Protecting Smart Homes from Unintended Application Actions

Aqsa Kashaf, Vyas Sekar, Yuvraj Agarwal
Carnegie Mellon University

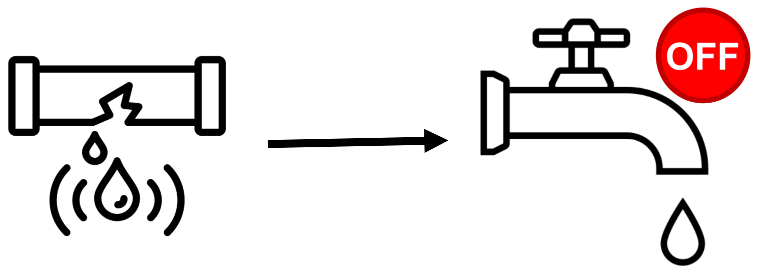
Let’s make a smart home…

IFTTT
SmartThings
Temperature
Bedroom
Entrance Hall
Office
Kitchen
Sensors

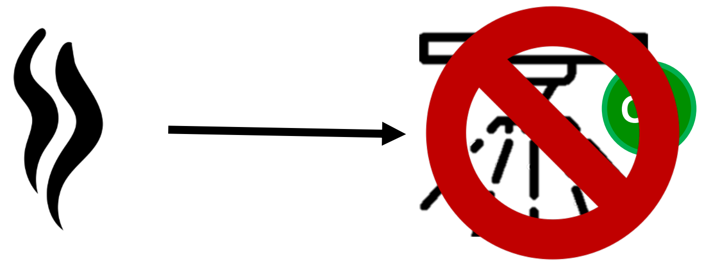
Let’s relax and hope that everything works!

Few moments later, the user's house is on fire…

Leak Detector App



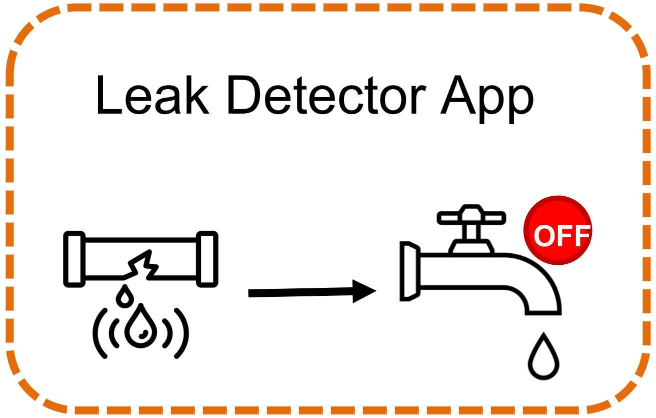
Smoke Detector App

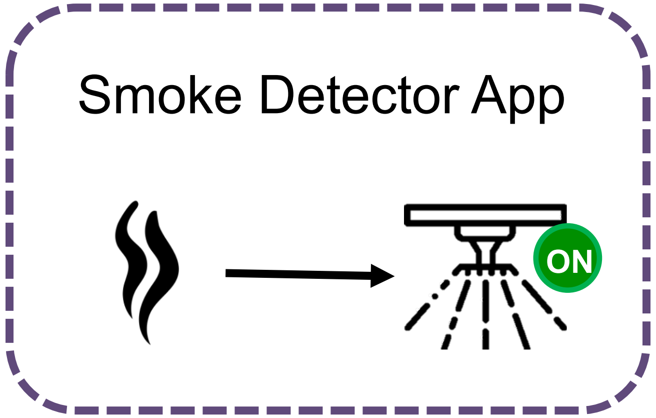


Apps can cause unintended actions in a smart home

Blocked Action Violation

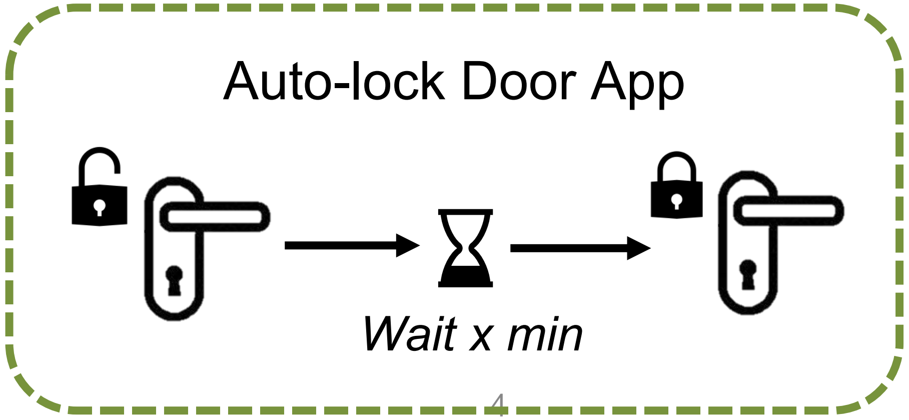
- Inter-app Interactions





- Miss-configuration

Deadline Violation



We formally verify if a set of apps result in an unintended action

“Given a set of apps deployed in a smart home, a set of devices, and a safety intent, identify app configurations which satisfy the given safety intent.”

Challenge 1: State of the art smart apps are complex

Smart Apps are event-driven programs

Devices

User-configurable Inputs

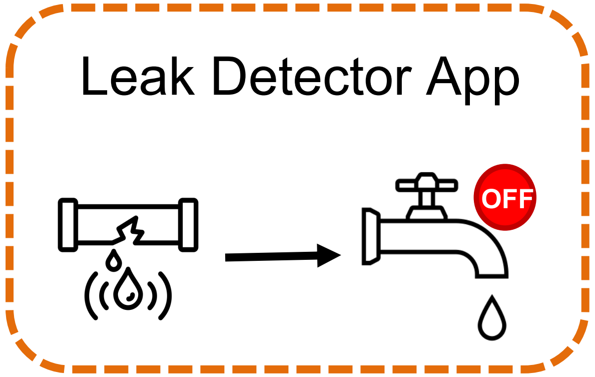
```
preferences
    input "sensor", "capability.waterSensor", input "d", "duration"
    input  "valve",  "capability.valve"
def installed()
    subscribe(sensor, "water", waterHandler)
def waterHandler(evt)
    if(evt.value == "wet")
        state.wet = True
        runIn(d, func)
    if( evt.value == "dry")
        state.wet = False
def func()
     if (state.wet) valve.off()
```

Events Subscription

State

Timed API

Device Action



If leak is detected for *d* duration, turn water valve off

Smart Apps are user-configurable, timed and stateful.

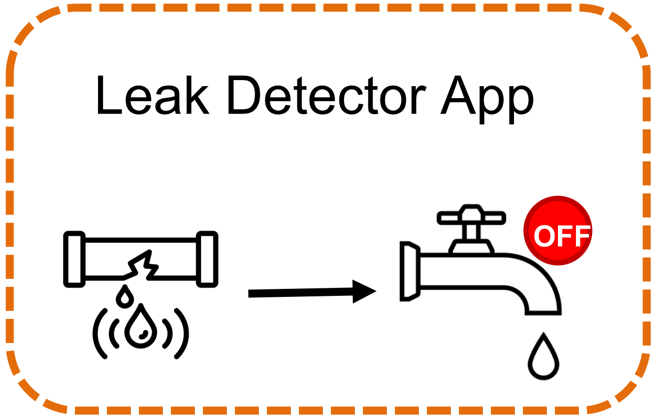
User-configurable Inputs

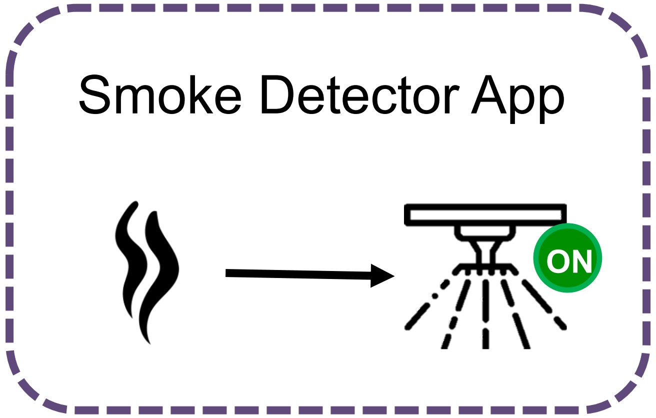
```
preferences
    input "sensor", "capability.waterSensor", input "d", "duration"
    input  "valve",  "capability.valve"
def installed()
    subscribe(sensor, "water", waterHandler)
def waterHandler(evt)
    if(evt.value == "wet")
        state.wet = True
        runIn(d, func)
    if( evt.value == "dry")
        state.wet = False
def func()
      if (state.wet)  valve.off()
```

State

Timed API

Challenge 2: There can be direct and indirect inter-app interactions that may lead to safety violations





Prior work does not suffice

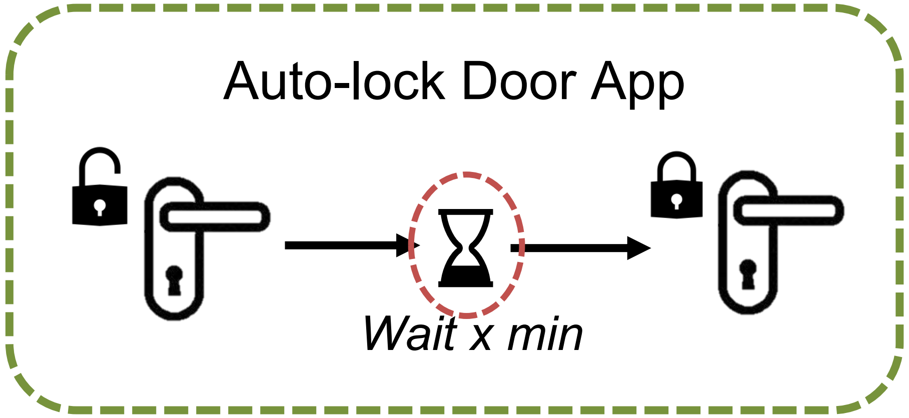
	User Inputs	State	Time	Environment
SiFT	✗	✗	✗	✓
Soteria	✗	discrete	✗	✗
IoTSan	✗	✓	✓	✗
IoTa	✗	✓	✓	✗
HomeGuard	✗	discrete	✗	✗
iRuler	✗	✗	✗	✓
AutoTap	✗	✗	✓	✗
Menshen	✗	✗	✓	✓
Salus	✓	✗	✗	✓
PSA	✓	✓	✓	✓

Liang, Chieh-Jan Mike, et al. "SIFT: building an internet of safe things." *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. 2015.
Celik, Z. Berkay, Patrick McDaniel, and Gang Tan. "Soteria: Automated {IoT} Safety and Security Analysis." *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 2018.
Nguyen, Dang Tu, et al. "IoTSan: Fortifying the safety of IoT systems." *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 2018.
Wang, Qi, et al. "Charting the attack surface of trigger-action IoT platforms." *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2019.

Within an app, modeling time, state, and user inputs is necessary to accurately detect violations.

Across apps, modeling the inter-app interactions in the same environment is necessary to accurately detect violations.

Modeling time is necessary to detect violations accurately

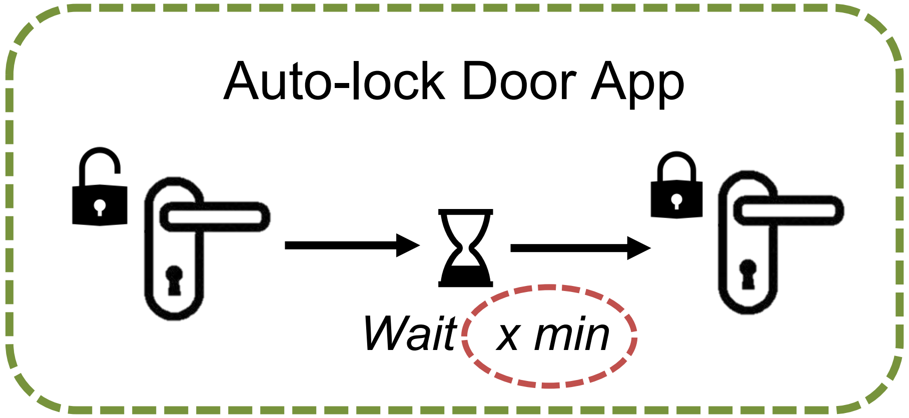


Miss violation!!

SiFT
Soteria
HomeGuard
iRuler
Salus

If we do not model the wait x min part, then there is no violation

Modeling user inputs is necessary to find safe configurations



If we do not check for all values of x, we will not know which values of x are safe

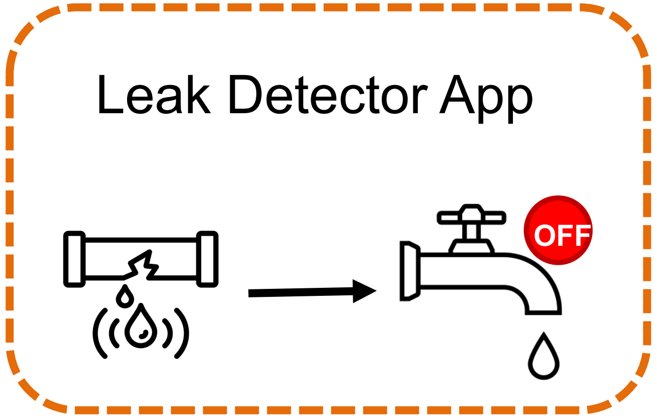
Miss violation!!

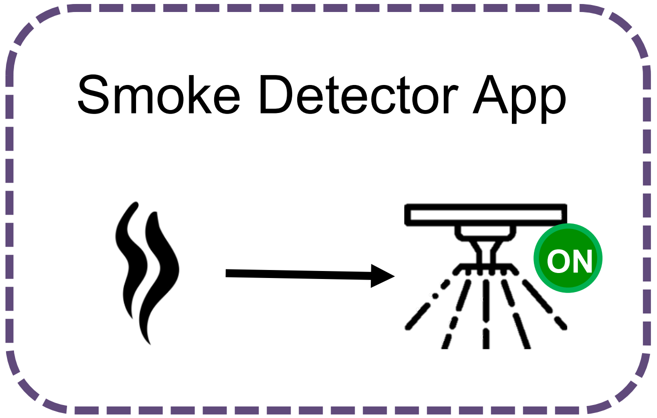
SiFT
Soteria
HomeGuard
iRuler
IoTSan
Iota
AutoTap
Menshen

Modeling environment interactions is necessary to accurately detect violations

Miss violation!!

Soteria
HomeGuard
IoTSan
Iota
AutoTap

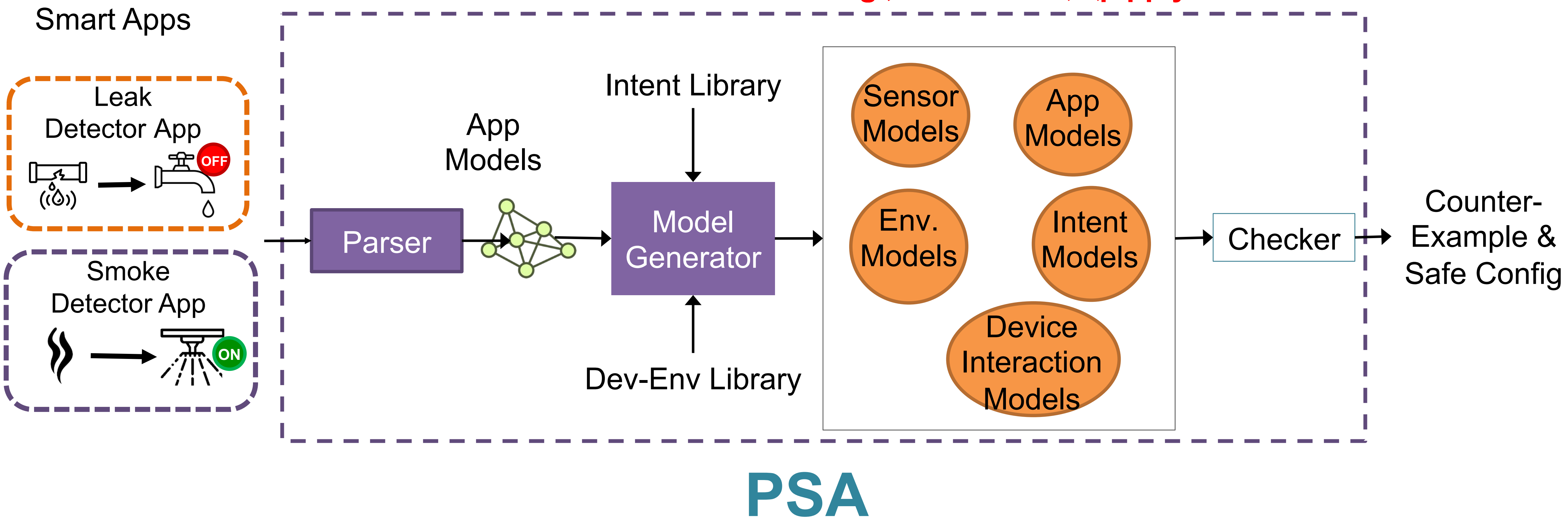




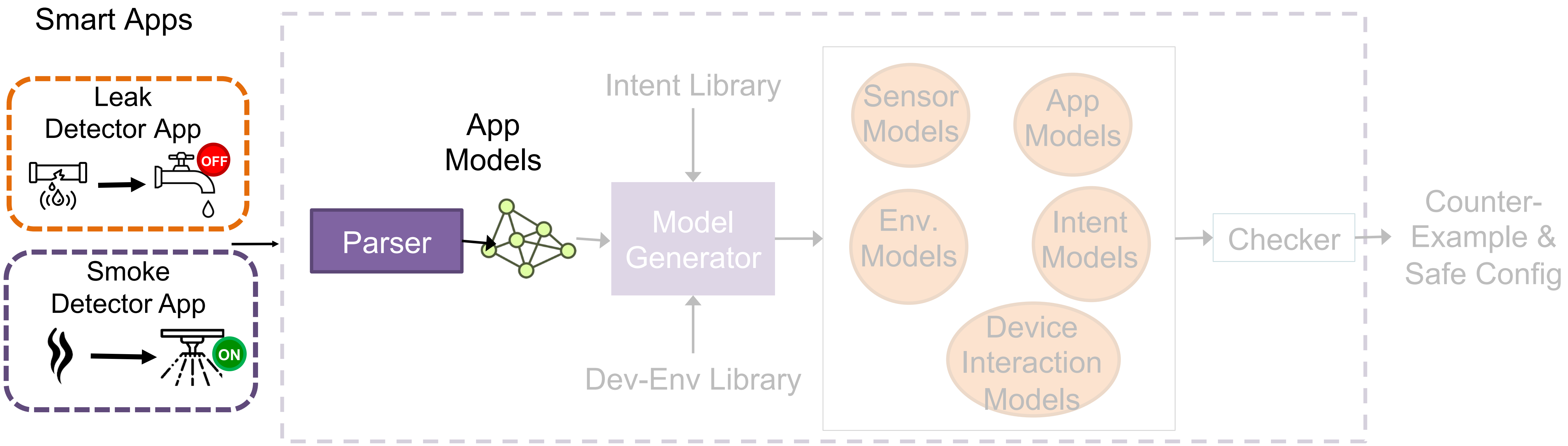
If we do not model the interaction between water supply and sprinklers, we will not know that water sprinkler action is blocked.

We propose PSA, a static analysis, model checking based tool to verify a home deployment for violations

E.g., Water Sensor



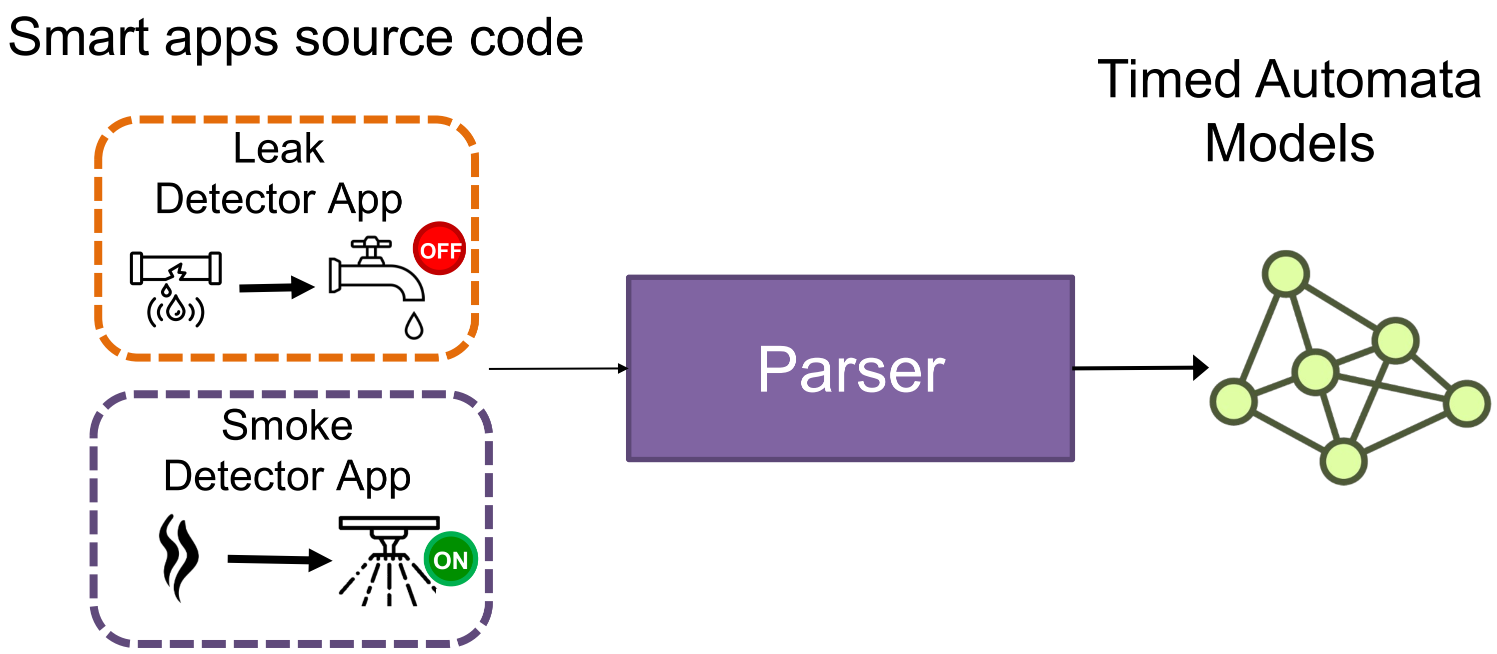
We propose PSA, a static analysis, model checking based tool to verify a home deployment for violations



Challenge 1: State of the art smart apps are complex

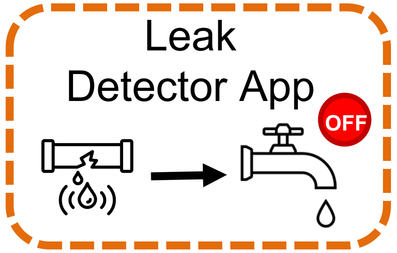
PSA uses timed automata to model stateful, timed and user-configurable apps

PSA uses timed automata to model apps

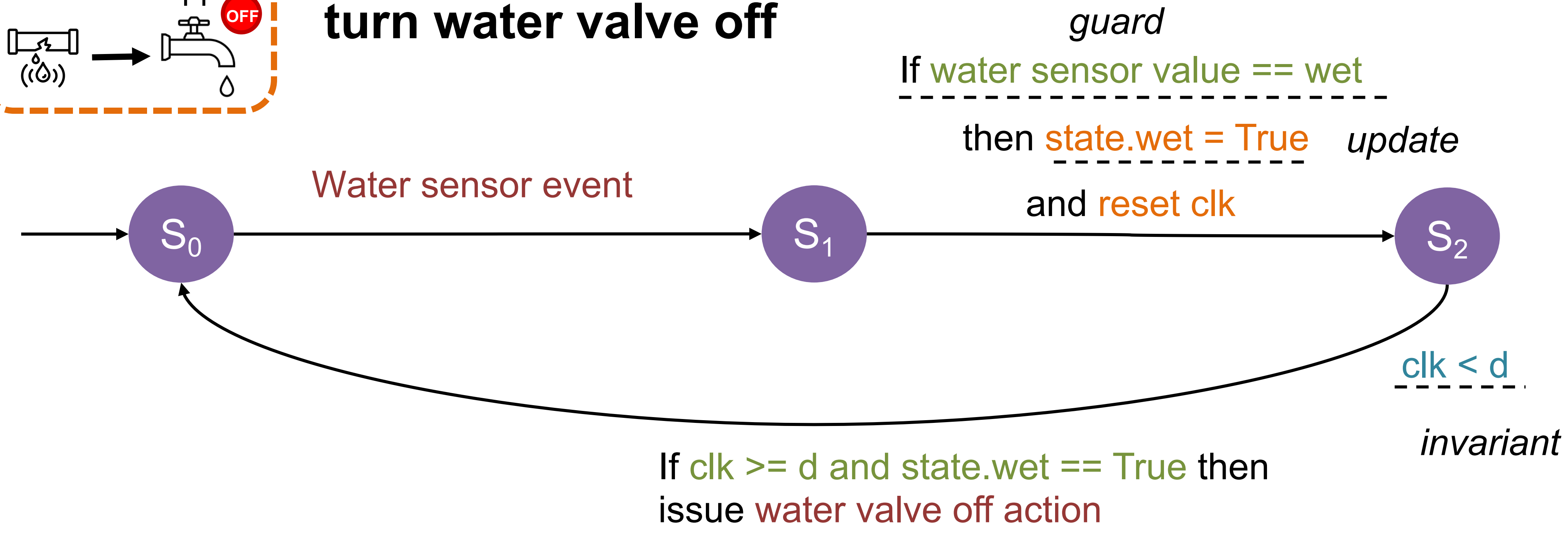


Timed Automata allows us to model time, state and user-configurable inputs

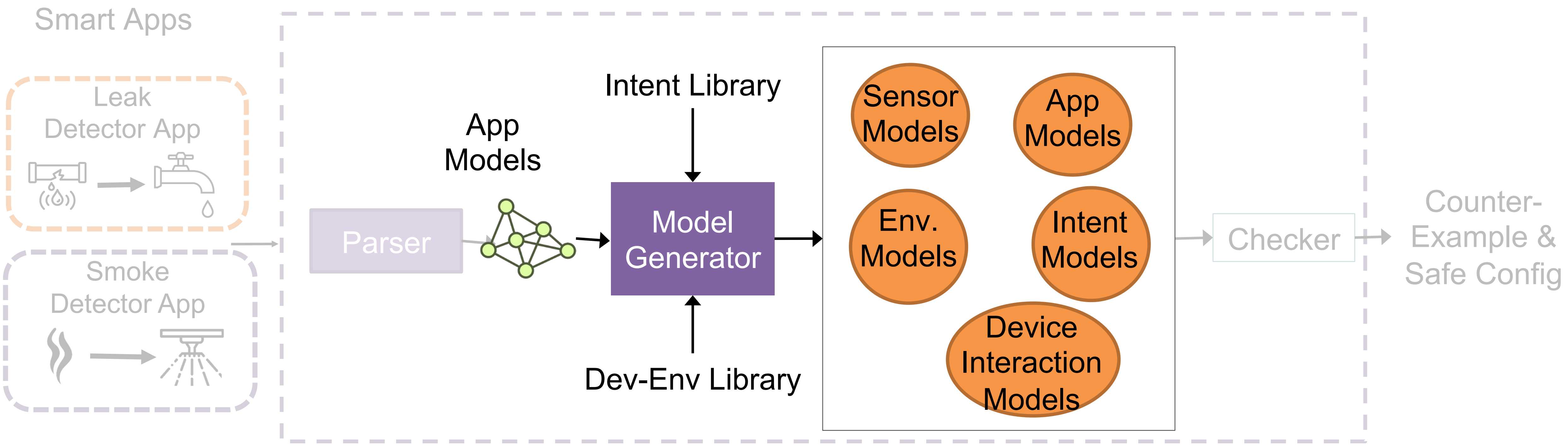
Timed Automata extends a FSM with real valued clocks



If leak is detected for d duration, turn water valve off


guard
If water sensor value == wet
then state.wet = True
update
and reset clk
Water sensor event
S_0
S_1
S_2
clk < d
invariant
If clk >= d and state.wet == True then
issue water valve off action

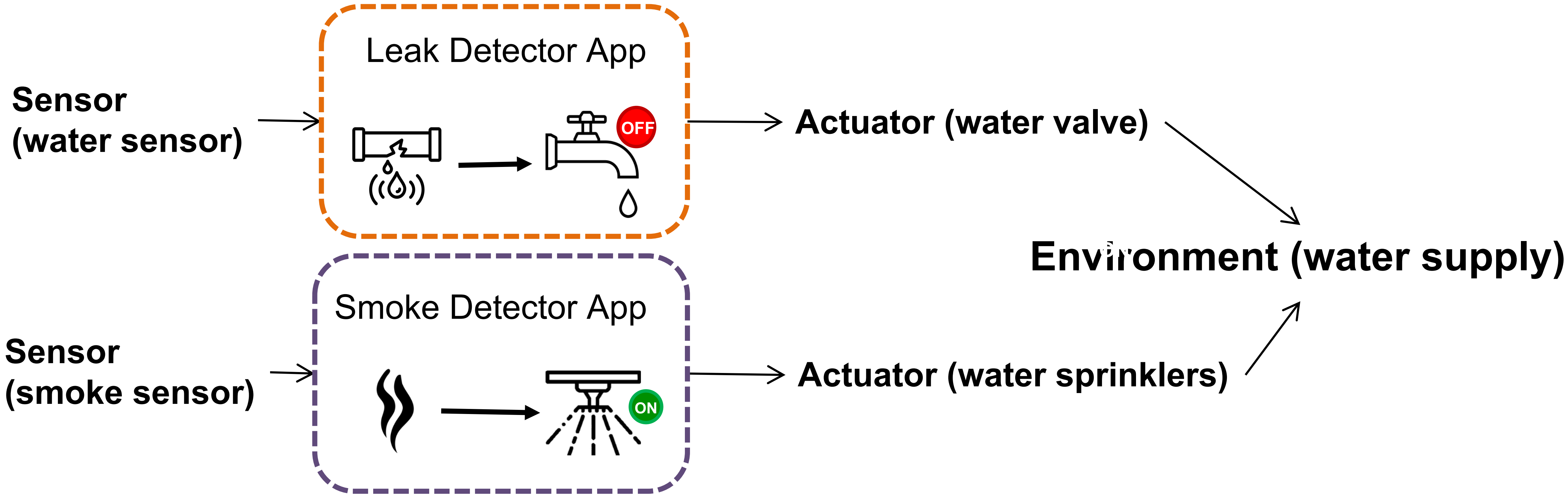
We propose PSA, a static analysis, model checking based tool to verify a home deployment for violations



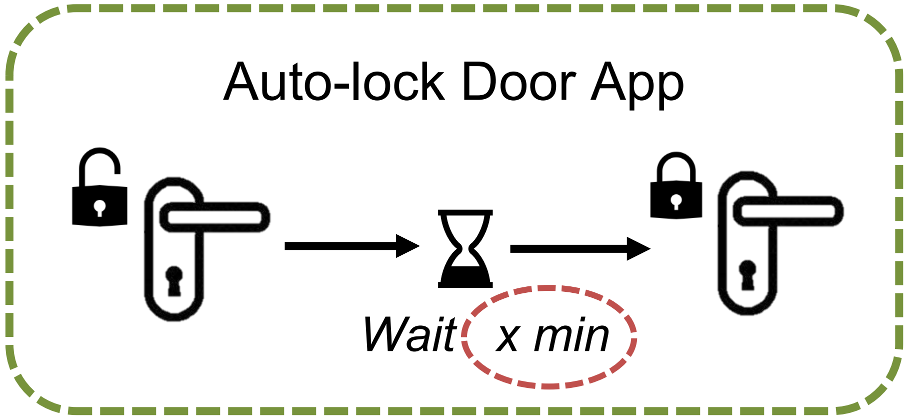
Challenge 2: There can be direct and indirect inter-app interactions that may lead to safety violations

PSA separately models environment attributes, sensors and actuator interactions to model the inter-app interactions

PSA models indirect inter-app interactions by separately modeling devices and environment attributes



PSA outputs a set of safe configurations and a counter-example



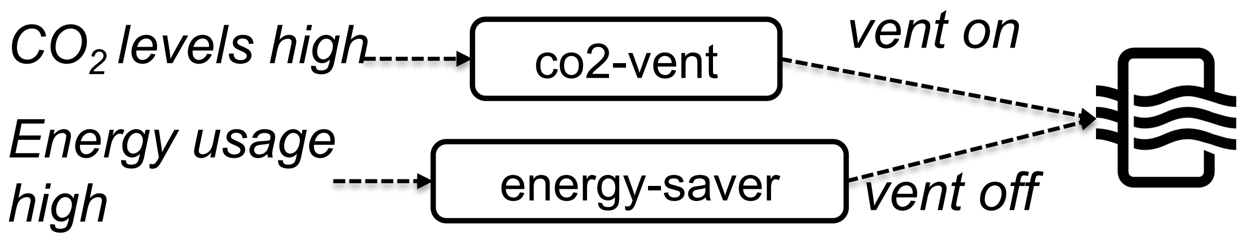
PSA outputs that x should be at most 1 min

We evaluate on 86 Samsung SmartThings Apps

Stateful Apps	Timed Apps	Apps with User Inputs
21%	43%	59%

We find 19 new violations

- We find 640 total violations



- We find 19 new violations
  - For example:
    - Lights are turned on and off frequently causing a strobing effect
    - Door remains unlocked for more than 5 min
    - Thermostat set-points cannot be set because thermostat is off

Not modeling state and time results in up to 35% false positives

	B1	B2	PSA
Device Conflict	**35%**	**27%**	**0%**
Environment Conflict	**21%**	**16%**	**0%**
Co-occurrence Violation	**11%**	**6%**	**0%**

Baselines:
B1: Stateless, untimed, no user inputs (SiFT)
B2: Discrete State only, untimed, no user inputs (Soteria)

To conclude…

- We propose PSA, a static analysis model checking based tool.
- PSA verifies smart home deployments for safety intent violations.
- We choose timed automata as a suitable abstraction to model state, time and user inputs in apps.
- We show that not modeling state, time and user inputs can result in up to 35% false positives.
- PSA finds 19 new violations as compared to prior work.