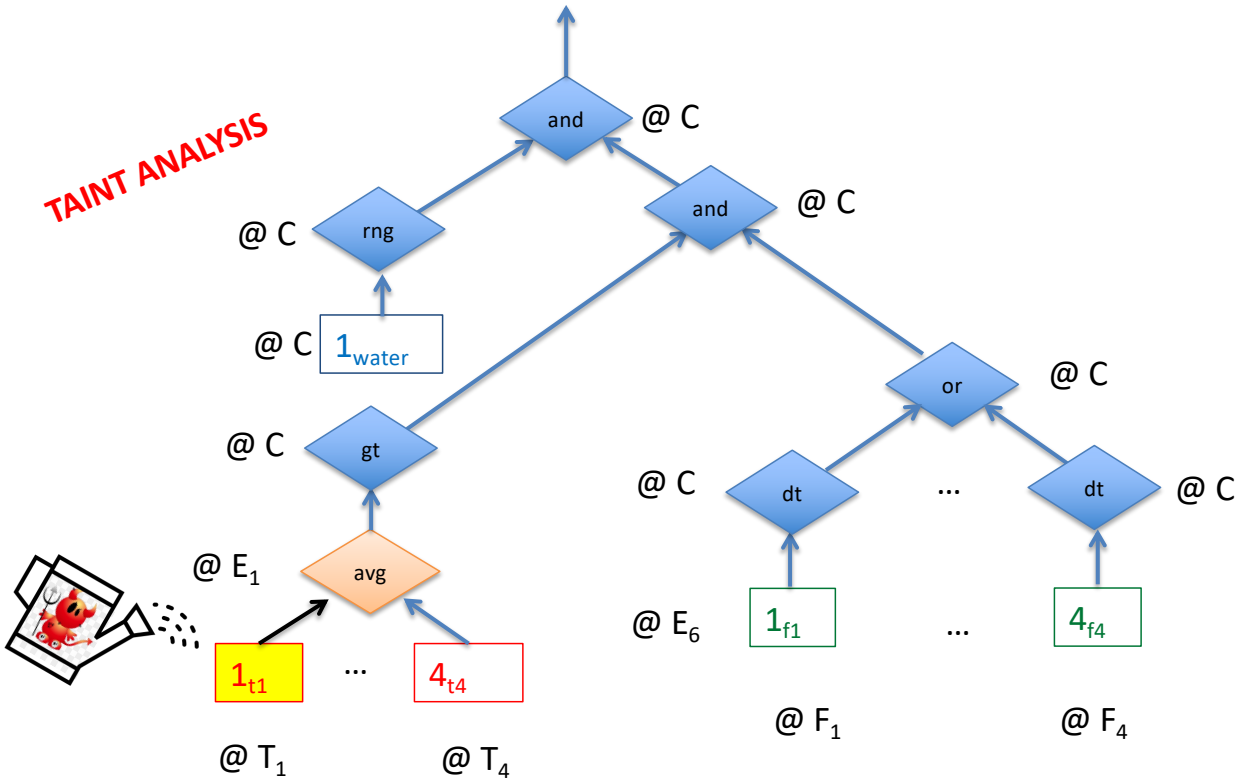
# Pictorial view of our abstract term

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, \ldots, 4_{t4}), T_{anomaly}), or(dt(1_{f1}), \ldots, dt(4_{f4}))))$$



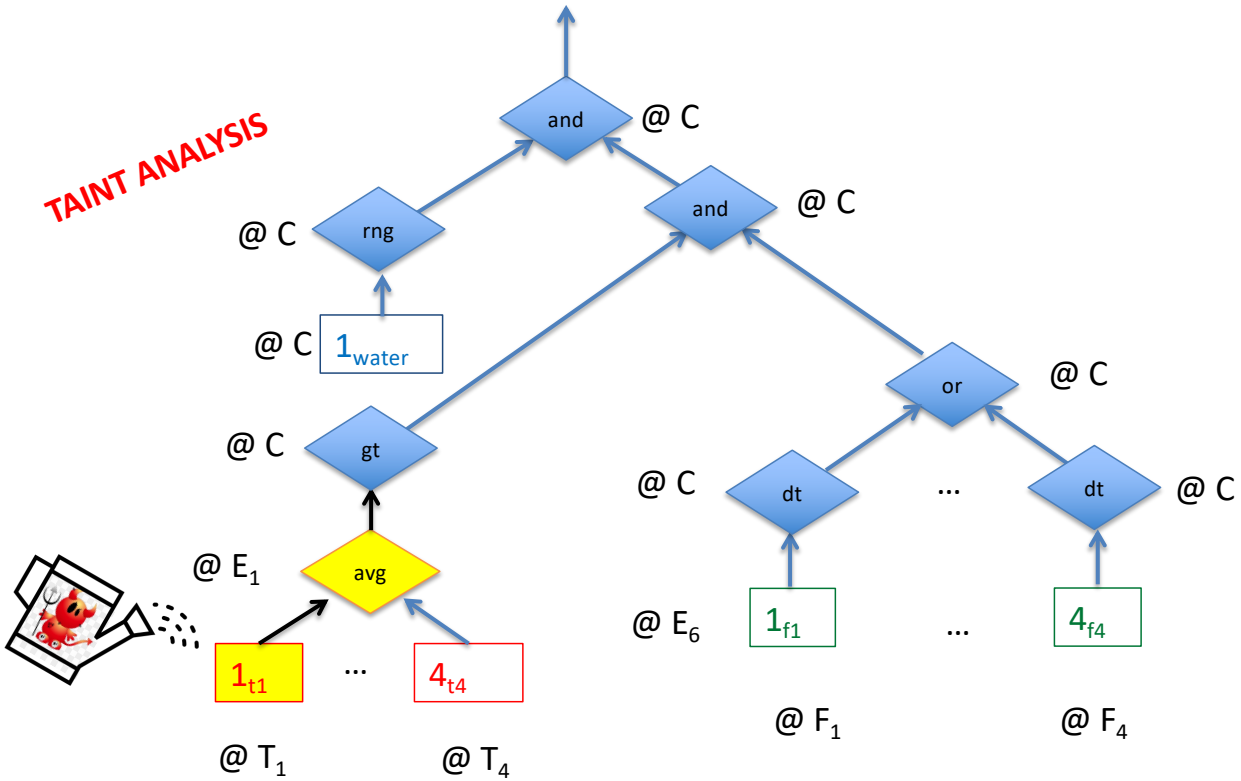
# Taint analysis: tagging sources

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$$



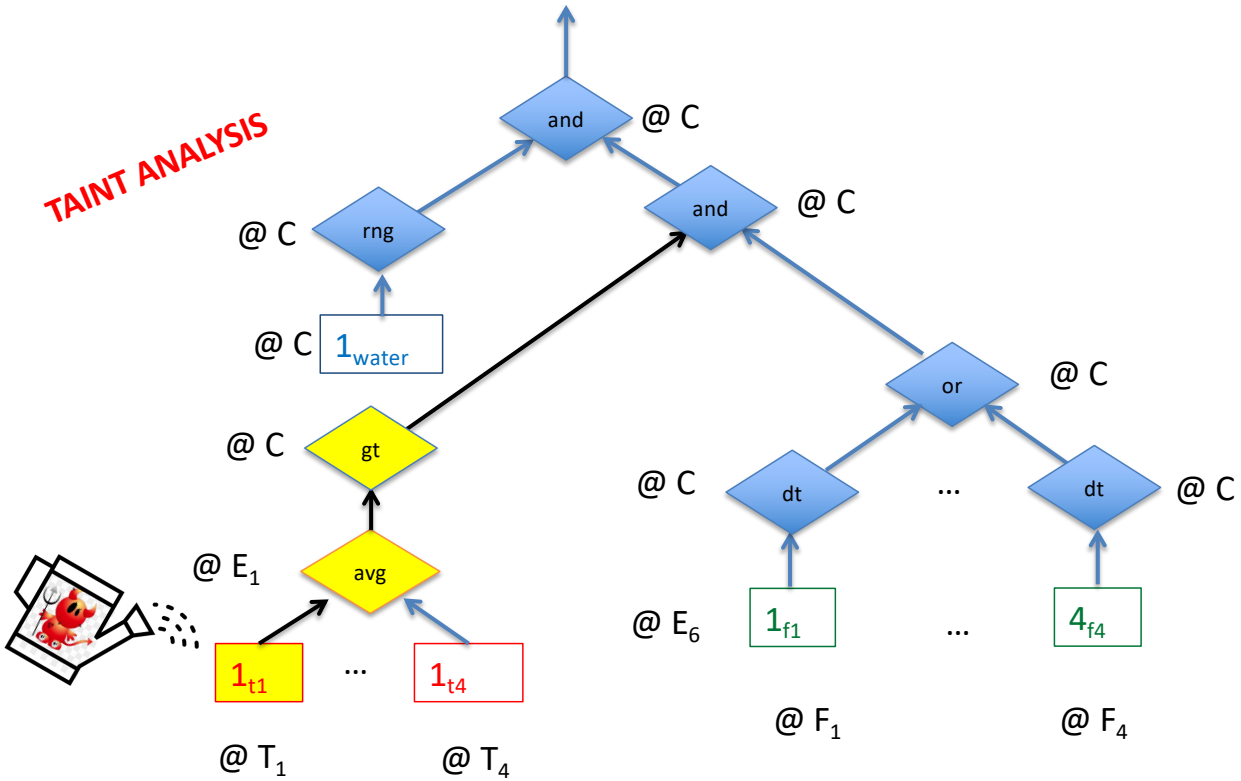
# Taint analysis: propagation

$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$



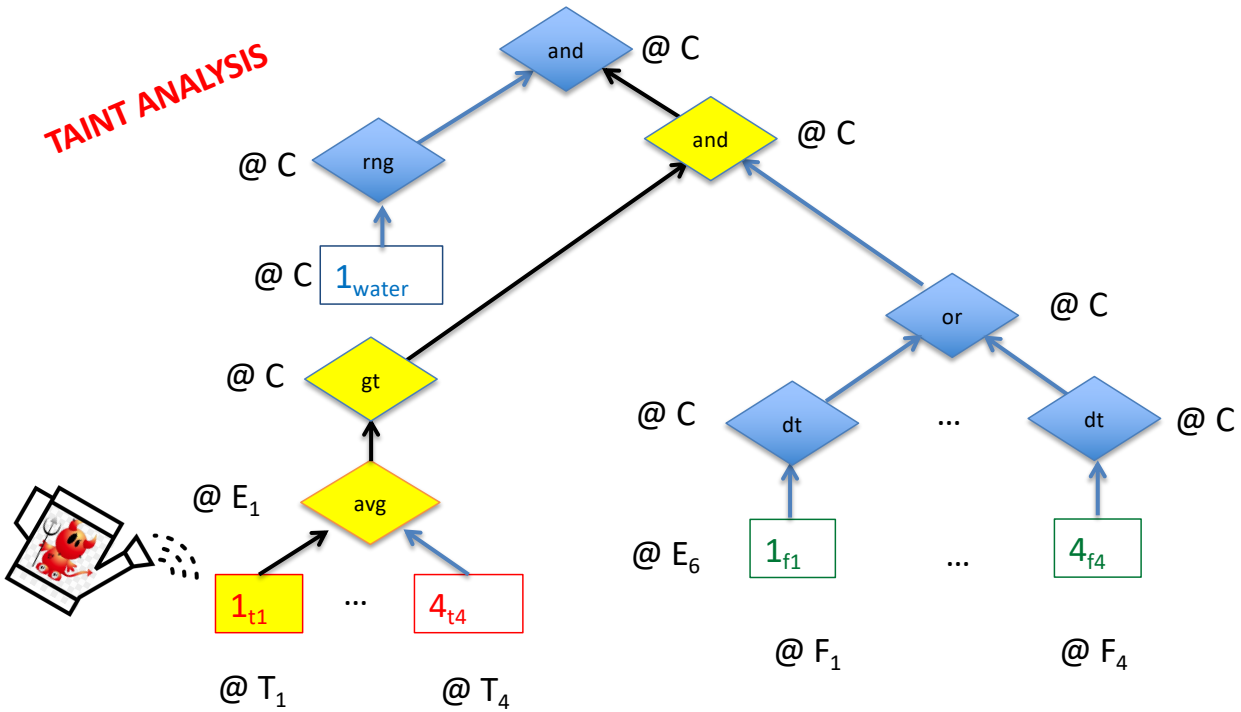
# Taint analysis: propagation (cont.)

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$$



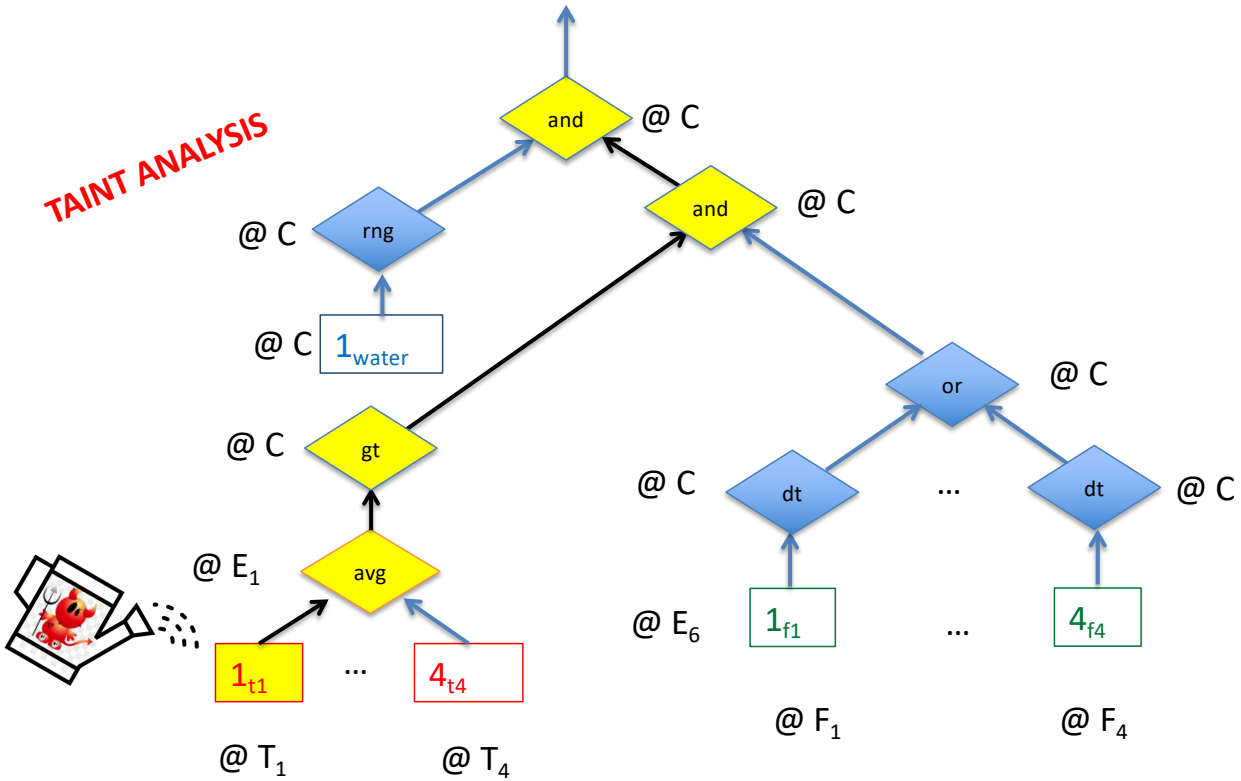
# Taint analysis: propagation (cont.)

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}),\ and(gt(avg(1_{t1}, \ldots, 4_{t4}), T_{anomaly}),\ or(dt(1_{f1}), \ldots, dt(4_{f4}))))$$



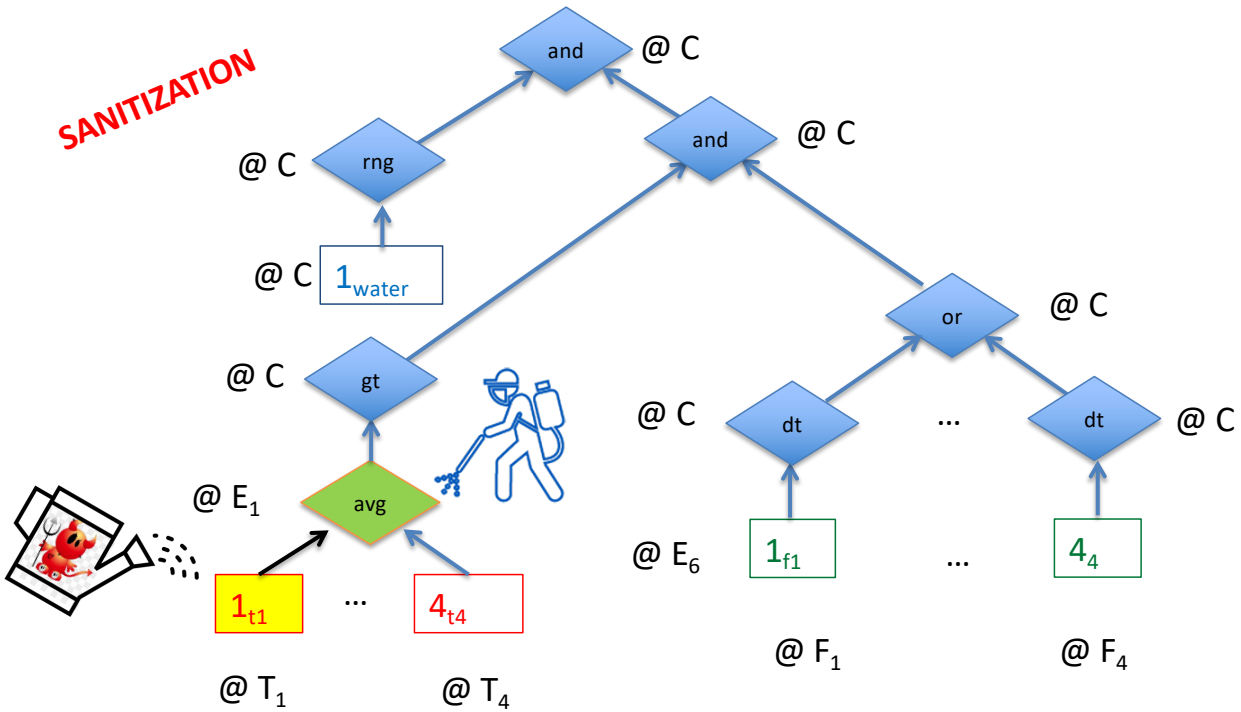
# Taint analysis: propagation (cont.)

$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$



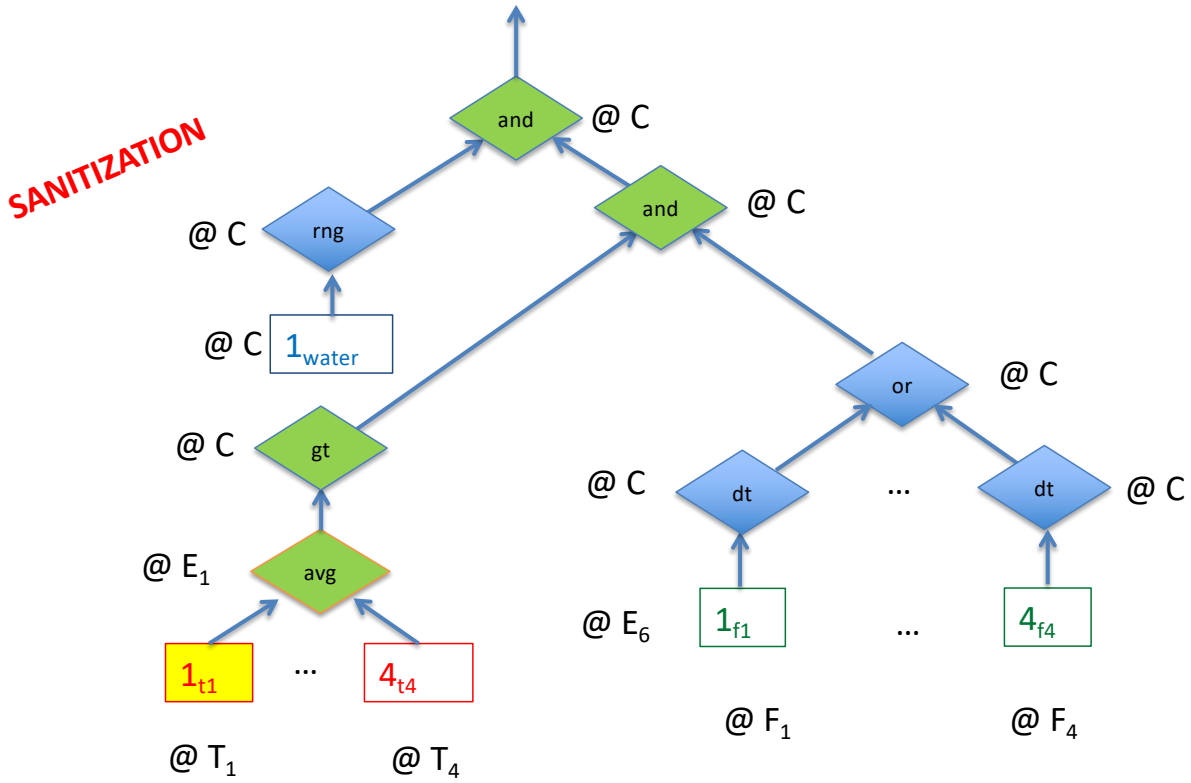
# Sanitisation

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$$



# Sanitisation (cont.)

$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$

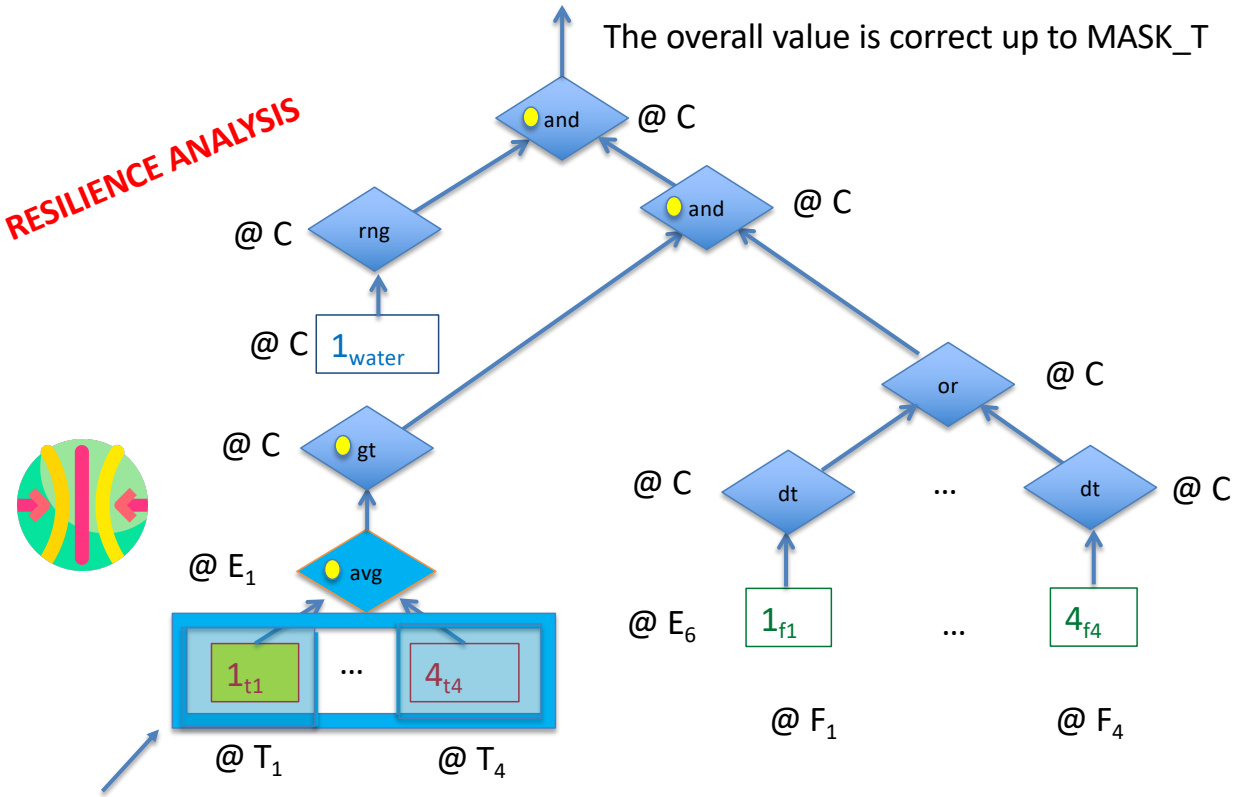


# Resilience analysis

- Given an aggregation function, how many values must be correct for giving the right answer?
- We could aim at a certain level of reliability of actuations even in the presence of unreliable data
- We can then embed some quality predicates inside our abstract terms

# Resilience analysis (cont.)

$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$




MASK_T: For ***avg: at least one btw*** the 1° and the 4° values must be correct, the other 2 must all be correct: $avg(\&_{forall}(\&_{exists}(1_{t1}, 4_{t4}), 2_{t2}, 3_{t3}))$

# Reasoning on possible impairment

- Finally, we associate abstract sensor data with numerical scores.
  - For security, scores are proportional to the attacker's efforts to compromise the corresponding physical devices
  - For robustness, the scores represent the reliability of the sensors, e.g./due to battery issues
- We can thus estimate and compare the different risks of impairment of each subset of sensors, and determine where it is worth taking countermeasures.

# Reasoning on possible impairment (cont.)

The graph describing $\hat{v}_{firealarm}$ is reminiscent of attack/fault trees
To impair the alarm trigger, the attacker can force just one AND branch to `false`, by tampering the sensors:

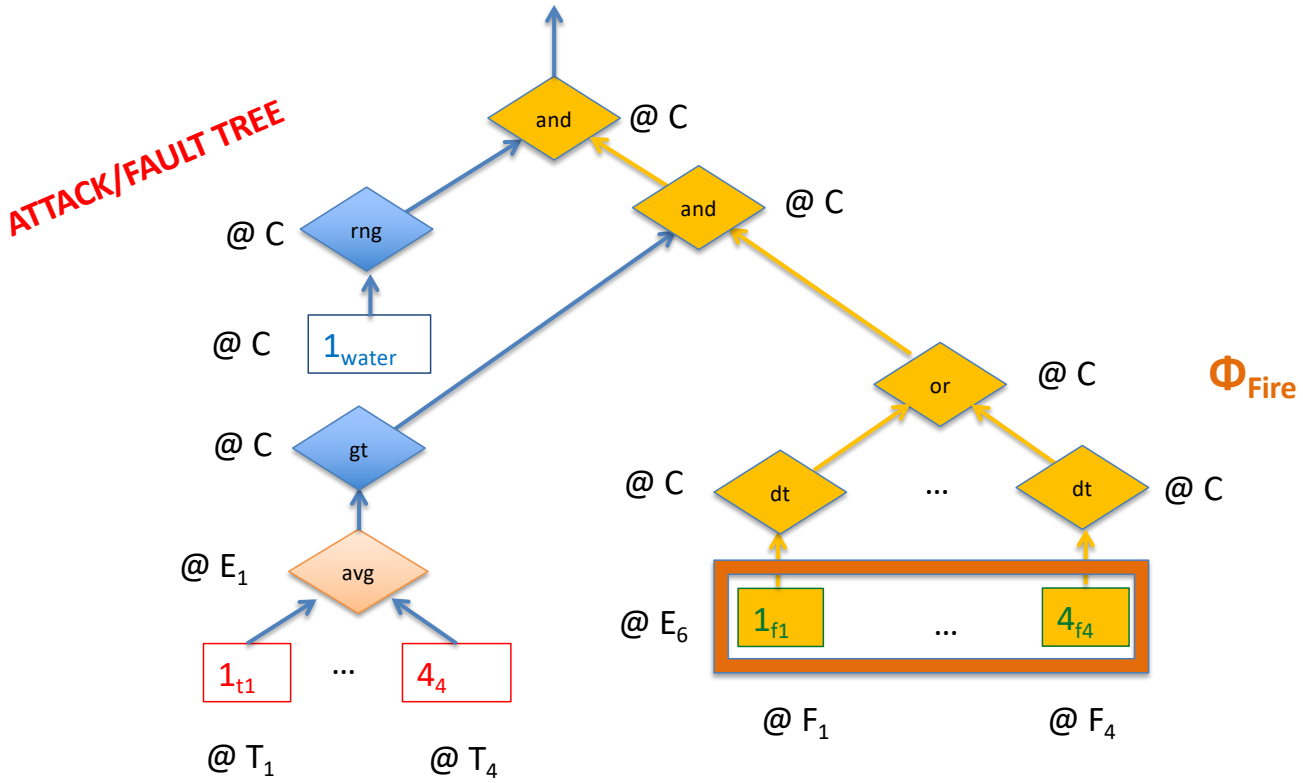
- $S_{water}$ that measure the water level **or**
- those for temperature $S_{Ti}$ **or**
- the ones for fire detection $S_{Fi}$

Each of these attacks requires different efforts for the attacker, that can be associated to suitable security scores $\phi$

We can compare the different attack strategies and devise which sensors need more protection, e.g. temperature sensors

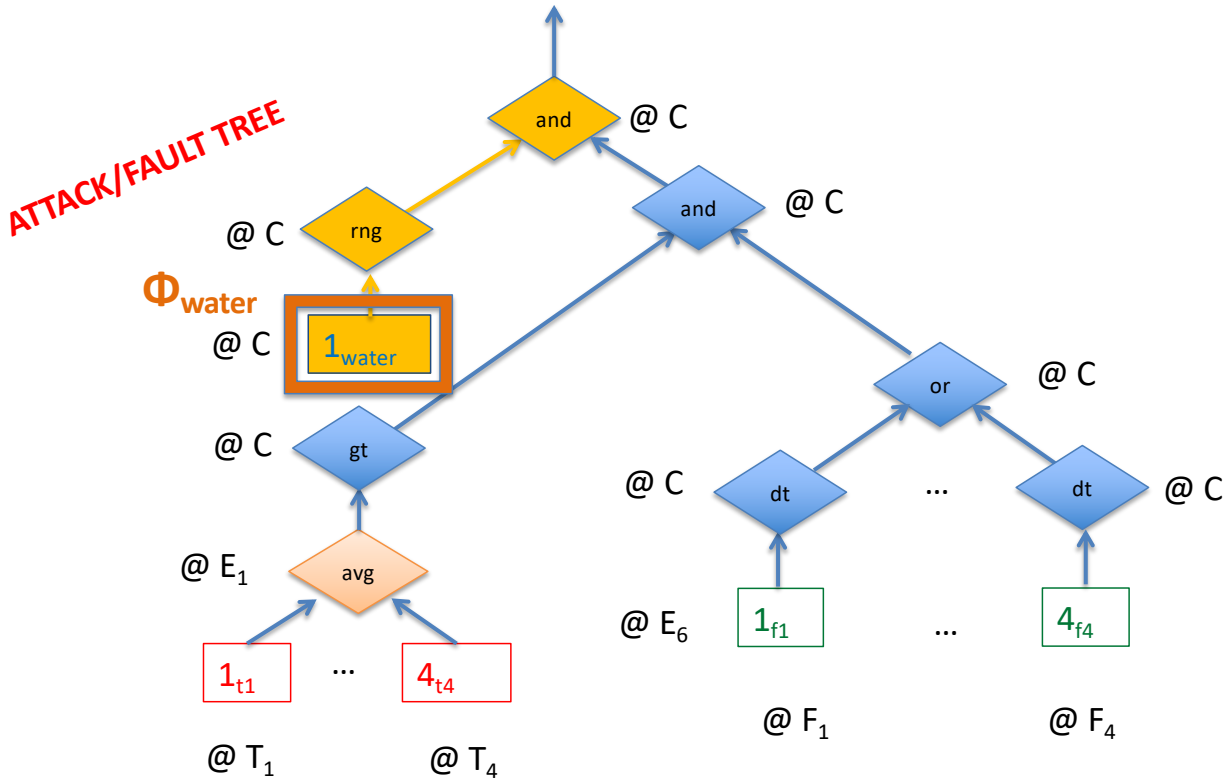
# Attack analysis

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, \ldots, 4_{t4}), T_{anomaly}), or(dt(1_{f1}), \ldots, dt(4_{f4}))))$$

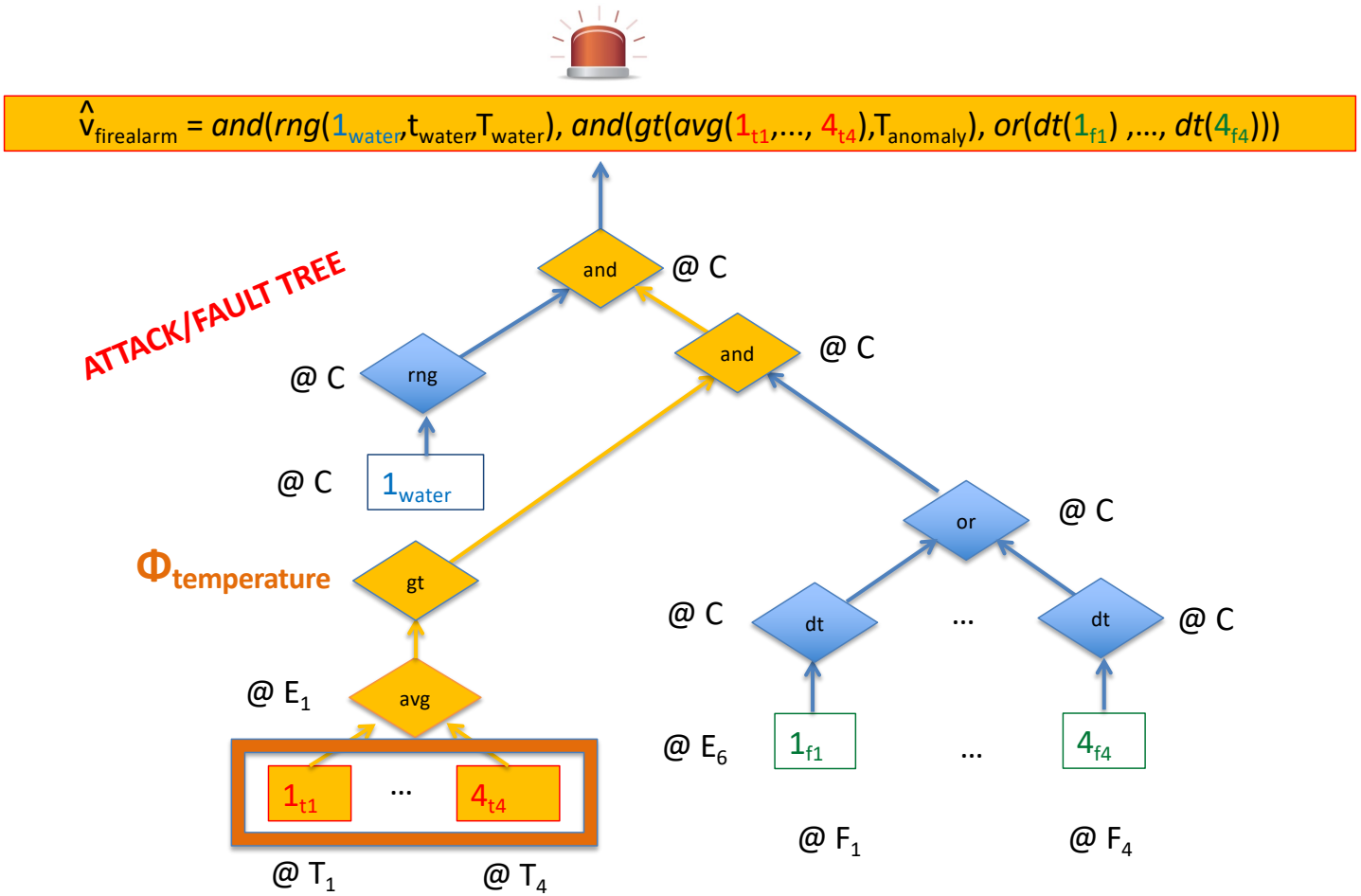


# Attack analysis (cont.)

$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, ..., 4_{t4}), T_{anomaly}), or(dt(1_{f1}), ..., dt(4_{f4}))))$

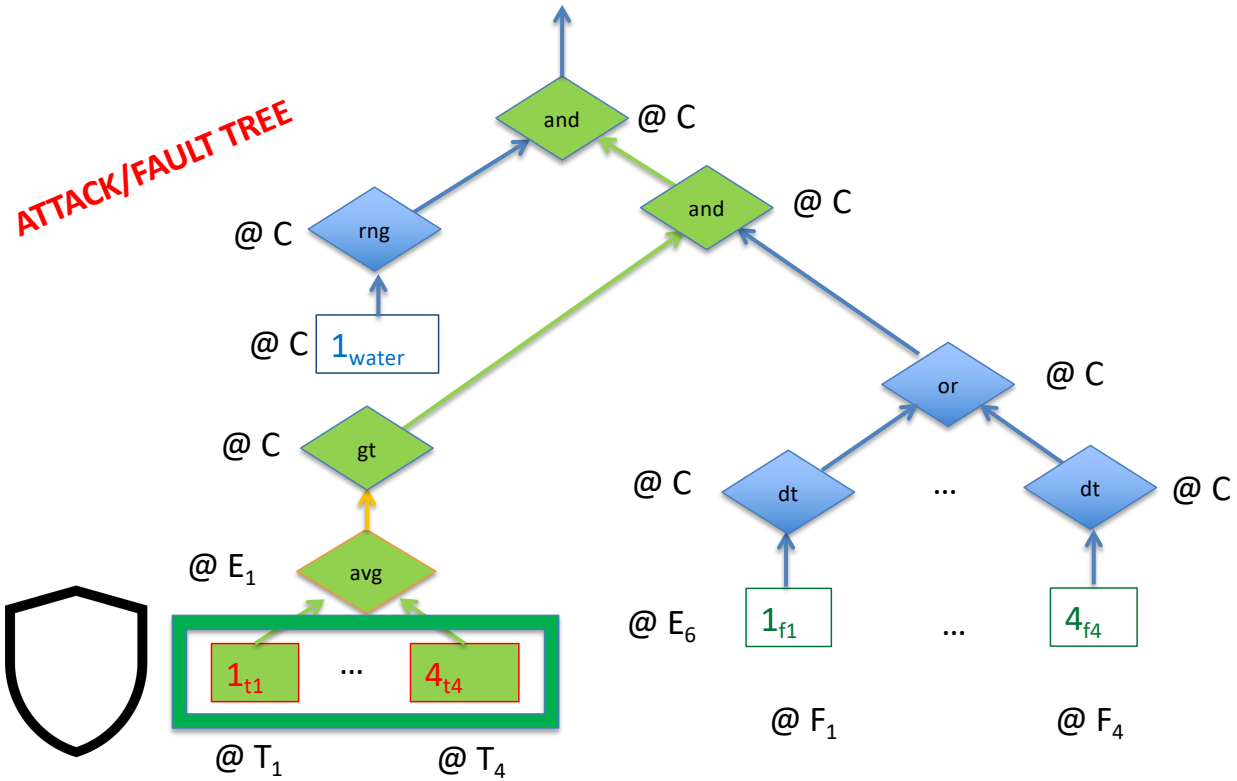


# Attack analysis (cont.)



# Attack analysis: targeted protection

$$\hat{v}_{firealarm} = and(rng(1_{water}, t_{water}, T_{water}), and(gt(avg(1_{t1}, \ldots, 4_{t4}), T_{anomaly}), or(dt(1_{f1}), \ldots, dt(4_{f4}))))$$



# Conclusions

- IoT-LySa to specify IoT systems
- CFA for data tracking
- Reasoning on the supply chain of data that are used to take critical decisions.
  - Identification of the more critical sensor nodes and of the places where security countermeasures are more crucial
  - Help in the reduction of the risk of impairing some critical actuations
- A single formal framework for several ex-post parametric investigations on analysis results that do not require the CFA to be re-computed

# Open problems

- Further reasoning on the quality and reliability of data
- Further formalisation of the logics of conditions
- Different kinds of metrics
- Explicit causality model of data dependencies

# Bibliography


- C. Bodei, P. Degano, G-L. Ferrari, L. Galletta: *Where Do Your IoT Ingredients Come From?* COORDINATION 2016: 35-50
- C. Bodei, P. Degano, G-L. Ferrari, L. Galletta: *Tracing where IoT data are collected and aggregated*. Log. Methods Comput. Sci. 13(3) (2017)
- C. Bodei, L. Galletta: *Analysing the Provenance of IoT Data*. ICISSP (Revised Selected Papers) 2019: 358-381
- C. Bodei, P. Degano, G-L. Ferrari, L. Galletta: *Modelling and analysing IoT systems*. J. Parallel Distributed Comput. 157: 233-242 (2021)
- C. Bodei, P. Degano, G-L. Ferrari, L. Galletta: '*Risk estimation in IoT systems*. To appear in Challenges of Software Verification, Intelligent Systems Reference Library 238, 2023.

# Thanks

**THANK YOU FOR YOUR ATTENTION**