基于流的去中心化运行时验证

Ritam Ganguly密歇根州立大学计算机科学与工程系，美国东兰辛
gangulyr@msu.edu

Borzoo Bonakdarpour密歇根州立大学计算机科学与工程系，美国东兰辛
borzoo@msu.edu

arXiv:2301.13266v1 [cs.DC] 30 Jan 2023

摘要——工业控制系统（ICS）通常由地理位置分散的组件构成，并常使用可编程逻辑控制器处理局部流程。由于系统规范对时间敏感，且分布式组件本身具有异步性，此类系统的验证颇具挑战。因此，开发运行时保障机制，不仅验证不同组件的正确性，还能生成系统的汇总统计信息，具有重要意义。本文首先提出一种针对分布式应用的通用运行时监控技术：其行为可建模为输入/输出流，并在部分同步语义下由内部计算模块处理，同时假设采用不完善的时钟同步算法。其次，我们提出一种通用的基于流的去中心化运行时验证技术。我们还通过大量合成实验以及若干工业控制系统和飞机SBS报文数据集，对该算法进行了严格评估。

I. 引言

工业控制系统（ICS）是一类用于控制制造、产品处理、分配等工业流程的信息系统。它包括用于控制地理位置分散资产的监控与数据采集系统，以及在每个本地化流程中使用可编程逻辑控制器的分布式控制系统。典型的可编程逻辑控制器（PLC）会接收系统各处大量传感器采集的数据。这些组件产生的数据往往会成为网络攻击和勒索软件攻击的目标，从而危及系统安全。由于这些系统与基本服务相连，针对这些设施的任何攻击都会使用户的生命安全受到威胁。这些分布式组件产生的数据完整性至关重要，因为可编程逻辑控制器的行为由这些数据决定。近期攻击案例表明，针对一家公司的工业控制系统发起攻击会使该公司蒙受约 500 万美元的损失，并导致系统停运约 50 天。另据近期报告 [1], 受影响的公司需要约 191 天才能完全恢复，而且约 54% 的组织容易遭受此类攻击。

本文倡导采用运行时验证（RV）方法，根据形式化规范监控分布式系统的行为。将 RV 应用于工业控制系统（ICS）的多个组件，可以视为分布式运行时验证的一般问题；在这一问题中，集中式监控器或去中心化监控器观察分布式系统的行为，而其中的进程不共享全局时钟。尽管运行时验证处理的是有限执行，但缺乏共同的全局时钟使其无法在分布式环境中建立事件全序。

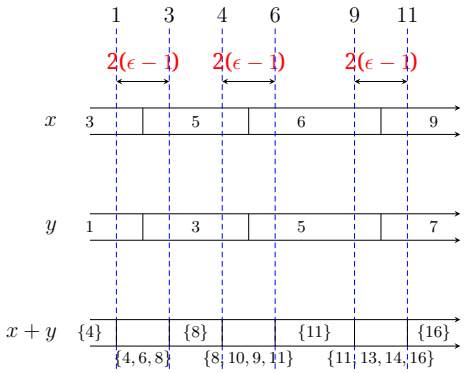


图 1：部分同步 LOLA

换言之，监视器只能形成事件的偏序，而这可能产生不同的求值结果。在运行时间枚举系统的所有可能交错会导致指数级增长，使这种方法难以扩展。更复杂的是，可编程逻辑控制器（PLC）通常需要对来自多个源的数据进行时间敏感的聚合。

我们提出了一种有效、可靠且完备的分布式运行时验证解决方案，适用于广受欢迎的基于流的规范语言 Lola [2]。与其他时序逻辑相比，Lola既能描述正确性/失效断言，也能描述可用于系统剖析和覆盖率分析的统计度量。为展示 Lola 的一个高层次示例，考虑两个输入流 x 和 y 以及一个输出流求和，如图1所示。流 x 的值为3，直到时间实例2，此时其值变为5，依此类推。

```
输入 x: 整数输入 y: 整
数输出 sum: = x+y
```

我们考虑一个具备容错能力的去中心化监视器集合，其中每个监视器只能获得系统的局部视图，且无法访问全局时钟。为了限制因缺少全局时钟而导致的状态数量膨胀，我们对存在有界时钟偏差作出一个实际假设 ϵ：所有本地时钟之间的偏差由时钟同步算法（如 NTP [3]）保证。这种设置已知是部分同步的。如图 1所示，任何相差小于 $\epsilon = 2$ 时间的两个事件都被视为并发事件，因此每个事件发生时间的不确定性被限制在其两侧各 $\epsilon - 1$ 的范围内。

在对输出流进行求和求值时，我们需要考虑各个值所有可能的发生时间。例如，在时间 1 对求和结果进行求值时，需要将 x （分别为 y）的值视为 3 和 5（分别为 1 和 3），从而得到 4、6 和 8。对所有时间实例进行求值时也可以观察到同样的情况。

本文的第一项贡献是为 LOLA 引入部分同步语义。换言之，我们定义了 LOLA，使其在评估流表达式时考虑 ϵ 时钟偏差。其次，我们引入了一种基于 SMT 的关联方程重写技术，用于部分可观测分布式系统；该技术考虑监视器观测到的值，并重写关联方程。监视器能够彼此通信，并能够将部分求值的方程解析为完全求值的方程。

我们已经证明了所提方法的正确性，以及消息复杂度的上界和下界。此外，我们已完整实现该技术，并报告了严格合成实验的结果，以及对多个工业控制系统的监控正确性和聚合结果。正如 [4], 所指出的，对工业控制系统组件发起的大多数攻击都试图篡改报告给可编程逻辑控制器的值，以使可编程逻辑控制器发生错误行为。通过我们的方法，即使不同组件之间存在时钟异步性，我们仍能够以确定性保证检测到这些攻击。我们还论证了，我们的方法能够评估系统行为聚合结果，从而便于人工运算符研究这些系统。与机器学习方法（例如 [5]–[7]）不同，我们的方法永远不会产生漏报。我们对监控技术进行了测试，研究不同参数对运行时间以及从一个监视器发送到另一个监视器的消息大小的影响，并分别报告了各项结果。

章节安排：第 II 节介绍背景概念。第 III 节介绍部分同步的 LOLA和形式化问题陈述。我们的运行时验证技术在第 IV – VII 节中进行了全面介绍，随后在第 VIII节给出实验结果。第 IX 节讨论相关工作，最后在第 X节作总结。LOLA的语法细节、正确性证明以及有关工业控制系统案例研究的更多细节见附录XI。

II. 预备知识 – 基于流的规范语言（LOLA） [2]

LOLA [2] 规范描述了给定一组输入流时输出流的计算。类型为 T 的流 α 是由值构成的有限序列，$t \in \mathsf{T}$。令 $\alpha(i)$，其中$i \geq 0$，表示该流在时间戳 i处的值。我们用 T^*（分别用 T^ω）表示有限长度（分别为无限长度）的流。

定义 1：LOLA 规范是一组方程

其形式为针对带类型的流变量：

$$\begin{aligned} s_1 &= e_1(t_1, \cdots, t_m, s_1, \cdots, s_n) \\ &\vdots \quad \vdots \\ s_n &= e_n(t_1, \cdots, t_m, s_1, \cdots, s_n) \end{aligned}$$

其中，s_1、s_2、$\cdots$、s_n 称为依赖变量，t_1、t_2、$\cdots$、t_m 称为独立变量，而 e_1、e_2、$\cdots$、e_n 是关于s_1、$\cdots$、s_n、t_1、$\cdots$、t_m的流表达式。 ■

通常，输入流称为独立变量，而输出流称为依赖变量。例如，考虑以下 LOLA 规范，其中 t_1 和 t_2 是布尔类型的独立流变量，t_3 是整数类型的独立流变量。

$$\begin{aligned} s_1 &= \texttt{true} \\ s_2 &= t_1 \vee (t_3 \leq 1) \\ s_3 &= \texttt{ite}(s_3, s_4, s_4 + 1) \\ s_4 &= s_9[-1, 0] + (t_3 \mod 2) \end{aligned}$$

其中，ite 是如果-那么-否则 的缩写，而流表达式 s_7 和 s_8 分别指具有偏移量 t_1 和+1 的流 -1。由于篇幅所限，我们在附录LOLA 中给出完整的语法XI-A。

LOLA 规范的语义通过求值模型来定义，该模型描述输入流与输出流之间的关系。

定义 2：给定一个定义在独立变量 φ、t_1、$\cdots$、t_m 上的 LOLA 规范，其类型为 T_1、$\cdots$、T_m，以及类型为 T_{m+1}、$\cdots$、T_{m+n} 的依赖变量 s_1、$\cdots$、s_n，设 τ_1、$\cdots$、τ_m 为长度为 $N+1$ 的流，其中 τ_i 的类型为 T_i。若对于 φ 中的每个方程，长度为 $N+1$ 的流元组 $\langle \alpha_1, \cdots, \alpha_n \rangle$ 都满足

$$s_i = e_i(t_1, \cdots, t_m, s_1, \cdots, s_n)$$

$\langle \alpha_1, \cdots, \alpha_n \rangle$ 满足以下关联方程：

$$\alpha_i(j) = val(e_i)(j) \qquad \text{for } (1 \leq i \leq n) \wedge (0 \leq j \leq N)$$

其中 val $(e_i)(j)$ 的定义如下。对于基础情形：

$$\begin{aligned} val(c)(j) &= c \\ val(t_i)(j) &= \tau_i(j) \\ val(s_i)(j) &= \alpha_i(j) \end{aligned}$$

对于归纳情形，当 f 是一个函数（例如算术函数）时：

$$val\Big(f(e_1, \cdots, e_k)\Big)(j) = f\Big(val(e_1)(j), \cdots, val(e_k)(j)\Big)$$

$$val\Big(\texttt{ite}(b, e_1, e_2)\Big)(j) = \text{if } val(b)(j) \text{ then } val(e_1)(j) \text{ else } val(e_2)(j)$$

$$val(e[k, c])(j) = \begin{cases} val(e)(j+k) & \textit{if } 0 \leq j + k \leq N \\ c & \textit{otherwise} \end{cases}$$ ■

与 φ 相关联的所有方程的集合记为 φ_α。

定义 3：LOLA 规范 φ 的依赖图是一个加权有向图 $G = \langle V, E\rangle$，其顶点集为 $V = \{s_1, \cdots, s_n, t_1, \cdots, t_m\}$。若 φ_α 中关于 $\alpha_i(j)$ 的方程包含 $\alpha_k(j+w)$（或 $\tau_k(j+w)$）作为子表达式，则带有权重 w 的边 e：$\langle s_i, s_k, w\rangle$（或 e：$\langle s_i, t_k, w\rangle$）属于 E。直观地说，一条边表示 s_i 在某个特定位置上的值取决于 s_k（或 t_k）的值，偏移 w 个位置。

给定一组类型分别为 $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$T $\mathbb{T} = \{$ T2, $\cdots$, T$_m\}$ 的同步输入流，以及一个 LOLA 规范 φ，我们对该 LOLA 规范进行求值，定义如下：

$$(\alpha_1, \alpha_2, \cdots, \alpha_m) \models_S \varphi$$

根据上述语义，其中 $\models_S$ 表示同步求值。

III. 部分同步 LOLA

本节将扩展 LOLA 的语义，使其能够用于对分布式系统进行推理。

A. 分布式流

这里，我们将全局时钟视为“真实”的计时器。需要注意的是，这个全局时钟的存在仅出于理论上的考虑，任何单独的流都无法访问它。

我们假设由 n 个流组成的部分同步系统，记为 $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$。对于每个流 α_i，其中 $i \in [1, |\mathcal{A}|]$，本地时钟可以表示为一个单调递增函数 c_i：$\mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$，其中 $c_i(\mathcal{G})$ 是全局时间 $\mathcal{G}$ 时的本地时钟值。由于我们处理的是离散时间系统，为简洁起见且不失一般性，我们用非负整数 $\mathbb{Z}_{\geq 0}$ 表示时间。对于任意两个流 α_i 和 α_j，其中 $i \neq j$，我们假设：

$$\forall \mathcal{G} \in \mathbb{Z}_{\geq 0}. \mid c_i(\mathcal{G}) - c_j(\mathcal{G}) \mid < \epsilon,$$

其中 $\epsilon > 0$ 是最大时钟偏差。的值为常量且已知（例如，监视器已知）。这一假设可通过采用现成的时钟同步算法（如 NTP [3], ）来满足，从而确保所有流之间的时钟偏差有界。流 在时间 σ 的局部状态为$\alpha_i(\sigma)$，其中 $\sigma = c_i(\mathcal{G})$，即某个全局时间 $\mathcal{G}$时该事件发生的本地时间。

定义 4：由 $N+1$ 个长度为 $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ 的分布式流组成的分布式流，表示为有序对 $(\mathcal{E}, \rightsquigarrow)$，其中 $\mathcal{E}$ 是所有局部状态的集合（即，$\mathcal{E} = \cup_{i\in[1,n],\ j\in[0,N]} \alpha_i(j)$），这些状态按照 Lamport 的先发生关系（ ）构成偏序 [8], ，并满足部分同步假设：

对于每个流 α_i，$1 \leq i \leq |\mathcal{A}|$，其上发生的所有事件都构成全序，即，

$$\forall i, j, k \in \mathbb{Z}_{\geq 0} : (j < k) \to (\alpha_i(j) \rightsquigarrow \alpha_i(k))$$

- 对于任意两个流 α_i 和 α_j 以及对应的两个事件 $\alpha_i(k)$ $\alpha_j(l)$ $\in \mathcal{E}$，若 $k+\epsilon < l$，则 $\alpha_i(k)$ $\alpha_j(l)$，其中 ϵ 是最大时钟偏差。
- 对于事件 e、f和 g，若 $e \rightsquigarrow f$ 且 $f \rightsquigarrow g$，则$e \rightsquigarrow g$。

■

定义 5：给定一个分布式流 $(\mathcal{E}, \rightsquigarrow)$，若事件子集 $\mathcal{C} \subseteq \mathcal{E}$ 满足：当 $\mathcal{C}$ 包含事件 e时，也包含所有在 e之前发生的事件，则称其构成一致割。当且仅当满足这一条件时，形式化地，

$$\forall e, f \in \mathcal{E}.(e \in \mathcal{C}) \wedge (f \rightsquigarrow e) \to f \in \mathcal{C}. \blacksquare$$

一致割 $\mathcal{C}$ 的前沿记为 front($\mathcal{C}$)，它是该割中每个流上最后发生的所有事件的集合。也就是说，front($\mathcal{C}$) 是每个 $i \in [1, |\mathcal{A}|]$ 和 α_i 对应的 α_i（最后一个）的集合。我们将 α_i(last) 记为 α_i 中满足 $\forall \alpha_i(\sigma) \in \mathcal{C}.(\alpha_i(\sigma) \neq \alpha_i(\text{last})) \to (\alpha_i(\sigma) \ \alpha_i(\text{last}))$ 的最后一个事件。

B. 部分同步LOLA

我们根据求值模型，定义部分同步分布式流中 LOLA 规范的语义。流变量之间缺少共同的全局时钟，而存在时钟同步，这使得输出流在任意给定的时间实例都可能有多个值。因此，我们更新求值模型，使得 $\alpha_i(j)$ 和 val(t_i)(j) 现在由集合而非单个值定义。这是由部分同步导致的非确定性所致，即有界时钟偏差 ϵ。

定义 6：给定一个 LOLA [2] 规范 φ，其包含类型为 T$_1$、$\cdots$、T$_m$ 的独立变量 t_1、$\cdots$、t_m，以及类型为 T^{m+1}、$\cdots$、T^{m+n} 的依赖变量 s_1、$\cdots$、s_n，并令 τ_1、$\cdots$、τ_m 为长度为 $N+1$、类型为 T$_i$ 的流。若长度为 $N+1$、类型相应的流元组 $\langle \alpha_1$、$\cdots$、$\alpha_n\rangle$ 满足以下条件，则称其为部分同步情形下的求值模型：对于 φ 中的每个方程：

$$s_i = e_i(t_1, \cdots, t_m, s_1, \cdots, s_n),$$

$\langle \alpha_1$、$\cdots$、$\alpha_n\rangle$ 满足以下关联方程：

$$\alpha_i(j) = \left\{val(e_i)(k) \mid \max\{0, j-\epsilon+1\} \leq k \leq \min\{N, j+\epsilon-1\}\right\}$$

其中，val (e_i)(j)定义如下。对于基础情形：

$$val(c)(j) = \{c\}$$
$$val(t_i)(j) = \left\{\tau_i(k) \mid \max\{0, j-\epsilon+1\} \leq k \leq \min\{N, j+\epsilon-1\}\right\}$$
$$val(s_i)(j) = \alpha_i(j)$$

对于归纳情形：

$$val\Big(f(e_1, \cdots, e_p)\Big)(j) = \Big\{f(e_1', \cdots, e_p') \mid e_1' \in val(e_1)(j), \cdots, e_p' \in val(e_p)(j)\Big\}$$

$$val(\texttt{ite}(b, e_1, e_2))(j) = \begin{cases} val(e_1)(j) & \texttt{true} \in val(b)(j) \\ val(e_2)(j) & \texttt{false} \in val(b)(j) \end{cases}$$

$$val(e[k, c])(j) = \begin{cases} val(e)(j+k) & \textit{if } 0 \leq j+k \leq N \\ c & \textit{otherwise} \end{cases}$$

■

示例 1：考虑以下关于相互独立的布尔值变量读和写的 LOLA 规范： φ

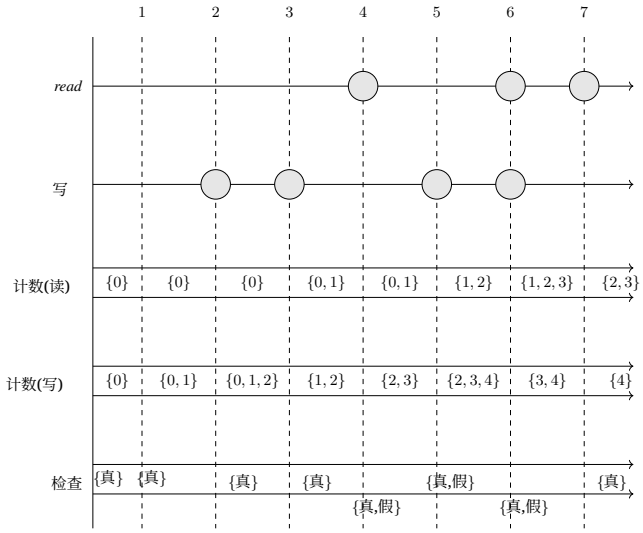


图 2：部分同步 LOLA 示例

输入 读：布尔值 输入 写：布尔值 输出 读取计数：= ite(读, 读取计数[-1,0]+ 1，读取计数[-1,0]) 输出 写入计数：= ite(写，写入计数[-1，0] + 1，写入计数[-1,0]) 输出 检查：= (写入计数- 读取计数) <= 2

在图 2中，我们有两个输入流 read 和 write，它们表示相应事件发生的时间实例。可以设想，read 和 write 是布尔值类型的流，分别在时间实例 4、6、7 和 2、3、5、6 处取值为 true，而在所有其他时间实例处取值为 false。考虑时间同步常数 $\epsilon = 2$，对上述 LOLA 规范进行求值。相应的关联方程 φ_α 为：

$$countRead(j) = \begin{cases} \texttt{ite}(read, 1, 0) & j = 0 \\ \texttt{ite}\Big(read, countRead(j-1) + 1, countRead(j)\Big) & j \in [1, N) \end{cases}$$

$$countWrite(j) = \begin{cases} \texttt{ite}(write, 1, 0) & j = 0 \\ \texttt{ite}\Big(write, countWrite(j-1) + 1, countWrite(j)\Big) & j \in [1, N) \end{cases}$$

$$check(j) = \Big(countWrite(j) - countRead(j)\Big) \leq 2$$

与同步情形类似，对部分同步 LOLA 规范进行求值需要创建依赖图。

定义 7：LOLA 规范 φ 的依赖图是一个加权有向多重图 $G = \langle V, E\rangle$，其顶点集为 $V = \{s_1, \cdots, s_n, t_1, \cdots, t_m\}$。若 $\alpha_i(j)$ 的方程包含 $\alpha_k(j+p)$（分别为 $\tau_k(j+p)$）作为子表达式，则带有权重 $w = \{\omega \mid p - \epsilon < \omega < p + \epsilon\}$ 的边 e：$\langle s_i, s_k, w\rangle$（分别为 e：$\langle s_i, t_k, w\rangle$）属于 E，其中某个 j 的偏移量为 p。■

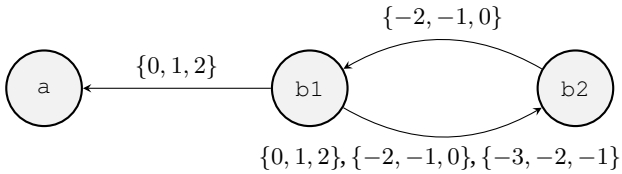


图 3：依赖图示例

直观而言，依赖图记录了在特定位置对 s_i 求值取决于 s_k（分别为 t_k）的值，偏移量为 w。需要注意的是，一对顶点 (s_i, s_k)（分别为 (s_i, t_k)）之间可能存在多条边。标记为 t_i 的顶点没有任何出边。

示例 2：考虑定义在独立整数变量 a 上的 LOLA 规范：

```
input a : uint
output b1 := b2[1, 0] + ite(b2[-1,7] <= a[1,
    0], b2[-2,0], 6)
output b2 := b1[-1,8]
```

其依赖图如图 3 所示 $\epsilon = 2$，其中有一条从 b1 指向 a 、权重为 $\{0, 1, 2\}$的边。同样，从 b1 指向 b2 有三条边，权重分别为 $\{0, 1, 2\}$、$\{-2, -1, 0\}$ 和$\{-3, -2, -1\}$；此外，还有一条从 b2 指向 b1 、权重为$\{-2, -1, 0\}$

给定一组部分同步输入流 $\{\alpha_1, \alpha_2, \cdots, \alpha_{|\mathcal{A}|}\}$，其类型分别为 $\mathbb{T} = \{\mathrm{T}_1, \mathrm{T}_2 \cdots、\mathrm{T}_{|\mathcal{A}|}\}$，以及一个 LOLA 规范，$\varphi$，$\varphi$ 的求值定义如下：

$$(\alpha_1, \alpha_2, \cdots, \alpha_{|\mathcal{A}|}) \models_{PS} \varphi$$

其中，$\models_{PS}$ 表示部分同步求值。

IV. Decentralized Monitoring Architecture

A. 总体概述

我们考虑一个去中心化在线监控系统，由固定数量的 $|\mathcal{M}|$ 可靠监控进程$\mathcal{M} = \{M_1, M_2, \cdots, M_{|\mathcal{M}|}\}$ 组成。它们可以通过完整的点对点双向通信链路收发消息并相互通信。我们还假设每条通信链路都是可靠的，即消息不会丢失或被篡改。与被观测的分布式系统类似，我们假设各个监视器的时钟是异步的，时钟同步常量为 $= \epsilon_M$。

本节中，我们假设由完整观测组成的全局分布式流 $|\mathcal{A}|$仅对各个监视器部分可见。每个监控进程在本地执行相同的顺序算法，该算法包含以下步骤（我们将在第VII节中对这种方法进行推广）。换言之，每个监视器的一次评估迭代包含以下步骤：

1) 读取一部分 $\mathcal{E}$ 事件（对 M_i可见）以及这些事件对应的时间和赋值，从而构建一个部分分布式流；

算法 1 监视器 M_i 的行为，针对 $i \in [1, |\mathcal{M}|]$

1: 对于 $j = 0$ 到 N，执行
2: 设 $(\mathcal{E}_i, \rightsquigarrow_i)_j$ 为 M_i 的部分分布式流视图
3: $LS_j \leftarrow \left[\!\left[(\mathcal{E}, \rightsquigarrow) \models_{PS} \varphi_\alpha \right]\!\right]$
4: 发送：广播符号视图 LS_j
5: 接收：$\Pi_j \leftarrow \{LS_j^k \mid 1 \le k \le \mathcal{M}\}$
6: 计算：$LS_{j+1} \leftarrow LC(\Pi_j)$
7: 结束循环

2) 每个监视器根据部分分布式流对 LOLA 规范 φ 求值；3) 每个监视器都会广播一条消息，其中包含经过改写的 φ关联方程，记为 LS；4) 收到包含关联方程的消息后，每个监视器都会汇总所有监视器的观测，组成一组关联方程。经过一次评估迭代后，每个监视器都将得到相同的一组关联方程，用于对接下来的分布式流进行求值。

在监视器的一次评估迭代期间，假定监视器 M_i 在时间 π 向另一个监视器 M_j发送的消息（对于所有 i、$j \in [1, |\mathcal{M}|]$）最迟会在时间$\pi + \epsilon_M$到达。因此，可以调整评估迭代的长度 k，以确保来自所有其他监视器的消息在下一次评估迭代开始前到达。

B. 详细说明

现在我们详细说明计算模型（参见算法 1）。每个监控进程 $M_i \in \mathcal{M}$，其中 $i \in [1, |\mathcal{M}|]$，都尝试读入 $e \in \mathcal{E}$，给定分布式流 $(\mathcal{E}, \rightsquigarrow)$。事件可以是可观察的，也可以是不可观察的。由于分配，这会产生一个部分分布式流 $(\mathcal{E}_i, \rightsquigarrow)$，其定义如下。

定义8：设 $(\mathcal{E}, \rightsquigarrow)$ 为一个分布式流。若 $(\mathcal{E}', \rightsquigarrow)$ 是 $(\mathcal{E}, \rightsquigarrow)$ 的部分分布式流，则称之为 $(\mathcal{E}', \rightsquigarrow) \sqsubseteq (\mathcal{E}, \rightsquigarrow)$，当且仅当 $\mathcal{E}' \subseteq \mathcal{E}$（先发生关系显然得以保留）。 ■

现在，我们将部分分布式流与一组去中心化监视器联系起来，并考虑去中心化监视器只能观测到分布式流的一部分这一事实。首先，所有未观测到的事件都替换为 $\natural$，即，对于所有 $\alpha_i(\sigma) \in \mathcal{E}$，若 $\alpha_i(\sigma) \notin \mathcal{E}_i$，则 $\mathcal{E}_i = \mathcal{E}_i \cup \{\alpha_i(\sigma) = \natural\}$。

定义9：设 $(\mathcal{E}, \rightsquigarrow)$ 为一个分布式流，且 $\mathcal{M} = \{M_1, M_2, \cdots, M_{|\mathcal{M}|}\}$为一组监视器，其中每个监视器 M_i（对于 $i \in [1, |\mathcal{M}|]$）都与一个部分分布式流 $(\mathcal{E}_i, \rightsquigarrow) \sqsubseteq (\mathcal{E}, \rightsquigarrow)$ 相关联。若这些监视器观测一致，则

- $\forall e \in \mathcal{E}. \exists i \in [1, |\mathcal{M}|]. e \in \mathcal{E}_i$, and
- $\forall e \in \mathcal{E}_i. \forall e' \in \mathcal{E}_j. (e = e' \wedge e \neq \natural) \oplus \Big((e = \natural \vee e' = \natural)\Big)$,

其中 $\oplus$ 表示异或运算符。

在部分同步系统中，事件存在不同的排序方式，而每种独特的事件排序都可能计算出不同的值。给定一个分布式流（$\mathcal{E}$，$\rightsquigarrow$），一致割序列的形式为 $\mathcal{C}_0\mathcal{C}_1\mathcal{C}_2\cdots\mathcal{C}_N$，其中对于所有 $i \ge 0$：(1) $\mathcal{C}_i \subseteq \mathcal{E}$，以及(2) $\mathcal{C}_i \subseteq \mathcal{C}_{i+1}$。

根据部分同步 LOLA 的语义，在时间实例 j 计算输出流变量 s_i 需要事件 α_i k $i \in [1, |\mathcal{A}|]$ $\{0, j - \epsilon + 1\} \le \pi \le \{N, j + \epsilon - 1\}\Big\}$ ()，其中且 $k \in \Big\{\pi \mid$最大值。为了将分布式流的监控转换为同步流，我们确保一致割的前沿中的事件 $\mathcal{C}_j$ 是 $\alpha_i(k)$。

设 $\mathbb{C}$ 表示所有有效一致割序列的集合。我们定义 $(\mathcal{E}, \rightsquigarrow)$ 的所有同步流的集合如下：

$$\mathsf{Sr}(\mathcal{E}, \rightsquigarrow) = \Big\{\mathsf{front}(\mathcal{C}_0)\mathsf{front}(\mathcal{C}_1)\cdots \mid \mathcal{C}_0\mathcal{C}_1\cdots \in \mathbb{C}\Big\}$$

直观地说，$\mathsf{Sr}(\mathcal{E}, \rightsquigarrow)$ 可以解释为所有可能“交错”的集合。对于 $(\mathcal{E}, \rightsquigarrow)$，LOLA 规范 φ 的求值如下：

$$\Big[(\mathcal{E}, \rightsquigarrow) \models_{PS} \varphi\Big] = \Big\{(\alpha_1, \cdots, \alpha_n) \models_S \varphi \mid (\alpha_1, \cdots, \alpha_n) \in \mathsf{Sr}(\mathcal{E}, \rightsquigarrow)\Big\}$$

这意味着，根据 LOLA 规范对部分同步分布式流进行求值，会得到一组求值结果，因为计算可能涉及多个流。这也使我们能够将部分同步分布式系统的求值问题，转化为对多个同步流的求值；每个同步流的输出流都求得唯一值，且消息复杂度为

$$O\big(\epsilon^{|\mathcal{A}|} N |\mathcal{M}|^2\big) \quad \Omega(N|\mathcal{M}|^2)$$

C. 问题陈述

总体问题陈述要求，在算法1终止时，去中心化监控架构中的所有监视器给出的判定结果，都与能够全局查看系统的集中式监控器给出的结果相同

$$\forall i \in [1, m] : \mathsf{Result}_i = \Big[(\mathcal{E}, \rightsquigarrow) \models_{PS} \varphi\Big]$$

其中，$(\mathcal{E}, \rightsquigarrow)$ 是全局分布式流，φ 是 LOLA 规范，Result_i 是监视器 M_i 求得的结果。

V. 计算 LS

本节介绍在给定系统的求值结果和观测的情况下，如何重写 LOLA 相关方程。在我们的分布式环境中，对 LOLA 规范进行求值，需要生成一组同步流，并在其上对给定的 LOLA 规范进行求值（详见第VI节）。在此，我们利用 LOLA 规范的求值结果构成局部观测，并与系统中的其他监视器共享。

给定同步流集合（α_1、α_2、$\cdots$、$\alpha_{|\mathcal{A}|}$），符号化局部计算结果 LS（见算法 1）由相关的 LOLA 方程组成。这些方程要么需要从其他监视器获取更多信息（数据未被观测到）才能求值，要么相关监视器需要等待（正偏移量）。无论哪种情况，相关的 LOLA 规范都会作为缺失数据与系统中的所有其他监视器共享

可由任一监视器观测到。我们根据求值表达式 e_i for all $i \in [1,n]$ 所需的独立变量值的可观测性，将重写规则分为三种情况。每个流表达式可归为三类：（1）完全未被观测到；（2）完全被观测到；或（3）部分被观测到。只需遍历依赖图，并检查相应监视器读取的部分分布式流，即可轻松完成分类。

情况 1（完全观测）。形式上，如果对于所有 s_k（分别为 t_k）$\langle s_i$，依赖图 $G = \langle V、E\rangle$ 中的流表达式 s_i 都可以被识别出来，那么 $s_k、w\rangle \in E$（分别为 $\langle s_i、t_k、w\rangle \in E$）以及 $s_k(j+w) \neq \natural$（分别为 $t_k(j+w) \neq \natural$）在时间实例 j 均已观测到。如果是这样，则表示监视器 M 已观测到计算 $s_i(j)$ 所需的所有独立变量和依赖变量，因此可通过计算 $s_i(j) = e_i(s_1,\cdots,s_n,t_1,\cdots,t_m)$，并将 $s_i(j)$ 重写为 LS。

情况 2（完全未观测）。形式上，对于所有 s_k（分别为 t_k）依赖图 $G = \langle V、E\rangle$ 中的流表达式 s_i 都完全未观测，即 $\langle s_i、s_k、w\rangle \in E$（分别为 $\langle s_i、t_k、w\rangle \in E$）以及 $s_k(j+w) = \natural$（分别为 $t_k(j+w) = \natural$）在时间实例 j 均未观测到。这表示监视器 M 不知道这些变量的赋值。因此，我们将以下流表达式重写

$$s'_k(j) = \begin{cases} s_k(j+w) & 0 \le j+w \le N \\ \texttt{default} & \text{otherwise} \end{cases}$$
$$t'_k(j) = \begin{cases} t_k(j+w) & 0 \le j+w \le N \\ \texttt{default} & \text{otherwise} \end{cases}$$

对于所有 $\langle s_i、s_k、w\rangle \in E$ 和 $\langle s_i、t_k、w\rangle \in E$，并将用于求值 $s_i(j)$ 的改写后关联方程纳入其中，形式如下

$$s_i(j) = e_i(s'_1,\cdots,s'_n,t'_1,\cdots,t'_m)$$

需要注意的是，流变量 s_k（分别为 t_k）的默认值取决于流的相应类型 T_k（分别为 T_{m+k}）。

情况 3（部分观测）。形式上，对于依赖图 $G = \langle V$ 中的部分观测流表达式 s_i，对于时间实例 j，所有 s_k（分别是 t_k）要么已观测，要么未观测。换句话说，对于所有 $\langle s_i、s_k、w\rangle \in E$，我们可以表示一个包含所有已观测流变量的集合 $\mathbb{V}_o = \{s_k \mid \exists s_k\,(j+w) \neq \natural\}$，以及一个包含所有未观测的依赖流变量的集合 $\mathbb{V}_u = \{s_k \mid s_k\,(j+w) = \natural\}$。该集合也可以扩展为包含独立变量。对于所有未观测的 $s_k \in \mathbb{V}_u$（分别是 $t_k \in \mathbb{V}_u$），将其替换为：

$$s^u_k(j) = \begin{cases} s_k(j+w) & 0 \le j+w \le N \\ \texttt{default} & \text{otherwise} \end{cases}$$
$$t^u_k(j) = \begin{cases} t_k(j+w) & 0 \le j+w \le N \\ \texttt{default} & \text{otherwise} \end{cases}$$

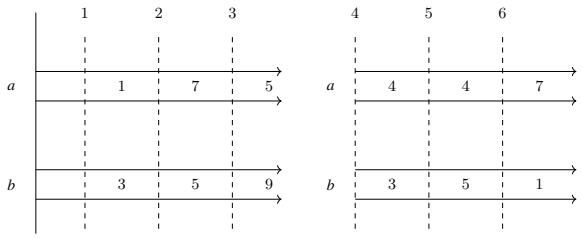


图4：生成 LS 的示例

对于所有已观测的 $s_k \in \mathbb{V}_o$（分别是 $t_k \in \mathbb{V}_o$），将其替换为：

$$s^o_k(j+w) = \texttt{value}$$
$$t^o_k(j+w) = \texttt{value}$$

并由此对 $s_i(j)$ 进行部分求值，形式如下

$$s_i(j) = e_i(s^o_1,\cdots,s^o_n,t^o_1,\cdots,t^o_m,s^u_1,\cdots,s^u_n,t^u_1,\cdots,t^u_m)$$

随后将 $s_i(j)$ 的部分求值关联方程添加到 LS 中。需要注意的是，一致的部分分布式流确保对于所有 s_k（分别为 t_k），只能是已观测或未观测，不能两者兼有，也不能两者皆非。

示例 3：考虑下述 LOLA 规范以及长度为 $N = 6$ 的流输入，该输入分为两个评估轮次；$\epsilon = 2$ 如图4 所示，监视器为 M_1 和 M_2。

```
input a : uint
input b : uint
output c := ite(a[-1,0] <= b[1, 0], a[1,0],
    b[-1, 0])
```

输出流对应的方程为：

$$c = \begin{cases} \texttt{ite}(0 \le b(i+1), a(i+1), 0) & i = 1 \\ \texttt{ite}(a(i-1) \le b(i+1), a(i+1), & \\ \qquad b(i-1)) & 2 \le i \le N-1 \\ \texttt{ite}(a(i-1) \le 0, 0, b(i-1)) & i = N \end{cases}$$

设监视器M_1读入的部分分布式流包括 $\{a,$(1,1),(3,5)$\}$，监视器 M_2 读入的部分分布式流包括$\{a,$(1, 1), (2, 7)$\}$，$\{b,$(1, 3), (3, 9)$\}$。监视器 M_1 计算c(2) $= 5$，并部分计算 c(1)和 c(3)。因此，$\mathsf{LS}^1_1 = \{c(1) = a(2)\ c(2) = 5,\ c(3) = \text{ite}(a(2) \le b(4)\ a(4), 5)\}$。监视器 M_2 部分计算所有的 c(1)、c(2)和 c(3)，因此 $\mathsf{LS}^2_1 = \{c(1) = \text{ite}(0 \le b(2)\ a(2), 0)\ c(2) = a(3)\ c(3) = \text{ite}(7 \le b(4)\ a(4)\ b(2))\}$。

监视器M_1读入的部分分布式流包含 $\{a,$(4,4),(5,4)$\}$，$\{b,$(4,3),(6,1)$\}$；监视器 M_2 读入的部分分布式流包含 $\{a,$(5,4),(6,7)$\}$，$\{b,$(4,3),(5,5)$\}$。监视器 M_1 计算c(4) $= 9$和 $c(5) = 3$，并部分计算 c(6)。因此 $\mathsf{LS}^1_2 = \{c(4) = 9, c(5) = 3, c(6) = b(5)\}$。监视器 M_2计算 c(6) $= 5$，并部分计算 c(4)和 c(5)，因此 $\mathsf{LS}^2_2 = \{c(4) = \text{ite}(a(3) \le 5, 4, 9)\ c(5) = \text{ite}(a(4) \le b(6), 7, 3)\ c(6) = 5\}$。

需要注意的是，第一轮评估后，相应的局部状态 LS^1_1 和 LS^1_2 将被共享，

从而能够对部分求值的输出流中的一部分进行求值（将在第 VII-A 节讨论）。这些状态将纳入下一评估轮次的局部状态中。

请注意，生成 LS 时考虑的是有序流，即其中事件的发生时间和值可以进行比较。可以想象，对于分布式系统，生成相同内容需要针对事件的所有可能排序分别进行生成。以下章节将对此进行详细讨论。

VI. 基于 SMT 的解决方案

A. SMT 实体

SMT 实体表示：(1) LOLA 方程；(2) 用于表示分布式流的变量。生成一致割序列后，我们使用第 V 节讨论的规律，构造所有局部计算或部分计算的 LOLA 方程集合。

分布式流。在我们的 SMT 编码中，事件集合 $\mathcal{E}$ 由一个位向量表示，其中每一位对应分布式流中的一个独立事件（$\mathcal{E}$、$\rightsquigarrow$）。观测中的流长度为 k，这使得 $|\mathcal{E}| = k \times |\mathcal{A}|$，而整个流的长度为 N。我们对分布式流进行预处理，创建一个 $\mathcal{E} \times \mathcal{E}$ 矩阵 hbSet，以纳入先发生关系。我们按如下方式填充 hbSet：当且仅当 $e \rightsquigarrow f$ 时，hbSet[e][f] = 1；否则 hbSet[e][f] = 0。为了将每个事件映射到其对应的流，我们引入一个函数 $\mu : \mathcal{E} \to \mathcal{A}$。

我们引入一个赋值函数 val : $\mathcal{E} \to$ T（LOLA 规范中所使用的类型），用于表示各个事件的值。根据系统的部分同步假设，事件可能发生的时间由函数 δ : $\mathcal{E} \to \mathbb{Z}_{\geq 0}$ 定义，其中 $\forall \alpha(\sigma) \in \mathcal{E}.\exists \sigma' \in [$最大值$\{0, \sigma - \epsilon + 1\}$，最小值$\{\sigma + \epsilon - 1\}, N].\delta\big(\alpha(\sigma)\big) = \sigma'$。引用输出流上的事件时，我们将时间同步常量更新为 ϵ_M，以此更新 δ 函数。这考虑了两个监视器之间的时钟偏差。最后，我们引入一个未解释函数ρ : $\mathbb{Z}_{\geq 0} \to 2^{\mathcal{E}}$，用于确定一致割序列，以计算 LOLA 规范的所有可能求值，同时满足第六节-B 中所述的若干给定约束。

B. 可满足性模理论约束

定义必要的 SMT 实体后，我们接下来定义 SMT 约束。我们首先定义用于生成一致割序列的 SMT 约束，随后定义用于计算给定 LOLA 方程的 SMT 约束 φ_α。

对 $\boldsymbol{\rho}$ 上的一致割施加约束：为了确保未解释函数 ρ 标识一个一致割序列，我们施加了一些约束。第一个约束确保 ρ 值域中的每个元素实际上都是一个一致割：

$$\forall i \in [0,k].\forall e, e' \in \mathcal{E}.\Big((e \rightsquigarrow e') \wedge (e' \in \rho(i))\Big) \to (e \in \rho(i))$$

接下来，我们确保每个后续一致割都包含前一个一致割中的所有事件：

$$\forall i \in [0, k-1].\rho(i) \subseteq \rho(i+1)$$

接下来，我们确保每个一致割的前端由事件构成，并且这些事件可能发生的时间符合部分同步 LOLA 的语义：

$$\forall i \in [0,k].\forall e \in \mathsf{front}(\rho(i)).\delta(e) = i$$

最后，我们确保每个一致割都包含来自所有流的事件：

$$\forall i \in [0,k].\forall \alpha \in \mathcal{A}.\exists e \in \mathsf{front}(\rho(i)).\mu(e) = \alpha$$

LOLA 规范的约束：这些约束将计算 LOLA 规范，并确保 ρ 不仅表示有效的一致割序列，还确保该一致割序列能够根据给定的流表达式计算 LOLA 方程。显然，分布式系统在每个时间实例上通常可能计算出多个值。因此，我们需要检查逻辑表达式的满足与违反情况，并计算算术表达式的所有可能值。请注意，对所有 LOLA 规范进行监控可以归结为计算逻辑表达式或算术表达式。下面，我们给出在时间实例 j 计算不同 LOLA 方程所需的 SMT 约束：

$$t_i[p,c] = \begin{cases} val(e) & 0 \leq j+p \leq N \\ c & \text{otherwise} \end{cases} \Big(\exists e \in \mathsf{front}(\rho(j+p)).(\mu(e) = \alpha_i)\Big)$$

$$s_i(j) = \texttt{true} \quad \mathsf{front}(\rho(j)) \models \varphi_\alpha$$

(Logical expression, satisfaction)

$$s_i(j) = e_i(\forall e \in \mathsf{front}(\rho(j)).val(e))$$

(Arithmetic expression, evaluation)

先前求得的结果会作为一个实体纳入 SMT 实例，并添加一个额外约束，使其每次只求得一个唯一值，从而生成所有可能的求值。当且仅当方程至少存在一个唯一求值时，SMT 实例才会返回可满足结果。此过程会反复执行，直到在给定约束的情况下无法再生成一致割序列，即无法再生成唯一值为止。需要注意的是，形式为 ite(s_i, s_k, s_j) 的流表达式可以化简为一组表达式：首先将 s_i 求值为逻辑表达式，然后据此对 s_j 和 s_k 求值。

VII. LOLA 规范的运行时验证

现在已经讨论了生成改写后的 LOLA方程V以及 SMT 编码的工作原理（第VI节），我们终于可以将二者结合起来，解决第IV节提出的问题。

算法 2 监视器上的计算 M_i

1: $LS_1^i[0] = \emptyset$
2: 对于 $r = 1$ 到 $\lceil N/k \rceil$，执行
3: 第 $r \leftarrow r$ 个一致的部分分布式流（$\mathcal{E}_i$, $\rightsquigarrow_i$）
4: $j = 0$
5: 执行
6: $j = j + 1$
7: $(\alpha_1, \alpha_2, \cdots, {}^{\alpha_{|\mathcal{A}|}}) \in \mathrm{Sr}(\mathcal{E}_i, \rightsquigarrow_i)$
8: $LS_r^i[j] \leftarrow LS_r^i[j-1] \cup [(\alpha_1, \alpha_2, \cdots, {}^{\alpha_{|\mathcal{A}|}}) \models_S \varphi_\alpha]$
9: 当（$LS_r^i[j] \neq LS_r^i[j-1]$）时
10: 发送：广播符号视图 $LS_r^i[j]$
11: 接收：$\Pi_r^i \leftarrow \{LS_r^k \mid 1 \leq k \leq \mathcal{M}\}$
12: 计算：$LS_{r+1}^i[0] \leftarrow LC(\Pi_r^i)$ ▷第 VII-A
13: 结束循环
14: 结果$^i \leftarrow \bigcup_{r \in [1, \lceil N/k \rceil + 1]} LS_r^i[0]$

A. 计算 LC

给定一组由 SMT 编码计算得出的局部状态，每个监控进程都会收到一组改写后的 LOLA 关联方程，记为 LS_j^i，其中 $i \in [1, |\mathcal{M}|]$ 表示第j个计算轮次。我们计算 LC 的思路很简单：对所有关联方程取优先并集。

$$LC(\Pi_j^i) = \biguplus_{i \in [1, |\mathcal{M}|]} LS_j^i$$

优先级背后的直觉是：已求值的 LOLA 方程优先于部分求值或未求值的 LOLA 方程，而两个部分求值的 LOLA 方程将合并为一个已求值或部分求值的 LOLA 方程。例如，取自示例 3 的局部计算结果 LS 11 和 LS_1^2，LC (LS 11, LS 21) 的计算结果为 $\{c(1) = a(2)\ c(2) = 5, c(3) = \mathrm{ite}(7 \leq b(4)\ a(4), 5)\}$ 在监视器 M_1 处，且 $\{c(1) = 7, c(2) = 5, c(3) = \mathrm{ite}(7 \leq b(4)\ a(4), 5)\}$ 在监视器 M_2处。随后，LC (LS_2^1, LS 22) 的计算结果为 $\{c(4) = 9, c(5) = 3, c(6) = 5\}$ 在监视器 M_1 处，且 $\{c(4) = 9, c(5) = 3, c(6) = 5\}$ 在监视器 M_2处。

B. 整合一切

如第 IV-A 节所述，监视器是去中心化且在线运行的。由于建立可满足性模理论实例的成本很高（正如第 VIII 节中的评估结果所示），我们通常发现，每经过 k个时间实例后再对 LOLA 规范进行求值会更加高效。这样既可将计算轮次减少到 $\lceil N/k \rceil$，也能减少网络上传输的消息数量，但会增大消息的大小。我们对算法 1 进行了更新，使其更贴近算法2，以反映我们的解决方案。

每轮评估从读取第 r 个部分分布式系统开始，该系统由发生在 $\max\{0, (r-1) \times \lceil N/k \rceil\}$ 与 $\min\{N, r \times \lceil N/k \rceil\}$（第 3 行）之间的事件组成。我们假设该部分分布式系统是一致的，符合每个事件至少已被一个监视器读取的假设。为考虑 $(r-1)$ 个计算轮次中的事件与第 r 个计算轮次中的事件之间的并发，我们将长度延长 ϵ 个时间单位，从而使第 r 个计算轮次的 $\max\{0,(r-1) \times \lceil N/k \rceil - \epsilon + 1\}$ 与 $\min\{N, r \times \lceil N/k \rceil\}$ 的长度。

接下来，我们将分布式流问题的求值转化为可满足性模理论问题（第 7 行）。我们使用 SMT 实体表示分布式系统，然后借助 SMT 约束，对生成的一致割序列求值 LOLA 规范。每个一致割序列都表示事件的一种唯一排序，该排序会使流表达式求值得出一个唯一值（第 8 行）。重复此过程，直到无法再生成能求值得到 φ_α 唯一值的一致割序列为止（第 9 行）。已求值和部分求值的结果都会作为相关的 LOLA 方程纳入 LS。随后进入通信阶段，每个监视器都会共享其本地计算出的 LS_r^i，涵盖所有 $i \in [1, |\mathcal{M}|]$ 和 r 求值轮次（第 10–11 行）。

收到所有监视器的局部状态后，我们对所有关联方程取优先并集，并将其纳入 LS_{r+1}^i set of associated equations（第 12 行）。随后，计算进入下一计算轮次，并重复上述步骤。计算结束时，所有求值结果都包含在 Result^i 中。

引理 1：设 $\mathcal{A} = \{S_1, S_2, \cdots, S_n\}$ 是一个分布式系统，且 φ 是一个 LOLA 规范。算法 1 在监控一个会终止的分布式系统时会终止。

定理 1：算法 2 解决了第 IV 节中陈述的问题。

定理 2：设 φ 是一个 LOLA 规范，且 $(\mathcal{E}, \rightsquigarrow)$ 是一个由 $|\mathcal{A}|$ 个流组成的分布式流。算法 2 在使用 $|\mathcal{M}|$ 个监视器时的消息复杂度为

$$O\left(\epsilon^{|\mathcal{A}|} N |\mathcal{M}|^2\right) \quad \Omega(N|\mathcal{M}|^2)$$

VIII. 案例研究与求值

本节分析我们基于可满足性模理论的去中心化监控方案。需要指出的是，我们不考虑数据收集、数据传输等问题，因为在给定的分布式环境中，实际 SMT 编码的运行时间将是监控过程中影响最大的因素。我们使用从合成实验中收集的轨迹（第 VIII-A 节）以及涉及若干工业控制系统和 RACE 数据集的案例研究（第 VIII-B 节）来评估所提出的方案。我们的方法实现可在 Google Drive（https://tinyurl.com/2p6ddjnr）上获取。

A. 合成实验

1) 设置：每个实验由两个阶段组成：(1) 生成分布式流；(2) 验证。对于数据生成，我们开发了一个合成程序，用于随机生成分布式流（即一组流的局部计算状态）。我们假设流的类型为浮点数、整数或布尔值。对于浮点数和整数类型的流，初始值是一个随机值 `s[0]`，后续值则通过 `s[i-1] + N(0, 2)` 生成，其中所有 $i \geq 1$ 均适用。我们还确保流的值始终为非负数。另一方面

对于布尔值类型的流，我们从 true 或 false 开始，后续值则保持不变或根据参数为B(0.8)的伯努利分布进行切换，其中 true 表示保持不变，false 表示值发生变化。

对于监视器，我们研究采用伯努利分布 B(0.2)、 B(0.5) 和 B(0.8) 作为事件读取分布的方法。可读性越高，每个事件被更多监视器读取的概率就越大。我们还确保按照所提出的方法，每个事件至少会被一个监视器读取。为了测试该方法在不同类型流表达式下的表现，我们使用以下算术表达式和逻辑表达式。

输入 a1：无符号整数 输入 a2：无符号整数 输出算术表达式：`= a1 + a2` 输出逻辑表达式：`= (a1 > 2) && (a2 < 8)`

2）结果分析：我们研究不同参数，并分析它们对我们方法的运行时间和消息大小有何影响。所有实验均在一台 2017 年款 MacBook Pro 上进行，该设备配备 3.5GHz 双核英特尔酷睿 i7 处理器和 16GB、2133 MHz LPDDR3 内存。除非另有说明，所有实验均设定流的数量为 $|\mathcal{A}| = 3$，时间同步常数为 $\epsilon_M = \epsilon = 3s$，监视器数量与流的数量相同，计算长度为 $N = 100$，其中 $k = 3$ 采用读取分布 B(0.8)。

时间同步常数。增大时间同步常数 ϵ会增加需要考虑的并发事件的可能数量。这会增加求值 LOLA 规范的复杂度，从而增加算法的运行时间。此外，较大的 ϵ 对应着需要考虑的更多可能流。正如预期，在图 ϵ 5a中可以看到，随着 k的值增大，运行时间以更快的速度增长，直到达到 $k = \epsilon$ 的阈值。这是因为需要考虑的流数量呈指数级增长，但最终会受限于计算中存在的事件数量。

增加时间同步常数的值，也会直接增加每个时间实例所求值结果的数量。这是因为，每条流都对应一个唯一值，在达到可能求值总数的上限之前，该值都会被求出，如图 6a 所示。然而，比较图 5a 和 6a，可以看出，运行时间随消息大小增长得更快。这是因为，最初，一个 SMT 实例会在所有时间实例上求出唯一值。然而，当某些时间实例的求值逐渐达到所有可能的求值时，只有一部分时间实例会产生唯一值。这就是消息大小比监视器的运行时间更快达到阈值的原因。

流表达式的类型。 流表达式可分为两大类：一类包含算术运算，另一类涉及逻辑运算。算术运算的求值结果可以是 $O(|\mathcal{A}|.\epsilon)$范围内的值，而逻辑运算的求值结果只能是真或 假。当监视器对分布式流具有较高的可读性时，通常意味着监视器能够对流表达式求值。因此，我们从图 5c 中观察到，算术表达式的求值运行时间呈指数增长，而逻辑表达式的求值运行时间则呈线性增长。然而，当计算的可读性较低时，无论表达式类型如何，两者都需要指数级时间，因为都无法完整地对流表达式求值。因此，每个监视器都必须生成所有可能的流。

类似地，对于高可读性和逻辑表达式，只要监视器能够对流表达式求值，消息大小就是常量。然而，对于低可读性，求值逻辑表达式的消息大小与其算术对应项相同。这一点可见于图 6c，这是因为低可读性下，监视器无法完整地对表达式求值，因此需要将重写表达式及观测到的值发送给其他监视器，由它们进行求值。

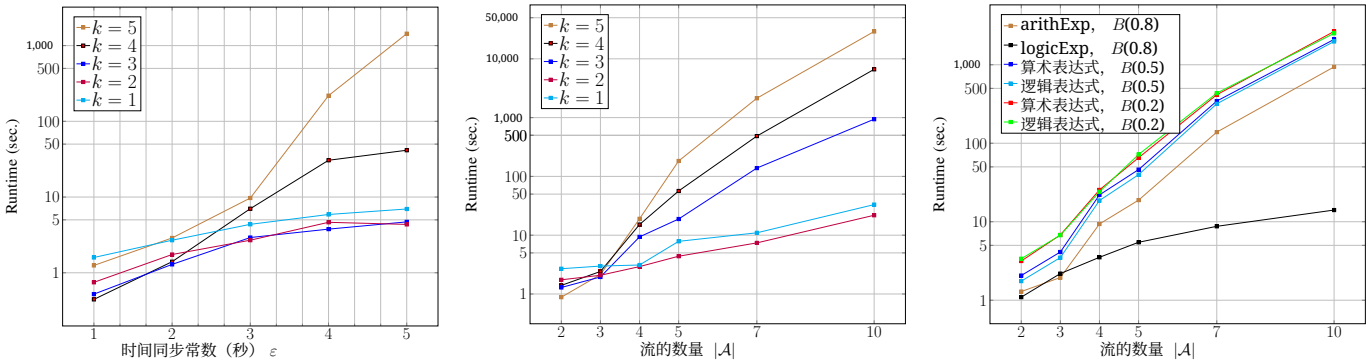
流的数量。随着流的数量增加，事件数量呈线性增长，从而导致可能的同步流数量（由于交错）呈指数增长。如图 5b 所示，分布式流中的流数量增加时，运行时间呈指数增长。同样，如图 6b 所示，流的数量线性影响 LOLA 表达式可求得的唯一值数量，进而增大消息的大小。

B. 案例研究：去中心化工业控制系统与飞行控制运行时验证

我们通过若干工业控制系统数据集检验了运行时验证方法，这些数据集包括由以下系统生成的数据：(1) 安全水处理厂 (SWaT)， [9]，该厂由六个流程构成，分别对应不同的物理组件和控制组件；(2) 一个电力分配系统，[10] 其中包含来自四个相量测量单元 (PMU) 的读数，这些单元用于测量电网上的电波；以及 (3) 一个燃气分配系统， [11]其中包含发送至可编程逻辑控制器 (PLC) 及其发出的消息。在这些工业控制系统中，我们监视系统属性的正确性。此外，我们还监视 RACE 中所有航空器对之间的相互间隔， [12]该数据集由航空器发送的 SBS 消息组成。有关各系统及其 LOLA 规约的更多详情，请参阅附录XI-C。

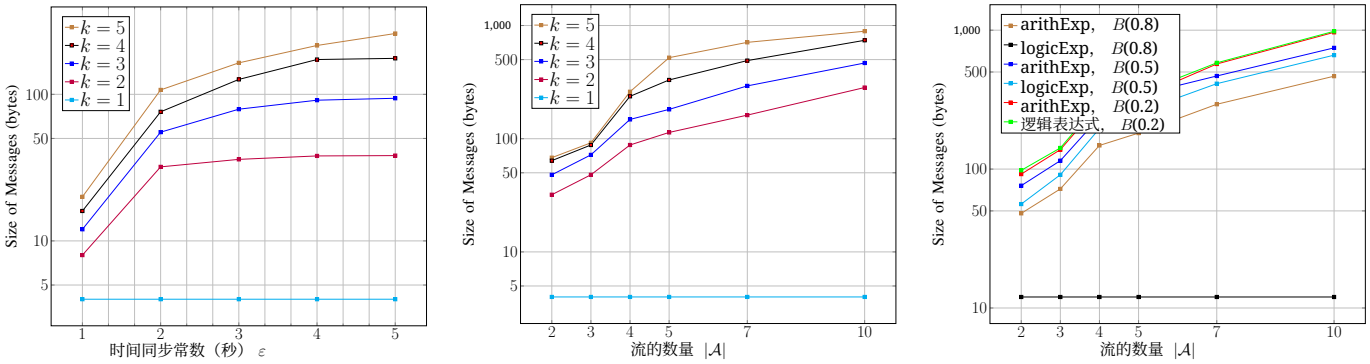
对于我们的场景，我们假设每个组件都有自己的异步本地时钟，并且时间同步常数各不相同。接下来，我们将讨论针对 LOLA 规范验证不同工业控制系统的结果。

结果分析：对于每个 ICS 案例研究，我们使用的监视器数量与组件数量相同，并将整个空域划分为 9 个不同区域，每个区域由一个监视器负责。我们观察到，我们的方法



(a) 埃普西隆　(b) 流的数量　(c) 不同的 LOLA 规范

图5：不同参数对合成数据运行时间的影响。



（a）埃普西隆　（b）流的数量　(c) 不同的 LOLA 规范

图 6：不同参数对合成数据消息大小的影响。

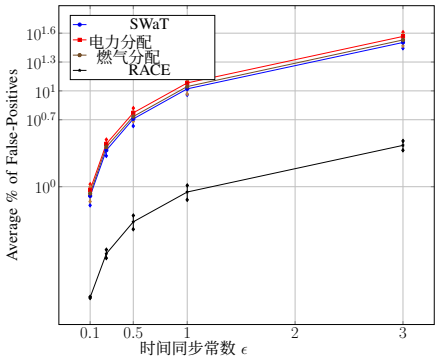


图 7：ICS 案例研究中的假阳性

当系统实际上遭到攻击时，不会报告系统属性得到满足（假阴性）。但是，由于假设组件之间部分同步，我们的方法可能会报告假阳性，也就是说，即使系统没有遭到攻击，也会报告系统属性遭到违反。如图7所示，随着时间同步常量的减小，假阳性的数量也会减少。这是因为随着 ϵ的减小，监视器会将较少的事件视为并发事件。因此，监视器观测到的事件偏序会更接近系统中实际发生的事件实际顺序。

我们在航空器监测方面取得了显著更好的结果与其他数据集相比，假阳性更少。这可归因于空中交通管制员始终让两架航空器之间保持比建议的最小间隔更大的距离。在监控其他工业控制系统的过程中，我们还想报告，我们的监控方法能够成功检测多种攻击，包括 SWaT 中储罐的下溢和溢出以及水质的突然变化，区分电力分配中断路器的人工跳闸与因短路导致的跳闸，以及燃气分配中的单点数据注入。

IX. 相关工作

针对集中式和去中心化监测设置的在线谓词检测已在 [13], [14] 中得到广泛研究。 [15], [16] 引入了对更具表达力的时态算子的扩展。[13], [15], [16] 中介绍的监控方法考虑了一个完全异步的分布式系统。 [17] 中提出了一种基于 SMT 的谓词检测解决方案。 [18]–[20] 研究了同步分布式系统的运行时验证。这种方法假设所有组件共享一个共同的全局时钟，这是其主要缺点。最后， [21] 研究了监视器可能发生崩溃的容错监控，针对异步分布式进程；[22] 则针对同步分布式进程进行了研究。

基于流的规范的运行时验证由 [2], [23], 引入，其中假设事件的发生是同步的。为了将基于流的运行时验证扩展到更复杂的系统，即事件发生具有异步性的系统，[24]–[26] 引入了一种基于实时的逻辑。然而，由于假设存在共享全局时钟，这些方法无法有效验证地理位置相距较远的大型分布式系统。相反，我们假设存在一种时钟同步算法，可将组件之间的最大时钟偏差限制在一个常量以内。这是一种现实的假设，因为大型工业系统中的不同组件都有自己的时钟，而它们之间必然存在偏差。[27], [28] 分别针对 LTL 和 MTL 规范研究了一种类似的基于 SMT 的解决方案；我们在此基础上将其扩展到包含更具表达力的基于流的规范。

X. 结论

本文研究了针对广受欢迎的基于流的规范语言 LOLA 的分布式运行时验证。我们提出了一种在线去中心化监控方法，其中每个监视器都以一组相关的 LOLA 规范和部分分布式流作为输入。通过假设所有流之间存在部分同步，并将验证问题转化为 SMT 问题，我们降低了该方法的复杂度，使其不再依赖于时间同步常数。我们还开展了大量合成实验，验证了大型工业控制系统的系统属性，并对 SBS 消息进行了空域监控。与基于机器学习的系统正确性验证方法相比，我们的方法能够在确定性保证下给出可靠且正确的结果。作为更佳实践，也可以在训练期间或作为检测系统违规时的安全保障，将我们的运行时验证方法与基于机器学习的方法结合使用。

未来，我们计划研究分布式系统的监控，其中监视器本身可能遭受故障，例如崩溃和拜占庭故障。这将使我们能够设计一种考虑故障和漏洞的技术，以模拟现实中的监控系统，从而拓展运行时验证在更多现实安全关键系统中的适用范围和应用。

参考文献

[1] W. Schwab 和 M. Poujol，“2018 年工业网络安全现状”，卡巴斯基报告趋势研究，第 33 卷，2018 年。[2] B. D’Angelo、S. Sankaranarayanan、C. Sanchez、W. Robinson、B. Finkbeiner、H. Sipma、S. Mehrotra 和 Z. Manna，“LOLA：同步系统的运行时监控”，载于第 12 届国际时间表示与推理研讨会（TIME’05），2005 年，第 166–174 页。[3] D. Mills，“网络时间协议第 4 版：协议和算法规范”，互联网征求意见文件，RFC 编辑部，RFC 5905，2010年6月。[4]T. Alladi、V. Chamola 和 S. Zeadally，“工业控制系统：网络攻击趋势与对策”，*Computer Communications*，第 155 卷，第 1–8 页，2020年。 [在线]。可访问：https://www.sciencedirect.com/science/article/pii/S0140366419319991

[5] S. Pan、T. Morris和U. Adhikari，“使用数据挖掘为电力系统开发混合入侵检测系统”，IEEE智能电网汇刊，第6卷，第6期，第3104–3113页，2015年。[6] ——，“利用异构时间同步数据对电力系统中的扰动和网络攻击进行分类”，IEEE工业信息学汇刊，第11卷，第3期，第650–662页，2015年。[7] R. C. Borges Hink、J. M. Beaver、M. A. Buckner、T. Morris、U. Adhikari和S. Pan，“用于区分电力系统扰动与网络攻击的机器学习”，载于《2014年第七届国际弹性控制系统研讨会》(ISRCS)，2014年，第1–8页。[8] L. Lamport，“分布式系统中的时间、时钟与事件排序”，Commun. ACM，第21卷，第7期，第558–565页，1978年7月。 [在线]。可访问：https://doi.org/ 10 1145 359545 359563 。/。J. Goh、S. Adepu、[9] K. N. Junejo和A. Mathur，“支持安全水处理系统设计研究的数据集”，载于《关键基础设施安全》，G. Havarneanu、R. Setola、H. Nassopoulos和S. Wolthusen编。施普林格：施普林格国际出版，2017年，第88–99页。[10] C.-C. Sun、C.-C. Liu和J. Xie，“电网的信息物理系统安全：最新进展”，《电子学》，第5卷，第3期，2016年。 [在线]。可访问：https://www.mdpi.com/ 2079 9292 5 3 40 -///J. M. Beaver、R. C. Borges-Hink和M. A. Buckner，[11] “用于检测恶意SCADA通信的机器学习方法评估”，载于《2013年第12届国际机器学习与应用会议》，第2卷，2013年，第54–59页。[12] P. Mehlitz、D. Giannakopoulou和N. Shafiei，“使用RACE分析空域数据”，载于《2019年IEEE/AIAA第三十八届数字航空电子系统会议》(DASC)，2019年，第1–10页。[13] H. Chauhan、V. K. Garg、A. Natarajan和N. Mittal，“用于在线谓词检测的分布式抽象算法”，载于《第三十二届IEEE可靠分布式系统研讨会》(SRDS)论文集，2013年，第101–110页。[14] N. Mittal和V. K. Garg，“计算切片的技术与应用”，《分布式计算》，第17卷，第3期，第251–277页，2005年。[15] V. A. Ogale和V. K. Garg，“检测分布式计算中的时序逻辑谓词”，载于《第二十一届分布式计算国际研讨会》(DISC)论文集，2007年，第420–434页。[16] M. Mostafa和B. Bonakdarpour，“分布式系统中LTL规范的去中心化运行时验证”，载于《第二十九届IEEE国际并行与分布式处理研讨会》(IPDPS)论文集，2015年，第494–503页。[17] V. T. Valapil、S. Yingchareonthawornchai、S. S. Kulkarni、E. Torng和M. Demirbas，“使用SMT求解器监控部分同步分布式系统”，载于《第十七届运行时验证国际会议》(RV)论文集，2017年，第277–293页。[18] L. M. Danielsson和C. S´anchez，“去中心化流运行时验证”，载于《第十九届运行时验证国际会议》(RV)论文集，2019年，第185–201页。[19] C. Colombo和Y. Falcone，“基于全局时钟组织分布式系统中的LTL监视器”，《系统设计中的形式化方法》，第49卷，第1–2期，第109–158页，2016年。[20] B. Bonakdarpour和B. Finkbeiner，“Hyper-LTL的运行时验证”，载于《第十六届运行时验证国际会议》论文集，2016年，第41–45页。[21] B. Bonakdarpour、P. Fraigniaud、S. Rajsbaum、D. A. Rosenblueth和C. Travers，“去中心化异步崩溃容错运行时验证”，载于《第二十七届并发理论国际会议》(CONCUR)论文集，2016年，第16:1–16:15页。[22] L. Lamport和N. Lynch，《理论计算机科学手册》。阿姆斯特丹：爱思唯尔科学出版社有限公司，1990年，第B卷，第18章：分布式计算：模型与方法。[23] L. Convent、S. Hungerecker、M. Leucker、T. Scheffel、M. Schmitz和D. Thoma，“TeSSLa：时序流规范语言”，载于《形式化方法：基础与应用》，T. Massoni和M. R. Mousavi编。施普林格：施普林格国际出版，2018年，第144–162页。[24] H. Torfah，“用于实时性质的基于流的监视器”，载于《运行时验证》，B. Finkbeiner和L. Mariani编。施普林格：施普林格国际出版，2019年，第91–110页。[25] M. Leucker、C. S´anchez、T. Scheffel、M. Schmitz和D. Thoma，“部分信息条件下定时事件流的运行时验证”，载于《运行时验证：第十九届国际会议，RV 2019，葡萄牙波尔图，2019年10月8日至11日，论文集》。柏林、海德堡：施普林格出版社，2019年，第273–291页。 [在线]。可访问：https://doi.org/10.1007/978-3-030-32079-9 16

[26] M. Leucker, C. S´anchez、T. Scheffel、M. Schmitz和A. Schramm，“非同步到达情况下实时事件流的运行时验证”，*Software Quality Journal*，第28卷，第2期，第745–787页，2020年。[27] R. Ganguly、A. Momtaz和B. Bonakdarpour，“部分同步条件下的分布式运行时验证”，载于第24届分布式系统原理国际会议 (OPODIS 2020)，第184卷，2021年，第20:1–20:17页。 [在线]。可访问：https://drops.dagstuhl.de/opus/volltexte/2021 13505 /R. Ganguly、Y. Xue、A. [28] Jonckheere、P. Ljung、B. Schornstein、B. Bonakdarpour和M. Herlihy，“跨链协议中度量时序性质的分布式运行时验证”，载于2022年IEEE第42届分布式计算系统国际会议 (ICDCS)。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，2022年7月，第23–33页。 [在线]。可访问：https://doi.ieeecomputersociety.org/10.1109/ICDCS54860.2022.00012

XI. 附录

A. LOLA 语法

流表达式的构造方式如下：

- 若 c 是类型为 T 的常量，则 c 是类型为 T 的原子流表达式
- 若 s 是类型为 T 的流变量，则 s 是类型为 T 的原子流表达式。
- 若 $f: \mathrm{T}_1 \times \mathrm{T}_2 \times \cdots \mathrm{T}_k \rightarrow \mathrm{T}$ 是一个 k 元运算符，并且对于 $1 \leq i \leq k$， e_i 是类型为 T_i 的表达式，则 $f(e_1, e_2, \cdots, e_k)$ 是类型为 T 的流表达式
- 如果 b 是布尔值类型的流表达式，且 e_1、 e_2是 T 类型的流表达式，那么 ite(b, e_1, e_2) 是 T 类型的流表达式，其中 ite 是如果-那么-否则的缩写形式。
- 如果 e 是 T 类型的流表达式， c 是 T 类型的常量，且 i 是整数，那么 $e[i, c]$ 是 T 类型的流表达式。 $e[i, c]$ 表示表达式e 从当前位置偏移 i 个位置后的值。如果偏移量使其超出流的末尾或位于流的开头之前，则默认值为 c。

此外，LOLA 可用于计算增量统计。对于给定的流 α，函数 $f_\alpha(v, u)$ 用于计算度量，其中 u 表示截至目前的度量值， v表示当前值。给定一个值序列v_1、 v_2、 $\cdots$、 v_n，以及默认值 d，数据的度量可表示为

$$u = f_\alpha(v_n, f_\alpha(v_{n-1}, \cdots, f_\alpha(v_1, d)))$$

此类函数包括计数、 $f_{count}(v, u) = u+1$、求和、 $f_{sum}(v, u) = u + v$、最大值、 $f_{max}(v, u)$ =最大值$\{v, u\}$等。平均值等聚合函数可以使用两个增量函数——计数和求和——来定义。

B. 证明

引理 2： 设 $\mathcal{A} = \{S_1, S_2, \cdots, S_n\}$ 是一个分布式系统，且 φ 是一个 LOLA 规范。算法 1 在监控一个会终止的分布式系统时会终止。

证明 1：首先，我们注意到，我们的算法是为会终止的系统设计的；此外，还要注意，会终止的程序只会产生有限的分布式计算。为了证明该引理，假设系统在终止时向所有监控进程发送停止信号。当监视器收到此类信号时，它会使用终止关联方程对输出流表达式进行求值。这可能产生两种情况：一种是求值所需的所有值都已被观察到；另一种是求值所需的值尚未被观察到。第一种情况下，监控进程的终止是显然的；而第二种情况下，监控进程的终止取决于将此类未观察到的流值替换为流表达式的默认值。因此，监控进程最终会终止。

定理 3： 算法 2 解决了第 IV 节中提出的问题。

证明 2： 我们通过将算法2分为三个步骤，证明其可靠性和正确性。第一步

我们证明，给定一个 LOLA 规范 φ，在长度为 N 的分布式计算 $(\mathcal{E}, \rightsquigarrow)$ 上计算时，输出流的值与将该分布式计算划分为 $\frac{N}{k}$ 个计算轮次、每个轮次长度为 k 时的值相同。其次，我们证明，对于所有时间实例，流方程最终都会在通信轮次之后得到评估。最后，我们证明，所有已评估结果的集合在系统中的所有监视器之间保持一致。

步骤 1：根据我们的方法，可以看出，输出流变量的值是在时间为 j 的一致割中所包含的事件上求得的。因此，我们可以将证明归结为：

$$\mathsf{Sr}(\mathcal{E}, \rightsquigarrow) = \mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}}, \rightsquigarrow)$$

- ($\Rightarrow$) 设 $\mathcal{C}_k$ 为一致割，使得对于某个 $k \in [0, |\mathcal{E}|]$，$\mathcal{C}_k$ 属于 $\mathsf{Sr}(\mathcal{E}, \rightsquigarrow)$，但不属于 $\mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}}, \rightsquigarrow)$。这意味着 $\mathcal{C}_k$ 的前沿 $\mathsf{front}(\mathcal{C}_k) \not\subseteq \mathcal{E}_1$，且 $\mathsf{front}(\mathcal{C}_k) \not\subseteq \mathcal{E}_2$，并且 $\cdots$ 与 $\mathsf{front}(\mathcal{C}_k) \not\subseteq \mathcal{E}_{\frac{N}{k}}$。然而，这是不可能的，因为根据第五节 VII-B 中的计算轮次构造，必定存在一个 $\mathcal{E}_i$，使得 $1 \leq i \leq \frac{N}{k}$，且 $\mathsf{front}(\mathcal{C}_k) \subseteq \mathcal{E}_i$。因此，这样的 $\mathcal{C}_k$ 不可能存在，并且 $(\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathsf{Sr}(\mathcal{E}, \rightsquigarrow) \implies (\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}}, \rightsquigarrow)$。
- ($\Leftarrow$) 令 $\mathcal{C}_k$ 为一个一致割，使得对于某个 $k \in [0, |\mathcal{E}|]$，$\mathcal{C}_k$ 属于 $\mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}} \rightsquigarrow \mathcal{E} \rightsquigarrow)$，但不属于 $\mathsf{Sr}(\ ,\ )$。这意味着，对于某个 $i \in [1, \frac{N}{k}]$，$\mathcal{C}_k \subseteq \mathcal{E}_i$，$\mathsf{front}(\ )$ 和 $\mathsf{front}(\ )$。然而，由于 $\forall i \in [1, \frac{N}{k}].\mathcal{E}_i \subset \mathcal{E}$，这是不可能的。因此，这样的 $\mathcal{C}_k$ 不可能存在，并且 $(\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}}, \rightsquigarrow) \implies (\alpha_1, \alpha_2, \cdots, \alpha_n) \in \mathsf{Sr}(\mathcal{E}, \rightsquigarrow)$。

因此，$\mathsf{Sr}(\mathcal{E}, \rightsquigarrow) = \mathsf{Sr}(\mathcal{E}_1.\mathcal{E}_2 \cdots \mathcal{E}_{\frac{N}{k}}, \rightsquigarrow)$。

步骤 2：给定一个输出流表达式 s_i 和依赖图 $G = \langle V, E \rangle$，对于每个 $\langle s_i, s_k, w \rangle \in E$，在时间实例 $j \in [1, N]$，α_k 处的值求值时，未观察到 $j+w$ 的 $\neq \natural$，或 $j+w$ 的 $= \natural$，或 $w+j$ 的 α_k。

- 如果 $\alpha_k(j+w) \neq \natural$，则我们对流表达式进行求值
- 如果 $\alpha_k(j+w) = \natural$，则至少存在另一个监视器，使得 $\alpha_k(j+w) \neq \natural$。因此对流表达式进行求值，然后将求值结果共享给所有其他监视器
- 如果未观测到 $\alpha_k(w+j)$，则在未来的某个评估轮次中，某个监视器上 $\alpha_k(j+w) \neq \natural$，从而对流表达式进行求值 s_i

类似地，可以证明 $\langle s_i, t_k, w \rangle \in E$。

步骤 3：我们方法中的每个监视器都具备容错能力，所有监视器之间均进行通信。我们还假设所有消息最终都会被监视器接收。这保证了每个监视器都能直接或间接地读到所有观测。

结合步骤 1 和步骤 2，算法的可靠性和正确性1 得以证明。

定理4：设 φ 是一个 LOLA 规范，且 $(\mathcal{E}, \rightsquigarrow)$ 是由 $|\mathcal{A}|$ 个流组成的分布式流。算法2 在使用 $|\mathcal{M}|$ 个监视器时的消息复杂度为

$$O\left(\epsilon^{|\mathcal{A}|} N |\mathcal{M}|^2\right) \quad \Omega(N|\mathcal{M}|^2)$$

证明 3：

我们分析算法 2 各部分的复杂度。该算法包含一个嵌套循环。外层循环迭代 $\lceil N/k \rceil$ 次，即 $O(N)$。内层循环取决于流表达式的唯一求值次数。

- 上界 由于我们假设系统具有部分同步性，每个事件的发生时间可能存在 ϵ 的偏差。因此，唯一求值的最大数量为 $O(\epsilon^{|\mathcal{A}|})$ 阶。
- 下界 唯一求值的最小数量为 $\Omega(1)$ 阶。

在通信阶段，每个监视器向所有其他监视器发送 $|\mathcal{M}|$ 条消息，并从所有其他监视器接收 $|\mathcal{M}|$ 条消息。即 $|\mathcal{M}|^2$。因此，消息复杂度为

$$O\left(\epsilon^{|\mathcal{A}|} N |\mathcal{M}|^2\right) \quad \Omega(N|\mathcal{M}|^2)$$

顺便提一下，当监视器和逻辑表达式的可读性较高时，复杂度更接近下界；而当可读性较低且涉及算术表达式时，复杂度则更接近上界。

C. 工业控制系统

a) SWaT 数据集：安全水处理 (SWaT) [9]采用了一座全面运行、规模缩小且占地面积小的水处理厂，每分钟产出 5 加仑经过双重过滤的水。该设施由六个主要流程组成，分别对应水处理设施的物理组件和控制组件。首先，在流程 P1 中，原水被引入并储存在储罐中。随后，水进入预处理流程 P2，在此评估水质并通过化学加药加以维护。接着，水进入 P3，利用精细过滤膜去除不需要的物质。在 P4 的脱氯过程中，剩余的氯会被去除；随后，水被泵入反渗透系统 (P5)，以减少无机杂质。最后，在 P6 中，反渗透系统中的水被储存起来，等待分配。

该数据集根据攻击发生的点和阶段，将系统受到的不同攻击分为四种类型：单阶段单点、单阶段多点、多阶段单点和多阶段多点。本文主要关注涉及多个阶段或多个点的攻击。以下列出部分 LOLA 规范。

输入 FIT-101：无符号整数 输入 MV-101：布尔值 输入 LIT-101：无符号整数 输入 P-101：布尔值 输入 FIT-201：无符号整数 输出 进水相关性：= ite(MV-101 == true, FIT-101 > 0, FIT-101 == 0) 输出 出水相关性：= ite(P-101 == true, FIT-201 > 0, FIT-201 == 0) 输出 水箱相关性：= ite(MV-101 == true | P-101 ==true, LIT-101 = LIT-101[-1, 0] +FIT-101[-1, 0]- FIT-201[-1, 0])

其中，FIT-101 是流量计，用于测量流入原水箱的水量；MV-101 是控制流向原水箱水流的电动阀；LIT-101 是原水箱的液位变送器；P-101 是将水从原水箱泵送至第二级的泵；FIT-201 是控制加药泵的流量变送器。上述 LOLA 规范通过进水相关性（分别为出水相关性）输出表达式检查进水流量计与阀门对（分别为出水流量计与泵对）的正确性。另一方面，水箱相关性用于检查储罐水位是否与进水和出水流量计的数据相符。

```
输入 AIT-201：无符号整数
输入 AIT-202：无符号整数
输入 AIT-203：无符号整数
输出 观测总数：=观测总数[-1,      0] + 1
输出 氯化钠平均值：= (氯化钠平均值[-1,     0] *
     观测总数[-1 ， 0] +AIT-201) / 观测总数
输出 氯化氢平均值 :=(氯化氢平均值[-1,     0] * 观测总数[-1,
     0] +AIT-202) / 观测总数
输出 次氯酸钠平均值 := (次氯酸钠平均值[-1, 0] *
     观测总数[-1, 0] + AIT-202) / 观测总数
```

其中，AIT-201、AIT-202 和 AIT-203 分别表示水中的氯化钠、氯化氢和次氯酸钠含量；NaClAvg、HClAvg 和 NaOClAvg 则跟踪水中相应化学物质的平均含量，而 numObv 跟踪监视器读入的观测总数。

b) 电力系统攻击数据集：电力系统攻击数据集 [10] 由密西西比州立大学和橡树岭国家实验室开发的三个数据集组成。它包含来自四个相量测量单元（PMU）或同步相量装置的读数，这些装置用于测量电网上的电波。每个 PMU 度量 29 项特征，包括 A至C相、正序、负序和零序的电压相角、电压相量幅值、电流相角和电流相量幅值。它还度量继电器的频率、继电器的频率变化量、继电器的状态标志等。除这 116 项 PMU 度量外，该数据集还包含 12 条控制面板日志、Snort警报以及这 4 个 PMU 的继电器日志。

数据集将事件分类为自然事件/无事件或攻击事件。下面列出了一些使用的 LOLA 规范。第一个旨在检测单相接地（1LG）故障。

```
输入 R1 电流 : 浮点数
输入 R2 电流 : 浮点数
输入 R1-继电器 : 布尔值
输入 R2-继电器 : 布尔值
输出 R1-I-low := R1-I  < 200
输出 R1 电流-高：= R1 电流  > 1000
输出 R2 电流-低：= R2 电流  < 200
输出 R2 电流-高：= R2 电流  > 1000
输出 单相接地：= R1 电流-高 && R2 电流-高 &&
     R1-继电器[+2, 假] && R2-继电器[+2,
     假] && R1 电流-low[+4 , 假] &&
     R2 电流-low[+4, 假]
```

其中，R1 电流和 R2 电流分别表示由 R1 和 R2 相量测量单元测得的电流。此外，R1-继电器和 R2-继电器会跟踪相应继电器的状态。作为单相接地攻击检测的一部分，我们首先根据测得电流的大小，将其归类为低电流或高电流。如果 R1 和 R2 均检测到高电流流动，随后继电器跳闸，再随后出现低电流，我们就将该攻击归类为单相接地攻击。

```
输入 R1-PA1 电流：浮点数输入 R1-PA2 电流：浮点数输
入 R1-PA3 电流：浮点数输出 相位平衡：= (R1-PA1 电流 -
R1-PA2 电流 ) <=10 && (R1-PA2 电流 - R1-PA3 电流 )
 <= 10&& (R1-PA3 电流 - R1-PA1 电流) <= 10
```

其中，R1-PA1 电流、R1-PA2 电流和 R1-PA3 电流分别是由 R1 相量测量单元在 A相、B相和 C 相测得的电流值。该监视器帮助我们检查三相负载是否均衡。

c) 燃气分配系统：燃气分布式系统 [11]是密西西比州立大学关键基础设施保护中心与橡树岭国家实验室合作，从燃气管道系统采集的一组带标签的远程终端单元（RTU）遥测数据流。遥测数据流包含正常运行期间以及发生涉及命令注入和数据注入攻击时，可编程逻辑控制器（PLC）发出和接收的消息。特征集包括管道压力、设定值、来自 PLC 的命令数据、对 PLC 的响应，以及电磁阀、泵和远程终端单元（RTU）自动控制的状态。

最常见的数据注入攻击之一是快速变化。攻击者会连续改变报告的管道压力值，使人们对系统是否正常运行失去信心。下面给出了用于监控此类攻击的相应 LOLA 规范：

```
输入 PipePress：浮点数输入 response：布尔值 输出
fastChange：= ite(response,mod(PipePress-
PipePress[-1, 1000])  <=10, true)
```

其中，PipePress 记录测得的管道压力，而 response 是一个标志变量，用于表示发送给可编程逻辑控制器的消息。这里我们假设默认压力为 1000 磅每平方英寸，且单位时间内允许的压力变化为 10 磅每平方英寸（这些值可根据系统需求进行调整）。同样，我们还制定了 LOLA 规范，用于监控其他数据注入攻击，例如数值波注入、设定值注入、单数据注入等，以及非法设定值、非法 PID 命令等命令注入攻击。

d) RACE 数据集：空域概念评估运行时（RACE）[12] 是美国国家航空航天局开发的一个框架，用于构建基于事件的响应式空域仿真。我们使用的是基于此 RACE 框架构建的数据集。该数据集包含在三个不同日期采集的三组数据。每组数据均在北纬约 37 度、西经约 121 度处记录。该数据集包含 SBS 单元通过 Telnet 发送的全部 8 种类型的消息

应用程序监听端口 30003，但我们只使用 ID 为“MSG 3”的消息，即空中位置消息，其中包含航班的纬度、经度和高度，我们据此验证所有航空器对之间的相互间隔。此外，计算两个坐标之间的距离在计算上开销较大，因为我们需要考虑地球曲率等参数。为了加快距离相关计算，我们将固定纬度距离和经度距离分别设为 111.2 千米和 87.62 千米，代价是会产生可忽略不计的误差范围。相应的 LOLA 规范如下所述：

```
输入 flight1_alt : 浮点数 输入 flight1_lat : 浮点数 输入
flight1_lon : 浮点数 输入 flight2_alt : 浮点数 输入 flight2_
lat : 浮点数 输入 flight2_lon : 浮点数 输出 distDiff :=
sqrt(pow(flight1_alt - flight2_alt, 2)  + pow((flight1_
lon- flight2_lon)*87620, 2) + pow((flight1_lat - flig
ht2_lat)*111200, 2)) 输出 check := distDiff  > 500
```