Automating Safety Proofs about Cyber-Physical Systems using Rewriting Modulo SMT

Vivek Nigam[2,3], abd Carolyn Talcott[1]

[1]SRI International, USA
[2]Huawei Munich Research Center, Germany
[3]Federal University of Paraíba, Brazil

Cyber-Physical Systems

Cyber-Physical Systems (CPSs) are being used in the most varied domains to carry out autonomously different task.

Vehicle Platooning

Precision Agriculture

Package Delivery

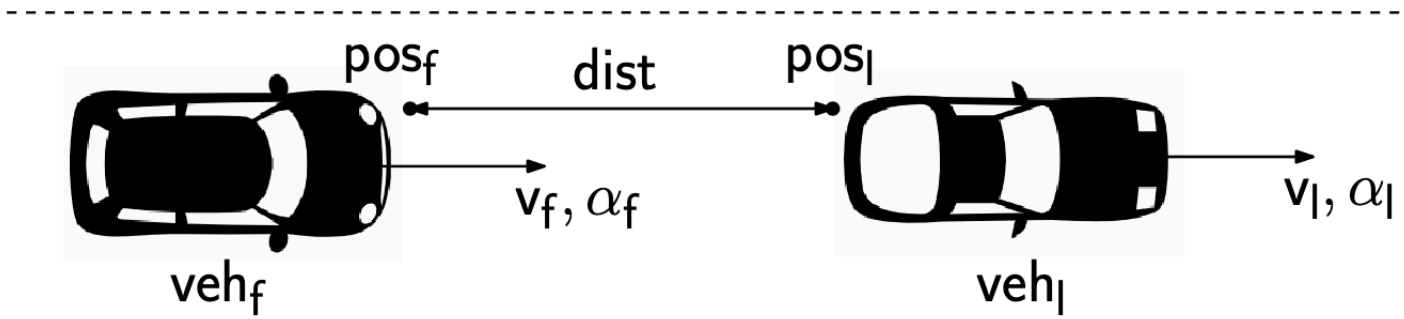
Autonomous Vehicles

Drone Taxi

Complexity Dimensions [Sifakis 2019]

CPSs are being used in many safety-critical applications with different levels of complexity.

Logical Scenario: Vehicle Following



Oper. Design Domain

Specifies the conditions for the logical scenario, e.g., speed bounds, maximum acceleration / deceleration.

$60km/h \leq v_f \leq 120km/h$

$-2m/s^2 \leq \alpha_f \leq 8m/s^2$

Key Challenge

Provide evidence that vehicles do not reach an unsafe situation for all instances of a logical scenario.

Safety Properties

- $P_{safer} := dist \geq v_f \times (1[s] + gap_{safer}) - v_l \times 1[s]$
- $P_{safe} := v_f \times (1[s] + gap_{safer}) - v_l \times 1[s]) > dist \geq v_f \times (1[s] + gap_{safe}) - v_l \times 1[s]$
- $P_{unsafe} := dist < v_f \times (1[s] + gap_{safe}) - v_l \times 1[s]$

Problems/Challenges

Autonomous CPS are based on ML.

These are **extremely** fragile. These systems present more failures than acceptable to safety-critical systems [Jha et al. SafeComp 2020].

Adaptive Control for Autonomous CPS.

To compensate the lack of direct human intervention, we advocate extensive use of **adaptive control techniques**. [Sifakis 2018].

Symbolic versus Enumerative.

The characterization of the effect of harmful events ... cannot be enumerative and exhaustive; it **should be symbolic and conservative**, the result of a global model-based analysis. [Sifakis 2018].

Safety Assurance Evidence

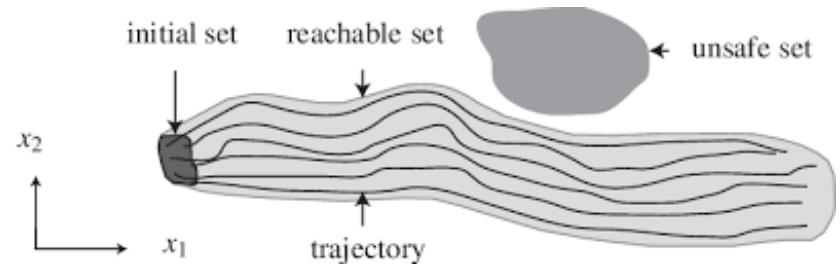
Main Existing Approaches

Simulation-based

Positive These methods can be used to validate concrete implementations, e.g., ML.

Negative requires to run a sufficiently large number of simulations and may miss corner-cases.

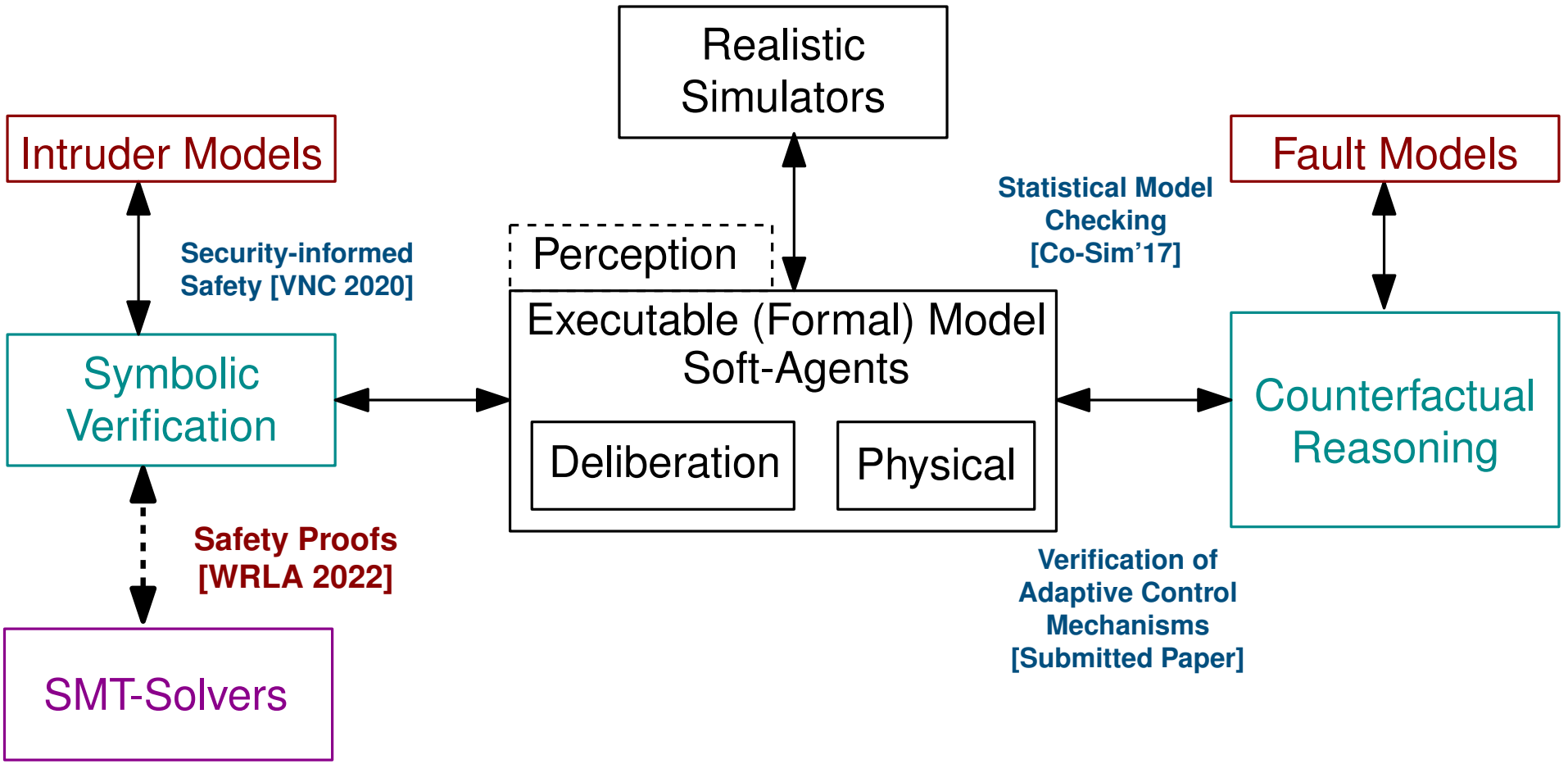
Verified Planners



Positive Planners are guaranteed to avoid unsafe conditions.

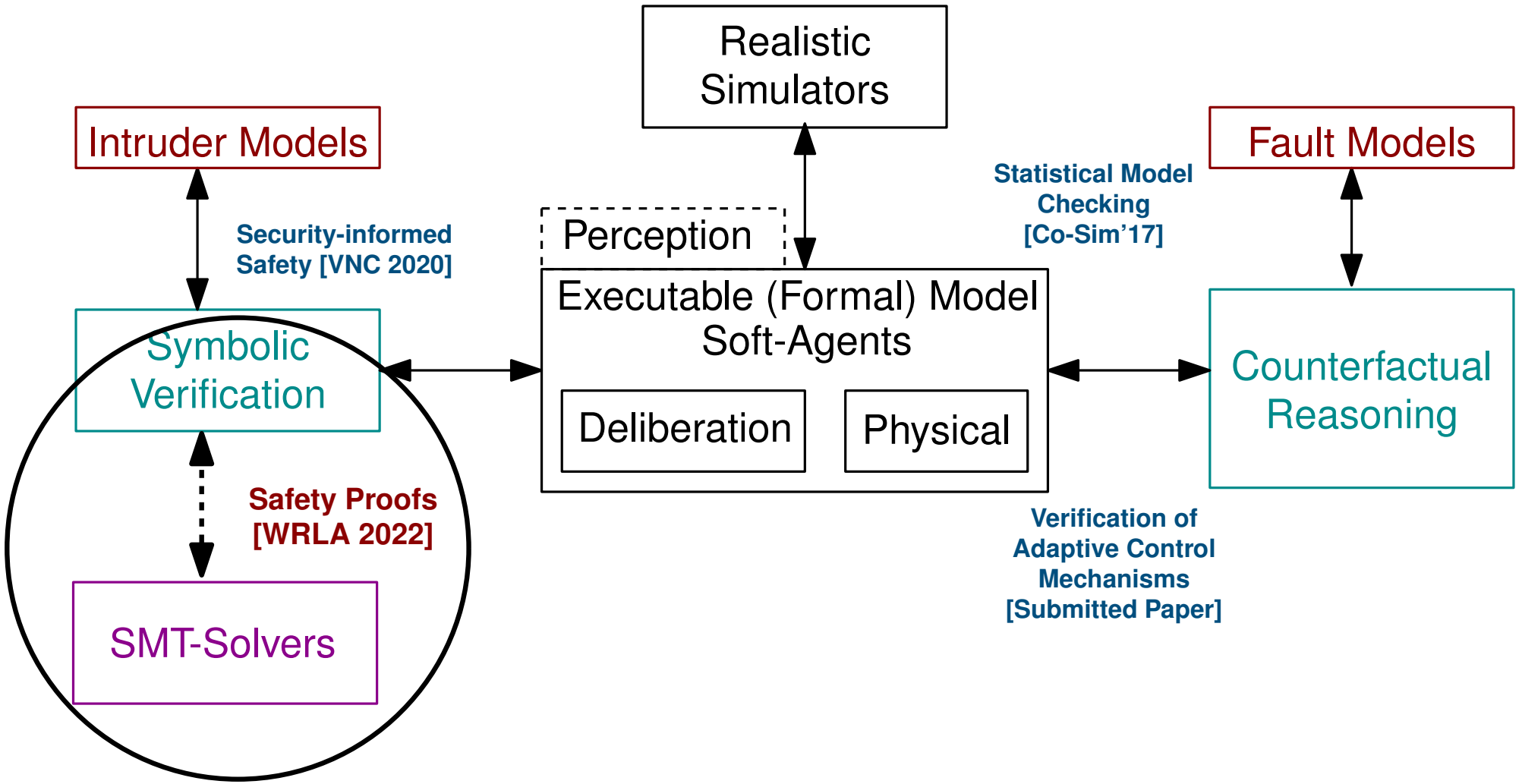
Negative algorithms have to be proved by hand. Moreover, they often do not consider other functions in the system, e.g., communication channels, and local knowledge bases.

Soft-Agents Framework



Soft-agents framework is implemented in Maude using Rewriting Logic.

Soft-Agents Framework



Soft-agents framework is implemented in Maude using Rewriting Logic.

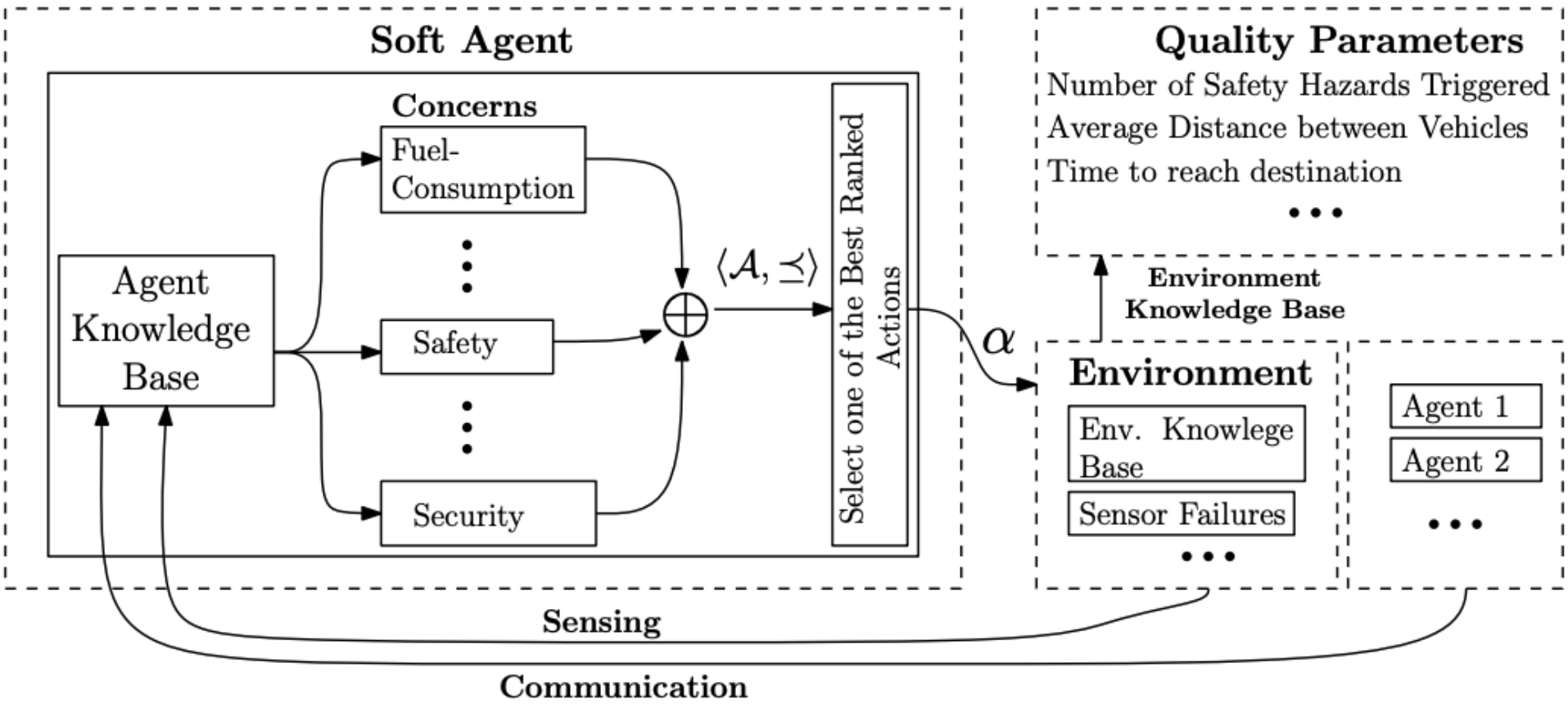
Contributions

Soft Agents Framework with Rewriting Modulo SMT:
Instead of using concrete values for speed, acceleration, etc, we enable the use of symbols constrained by (non-linear) theories.

Vehicle Platooning Specification:
We demonstrate the Soft Agent frame- work on the vehicle following scenario.

Verification Trade-off between Rewriting and Constraint Solving:
We investigate the trade-offs of delegating verification to Z3 and to Rewriting.

Soft-Agent: Overview



Symbolic Configurations

Example of symbols:

```
eq v1posx = vv(2,"ag1-positionX") . eq v1posy = vv(3,"ag1-positionY") .
eq v1vel = vv(5,"ag1-speed") .      eq maxacc1 = vv(9,"ag1-maxAcc") .
eq maxdec1 = vv(10,"ag1-maxDec") .  eq acc1 = vv(32,"ag1-acc") .
```

Example of constraints on symbols:

```
(acc1 <= maxacc1) and (acc1 >= maxdec1)
```

Library of Symbolic Functions:

```
op ldist : Nat Loc Loc -> NatSymTermBoolean .
eq ldist(i,loc(x0,y0),loc(x1,y1))
 = {s(i),vv(i,"dist"), (vv(i,"dist") >= 0/1) and
    vv(i,"dist") * vv(i,"dist") === ((y1 - y0) * (y1 - y0) +
                                     (x1 - x0) * (x1 - x0)) } .
```

non-linear constraint on the fresh symbol

fresh symbol

Symbolic Configurations

Local Knowledge Base: A set of possibly timestamped facts.

```
(at(ag1,loc(v1posx,v1posy)) @ 0) (speed(ag1,v1vel) @ 0)
(accel(ag1,acc1) @ 0)
(dir(v(1),loc(v1ix,v1iy),loc(v1tx,v1ty),v1mag) @ 0)
```

Configurations:

Environment agent specifies the actual situation.

Set of constraints on symbols.

```
asysI = { [eid | (kb constraint(i,condI))]
          [v(0) : veh | lkb : kb0 ] [v(1) : veh | lkb : kb1 ] }
```

Two agents with their local knowledge bases.

Semantics:

Advances time updating physical attributes. The symbol timeDuration specifies the duration of a logical tick.

$$\mathcal{A} \longrightarrow_{Tick} \mathcal{A}_1 \longrightarrow_{doTasks} \mathcal{A}_2 \cdots$$

Agents compute their next actions.

Safety Properties

Safety Properties:

```
op mkSPCond : SP ASystem -> SPSpec .
ceq mkSPCond(saferSP, { conf env }) = {k + 1,dis,cond00,nucond}
if [id0 | kb] := env
/\ (atloc(v(0),l0) @ t0) (atloc(v(1),l1) @ t1)
   (speed(v(0),v0) @ t2) (speed(v(1),v1) @ t3)
   (gapSafety(v(1),gapSafer,gapSafe)) (constraint(n,cond)) kb1 := kb
/\ {k,dis,cond00} := ldist(n,l1,l0)
/\ nucond := (dis >= ((1/1 + gapSafer) * v1) - v0) .
```

Based on the current environment KB, a constraint specifying the property is constructed.

```
op enforceSP : SP AS
```

Returns a configuration that satisfies a safety property.

Example of instance of a configuration satisfying saferSP

```
ag0-positionX |-> (0/1).Real,  ag0-positionY |-> (1/1).Real
ag1-positionX |-> (0/1).Real,  ag1-positionY |-> (0/1).Real,
ag0-speed |-> (7/1).Real,      ag1-speed |-> (2/1).Real,
ag1-safer |-> (3/1).Real
```

Verification Properties:

Starting from any safe configuration.

Bound search to two logical ticks.

```
search enforceSP(safeSP,setStopTime(asysI,2)) =>*
    asys such that checkSP(unsafeSP,asys) .
No solution. states: 63  rewrites: 394686 in 20134ms
```

Check whether an unsafe configuration is reachable.

Design Choices

More SMT, Less Rewriting

```
ceq symValSpeedRed(i,str,vmin,vmax,vminD,vmaxD,cond) =
{i + 2, [vv(i),vv(i + 1),nuCond and cond]}
if cond1 := vmin >= vmaxD
/\ cond11 := vv(i) === vmin and
             vv(i + 1) === ((vmin + vmax) / 2/1) and cond1
/\ cond2 := vmax <= vminD
/\ cond21 := vv(i) === ((vmin + vmax) / 2/1)
        and vv(i + 1)  === vmax and cond2
...
/\ cond6 := vmin >= vminD and vmax < vmaxD
/\ cond61 := vv(i) === vmin and vv(i + 1)  === max and cond6
/\ nuCond := (cond11 or cond21 or cond31 or cond41 or cond51 or cond61) .
```

Several cases for the agent's controller are incorporated in the constraints using an logical or. (Similar to Guarded Terms). This means that the SMT-solver will need to consider all cases.

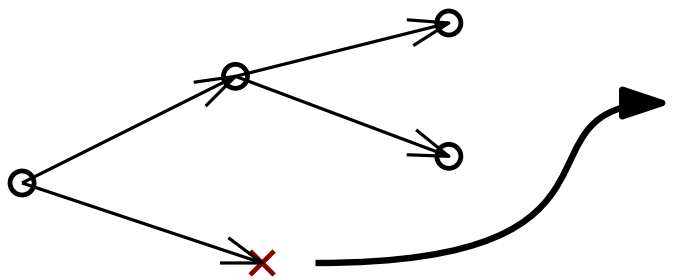
Less SMT, More Rewriting

```
ceq symValSpeedRed-Split(i,str,vmin,vmax,vminD,vmaxD,cond) =
  {i + 2, [vv(i),vv(i + 1),cond11 and cond]}
  {i + 2, [vv(i),vv(i + 1),cond21 and cond]}
  ...
  {i + 2, [vv(i),vv(i + 1),cond61 and cond]}
 if cond1 := vmin >= vmaxD
 ...
 /\ cond61 := vv(i) === vmin and vv(i + 1)  === vmax and cond6.
```

The different conditions are split into different terms. (Kind of the opposite of Guarded Terms.) This means that the rewriting engine will need to traverse these cases.

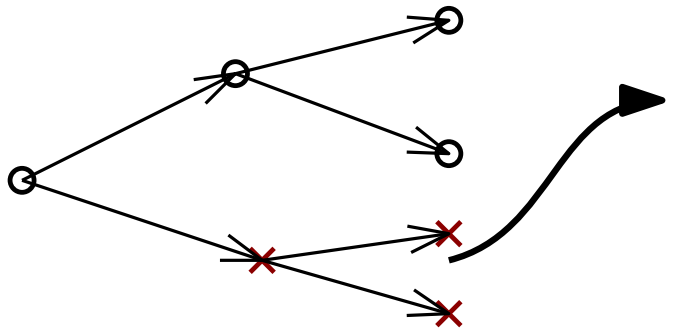
More Design Options

All Pruning



Search is interrupted whenever a configuration's constraints is unsat. This means that there more calls to the SMT-solver, but less states to traverse.

No Pruning



Search continues although a configuration's constraints is unsat. This means that there more less calls to the SMT-solver, but more states to traverse. The satisfiability of the configuration is only checked when the time bound is reached.

Tick Prunning

Satisfiability of configuration is checked after tick rules only. This leads to less calls to the SMT-solver and still prunes the search tree.

Experiments

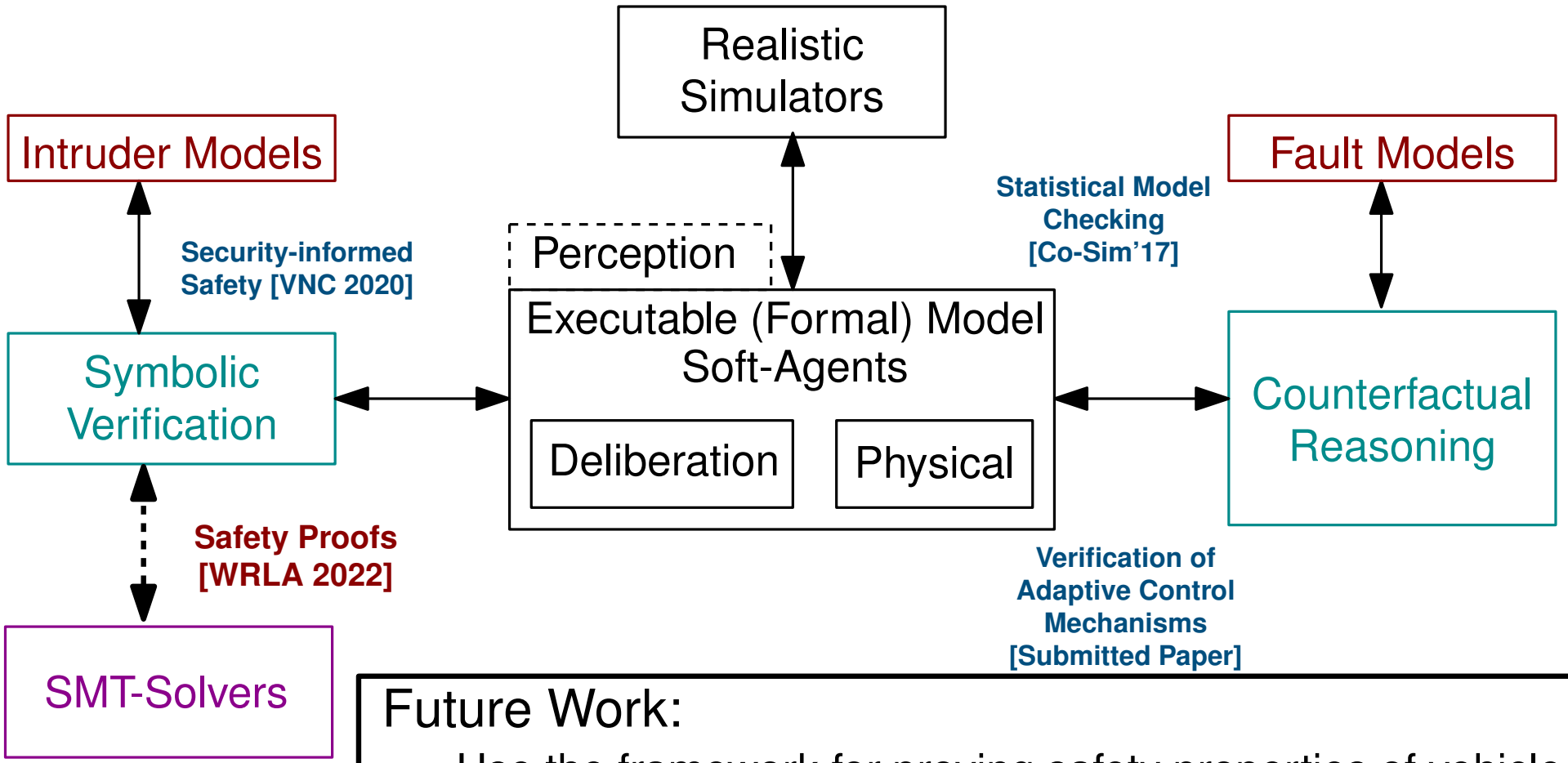
For simpler properties, the balanced case performs better due to the smaller search space and lower overhead in calling the SMT-solver.

Time Bound	Pruning	More SMT Less Search	Balanced	Less SMT More Search
2	No	19/20.4s	71/2.5s	1427/29.7s
	Tick	19/32.4s	63/8.3s	497/47.4s
	All	19/56.0s	63/11.6s	296/52.7s
3	No	DNF	DNF	42827/3054s
	Tick	DNF	DNF	2484/3412s
	All	DNF	DNF	1976/5238s

For more complicated properties, no pruning and less SMT leads to better results. We believe that all pruning may be improved if one can exploit the incremental verification available in SMT-solvers. Search does not yet support this.

DNF – aborted after 5h.

Soft-Agents Framework



Future Work:

- Use the framework for proving safety properties of vehicle scenarios, such as resilience properties (**see our ICTAC 2022 paper**).
- Research how to exploit the incremental features of SMT-solvers to improve search.