Automating Recoverability Proofs for Cyber-Physical Systems with Runtime Assurance Architectures

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

[1] SRI International, Menlo Park, USA, carolyn.talcott@gmail.com
[2] Federal University of Paraíba, João Pessoa, Brazil, vivek.nigam@gmail.com
[3] Huawei Munich Research Center, Germany

Abstract. Cyber-Physical Systems (CPSes), such as Autonomous Vehicles, make use of sophisticated components like ML-based controllers. It is difficult to provide evidence about the safe functioning of such components. To overcome this problem, Runtime Assurance Architecture (RTA) solutions have been proposed. The RTA's decision component evaluates the system's safety risk and whenever risk is higher than acceptable the RTA switches to a safety mode that, for example, activates a controller with strong evidence for its safe functioning. In this way, RTAs increase CPS runtime safety and resilience by recovering the system from higher to lower risk levels. The goal of this paper is to automate recovery proofs of CPSes using RTAs. We first formalize the key verification problems, namely, the decision sampling-time adequacy problem and the time-bounded recoverability problem. We then demonstrate how to automatically generate proofs for the proposed verification problems using symbolic rewriting modulo SMT. Automation is enabled by integrating the rewriting logic tool (Maude), which generates sets of non-linear constraints, with an SMT-solver (Z3) to produce proofs

1 Introduction

Cyber-physical systems (CPSes) are increasingly performing complex safety-critical missions in an autonomous fashion, Autonomous Vehicles (AVs) being a current prime example. Given the complexity of the environment in which such CPSes operate, they often rely on highly complex machine learning (ML) based controllers [1] because of ML's capability of learning implicit requirements about the vehicle operation conditions. It has been notably hard, however, to provide safety arguments using only such ML based components. Despite the great amount of effort in building methods for verifying (or providing evidence) about behavior of ML-based components, they still present more faults than acceptable [15].

Runtime assurance architectures (RTAs), based on the well-known Simplex Architecture [28,27], have been proposed [12,22,18] as a means to overcome this challenge. An RTA contains a decision module that, during runtime, evaluates the system's safety risk formalized as a collection of safety properties during design phase. Whenever a safety risk is higher than acceptable, the RTA moves the system to a safe state. As illustrated by Figure 1, RTA increases CPS safety and resilience by dynamically adapting the CPS behavior according to the perceived system risk level, recovering the CPS from a higher risk situation. We use the symbol $\mathtt{dt}$ to denote the sampling interval in which the

decision module evaluates the system's level of risk. These levels of risks are formalized as properties tailored according to the operational domain of the system [20]. For example, vehicles on the highway have a different formalization of risk level than vehicles in urban scenarios where pedestrians may be crossing roads. In the diagram in Figure 1 there are four increasing levels of risk (safer, safe, unsafe, bad), e.g., denoting risks of an accident, from safer denoting the lowest and desirable risk level to bad denoting the highest level of risk that has to be avoided at all costs, to avoid possible accidents. If the risk is safer, then the decision module uses the output from the primary, unverified controller. However, if a higher risk safe is detected, then the decision module uses the output of the safe controller. The expectation is then that the safe controller recovers eventually from the high risk situation leading the system to return to a situation that is safer. It may be that in the process the CPS will pass through situations that are unsafe, but it definitely shall not pass through situations that are bad, e.g., situations of imminent crash that trigger other safety mechanisms, such as emergency brakes.

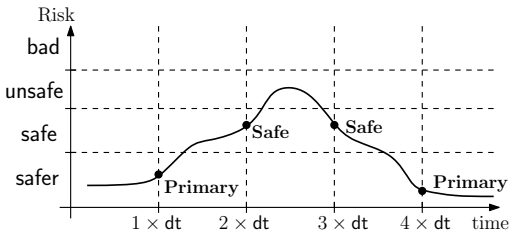

Fig. 1. Illustration of how one expect RTA to maintain safety during runtime. dt is the sampling time of the decision module. **Primary** (respectively, **Safe**) denotes that the decision module switches to the primary (respectively, safe) controller.

There are two key properties about RTAs which engineers have to demonstrate by providing sufficient evidence:

- dt **Adequacy:** the sampling time interval is small enough that bad situations are not missed by the RTA;
- **Time Bounded Recoverability:** if the system risk becomes greater than acceptable (safer) the safe controller can bring the system back to a safer state within a specified time bound, without entering a bad state.

The main goal of this paper is to develop methods to generate formal proofs for these properties for RTA instances in an automated fashion. This is accomplished by using the Symbolic Soft-Agents framework [20] which enables the automated generation of safety proofs for CPS using symbolic rewriting modulo SMT [24]. Our contributions here are in two areas:

- **Formal foundation.** We provide formal definitions for three variants of dt adequacy, and prove the relations among them. We also provide a formal definition of time bounded recoverability. We define a notion of one period recoverability, and prove that one period recoverability together with any one of the dt adequacy properties implies time bounded recoverability. The formal definitions are tailored so that they are amenable to automated verification.
- **Automated Checking of RTA Properties:** Based on the specification of RTA properties and of abstract descriptions of situations in which CPSes operate, called logical scenarios [23,19], we present algorithms for verifying two forms of dt adequacy

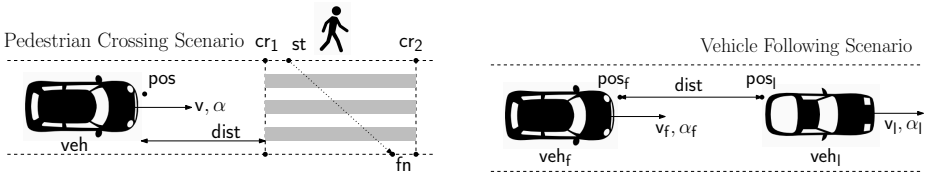

Fig. 2. Pedestrian Crossing and Vehicle Following Logical Scenarios Diagrams. The road is on the Y-axis, so imagine the illustrations rotated counterclockwise.

and for one period recoverability, and report results of experiments for two logical scenarios. The experiments demonstrate the feasibility of automated proof and also illustrate some of the challenges.

Section 2 describes the logical scenarios of our running examples. Section 3 formalizes the notion of levels of risk using safety properties. These are then used to define several notions of Sampling Time Adequacy in Section 4 and recoverability properties in Section 5. Section 6 describes experiments based on the logical scenarios in Section 2. We conclude with related and future work in Sections 7 and 8.

2 Logical Scenarios and Motivating Examples

A key step in the development of autonomous CPSes is the definition of the situations in which these systems will operate [23,19,32]. These situations are specified as abstract scenarios, called logical scenarios [23,19], such as lane changing or vehicle following or pedestrian crossing, in which an AV has to avoid harm. These logical scenarios contain details about the situations in which a vehicle shall be able to safely operate such as which types and number of actors, e.g., vehicles, pedestrians, operating assumptions, e.g., range of speeds, and road topology, e.g., number of lanes. Moreover, these logical scenarios are associated with safety metrics that formalize the properties that need to be satisfied by the vehicle. For a comprehensive list of logical scenarios and associated properties we refer to [32] and references therein. Examples of scenario description and generation formalisms can be found in [13,8]. As a logical scenario may have infinitely many concrete instances, it is challenging to demonstrate that a vehicle will satisfy such safety properties in all instances.

We use two running examples illustrated by the diagrams in Figure 2: a pedestrian crossing scenario and a vehicle following scenario.

Pedestrian Crossing In this scenario an ego vehicle, $\mathtt{vh}$, is at position pos and is approaching with speed v and acceleration α, with a pedestrian crossing situated between the positions cr_1 and cr_2. Moreover, a pedestrian is attempting to cross the road using the pedestrian crossing. As long as the pedestrian does not move outside the pedestrian crossing, the exact shape of the pedestrian crossing is not important as $\mathtt{vh}$ shall always stop before the pedestrian crossing whenever a pedestrian is intending to cross it. To keep things simple, assume that the pedestrian is crossing the street at constant speed, v_p, following a straight line as illustrated in Figure 2 by the dashed line from st to fn.

The Operational Design Domain (ODD) of such a logical scenario is specified by constraints on its parameters (pos, v, α, cr_1, cr_2, v_p). Typically, one specifies the bounds on the speeds and accelerations. Consider for example:

$2m/s \leq \mathsf{v} \leq 10m/s \quad -8m/s^2 \leq \alpha \leq 2m/s^2 \quad 1m/s \leq \mathsf{v}_\mathsf{p} \leq 4m/s$

Moreover, $\mathsf{pos}.y < \mathsf{cr}_1.y$, that is the vehicle is approaching the pedestrian crossing and $\mathsf{cr}_1.y \leq \mathsf{st}.y, \mathsf{fn}.y \leq \mathsf{cr}_2.y$, that is st, fn are in the pedestrian crossing area, where for any position $\mathsf{l} = (\mathsf{px}, \mathsf{py})$, $\mathsf{l}.x$ and $\mathsf{l}.y$ denote, respectively, px and py.

Vehicle Following Our second running example is a vehicle following scenario as depicted in Figure 2. This example commonly appears in the literature and therefore, we do not describe in the same level of detail, but simply refer to [20]. In a nutshell, it consists of two vehicles, a follower vehicle (veh_f) and a leader vehicle (veh_l). Typically, these vehicles are in a highway with multiple lanes at reasonably high speeds, e.g., speeds between $60km/h$ and $140km/h$ and the same acceleration bounds as in the vehicle in the pedestrian crossing scenario. Moreover, there are only vehicles, i.e., no pedestrians. no bicycles, etc. The following vehicle shall avoid approaching dangerously close to the leader vehicle while still maintaining a reasonable speed.

We assume that from an instance, conf, of a logical scenario (LS), we can compute the relation $\mathsf{conf} \longrightarrow_\Delta \mathsf{conf}_1$, where conf_1 is an LS instance specifying the physical attributes (speeds, directions, accelerations) of the agents in conf obtained according to their speeds, direction and accelerations in conf after a period of $\Delta > 0$ time units. For example, consider the instance of the pedestrian crossing scenario where the vehicle has speed of $10m/s$, acceleration of $2m/s^2$, and position $\mathsf{pos}.x = 0m$ After $\Delta = 0.1s$, the speed of the vehicle will be $10.2m/s$ and new position $1.1m$.

3 Safety Properties and Levels of Risk

A key aspect of RTA mechanisms is the ability to check for the level of risk of the system, e.g., whether it is safe or not. We formalize the notion of level of risk as a partial order on safety properties as follows:

Definition 1. *An RTA safety property specification for a logical specification* LS *is a tuple* $\langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *where*

- $\mathcal{S} = \{\mathsf{SP}_1, \ldots, \mathsf{SP}_n\}$ *is a finite set of safety properties;*
- $\prec_1 : \mathcal{S} \times \mathcal{S}$ *is an asymmetric binary relation over* $\mathcal{S}$*, where* $\mathsf{SP}_1 \prec_1 \mathsf{SP}_2$ *denotes that the safety property* SP_2 *specifies a less risky condition than the safety property* SP_1*. The pre-order* $\prec$ *is derived from* $\prec_1$ *by applying transitivity.*
- *the safety property* $\mathsf{bad} \in \mathcal{S}$ *is the least element of* $\prec$*, specifying the condition that shall be avoided, i.e., the highest risk*
- $\models$ *specifies when an instance* conf *of* LS *satisfies a property* $\mathsf{SP} \in \mathcal{S}$*, written* $\mathsf{conf} \models \mathsf{SP}$*. Moreover, we assume that if* $\mathsf{conf} \models \mathsf{SP}_1$ *and* $\mathsf{SP}_1 \prec \mathsf{SP}_2$ *or* $\mathsf{SP}_2 \prec \mathsf{SP}_1$*, then* $\mathsf{conf} \not\models \mathsf{SP}_2$*. That is any instance of a logical scenario can only satisfy one level of risk. We also assume that any instance of a logical scenario is at some level of risk, that is, for all instances* conf *of* LS*, there is at least one* SP *such that* $\mathsf{conf} \models \mathsf{SP}$.

The following two examples illustrate different options of safety properties for the pedestrian crossing and the vehicle following examples described in Section 2.

Example 1. Consider the pedestrian crossing shown in Figure 2. We define the following RTA safety property specification $\langle\{\mathsf{bad}, \mathsf{unsafe}, \mathsf{safe}, \mathsf{safer}\}, \prec_1, \mathsf{bad}, \models\rangle$ with $\mathsf{bad} \prec_1 \mathsf{unsafe} \prec_1 \mathsf{safe} \prec_1 \mathsf{safer}$ based on the Time to Zebra metric [32] [4]

$$\begin{aligned} \mathsf{safer} &:= \mathsf{dist} \geq \mathsf{dStop} + \mathsf{gap}_{\mathsf{safer}} * \mathsf{v} \\ \mathsf{safe} &:= \mathsf{dStop} + \mathsf{gap}_{\mathsf{safer}} * \mathsf{v} > \mathsf{dist} \geq \mathsf{dStop} + \mathsf{gap}_{\mathsf{safe}} * \mathsf{v} \\ \mathsf{unsafe} &:= \mathsf{dStop} + \mathsf{gap}_{\mathsf{safe}} * \mathsf{v} > \mathsf{dist} \geq \mathsf{dStop} + \mathsf{gap}_{\mathsf{unsafe}} * \mathsf{v} \\ \mathsf{bad} &:= \mathsf{dStop} + \mathsf{gap}_{\mathsf{unsafe}} * \mathsf{v} > \mathsf{dist} \end{aligned} \tag{1}$$

where $\mathsf{dist} = \mathsf{cr}_1.y - \mathsf{pos}.y$ is the distance between the ego vehicle and the pedestrian crossing, $\mathsf{dStop} = -(\mathsf{v} * \mathsf{v})/(2 * \mathsf{maxDec})$ is the distance necessary to stop the ego vehicle by applying its maximum deceleration maxDec, e.g., when issuing an emergency brake, and $\mathsf{gap}_{\mathsf{safer}} > \mathsf{gap}_{\mathsf{safe}} > \mathsf{gap}_{\mathsf{unsafe}} > 0$ are used with v to specify a safety margin distance in the safety property. The values for $\mathsf{gap}_{\mathsf{safer}}, \mathsf{gap}_{\mathsf{safe}}, \mathsf{gap}_{\mathsf{unsafe}}$ shall be defined according to the ego vehicle's capabilities, e.g., the sampling time dt, and the ODD specifications, e.g., bounds on acceleration and speed. It is then straightforward to check whether an instance of a pedestrian logical scenario satisfies ($\models$) any one of the properties above.

While this may seem like a good candidate safety property specification for the pedestrian crossing, it turns out that it is hard to demonstrate vehicle recoverability as we show in Section 6. The problem lies in the fact that the three properties tend to be all the same when the vehicle speed (v) tends to zero, and similarly, when dist is too large. We, therefore, establish an alternative definition for safer as follows:

$$\mathsf{safer} := \mathsf{dist} \geq \mathsf{dStop} + \mathsf{gap}_{\mathsf{safer}} * \mathsf{v} \text{ or } \mathsf{v} \leq \mathsf{lowSpd} \text{ or } \mathsf{dist} \geq \mathsf{farAway} \tag{2}$$

where lowSpd and $\mathsf{farAway}$ are constants specifying a maximum speed for which the vehicle is very safe, e.g., the speed lowSpd is less than the speed of a pedestrian, and the distance $\mathsf{farAway}$ that is far enough from the pedestrian crossing.

Example 2. One well-known example for vehicle safety assurance for the vehicle following scenario is the Responsibility-Sensitive Safety (RSS) [29,32] safe distance metric. The RSS safety distance $\mathsf{drss}(\mathsf{react})$ is specified as follows:

$$\mathsf{drss}(\mathsf{react}) = \mathsf{v} \times \mathsf{react} + \frac{\mathsf{maxacc}_f \times \mathsf{react}^2}{2} - \frac{(\mathsf{v} + \mathsf{maxacc}_f \times \mathsf{react})^2}{2 \times \mathsf{maxdec}_f} - \frac{\mathsf{v}_l^2}{2 \times \mathsf{maxdec}_l}$$

where react is a parameter for the time for the vehicle to react; v and v_l are, respectively, the follower and leader vehicle speeds; maxacc_f is the maximum acceleration of the follower vehicle; and maxdec_f and maxdec_l are, respectively, the maximum deceleration of the follower and leader vehicles. Based on $\mathsf{drss}(\mathsf{react})$ two properties are defined: bad when $\mathsf{dis} < \mathsf{drss}$ and safer otherwise.

As RSS has only two properties, the definition of recoverability using RTA implies that the system must always satisfy the safer property; otherwise it must satisfy bad. This means that the primary controller shall be trusted and that RTA is not necessary from the beginning (and probably not desired as the primary controller is assumed not to be verified). It is possible to adapt the RSS definitions by adding additional levels in

[4] Zebra is the pedestrian crossing zone.

between safer and bad based on the react time:

$$\mathsf{safer} := \mathsf{dis} \geq \mathsf{drss}(3 \times \mathsf{dt}) \qquad \mathsf{safe} := \mathsf{drss}(2 \times \mathsf{dt}) \leq \mathsf{dis} < \mathsf{drss}(3 \times \mathsf{dt})$$
$$\mathsf{unsafe} := \mathsf{drss}(\mathsf{dt}) \leq \mathsf{dis} < \mathsf{drss}(2 \times \mathsf{dt}) \qquad \mathsf{bad} := \mathsf{dis} < \mathsf{drss}(\mathsf{dt})$$

Intuitively, when a vehicle is in a configuration satisfying safer it can wrongly evaluate safety risk, e.g., due to distance sensor errors, for two cycles before the RSS property is invalidated. Similarly, safe it can evaluate wrongly for one cycle and unsafe it always has to evaluate correctly the risk.

4 Sampling Time (dt) Adequacy

The RTA monitor has to detect when the system risk changes, and even more so when risk increases, that is, when systems satisfy properties SP that are closer to bad, i.e., move lower in the order $\prec$. This means that the sampling time dt plays an important role in the correctness of a RTA system. For example, if the sampling time is $4 \times \mathsf{dt}$ in Figure 1, the RTA monitor may fail to detect elevation of risk from safer to safe thus not activating the trusted controller soon enough to avoid further escalation of risk.

There is a trade-off between the ability of the system to detect changes of risk and therefore its ability to quickly react to changes, and the performance requirements of monitor system in determining risk, i.e., dt time. The lower the dt, the greater is the ability of the system to detect changes and also greater are the performance requirements on the monitoring components.

Moreover, a key challenge is that dt shall be appropriate in detecting risk changes for all instances of the ODD, i.e., all possible instances of speeds and accelerations. Our approach is to use SMT-solvers to generate dt adequacy proofs automatically building on ideas in [20]. Depending on the definition of dt adequacy, the complexity of the problem can increase substantially, making automation difficult or not feasible.

We propose three alternative definitions of requirements on dt, defined below, that illustrate the trade-offs between the capability of the system to detect risk changes and the development and verification efforts. Figure 3 illustrates these definitions. The first definition, called *one transition adequacy*, is illustrated by left-most diagram in Figure 3. Intuitively, this definition states that the dt shall be fine enough to detect whenever the configuration of the scenario evolves from satisfying a property, SP_1, to satisfying another property, SP_2. As an example, the dotted evolution of the system passing through conf'_d contains multiple property changes within a period of dt.

Definition 2. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \vDash \rangle$ *be a RTA safety property specification for a logical scenario* LS*; and* dt *be a sampling time.* dt *is* one transition adequate *with respect to* $Spec$ *and* LS *if for all instances* conf, conf_1 *of* LS *such that* $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *we have:*

- $\mathsf{conf} \vDash \mathsf{SP}_1$ *and* $\mathsf{conf}_1 \vDash \mathsf{SP}_2$*, then there is a decomposition* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}_d \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *of* $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *for some* $0 \leq \mathsf{dt}' < \mathsf{dt}$*, such that:*
  - *For all decompositions of* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}_d$ *as* $\mathsf{conf} \rightarrow_{\mathsf{dt}_2} \mathsf{conf}_2 \rightarrow_{\mathsf{dt}'-\mathsf{dt}_2} \mathsf{conf}_d$ *where* $0 < \mathsf{dt}_2 < \mathsf{dt}'$*, we have that* $\mathsf{conf}_2 \vDash \mathsf{SP}_1$*;*
  - *For all decompositions of* $\mathsf{conf}_d \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *as* $\mathsf{conf}_d \rightarrow_{\mathsf{dt}_3} \mathsf{conf}_3 \rightarrow_{\mathsf{dt}-\mathsf{dt}'-\mathsf{dt}_3} \mathsf{conf}_1$ *where* $0 \leq \mathsf{dt}_3 < \mathsf{dt} - \mathsf{dt}'$*, we have that* $\mathsf{conf}_3 \vDash \mathsf{SP}_2$.

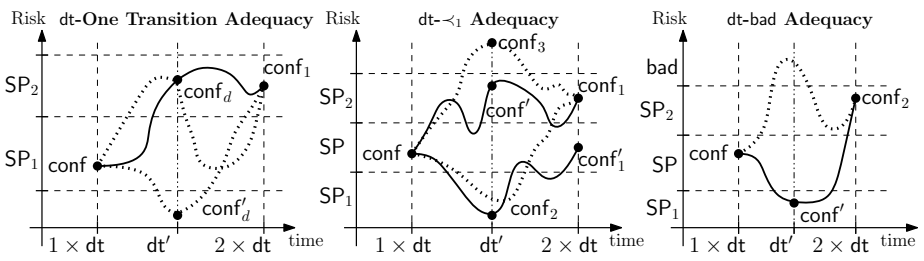

Fig. 3. Illustration of dt adequacy properties. Full line system evolutions illustrate allowed evolutions and dotted evolutions illustrate not allowed evolutions.

The following proposition follows immediately from Definition 2. It states that if dt is one transition adequate, then to check that a configuration satisfying bad is not reachable, it is enough to check whether the configurations during sampling are not bad, instead of checking all decompositions.

Proposition 1. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a RTA safety property specification for a logical scenario* LS. *Let* dt *be one-transition-adequate w.r.t.* $Spec$. *For all decompositions* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *of* LS, $\mathsf{conf}' \models \mathsf{bad}$ *if and only if* $\mathsf{conf} \models \mathsf{bad}$ *or* $\mathsf{conf}_1 \models \mathsf{bad}$.

Definition 2 is rather complex involving many quantifier alternations thus being very difficult to generate proofs for. In fact, due to limitations on computing time, it is not always possible to guarantee that dt can satisfy one-transition-adequate. Therefore, we propose two alternative definitions of weaker properties illustrated by the center and right-most diagrams in Figure 3. These properties are amenable to the automated generation of proofs as we detail in Section 6.

The first alternative definition is $\prec_1$ adequacy. Instead of requiring dt to be fine enough to detect when the system satisfies different properties, $\prec_1$ adequacy allows system evolution to migrate within $\prec_1$ range of a safety property multiple times, as illustrated by the system evolution passing through conf'. The system shall be able to detect whenever the risk of the system increases at least two levels.

Definition 3. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a RTA safety property specification for a logical scenario* LS*; and* dt *be a sampling time.* dt *is* $\prec_1$ adequate *with respect to* $Spec$ *and* LS *if for all instances* conf *of* LS *and relations* $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *if for all* $0 < \mathsf{dt}' \leq \mathsf{dt}$ *and decompositions* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *we have:*

- *If* $\mathsf{conf} \models \mathsf{SP}_1$ *and* $\mathsf{conf}_1 \models \mathsf{SP}_2$ *for* $\mathsf{SP}_1 \neq \mathsf{SP}_2$, *then* $\mathsf{conf}' \models \mathsf{SP}_1$ *or* $\mathsf{conf}' \models \mathsf{SP}_2$.
- *If* $\mathsf{conf} \models \mathsf{SP}$ *and* $\mathsf{conf}_1 \models \mathsf{SP}$, *then* $\mathsf{conf}' \models \mathsf{SP}$ *or* $\mathsf{conf}' \models \mathsf{SP}'$ *where* $\mathsf{SP}' \prec_1 \mathsf{SP}$ *or* $\mathsf{SP} \prec \mathsf{SP}'$.

One can generalize the definition of $\prec_1$ to allow evolutions on larger ranges of safety properties, e.g.$\prec_n$ adequacy for $n \geq 1$ allow evolutions within n safety risk levels.

The following property of $\prec_1$-adequacy provides a basis for defining recoverability based on $\prec_1$-adequate dt. It is enough to check that no configuration satisfying bad or a property immediately greater to bad is reachable.

Proposition 2. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a RTA safety property specification for a logical scenario* LS*; and* dt *be* $\prec_1$ *adequate sampling time. If* $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *with* $\mathsf{conf} \models \mathsf{SP}_1$ *and* $\mathsf{conf}_1 \models \mathsf{SP}_2$ *where* $\mathsf{SP}_1 \neq \mathsf{bad}$ *and* $\mathsf{SP}_2 \neq \mathsf{bad}$ *and* $\mathsf{bad} \not\prec_1 \mathsf{SP}_1$ *or* $\mathsf{bad} \not\prec_1 \mathsf{SP}_2$*, then for all* $0 < \mathsf{dt}' \leq \mathsf{dt}$ *and decompositions* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *we have* $\mathsf{conf}' \not\models \mathsf{bad}$.

Consider for example the safety property specification in Example 1 and assume that dt is $\prec_1$-adequate. From Proposition 2, if there is no transition $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ where $\mathsf{conf} \models \mathsf{unsafe}$ and $\mathsf{conf}_1 \models \mathsf{unsafe}$, then we can guarantee that the system does not pass through a configuration conf' with $\mathsf{conf}' \models \mathsf{bad}$ including the intermediate configurations that have not been sampled by the vehicle system.

The next adequacy only requires that the dt is fine enough to detect when a system evolution satisfies the bad property. As illustrated by the right-most diagram in Figure 3, the dotted evolution satisfying bad within dt would invalidate dt adequacy.

Definition 4. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a RTA safety property specification for a logical scenario* LS*; and* dt *be a sampling time.* dt *is* bad-adequate *with respect to* $Spec$ *and* LS *if for all instances* conf *of* LS *and* $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *if for all* $0 < \mathsf{dt}' \leq \mathsf{dt}$ *and decompositions* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *we have:*
– *if* $\mathsf{conf} \not\models \mathsf{SP}$ *and* $\mathsf{conf}_1 \not\models \mathsf{SP}$ *with* $\mathsf{SP} = \mathsf{bad}$ *or* $\mathsf{bad} \prec_1 \mathsf{SP}$*, then* $\mathsf{conf}' \not\models \mathsf{bad}$.

The following proposition is similar to Proposition 1 establishing the conditions for verifying for bad-adequacy.

Proposition 3. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a RTA safety property specification for a logical scenario* LS*. Let* dt *be* bad*-adequate w.r.t.* $Spec$*. For all decompositions* $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ *of* LS*,* $\mathsf{conf}' \models \mathsf{bad}$ *if and only if* $\mathsf{conf} \models \mathsf{SP}_0$ *and* $\mathsf{conf}_1 \models \mathsf{SP}_1$ *with* $\{\mathsf{SP}_0, \mathsf{SP}_1\} \subseteq \{\mathsf{bad}\} \cup \{\mathsf{SP} \mid \mathsf{bad} \prec_1 \mathsf{SP}\}$.

The following proposition establishes relations between the different adequacy definitions. Our experiments show that it is possible for dt to be bad-adequate and not $\prec_1$-adequate.

Proposition 4. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \models \rangle$ *be a safety property specification for a logic scenario* LS *and* dt *a sampling time.*
– *If* dt *is one transition adequate, then* dt *is* $\prec_1$*-adequate and* dt *is* bad*-adequate.*
– *If* dt *is* $\prec_1$*-adequate then* dt *is* bad*-adequate.*

5 RTA-based Recoverability Properties

There are many informal definitions of resilience [2,4,5,16]. In the broadest sense, resilience is "the ability of a system to adapt and respond to changes (both in the environment and internal)" [5]. NIST [25] provides a more precise, but still informal definition of resilience and more focused on attacks: "The ability to anticipate, withstand, recover, and adapt to adverse conditions, stresses, attacks or compromises on systems that use or are enabled by cyber resources."

Intuitively, systems, such as an autonomous vehicle in an LS instance, implementing RTA can be shown to exhibit a basic form of resilience we refer to as recoverability: they detect when a specified risk level is reached and adapt to reduce the risk. Our goal is to formalize this intuition of RTA recoverability with precise definitions. To accomplish this, we augment the semantic relation $\rightarrow_{\mathsf{dt}}$ which models the physical aspect of behavior with a relation $\rightarrow_{\mathsf{tasks}}$ that models the control aspect, typically sensing, analyzing, and deciding/planning. Formally, the system behavior is a set of (possibly infinite) execution traces:

$$\mathsf{conf}_0 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_0 \rightarrow_{\mathsf{dt}} \mathsf{conf}_1 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_1 \rightarrow_{\mathsf{dt}} \mathsf{conf}_2 \rightarrow_{\mathsf{tasks}} \cdots$$

where dt is the system's sampling time, $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}} \mathsf{conf}_{i+1}$ is as before, and $\mathsf{conf}_i \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_i$ is an internal transition specifying the behavior of the agents in conf_i, e.g., sensing, updating local knowledge bases, and deciding which actions to take. The exact definition of this transition depends on system specification. Since safety properties are related to the physical attributes of the system, e.g., speed, location, we normally assume that if $\mathsf{conf}_i \vDash \mathsf{SP}$, then also $\mathsf{conf}'_i \vDash \mathsf{SP}$. For example, this is the case with the safety properties in Example 1. This assumption is not strictly necessary as the definitions below can be extended to cover cases when this assumption does not hold.

Definition 5. *Let* $Spec = \langle \mathcal{S}, \prec_1, \mathsf{bad}, \vDash \rangle$ *be a safety property specification for a logical scenario* LS *and* dt *a sampling time, where* $\mathsf{SP}_{\mathsf{safe}} \in \mathcal{S}$ *is the minimal acceptable safe property and* $\mathsf{SP}_{\mathsf{safer}} \in \mathcal{S}$ *is the acceptable safer property where* $\mathsf{SP}_{\mathsf{safe}} \prec \mathsf{SP}_{\mathsf{safer}}$. *Let* t *be a positive natural number. A system* S *is* $\langle \mathsf{SP}_{\mathsf{safe}}, \mathsf{SP}_{\mathsf{safer}}, t \rangle$*-recoverable if for all instances* conf_0 *of* LS *and traces* $\tau = \mathsf{conf}_0 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_0 \rightarrow_{\mathsf{dt}} \mathsf{conf}_1 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_1 \rightarrow_{\mathsf{dt}} \cdots$ *such that* $\mathsf{conf}_0 \vDash \mathsf{SP}$ *with* $\mathsf{SP} = \mathsf{SP}_{\mathsf{safe}}$ *or* $\mathsf{SP}_{\mathsf{safe}} \prec \mathsf{SP}$*:*

- *For all* $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}} \mathsf{conf}_{i+1}$ *in* τ*, there is no decomposition* $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}_1} \mathsf{conf} \rightarrow_{\mathsf{dt}-\mathsf{dt}_1} \mathsf{conf}_{i+1}$*, with* $0 \leq \mathsf{dt}_1 \leq \mathsf{dt}$*, such that* $\mathsf{conf} \vDash \mathsf{bad}$*. That is, the system never reaches a configuration that satisfies* bad*.*
- *For all* conf_i *in* τ*, such that* $\mathsf{conf}_i \vDash \mathsf{SP}_{\mathsf{safe}}$*, then* $\mathsf{conf}_{i+t} \vDash \mathsf{SP}$ *with* $\mathsf{SP}_{\mathsf{safer}} \prec \mathsf{SP}$ *or* $\mathsf{SP}_{\mathsf{safer}} = \mathsf{SP}$*. That is, if the system reaches the minimal safe property, it necessarily returns to the acceptable safer property.*

This definition formalizes the ability of the system to recover from a higher level of risk as illustrated by Figure 1. Intuitively, the property $\mathsf{SP}_{\mathsf{safe}}$ specifies the highest acceptable risk before the system shall react to reduce risk, i.e., when the RTA instance triggers the safe controller, while $\mathsf{SP}_{\mathsf{safer}}$ specifies the risk that shall be achieved within t logical ticks of the system, i.e., $t \times \mathsf{dt}$, that is when the RTA instance resumes using the output of the primary controller.

There are some subtleties in this definition that are worth pointing out:

- Recovery Period: The time t in Definition 5 specifies the time that the system has to recover. On the one hand, it avoids that the system stays in a higher risk situation, albeit still safe, for a long period of time, thus reducing the chance of safety accidents. On the other hand, if t is too small, it will require a stricter safe controller or not be realizable given the vehicle's capabilities, e.g., maximum deceleration. Therefore, the value of t will depend on situation under consideration. To mitigate this problem, we propose automated ways to prove recoverability in Section 6.
- Recoverability Smoothness: Notice that we require that $\mathsf{SP}_{\mathsf{safe}} \prec \mathsf{SP}_{\mathsf{safer}}$ and not $\mathsf{SP}_{\mathsf{safe}} \prec_1 \mathsf{SP}_{\mathsf{safer}}$, i.e., $\mathsf{SP}_{\mathsf{safer}}$ can be multiple levels of risk safer than $\mathsf{SP}_{\mathsf{safe}}$. By

selecting appropriately these properties, e.g., setting $\mathsf{SP}_{\mathsf{safer}}$ with a much lower risk than $\mathsf{SP}_{\mathsf{safe}}$, one can avoid the oscillation of the system between normal operation (using the primary controller) and recovery operation (using the safe controller).

Procedure to Demonstrate Recoverability A challenge in proving a system resilient as per Definition 5 is that one needs to reason about all traces which may have infinite length and furthermore all decomposition of traces. To address this challenge we demonstrate (Theorem 1 below) that it is enough that the dt is adequate (as in Section 3), dt is fine enough not to skip properties (Definition 7 below), and consider only traces of bounded size as specified by the following definition:

Definition 6. *Let* $Spec, \mathsf{LS}, \mathsf{SP}_{\mathsf{safer}}, \mathsf{SP}_{\mathsf{safe}}, t$ *be as in Definition 5 and* dt *be the sampling time. A system* S *is* $\langle \mathsf{SP}_{\mathsf{safe}}, \mathsf{SP}_{\mathsf{safer}}, t \rangle$*−one-period-recoverable if for all traces* $\tau = \mathsf{conf}_0 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_0 \rightarrow_{\mathsf{dt}} \mathsf{conf}_1 \rightarrow_{\mathsf{tasks}} \cdots \rightarrow_{\mathsf{dt}} \mathsf{conf}_t$ *such that* $\mathsf{conf}_0 \vDash \mathsf{SP}_{\mathsf{safe}}$*:*

1. $\mathsf{conf}_t \vDash \mathsf{SP}_{\mathsf{safer}}$*–the system recovers in* t *time ticks to a lower risk situation.*
2. *For all* $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}} \mathsf{conf}_{i+1}$ *in* τ*, there is no decomposition* $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}_1} \mathsf{conf} \rightarrow_{\mathsf{dt}-\mathsf{dt}_1} \mathsf{conf}_{i+1}$*, with* $0 \leq \mathsf{dt}_1 \leq \mathsf{dt}$*, such that* $\mathsf{conf} \vDash \mathsf{bad}$.

To prove recoverability for unbounded traces (Theorem 1), we also need to ensure that property $\mathsf{SP}_{\mathsf{safe}}$ that triggers an RTA is not skipped. This is formalized by the following definition.

Definition 7. *Let* $Spec, \mathsf{LS}, \mathsf{SP}_{\mathsf{safer}}, \mathsf{SP}_{\mathsf{safe}}, t$ *be as in Definition 5 and* dt *be the sampling time. We say that* dt *does not skip a property* $\mathsf{SP}_{\mathsf{safe}}$ *if there is no transition of the form* $\mathsf{conf} \longrightarrow_{\mathsf{dt}} \mathsf{conf}_1$ *such that* $\mathsf{conf} \vDash \mathsf{SP}$ *and* $\mathsf{conf}_1 \vDash \mathsf{SP}_1$ *with* $\mathsf{SP}_{\mathsf{safe}} \prec \mathsf{SP}$ *and* $\mathsf{SP}_1 \prec \mathsf{SP}_{\mathsf{safe}}$.

Theorem 1. *Let* dt *be one-transition or* $\prec_1$ *or* bad*-adequate where* dt *does not skip* $\mathsf{SP}_{\mathsf{safe}}$*. A system* S *is* $\langle \mathsf{SP}_{\mathsf{safe}}, \mathsf{SP}_{\mathsf{safer}}, t \rangle$*-one-period-recoverable if and only if* S *is* $\langle \mathsf{SP}_{\mathsf{safe}}, \mathsf{SP}_{\mathsf{safer}}, t \rangle$*-recoverable.*

Condition for checking one-recovery-period recoverability: Even when considering only one-recovery-period recoverability, it is still necessary to consider all possible decompositions of dt transitions (item 2 in Definition 6). This can be overcome depending on the type of dt adequacy: using Propositions 1, 2, and 3, it is enough to check that that all configurations conf_i for $0 \leq i \leq t$ do not satisfy bad nor a SP such that $\mathsf{bad} \prec_1 \mathsf{SP}$.

6 Experimental Results

We carried out a collection of experiments using the symbolic soft agents framework [20] and symbolic rewriting modulo SMT as described in Section 6.1. Section 6.2 describes the experiments for automatically proving dt-adequacy. Section 6.3 describes the experiments for automatically proving timed recoverability. We used a value of $\mathsf{dt} = 0.1s$ for all experiments. If an answer has not been returned after one hour, an experiment is aborted. All experiments were carried out on a 2.2 GHz 6-Core Intel Core i7 machine with 16 GB memory. The code is available in the folder rta_symbolic_agents at https://github.com/SRI-CSL/VCPublic.

We considered the scenarios described as follows:

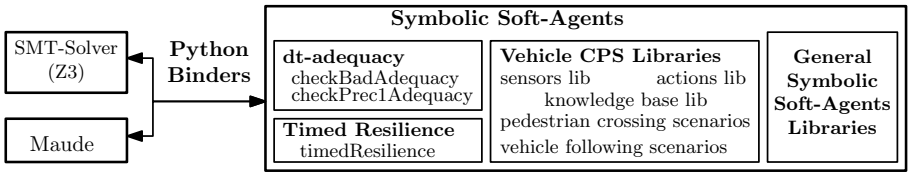

Fig. 4. Key libraries and tools used for automating recoverability proofs.

– pedCross($\mathsf{gap}_{\mathsf{safer}}$,$\mathsf{gap}_{\mathsf{safe}}$,$\mathsf{gap}_{\mathsf{unsafe}}$,senerr) – Pedestrian Crossing using only Relative Distances: This scenario is the pedestrian crossing scenario described in Section 2. The safety properties of the scenario are those as described in Equation 1 using only relative distances and parametrized by the values $\mathsf{gap}_{\mathsf{safer}}$, $\mathsf{gap}_{\mathsf{safe}}$, $\mathsf{gap}_{\mathsf{unsafe}}$. We assume that the sensor that detects pedestrians and their properties, namely, speed, position and direction, may not be perfect. That is, the vehicle's local knowledge base, used to decide which action it will take, may not correspond to the ground truth. In particular, the position of the pedestrian inferred by the vehicle may differ by some amount proportional to the actual distance to the pedestrian.
The error, err, is proportional to the distance ($\mathsf{pos}_p - \mathsf{pos}$) between the vehicle and the pedestrian as specified by the formula

$$\mathsf{err} \leq (\mathsf{pos}_p - \mathsf{pos}) \times \mathsf{senerr} \text{ and } \mathsf{err} \geq 0.$$

In this case the safe controller of the vehicle is conservative, e.g., reducing the speed of the vehicle more aggressively, so to still satisfy the timed recoverability property. When $\mathsf{senerr} = 0$, then the sensors are not faulty.
– pedCrBnds($\mathsf{gap}_{\mathsf{safer}}$,$\mathsf{gap}_{\mathsf{safe}}$,$\mathsf{gap}_{\mathsf{unsafe}}$,senerr) – Pedestrian Crossing with safer specified using low speeds and great distances: This is similar to the previous case, but now we are using the safety property for safer as specified by Equation 2.
– folRSS(maxdec_l) – Vehicle Following with RSS Properties: This scenario involves the vehicle following scenario using the safety properties based on the RSS property [29] described in Example 2. We parametrize the safety property according to the assumed maximum deceleration of the leader (maxdec_l). We follow the analysis carried out in [17]. This work identifies three scenarios based on the expected occurrence of leader vehicle deceleration. The first scenario, which is highly unlikely, is that the leader makes an emergency brake ($\mathsf{maxdec}_l = -8m/s^2$); the second when the leader vehicle decelerates heavily ($\mathsf{maxdec}_l = -5m/s^2$); and the most likely case when the leader vehicle decelerates normally ($\mathsf{maxdec}_l = -2m/s^2$).
– folGap($\mathsf{gap}_{\mathsf{safer}}$,$\mathsf{gap}_{\mathsf{safe}}$,$\mathsf{gap}_{\mathsf{unsafe}}$) – Vehicle Following with Gap Distances Properties: This scenario is described in more detail in [20]. In particular, we use safety properties based gap distances, similar to the pedestrian crossing.

6.1 Automating Recoverability Proofs using Symbolic Soft-Agents

Figure 4 depicts the main machinery that has been implemented and used. It is based on the soft-agents framework [30] and the general symbolic libraries described in [20]. The general symbolic soft-agents libraries specify the executable semantics of CPS

based on rewriting rules. The symbolic soft-agents rewrite rules correspond directly to the two LS relations $\rightarrow_{\mathsf{tasks}}$ and $\rightarrow_{\mathsf{dt}}$. We implemented the vehicle-specific libraries for specifying vehicle scenarios. We have also implemented the machinery for checking for dt-adequacy (bad-adequacy and $\prec_1$-adequacy) and Timed Recoverability.

The symbolic soft-agents are executable specifications. In particular, the execution traces are enumerated by Maude [9] search. The constraints in the traces (non-linear arithmetic formulas) are solved by the SMT-solver (Z3 [11]). We implemented the connection between the symbolic soft-agents libraries and SMT solvers using the Python Binders described in [26], thus enabling easy extensions to additional solvers and other tools in the future.

The basic idea is to search for a counter-example to the property of interest. Because the symbolic search is complete, failure to find a counter-example means that the property holds for all instances of the LS under consideration.

As an example, to check bad-adequacy, the algorithm follows Proposition 2 by searching for a counter example, i.e. properties $\mathsf{SP}_0 \prec_1 \mathsf{SP}_1$ (not bad) and LS instances conf_0, conf_1 such that conf_0 satisfies SP_0, conf_1 satisfies SP_1, $\mathsf{conf}_0 \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$, and there is dt_0 with $0 < \mathsf{dt}_0 < \mathsf{dt}$, conf_2 such that $\mathsf{conf}_0 \rightarrow_{\mathsf{dt}_0} \mathsf{conf}_2 \rightarrow_{\mathsf{dt}-\mathsf{dt}_0} \mathsf{conf}_1$, where conf_2 satisfies bad. If no counterexample is found then bad-adequacy holds for the given LS, dt, and property specification.

Using symbolic rewriting, an arbitrary instance of LS is represented by a term, asys, consisting of a symbolic agent configuration and a symbolic environment. The environment contains knowledge of the physical state and the constraint on symbol values. The assertion that a property SP holds for a configuration is represented by the term $\mathsf{enforce}(\mathsf{asys}, \mathsf{SP})$ that conjoins the boolean term specifying SP in terms of the symbols of asys to the constraint in the environment. $\mathsf{cond}(\mathsf{asys})$ is the constraint in the environment part of asys.

The base case is adequacy for a pair of properties, SP_0, SP_1. The algorithm for this case does the following. First, use symbolic search from $\mathsf{asys}_0 = \mathsf{enforce}(\mathsf{asys}, \mathsf{SP}_0)$ for some asys_1 such that $\mathsf{asys}_0 \rightarrow_{\mathsf{dt}} \mathsf{asys}_1$ and $\mathsf{cond}(\mathsf{enforce}(\mathsf{asys}_1, \mathsf{SP}_1)))$ is satisfiable. If no such asys_1 is found, dt-adequacy holds for the given $\mathsf{SP}_0, \mathsf{SP}_1$. Otherwise, for some found asys_1 do a symbolic search from (a copy of) asys_0 for some $\mathsf{asys}_2, \mathsf{dt}_0$, where dt_0 is symbolic, such that $\mathsf{asys}_0 \rightarrow_{\mathsf{dt}_0} \mathsf{asys}_2$ and

$\mathsf{cond}(\mathsf{enforce}(\mathsf{asys}_1, \mathsf{SP}_1)) \wedge \mathsf{cond}(\mathsf{enforce}(\mathsf{asys}_2, \mathsf{bad})) \wedge 0 < \mathsf{dt}_0 < \mathsf{dt}$

is satisfiable. If such $\mathsf{asys}_1, \mathsf{asys}_2, \mathsf{dt}_0$ are found we have a counter-example, otherwise bad-adequacy holds for $\mathsf{SP}_0, \mathsf{SP}_1$.

The remaining algorithms for $\prec_1$-adequacy, noSkip property, and t-recoverability follow the same pattern as for bad-adequacy.

6.2 dt-adequacy Experiments

Table 1 presents our main experiments for proving dt-adequacy. Since for each scenario there are four levels of properties (bad, unsafe, safe, safer), there are ten cases to consider, e.g., the case from starting at a configuration satisfying safer and ending at another configuration satisfying safe and so on.

Pedestrian Crossing Scenarios: The soft-agents machinery is able to prove bad-adequacy in less than 3 minutes. However, for $\prec_1$-adequacy, the soft-agents machinery fails to

Pedestrian Crossing Scenarios		
Scenario	bad**-adequacy**	$\prec_1$**-adequacy**
pedCross$(3, 2, 1, 0)$	Yes (130s)	DNF
pedCrBnds$(3, 2, 1, 0)$	Yes (172s)	No (358s), failed case from safe to safer.
pedCross$(5, 2, 1, 0)$	Yes (89s)	Yes(149s)
pedCrBnds$(5, 2, 1, 0)$	Yes (78s)	No(172s), failed case from safe to safer.
folGap$(3, 2, 1)$	DNF	Yes (1413s)
folGap$(6, 4, 2)$	Yes (51s)	No (52s), failed case from safe to safe.
folGap$(7, 5, 1)$	Yes (55s)	Yes (83s)
folRSS(-8)	DNF	DNF
folRSS(-5)	DNF	DNF
folRSS(-2)	Yes (304s)	Yes (533s)

Table 1. Automated proofs for bad and $\prec_1$-adequacy for different scenarios. DNF denotes that the experiment was aborted after one hour.

return a result for the scenario pedCross$(3, 2, 1, 0)$ (without the explicit bounds). In particular, the SMT-solver cannot prove or find a counter-example within one hour. If we increase the values of $\mathsf{gap}_{\mathsf{safer}}$and $\mathsf{gap}_{\mathsf{safe}}$to 5 and 2, then the soft-agent machinery terminates positively. While it is hard to formally justify this as the SMT-solver applies several heuristics, this is, intuitively, expected as these new values result in more coarse safety properties.

Moreover, the $pedCrBnds$ scenarios do not satisfy the $\prec_1$-adequacy. In particular, it fails one case, namely, from safe to safer. This seems to suggest that one can merge safe and safer in the analysis of recoverability, as we are still able to detect transitions to the lower properties (unsafe and bad).

Vehicle Following Scenarios: Both sets of scenarios were challenging for the soft-agents machinery. Differently from the pedestrian crossing example, folGap was easier to prove $\prec_1$-adequacy and not terminating for bad-adequacy. Interestingly, when increasing the $\mathsf{gap}_{\mathsf{safer}}$, $\mathsf{gap}_{\mathsf{safe}}$, $\mathsf{gap}_{\mathsf{unsafe}}$bounds to 6,4, and 2, respectively, $\prec_1$-adequacy failed in the case from safe to safe, but increasing further the values to 7,5 and 1, the proof is established. This indicates that the value of 2 for $\mathsf{gap}_{\mathsf{unsafe}}$ is not adequate as the system is capable of traversing a configuration satisfying bad within a dt. For folRSS, the soft-agents machinery was only able to prove both adequacy properties when assuming a maximum deceleration for the leader vehicle of $-2m/s^2$.

In summary, all the scenarios, except folRSS(-5) and folRSS(-8), the soft-agents machinery is capable of demonstrating automatically bad and $\prec_1$ adequacy. The cases of folRSS(-5) and folRSS(-8) are more challenging and the investigation on how to improve the machinery or CPS modeling to handle them is left to future work.

6.3 Time-bounded Recoverability Experiments

Table 2 summarizes our main experiments for recoverability involving the pedestrian crossing and vehicle following scenarios. Recall that the objective of $\langle \mathsf{safe}, \mathsf{safer}, t\rangle$-Recoverability is to prove that the safety controller is capable of reducing vehicle risk to safer. For the experiments we generally used simple, rather cautious controllers.

$\langle \mathsf{safe}, \mathsf{safer}, t \rangle$-**One-Recovery-Period**		
Pedestrian Crossing Scenarios		
Scenario	$t = 4$	$t = 5$
$\mathsf{pedCross}(3, 2, 1, 0)$	No (34s)	No (115s)
$\mathsf{pedCrBnds}(3, 2, 1, 0)$	No (27s)	Yes (621s)
$\mathsf{pedCross}(5, 2, 1, 0)$	No (27s)	No (93s)
$\mathsf{pedCrBnds}(3, 2, 1, 0.50)$	–	No (103s)
$\mathsf{pedCrBnds}(3, 2, 1, 0.33)$	–	No (104s)
$\mathsf{pedCrBnds}(3, 2, 1, 0.125)$	–	Yes (637s)
$\mathsf{pedCrBnds}(3, 2, 1, 0.1)$	–	Yes (734s)
Vehicle Following Scenarios		
Scenario	**Recoverability**	
$\mathsf{folGap}(3, 2, 1)$	$t = 5$	No (12s)
$\mathsf{folGap}(6, 4, 2)$	$t = 5$	No (11s)
$\mathsf{folGap}(7, 5, 1)$	$t = 5$	No (12s)
$\mathsf{folRSS}(-5)$	$t = 2$	No (5s)
$\mathsf{folRSS}(-5)$	$t = 3$	No (81s)
$\mathsf{folRSS}(-5)$	$t = 4$	No (1126s)
$\mathsf{folRSS}(-5)$	$t = 2$	Yes (38s) $\star$
$\mathsf{folRSS}(-2)$	$t = 2$	Yes (43s)

Table 2. Automated proofs for Timed-Recoverability. The symbol $\star$ denotes that the experiment used a very aggressive controller. As the scenario $\mathsf{pedCrBnds}(3, 2, 1, 0)$ is not recoverable for $t = 4$, it is not necessary to carry out experiments for the scenarios marked with –.

For the purpose of illustration, we specified two controllers for the vehicle follower scenarios: a non-aggressive safety controller and an aggressive controller. The latter always activates the emergency brake, i.e., maximum deceleration. Finally, for each scenario, our machinery showed that dt does not skip safe (see Definition 7) in around one second.

Pedestrian Crossing Scenarios: The first observation is that one is not able to establish recoverability with the safety properties used for pedCross. Our machinery returns a counter-example where the vehicle has very low speeds and is very close to the pedestrian crossing with distance around $0.5m$. This illustrates the importance of including the bounds to safety properties as done in pedCrBndsas in Equation 2.

For the scenario $\mathsf{pedCrBnds}(3, 2, 1, 0)$, the safety controller always returns to a safer risk situation after 5 ticks, but not 4 ticks. Notice that for $\mathsf{pedCrBnds}(5, 2, 1, 0)$ this is no longer the case as it fails also after 5 ticks. This is expected as the "distance" between the properties safe and safer has increased.

Finally, the experiments for $\mathsf{pedCrBnds}(\mathsf{gap}_{\mathsf{safer}}, \mathsf{gap}_{\mathsf{safe}}, \mathsf{gap}_{\mathsf{unsafe}}, \mathsf{senerr})$ illustrate how to check the recoverability of safety controllers in the presence of faulty sensors. If we assume faults of 50% or 33% on the pedestrian sensor, the safety controller cannot

guarantee that it will always return to a safer risk condition. However, it is able to do so for errors of 12.5% or 10%.

Vehicle Following Scenarios: Our experiments demonstrate that it seems harder to establish recoverability when using time gaps to establish levels of risk. It probably requires a more sophisticated safety controller. On the other hand, when using RSS-based properties, it is possible to establish recoverability, even with small time frames, albeit when assuming normal decelerations of the leader vehicle. It is possible to establish recoverability for scenarios assuming higher values for deceleration, but then a more aggressive controller is required.

7 Related Work

RTAs. Since the first proposal of RTAs, called Simplex Architecture [27], there has been several recent proposals of RTA variants [22,18,10] (to name a few). While there are some differences on their architectures and functions, they all contain a decision module that evaluates the system risk level to decide which controller to use (the safe or the advanced controller). Therefore, all the requirements formalized in this paper, namely, the time sampling adequacy and recoverability are still relevant and applicable. Indeed, we advance the state of the art by providing suitable definitions that are amenable to automated verification.

We have been inspired by [12] that proposes high-level requirements on the recoverability of RTAs based on the level of risk of the system. In particular, the methods for checking adequacy of the sampling and for checking t-recoverability correspond to the safety and liveness requirements of RTA wellformedness. The third condition concerns the minimum time to become unsafe (non-safe) with any controller in charge, needed to ensure that the monitor can switch controllers and the safe controller can react before reaching an unsafe condition. This can be shown using $<_1$-adequacy and continuity of properties in a $<_1$-chain. Summarizing, symbolic rewriting combined with SMT solving provides automated methods to verify correctness of time sampling mechanisms and safety requirements such as those of the RTA framework of [12].

In a similar direction, [18] proposes high-level requirements for the correctness of the decision module based on the definition of what is safe and existence of "permanently safe command sequences", which seems related to our time recoverability property. They do not investigate, however, the effect of the time sampling and the correctness of the decision module.

CPS Verification and Validation Much of the literature in CPS verification, e.g. [14] to name one, including some of the previous work on RTA [22,10,18], rely on simulation-based methods. These approaches are complementary to the one introduced in this paper. While this paper's approach targets more early phase development by providing proofs that RTA specifications are suitable for all instances of a logical scenario, simulation-based approaches focus on later approaches for validating and testing implementations of RTA systems on particular instances of logical scenarios.

dL, KeYmaera X, and VeriPhy. The KeYmaera X prover [31,21] uses differential dynamic logic (dL) to specify and verify CPS controller designs. It is the starting point

of the VeriPhy pipeline [6,7] for producing code from logical specifications. dL specifications and logical scenarios have in common that they are given by terms with constrained variables representing all instances where values of variables satisfy the given constraints. Our methods differ in that dL specifications are not directly executable and therefore, one uses interactive theorem proving methods to verify dl specifications, whereas logical scenarios are executable thus enabling further automation of verification proofs using rewriting modulo SMT.

Formal Definitions of Resilience: Alturki *et al.* [3] propose formal definitions for resilience and shows them to be undecidable in general and PSPACE-complete for some cases. While formal connections are left to future work, our definition of timed recoverability seem to specialize their definition so to be applicable for RTA architectures, e.g., considering dt-adequacy.

8 Conclusions

In this paper we present a formal foundation for logical scenarios (LS) and methods to automate proving safety properties. An LS consists of instances of a pattern satisfying given ODD constraints, together with a two-step transition relation giving the semantics. The first step corresponds to reading sensors, analyzing and deciding on actions (setting control parameters). The second step evolves the system for the sampling time between observations. Towards a formal foundation we introduce a notion of Safety Property Specification for an LS as a set of property (names) with a risk level ordering relation, a unique least (most risky) element, bad, and a satisfaction relation. An adequate sampling time should ensure that nothing important is missed. We define three notions of dt adequacy and show that they are distinct and totally ordered. A system may be allowed to enter a situation that is safe but risky, but a resilient system will recover to an acceptably safe situation. This is formalized in a definition of t-recoverability. A notion of one-period-recovery t-recoverability is defined that is amenable to verification, and shown to be equivalent to t-recoverability for adequate dt using an inductive argument.

Towards automation of proofs, we use symbolic rewriting modulo SMT as the execution and search engine [20]. Algorithms were developed to prove all (infinitely many) instances of an LS satisfy different notions of dt adequacy or t-recoverability (or to provide counter example instances). We report a set of experiments checking dt adequacy and t-recoverability properties for LSs and safety property specifications related to vehicle automation: vehicle following and pedestrian crossing. The experiments show that it possible to find values of dt and safety parameters where adequacy holds and very simple controllers satisfy t-recoverability. They also highlight corner cases where things go awry.

One direction of future work is to investigate a wider range of case studies to better understand how the different design parameters interact. Another important direction is to develop methods to compose Logical Scenarios and proofs, thus scaling analysis of complex systems.

Acknowledgments. Talcott was partially supported by the U. S. Office of Naval Research under award numbers N00014-15-1-2202 and N00014-20-1-2644, and NRL grant N0017317-1-G002.

References

1. Apollo. An Open Autonomous Driving Platform. https://github.com/ApolloAuto/apollo.
2. B. Allenby and J. Fink. Toward inherently secure and resilient societies. *Science*, 309(5737):1034–1036, 2005.
3. M. A. Alturki, T. B. Kirigin, M. I. Kanovich, V. Nigam, A. Scedrov, and C. L. Talcott. On the formalization and computational complexity of resilience problems for cyber-physical systems. In H. Seidl, Z. Liu, and C. S. Pasareanu, editors, *Theoretical Aspects of Computing - ICTAC 2022 - 19th International Colloquium, Tbilisi, Georgia, September 27-29, 2022, Proceedings*, volume 13572 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2022.
4. K. Barker, J. E. Ramirez-Marquez, and C. M. Rocco. Resilience-based network component importance measures. *Reliability Engineering & System Safety*, 117:89–97, 2013.
5. R. Bloomfield, G. Fletcher, H. Khlaaf, P. Ryan, S. Kinoshita, Y. Kinoshit, M. Takeyama, Y. Matsubara, P. Popov, K. Imai, et al. Towards identifying and closing gaps in assurance of autonomous road vehicles–a collection of technical notes part 1. *arXiv preprint arXiv:2003.00789*, 2020.
6. B. Bohrer, Y. K. Tan, S. Mitsch, M. O. Myreen, and A. Platzer. VeriPhy: Verified controller executables from verified cyber-physical system models. In *Proceedings of 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM New York, 2018.
7. B. Bohrer, Y. K. Tan, S. Mitsch, A. Sogokon, and A. Platzer. A formal safety net for waypoint following in ground robots. *IEEE RObotics and Automation Letters*, 2019. arxiv:1903.15073.
8. M. Bozga and J. Sifakis. Specification and validation of autonomous driving systems: A multilevel semantic framework. *CoRR*, abs/12109.06478, 2021.
9. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.
10. A. Damare, S. Roy, S. A. Smolka, and S. D. Stoller. A barrier certificate-based simplex architecture with application to microgrids. In T. Dang and V. Stolz, editors, *Runtime Verification - 22nd International Conference, RV 2022, Tbilisi, Georgia, September 28-30, 2022, Proceedings*, volume 13498 of *Lecture Notes in Computer Science*, pages 105–123. Springer, 2022.
11. L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
12. A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari. SOTER: A runtime assurance framework for programming safe robotics systems. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 138–150. IEEE, 2019.
13. D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. Scenic: a language for scenario specification and scene generation. In K. S. McKinley and K. Fisher, editors, *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pages 63–78. ACM, 2019.

14. D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *23rd IEEE International Conference on Intelligent Transportation Systems, ITSC 2020, Rhodes, Greece, September 20-23, 2020*, pages 1–8. IEEE, 2020.
15. S. Jha, J. Rushby, and N. Shankar. Model-centered assurance for autonomous systems. In A. Casimiro, F. Ortmeier, F. Bitsch, and P. Ferreira, editors, *Computer Safety, Reliability, and Security - 39th International Conference, SAFECOMP 2020, Lisbon, Portugal, September 16-18, 2020, Proceedings*, volume 12234 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2020.
16. J.-C. Laprie. From dependability to resilience. In *38th IEEE/IFIP Int. Conf. On dependable systems and networks*, pages G8–G9. Citeseer, 2008.
17. P. D. Luca Mengani. Hazard analysis and risk assessment and functional safety concept. Technical report, 2019. D2.11 of H2020 project ENSEMBLE, www.platooningensemble.eu.
18. U. Mehmood, S. Sheikhi, S. Bak, S. A. Smolka, and S. D. Stoller. The black-box simplex architecture for runtime assurance of autonomous CPS. In J. V. Deshmukh, K. Havelund, and I. Perez, editors, *NASA Formal Methods - 14th International Symposium, NFM 2022, Pasadena, CA, USA, May 24-27, 2022, Proceedings*, volume 13260 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2022.
19. T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, pages 1821–1827. IEEE, 2018.
20. V. Nigam and C. Talcott. Automating safety proofs about cyber-physical systems using rewriting modulo smt. In K. Bae, editor, *14th International Workshop on Rewriting Logic and its Applications*, volume 13252 of *LNCS*, pages 212–229. Springer, 2022.
21. J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer. How to model and prove hybrid systems with KeYmaera: a tutorial on safety. *Int J Software Tools Technology Transfer*, 18:67–91, 2016.
22. S. Ramakrishna, C. Hartsell, M. P. Burruss, G. Karsai, and A. Dubey. Dynamic-weighted simplex strategy for learning enabled cyber physical systems. *J. Syst. Archit.*, 111:101760, 2020.
23. S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer. Survey on scenario-based safety assessment of automated vehicles. *IEEE Access*, 8:87456–87477, 2020.
24. C. Rocha, J. Meseguer, and C. Muñoz. Rewriting modulo SMT and open system analysis. *Journal of Logical and Algebraic Methods in Programming*, pages 269–297, 2017.
25. R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid. Developing cyber resilient systems: a systems security engineering approach. Technical report, National Institute of Standards and Technology, 2019.
26. R. Rubio. Maude as a library: An efficient all-purpose programming interface. In K. Bae, editor, *Rewriting Logic and Its Applications - 14th International Workshop*, volume 13252 of *Lecture Notes in Computer Science*, pages 274–294. Springer, 2022.
27. D. Seto, B. Krogh, L. Sha, and A. Chutinan. The simplex architecture for safe online control system upgrades. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, volume 6, pages 3504–3508 vol.6, 1998.
28. L. Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, 2001.
29. S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017.
30. C. Talcott, V. Nigam, F. Arbab, and T. Kappé. Formal specification and analysis of robust adaptive distributed cyber-physical systems. In M. Bernardo, R. D. Nicola, and J. Hillston, editors, *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems*,

LNCS. Springer, 2016. 16th edition in the series of Schools on Formal Methods (SFM), Bertinoro (Italy), 20-24 June 2016.
31. T. K. X. team. KeYmaera X: An aXiomatic tactical theorem prover for hybrid systems, 2022. Last accessed Sept 22, 2022.
32. L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Kramer, C. Gutenkunst, and E. Böde. Criticality metrics for automated driving: A review and suitability analysis of the state of the art. *Archives of Computational Methods in Engineering*, abs/2108.02403, 2022.