CS 498: Bachelor's Thesis Project

Shield Synthesis for Cyber-Physical Systems

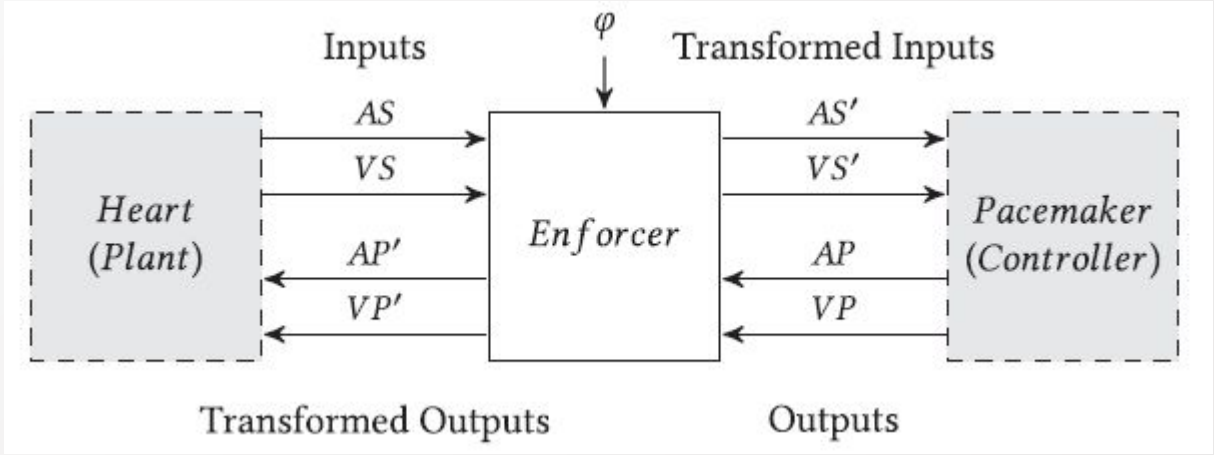
Supervisor: Dr. Purandar Bhaduri

Presented by:
Samay Varshney (180101097)
Siddhartha Jain (180101078)

Existing Approach 1

Runtime Enforcement of CPS

Bidirectional Runtime Enforcement



Bidirectional Runtime Enforcement

- Used in enforcing security policies.
- Enforcer suppresses malicious inputs from an attacker by modifying inputs.
- In Unmanned Aerial Systems, an attacker may feed-in bad inputs to take system control.

Limitations of the Approach

- Using this approach, only enforceable properties can be enforced.
- Enforceable properties are just a subset of safety properties.

Property is enforceable if and only if an accepting state is reachable from every non-violating state in one or more steps.

Example: Let's take a non-enforceable property as defined using DTA in figure.

- Inputs (Σ_I) = {0,1}, Outputs (Σ_O) = {0,1}, $\Sigma = \Sigma_I \times \Sigma_O$
- Input-output sequence $\sigma = (1, 1) \cdot (1, 0) \in \Sigma^*$.

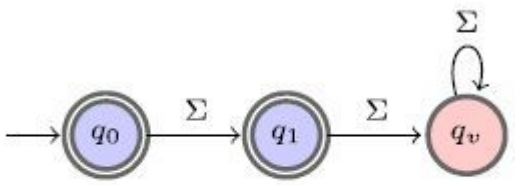


Fig. 6. A non-enforceable property.

Problems Faced in usage of Tool

Explored the runtime enforcement tool (https://github.com/PRETgroup/easy-rte).

- In pacemaker example, on removing manual recover statements, in some cases it was giving wrong outputs.
- In some cases it says "no solution is found" although solution exists.
- In other, it was editing adding unnecessary statements like ":=0" which does not make sense.

```
samay@samay-VM:~/Documents/easy-rte-master$ make run_cbmc PROJECT=pacemaker FILE=p1p2
cbmc example/pacemaker/cbmc_main_p1p2.c example/pacemaker/F_p1p2.c
CBMC version 5.10 (cbmc-5.10) 64-bit x86_64 linux
Parsing example/pacemaker/cbmc_main_p1p2.c
Parsing example/pacemaker/F_p1p2.c
file example/pacemaker/F_p1p2.c line 105 function p1p2_run_output_enforcer_p1p2: syntax error
PARSING ERROR
Numeric exception : 0
make: *** [Makefile:37: run_cbmc] Error 6
```

```
NOTE: I will perform the following edits:
        := 0;
        VP := 1;
```

Problems Faced in usage of Tool (contd.)

- In our policies, we need to add manual recover statements
- otherwise it is saying "no solution is found"
- Making the whole tool useless if we have to correct violation manually

```
samay@samay-VM:~/Documents/easy-rte-master$ make c_enf PROJECT=pacemaker FILE=p4
./easy-rte-parser  -i example/pacemaker/p4.erte -o example/pacemaker/p4.xml
Writing to example/pacemaker/p4.xml
./easy-rte-c -i example/pacemaker/p4.xml -o example/pacemaker
Problem when deriving a solution for violation transition
        s1 -> violation on (VP)
        (If this is undesirable behaviour, use a 'recover' keyword in the erte file
        NOTE: No solution found!
```

```
** Results:
[p4_run_output_enforcer_p4.assertion.1] assertion
[p4_run_output_enforcer_p4.assertion.2] assertion
[p4_run_output_enforcer_p4.assertion.3] assertion

** 1 of 3 failed (2 iterations)
VERIFICATION FAILED
make: *** [Makefile:37: run_cbmc] Error 10
```

Existing Approach 2

Shield Synthesis for Timed Properties

Timed Shield

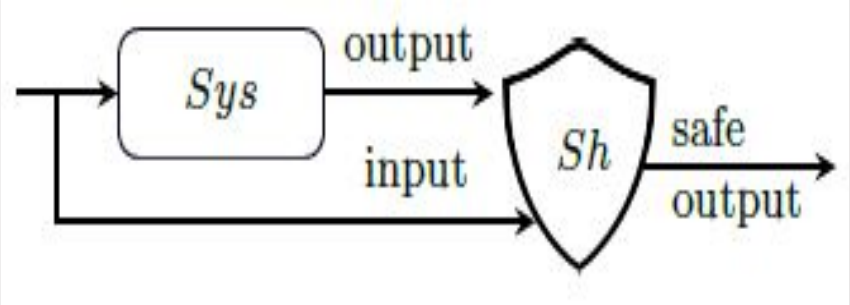
- A timed shield is attached after the controller,
- Monitors its inputs and outputs,
- Corrects and forwards the correct output to the environment/plant.

Desired Properties of the shield:

- Correctness: Shield ensures correctness if and only if a shielded system refines specification.

$$(\text{System} \mid \text{Shield}) \leq \text{Specification}.$$

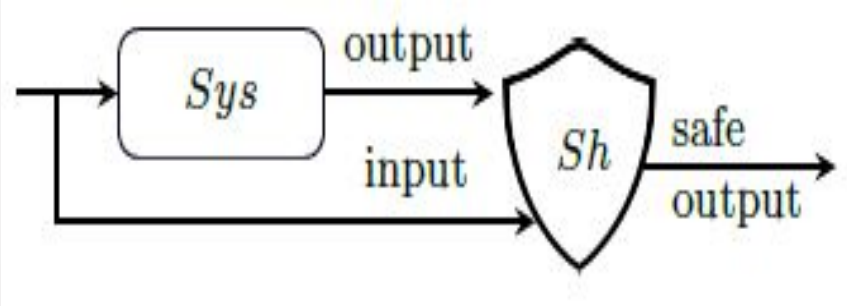
  - Refinement (≤): containment of timed behaviour. i.e.
  - Behaviour of shielded system is a subset of behaviour of specification.



Timed Shield

Desired Properties of the shield:

- No Unnecessary Deviation: Shield does not deviate from System unnecessarily.



Timed Shield Synthesis

- Timed shield can be synthesized by solving a timed game.
- Played by 2 players: System and Environment on Timed game automata (TGA).

- Timed Game Automata is timed automata in which set of output actions partitioned into controllable actions and uncontrollable actions.
- System can choose controllable actions.
- Environment chooses uncontrollable actions.

- System need to satisfy the safety objective. How?
  - **Strategy:** A function over the states of TGA to the set of controllable actions or a special nothing symbol.
  - **Winning**: Safety objective is never violated no matter what environment does.
  - **Winning strategy**: A strategy which is always winning no matter what strategy environment chooses.

A **Timed Shield** is the network of Timed Automata obtained by composing Timed Game Automata (TGA) with the winning strategy.

Implemented Examples using UPPAAL Tool

- Car Platooning System
- Heart Pacemaker System

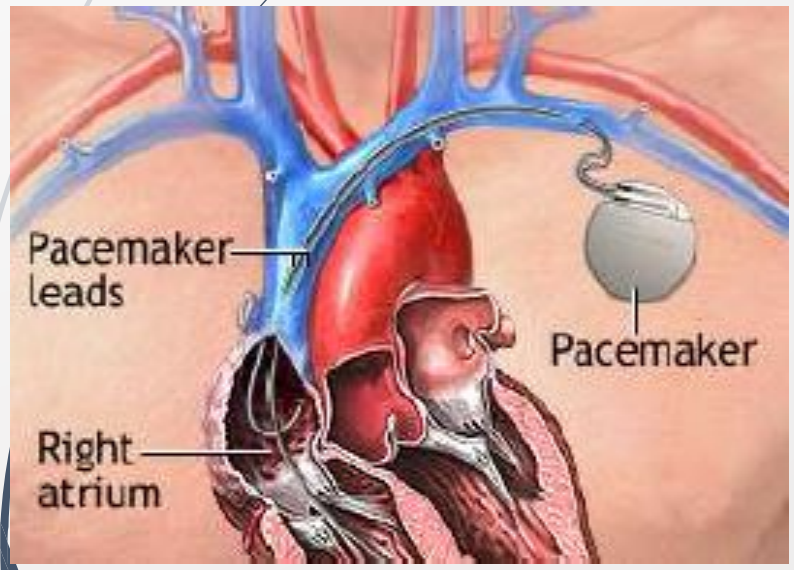


Fig. 1. Distance, velocity and acceleration between two cars.

Limitations of the Algorithm

- Recovers the system in case of **transient faults** only.
- In case another fault is encountered afterwards, then shield will not be able to recover i.e. cannot go in sync with the system.
- To capture all transient faults in a system, a large number of fault models is needed.
- It may not be feasible to synthesize shields with guaranteed recovery for large specifications considering a large number of fault models.

Future Work

- We could apply the timed shield algorithm and the work to other real life applications as well. We could think of more such case study.
- We could explore other real life applications as cyber physical system with real values as input and output.
- All the work done by us can be seen here: https://github.com/BTP-CPS/Phase-1