arXiv:2304.11564v1 [cs.LO] 23 Apr 2023

技术报告：利用运行时保障架构自动化信息物理系统的可恢复性证明

Vivek Nigam[2,3] 和 Carolyn Talcott[1]

[1] SRI International，门洛帕克，美国，carolyn.talcott@gmail.com[2] 帕拉伊巴联邦大学，若昂佩索阿，巴西，vivek.nigam@gmail.com[3] 华为慕尼黑研究中心，德国

摘要。 信息物理系统（CPSes），例如自动驾驶车辆，使用基于机器学习的控制器等复杂组件。要为此类组件的安全运行提供证据并非易事。为解决这一问题，人们提出了运行时保障架构（RTA）方案。RTA 的决策组件会评估系统的安全风险；每当风险高于可接受水平时，RTA 就会切换到安全模式，例如激活一个有充分证据证明其能够安全运行的控制器。通过这种方式，实时自主系统通过将系统从较高风险等级恢复到较低风险等级，提升信息物理系统的运行时安全性和韧性。本文旨在利用实时自主系统自动生成信息物理系统的恢复性证明。我们首先形式化关键验证问题，即决策采样时间充分性问题和有时限的可恢复性问题。随后，我们展示如何使用模 SMT 符号重写，自动为所提出的验证问题生成证明。通过集成重写逻辑工具 Maude（用于生成非线性约束集合）与 SMT 求解器 Z3，实现了自动化。

1 引言

信息物理系统（CPSes）正日益以自主方式执行复杂的安全关键任务，自动驾驶车辆（AV）便是当前的典型例子。鉴于此类信息物理系统运行环境的复杂性，它们通常依赖高度复杂的基于机器学习（ML）的控制器 [1]，这是因为机器学习能够学习有关车辆运行条件的隐含要求。然而，由于此类基于机器学习的组件存在功能不足 [2]，仅使用这些组件来提出安全性论证一直非常困难。尽管人们投入了大量精力来构建用于验证基于机器学习的组件的系统的方法，但这些组件仍然存在多于可接受数量的故障 [17]。

基于广为人知的 Simplex 架构，运行时保障架构（RTAs） [31,30], 已被提出 [14,25,20] 作为应对这一挑战的方法。RTA 包含一个决策模块，用于评估系统的安全风险，该风险通过一组安全属性形式化定义。只要安全风险高于可接受水平，RTA 就会将系统转移到安全状态。如图 1所示，RTA 会根据感知到的

系统风险等级动态调整网络物理系统的行为，使网络物理系统从较高风险状态中恢复。我们用符号 dt 表示决策模块评估系统风险等级的采样间隔。这些风险等级通过性质形式化定义，并根据系统的运行领域量身定制 [23]。例如，高速公路上的车辆与城市场景中的车辆，其风险等级的形式化定义不同；在城市场景中，行人可能会穿越道路。图 1 的示意图中有四个逐渐升高的风险等级（更安全、安全、不安全、不安全状态），例如表示发生事故的风险：从 更安全（表示最低且理想的风险等级）到 不安全状态（表示必须不惜一切代价避免的最高风险等级），以避免可能发生的事故。

如果风险较低，决策模块就会使用主控制器（未经验证）的输出。然而，如果检测到风险较高但仍安全的状态，决策模块就会使用安全控制器的输出。预期安全控制器最终能够从高风险状态中恢复，使系统回到风险较低的状态。在此过程中，网络物理系统可能会经过不安全状态，但绝不能经过不安全状态，例如会触发其他安全机制（如紧急制动器）的即将发生碰撞状态。

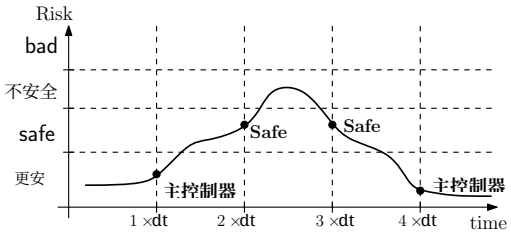

图 1. 示意图说明 RTA 预期如何在运行时维持安全性。dt 是决策模块的采样时间。**主控制器**（相应地，**安全**）表示决策模块切换到主控制器（相应地，安全控制器）。

工程师必须通过提供充分证据来证明实时自主系统具备以下两项关键性质：

– dt **充分性：** 采样时间间隔足够小，因此 RTA 不会遗漏不安全状态；– **有界时间可恢复性：** 如果系统风险变得高于可接受水平（更安全），安全控制器可以在指定的时间界限内将系统恢复到更安全的状态，而不会进入不安全状态。

本文的主要目标是开发方法，以自动化方式为 RTA 实例生成这些性质的形式化证明。为此，本文使用符号软智能体框架 [23]，该框架能够利用模 SMT 符号重写，为网络物理系统自动生成安全性证明 [27]。本文的贡献主要集中在两个方面：

– **形式化基础。** 我们为 dt 充分性的三种变体提供形式化定义，并证明它们之间的关系。我们还提供有界时间可恢复性的形式化定义。我们定义了单周期可恢复性的概念，并证明单周期可恢复性与任一种 dt 充分性性质相结合即可推出有界时间可恢复性。这些形式化定义经过专门设计，适用于自动化验证。

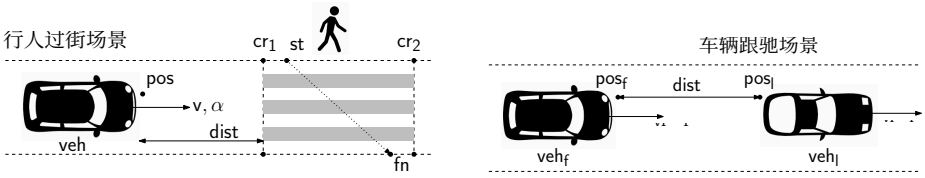

图 2. 行人过街和车辆跟驰逻辑场景示意图。道路位于 Y 轴上，因此可以想象这些示意图逆时针旋转了 90 度。

– RTA 属性的自动化检查： 基于对 RTA 属性的规范以及对网络物理系统运行情况的抽象描述（称为逻辑场景）的规范 [26,21], 我们提出了用于验证两种 dt 充分性和单周期可恢复性的算法，并报告了针对两个逻辑场景的实验结果。实验表明，自动化证明是可行的，同时也展示了其中的一些挑战。

第2 节介绍了我们的运行示例中的逻辑场景。第 3节使用安全属性形式化了风险等级的概念。随后，在第 4 节中，利用这些概念定义了若干采样时间充分性的概念，并在第5节中定义了可恢复性属性。第6 节介绍了基于第2节逻辑场景的实验。最后，我们在第7 节和第 8节中总结相关工作并展望未来工作。

2 逻辑场景与启发性示例

自主信息物理系统开发中的一个关键步骤，是定义这些系统将运行于其中的情境 [26,21,35]。这些情境以抽象场景的形式加以描述，称为逻辑场景 [26,21],，例如变道、车辆跟驰或行人穿行，在这些场景中，自动驾驶车辆必须避免造成伤害。这些逻辑场景包含车辆能够安全运行所需情境的详细信息，例如参与体的类型和数量（如车辆、行人）、运行假设（如速度范围）以及道路拓扑（如车道数）。此外，这些逻辑场景还关联着安全性度量，用于形式化车辆需要满足的性质。有关逻辑场景及其相关性质的完整列表，请参阅 [35] 及其中的参考文献。场景描述和生成形式化方法的示例见 [15,9]。由于一个逻辑场景可能包含无限多个具体实例，因此要证明车辆在所有实例中都能满足此类安全属性颇具挑战。

我们使用两个贯穿全文的示例，如图 2所示：一个行人穿越场景和一个车辆跟驰场景。

人行横道 在此场景中，自车[4]（vh）位于位置 pos，正以速度 v 和加速度 α驶近，而一条人行横道位于位置 cr_1 和 cr_2之间。此外，一名行人正试图通过人行横道过马路。只要行人没有移出人行

[4]自车是指在测试、跟驰或运行场景中重点关注的车辆。

横道之外，人行横道的具体形状并不重要vh 只要有行人打算过人行横道，就必须始终在人行横道前停车。为简化问题，假设行人以恒定速度v_p沿直线穿过街道，如图2 中从st 到fn所示的虚线。

此类逻辑场景的运行设计域（ODD）由其参数（pos、v、 α、cr_1, cr_2, v_p）的约束规定。通常，需要规定速度和加速度的范围。例如：

$$0m/s \leq v \leq 10m/s \quad -8m/s^2 \leq \alpha \leq 2m/s^2 \quad 1m/s \leq v_p \leq 4m/s$$

此外，$pos.y < cr_1.y$，即车辆正在接近人行横道，并且 $cr_1.y \leq st.y$，$fn.y \leq cr_2.y$，即 st，fn 位于人行横道区域内；对于任意位置 $l = (px, py)$，$l.x$ 和 $l.y$ 分别表示 px 和 py。

车辆跟驰 我们的第二个运行示例是车辆跟驰场景，如图 2所示。该示例在相关文献中十分常见，因此我们不再进行同等详细的描述，而仅引用 [23]。简而言之，该场景包含两辆车辆：跟车（veh_f）和前车（veh_l）。通常，这些车辆行驶在具有多条车道的高速公路上，速度相当高，例如速度介于 $60km/h$和 $140km/h$ 之间，并且其加速度约束与人行横道场景中的车辆相同。此外，场景中只有车辆，即没有行人、自行车等。跟车应避免过于危险地接近前车，同时保持合理的速度。

我们假设，对于逻辑场景（LS）的一个实例 conf，可以计算函数 $conf \longrightarrow_{\Delta} conf_1$，，其中 $conf_1$ 是一个 LS 实例，用于指定智能体的物理属性，例如速度、方向和加速度；这些属性是根据 conf 中智能体的速度、方向和加速度，在经过 $\Delta > 0$ 个时间单位后获得的。此外，我们假设，如果 $conf \longrightarrow_{\Delta_1+\Delta_2} conf_1$，，那么存在 conf′，使得 $conf \longrightarrow_{\Delta_1} conf' \longrightarrow_{\Delta_2} conf_1$。[5] 例如，考虑行人过街场景的一个实例，其中车辆的速度为 $10m/s$，加速度为 $2m/s^2$，位置为 $pos.x = 0m$。经过 $\Delta = 0.1s$ 后，车辆的速度将为 $10.2m/s$，新位置为 $1.1m$。车辆沿道路保持恒定方向行驶，此处将其省略。

3 安全属性与风险等级

RTA机制的一个关键方面是能够检查系统的风险等级，例如判断系统是否安全。我们将风险等级的概念形式化为安全属性上的偏序，如下所示：

定义 1. 对于逻辑规约 LS，*RTA* 安全属性规范是一个元组 $\langle \mathcal{S}, \prec_1, \text{不安全状态} \models \rangle$，其中– $\mathcal{S} = \{SP_1, \ldots, SP_n\}$ 是有限个安全属性组成的集合；– $\prec_1$: $\mathcal{S} \times \mathcal{S}$ 是 $\mathcal{S}$ 上的非对称二元关系，其中 $SP_1 \prec_1 SP_2$ 表示安全属性 SP_2 所规定的条件比安全属性 SP_1所规定的条件风险更低。令 $\prec$ 为对 $\prec_1$ 应用传递性后得到的序。我们假设 $\prec$ 是严格预序（不存在环）。

[5] 由于 $\longrightarrow_{\Delta}$ 是一个函数，因此 conf′ 是唯一的。

– 安全属性 不安全状态 $\in \mathcal{S}$ 是 $\prec$ 的最小元素，规定了应当避免的条件，即风险最高– $\models$ 规定逻辑场景 LS 的实例 conf 何时满足性质 SP $\in \mathcal{S}$，记作 conf $\models$SP。此外，我们假设，如果 conf $\models \mathsf{SP}_1$，且 $\mathsf{SP}_1 \prec \mathsf{SP}_2$ 或 $\mathsf{SP}_2 \prec \mathsf{SP}_1$,，那么conf $\not\models \mathsf{SP}_2$。也就是说，逻辑场景的任一实例只能满足一个风险等级。我们还假设，逻辑场景的任一实例都处于某个风险等级，即对于LS的所有实例 conf，至少存在一个SP，使得 conf $\models$ SP。

下面两个示例展示了第 2 节中所述的人行横道和车辆跟驰示例所对应的不同安全属性选项。

示例*1.* 考虑图 2 所示的人行横道。我们定义以下 RTA 安全属性规范 $\langle${不安全状态，不安全，安全，更安全}，$\prec_1$，不安全状态 $\models\rangle$，其中不安全状态 $\prec_1$ 不安全 $\prec_1$ 安全 $\prec_1$ 更安全 是基于到斑马线的时间 度量 [35] [6]

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

其中 $\mathrm{dist} = \mathrm{cr}_1.y - \mathrm{pos}.y$ 是自车与人行横道之间的距离，$\mathrm{dStop} = -(\mathrm{v} * \mathrm{v})/(2 * \mathrm{maxDec})$ 是通过施加最大减速度 maxDec 停下自车所需的距离，例如发出紧急制动指令时所需的距离，而 $\mathrm{gap}_{\mathsf{safer}} > \mathrm{gap}_{\mathsf{safe}} > \mathrm{gap}_{\mathsf{unsafe}} > 0$ 与 v 配合使用，用于在安全属性中指定安全裕度距离。$\mathrm{gap}_{\mathsf{safer}}$，$\mathrm{gap}_{\mathsf{safe}}$，$\mathrm{gap}_{\mathsf{unsafe}}$ 的取值应根据自车的能力（例如采样时间 dt）以及运行设计域规范（例如加速度和速度的上下限）来确定。随后，只需直接检查一个行人逻辑场景实例是否满足上述任一性质($\models$)。

虽然这似乎是适用于人行横道的一种不错的安全属性规范，但事实证明，要证明车辆的可恢复性并不容易，正如我们在第 6 节所示。问题在于，当车辆速度（v）趋近于零时，这三个性质往往会变得完全相同；当 dist 过大时，情况也类似。因此，我们将更安全 另行定义如下：

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

其中，lowSpd 和 farAway 是常量，分别表示车辆非常安全时的最高速度和距离。例如，lowSpd 表示低于行人速度的速度，而 farAway 表示距人行横道足够远的距离。

示例 *2*。 车辆跟车场景中用于车辆安全性保障的一个知名示例，是责任敏感安全（RSS） [32,35] 安全距离指标。RSS 安全距离 drss react 定义如下：

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

[6] 斑马线是行人过街区域。

其中，react 是车辆作出反应所需时间的参数；v 和 v_l 分别表示跟车与前车的速度；maxacc$_f$ 表示跟车的最大加速度；maxdec$_f$ 和 maxdec$_l$ 分别表示跟车与前车的最大减速度。基于 drss(react)，定义了两个性质：当 dis $<$ drss 时为不安全状态，否则为更安全。

由于责任敏感安全只有两种性质，使用 RTA 定义可恢复性意味着系统必须始终满足“更安全”性质；否则就必须处于不安全状态。这意味着主控制器必须可信，因此从一开始就不需要 RTA（而且主控制器被假定未经验证，所以可能也不希望采用 RTA）。可以根据反应时间，在“更安全”和不安全状态之间增加额外级别，以此调整责任敏感安全的定义：

更安全：$=$ dis $\geq$ drss($3\times$ dt) 安全：$=$ drss($2\times$ dt) $\leq$ dis $<$ drss($3\times$ dt) 不安全：$=$ drss(dt) $\leq$ dis $<$ drss($2\times$ dt) 不安全状态：$=$ dis $<$ drss(dt)

直观地说，当车辆处于满足“更安全”性质的配置时，即使错误地评估安全风险（例如由于距离传感器误差），也可以持续两个周期，之后 RSS 属性才会失效。同样，处于安全状态时可以错误评估一个周期；而处于不安全状态时，则必须始终正确评估风险。

4 采样时间（dt）充分性

RTA 监视器必须检测系统风险何时发生变化，尤其是风险何时增加；也就是说，系统满足的性质 SP 变得更接近 不安全状态，即在顺序 $\prec$中向下移动。这意味着采样时间 dt 对 RTA 系统的正确性至关重要。例如，如果图 $4\times$ dt 1中的采样时间为 ，RTA 监视器可能无法检测到风险从 更安全 上升到 安全，因而无法及时激活可信控制器以避免风险进一步升级。

系统检测风险变化并据此快速响应的能力，与监控系统确定风险时的性能要求（即 dt 时间）之间存在权衡。dt 越小，系统检测变化的能力越强，同时对监控组件的性能要求也越高。

此外，一项关键挑战是，dt 必须适用于检测运行设计域的所有实例中的风险变化，即所有可能的速度和加速度实例。我们的方法是借鉴文献 [23]中的思路，使用 SMT 求解器自动生成 dt 充分性证明。根据 dt 充分性的定义，问题的复杂度可能会大幅增加，从而使自动化变得困难，甚至无法实现。

我们提出了三种对dt的替代性要求定义，如下所述。这些定义体现了系统检测风险变化能力与开发和验证工作量之间的权衡。图3 展示了这些定义。第一种定义称为单次转换充分性，见图3最左侧的示意图。直观地说，该定义要求 dt 足够小，以便在场景配置从满足某个性质SP_1,转变为满足另一个性质SP_2时，能够检测到这一变化。例如，系统经过$conf'_d$ 的虚线演化过程，在一个dt时间段内包含多次性质变化。

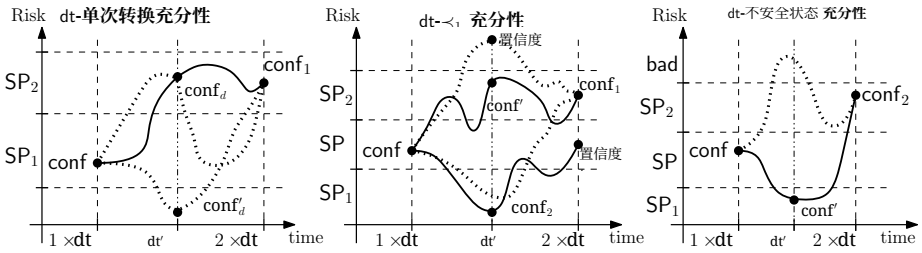

图 3。 dt 充分性性质示意图。完整线条表示允许的系统演化，虚线表示不允许的系统演化。

定义 2。 令 $Spec = \langle \mathcal{S}, \prec_1$,不安全状态 $\vDash \rangle$ 为逻辑场景LS的 *RTA* 安全属性规范；令dt 为采样时间。对于 $Spec$ 和LS ， 若对LS 的所有实例conf，conf_1 满足LS 且 $\mathsf{conf} \rightarrow_{\mathsf{dt}} \mathsf{conf}_1$，则称dt 是单步转移充分的；此时有：

– 如果 $\mathsf{conf} \vDash \mathsf{SP}_1$ 和 $\mathsf{conf}_1 \vDash \mathsf{SP}_2$, ， 那么对于某个 $0 \leq \mathsf{dt}' < \mathsf{dt}$，存在一个分解$\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}_d \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ ， 使得：

- 对于 $\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}_d$ 的所有分解，即 $\mathsf{conf} \rightarrow_{\mathsf{dt}_2} \mathsf{conf}_2 \rightarrow_{\mathsf{dt}'-\mathsf{dt}_2} \mathsf{conf}_d$，其中 $0 < \mathsf{dt}_2 < \mathsf{dt}'$，我们有$\mathsf{conf}_2 \vDash \mathsf{SP}_1$；
- 对于 $\mathsf{conf}_d \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ 的所有分解，即 $\mathsf{conf}_d \rightarrow_{\mathsf{dt}_3} \mathsf{conf}_3 \rightarrow_{\mathsf{dt}-\mathsf{dt}'-\mathsf{dt}_3} \mathsf{conf}_1$ ， 其中 $0 \leq \mathsf{dt}_3 < \mathsf{dt} - \mathsf{dt}'$，我们有 $\mathsf{conf}_3 \vDash \mathsf{SP}_2$。

以下命题可由定义 2 直接推出。该命题指出，若dt 是单次转换充分的，那么要检查满足 不安全状态 的配置是否不可达，只需检查采样时刻的配置是否均非不安全状态，而无需检查所有分解。

命题 1。令 $Spec = \langle \mathcal{S}, \prec_1$,不安全状态 $\vDash \rangle$ 为逻辑场景逻辑场景的 *RTA* 安全属性规范dt 。设其相对于 $Spec$ 是单次转换充分的。对于$\mathsf{conf} \rightarrow_{\mathsf{dt}'} \mathsf{conf}' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} \mathsf{conf}_1$ 的所有分解，若逻辑场景的 $\mathsf{conf}' \vDash$ 不安全状态 ， 当且仅当$\mathsf{conf} \vDash$ 不安全状态 或$\mathsf{conf}_1 \vDash$不安全状态。

定义 2 相当复杂，涉及许多量词交替，因此很难生成证明。实际上，由于计算时间的限制，并不总能保证dt 满足单次转换充分性。因此，我们提出了两种较弱性质的替代定义，如图 3 中的中间图和最右侧图所示。正如我们将在第 6 节中详细说明的那样，这些性质适用于自动生成证明。

第一种替代定义是 $\prec_1$ 充分性。 $\prec_1$ 充分性不要求 dt 足够精细，能够检测系统何时满足不同性质，而是允许系统演化在安全属性的 $\prec_1$ 范围内多次迁移，如系统演化经过 conf' 所示。系统应能够检测到系统风险至少上升两个等级的每一次情况。

定义 3。 令 $Spec = \langle \mathcal{S}, \prec_1$,不安全状态 $\vDash \rangle$ 为逻辑场景 逻辑场景的 *RTA* 安全属性规范；令 dt 为采样时间。相对于采样时间 ， dt 是 $\prec_1$ 足够的

$Spec$ 和逻辑场景（LS）满足以下条件：对于逻辑场景（LS）的所有实例配置，以及所有关系配置 $\rightarrow_{\mathsf{dt}}$ 配置$_1$ 对于所有$0 < dt' \leq dt$和分解配置 $\rightarrow_{\mathsf{dt}'}$ 配置$'$ $\rightarrow_{\mathsf{dt}-\mathsf{dt}'}$ 配置$_1$，均有：– 如果配置 $\vDash SP_1$ 且配置$_1 \vDash SP_2$ 对于 $SP_1 \neq SP_2$, 成立，那么配置$' \vDash SP_1$ 或配置$' \vDash SP_2$成立。– 如果配置 $\vDash SP$ 且配置$_1 \vDash SP$，那么配置$' \vDash SP$ 或配置$' \vDash SP'$ 成立，其中 $SP' \prec_1 SP$ 或 $SP \prec SP'$。

可以推广 $\prec_1$ 的定义，使其允许在更大范围的安全属性上演化，例如，$\prec_n$ 充分性允许 $n \geq 1$ 在 n 安全风险等级内演化。

$\prec_1$-充分性的以下性质为基于 $\prec_1$-充分性 dt 定义可恢复性提供了基础。只需检查不存在满足不安全状态或紧邻不安全状态的性质的配置可达即可。

命题*2*。设 $Spec = \langle \mathcal{S}, \prec_1$,不安全状态 $\vDash \rangle$ 是逻辑场景 LS 的 RTA 安全属性规范；dt 是 $\prec_1$ 充分性的采样时间。若 $conf \rightarrow_{\mathsf{dt}} conf_1$ 满足 $conf \vDash SP_1$ 且 $conf_1 \vDash SP_2$，其中 $SP_1 \neq$ 不安全状态，且 $SP_2 \neq$ 不安全状态，并且不安全状态 $\not\prec_1 SP_1$ 或不安全状态 $\not\prec_1 SP_2$,，则对于所有 $0 < dt' \leq dt$ 以及分解 $conf \rightarrow_{\mathsf{dt}'} conf' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} conf_1$，均有 $conf' \nvDash$ 不安全状态。

例如，考虑示例1中的安全属性规范，并假设dt 是 $\prec_1$充分的。根据命题2，如果不存在转换conf $\rightarrow_{\mathsf{dt}}$ conf$_1$，其中conf $\vDash$ 不安全且conf$_1 \vDash$ 不安全，那么我们可以保证系统不会经过配置conf$'$，其中conf$' \vDash$ 处于不安全状态，包括车辆系统未采样的中间配置。

下一个充分性条件仅要求dt 足够小，以便检测系统演化何时满足不安全状态性质。如图3最右侧示意图所示，虚线演化在dt 时间内满足dt 不安全状态会使dt 充分性失效。

定义*4*。令 $Spec = \langle \mathcal{S}, \prec_1, bad \vDash \rangle$ 为逻辑场景 LS 的 RTA 安全属性规范；令 dt 为采样时间。若对于 LS 的所有实例 $conf$ 和 $conf \rightarrow_{\mathsf{dt}} conf_1$，对于所有 $0 < dt' \leq dt$ 以及分解 $conf \rightarrow_{\mathsf{dt}'} conf' \rightarrow_{\mathsf{dt}-\mathsf{dt}'} conf_1$，均有：– 若 $conf \nvDash SP$ 且 $conf_1 \nvDash SP$，其中 $SP = bad$ 或 $bad \prec_1 SP$，则 $conf' \nvDash bad$，则称 dt 关于 $Spec$和 LS 是坏充分的。

以下命题类似于命题1，它确立了验证坏充分性的条件。

命题 3。设 $Spec = \langle \mathcal{S}, \prec_1$,坏 $\vDash \rangle$ 是逻辑场景 逻辑场景的 RTA 安全属性规范。设 dt 相对于 $Spec$ 是坏充分的。对于 逻辑场景 的所有分解conf $\rightarrow_{\mathsf{dt}'}$ conf$' \rightarrow_{\mathsf{dt}-\mathsf{dt}'}$ conf$_1$，conf$' \vDash$ 坏 当且仅当 conf $\vDash$ SP$_0$ 且conf$_1 \vDash$ SP$_1$，其中 {SP$_0$,SP$_1$} $\subseteq$ {坏} $\cup$ {SP $\mid$ 坏 $\prec_1$ SP}。

以下命题确立了不同充分性定义之间的关系。我们的实验表明，dt 可能是坏充分的，但不是$\prec_1$充分的。

命题 4。令 $Spec = \langle \mathcal{S}, \prec_1$,不安全状态 $\vDash \rangle$ 为逻辑场景 逻辑场景的安全属性规范，并令采样时间 为采样时间。– 如果采样时间 是单步转移充分的，那么采样时间 是 $\prec_1$-充分的，并且采样时间 是不安全状态-充分的。– 如果采样时间 是 $\prec_1$-充分的，那么采样时间是不安全状态-充分的。

5 基于 RTA 的可恢复性性质

韧性有许多非正式定义 [3,5,6,18]。从最广义上讲，韧性是“系统适应和应对变化的能力（包括环境变化和内部变化）”[6]。NIST [28] 给出了一个更精确但仍然非正式、且更侧重于攻击的韧性定义：“预见、承受、恢复并适应使用网络资源或由网络资源支持的系统所面临的不利条件、压力、攻击或破坏的能力。”

直观而言，实施 RTA 的系统（例如逻辑场景实例中的自动驾驶车辆）能够展现一种基本形式的韧性，我们称之为可恢复性：它们会检测到指定风险等级是否已达到，并采取调整以降低风险。我们的目标是通过精确定义，将对 RTA 可恢复性的这一直观认识形式化。

为此，我们必须对系统的控制方面以及物理状态的变化进行建模。因此，我们用关系 $\rightarrow_{\mathsf{tasks}}$ 扩展语义关系 $\rightarrow_{\mathsf{dt}}$（上文称为 $\longrightarrow_{\Delta}$），该关系用于对控制方面进行建模，通常包括感知、分析以及决策/规划。形式上，系统行为是一组（可能无限的）执行轨迹：

配置$_0 \rightarrow_{\mathsf{tasks}}$配置$'_0 \rightarrow_{\mathsf{dt}}$配置$_1 \rightarrow_{\mathsf{tasks}}$配置$'_1 \rightarrow_{\mathsf{dt}}$配置$_2 \rightarrow_{\mathsf{tasks}} \cdots$其中 dt 是系统的采样时间，配置$'_i \rightarrow_{\mathsf{dt}}$配置$_{i+1}$是一个函数，并且 配置$_i \rightarrow_{\mathsf{tasks}}$配置$'_i$是一个内部转换，用于规定 配置$_i$中智能体的行为，例如感知、更新局部知识库以及决定要采取哪些动作。该转换的确切定义取决于系统规范。由于安全属性与系统的物理属性有关，例如速度和位置，我们通常假设，如果 配置$_i \vDash$ SP，那么 配置$'_i \vDash$ SP也成立。例如，示例 1中的安全属性就是这种情况。该假设并非严格必要，因为下面的定义可以扩展到不满足该假设的情况。

定义5。设 $Spec = \langle \mathcal{S}, \prec_1, \text{不安全状态} \vDash \rangle$ 是逻辑场景 LS 的安全属性规范，dt 为采样时间，其中 $\mathsf{SP}_{\mathsf{safe}} \in \mathcal{S}$ 是最小可接受安全属性，而 $\mathsf{SP}_{\mathsf{safer}} \in \mathcal{S}$ 是可接受的更安全属性，且 $\mathsf{SP}_{\mathsf{safe}} \prec \mathsf{SP}_{\mathsf{safer}}$。设 t 为正自然数。若对于逻辑场景 LS 的所有实例 conf_0 及轨迹 $\tau = \mathsf{conf}_0 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_0 \rightarrow_{\mathsf{dt}} \mathsf{conf}_1 \rightarrow_{\mathsf{tasks}} \mathsf{conf}'_1 \rightarrow_{\mathsf{dt}} \cdots$ 使得 $\mathsf{conf}_0 \vDash \mathsf{SP}$ 满足 $\mathsf{SP} = \mathsf{SP}_{\text{安全}}$ 或 $\mathsf{SP}_{\text{安全}} \prec \mathsf{SP}$：
– 对于 τ中的所有 $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}} \mathsf{conf}_{i+1}$，不存在分解 $\mathsf{conf}'_i \rightarrow_{\mathsf{dt}_1} \mathsf{conf} \rightarrow_{\mathsf{dt}-\mathsf{dt}_1} \mathsf{conf}_{i+1}$，使得 $0 \leq \mathsf{dt}_1 \leq \mathsf{dt}$，且 $\mathsf{conf} \vDash$ 不安全状态。也就是说，系统永远不会到达满足不安全状态的配置。– 对于 τ中的所有 conf_i，若 $\mathsf{conf}_i \vDash \mathsf{SP}_{\mathsf{safe}}$，则 $\mathsf{conf}_{i+t} \vDash \mathsf{SP}$ 满足 $\mathsf{SP}_{\mathsf{safer}} \prec \mathsf{SP}$ 或$\mathsf{SP}_{\mathsf{safer}} = \mathsf{SP}$。也就是说，如果系统达到最小安全属性，就必然会恢复到可接受的更安全属性。

该定义形式化了系统从更高风险等级中恢复的能力，如图1所示。直观而言，性质$\mathsf{SP}_{\mathsf{safe}}$ 规定了系统在应采取措施降低风险之前所能接受的最高风险，即 RTA实例触发安全控制器的时刻；而$\mathsf{SP}_{\mathsf{safer}}$ 规定了系统应在t 个逻辑时钟周期内达到的风险等级，即 $t \times \mathsf{dt}$，也就是 RTA实例恢复使用主控制器输出的时刻。

此定义中有一些值得指出的细微之处：

– 恢复周期：定义 t 中规定的时间 5 指定了系统进行恢复所需的时间。一方面，它避免系统在较长时间内处于较高的风险状态（尽管仍然安全），从而降低发生安全事故的可能性。另一方面，如果 t 过小，则需要更严格的安全控制器，或者考虑到车辆的能力（例如最大减速度）而无法实现。因此， t 的取值将取决于所考虑的具体情况。为缓解这一问题，我们在第 6 节提出了验证可恢复性的自动化方法。

– 可恢复性平滑性：请注意，我们要求 $\mathrm{SP}_{\mathsf{safe}} \prec \mathrm{SP}_{\mathsf{safer}}$ ，而不是 $\mathrm{SP}_{\mathsf{safe}} \prec_1 \mathrm{SP}_{\mathsf{safer}}$，即 $\mathrm{SP}_{\mathsf{safer}}$ 的风险可以比 $\mathrm{SP}_{\mathsf{safe}}$低多个等级。通过适当选择这些性质，例如将 $\mathrm{SP}_{\mathsf{safer}}$ 的风险设置得远低于 $\mathrm{SP}_{\mathsf{safe}}$，可以避免系统在正常运行（使用主控制器）和恢复运行（使用安全控制器）之间振荡。

证明可恢复性的流程 根据定义 5 证明系统具有弹性的一个挑战是 需要对所有可能具有无限长度的轨迹以及轨迹的所有分解进行推理。为应对这一挑战，我们证明（下文的定理 1 ），只要 dt 是充分的（如第 3节所述），dt 足够精细，不会跳过性质（下文的定义 7 ），并且仅考虑以下定义所规定的有界大小轨迹即可：

定义 *6*。令 $Spec$、*LS*、SP_{safer}、SP_{safe}、t 如定义*5*所述，且 *dt* 为采样时间。如果对于所有满足 $conf_0 \vDash SP_{\mathsf{safe}}$ 的轨迹$\tau = conf_0 \rightarrow_{\mathsf{tasks}} conf'_0 \rightarrow_{\mathsf{dt}} conf_1 \rightarrow_{\mathsf{tasks}} \cdots \rightarrow_{\mathsf{dt}} conf_t$，系统 S 是 $\langle SP_{\mathsf{safe}}、SP_{\mathsf{safer}}、t\rangle$−单周期可恢复的：
1. 置信$_t \vDash \mathrm{SP}_{\mathsf{safer}}$——系统在 t 个时间刻度内恢复到风险较低的状态。*2.* 对于所有 τ 中的 置信$'_i \rightarrow_{\mathsf{dt}}$ 置信$_{i+1}$，不存在分解 置信$'_i \rightarrow_{\mathsf{dt}_1}$ 置信 $\rightarrow_{\mathsf{dt}-\mathsf{dt}_1}$置信$_{i+1}$，使得 $0 \leq \mathsf{dt}_1 \leq \mathsf{dt}$，且 置信 $\vDash$ 不安全状态。

为证明无界轨迹的可恢复性（定理 1），我们还需要确保不会跳过触发 RTA 的性质 $\mathrm{SP}_{\mathsf{safe}}$ 。以下定义对此进行了形式化说明。

定义 *7.* 设 $Spec$、逻辑场景、SP_{safer}、SP_{safe} t 如定义*5*所述，*dt* 为采样时间。若不存在如下形式的转移：$conf \longrightarrow_{\mathsf{dt}} conf_1$，其中 $conf \vDash SP$ 且 $conf_1 \vDash SP_1$，并且 $SP_{\mathsf{safe}} \prec SP$、$SP_1 \prec SP_{\mathsf{safe}}$，则称 *dt* 不会跳过性质 SP_{safe} 。

定理1。 设dt 为单次转换或 $\prec_1$ 或不安全状态-充分，其中dt 不跳过$\mathrm{SP}_{\mathsf{safe}}$。系统 S 是 $\langle \mathrm{SP}_{\mathsf{safe}}、\mathrm{SP}_{\mathsf{safer}}、t\rangle$-单周期可恢复的，当且仅当 S 是$\langle \mathrm{SP}_{\mathsf{safe}}、\mathrm{SP}_{\mathsf{safer}}、t\rangle$-可恢复的。

检查单恢复周期可恢复性的条件：即使只考虑单恢复周期可恢复性，仍有必要考虑所有可能的 dt 状态转移（定义2 中的6）的分解。根据 dt 充分性的类型，这一点可以克服：利用命题 1、2和 3，只需检查所有配置 conf_i 对于 $0 \leq i \leq t$ 既不满足 不安全状态，也不满足某个 SP，使得 不安全状态 $\prec_1$ SP。

6 实验结果

我们使用符号软代理框架[23] 和模 SMT 符号重写开展了一系列实验，具体描述见第 6.1 节。第 6.2 节介绍了自动证明 dt-充分性的实验。第 6.3 节介绍了自动证明定时可恢复性的实验。所有实验均使用dt $= 0.1s$ 值。如果一小时后仍未返回结果，则中止实验。所有实验均在一台配备 16 GB 内存的 2.2 GHz 六核 Intel Core i7 机器上进行。代码位于文件夹 rta_symbolic_agents 中，网址为 https://github.com/SRI-CSL/VCPublic。

我们考虑了以下所述的场景：
– pedCross($\text{gap}_{\text{safer}}$,$\text{gap}_{\text{safe}}$,$\text{gap}_{\text{unsafe}}$,senerr) – 仅使用相对距离的行人穿越：该场景是第 2 节中描述的行人过街场景。该场景的安全属性如公式 1 所述，仅使用相对距离，并由以下数值参数化：$\text{gap}_{\text{safer}}$、$\text{gap}_{\text{safe}}$、$\text{gap}_{\text{unsafe}}$。我们假设，用于检测行人及其性质（即速度、位置和方向）的传感器可能并不完美。也就是说，车辆用于决定将采取何种动作的局部知识库，可能与真实值不一致。具体而言，车辆推断出的行人位置可能会与行人的实际距离存在一定的成比例偏差。

误差 err 与车辆和行人之间的距离（pos^p $^-$pos）成正比，具体如公式所示

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

在这种情况下，车辆的安全控制器会采取保守策略，例如更积极地降低车辆速度，从而仍然满足定时可恢复性性质。当 senerr $= 0$ 时，传感器没有故障。

– pedCrBnds(间隙$_{\text{safer}}$,间隙$_{\text{safe}}$,间隙$_{\text{unsafe}}$,传感器误差) – 使用 更安全 规范，通过低速度和较大距离进行设定：这与前一种情况类似，但现在我们使用由公式 更安全 所规定的安全属性，如公式 2所示。

– folRSS(maxdec_l) – 基于 RSS 属性的车辆跟驰：此场景涉及使用基于 RSS 属性的安全属性进行车辆跟驰，该安全属性如示例 2中所述。我们根据假设的前车最大减速度（maxdec_l）对安全属性进行参数化。我们遵循 [19] 中开展的分析。该工作根据预期的前车减速情况确定了三个场景。第一种场景发生的可能性极低，即前车紧急制动（$\text{maxdec}_l = -8m/s^2$）；第二种是前车大幅减速（$\text{maxdec}_l = -5m/s^2$）；最可能的情况是前车正常减速（$\text{maxdec}_l = -2m/s^2$）。

– folGap($\text{gap}_{\text{safer}}$,$\text{gap}_{\text{safe}}$,$\text{gap}_{\text{unsafe}}$) – 具有间隔距离性质的车辆跟驰：该场景在 [23] 中有更详细的描述。具体而言，我们使用基于间隔距离的安全属性，类似于人行横道场景。

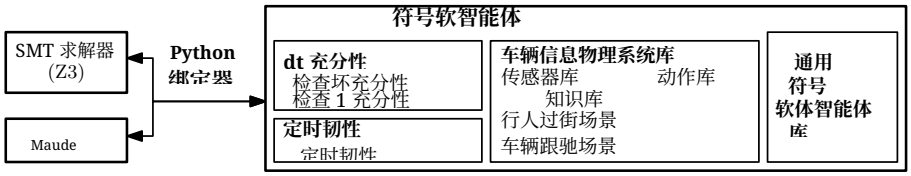

图 4. 用于自动化可恢复性证明的关键库和工具。

6.1 使用 SymbolicSoft-Agents 自动化可恢复性证明

图 4展示了已实现并使用的主要机制。该机制基于软代理框架 [33]以及 [23] 中所述的通用符号库。通用符号软智能体库通过重写规则规定了基于网络物理系统的可执行语义。符号软智能体重写规则与两个逻辑场景 关系 $\rightarrow_{\mathsf{tasks}}$ 和 $\rightarrow_{\mathsf{dt}}$直接对应。我们实现了用于描述车辆场景的车辆专用库。此外，我们还实现了用于检查定时充分性（不安全状态充分性和 $\prec_1$充分性）以及定时可恢复性的机制。

符号软智能体是可执行规范。具体而言，Maude [10] search 会枚举执行轨迹。轨迹中的约束（非线性算术公式）由 SMT 求解器（Z3 [13]）求解。我们使用 [29]，中所述的 Python 绑定器，实现了符号软智能体库与 SMT 求解器之间的连接，从而便于未来扩展到其他求解器和工具。

基本思路是搜索针对所关注性质的反例。由于符号搜索是完备的，未能找到反例意味着该性质对所考虑 LS 的所有实例均成立。[7]

例如，为了检查坏充分性，该算法依据命题2 搜索反例，即搜索满足以下条件的性质 $\mathrm{SP}_0 \prec_1 \mathrm{SP}_1$ （不安全状态）和逻辑场景实例 $\mathrm{conf}_0, \mathrm{conf}_1$： conf_0 满足 SP$_0$, conf_1 满足 SP_1, $\mathrm{conf}_0 \rightarrow_{\mathsf{dt}} \mathrm{conf}_1$, ，并且存在 dt_0 ，使得 $0 < \mathrm{dt}_0 < \mathrm{dt}$，conf$_2$ ，且 $\mathrm{conf}_0 \rightarrow_{\mathrm{dt}_0} \mathrm{conf}_2 \rightarrow_{\mathrm{dt}-\mathrm{dt}_0} \mathrm{conf}_1$，其中 conf_2 满足不安全状态。如果未找到反例，则对于给定的逻辑场景、dt 和性质规约，坏充分性成立。

使用符号重写，逻辑场景的任意实例都由一个项表示，即 asys，该项由符号代理配置和符号环境组成。环境包含关于物理状态的知识以及对符号值的约束。配置满足性质 SP 的断言由项 enforce(asys，SP) 表示；该项将用 asys 的符号表示 SP 的布尔项与环境中的约束合取起来。cond(asys) 是 asys 的环境部分中的约束。

基本情况是对一对性质 $\mathrm{SP}_0, \mathrm{SP}_1$ 的充分性。此情况的算法执行以下操作。首先，从 asys_0 = enforce(asys,SP_0) 开始进行符号搜索，寻找某个 asys_1 ，使得 $\mathrm{asys}_0 \rightarrow_{\mathsf{dt}} \mathrm{asys}_1$ 且 cond(enforce(asys_1,SP_1))) 可满足。如果找不到这样的 asys_1 ，则对于给定的 SP_0,SP_1，dt 充分性成立。否则，对于某个

[7] 在我们的场景中，我们考虑有限长度的轨迹。状态的数量可能是无限的，但由于状态由符号项有限表示，搜索最终会终止。

行人过街场景		
场景	不安全状态-**充分性**	$\prec_1$-**充分性**
pedCross(3，2，1，0)	是（130秒）	DNF
pedCrBnds(3，2，1，0)	是（172秒）	否（358秒），从安全到更安全的案例失败。
pedCross(5，2，1，0)	是（89秒）	是（149秒）
pedCrBnds(5，2，1，0)	是（78秒）	否（172秒），从安全到更安全的案例失败。
folGap(3，2，1)	DNF	是（1413秒）
folGap(6，4，2)	是（51秒）	否（52秒），从安全状态到安全状态的案例失败。
folGap(7，5，1)	是（55秒）	是（83秒）
folRSS(-8)	DNF	DNF
folRSS(-5)	DNF	DNF
folRSS(-2)	是（304秒）	是（533秒）

表 1. 针对不同场景的不安全状态和 $\prec_1$-充分性的自动化证明。析取范式表示实验在一小时后中止。

找到的 asys_1，从（asys_0 的副本）开始进行符号搜索，寻找某个 $\text{asys}_2, \text{dt}_0,$，其中 dt_0 为符号的，并且满足 $\text{asys}_0 \rightarrow_{\text{dt}_0} \text{asys}_2$，以及

$\text{cond}(\text{enforce}(\text{asys}_1, \text{SP}_1)) \wedge \text{cond}(\text{enforce}(\text{asys}_2, \text{不安全状态})) \wedge 0 < \text{dt}_0 < \text{dt}$ 可满足。如果找到这样的 $\text{asys}_1, \text{asys}_2, \text{dt}_0$，我们就得到了一个反例；否则，$\text{SP}_0, \text{SP}_1$ 满足坏充分性。

其余用于证明 $\prec_1$-充分性、noSkip 性质和 t-可恢复性的算法遵循与坏充分性相同的模式。

6.2 dt-充分性实验

表 1 展示了我们用于证明dt-充分性的主要实验。由于每个场景都有四个层级的性质（不安全状态、不安全、安全、更安全），因此需要考虑十种情况，例如，从满足更安全 的配置开始，并最终到达另一个满足安全 的配置，其他情况以此类推。

行人过街场景： 软体智能体机制能够在不到 3 分钟内证明坏充分性。然而，对于 $\prec_1$-充分性，软体智能体机制无法针对场景 pedCross(3，2，1，0) 返回结果（不使用显式边界）。具体而言，SMT 求解器在一小时内既无法证明，也无法找到反例。如果将 $\text{gap}_{\text{safer}}$和 gap_{safe}的值分别增加到 5 和 2，那么软体智能体机制将能够成功终止。虽然由于 SMT 求解器会应用多种启发式方法，很难对此给出形式化解释，但直观上这是预期结果，因为这些新值会产生更粗粒度的安全属性。

此外，这些 $pedCrBnds$场景不满足 $\prec_1$-充分性。具体而言，有一个案例未通过，即从安全状态到更安全状态。这似乎表明，在可恢复性分析中可以将安全状态和更安全状态合并，因为我们仍然能够检测到向较低层级性质（不安全状态和不安全状态）的状态转移。

车辆跟驰场景： 这两组场景对软体智能体机制而言都颇具挑战性。与人行横道示例不同，证明 folGap 的 $\prec_1$-充分性以及针对不安全状态的坏充分性不可终止性更为容易。有趣的是，当将 $\text{gap}_{\text{safer}}$、$\text{gap}_{\text{safe}}$、$\text{gap}_{\text{unsafe}}$ 的边界分别增加到 6、4 和 2 时，$\prec_1$-充分性

⟨安全，更安全 t⟩-One-Recovery-Period		
行人过街场景		
场景	$t=4$	$t=5$
pedCross(3，2，1，0)	否（34 秒）	否（115 秒）
pedCrBnds(3，2，1，0)	否（27秒）	是（621秒）
pedCross(5，2，1，0)	否（27秒）	否（93秒）
pedCrBnds(3，2，1，0。50)	–	否（103秒）
pedCrBnds(3，2，1，0。33)	–	否（104秒）
pedCrBnds(3,2,1,0.125)	–	是（637秒）
pedCrBnds(3，2，1。1)	–	是（734秒）
车辆跟驰场景		
场景	**可恢复性**	
folGap(3,2,1)	$t=5$	否（12 秒）
folGap(6，4，2)	$t=5$	否（11 秒）
folGap(7,5,1)	$t=5$	否（12 秒）
folRSS(-5)	$t=2$	不（5秒）
folRSS(-5)	$t=3$	否（81 秒）
folRSS(-5)	$t=4$	不（1126秒）
folRSS(-5)	$t=2$	是（38秒）$\star$
folRSS(-2)	$t=2$	是（43秒）

表2。 有时限可恢复性的自动化证明。符号 $\star$ 表示该实验使用了非常激进的控制器。由于场景 pedCrBnds(3,2,1, 0) 对于 $t=4$ 不可恢复，因此无需对标记为–的场景开展实验。

在从安全状态到安全状态的情况下证明失败了；但当进一步将这些值增加到 7、5 和 1 时，证明得以成立。这表明 $\text{gap}_{\text{unsafe}}$ 的值为 2 并不具备充分性，因为系统能够在 dt 内遍历一个满足不安全状态的配置。对于 folRSS，只有在假设前车的最大减速度为 $-2m/s^2$时，软体智能体机制才能证明这两个充分性性质。

总而言之，除 folRSS(-5) 和 folRSS(-8) 之外，对于所有场景，软体智能体机制都能够自动证明不安全状态和 $\prec_1$ 充分性。folRSS(-5) 和 folRSS(-8) 的情况更具挑战性，关于如何改进该机制或 CPS建模以处理这些情况的研究将留待未来进行。

6.3 有时限可恢复性实验

表 2 总结了我们在行人横道和车辆跟驰场景中针对可恢复性开展的主要实验。请回想， ⟨安全，更安全 t⟩- 可恢复性的目标是证明安全控制器有能力将车辆风险降低至更安全。实验中，我们通常使用简单且相当谨慎的控制器。为便于说明，我们针对车辆跟驰场景指定了两个控制器：非激进安全控制器和激进控制器。后者始终启用紧急制动，即最大减速度。最后，对于每个

场景，我们的机制表明dt 不会跳过安全（参见定义7），耗时约一秒。

行人过街场景：首先观察到，使用 pedCross 中的安全属性无法证明可恢复性。我们的工具返回了一个反例，其中车辆速度极低，且非常接近人行横道，距离约为 $0.5m$。这说明了按方程 pedCrBnds 中的做法，将边界纳入安全属性的重要性，见方程 2。

对于场景 pedCrBnds(3，2，1，0)，安全控制器总能在 5 个时步后回到更安全的风险状态，但在 4 个时步后则不行。请注意，对于 pedCrBnds(5，2，1，0)，情况就不再如此，因为即使在 5 个时步后也会失败。这是意料之中的，因为安全与更安全这两个性质之间的“距离”增大了。

最后，针对 pedCrBnds(gap_{safer},gap_{safe},gap_{unsafe},senerr) 的实验说明了如何检查存在故障传感器时安全控制器的可恢复性。假设行人传感器存在 50% 或 33% 的故障，安全控制器就无法保证总能回到更安全的风险状态。不过，当误差为 12.5% 或 10% 时，它能够做到这一点。

车辆跟驰场景： 我们的实验表明，使用时间间隔来确定风险等级时，似乎更难建立可恢复性。这可能需要更复杂的安全控制器。另一方面，使用基于 RSS 的属性时，即使时间范围较小，在假设前车正常减速度的情况下，也可以建立可恢复性。对于假设更高减速度值的场景，同样可以建立可恢复性，但这需要更加激进的控制器。

7 相关工作

实时自主系统。 自最初提出称为单纯形架构的实时自主系统 [30], 以来，近期已经提出了多种实时自主系统变体 [25,20,11] （仅举数例）。尽管它们在架构和功能上存在一些差异，但都包含一个决策模块，用于评估系统风险等级，以决定使用哪个控制器（安全控制器或高级控制器）。因此，本文形式化的所有要求，即时间采样充分性和可恢复性，仍然具有相关性和适用性。事实上，我们通过提供适用于自动化验证的恰当定义，推进了该领域的研究现状。

我们的工作受到 [14] 的启发，该文根据系统的风险等级，提出了关于实时自主系统可恢复性的高层要求。具体而言，检查采样充分性和 t-可恢复性的方法，分别对应于 RTA 良构性的安全性要求和活性要求。第三个条件涉及任何控制器负责时系统变为不安全（非安全）状态所需的最短时间，这对于确保监控器能够切换控制器，且安全控制器能够在系统到达不安全状态之前作出反应至关重要。借助 $<_1$-充分性以及 $<_1$-链中性质的连续性，可以证明这一点。总而言之，符号重写结合 SMT 求解，为验证时间采样机制的正确性以及安全要求（例如 [14] 的 RTA 框架所提出的要求）提供了自动化方法。

在类似的方向上， [20] 提出了关于决策模块正确性的高层要求，这些要求基于对安全的定义以及“永久安全的命令序列”的存在性；这似乎与我们的时间可恢复性性质相关。然而，他们没有研究时间采样的影响，也没有探究决策模块的正确性。

CPS 验证与确认 CPS 验证领域的大量文献，例如 [16] 其中一项，包括此前一些关于 RTA 的研究 [25,11,20], , 都依赖基于仿真的方法。这些方法与本文提出的方法相辅相成。本文的方法着眼于开发的早期阶段，通过提供证明来确保 RTA 规约适用于某个逻辑场景的所有实例；而基于仿真的方法则侧重于后期工作，即验证和测试 RTA 系统在逻辑场景特定实例上的实现。

微分动态逻辑（*dL*）、*KeYmaera X* 和 *VeriPhy*。 KeYmaera X 证明器 [34,24] 使用微分动态逻辑（dL）来规约和验证信息物理系统控制器设计。它是 VeriPhy 流程 [7,8] 的起点，该流程用于根据逻辑规约生成代码。dL 规约和逻辑场景的共同点在于，它们都由带约束变量的项表示，这些变量涵盖所有满足给定约束的取值实例。二者的不同之处在于，dL 规约不能直接执行，因此需要使用交互式定理证明方法来验证 dL 规约；而逻辑场景可以执行，因而能够通过模 SMT 重写进一步自动化验证证明。

韧性的形式化定义： Alturki 等人 [4]提出了韧性的形式化定义，并证明一般情况下这些定义是不可判定的，而在某些情况下是 PSPACE 完全的。虽然形式化联系留待未来研究，但我们对定时可恢复性的定义似乎是对他们定义的特化，使其适用于 RTA 架构，例如考虑 dt 充分性。

8 结论

本文介绍了使用抽象逻辑场景（LS）自动证明安全属性的方法。一个 LS 由满足给定运行设计域约束的模式实例以及一个赋予其语义的两步转移关系构成。第一步对应于读取传感器、分析并决定动作（设置控制参数）。第二步则使系统在两次观测之间演进一个采样时间。为建立形式化基础，我们引入了 LS 的安全属性规约概念，将其定义为一组性质（名称），并配有一个风险等级序关系、一个唯一的最小（风险最高）元素 bad，以及一个满足关系。充分的采样时间应确保不会遗漏任何重要信息。我们定义了三种 dt 充分性概念，并证明它们彼此不同且构成全序。系统可能被允许进入一种安全但有风险的情形，而弹性系统则会恢复到可接受的安全状态。我们通过 t-可恢复性的定义将这一点形式化。我们定义了一种便于验证的单周期恢复 t-可恢复性概念，并通过归纳论证证明，在 dt 充分时，它等价于 t-可恢复性。

为实现证明的自动化，我们使用模 SMT 符号重写作为执行和搜索引擎 [23]。我们开发了算法，以证明某个逻辑场景的所有（无限多个）

实例满足不同的 dt 充分性概念或 t-可恢复性（或提供反例实例）。我们报告了一组实验，检查与车辆自动化相关的逻辑场景和安全属性规范的 dt 充分性及 t-可恢复性性质：车辆跟驰和人行横道。实验表明，可以找到使充分性成立的 dt 和安全参数取值，并且非常简单的控制器也能满足 t-可恢复性。实验还揭示了一些可能出问题的边界情况。

未来工作的一个方向是研究更广泛的案例研究，以更好地理解不同设计参数之间的相互作用。另一个重要方向是开发组合逻辑场景与证明的方法，从而扩展对复杂系统的分析。

致谢。 Talcott 获得了美国海军研究办公室部分资助，资助编号为 N00014-15-1-2202 和 N00014-20-1-2644，并获得了海军研究实验室拨款 N0017317-1-G002。我们还要感谢匿名评审者对本文早期版本提出的宝贵意见。

参考文献

1. Apollo：开放式自动驾驶平台。 https://github.com/ApolloAuto/apollo。2. I. 21448:2019。 https://www.iso.org/standard/70939.html。2021。3. B. Allenby 和 J. Fink。迈向内在安全且具韧性的社会。 *Science*，309(5737):1034–1036，2005。4. M. A. Alturki、T. B. Kirigin、M. I. Kanovich、V. Nigam、A. Scedrov 和 C. L. Talcott。信息物理系统韧性问题的形式化与计算复杂性。载于 H. Seidl、Z. Liu 和 C. S. Pasareanu 编，计算理论方面 - *ICTAC 2022* - 第 *19* 届国际学术讨论会，格鲁吉亚第比利斯，*2022* 年 *9* 月 *27-29* 日，论文集，收录于 计算机科学讲义第 13572 卷，第 96–113 页。施普林格，2022。5. K. Barker、J. E. Ramirez-Marquez 和 C. M. Rocco。基于韧性的网络组件重要性度量。 可靠性工程与系统安全，117:89–97，2013。6. R. Bloomfield、G. Fletcher、H. Khlaaf、P. Ryan、S. Kinoshita、Y. Kinoshit、M. Takeyama、Y. Matsubara、P. Popov、K. Imai 等。旨在识别并弥合自动驾驶道路车辆保障缺口：技术说明汇编，第 1 部分。 *arXiv* 预印本 *arXiv:2003.00789*，2020。7. B. Bohrer、Y. K. Tan、S. Mitsch、M. O. Myreen 和 A. Platzer。VeriPhy：基于经验证的信息物理系统模型生成经验证的控制器可执行程序。载于 第 *39* 届 *ACM SIGPLAN* 编程语言设计与实现会议论文集。AC M New York，2018。8. B. Bohrer、Y. K. Tan、S. Mitsch、A. Sogokon 和 A. Platzer。地面机器人航点跟随的形式化安全网。 *IEEE* 机器人与自动化快报，2019。arxiv:1903.15073。9. M. Bozga 和 J. Sifakis。自动驾驶系统的规约与验证：多层语义框架。 *CoRR*，abs/12109.06478，2021。10. M. Clavel、F. Durán、S. Eker、P. Lincoln、N. Martí-Oliet、J. Meseguer 和 C. Talcott。 *Maude* 全书：高性能逻辑框架，第 4350 卷，收录于 计算机科学讲义。施普林格，2007。11. A. Damare、S. Roy、S. A. Smolka 和 S. D. Stoller。基于屏障证书并应用于微电网的单纯形架构。载于 T. Dang 和 V. Stolz 编， 运行时验证 - 第 *22* 届国际会议，*RV 2022*，格鲁吉亚第比利斯，*2022* 年 *9* 月 *28-30* 日，论文集，第 13498 卷，收录于计算机科学讲义，第 105–123 页。施普林格，2022。12. Y. G. Dantas、V. Nigam 和 C. L. Talcott。协同自适应巡航控制的形式化安全评估框架。载于 *IEEE* 车辆网络会议，*VNC 2020*，美国纽约州纽约市，*2020* 年 *12* 月 *16-18* 日，第 1–8 页。IEEE，2020。13. L. M. de Moura 和 N. Bjørner。Z3：高效的 SMT 求解器。载于 C. R. Ramakrishnan 和 J. Rehof 编， 系统构造与分析的工具和算法，第 *14* 届国际会议，*TACAS 2008*，作为软件理论与实践联合欧洲会议 *ETAPS 2008* 的一部分举行，匈牙利布达佩斯，*2008* 年 *3* 月 *29* 日至 *4* 月 *6* 日，论文集，第 4963 卷，收录于 计算机科学讲义，第 337–340 页。施普林格，2008。14. A. Desai、S. Ghosh、S. A. Seshia、N. Shankar 和 A. Tiwari。SOTER：用于编程安全机器人系统的运行时保障框架。载于第 *49* 届 *IEEE/IFIP* 可信系统与网络国际会议，*DSN 2019*，美国俄勒冈州波特兰，*2019* 年 *6* 月 *24-27* 日，第 138–150 页。IEEE，2019。

15. D. J. Fremont、T. Dreossi、S. Ghosh、X. Yue、A. L. Sangiovanni-Vincentelli 和 S. A. Seshia。Scenic：一种用于场景规范和场景生成的语言。载于 K. S. McKinley 和 K. Fisher 编，第*40*届 *ACM SIGPLAN* 编程语言设计与实现会议论文集，*PLDI 2019*，美国亚利桑那州菲尼克斯，*2019*年*6*月*22*日至*26*日，第63–78页。ACM，2019年。16. D. J. Fremont、E. Kim、Y. V. Pant、S. A. Seshia、A. Acharya、X. Bruso、P. Wells、S. Lemke、Q. Lu 和 S. Mehta。自动驾驶车辆的形式化的基于场景测试：从仿真到现实世界。载于第*23*届 *IEEE* 智能交通系统国际会议，*ITSC 2020*，希腊罗得岛，*2020*年*9*月*20*日至*23*日，第1–8页。IEEE，2020年。17. S. Jha、J. Rushby 和 N. Shankar。自主系统的以模型为中心的保障。载于 A. Casimiro、F. Ortmeier、F. Bitsch 和 P. Ferreira 编，计算机安全性、可靠性与安全保障——第*39*届国际会议，*SAFECOMP 2020*，葡萄牙里斯本，*2020*年*9*月*16*日至*18*日，论文集，计算机科学讲义第12234卷，计算机科学讲义，第228–243页。施普林格，2020年。18. J.-C. Laprie。从可信性到韧性。载于第*38*届 *IEEE/IFIP* 可信系统与网络国际会议，第G8–G9页。Citeseer，2008年。19. P. D. Luca Mengani。危险分析、风险评估和功能安全概念。技术报告，2019年。H2020项目 ENSEMBLE 的 D2.11，www.platooningensemble.eu。20. U. Mehmood、S. Sheikhi、S. Bak、S. A. Smolka 和 S. D. Stoller。用于自主网络物理系统运行时保障的黑盒单纯形架构。载于 J. V. Deshmukh、K. Havelund 和 I. Perez 编，*NASA*形式化方法——第*14*届国际研讨会，*NFM 2022*，美国加利福尼亚州帕萨迪纳，*2022*年*5*月*24*日至*27*日，论文集，计算机科学讲义第13260卷，计算机科学讲义，第231–250页。施普林格，2022年。21. T. Menzel、G. Bagschik 和 M. Maurer。自动化车辆开发、测试和验证的场景。载于 *2018 IEEE* 智能车辆研讨会，*IV 2018*，中国江苏苏州常熟，*2018*年*6*月*26*日至*30*日，第1821–1827页。IEEE，2018年。22. J. Meseguer。作为并发统一模型的条件重写逻辑。理论计算机科学，96(1)：73–155，1992年。23. V. Nigam 和 C. Talcott。使用模 SMT 重写自动化信息物理系统的安全性证明。载于 K. Bae 编，第*14*届重写逻辑及其应用国际研讨会，第13252卷计算机科学讲义，第212–229页。施普林格，2022年。24. J.-D. Quesel、S. Mitsch、S. Loos、N. Aréchiga 和 A. Platzer。如何使用 KeYmaera 对混合系统建模并进行证明：安全性教程。国际软件工具与技术转移杂志，18：67–91，2016年。25. S. Ramakrishna、C. Hartsell、M. P. Burruss、G. Karsai 和 A. Dubey。用于学习赋能信息物理系统的动态加权单纯形策略。系统架构期刊，111：101760，2020年。26. S. Riedmaier、T. Ponn、D. Ludwig、B. Schick 和 F. Diermeyer。自动化车辆基于场景的安全性评估综述。*IEEE Access*，8：87456–87477，2020年。27. C. Rocha、J. Meseguer 和 C. Muñoz。模 SMT 重写与开放系统分析。逻辑与代数编程方法期刊，第269–297页，2017年。28. R. Ross、V. Pillitteri、R. Graubart、D. Bodeau 和 R. McQuaid。开发网络弹性系统：一种系统安全工程方法。技术报告，美国国家标准与技术研究院，2019年。29. R. Rubio。作为库的 Maude：一种高效的通用编程接口。载于 K. Bae 编，重写逻辑及其应用——第*14*届国际研讨会，第13252卷计算机科学讲义，第274–294页。施普林格，2022年。30. D. Seto、B. Krogh、L. Sha 和 A. Chutinan。用于安全在线控制系统升级的单纯形架构。载于*1998*年美国控制会议论文集。*ACC*（*IEEE*目录编号：*98CH36207*），第6卷，第3504–3508页，第6卷，1998年。

31. L. Sha。利用简单性控制复杂性。*IEEE Software*，18(4)：20–28，2001年。32. S. Shalev-Shwartz、S. Shammah 和 A. Shashua。关于安全且可扩展的自动驾驶汽车形式化模型。*CoRR*，abs/1708.06374，2017年。33. C. Talcott、V. Nigam、F. Arbab 和 T. Kappé。鲁棒自适应分布式信息物理系统的形式化规范与分析。载于 M. Bernardo、R. D. Nicola 和 J. Hillston 编，集体自适应系统定量评估的形式化方法，计算机科学讲义。施普林格，2016年。形式化方法暑期学校系列第16版，贝尔蒂诺罗（意大利），2016年6月20日至24日。34. T. K. X. 团队。KeYmaera X：用于混合系统的公理化战术定理证明器，2022年。最后访问于2022年9月22日。35. L. Westhofen、C. Neurohr、T. Koopmann、M. Butz、B. Schütt、F. Utesch、B. Kramer、C. Gutenkunst 和 E. Böde。自动驾驶的关键性度量指标：现状综述与适用性分析。《工程计算方法档案》，abs/2108.02403，2022年。

命题 2 的证明

假设命题中的条件成立，其中 conf 是一个逻辑场景（LS）实例，且 conf $\rightarrow_{\mathrm{dt}'}$conf′ $\rightarrow_{\mathrm{dt}-dt'}$ conf$_1$。分两种情况。首先，假设 $\mathrm{SP}_1 \neq \mathrm{SP}_2$。根据 $\prec_1$性质，我们有 conf′ $\vDash \mathrm{SP}_1$ 或 conf′ $\vDash \mathrm{SP}_2$。由于 $\mathrm{SP}_1 \neq$ 不安全状态，且 $\mathrm{SP}_2 \neq$ 不安全状态，并且各性质互不相交，因此 conf′ $\nvDash$ 不安全状态。其次，假设 $\mathrm{SP}_1 = \mathrm{SP}_2$。根据$\prec_1$ 性质，对于某个性质 SP，我们有 conf′ $\vDash \mathrm{SP}_1$ 或 conf′ $\vDash$ SP，且该性质满足 SP $\prec_1 \mathrm{SP}_1$ 或 $\mathrm{SP}_1 \prec_1$ SP。由于不安全状态 $\neq \mathrm{SP}_1$，且不安全状态 $\not\prec_1 \mathrm{SP}_1$, conf′ $\nvDash$ 不安全状态。

命题 B 的证明 4

假设 dt 是单次转换充分的。令 conf $\rightarrow_{\mathrm{dt}}$ conf$_1$ 为任意转移。对于 $\prec_1$-充分性，需要考虑两种情况：

情况 1：conf $\vDash \mathrm{SP}_1$ 且 conf $\vDash \mathrm{SP}_2$，其中 $\mathrm{SP}_1 \neq \mathrm{SP}_2$,，因此对于任意分解 conf $\rightarrow_{\mathrm{dt}'}$ conf$_2$ $\rightarrow_{\mathrm{dt}-\mathrm{dt}'}$ conf$_1$, ，都有 conf$_2 \vDash \mathrm{SP}_1$ 或 conf$_2 \vDash \mathrm{SP}_2$。根据 dt 的单次转换充分性，我们知道存在一个 $0 < \mathrm{dt}'' < \mathrm{dt}$，即从 SP_1 到 SP_2的转移。我们可以分成两个子情况：

– 情况 1.1：令 $0 < \mathrm{dt}' < \mathrm{dt}''$。根据 dt 的单次充分性，对于所有满足 conf$_2$ 其中 conf $\rightarrow_{\mathrm{dt}_2}$conf$_2$ $\rightarrow_{\mathrm{dt}''-\mathrm{dt}_2}$ conf$_d$的 conf$_2 \vDash$ ，都有 conf$_2 \vDash \mathrm{SP}_1$。因此，对于所有满足 dt′的可达配置 conf′，都有 conf′ $\vDash \mathrm{SP}_1$；– 情况 1.2：令 $\mathrm{dt}'' \leq \mathrm{dt}' < \mathrm{dt}$。推理过程类似，但 SP_2, 导致 conf′ $\vDash\mathrm{SP}_2$。第二种情况，即 conf $\vDash$ SP 且 conf $\vDash$ SP，也遵循相同的推理过程，但此时 $\mathrm{SP}_1 = \mathrm{SP}_2 =$SP。

对于坏充分性，假设配置 $\vDash \mathrm{SP}_1$ 和配置 $_1 \vDash \mathrm{SP}_2$，其中 $\mathrm{SP}_1 \neq$ 不安全状态，且 $\mathrm{SP}_2 \neq$ 不安全状态。根据单次转换充分性，所有中间配置均为配置 ′，要么是配置 ′ $\vDash \mathrm{SP}_1$，要么是配置 ′ $\vDash \mathrm{SP}_2$, ，这意味着配置 ′ $\nvDash$ 不安全状态。

最后，假设 dt 具有 $\prec_1$-充分性。设 conf $\vDash \mathrm{SP}_1$ 并且 conf $\vDash \mathrm{SP}_2$。设 conf $\rightarrow_{\mathrm{dt}'}$ conf′ $\rightarrow_{\mathrm{dt}-\mathrm{dt}'}$ conf$_1$ 为任意分解。假设对于 $i \in \{1, 2\}$ 不安全状态 $\not\prec_1 \mathrm{SP}_i$ 以及不安全状态 $\neq \mathrm{SP}_i$。那么存在两种子情况。情况 1）如果根据充分性 $\mathrm{SP}_1 \neq \mathrm{SP}_2$,，我们有所有 conf′ $\vDash \mathrm{SP}_1$ 或 conf′ $\vDash \mathrm{SP}_2$。情况 2）如果 $\mathrm{SP}_1 = \mathrm{SP}_2$,

根据 $\prec_1$-充分性，$\text{conf}' \vDash \text{SP}'$，且满足 $\text{SP}' \prec_1 \text{SP}_1$ 或 $\text{SP}_1 \prec \text{SP}'$ 之一。在这两种情况下，$\text{SP}' \neq$不安全状态。

定理证明1

假设系统具有一个恢复周期的韧性。令 $\text{conf}_0 \to_{\text{tasks}} \text{conf}' \to_{\text{dt}} \text{conf}_1 \to_{\text{tasks}} \text{conf}'_1 \to_{\text{dt}} \to \text{conf}_2 \cdots$ 为任意一条轨迹，使得 $\text{conf}_0 \vDash \text{SP}$，其中 $\text{SP} = \text{SP}_{\text{safer}}$或 $\text{SP}_{\text{safer}} \prec \text{SP}$。我们可以将轨迹 $\text{conf}_0 \to_{\text{tasks}} \text{conf}' \to_{\text{dt}} \text{conf}_1 \to_{\text{tasks}} \text{conf}'_1 \to_{\text{dt}} \to \text{conf}_2 \cdots$ 中的配置划分为正常运行模式的一部分和恢复期模式的一部分。具体而言，我们首先将状态划分为正常运行模式；随后，每当 $\text{conf}_k \vDash \text{SP}_{\text{safe}}$时，就将该配置划分为恢复模式，直到到达满足 $l > k$ 的 $\text{conf}_l \vDash \text{SP}_{\text{safer}}$。由于韧性性质，可以保证这样的 conf_l 存在。此时，我们再次将配置划分为正常运行。conf_k 到 conf_l 之间的片段定义了一个恢复周期。其余片段为正常运行期。

现在，证明通过对恢复周期和正常运行期的数量进行归纳法来完成。正常运行期内的所有配置都满足性质 SP，其中 $\text{SP}_{\text{safe}} \prec \text{SP}$，因此不会处于不安全状态。此外，根据 dt 充分性，任何分解也都不会满足不安全状态。在恢复周期内，根据单周期恢复韧性的定义，任何分解都不会满足不安全状态。（事实上，没有任何分解满足不安全状态。）此外，由于 dt 不会跳过 SP_{safe}，因此不可能不激活恢复周期。

对于另一个方向，假设系统具有韧性。这尤其意味着，所有以满足 $\text{conf} \vDash \text{SP}_{\text{safe}}$ 的配置 conf 开始的轨迹，都将在 t 个时间单位内恢复，并且没有任何分解会经过满足不安全状态的配置。也就是说，轨迹的第一部分已经是一个单周期恢复。这意味着系统也具有单周期恢复韧性。

D 软体智能体中的人行横道规范

遵循 [23] 中介绍的自动生成信息物理系统安全性证明的方法，我们已经实现了为逻辑场景编写规范所需的机制，例如上述人行横道。规范使用 Maude [10]，编写，Maude 实现了重写逻辑 [22]。虽然熟悉 Maude 会有所帮助，但我们认为，即使不了解 Maude，也能够理解这些规范。

图 5 展示了软体智能体（SA）的整体架构。它采用传统的感知-理解-决策-行动架构。具体而言，SA 具有本地知识库（lkb），用于描述SA对环境的理解。lkb会根据读取传感器所提供的信息进行更新。传感器可能会提供错误信息，即与环境不符的信息。基于lkb，SA 会根据不同的考量对其动作集合进行排序，例如加速或减速。对于安全控制器而言，安全性是首要考量。最后，SA会执行排名最高的动作。

整个周期对应一个逻辑时间刻度。一个逻辑时间刻度对应一个实数。对于一个RTA实例，该时间刻度对应采样间隔dt，因为RTA应在dt个时间单位内完成感知-理解-决策-执行。

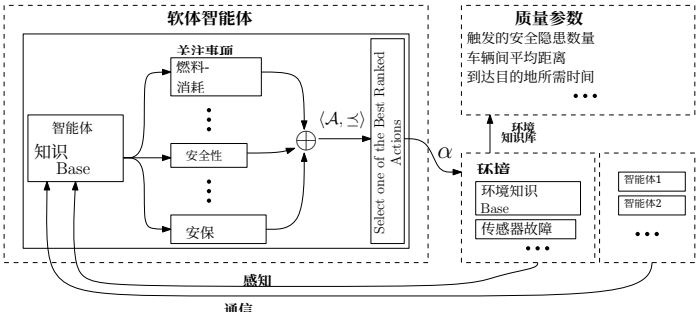

图5. 软智能体（SA）架构。

D.1 符号知识规范

我们先前工作的主要创新之处 [23] 在于，我们不再像以往那样使用具体值来表示诸如位置和速度之类的量， [12], 而是现在使用排序为`RealSym`的受约束符号来表示这些值。`RealSym` 的元素要么是实数常量，要么是形如`vv(i)` 或`vv(i,str)` 的符号，其中`i` 是一个`Nat` ，用于唯一标识一个符号，而`str` 是描述该符号直观含义的字符串，用于提高可读性。

以下符号表示 vh 的初始条件，即其位置、速度、最大加速度、最大减速度和初始加速度。

```
eq vposx  =vv(2,"vh-posX") . eq vposy  =vv(3,"vh-posY") . eq vvel  =vv(5,"vh-speed") . eq
maxacc  =vv(9,"vh-maxAcc") . eq maxdec  =vv(10,"vh-maxDec") . eq acc  =vv(32,"vh-acc") .
```

这些符号可能表示的值由一组（非线性）算术约束指定。

示例*3.* 以下约束指定了行人穿越 ODD 中定义的车辆初始速度和加速度的取值：

baseCond ={`2 <=`vvel 和 vvel `<= 10`以及 ACC `>=` maxdec 和
ACC `<=` maxacc，maxacc `== 2` 以及 maxdec `==` -8}

这些约束指定，例如，车辆速度可以取 2m/s 到 10m/s 之间的任意值，与行人过街逻辑场景中的指定完全一致。

车辆 vh 在逻辑时钟周期 0 的初始知识库包含以下项，用于指定其初始位置、速度、加速度和方向：

```
lkbI = (atloc(vh,loc(vposx,vposy)) @ 0) (speed(vh,vvel) @ 0)
       (accel(vh,acc1) @ 0) (dir(vh,loc(0,0),loc(0,1),1) @ 0)
```

车辆的方向是由两个点和一个大小指定的向量。初始时，车辆沿 Y 轴行驶。此外，vh 尚未检测到行人。

SA 通过传感器观察环境来更新其局部知识库。例如，vh 具有以下传感器集合 sset：

eq sset = (locS 速度S pedS(pedSDist,errPedS)) .

其中，locS 是位置传感器，即用于确定 vh 的位置；speedS 用于确定其速度；pedS 用于检测行人。pedS 有两个参数。第一个参数 pedSDist 指定可检测到行人的距离范围。第二个参数 errPedS 指定行人位置的最大相对误差。这些参数抽象了摄像头分辨率等可能导致对象检测误差或无法检测到对象的问题。

在逻辑时间 t 的执行过程中，传感器用于更新 SA 的局部知识库中的信息。例如，如果行人 p1 位于感知范围内，pedS 将包含如下形式的知识项：

```
(ped(p1,loc(px,py),spd1,loc(px1,py1),loc(px2,py2) @ t)
```

其中，loc(px,py) 是感知到的 p1 位置，该位置可能存在误差；spd1 是其速度；而 loc(px1,py1),loc(px2,py2) 这一对位置指定了其方向。由于行人传感器可能存在误差，如参数 errorPedS 所规定，感知到的行人位置 loc(px,py) 可能与行人的实际位置不对应。这也通过约束以符号方式进行了规定。

示例*4*。 下面的约束称为 ErrSensor，用于规定感知到的行人 X 轴位置，其中 loc(vx,vy) 和 loc(px0,py0) 分别是车辆和行人的实际位置：

(vv(i,"errY") <= (vy - py0) * errPedS) 且 (vv(i,"errY") >= 0)
py <= py0 + vv(i,"errY") 且 py >= py0 - vv(i,"errY")

符号 vv(i,"errY") 是一个新符号，用于指定 Y 轴上的最大误差。类似的约束可用于指定 X 轴上的误差。但是，由于在该逻辑场景中，车辆沿 Y 轴移动，因此不需要指定 X 轴误差。

D.2 智能体、环境、系统配置及人行横道的可执行语义

代理配置的形式为 [id: 类 | 属性]，其中 id 是智能体的唯一标识符，类是其类别，例如车辆（veh）；属性是其属性，包括以 lkb: kb 形式书写的局部知识库，其中 lkb 是标签，kb 是局部知识库内容；还包括传感器 sensors: sset，其中 sset 是传感器集合，包括用于感知行人的 pedS(pedSDist,errPedS)。

环境配置的形式为 [eId | ekb]，其中 ekb 是用于规定世界状态的环境知识库。环境知识库包含知识项 constraints(i,cond)，其中 i 是新变量的当前索引，cond 是对现有符号的（累积）约束。

系统配置由智能体配置和环境配置构成。例如，指定行人过街逻辑场景的配置如下：

asysI = { [eid | (kb constraint(i,condI)) ∈vh : veh |
lkb : kb0 ; 传感器 : sset ∈p1 : ped | lkb : kb1] }

其中包含车辆 vh 和行人 p1 的智能体配置。

（初始）约束集合 condI 指定了逻辑场景的实例。以下是一些如何使用约束来指定逻辑场景假设的示例。

示例 *5*。 以下初始约束集合 condI 指定了一个恰好包含一个实例的逻辑场景。

vposx == 5且 vposy == 0 且 vvel == 5 且 acc == 1st == 30 且
fn == 45 且 pvel == 1

其中，车辆的初始位置为 (0,5)，速度为 $5m/s$，加速度为$1m/s^2$；行人以恒定速度 $1m/s$从道路边缘的位置 30 行走到道路另一侧的位置 45。

更有趣的是，当逻辑场景表示不可数个实baseCond例时。以下两个示例也假设3。

示例 *6*。 与 baseCond 一起，以下约束

vposy >= 0以及 vposy <cr1 和 st == fn 以及 st >=cr1 和 sn <=cr2

规定车辆正朝向人行横道，并且从 st == fn 起，行人沿 X 轴在人行横道内移动。

移除约束 st == fn 并添加 fn >= cr1 和 fn <= cr2 后，所得约束规定了这样的实例：行人沿直线穿过人行横道，从道路一侧横道上的任意点到达道路另一侧横道上的任意点。

示例 *7*。 最后，我们还可以指定具有不同行人检测传感器误差假设的场景。例如，如果添加约束 errPedS== 0，则假设使用完美行人传感器。另一方面，约束 errPedS <= 0.5 指定了相对于行人的相对距离最大为 50% 的误差（如上文所述）。

可执行语义 该框架的一个关键特征是，这类符号系统配置是可执行的。语义通过两个步骤进行规定，并以系统配置上的重写规则形式化。

– 在第一阶段，智能体使用其传感器获取的信息更新本地知识库，并决定采取哪些动作，例如加速或减速。– 随后，第二阶段执行智能体决定的动作。这意味着根据预期物理规律，为智能体的位置、速度和其他物理属性生成新的符号，并为这些新符号添加新的约束。

例如，考虑vh的状态，如示例 3中由受约束符号所规定的状态。此外，假设vh 决定保持恒定速度，即加速度为零且方向不变。再假设 dt 表示与一个逻辑时钟周期对应的时间。时间滴答规则将产生以下新的知识库：

(atloc(vh,loc(vxnu,vynu)) @ 1) (速度(vh,velnu) @ 1)
(accel(vh,accnu) @ 1)

并添加以下约束，用于指定 vh 在逻辑时间 1 的可能状态：

vynu == vposy + (vvel + velnu) *dt / 2，且 velnu == vvel + accnu * dt，且 vxnu == vposx，且 accnu == 0

请注意，新配置同样是符号配置，即对应于数量可能不可数的具体配置。