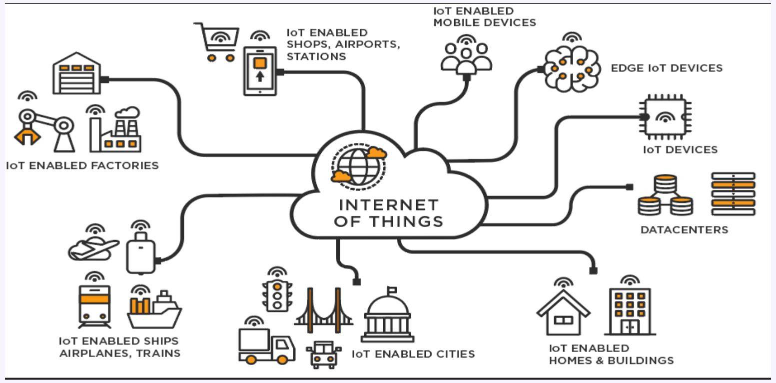
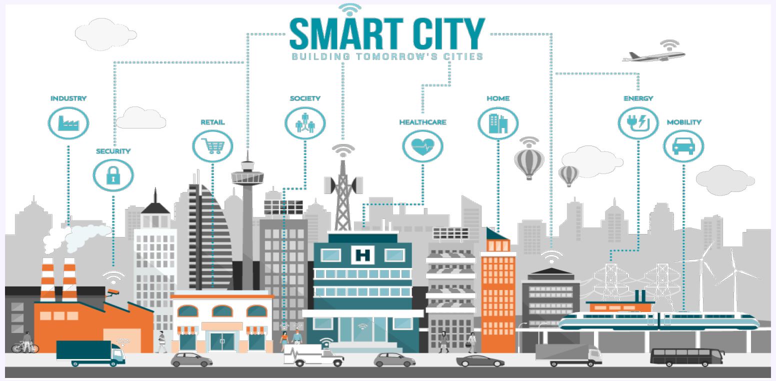
STL 모델 검증 및 응용

Jia Lee, Geunyeol Yu, Kyungmin Bae

Software Verification Lab, POSTECH

Cyber-Physical Systems

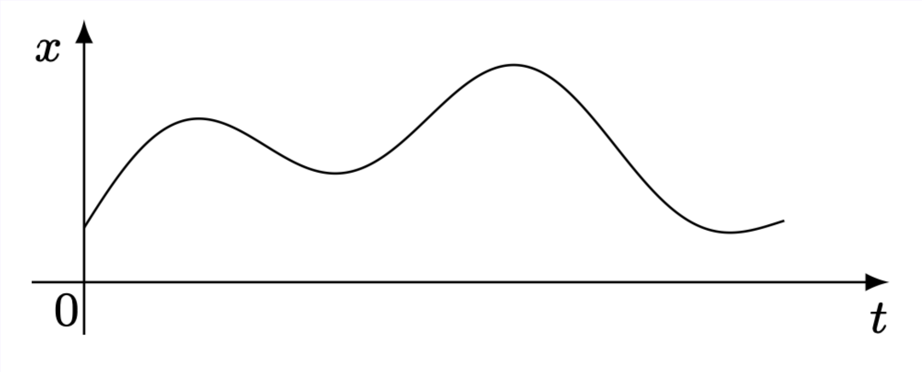




⇒ Fault diagnosis is key to reducing huge losses of both life and property

Signal Temporal Logic (STL)

- Requirement of CPS: CPS satisfies a desired property?



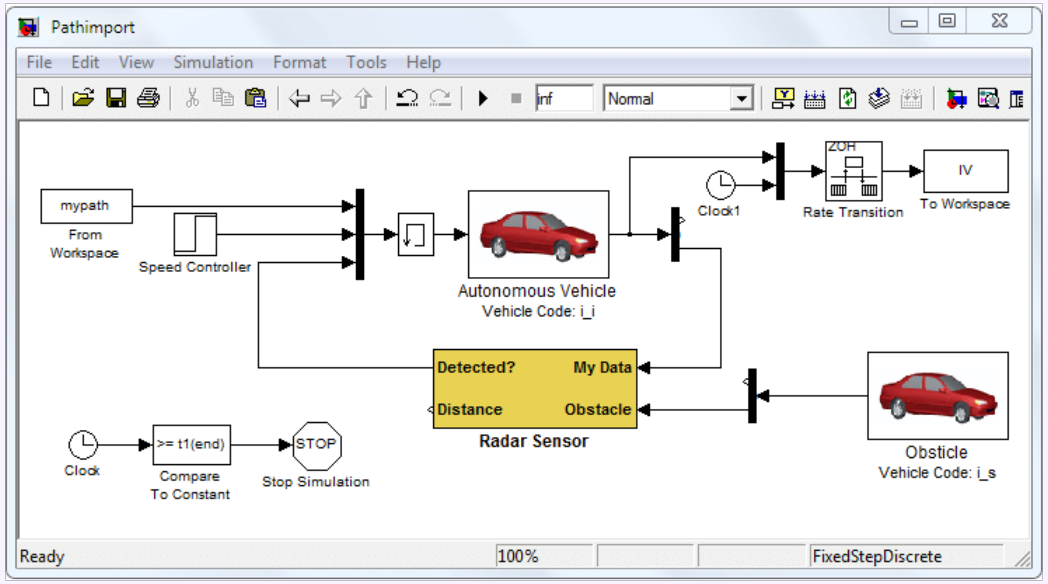
- Signal Temporal Logic (STL) : Specify properties of continuous real-valued signals

Ex) At some time in the first 10 seconds, x position is between $5m$ and $8m$ for 5 seconds.

$$\Rightarrow \Diamond_{[0,10]}(\Box_{[0,5]}(5m < x < 8m))$$

Verification Methods: Monitoring and Falsification

- Monitoring and Falsification:
  1) Model a system and simulate the model



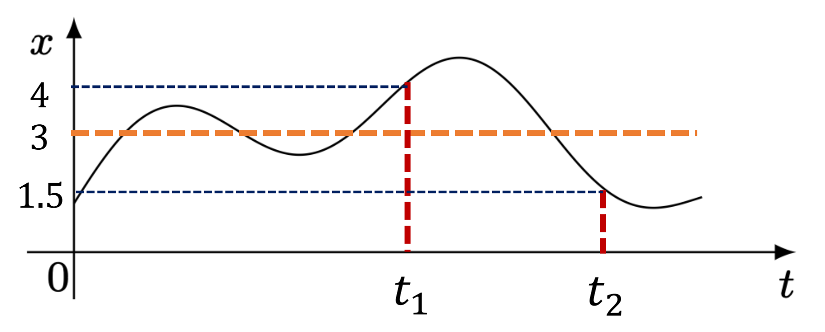
  2) Check whether a property holds for a given simulation trace

Verification Methods: Monitoring and Falsification

- Monitoring and Falsification:

  2) Check whether a property holds for a given simulation trace

  - Boolean semantics: True / False
  - Quantitative semantics: Indicate how well the property is satisfied (robustness degree)
  - Example: $x > 3$?



	t_1	t_2
Boolean semantics	True	False
Quantitative semantics (Robustness degree)	1	-1.5

- Limitation: Can’t guarantee correctness

Verification Method: Model Checking

- STL model checking ： Do all possible signals of CPS satisfy STL φ?
  - If a counterexample exists, it can always be found
- Limitation

1) Incomplete even for bounded signals

2) Only boolean semantics approach：

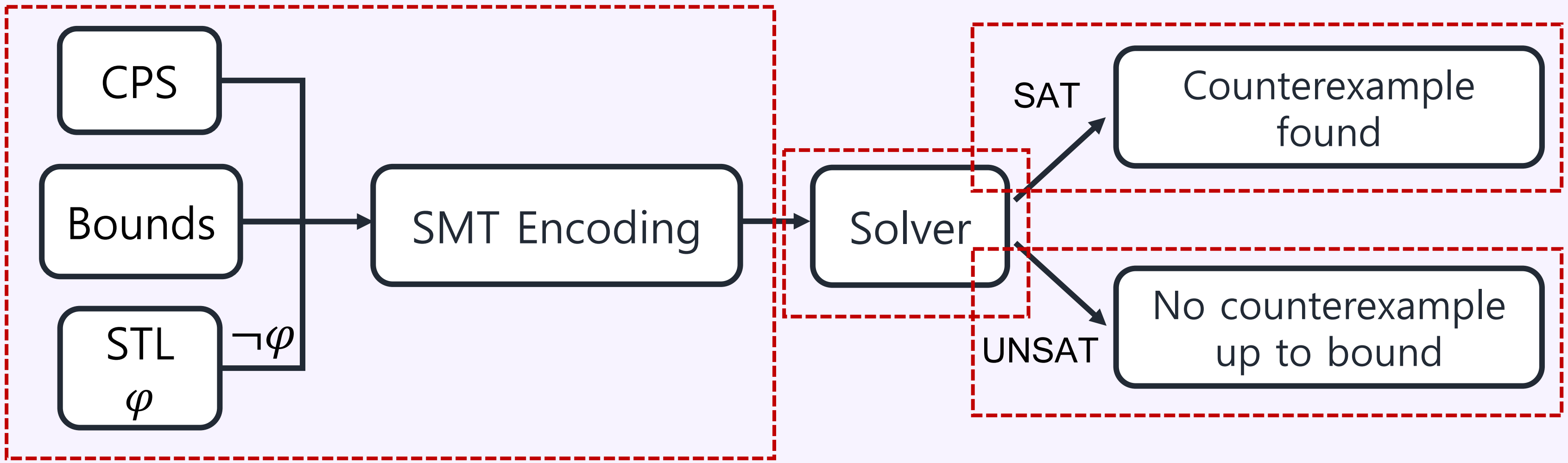
   small perturbation of signals can cause the system to violate a property

Contribution

- Propose Boolean STL model checking algorithms (POPL '19, ASE '21)
  - Refutationally complete for bounded signals
- Propose robust STL model checking
  - Quantitative semantics approach
- Develop a robust STL model checker ***STLMC***

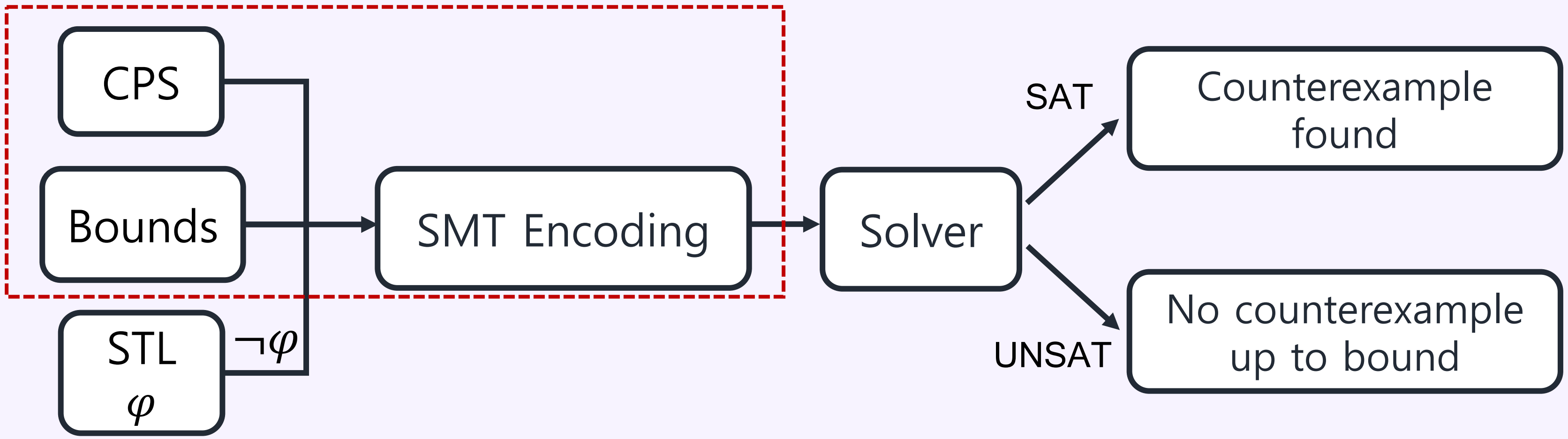
SMT-based Bounded STL Model Checking

- SMT-based bounded STL model checking framework



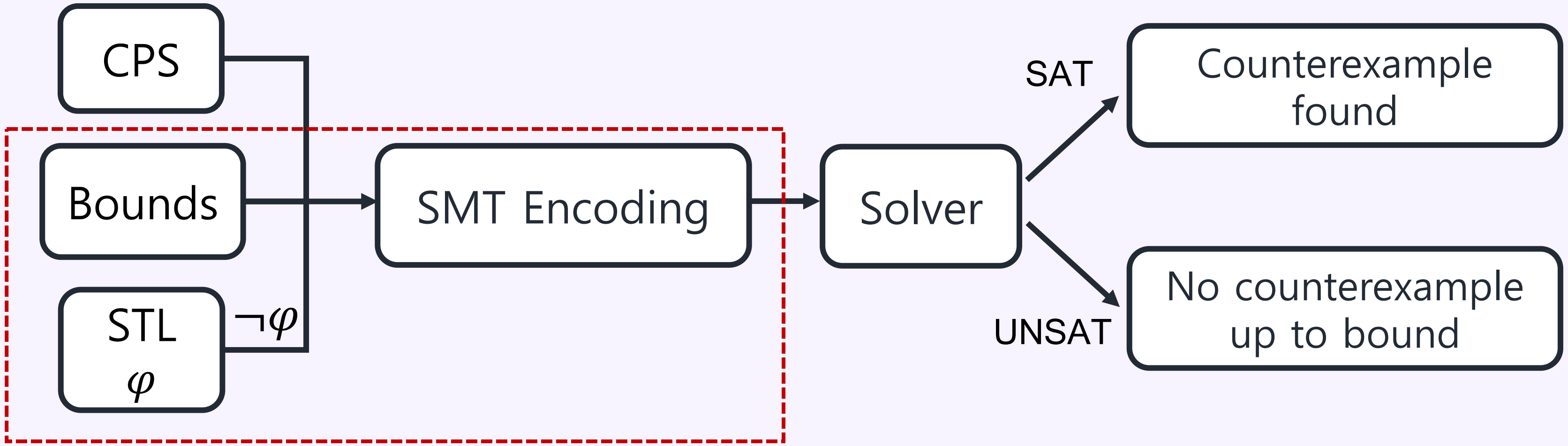
SMT-based Bounded STL Model Checking

- SMT-based bounded STL model checking framework



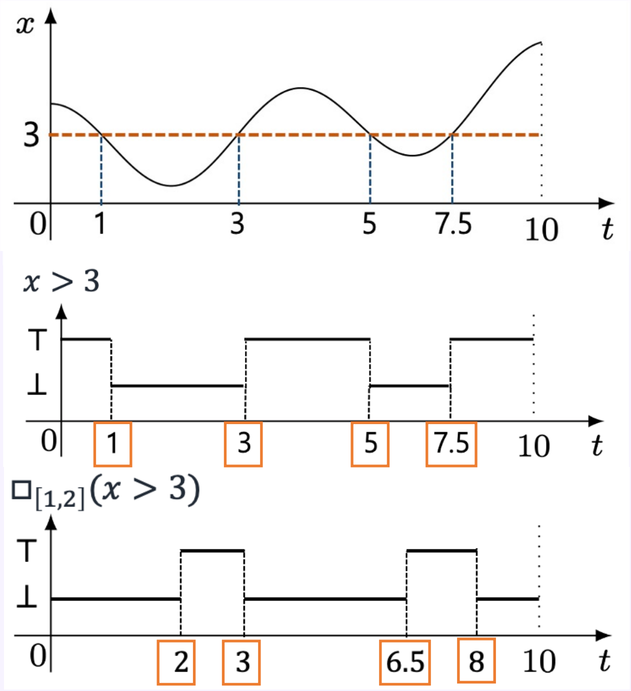
SMT-based Bounded STL Model Checking

- SMT-based bounded STL model checking framework



Key Idea

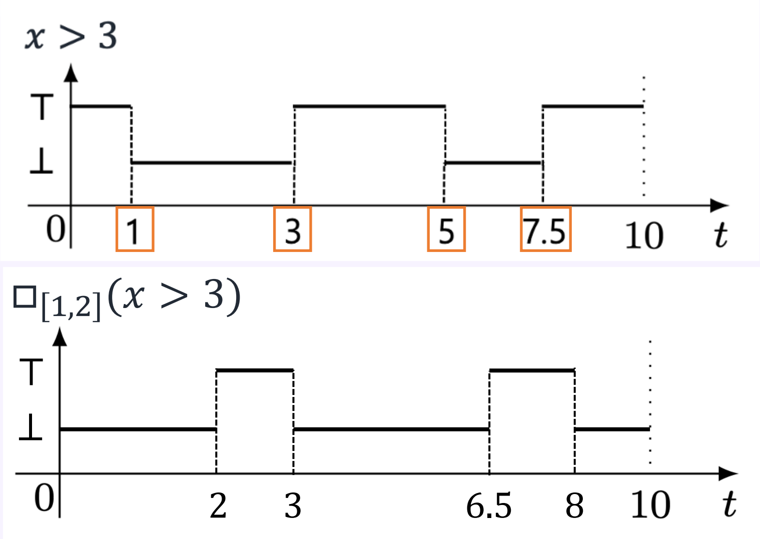
Ex) STL formula $\varphi = \Box_{[1,2]}(x > 3)$



- Truth values of STL are change discontinuously
- Variable point: a time point at which the **truth value** of subformula of φ **changes**
- SMT encoding of STL based on variable points

Calculation of Variable Points

Ex) STL formula $\varphi = \square_{[1,2]}(x > 3)$



- Calculation variable points for STL
  1. Time points when the truth values of subformula are changed (ex. $\bot \rightarrow \top$)
  2. Time interval in STL temporal operator

SMT Encoding of STL

- Bound parameters: time domain and the number of variable points

- SMT encoding of STL
  - translate each subformula of STL to first order logic
  - translate STL to first order logic using the translation result of subformula

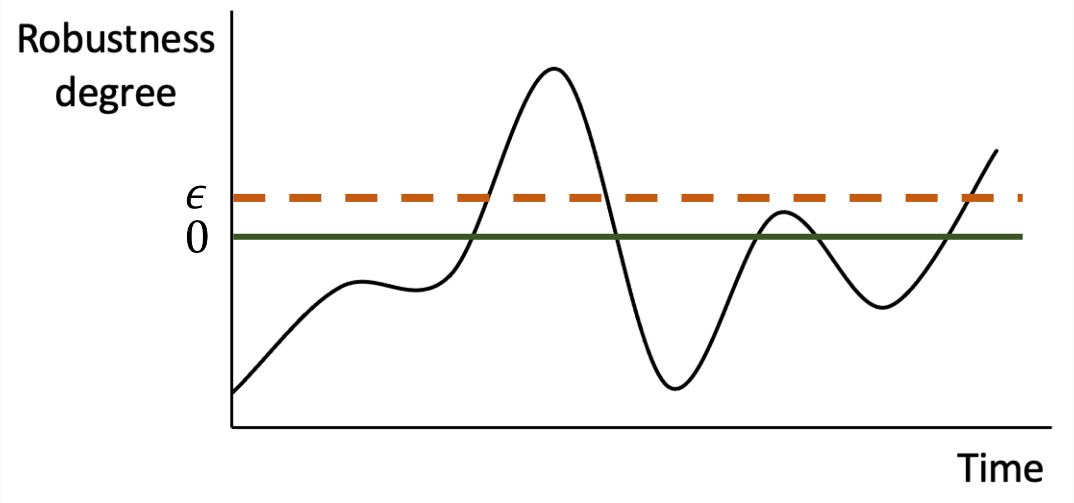
Contribution

- Propose bounded model checking algorithms for signal temporal logic
  - Refutationally complete for bounded signals
- Propose robust STL model checking
  - Check robustness degrees of STL with respect to all possible signals of CPS
- Develop a robust STL model checker ***STLMC***

Robust STL model checking

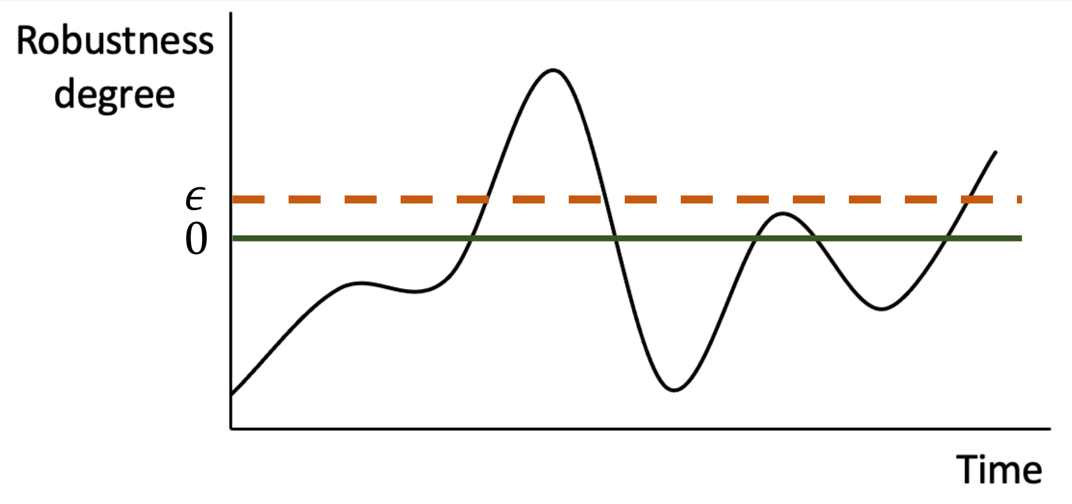
- Problem: Only boolean semantics approach
  - Small perturbations of signals can cause the system to violate the properties
- Robust STL model checking:

  Check whether *robustness degrees* with respect to all possible signals are greater than a robustness threshold $\epsilon > 0$

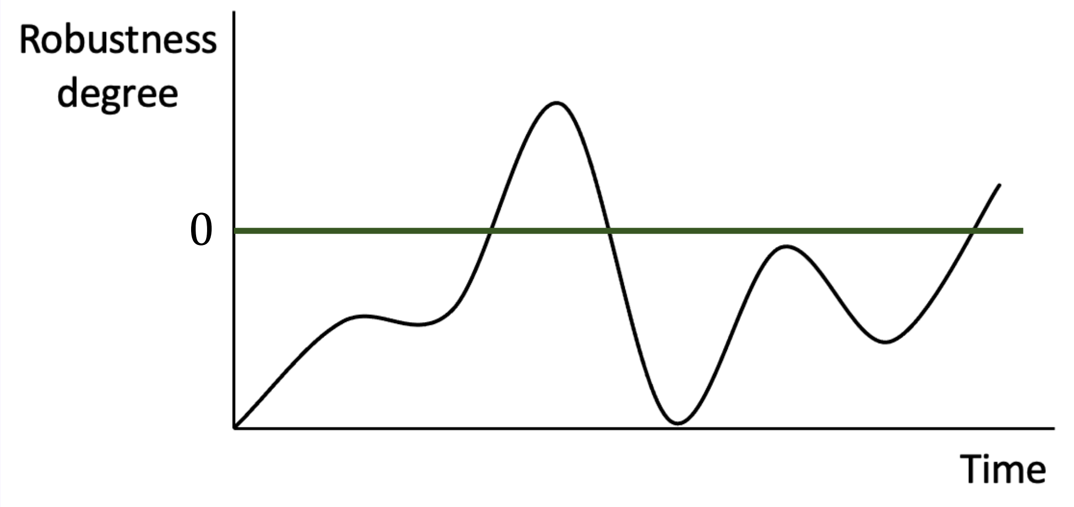


ϵ-Strengthening

- Robustness degree of $x \geq 0$



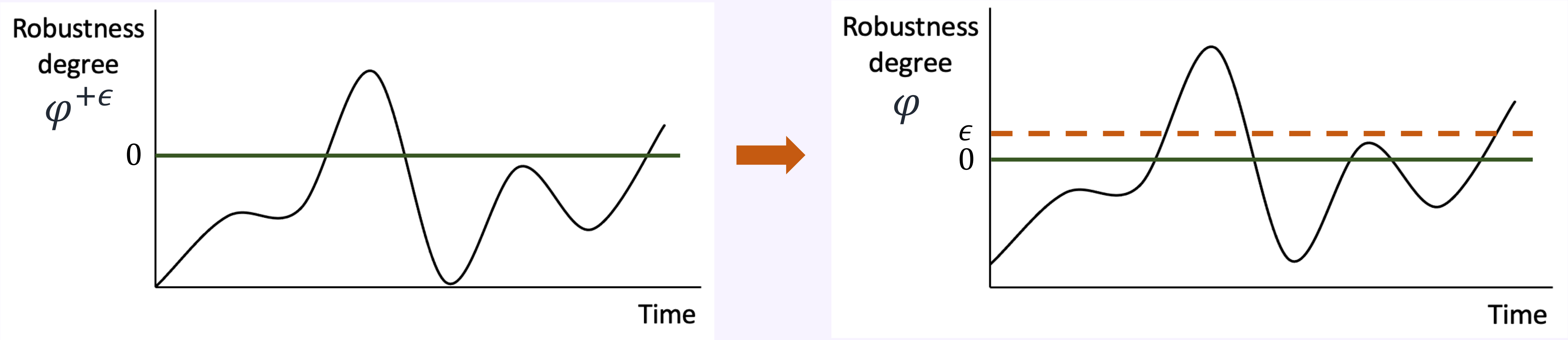
- Robustness degree of $x \geq \epsilon$



- $x \geq \epsilon$ is stronger than $x \geq 0$ by ϵ
- Extend the definition of ϵ-strengthening to STL, $\varphi^{+\epsilon}$

Reduction to Boolean STL Model Checking

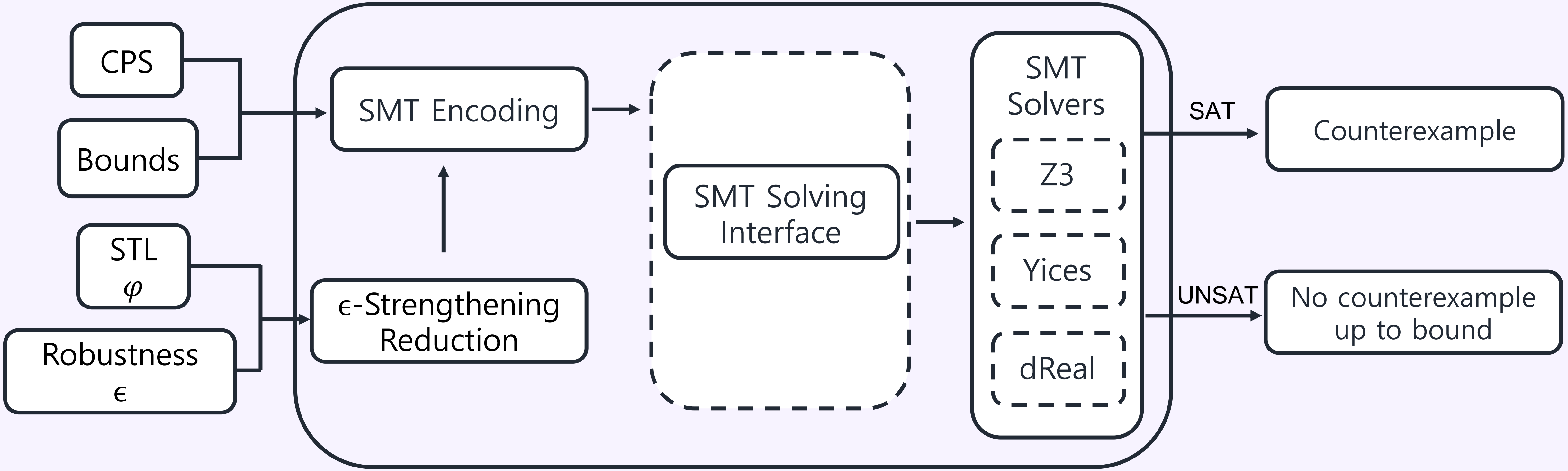
- Find a counterexample of $\varphi^{+\epsilon}$ for Boolean STL model checking



$\Rightarrow$ That is also a counterexample of φ for robust STL model checking

- Can reduce robust model checking to Boolean STL model checking

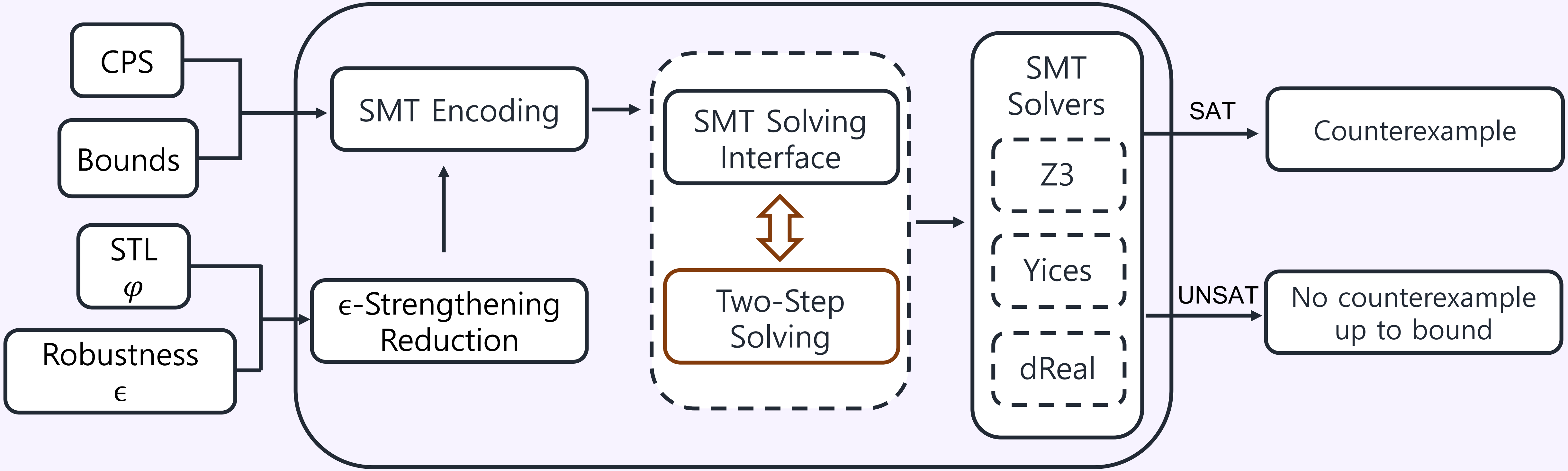
Robust STL Model Checking Framework



- Problem: Computation cost of ODE dynamics is highly expensive

  ⇒ Cannot obtain results in time

Robust STL Model Checking Framework



Parallelized Two-Step Solving

- Two-step solving procedure
    1) Abstract of flow and invariant conditions

Parallelized Two-Step Solving

- Two-step solving procedure
  1) Abstract of flow and invariant conditions
  2) Enumerate a **possible scenario** of the abstraction

Parallelized Two-Step Solving

- Two-step solving procedure
  1) Abstract of flow and invariant conditions
  2) Enumerate a **possible scenario** of the abstraction
  3) Check the scenario with the flow and invariant
- Can parallelize the enumerations and the scenario checking

Minimization of Enumeration Scenarios

- Too many possible scenarios

Ex)

```
if (x > 40) or (y > 50) or (v > 20):
    setVelocity(v_low)
```

  - There are 7 possible scenarios
  - However, when $x > 40$ is satisfied, $\mathrm{y} > 50$ and $\mathrm{v} > 20$ are not important

  ⇒ Suffices to consider only 3 scenarios

- Use a dual propagation approach to minimize enumerated scenarios

STLMC model checker: *STLMC*

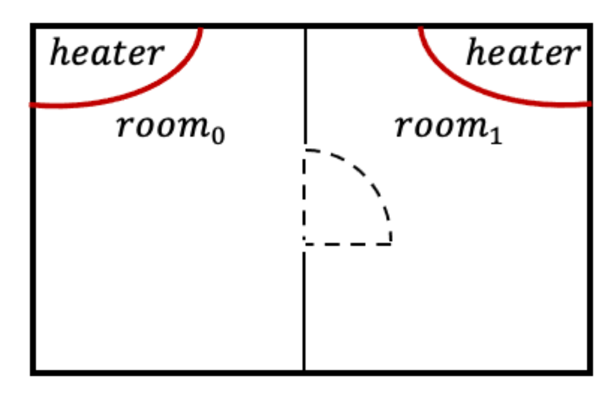
- Develop a robust STL model checker ***STLMC***
- Functions:
  - Connect with various SMT solvers, such as Z3, Yices, and dReal

    ⇒ Can verify CPS with ODE dynamics
  - Implement several optimization techniques
  - Visualization of counterexample signals and robustness degrees

    ⇒ Can analyzing counterexamples and debugging CPS

Example

- Networked Thermostat Controllers



- x_i: temperature of each room
- x_i changes depending on the heater and the temperature of the other room

- Control heaters to keep the temperatures within a certain range
- STL property: $\Box_{[2,4]}\big((x_0 - x_1 \geq 4) \rightarrow \Diamond_{[3,10]}\,(x_0 - x_1 \leq -3)\big)$

Example

- Input model

```
const k0 = 0.015;        const k1 = 0.045;
const h0 = 100;          const h1 = 200;
const c0 = 0.98;         const c1 = 0.97;
const d0 = 0.01;         const d1 = 0.03;

int on0;                 int on1;
[10, 35] x0;             [10, 35] x1;

{ mode: on0 = 0;          on1 = 1;
  inv:  10 < x0; x1 < 30;
  flow: d/dt[x0] = - k0 * (c0 * x0 - d0 * x1);
        d/dt[x1] = k1 * (h1 - (c1 * x1 - d1 * x0));
  jump: x0 <= 17 => (and (on0' = 1) (on1' = 0)
                         (x0' = x0) (x1' = x1));
        x1 >= 26 => (and (on1' = 0) (on0' = on0)
                         (x0' = x0) (x1' = x1));
}
{ mode: on0 = 1;          on1 = 0;
  inv:  x0 < 30; x1 > 10;
  flow: d/dt[x0] = k0 * (h0 - (c0 * x0 - d0 * x1));
        d/dt[x1] = - k1 * (c1 * x1 - d1 * x0);
  jump: x1 <= 16 => (and (on0' = 0) (on1' = 1)
                         (x0' = x0) (x1' = x1));
    x0 >= 25 => (and (on0' = 0) (on1' = on1)
                     (x0' = x0) (x1' = x1));
}
{ mode: on0 = 0;          on1 = 0;
  inv:  x0 > 10; x1 > 10;
  flow: d/dt[x0] = - k0 * (c0 * x0 - d0 * x1);
        d/dt[x1] = - k1 * (c1 * x1 - d1 * x0);
  jump:
    x0 <= 17 => (and (on0' = 1) (on1' = on1)
                     (x0' = x0) (x1' = x1));
    x1 <= 16 => (and (on1' = 1) (on0' = on0)
                     (x0' = x0) (x1' = x1));
}

init: on0 = 0;  18 <= x0;  x0 <= 22;
      on1 = 0;  18 <= x1;  x1 <= 22;

proposition:
  [p1]: x0 - x1 >= 4;    [p2]: x0 - x1 <= -3;

goal:
  [f1]: <>[0, 3](x0 >= 13 U[0, inf) x1 <= 22);
  [f2]: [][2, 4](p1 -> <>[3, 10] p2);
```

- Command

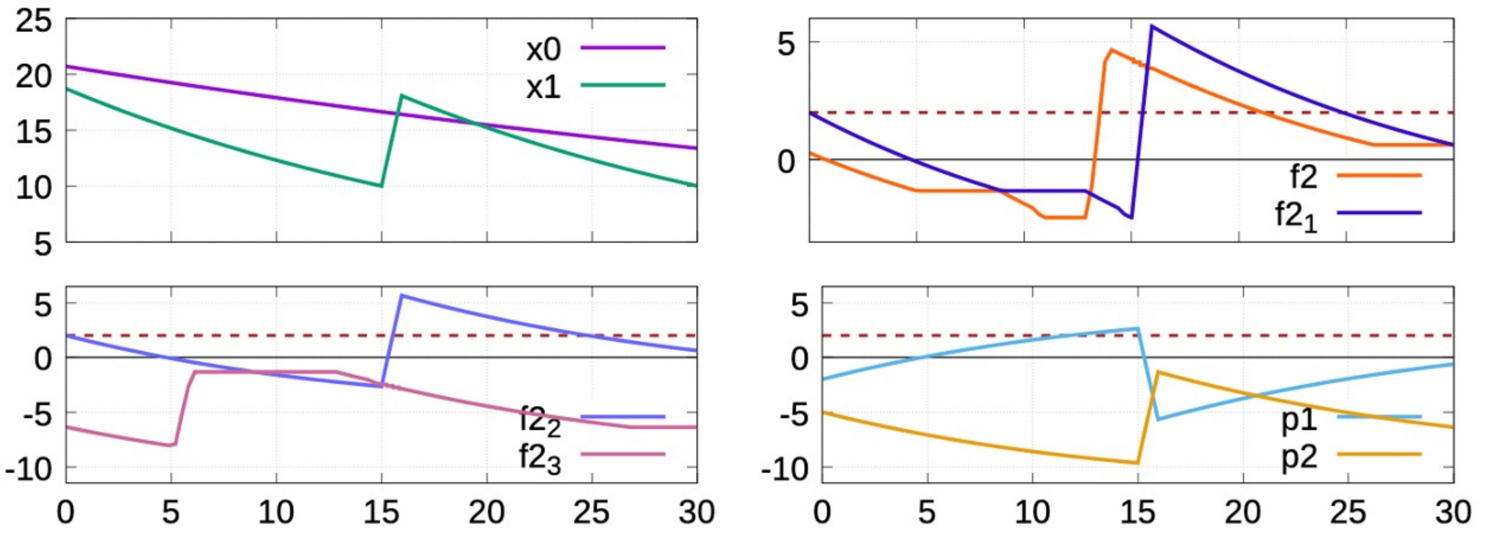
```
$./stlmc ./therm.model -bound 5 -time-bound 30 -threshold 2 \
         -goal f2 -solver dreal -two-step -parallel -visualize
result: counterexample found at bound 2 (7.46335 seconds)
```

Analyzing counterexamples

- Visualize counterexample signals and robustness degrees

$\mathtt{f2}_1 = x_0 - x_1 \geq 4 \rightarrow \Diamond_{[3,10]}(x_0 - x_1 \leq -3)$ $\quad\mathtt{f2}_2 = \neg(x_0 - x_1 \geq 4)$

$\mathtt{f2}_3 = \Diamond_{[3,10]}(x_0 - x_1 \leq -3)$ $\quad\mathtt{p}_1 = x_0 - x_1 \geq 4$ $\quad\mathtt{p}_2 = x_0 - x_1 \leq -3$



- Can analyzing counterexamples and debugging CPS

Experiment: Robust STL Model Checking

- Benchmark models
  - Two networked thermostat
  - A filtered oscillator
  - Load management for two batteries
  - Autonomous driving of two cars
  - A railroad gate controller
  - Two networked water tank systems

Experiment: STL Model Checking

- Robust STL bounded model checking (Timeout: 30 min)
  - 3 STL formulas with nested temporal operators for each model
  - Use Yices and dReal as the underlying SMT solver
  - Use both direct SMT solving (1-step) and two-step SMT solving (2-step)
- The tool and models are available at https://stlmc.github.io

Experiments

Dyn.	Model	STL formula	ϵ	$\lvert\Psi\rvert$	Time	Result	k	Alg.	$\#\pi$
Linear (N = 20)	Bat	$\Diamond_{[4,10]}(p_1 \rightarrow \Box_{[4,10]}\, p_2)$	0.1	12.9	137	$\top$	-	1-step	-
		$(\Diamond_{[1,5]}\, p_1)\, \mathbf{R}_{[5,20]}\, p_2$	3.5	2.76	5.71	$\bot$	5	1-step	-
		$\Box_{[4,14]}(p_1 \rightarrow \Diamond_{[0,10]}\, p_2)$	0.1	3.8	22.1	$\bot$	8	1-step	-
	Wat	$\Box_{[1,3]}(p_1\, \mathbf{R}_{[1,10]}\, p_2)$	2.5	18.8	26.2	$\top$	-	1-step	-
		$(\Diamond_{[1,10]}\, p_1)\, \mathbf{U}_{[2,5]}\, p_2$	0.1	1.9	4.22	$\bot$	4	1-step	-
		$\Diamond_{[4,10]}(p_1 \rightarrow \Box_{[2,5]}\, p_2)$	0.01	11.2	20.2	$\top$	-	1-step	-
Poly (N = 10)	Car	$\Box_{[0,4]}(p_1 \rightarrow \Diamond_{[2,5]}\, p_2)$	0.5	2.2	7.24	$\bot$	5	1-step	-
		$(\Diamond_{[0,4]}\, p_1)\, \mathbf{U}_{[0,5]}\, p_2$	2.0	1.7	6.27	$\bot$	3	1-step	-
		$\Diamond_{[0,3]}(p_1\, \mathbf{U}_{[0,5]}\, p_2)$	0.1	7.3	9.72	$\top$	-	1-step	-
	Rail	$\Diamond_{[0,5]}(p_1\, \mathbf{U}_{[1,8]}\, p_2)$	1.0	2.3	3.43	$\bot$	5	1-step	-
		$\Diamond_{[0,4]}(p_1 \rightarrow \Box_{[2,10]}\, p_2)$	5.0	3.8	0.86	$\top$	-	1-step	-
		$(\Box_{[0,5)}\, p_1)\, \mathbf{U}_{[2,10]}\, p_2$	4.0	1.9	2.83	$\bot$	4	1-step	-
ODE (N = 5)	Thm	$\Diamond_{[0,3]}(p_1\, \mathbf{U}_{[0,\infty)}\, p_2)$	1.0	1.2	817	$\top$	-	2-step	3,646
		$\Box_{[2,4]}(p_1 \rightarrow \Diamond_{[3,10]}\, p_2)$	2.0	0.7	7.46	$\bot$	2	2-step	47
		$\Box_{[0,10]}(p_1\, \mathbf{R}_{[0,\infty)}\, p_2)$	2.0	1.2	59.3	$\bot$	4	2-step	212
	Oscil	$\Diamond_{[0,3]}(p_1\, \mathbf{R}_{[0,\infty)}\, p_2)$	0.1	1.5	110	$\top$	-	2-step	289
		$\Diamond_{[2,5]}(\Box_{[0,3]}\, p_1)$	1.0	1.2	224	$\bot$	3	2-step	259
		$(\Box_{[1,3]}\, p_1)\, \mathbf{R}_{[2,5]}\, p_2$	0.1	1.2	266	$\bot$	3	2-step	266

Concluding Remarks

- Propose SMT-based bounded model checking algorithm for STL
- Propose robust STL model checking
- Propose several optimization techniques:

  two-step solving algorithm and the minimization of enumerated scenarios
- Developed a robust STL model checker **STLMC**
- Future work
  - Integrated with reachable-set computation methods
  - Extend the method to verify STL properties for unbounded time