arXiv:2603.11104v1 [cs.SE] 11 Mar 2026

参数化流的类型安全监控

Jan Baumeister[1], Bernd Finkbeiner[1], 和 Florian Kohn[1]

CISPA 亥姆霍兹信息安全中心，德国萨尔布吕肯 66123
{jan.baumeister, finkbeiner, florian.kohn}@cispa.de

摘要。 基于流的监控是一种实时安全保障机制，适用于无人驾驶飞行器等复杂的信息物理系统。监控器聚合来自传感器及其他来源的输入数据流，实时提供统计信息并评估系统运行状况。由于监控器是安全关键组件，因此必须确保其不存在运行时错误。当监控器必须处理无界数据域（例如数量不受限制的空域参与者）时，这一保证尤其难以实现，因为这需要使用动态数据结构。本文将参数化流以类型安全的方式集成到基于流的监控框架 RTLola 中。参数化流将单个流推广为由无界数量的流实例构成的集合，并提供了一种系统化的内存管理机制。我们证明，不存在运行时错误这一性质通常是不可判定的，但可以通过精化类型系统有效地加以保证，该类型系统确保所有内存引用要么访问成功，要么由默认值提供支持。我们报告了该类型分析在一系列基准测试中的示例规格上的性能，其中包括用于监控自主飞行器的规格。

1 引言

基于流的监控是一种运行时验证方法，适用于无人飞行器等复杂信息物理系统（CPS）。与使用线性时序逻辑（LTL）等逻辑的监控方法不同， [4], 基于流的语言能够处理整数或字符串等丰富数据。运行时监控器（简称监控器）通过输入流收集数据，再将这些数据组合并聚合到输出流中，从而实时评估系统的健康状况。在根据规范生成监控器之前，会对规范进行静态分析，以确定静态内存上界、是否存在运行时错误等关键属性。

这种方法已成功应用于现实世界的应用中。一个突出的例子是基于流的规范语言 RT-Lola，它已用于监控无人驾驶飞机 [5] ，也用于监控汽车的排放 [10]。随着系统日益复杂，监控器必须处理无界数据域。例如，在监控无人机的空域时，附近车辆的数量可能没有上限。一种简单的解决方案是扩展流数据类型，以包含映射或列表等结构。然而，

规范编写者必须确保这些结构中存在相应的值，否则监控器可能会在运行时出错。

Faymonville 等人提出了另一种颇具竞争力的方法，可在规范中直接处理无界域，即参数化流 [13]。参数化流将流推广到无界的实例集合，这些实例可在运行时创建和关闭，以管理内存。实例流的创建、求值和关闭时机由逻辑表达，从而能够静态保证不会因无效内存访问而产生运行时错误。然而，当前方法无法表达异步属性或实时属性，而这对于监控信息物理系统至关重要。

本文将参数化流引入实时流式规范语言 RTLola，以应对在运行时动态创建流实例这一挑战。在 RTLola 的早期版本中，可以使用注释类型（如频率或事件）静态确定流值是否存在。然而，随着参数化流的引入，流值是否存在取决于运行时数据。因此，以往的静态分析方法已不再适用。

我们为 RTLola 提出了一种新的精化类型系统，用于论证流进行求值的时间点。在该系统中，可以使用依赖于运行时值的布尔流表达式，对由频率注释等方式定义的静态已知时间点进行精化。我们证明，这些类型能够捕获 RTLola 中参数化流的动态行为，并展示类型正确的规范可以在不会发生运行时错误的情况下进行监控。

为此，我们给出了 RTLola 中参数化流的语义，并证明保证不存在运行时错误是不可判定的。我们扩展了良构性的定义，以防止监控器出现非确定性行为。我们在 RTLola 框架 [5]中使用类型系统的过近似实现了该方法，并使用航空航天领域的规范和合成规范评估其运行时性能，以验证其可扩展性。

第 1 节的其余部分展示了参数化流的优点，并讨论了相关工作。第 2 节定义语法，第 3 节给出语义。第 4 节定义新的类型系统，并进一步解释其局限性。第 5 节扩展了良构性的定义，第 6 节评估我们的类型检查方法的性能，第 7 节总结全文。

1.1 引入示例

本示例展示了如何结合使用 RTLola 的实时功能和流实例动态创建功能，来监控自主无人机周围的空域。这是 Baumeister 等人在 [6] 中提出示例的简化版本。该监控器使用 RTLola 规范语言编写，用于跟踪与数量先验无上限的空域参与者之间的距离。如果某个空域参与者（称为入侵者）接近飞机并达到危险距离，监控器就会通知飞行器操作员。

```
输入 入侵者_标识符: 无符号整数
输入 距离: 浮点数

输出 距离_按(标识符)
    随其生成 入侵者_标识符
    在以下情况求值 标识符 = 入侵者_标识符 与 距离
    关闭, 当 标识符 = 入侵者_标识符 && 距离 > 10.0

输出 平均值_距离(标识符)
    生成, 与 入侵者_标识符
    求值@1Hz与距离_每(标识符)。聚合(范围: 1秒, 使用: 平均值)。默认值(为: 0.0)
    关闭, 当 标识符 = 入侵者_标识符 && 距离 > 10.0

触发器(标识符)
    在距离 < 5.0 时生成入侵者_标识符
    求值@1Hz 当avg_distance(id).last(或: 0.0) > avg_distance(id) 使用"Intruder"
    关闭, 当标识符 = 入侵者_标识符 && 距离 > 10.0
```

我们假设飞机配备了一个用于测量与入侵者之间距离的传感器。本示例包含两个输入流，用于记录飞机的观测数据。第一个输入流 入侵者_标识符表示入侵者的标识符，第二个流 距离表示与该入侵者之间的距离。

第一个输出流 distance_per按入侵者分组距离测量值，使用单个参数标识符，该参数由流名称后的括号表示。生成子句将此参数赋值为入侵者_标识符输入流，为每个入侵者创建一个流实例。每个实例求值为当前距离，当入侵者与其标识符匹配时；当距离超过指定阈值时，该实例会从内存中移除。

为减少传感器波动，avg_distance 流以 1Hz 的频率计算平均距离。最终的触发器通过比较 avg_distance 流的当前值和先前值，判断入侵者是否正在靠近；当入侵者距离小于阈值时进行此判断。监控此示例实际上可能导致运行时错误，因为触发器的周期性与avg_distance 流可能不同步。这种失同步可能导致触发器访问avg_distance 流中无效或不可用的值，进而造成内存错误。以下各节将展示所提出的类型系统如何有效检测这些错误，从而确保监控安全可靠。

1.2 相关工作

运行时验证已应用于各种系统和属性 [19,18]。例如，可以使用 LTL [4] 或 MTL [3]等时序逻辑来指定属性。基于流的监控方法 [12,20,16] 使用能够处理非二进制数据的规范语言，并受到同步语言 [17,9]的启发。一种使这些语言能够处理动态内存的方法是引入新的数据类型。例如，HLola [15]，将 Haskell 中定义的数据类型嵌入 Lola[12]。采用这种方法时，管理动态内存的责任交由规范编写者承担。另一种方法是使用参数化流 [13]。通过描述流模板，将动态内存直接嵌入规范语言中。随后，这种方法被嵌入其他基于流的语言，例如 HStriver [16]，甚至扩展为追溯式动态参数化 [21]。在这里，一个参数化流具有

不同的配置，以访问监控器的完整历史记录。最近，研究人员开始研究同步语言的细化类型 [11]，以便为流的值建立保证。这些方法都没有提供能够对参数化流的时序进行推理的类型系统。然而，这样的类型系统已被证明具有很高的实用价值[14,7,8]。

2 RTLola 语法

本节介绍 RTLola 规范在抽象语法树（AST）格式中的语法。不过，为了便于阅读，本文中的示例规范采用了一种更易读的语法编写，并且可以解析为本文所述的语法。

RTLola 规范由多个流定义组成。流分为两类：输入流表示系统状态读数，输出流则根据其流声明执行计算。

定义 1（规范）。*RTLola* 规范由一组输入流和一组输出流构成。输入流和输出流都有唯一的标识符，供流表达式使用。输出流定义其参数范围以及三种流声明：一种定义流实例的开始，一种描述流计算，还有一种描述流的关闭行为。规范的语法如下：

规范 输入流 输入 ::=输入(标识符_in) 输出流 输出 ::=输出(标识符_out, #para, 生成, 求值, 关闭)

参数数量 #para ∈ ℕ

生成声明 spawn ::=生成(pac, expr, expr) eval 声明 eval ::=求值(pac, expr, expr)
关闭声明 close ::=关闭(pac, expr) 节奏类型 pac ::=任意 | 事件(ac) | 全局(freq) | 局部(freq) 频率 freq ∈ ℕ激活条件 ac ::=id_属于 | ac 且 ac | ac 或 ac

参数化输出流通过参数扩展了常规输出流，与函数类似。每个流定义多个实例，每个实例对应一组唯一的参数值。输出流通过三个声明指定，这些声明定义实例的生命周期和计算方式。每个声明都包含一种节奏类型和一个用于细化时序的语义条件。生成 声明指定何时创建新实例并设置其参数，求值 声明定义实例何时以及如何计算其值，而关闭 声明则确定何时移除该实例。输出

不带参数的流被视为参数化流，其生成、求值和关闭子句均采用默认值。

我们将每个声明的第一个表达式称为 $When$ 子句，并将第二个表达式（若存在）称为 $With$ 子句。在本文其余部分，我们假定这些表达式以及声明的节奏可通过函数访问。例如，$SpawnWith(sid)$ 返回由 sid 标识的流的生成声明的 With 子句。

节奏类型 可以是固定的全局频率、固定的局部频率，或一个事件；其中事件是同时接收到新值的输入流的组合。若未给出节奏类型，则假定为任意类型。

定义 2（节奏类型）。 节奏类型定义了对表达式进行求值的时间点。其语法在定义 *1* 中给出。

流表达式 以算法方式定义如何根据其他流值、常量和函数计算一个值。其语法定义如下，完整定义见 A 节。它们提供不同的访问运算符来引用流值。每次访问都包含一个流标识符以及一个用于计算目标实例的表达式。同步访问获取当前值，保持访问返回最近的值，偏移访问则引用流在离散点上的历史记录。聚合在过去的值上提供滑动窗口，由一个函数和一段实时持续时间定义。默认表达式处理失败的访问，例如监控器启动时的偏移量。算术表达式、常量和函数由函数表达式处理，而参数访问则引用当前实例的参数值。

表达式 expr ::=同步(id, p) | 保持(id, p)| Offset(id, p, off) | 默认值(expr, expr)| 聚合(id, p, dur, f_a)| 函数(f, expr*) | 常量(const)| 参数(p_idx) | 元组(expr*) 常量 const $\in$ $\mathbb{V}$流标识符 id ::=id_in | id_out 输入流标识符 id_in $\in$ 标识符$^{\uparrow}$输出流标识符 id_out $\in$ 标识符$^{\downarrow}$参数 p $\in$ expr $\times \cdots \times$ expr

3 关系语义

RTLola规范的语义定义为求值模型中的值与流声明所计算出的值之间的关系。一个

求值模型 $\omega \in \mathbb{W}$ 是$StreamMap$ 和$TimeMap$的组合。

$$\begin{aligned} Stream &:= InstanceID \to Time \to \mathbb{V}_\bot \\ StreamMap &:= \mathtt{ID}^\uparrow \uplus \mathtt{ID}^\downarrow \to Stream \\ TimeMap &:= Time \to \mathbb{R} \\ \mathbb{W} &:= StreamMap \times TimeMap \end{aligned}$$

$StreamMap$ 为每个流和实例标识符分配一个由可选值构成的无限定时序列。$TimeMap$ 是一个从离散时间戳到实时值的全函数，后续将用它来推理实时访问。为方便起见，我们根据输入对求值模型上的访问函数进行重载。给定一个求值模型 $\omega \in \mathbb{W}$、对流的流访问 $sid \in \mathrm{ID}^\uparrow \uplus \mathrm{ID}^\downarrow$，以及一个离散时间戳 $t \in Time$，有：

$$\begin{aligned} \omega(sid) &:= stream\_map(sid) \text{ if } \omega = (stream\_map, time\_map) \\ \omega(t) &:= time\_map(t) \qquad \text{if } \omega = (stream\_map, time\_map) \end{aligned}$$

为验证求值模型，语义要求时间保持单调，且模型中的每个值都与流声明所描述的计算值相匹配。

定义 3（规约语义）。设 φ 为一个 *RTLola* 规范。该规范的有效求值模型集合为

$$[\![\varphi]\!] := \left\{ \omega \mid \forall sid \in \mathit{ID}^\downarrow . \omega \vDash \varphi(sid) \wedge \forall t \in Time . \omega(t) < \omega(t+1) \right\}.$$

求值模型 ω 满足某个输出流，当且仅当

$$\begin{aligned} &\omega \vDash \mathit{Output}(sid, p, s, e, c) := \\ &\forall t \in Time, iid \in \mathbb{V}^p . \mathit{Output}(sid, p, s, e, c) \Downarrow^t_{\omega, inst} \omega(sid)(iid)(t) \end{aligned}$$

对于流声明的求值，一组推理规则定义$\Downarrow^t_{\omega,inst}$ 如何在给定模型 ω、离散时间戳 t 和当前流实例 iid的情况下计算其值。这些推理规则会对节奏求值，并使用流实例的前缀，其定义见下一小节。

节奏求值。对流节奏的求值区分四种选择：任意表示始终为真的条件，激活条件将节奏限制在输入数据更新时，而两种频率类型（全局频率和局部频率）则描述实时中的固定时刻。这两种频率类型的区别在于它们的起始点。全局频率从监控器启动时开始，而局部频率的起始点则受流的生成定义所描述的条件限制。

给定求值模型 $\omega \in \mathbb{W}$、离散时间戳 $t \in$ 时间，以及离散起始时间 $start_t \in$ 时间，节奏的评估 $[\![]\!]^t_{\omega, start_t}$ 为

$$\begin{aligned} [\![]\!]^t_{\omega,start_t} &: Pacing\ types \to \mathbb{B} \\ [\![\mathtt{Any}]\!]^t_{\omega,start_t} &:= \top \\ [\![\mathtt{Event}(ac)]\!]^t_{\omega,start_t} &:= [\![ac]\!]^t_\omega \\ [\![\mathtt{Global}(freq)]\!]^t_{\omega,start_t} &:= \omega(t) \bmod freq = 0 \\ [\![\mathtt{Local}(freq)]\!]^t_{\omega,start_t} &:= (\omega(t) - \omega(start_t)) \bmod freq = 0, \end{aligned}$$

其中激活条件 ac 的评估为

$$\begin{aligned}[\![]\!]^t_\omega &: \mathit{Activation\ conditions} \to \mathbb{B} \\ [\![sr]\!]^t_\omega &:= \omega(sr)(t) \neq \bot \\ [\![lhs\ \texttt{and}\ rhs]\!]^t_\omega &:= [\![lhs]\!]^t_\omega \wedge [\![rhs]\!]^t_\omega \\ [\![lhs\ \texttt{or}\ rhs]\!]^t_\omega &:= [\![lhs]\!]^t_\omega \vee [\![rhs]\!]^t_\omega .\end{aligned}$$

前缀 流实例的前缀是截至离散时间戳 t 的有限流值序列 $\mathbb{V}$。它使表达式求值能够访问过去的值，或对实例的全部或部分历史记录进行聚合。然而，前缀并不总是从实例的第一个值开始，因为关闭声明会使监控器从内存中移除过去的值。起始时间定义为实例生成且未关闭的第一个时间，直到 t。我们定义 *IsSpawned* 和 *IsClosed* 谓词，以确定实例何时开始和停止；完整定义见 B.3 节。

据此，*AliveSince* 谓词用于判断流实例当前是否存在。给定一个求值模型 $\omega \in \mathbb{W}$、一个流标识$sid \in$ 标识符$^\uparrow \uplus$标识符$^\downarrow$、一个实例标识 $iid \in InstanceID$，以及一个时间戳$t \in$时间，流实例生成且未关闭的第一个时间戳（截至 t）描述如下：

$$\begin{aligned}&\textit{AliveSince}: \ \mathbb{W} \times \text{标识符}^\uparrow \uplus \text{标识符}^\downarrow \times \textit{InstanceID} \times \text{时间} \ \to \text{时间} \quad \bot \\ &\textit{AliveSince}(\omega, \textit{sid}, \ \textit{iid}, \ \text{结束时间}_{\ t}) \\ &:= \begin{cases} 0 & \text{若 } \textit{sid} \in \text{标识符}^\uparrow \\ \text{开始}_t \text{若 开始}_t = \text{最小值}\{t \mid \textit{IsSpawned}(\omega, \textit{sid}, \ \textit{iid} \ t) & \\ \qquad \wedge \forall t \le t' < \text{结束时间}_{\ t}. \neg \textit{IsClosed}(\omega, \textit{sid}, \ \textit{iid} \ \ t')\} & \\ \bot \qquad \text{否则} & \end{cases}\end{aligned}$$

A前缀 然后收集前缀起始位置之前的所有非可选值；如果该实例在时间 t 时不存活，则返回空序列。

$$\begin{aligned}&\mathit{Prefix} : \mathbb{W} \times \mathtt{ID}^\uparrow \uplus \mathtt{ID}^\downarrow \times \mathit{InstanceID} \times \mathit{Time} \to (\mathit{Time}, \mathbb{V})^* \\ &\mathit{Prefix}(\omega, sid, iid, end_t) \\ &:= \begin{cases} \mathit{Prefix}'(\omega(sid)(iid), start_t, end_t) & \text{if } \mathit{AliveSince}(\omega, sid, iid, t) = start_t \\ & \quad \wedge\ start_t \neq \bot \\ \epsilon & \text{otherwise} \end{cases}\end{aligned}$$

其中前缀′返回从起始$_t$到结束$_t$的非可选值序列：

$$\begin{aligned}&\mathit{Prefix}' : (\mathit{Time} \to \mathbb{V}_\bot) \times \mathit{Time} \times \mathit{Time} \to (\mathit{Time}, \mathbb{V})^* \\ &\mathit{Prefix}'(inst, start_t, end_t) \\ &:= \begin{cases} (end_t, v) & \text{if } end_t = start_t \wedge inst(end_t) = v \\ & \quad \wedge\ v \in \mathbb{V} \\ \epsilon & \text{if } end_t = start_t \wedge inst(end_t) = \bot \\ \mathit{Prefix}'(inst, start_t, end_t - 1) \cdot (end_t, v) & \text{if } inst(end_t) = v \wedge v \in \mathbb{V} \\ \mathit{Prefix}'(inst, start_t, end_t - 1) & \text{if } inst(end_t) = \bot \end{cases}\end{aligned}$$

$$\text{EVAL-EXPR-SYN}\quad \frac{expr \Downarrow^{t}_{\omega,inst} inst' \qquad \omega(sid)(inst')(t) = v}{\texttt{Sync}(sid, expr) \Downarrow^{t}_{\omega,inst} v}$$

$$\text{EVAL-EXPR-HOLD}\quad \frac{expr \Downarrow^{t}_{\omega,inst} inst' \qquad prefix = Prefix(\omega, sid, inst', t) \qquad prefix = prefix' \cdot (t', v)}{\texttt{Hold}(sid, expr) \Downarrow^{t}_{\omega,inst} v}$$

$$\text{EVAL-EXPR-HOLD-DFT}\quad \frac{expr \Downarrow^{t}_{\omega,inst} inst' \qquad prefix = Prefix(\omega, sid, inst', t) \qquad prefix = \epsilon}{\texttt{Hold}(sid, expr) \Downarrow^{t}_{\omega,inst} \bot}$$

图 1：流表达式的推理规则示例

对于实时持续时间的**窗口聚合**，我们接着将流实例的前缀限制为一个窗口切片，该切片从一个实时时间戳开始。给定一个流实例前缀前缀和一个实时时间戳起始$_t \in \mathbb{R}$，流实例的窗口定义如下：

$$\text{窗口}: \ \mathbb{W} \times (\text{时间 } \mathbb{V})^* \to \mathbb{R} \to (\text{时间 } \mathbb{V})^* \text{窗口}(\omega, \text{ 前缀}, \text{ 开始}_t): = \begin{cases} \text{窗口}(\omega, \text{前缀}', \text{开始}_\epsilon) & \text{前缀} = \text{前缀}' \cdot (t, v) \wedge \omega(t) > \text{开始} \\ (t, v) & \text{否则} \end{cases}$$

表达式求值。 我们给出一组推理规则，用于定义流表达式的求值。给定一个求值模型 ω、一个离散时间戳t以及一个实例上下文（即当前流表达式在哪个实例下求值），流表达式的求值$\Downarrow^{t}_{\omega,\ inst}$由 B.1 节中定义的推理规则集确定。

以图 1 中的三条规则为例，它们定义了同步 和 保持 表达式的求值。这两种表达式都引用了由流标识符 和 实例标识符描述的流实例的值。直观而言，同步表达式会访问流实例在求值模型中的当前值。相应的推理规则首先对第二个参数求值，以确定具体的流实例，然后再访问该实例在求值模型中的值。保持访问有两条规则，二者都会计算流实例的前缀，以确定是否存在最后一个值。如果该前缀非空，第一条推理规则会将保持访问求值为这个最后值。否则，适用第二条规则，将表达式求值为 $\bot$ 。

流实例求值。 借助表达式求值与节奏求值，给出一组推理规则，用于对输出流实例进行求值。这组推理规则涵盖求值节奏、动态过滤器$EvalWhen$以及可能并不存活的流实例的不同组合。一般而言，当且仅当流实例处于存活状态，且求值节奏与动态过滤器均求值为真时，该流实例才求值为 $v \in \mathbb{V}$。给定

$$\text{eval-instance-value}\quad \frac{\mathit{AliveSince}(\omega,\ sid,\ iid,\ t) = \mathit{start}_t \qquad \mathit{start}_t \neq \bot \qquad [\![pac]\!]^t_{\omega,\ \mathit{start}_t} \qquad v \in \mathbb{V} \qquad \mathit{when} \Downarrow^t_{\omega,inst} \mathit{true} \qquad \mathit{with} \Downarrow^t_{\omega,inst} v}{\text{输出}(sid,\ \_,\ \_,\ \text{求值}(pac,\ \mathit{when},\ \mathit{with}), \_) \Downarrow^t_{\omega,inst} v}$$

$$\text{eval-instance-false-eval-when}\quad \frac{\mathit{AliveSince}(\omega, sid, iid, t) = \mathit{start}_t \qquad \mathit{start}_t \neq \bot \qquad [\![pac]\!]^t_{\omega,start\ t} \qquad v \in \mathbb{V} \qquad \mathit{when} \Downarrow^t_{\omega,inst} \mathit{false}}{\text{输出}(sid,\ \_,\ \_,\ \text{求值}(pac,\ \mathit{when},\ \mathit{with}), \_) \Downarrow^t_{\omega,inst} \bot}$$

图 2：流实例求值的推理规则

求值模型 ω、离散时间戳 t和实例上下文，流实例的求值$\Downarrow^t_{\omega,\ inst}inst$ 由B.2节中的推理规则定义，部分如图2所示。

4 RTLola类型系统

第 3 节中描述的语义以可能失败的方式处理流访问，这意味着它们可能导致未定义值。对于某些访问，例如 保持(*sid*, *expr*)或Offset(*sid*, *expr* *o*)，这是预期的，规格编写者必须提供默认值。然而，同步访问明确要求被访问的值存在，因此不应产生未定义值。在下一节中，我们将介绍 RTLola 的细化类型系统，以确保满足这一条件。

请看以下示例：

```
1 输入a2 输出 b@1Hz :
= 423 输出 c@a : = b
```

在此示例中，每当 a 获得新值时，流 c 就会求值。然而，并不能保证流 b 会同时求值，因此 c 可能无法访问 b 的值。只有当输入流 a 每隔一秒接收一次值时，才会出现这种情况。但在实践中，输入由系统控制，监控器不应对它们的值或时序作任何假设。

与编程语言类似，RTLola 具有值类型分析，可确保函数使用适当的值调用。本文假设所有规格都基于此分析而类型正确。为求完整，D节中收录了值类型系统的定义。有关所有类型分析的扩展版本，请参阅 Maximilian Schwenger 的博士论文 [23]。

本节安排如下：定义4形式化了语义不会失败这一条件。在此基础上，我们构建了若干引理和定理，用以说明良类型与语义安全性之间的关系，即每个良类型的规范在运行时都不会失败。最后，我们给出定义类型系统的推理规则。

4.1 类型检查可靠性

我们给出一系列定理，以确立类型检查过程的可靠性。定理和引理的完整证明见 C 节。首先，我们定义规范安全性。

定义4（规范安全性）。当且仅当规范的语义不会失败时，该规范才是安全的；也就是说，对于任意输入序列，至少存在一个与该规范一致的世界φ：

$$safe(\varphi) \iff (\forall i \in \mathit{ID}^{\uparrow} \to Time \to \mathbb{V}_{\perp}.\ \exists \omega \in \mathbb{W}.\ \forall s \in \mathit{ID}^{\uparrow}.\ \omega(s)(\epsilon) = i(s) \wedge \omega \in [\![\varphi]\!])$$

接下来，我们定义一个活动谓词，从语义上定义流实例求值为有定义的值的时间点。

定义5（流激活）。当流实例处于存活状态、其语义筛选条件求值为真，且其节奏类型保持成立时，该流实例被声明为活动：

$$\begin{aligned} active(sid, t, inst, \omega) \iff & AliveSince(\omega, sid, inst, t) = start_t \\ & \wedge\ start_t \neq \perp \wedge EvalWhen(sid) \Downarrow^{t}_{\omega, inst} true \\ & \wedge\ [\![EvalPac(sid)]\!]^{t}_{\omega, start_t} \end{aligned}$$

下一个引理将探讨语义可能失败的判据。

引理 1（可失败语义）。当且仅当存在会产生 $\perp$ 的同步访问时，规范的语义才会失败。

该引理的证明是对 RTLola 语法进行结构归纳。我们确定，在归纳过程中，只有 表达式求值（语法）推理规则能够产生 $\perp$。基于此引理，抽象安全性刻画了同步访问不会失败的判据。

定理 1（抽象安全性）。给定一个规范 φ。令同步(标识符 $expr$)为流输出(标识符$'$, n_p, s, e, c)中的一次同步访问。

($\forall t \in$ 时间 $\omega \in \mathbb{W}$, 实例$' \in \mathbb{V}_{\perp}^{n_p}$.

$$\begin{aligned} & active(sid', t, inst', \omega) \wedge expr \Downarrow^{t}_{\omega, inst'} inst \implies active(sid, t, inst, \omega)) \\ & \implies safe(\varphi) \end{aligned}$$

根据引理 1，通过对 活动谓词的定义与 表达式求值（语法）推理规则进行反证即可得出该定理。在最后一步，我们推导出类型检查安全性是抽象安全性的一个充分判据。

定理 2（类型检查安全性）。令谓词类型(φ)表示规范 φ 中的每个流都根据下文给出的推理规则正确地进行了类型检查。那么，对于流同步(标识符 $expr$)中的任意一次同步访问输出(标识符$', n_p, s, e, c$)属于 φ，均有：

$$\begin{aligned} & types(\varphi) \implies \forall t \in Time, \omega \in \mathbb{W}, inst' \in \mathbb{V}_{\perp}^{n_p}. \\ & \quad active(sid', t, inst', \omega) \wedge expr \Downarrow^{t}_{\omega, inst'} inst \implies active(sid, t, inst, \omega) \end{aligned}$$

该定理可直接由同步访问和输出流的类型推断规则得出。

节奏类型系统。 RTLola 节奏类型系统是一种精化类型系统，用于推理流实例进行求值的时间点。节奏类型并不表示整数型或字符串型等传统数据类型，而是定义了一组离散实时刻。例如，节奏类型 @Global(1Hz) 表示整秒。语义类型是布尔流表达式，它们会根据自身的求值结果对这一集合进行精化：当且仅当表达式在某个时间点的求值结果为真时，该时间点才会保留在集合中。因此，节奏类型 @Global(1Hz) when $x > 5$ 表示流 x 求值结果大于 5 时所对应的整秒集合。为简化推理，我们将节奏类型和语义类型分别拆分为独立的格及推理规则。这些格是有界交半格 [24] ，并针对任意但固定的规范进行定义。在本小节中，我们通过类型集合上的 $\sqsubseteq$ “更具体”关系来定义这些格。

节奏类型格有两个组成部分：基于事件的类型和周期类型。基于事件的类型将输入流上的事件表示为正布尔公式，当输入流更新时命题求值为真；其顺序由布尔蕴含定义。周期类型表示固定频率，按其周期的最大公约数排序，并分为局部频率和全局频率。全局频率从监控器启动时开始，而局部频率在实例创建时开始。请考虑以下示例：

```
1 输入 a:整数型2 输出 b@1Hz := 53 输出 c生成当 a
> 42 求值 @1Hz 使用 b
```

在此规范中，输出流 b 以与监控器启动对齐的周期区间产生常量值。输出 c 也以该速率产生值，但它在条件 a > 42 求值为 $true$ 之后才开始。因此，流 b 的节奏类型为 全局频率(1)，而流 c 的节奏类型为局部频率(1)。b 与 c 的这些频率可能不同步，类型检查器应拒绝从 c 访问 b。虽然节奏类型格允许强制转换所有局部频率，但语义类型分析会根据其生成声明对这些局部周期时序进行分析。

节奏类型格。节奏类型集合定义如下：

$$PT ::= \{\text{周期性、全局频率}（x）\text{、局部频率}（x）\text{、事件}（e）\ \top,\ \bot \mid x \in \mathbb{N},\ e \in AC\}$$

$$AC \ni \phi,\ \psi ::= \text{标识符}^{\uparrow} \mid \phi \wedge \psi \mid \phi \vee \psi$$

关于以下方程，关系 $\sqsubseteq_{PT}$ 是最小的：

$$\begin{aligned}
\tau &\sqsubseteq_{PT} \top && \iff \tau \in \mathit{PT} \\
\bot &\sqsubseteq_{PT} \tau && \iff \tau \in \mathit{PT} \\
\text{全局频率}(x) &\sqsubseteq_{PT} \text{周期性} && \iff x \in \mathbb{N} \\
\text{局部频率}(x) &\sqsubseteq_{PT} \text{周期性} && \iff x \in \mathbb{N} \\
\text{全局频率}(x_1) &\sqsubseteq_{PT} \text{全局频率}(x_2) && \iff \exists k \in \mathbb{N}. k \cdot x_2 = x_1 \\
\text{局部频率}(x_1) &\sqsubseteq_{PT} \text{局部频率}(x_2) && \iff \exists k \in \mathbb{N}. k \cdot x_2 = x_1 \\
\text{事件}(e_1) &\sqsubseteq_{PT} \text{事件}(e_2) && \iff e_2 \implies e_1
\end{aligned}$$

请注意，虽然集合 PT 通常是无限的，但对于单个规范中出现的所有类型而言，它是有限的。我们为每个流声明关联一个独立的节奏类型，从而得到一个推断类型三元组。我们用 τ_s 表示生成节奏，用 τ_e 表示求值节奏，用 τ_c 表示关闭节奏。我们用 $\mathcal{T}$ 表示上述类型的三元组，其中 $\sqsubseteq_{PT}$ 关系定义如下：

$$\mathcal{T} \sqsubseteq_{PT} \mathcal{T}' \iff \tau_s \sqsubseteq_{PT} \tau_s' \wedge \tau_e \sqsubseteq_{PT} \tau_e' \wedge \tau_c' \sqsubseteq_{PT} \tau_c$$

所有节奏类型推理规则均列于 E 节。为了帮助理解这些规则，我们讨论节奏输出 推理规则：

PACING-OUTPUT

$$\frac{\begin{array}{c}(\top, \tau_e^s, \top) \models \texttt{Spawn}(\tau_s', e_s^f, e_s) \\ (\tau_s^e, \tau_e^e, \tau_c^e) \models \texttt{Eval}(\tau_e', e_e^f, e_e) \qquad (\tau_s^c, \tau_e^c, \tau_c^c) \models \texttt{Close}(\tau_c', e_c^f) \\ \tau_s^c \in \{\top, \tau_e^s\} \qquad \tau_c^c \in \{\top, \tau_e^c\} \qquad \tau_e^e = \mathit{LocalFreq}(x) \to \tau_e^s = \tau_s^e \wedge \tau_e^c = \tau_c^e \\ \tau_e^c = \mathit{LocalFreq}(x) \to \tau_e^s = \tau_s^c \wedge \tau_e^c = \tau_c^c \qquad \tau_e^s \sqsubseteq_{PT} \tau_s^e \qquad \tau_c^e \sqsubseteq_{PT} \tau_e^c\end{array}}{(\tau_e^s, \tau_e^e, \tau_e^c) \models \texttt{Output}(sid, p\_def, \texttt{Spawn}(\tau_s', e_s^f, e_s), \texttt{Eval}(\tau_e', e_e^f, e_e), \texttt{Close}(\tau_c', e_c^f))}$$

该规则适用于输出流，并首先为生成、求值和关闭 声明推断节奏类型。对于每个声明，都会推断出生成、求值和关闭节奏，分别由元组元素的下标表示。这些类型根据声明中的当 和与 表达式推断得出，而子句的求值节奏还受到其注释类型的限制。所得节奏类型即为相应声明的求值节奏。

对于生成声明，我们要求其生成与关闭节奏为$\top$。对于关闭声明，同样的条件适用，但有一个例外：它可以以与该流相同的局部频率运行。因此，其生成与关闭节奏仅允许为 $\top$或等于该流的节奏。最后，推理规则要求，在求值声明中被同步访问的流的生成与关闭节奏，必须与为输出流推断出的生成与关闭节奏兼容。若求值声明的节奏是局部频率，则该条件会从兼容加强为相等，以防止出现移位周期性，如上例所示。

移位周期性也会导致第1.1节中的示例并非类型良好。在那里，该移位是由于添加了触发器的生成当 注释。

语义类型格。语义类型允许根据运行时信息细化节奏类型。它们被定义为布尔流表达式，其类型格由流表达式的语义蕴含给出。语义类型集合定义为ST:
$= \{Expr(e)\ \top, \bot \mid e \in$ 表达式$\}$，且关系 $\sqsubseteq_{ST}$ 相对于以下等式是最小的:

$$\begin{aligned} \pi \sqsubseteq_{ST} \top \quad &\iff \pi \in ST \\ \bot \sqsubseteq_{ST} \pi \quad &\iff \pi \in ST \\ Expr(e_1) \sqsubseteq_{ST} Expr(e_2) &\iff \forall t \in Time. [\![e_2]\!]^t \to [\![e_1]\!]^t \end{aligned}$$

其中，$[\![e]\!]$ 根据第 3 节中介绍的语义，在适当的上下文中定义了 e的求值。

与节奏类型一样，我们同时推理生成、求值和关闭声明的语义类型。我们使用 Π 来指代语义类型的三元组 π_s、 π_e 和 π_c，对于该三元组， $\sqsubseteq_{ST}$ 关系定义如下:

$$\Pi \sqsubseteq_{ST} \Pi' \iff \pi_s \sqsubseteq_{ST} \pi'_s \wedge \pi_e \sqsubseteq_{ST} \pi'_e \wedge \pi'_c \sqsubseteq_{ST} \pi_c$$

语义类型检查的推理规则见附录 F。如有必要，将包含该表达式的流的标识符标注为 $\models$ 关系的上下文。下面，我们使用 *Sem-Sync-Output* 规则直观地说明语义类型推断。

$$\textsc{Sem-Sync-Out}\quad \frac{\begin{array}{c}\texttt{Output}(sid, n_p, s, e, c) \in \texttt{Output} \qquad \texttt{Output}(sid', n'_p, s', e', c') \in \texttt{Output} \\ \forall 1 \le k \le n.\ e_k = \texttt{Parameter}(i) \wedge k \cong^{sid}_{sid'} i \\ \Pi \models \texttt{Output}(sid, n_p, s, e, c) \qquad \Pi' \sqsubseteq_{ST} \Pi\end{array}}{\Pi' \models_{sid'} \texttt{Sync}(sid, (e_1, ..., e_n))}$$

该推理规则为从标识符为sid'的输出流到标识符为 sid的输出流的同步访问赋予类型。它要求访问所得的类型 Π' 比被访问输出流的推断类型更具体。全称量化陈述对同步访问提出了额外要求，我们称之为参数相等性。

确保访问实例与被访问实例的时序兼容，事实上还不够。请看下面的示例:

```
1 输入i1: 64 位整数2 64 3 21输入 i: 整数型输出a(p: 64 位整数) 使用以下内容生成 i1 ...4 输出b(p2: 64 位整数, p3: 64 位整数, p4: 64 位整数)
5     使用以下内容生成(i2,i1,i1)6     使用以下内容求值7
          a(p2)//无效8
a(p3)//有效9          a(p4)//有效
```

虽然流 b的生成和求值节奏类型与流 a的节奏兼容，但第 7 行中的访问仍然无效。系统无法保证输入 i1 和 i2 会收到相同的值。

不过，第 8 行和第 9 行中的参数访问是有效的，因为参数 p3 和 p4都由流的生成表达式中的 i1 实例化。访问流的参数 p1也同样如此，这保证了它们始终求值得到相同的值。

直观而言，节奏和语义类型保证访问流存在任意实例，但不一定能确保由访问流名称后构成参数值的表达式所标识的访问实例存在。为此，必须满足两个条件：首先，构成访问流参数值的所有表达式都必须是对访问流参数的参数访问。其次，这些参数必须相等，即它们由同一个流表达式实例化：设 $o^1 =$ Output(sid^1, n, Spawn(pt^1, e_f^1, (e_1^1, ..., e_n^1)) e^1, c^1) 和 $o^2 =$ Output(sid^2, m, Spawn(pt^2, e_f^2, (e_1^2, ..., e_m^2)) e^2, c^2) 均为输出流。那么：

$$\forall 1 \leq i \leq n. \forall 1 \leq j \leq m.\ i \cong_{sid^2}^{sid^1} j \iff e_i^1 =_s e_j^2$$

其中 $=_s$ 表示表达式的语法相等。

对于节奏兼容的流，参数相等性可静态保证不同流的参数在运行时具有相同的值。通过将同步访问中用于标识被访问实例的表达式限制为访问流中与被访问流参数匹配的参数，语义同步输出 推理规则可保证被访问的确切实例存在。

4.2 类型检查过程的局限性。

本节将详细讨论类型推断的局限性。首先，我们分析流表达式蕴含的可判定性，这一概念用于语义类型格以及 RTLola 的一般情形。我们给出了一个不可判定性结果，并展示如何通过对语义类型格进行过近似来规避这一问题。

定理 3（RTLola 不可判定性）。良构 *RTLola* 规范的语法片段的可满足性是不可判定的。

证明。 我们从 Robinson 算术的本质不可判定性推导出不可判定性 [22]。设 φ 是任意一个良构规范。由于该规范是良构的，对于每个输入序列都只存在一个求值模型，因此满足一致性判据。根据 Robinson 算术的本质不可判定性，可以得出流表达式的可满足性是不可判定的，因为它们是 Robinson 算术的一致扩展。因此，RTLola 整体的可满足性是不可判定的。

定理 3 推导出，流表达式蕴含是不可判定的。因此，在类型检查器 [2],的实现中，流表达式蕴含通过表达式的语法等价进行过近似。当需要借助算术理论推理某个蕴含时，这种过近似可能会导致特定规格被拒绝；例如，类型推断无法确保 $i > 7 \rightarrow i > 5$。

5 定义良好的参数化规格

第 4 节中的类型系统确保任意输入序列都存在一个求值模型。然而，语义仍允许存在多个求值模型。这可能导致监控器出现非确定性，而这在安全关键环境中是不理想的。我们遵循 Lola [12] 中的定义，以识别具有确定性行为的规格。

定义 6（定义良好性）。*RTLola* 规范定义良好，当且仅当对于任意一组输入流，它恰好有一个求值模型。

由于无法通过语法检查此定义，我们定义了一个用于分析流之间依赖关系的语法充分判据。给定一个 RTLola 规范 φ。φ 的依赖图是一个带权有向多重图 $G = (V, E)$，其中

$$V = \text{标识符}^{\uparrow} \uplus \text{标识符}^{\downarrow} E \subseteq V \times (L \times A) \times VL = \{\text{生成、}\textit{EvalWhen}\text{、求值、关闭}\} A = \{\text{同步}\} \cup \{\text{保持}\} \cup (\textit{Offset} \times \mathbb{N}) \cup \{(\textit{Aggr} \quad \mathbb{R})\}$$

其中 E 满足：对于流声明$sid \in$标识符$^{\downarrow}$中的任意表达式$expr$，根据其位置 $l \in L$，有：

$$\begin{aligned} expr = \texttt{Sync}(sid', p) &\implies (sid, (l, Sync), sid') \in E \\ expr = \texttt{Offset}(sid', p, off) &\implies (sid, (l, (Offset, off)), sid') \in E \\ expr = \texttt{Hold}(sid', p) &\implies (sid, (l, Hold), sid') \in E \\ expr = \texttt{Aggregate}(sid', p, dur, f) &\implies (sid, (l, (Aggr, dur)), sid') \in E \end{aligned}$$

与 Lola 中的流方程相比，参数化流包含多个流表达式。因此，我们在边中加入访问位置，以编码这些位置之间的约束，例如，在时间 t 给出的关闭表达式不会影响流在 t 时的求值。此外，还需要在依赖图中表示不同的访问。我们根据规范的依赖图定义其良构性标准。

定义 7（良构性）。当且仅当依赖图 $G = (V, E)$ 中的所有环 $\{(v_1, (l_1, a_1)\ v_2) \ldots (v_n, (l_n, a_n)\ v_1)\} \in E^*$ 都满足以下条件时，规范 φ 是良构的：

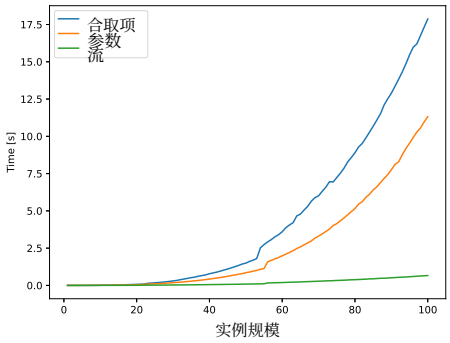
$$(\exists i. l_i = Close) \vee (\forall i. l_i = EvalWith \wedge \exists i'. a_{i'} = (Offset, o) \wedge o \neq 0)$$

采用这一调整后的良构性定义后，我们可以证明与原始 Lola 论文 [12] 中相同的定理。

命题 1（良构性蕴含良定义性）。每个良构的 *RTLola* 规范都是良定义的。

规范 #	行数	时间 [毫秒]
看门狗	9	0.69
RCC	10	0.84
FFD	15	2.82
入侵者	14	4.76
航路点	15	5.51
地理围栏	86	611.32

(a) 性能结果 of the 航空航天基准测试。



(b) 人工基准测试的性能结果。

图 3：评估结果

6 评估

我们使用来自航空航天领域 [5,6] 的现实规范来评估类型分析的性能。此外，我们还使用人工生成的规格，这些规格的流数量逐步增加，以量化其可扩展性。本文的 G 节收录了所有规格。

实验在一台配备 M1处理器和 16 GB 内存的 2020年 MacBook Pro 上原生运行。我们使用 Criterion [1] 微基准测试库，自动预热并采样类型分析的运行时间中位数；采样数据来自至少 100 次执行。

对于现实基准测试，图3a显示，分析地理围栏等真实世界的复杂规格所带来的性能影响微乎其微，分析耗时不到一秒。

这三个人工规格均经过设计，可任意扩展，并代表分析所面临的最坏情况。第一个规范评估了类型检查分析的性能：随着流的数量增加，这些流会以同步方式在链中相互访问。第二个规范通过为前一个案例中的每个流逐步增加参数，评估参数带来的性能影响。最后，我们通过在求值声明中引入$when$ 语句，并在每一步添加一个合取项，来测试语义类型检查的性能。

图 3b 中的人工性能基准测试表明，与不含参数或合取项的流基准测试相比，为流添加参数或为求值声明添加合取项，会影响类型分析的性能。虽然运行时似乎会随实例规模呈指数级增长，但这很可能是二次关系，源于类型检查器底层的约束求解算法。总体而言，即使最大表达式中包含多达 100 个流和同样多的合取项，类型分析的运行时间也没有超过 18 秒。

7 结论

本文将参数化流引入 RTLola，使其能够在实时环境中处理无限数据域。我们为使用这些流监控安全关键型信息物理系统（例如无人飞行器）提供了形式化基础。为确保可靠性，我们确定了 RTLola 规范的一个安全子集，并保证这些规范在运行时不会失败。为此，我们提出了一种新颖的精化类型系统，用于静态验证这些规范。与编程语言中的传统类型系统不同，这种方法会对流的时序行为进行推理，确保所需的值在需要时可用。鉴于该问题的不可判定性，我们实现了语法过近似。此外，我们还扩展了良构规范的定义，以检测可能导致非确定性的时序环。

基于航空航天示例和最坏情况基准测试的评估表明，所提出的类型分析具有良好的可扩展性，且语法片段足以满足现实应用的需求。展望未来，我们计划通过抽象解释来完善安全规范片段，并将语义推理纳入分析。此外，我们将探索在实时环境中为参数化规格生成更高效的监控器。

参考文献

1. Criterion——由统计数据驱动的微基准测试库（2023），https://crates.io/crates/criterion 2. Rtlola 前端——crates.io 上的 Rust crate（2023），https://crates.io/crates/ rtlola-frontend 3. Basin, D.A., Klaedtke, F., Zalinescu, E.：乱序数据流上的时序属性运行时验证。载于：Majumdar, R., Kuncak, V.（编）《计算机辅助验证——第29届国际会议，CAV 2017》，德国海德堡，2017年7月24日至28日，论文集，第一部分。《计算机科学讲义》，第10426卷，第356–376页。施普林格（2017）。https://doi.org/10.1007/ 978-3-319-63387-9 18 4 . Bauer, A., Leucker, M., Schallhart, C.：LTL 和 TLTL 的运行时验证。《ACM 软件工程与方法学汇刊》（TOSEM）**20**(4), 1–64（2011）5. Baumeister, J., Finkbeiner, B., Kohn, F., Löhr, F., Manfredi, G., Schirmer, S., Torens, C.：无人驾驶航空器监控：规范、集成与经验教训。载于：Gurfinkel, A., Ganesh, V.（编）《计算机辅助验证——第36届国际会议，CAV 2024》，加拿大魁北克省蒙特利尔，2024年7月24日至27日，论文集，第二部分。《计算机科学讲义》，第14682卷，第207–218页。施普林格（2024）。https://doi.org/10.1007/978-3-031-65630-9_10 6. Baumeister, J., Finkbeiner, B., Kohn, F., Scheerer, F.：基于流的监控教程。载于：Platzer, A., Rozier, K.Y., Pradella, M., Rossi, M.（编）《形式化方法——第26届国际研讨会，FM 2024》，意大利米兰，2024年9月9日至13日，论文集，第二部分。《计算机科学讲义》，第14934卷，第624–648页。施普林格（2024）。https://doi.org/10.1007/978-3-031-71177-0_33

7. Baumeister, J., Finkbeiner, B., Schirmer, S., Schwenger, M., Torens, C.：RTLola 获准起飞：自主飞行器监控。载于：Lahiri, S.K., Wang, C.（编）《计算机辅助验证——第32届国际会议，CAV 2020》，美国加利福尼亚州洛杉矶，2020年7月21日至24日，论文集，第二部分。《计算机科学讲义》，第12225卷，第28–39页。施普林格（2020）。https://doi.org/10.1007/ 978-3-030-53291-8 38 . Baumeister, J., Finkbeiner, B., Schwenger, M., Torfah, H.：实时属性的现场可编程门阵列流监控。《ACM 嵌入式计算系统汇刊》**18**(5s), 88 3358220 9 1 88 24 2019 10 1145 : – : ( ). https://doi.org/ . / . Berry, G., Cosserat, L.：ESTEREL 同步编程语言及其数学语义。载于：Brookes, S.D., Roscoe, A.W., Winskel, G.（编）《卡内基梅隆大学并发研讨会》，美国宾夕法尼亚州匹兹堡，1984年7月9日至11日。《计算机科学讲义》，第197卷，第389–448页。施普林格（ 1984 10 1007 3 540 15670 4_19 ）。https://doi.org/ . / - - - . Biewer, S.[10], Finkbeiner, B., Hermanns, H., Köhl, M.A., Schnitzer, Y., Schwenger, M.：RTLola 上机运行：在手机上测试实际驾驶排放。载于：Groote, J.F., Larsen, K.G.（编）《系统构造与分析工具和算法——第27届国际会议，TACAS 2021》，作为欧洲软件理论与实践联合会议 ETAPS 2021 的一部分举行，卢森堡卢森堡市，2021年3月27日至4月1日，论文集，第二部分。《计算机科学讲义》，第12652卷，第365–372页。施普林格（2021）。https: //doi.org/ 10 1007 978 3 030 72013 1_20 . / - - - - . Chen, J., de Mendonça, J.L.V.[11], Ayele, B., Bekele, B.N., Jalili, S., Sharma, P., Wohlfeil, N., Zhang, Y., Jeannin, J.：使用细化类型进行同步编程。CoRR **abs/2406.06221** （2024）。https://doi.org/10.48550/ARXIV.2406. 06221 12. d'Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H.B., Mehrotra, S., Manna, Z.：Lola：同步系统的运行时监控。载于：第12届时间表示与推理国际研讨会（TIME'05）。第166–174页。IEEE（2005）13. Faymonville, P., Finkbeiner, B., Schirmer, S., Torfah, H.：用于网络监控的基于流的规范语言。载于：Falcone, Y., Sánchez, C.（编）《运行时验证》。第152–168页。Springer国际出版公司，Cham（2016）14. Faymonville, P., Finkbeiner, B., Schledjewski, M., Schwenger, M., Stenger, M., Tentrup, L., Torfah, H.：Streamlab：信息物理系统的基于流的监控。载于：Dillig, I., Tasiran, S.（编）《计算机辅助验证——第31届国际会议，CAV 2019》，美国纽约市，2019年7月15日至18日，论文集，第一部分。《计算机科学讲义》，第11561卷，第421–431页。施普林格（ 2019 10 1007 978 3 030 25540 4_24 ）。https://doi.org/ . / - - - - . Gorostiaga, F., Sánchez, C.[15]：HLola：用于可扩展流运行时验证的高度函数式工具。载于：Groote, J.F., Larsen, K.G.（编）《系统构造与分析工具和算法——第27届国际会议，TACAS 2021》，作为欧洲软件理论与实践联合会议 ETAPS 2021 的一部分举行，卢森堡卢森堡市，2021年3月27日至4月1日，论文集，第二部分。《计算机科学讲义》，第12652卷，第349–356页。施普林格（ 2021 10 1007 978 3 030 72013 1_18 ）。https://doi.org/ . / - - - - . Gorostiaga, F., Sánchez, C.[16]：HStriver：用于实时事件流运行时验证的高度函数式可扩展工具。载于：Huisman, M., Pasareanu, C.S., Zhan, N.（编）《形式化方法——第24届国际研讨会，FM 2021》，虚拟活动，2021年11月20日至26日，论文集。《计算机科学讲义》

机科学，第 13047 卷，第 563–580 页。施普林格（2021）。https://doi.org/10.1007/978-3-030-90870-6 30_

17. Halbwachs, N., Caspi, P., Raymond, P., Pilaud, D.：同步数据流编程语言 LUSTRE。IEEE 会议录 **79**(9)，1305–1320（1991）。https:// doi.org/10.1109/5.97300

18. Henzinger, T.A., Karimi, M., Kueffner, K., Mallik, K.：监控算法公平性。载于：Enea, C., Lal, A.（编）《计算机辅助验证——第 35 届国际会议，CAV 2023》，法国巴黎，2023 年 7 月 17–22 日，论文集，第二部分。《计算机科学讲义》，第 13965 卷，第 358–382 页。施普林格（2023）。https://doi.org/10.1007/978-3-031-37703-7_17

19. Junges, S., Torfah, H., Seshia, S.A.：马尔可夫决策过程的运行时监视器。载于：Silva, A., Leino, K.R.M.（编），《计算机辅助验证——第 33 届国际会议，CAV 2021》，虚拟活动，2021 年 7 月 20–23 日，论文集，第二部分。《计算机科学讲义》，第 12760 卷，第 553–576 页。施普林格（2021）。https://doi.org/10.1007/978-3-030-81688-9_26

20. Kallwies, H., Leucker, M., Schmitz, M., Schulz, A., Thoma, D., Weiss, A.：Tessla——一个用于运行时验证的生态系统。载于：Dang, T., Stolz, V.（编），《运行时验证——第 22 届国际会议，RV 2022》，第比利斯，格鲁吉亚，2022 年 9 月 28–30 日，论文集。《计算机科学讲义》，第 13498 卷，第 314–324 页。施普林格（2022）。https://doi.org/10.1007/978-3-031-17196-3_20

21. Pedregal, P., Gorostiaga, F., Sánchez, C.：追溯式参数化监控。CoRR **abs/2307.06763**（2023）。https://doi.org/10.48550/ARXIV.2307.06763

22. Robinson, R.M.：一个本质上不可判定的公理系统。载于：国际数学家大会论文集。第 1 卷，第 729–730 页。American Mathematical Society，80 Waterman Street，普罗维登斯，RI（1950）

23. Schwenger, M.：面向信息物理系统的静态分析流监控（2022）。https://doi.org/http://dx.doi.org/10.22028/D291-37014

24. Vickers, S.：《通过逻辑的拓扑学》。剑桥理论计算机科学论丛；5，剑桥大学出版社，剑桥 [等]，1。平装版（1996），http://www.gbv.de/dms/ilmenau/toc/21309293X.PDF;https:// zbmath.org/?q=an:0922.54002，原版出版于：1989。- 包含参考书目kart.；eb

RTLola 表达式的 RTLola 抽象语法树语法

表达式 expr ::=同步(id, p) | 保持(id, p)| Offset(id, p, off) |默认值(expr, expr)| 聚合(id, p, dur, f_a)| 函数(f, expr*) |常量(const)| 参数(p_idx) | 元组(expr*) 常量 const $\in$ $\mathbb{V}$ 流标识符 id ::=id_in| id_out 输入流标识符 id_in $\in$ 标识符$^{\uparrow}$输出流标识符 id_out $\in$ 标识符$^{\downarrow}$参数 p $\in$ expr $\times \cdots \times$ expr 参数索引 p_idx $\in$ $\mathbb{N}$偏移量 off $\in$ $\mathbb{Z}$持续时间 dur $\in$ $\mathbb{R}$函数 f $\in$ $\mathbb{V}_{\perp}^{*} \to \mathbb{V}_{\perp}$聚合函数 f_a $\in (\mathbb{R}, \mathbb{V})^{*} \to \mathbb{V}_{\perp}$

B 关系语义

B.1 表达式的语义推理规则

表达式求值（值）
$$\frac{\text{值} \in \mathbb{V}}{\text{常量}(\text{值}) \Downarrow^{t}_{\omega,inst} \text{值}}$$

表达式求值（语法）
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} \text{实例}' \qquad \omega(\text{标识符})(\text{实例}')(t) = v}{\text{同步}(\text{标识符}, \text{表达式}) \Downarrow^{t}_{\omega,inst} v}$$

表达式求值函数
$$\frac{p_1 \Downarrow^{t}_{\omega,inst} v_1 \cdots p_n \Downarrow^{t}_{\omega,inst} v_n \qquad f(v_1, \ldots\ldots, v_n) = v_{res}}{\text{函数}(f, p_1, \ldots\ldots, p_n) \Downarrow^{t}_{\omega,inst} v_{res}}$$

表达式求值关闭
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} \text{实例}' \qquad \text{前缀} = \text{前缀}(\omega, sid, inst', t) \qquad |\text{前缀}| > off \qquad \text{前缀}[|\text{前缀}| - off] = (t', v)}{\texttt{Offset}(sid, \text{表达式}, off) \Downarrow^{t}_{\omega,inst} v}$$

表达式求值关闭-dft
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} \text{实例}' \qquad \text{前缀} = \text{前缀}(\omega, sid, \text{实例}', t) \qquad |\text{前缀}| \leq off}{\texttt{Offset}(sid, \text{表达式}, off) \Downarrow^{t}_{\omega,inst} \bot}$$

求值-表达式-保持
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} inst' \qquad \text{前缀} = \text{前缀}(\omega, sid, inst', t) \qquad \text{前缀} = \text{前缀}' \cdot (t', v)}{\text{保持}(sid, \text{表达式}) \Downarrow^{t}_{\omega,inst} v}$$

eval-expr-hold-dft
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} \text{实例} \qquad \text{前缀} = \text{前缀}(\omega, sid, \text{实例}', t) \qquad \text{前缀} = \epsilon}{\text{保持}(sid, \text{表达式}) \Downarrow^{t}_{\omega,inst} \bot}$$

无默认值表达式求值
$$\frac{expr \Downarrow^{t}_{\omega,inst} v \qquad v \in \mathbb{V}}{\text{默认值}(\text{表达式}, dft) \Downarrow^{t}_{\omega,inst} v}$$

采用默认值表达式求值
$$\frac{expr \Downarrow^{t}_{\omega,inst} \bot \qquad \text{默认值} \Downarrow^{t}_{\omega,inst} v}{\text{默认值}(\text{表达式}, \text{默认值}) \Downarrow^{t}_{\omega,inst} v}$$

表达式参数访问
$$\frac{iid(p\_idx) = v}{\text{参数}(p\_idx) \Downarrow^{t}_{\omega,inst} v}$$

表达式求值聚合
$$\frac{\text{表达式} \Downarrow^{t}_{\omega,inst} \text{实例}' \qquad \text{前缀} = \text{前缀}(\omega, sid, \text{实例}', t) \qquad window = \text{窗口}(\omega, prefix, \omega(t) - dur) \qquad f_a(window) = v}{\text{聚合}(sid, \text{表达式}, \text{时长} \quad f_a) \Downarrow^{t}_{\omega,inst} v}$$

表达式求值元组
$$\frac{\text{表达式}_1 \Downarrow^{t}_{\omega,inst} v_1 \ldots \text{表达式}_n \Downarrow^{t}_{\omega,inst} v_n \qquad (v_1, \ldots, v_n) = n}{\text{元组}(\text{表达式}_1, \ldots, \text{表达式}_n) \Downarrow^{t}_{\omega,inst} v}$$

B.2 流的语义推理规则

$$\frac{\textit{eval-instance-value AliveSince}(\omega, \textit{sid}, \textit{iid}\ \ t) = \text{开始}_t\text{开始}_t \neq \bot \qquad [\![\text{节奏}]\!]^t_{\omega, start_t} \qquad v \in \mathbb{V} \text{ 当 } \Downarrow^t_{\omega,inst} true \text{ 使用 } \Downarrow^t_{\omega,inst} v\text{输出}(\textit{sid}\ \_, \_, \text{求值}(\textit{pac, when, with})\ \_) \Downarrow^t_{\omega,inst} v}{\textit{eval-instance-false-eval-when AliveSince}(\omega, \textit{sid}, \textit{iid}\ \ t) = \text{开始}_t\text{开始}_t \neq \bot \qquad [\![\text{节奏}]\!]^t_{\omega, start_t} \qquad v \in \mathbb{V} \text{ 当 } \Downarrow^t_{\omega,inst} false\text{输出}(\textit{sid}\ \_, \_, \text{求值}(\textit{pac, when, with})\ \_) \Downarrow^t_{\omega,inst} \bot}$$

$$\frac{\textit{eval-instance-false-pacing AliveSince}(\omega, \textit{sid}, \textit{iid}\ \ t) = \text{开始}_t\text{开始}_t \neq \bot \qquad \neg[\![\text{节奏}]\!]^t_{\omega, start_t}\text{输出}(\textit{sid}\ \_, \_, \text{求值}(\textit{pac, when, with})\ \_) \Downarrow^t_{\omega,inst} \bot}{\textit{eval-instance-not-alive AliveSince}(\omega, \textit{sid}, \textit{iid}\ \ t) = \bot\text{输出}(\textit{sid}\ \_, \_, \text{求值}(\textit{pac, when, with})\ \_) \Downarrow^t_{\omega,inst} \bot}$$

B.3 IsSpawned 与 IsClosed 的定义

定义 8（IsSpawned 与 IsClosed）。给定求值模型 $\omega \in \mathbb{W}$、流标识 $\textit{sid} \in$ 标识符$^{\uparrow} \uplus$ 标识符$^{\downarrow}$、实例标识 $\textit{iid} \in \textit{InstanceID}$ 以及时间戳 $t \in$ 时间，流 $\textit{sid}$ 的实例 $\textit{iid}$ 在 t 被生成，当且仅当

$$\begin{aligned}&\mathit{IsSpawned} : \mathbb{W} \times \mathit{ID}^{\uparrow} \uplus \mathit{ID}^{\downarrow} \times \mathit{InstanceID} \times \mathit{Time} \to \mathbb{B}\\&\mathit{IsSpawned}(\omega, \mathit{sid}, \mathit{iid}, t)\\&\quad := [\![\mathit{SpawnPac}(\mathit{sid})]\!]^{start_t}_{\omega,0} \wedge \mathit{SpawnWhen}(\mathit{sid}) \Downarrow^t_{\omega,\mathit{iid}} \mathit{true}\\&\qquad \wedge \mathit{SpawnWith}(\mathit{sid}) \Downarrow^t_{\omega,\mathit{iid}} \mathit{iid}\end{aligned}$$

且被关闭，当且仅当

$$\begin{aligned}&\mathit{IsClosed} : \mathbb{W} \times \mathit{ID}^{\uparrow} \uplus \mathit{ID}^{\downarrow} \times \mathit{InstanceID} \times \mathit{Time} \times \mathit{Time} \to \mathbb{B}\\&\mathit{IsClosed}(\omega, \mathit{sid}, \mathit{iid}, t, t_{\mathit{iidstart}})\\&\quad := [\![\mathit{ClosePac}(\mathit{sid})]\!]^t_{\omega, t_{\mathit{iidstart}}} \to \mathit{CloseWhen}(\mathit{sid}) \Downarrow^t_{\omega,\mathit{iid}} \mathit{true}\end{aligned}$$

C 类型系统可靠性证明

C.1 可失败语义引理的证明

证明（可失败语义）。通过对 RTLola 规范的语法进行结构归纳来证明。作为基本情况，考虑常量和参数访问：根据上文所作的假设，这些规则不会失败。对于所有其他表达式，我们依据其语义定义进行论证。对于它们的所有子表达式，我们直接应用归纳假设。值类型分析确保保持算子与偏移算子始终配备默认算子，因而永远不会

产生 $\bot$。默认表达式本身永远不会产生 $\bot$。对于函数表达式和聚合表达式，值类型分析强制规定：若该函数可能返回 $\bot$，则它会再次配备一个默认值。对于元组，归纳假设成立。因此，只有同步访问可能求值为 $\bot$，当 $\omega(sid)(inst')(t) = \bot$时。容易看出，只有当某个表达式在不应当产生 $\bot$ 时却产生了它，语义才会失败；而由于同步访问是唯一能够产生 $\bot$ 的表达式，该引理随之成立。

C.2 抽象安全性定理的证明

证明（抽象安全性）。反证法。非 $safe(\varphi)$，意味着存在一个没有一致世界的输入序列。根据引理 1，这是由于某次同步访问产生了 $\bot$。然而，对于所有同步访问，该蕴含式的条件都成立。因此，该条件也必定适用于产生 $\bot$ 的同步访问。这就产生了矛盾，因为该条件要求被访问的流处于活动状态且已产生一个值。

C.3 类型检查安全性定理的证明

证明。该定理可直接由同步访问和输出流的类型推断规则得出。蕴含式左侧的活动谓词的每个合取项，都蕴含右侧活动谓词中对应的合取项。例如，[Sem-Output] 规则要求，流的生成条件的语义类型比为求值子句推断出的语义生成类型更具体（或相等）。应用于同步访问的$Sem\text{-}Sync\text{-}Output$ 规则，推断出的语义生成类型不弱于被访问流的语义生成类型。根据这条推导链以及语义类型格的定义，可以得出，访问流生成子句中的$when$ 表达式必定蕴含被访问流生成子句中的$when$ 表达式。由此可以得出，访问流的$AliveSince$ 谓词求值所得的时间点等于或晚于被访问流的$AliveSince$ 谓词所对应的时间点。对于活动谓词中的其他合取项，也可以用同样的论证得出相应结论。

D RTLola 值类型系统

定义 9（值类型格）。基本值类型集合定义如下：

$$VT^{prim} := \{Bool, String, UInt(x)\ Int(x)\ Float(y)\ \mid x \in \{8,16,32,64\}, y \in \{32,64\}\}$$

这些基本类型都可以用选项包装，以表示值缺失。

$$VT :=VT_{prim} \cup \{\text{选项}（v）\mid v \in VT_{prim}\} \cup \{（v_1 \times ... \times v_n）\mid v_1, ..., v_n \in VT\} \cup \{\top, \bot\}$$

相对于以下方程，关系 $\sqsubseteq_{VT}$ 是最小的：

$$\begin{aligned}
v &\sqsubseteq_{VT} \top && \iff v \in VT \\
\bot &\sqsubseteq_{VT} v && \iff v \in VT \\
Bool &\sqsubseteq_{VT} Bool && \iff true \\
String &\sqsubseteq_{VT} String && \iff true \\
UInt(x_1) &\sqsubseteq_{VT} UInt(x_2) && \iff x_1 \geq x_2 \\
Int(x_1) &\sqsubseteq_{VT} Int(x_2) && \iff x_1 \geq x_2 \\
Float(y_1) &\sqsubseteq_{VT} Float(y_2) && \iff y_1 \geq y_2 \\
Option(v_1) &\sqsubseteq_{VT} Option(v_2) && \iff v_1 \sqsubseteq_{VT} v_2
\end{aligned}$$

请注意，值类型可以轻松扩展为定长数组（如字符串），或固定元数复合类型（如元组或结构体）。为了支持显式类型注释，我们假设给定以下两个函数：

$$\begin{aligned}
\alpha &: \texttt{ID} \to VT \\
\rho &: \texttt{ID} \to VT*
\end{aligned}$$

其中， α 为每个流指定一个值类型（若未注释，则为 $\top$ ），而 ρ为给定流的每个参数指定一个值类型。

$$\frac{}{v \models \texttt{Constant}(x)}\ \text{VALUE-CONSTANT}$$

值同步输出
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输出}(sid,\ p\ \_^{def,\ s,\ e,\ c}) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{v \models \text{同步}(sid, p)}$$

值同步输入
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输入}(sid) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{v \models \text{同步}(sid,\ p)}$$

值偏移输出
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输出}(sid,\ p\ \_^{def,\ s,\ e,\ c}) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{\text{选项}(v) \models \texttt{Offset}(sid,\ p,\ o)}$$

value-offset-in
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输入}(sid) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{\text{选项}(v) \models \texttt{Offset}(sid, p, o)}$$

value-保持-输出
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输出}(sid, p\ \_def, s, e, c) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{Option(v) \models \texttt{Hold}(sid, p)}$$

值-保持-输入
$$\frac{sid \in \text{标识符} \qquad v' \models \text{输入}(sid) \qquad v \sqsubseteq_{VT} v' \qquad \rho(sid) \models p}{\text{选项}(v) \models \text{保持}(sid, p)}$$

值聚合输出
$$\frac{sid \in \text{标识符}^{\downarrow} \qquad v \models \text{输出}(sid, p\ \_def, s, e, c) \qquad f\colon\ v_f \times \ldots \times v_f \to v' \qquad v_f \sqsubseteq_{VT} v \qquad \rho\ (sid)\ \models p}{v' \models \text{聚合}(sid,\ p,\ d,\ f)}$$

值聚合输入
$$\frac{sid \in \text{标识符} \qquad v \models \text{输入}(sid) \qquad f\colon\ v_f \times \ldots \times v_f \to v' \qquad v_f \sqsubseteq_{VT} v \qquad \rho(sid) \models p}{v' \models \texttt{Aggregate}(sid, p, d, f)}$$

值默认
$$\frac{\text{选项}(v_1) \models exp_1 \qquad v_2 \models exp_2 \qquad v \sqsubseteq_{VT} v_1 \qquad v \sqsubseteq_{VT} v_2}{v \models \text{默认值}(exp_1, exp_2)}$$

值参数
$$\frac{\rho(sid)[idx] = v}{v \models_{sid} \text{参数}(idx)}$$

值函数
$$\frac{\begin{array}{c} f\colon\ v_1 \times \ldots \times v_n \to v \\ v_1' \models exp_1 \qquad v_1 \sqsubseteq_{VT} v_1' \qquad \ldots \qquad v_n' \models exp_n \qquad v_n \sqsubseteq_{VT} v_n' \end{array}}{v \models \text{函数}(f, expr_1, \ldots, expr_n)}$$

值元组
$$\frac{v_1 \models expr_1 \qquad \ldots \qquad v_n \models expr_n}{(v_1 \times \ldots \times v_n) \models \text{元组}(expr_1, \ldots, expr_n)}$$

图 4：RTLola 值类型推断规则

$$\frac{\text{值-输出} s = \text{生成}(pt_s, e_s^f, e_s) v_s \models \text{生成}(pt, e_s^f, e_s)\ \rho(sid) \sqsubseteq_{VT} v_s \qquad e = \text{求值}(pt_e, e_e^f, e_e)\ v_e \models \text{求值}(pt_e, e_e^f, e_e)\ v \sqsubseteq_{VT} v_e \qquad c = \text{关闭}(pt_c, e_c^f)\ ()\ \models \text{关闭}(pt_c, e_c^f) v \models \text{输出}(sid, p\_def, s, e, c)\ \text{值-输入}\ \text{值-生成}\ \text{值-求值} \alpha(sid) = v\ \text{布尔值}\ \models e_s^f \qquad v \models e_s\ \text{布尔值}}{\models e_e^f \qquad v \models e_e v \models \text{输入}(sid)\ v \models \text{生成}(pt_s, e_s^f, e_s)\ v \models \text{求值}(pt_e, e_e^f, e_e)\ \text{值-关闭}\ \text{布尔值} \models e_c^f()\ \models \text{关闭}(pt_c, e_c^f)}$$

图 5：流的值类型规则

E RTLola 节奏类型系统

$$\frac{}{(\top, (Event(id), \top) \models \texttt{Input(}sid\texttt{)}}\ \text{PACING-INPUT}$$

$$\frac{\text{节奏-输出}\ (\top, \tau_e^s, \top)\ \models \text{生成}(\tau_s', e_s^f, e_s)\ (\tau_s^e, \tau_e^e, \tau_c^e)\ \models \text{求值}(\tau_e', e_e^f, e_e)\ (\tau_s^c, \tau_e^c, \tau_c^c)\ \models \text{关闭}(\tau_c', e_c^f) \tau_s^c \in \{\top, \tau_e^s\} \qquad \tau_c^c \in \{\top, \tau_e^c\} \qquad \tau_e^e = \mathit{LocalFreq}(x)\ \rightarrow \tau_e^s = \tau_s^e \wedge \tau_e^c = \tau_c^e\ \tau_e^c = \mathit{LocalFreq}(x)\ \rightarrow \tau_e^s = \tau_s^c \wedge \tau_e^c = \tau_c^c \qquad \tau_e^s \sqsubseteq_{PT} \tau_s^e \qquad \tau_c^e \sqsubseteq_{PT} \tau_e^c (\tau_e^s, \tau_e^e, \tau_e^c)\ \models \text{输出}(sid, p\_def, \text{生成}(\tau_s', e_s^f, e_s), \text{求值}(\tau_e', e_e^f, e_e), \text{关闭}(\tau_c', e_c^f))\ \text{节奏-生成}}{\mathcal{T}_f \models e_s^f \qquad \mathcal{T}' \sqsubseteq_{PT} \mathcal{T}_f \qquad \mathcal{T}_e \models e_s \qquad \mathcal{T}' \sqsubseteq_{PT} \mathcal{T}_e \qquad \tau_e \sqsubseteq_{PT} \tau_e' (\tau_s', \tau_e, \tau_c')\ \models \text{生成}(\tau_e, e_s^f, e_s)\ \text{节奏-求值}}$$

$$\frac{\mathcal{T}_f \models e_e^f \qquad \mathcal{T}' \sqsubseteq_{PT} \mathcal{T}_f \qquad \mathcal{T}_e \models e_e \qquad \mathcal{T}' \sqsubseteq_{PT} \mathcal{T}_e \qquad \tau_e \sqsubseteq_{PT} \tau_e' (\tau_s', \tau_e, \tau_c')\ \models \text{求值}(\tau_e, e_e^f, e_e)\ \text{节奏-关闭} \mathcal{T}_f \models e_c^f \qquad \mathcal{T}' \sqsubseteq_{PT} \mathcal{T}_f \qquad \tau_e' \sqsubseteq_{PT} \tau_e (\tau_s', \tau_e, \tau_c')\ \models \text{关闭}(\tau_e, e_c^f)}{}$$

图 6：流的步调类型分析推理规则。

$$\frac{}{(\top,\top,\top) \models \text{常量}(c)}\ \text{节奏常量} \qquad \frac{\mathcal{T} \models \text{输入}(sid)}{\mathcal{T} \models \text{同步}(sid, p)}\ \text{节奏同步输入}$$

$$\frac{\mathcal{T} \sqsubseteq_{PT} \mathcal{T}' \quad \mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_{n_p} \models e_{n_p} \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_{n_p} \quad \mathcal{T}' \models \text{输出}(sid, n_p, s, e, c)}{\mathcal{T} \models \text{同步}(sid, e_1, \ldots, e_{n_p})}\ \text{节奏同步输出}$$

$$\frac{\mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_{n_p} \models e_{n_p} \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_{n_p}}{\mathcal{T} \models \text{保持}(sid, e_1, \ldots, e_{n_p})}\ \text{节奏保持}$$

$$\frac{\mathcal{T} \models \text{输入}(sid)}{\mathcal{T} \models \texttt{Offset}(sid, p, o)}\ \text{步调偏移输入}$$

$$\frac{\mathcal{T} \sqsubseteq_{PT} \mathcal{T}' \quad \mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_{n_p} \models e_{n_p} \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_{n_p} \quad \mathcal{T}' \models \text{输出}(sid, n_p, s, e, c)}{\mathcal{T} \models \texttt{Offset}(sid, e_1, \ldots, e_{n_p}, o)}\ \text{步调偏移输出}$$

$$\frac{\mathcal{T}_1 \models e_1 \quad \mathcal{T}_2 \models e_2 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_2}{\mathcal{T} \models \texttt{Default}(e_1, e_2)}\ \text{PACING-DEFAULT} \qquad \frac{}{(\top,\top,\top) \models \texttt{Parameter}(i)}\ \text{PACING-PARAMETER}$$

$$\frac{\mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_n \models e_n \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_n}{\mathcal{T} \models \texttt{Function}(f, e_1, \ldots e_n)}\ \text{PACING-FUNCTION}$$

$$\frac{\mathcal{T} \sqsubseteq_{PT} \mathit{Periodic} \quad \mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_{n_p} \models e_{n_p} \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_{n_p}}{\mathcal{T} \models \texttt{Aggregate}(sid, e_1, \ldots, e_{n_p}, d, f)}\ \text{PACING-AGGREGATION}$$

$$\frac{\mathcal{T}_1 \models e_1 \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_1 \quad \ldots \quad \mathcal{T}_{n_p} \models e_n \quad \mathcal{T} \sqsubseteq_{PT} \mathcal{T}_n}{\mathcal{T} \models \texttt{Tuple}(e_1, \ldots, e_n)}\ \text{PACING-TUPLE}$$

图 7：表达式步调类型分析的推理规则。

F RTLola 语义类型系统

语义常量

$$\frac{}{(\top,\ \top,\ \bot) \models \text{常量}(v)}$$

语义同步输入

$$\frac{sid \in \text{标识符}^{\uparrow}}{(\top,\ \top,\ \bot) \models \text{同步}(sid,\ p)}$$

语义同步输出

$$\frac{\begin{array}{c}\text{输出}(sid,\ n_p,\ s,\ e,\ c) \in \text{输出} \\ \text{输出}(sid',\ n_p',\ s',\ e',\ c') \in \text{输出} \qquad \forall 1 \le k \le n.\ e_k = \text{参数}(i) \wedge k \cong^{sid}_{sid'} i \\ \Pi \models \text{输出}(sid,\ n_p,\ s,\ e,\ c) \qquad \Pi' \sqsubseteq_{ST} \Pi\end{array}}{\Pi' \models_{sid'} \text{同步}(sid,\ (e_1,\ \ldots\ldots,\ e_n))}$$

语义-Offset-输入

$$\frac{sid \in \text{标识符}^{\uparrow}}{(\top,\top,\bot) \models \texttt{Offset}(sid,p,o)}$$

语义-聚合

$$\frac{\Pi^1 \models e_1 \qquad \Pi \sqsubseteq_{ST} \Pi^1 \qquad \ldots \qquad \Pi^n \models e_n \qquad \Pi \sqsubseteq_{ST} \Pi^n}{\Pi \models \text{聚合}(id,\ e_1,\ \ldots\ldots,\ e_n,\ d,\ f)}$$

语义-Offset-输出

$$\frac{\begin{array}{c}\text{输出}(sid,\ n_p,\ s,\ e,\ c) \in \text{输出} \qquad \text{输出}(sid',\ n_p',\ s',\ e',\ c') \in \text{输出} \\ \forall 1 \le k \le n.\ e_k = \text{参数}(i) \wedge \exists 1 \le l \le n_p.\ i \cong^{sid}_{sid'} l \\ \Pi \models \text{输出}(sid,\ n_p,\ s,\ e,\ c) \qquad \Pi' \sqsubseteq_{ST} \Pi\end{array}}{\Pi' \models_{sid'} \texttt{Offset}(sid,p,o)}$$

语义保持

$$\frac{\Pi^1 \models e_1 \qquad \Pi \sqsubseteq_{ST} \Pi^1 \qquad \ldots \qquad \Pi^n \models e_n \qquad \Pi \sqsubseteq_{ST} \Pi^n}{\Pi \models \text{保持}(id,\ e_1,\ \ldots, e_n)}$$

语义参数访问

$$\frac{}{(\top,\ \top,\ \bot) \models \text{参数}(i)}$$

语义默认值

$$\frac{\Pi^1 \models e_1 \qquad \Pi^2 \models e_2 \qquad \Pi \sqsubseteq_{ST} \Pi^1 \qquad \Pi \sqsubseteq_{ST} \Pi^2}{\Pi \models \text{默认值}(e_1,\ e_2)}$$

语义函数

$$\frac{\Pi^1 \models e_1 \qquad \Pi \sqsubseteq_{ST} \Pi^1 \qquad \ldots \qquad \Pi^n \models e_n \qquad \Pi \sqsubseteq_{ST} \Pi^n}{\Pi \models \text{函数}(f,\ e_1,\ \ldots\ldots e_n)}$$

语义元组

$$\frac{\Pi^1 \models e_1 \qquad \Pi \sqsubseteq_{ST} \Pi^1 \qquad \ldots \qquad \Pi^n \models e_n \qquad \Pi \sqsubseteq_{ST} \Pi^n}{\Pi \models \text{元组}(e_1,\ldots, e_n)}$$

图 8：表达式语义类型分析的推理规则。

语义输出 $(\top, \pi_e^s, \bot) \models_{sid}$ 生成$(p_s, e_s^f, e_s)\Pi^e \models_{sid}$ 求值(τ_e, e_e^f, e_e) $(\pi_s^c, \pi_e^c, \pi_c^c) \models_{sid}$ 关闭$(\tau_c, e_c^f)\pi_s^c \in \{\top, \pi_e^s\}$ $\pi_c^c \in \{\bot, \pi_e^c\}$ $\tau_e = LocalFreq(x) \rightarrow \pi_e^s = \pi_s^e \wedge \pi_e^c \wedge \pi_c^e$ $\tau_c = LocalFreq(x) \rightarrow \pi_e^s = \pi_s^c \wedge \pi_e^c = \pi_c^c$ $\pi_e^s \sqsubseteq_{ST} \pi_s^e$ $\pi_c^e \sqsubseteq_{ST} \pi_e^c(\pi_e^s, \pi_e^e, \pi_e^c) \models$ 输出$(sid, p_def,$ 生成$(p_s, e_s^f, e_s),$ 求值$(p_e, e_e^f, e_e),$ 关闭$(p_c, e_c^f))$ 语义输入 $(\top, \top, \bot) \models$ 输入(sid) 语义生成$\Pi_f \models_{sid} e_s^f$ $\Pi' \models_{sid} e_s$ $\Pi \sqsubseteq_{ST} \Pi_f$ $\Pi \sqsubseteq_{ST} \Pi'$ $Expr$ $(e_s^f) \sqsubseteq_{ST} \pi_e(\pi_s, Expr(e_s^f)\ \pi_c) \models_{sid}$ 生成(p_def, e_s^f, e_s) 语义求值 $\Pi_f \models_{sid} e_e^f$ $\Pi' \models_{sid} e_e$ $\underline{\Pi \sqsubseteq_{ST} \Pi_f \quad \Pi \sqsubseteq_{ST} \Pi'}$ $Expr(e_e^f) \sqsubseteq_{ST} \pi_e(\pi_s,$ $Expr(e_e^f)\ \pi_c) \models_{sid}$ 求值(p_def, e_e^f, e_e) 语义关闭$\Pi \models_{sid} e_c^f$ $\pi_c \sqsubseteq_{ST} Expr(e_c^f)$ $(\pi_s,$ $Expr(e_c^f)\ \pi_c) \models_{sid}$ 关闭(p_def, e_c^f)

图 9：流的语义类型分析的推理规则。

G 基准测试规范

G.1 入侵者

以下规范监控一架无人机到多架其他无人机的距离。它是 [6] 中所呈现的第 1.1节简化规范的原始版本。

```
1 导入 math 2 输入 纬度：浮点数 3 输入 经度：浮点数 4 输入 入侵者_标识符：无符号整数 5 输入 入侵者_纬度：浮点数 6 输入 入侵者_经度：浮点数 7 8 输出 入侵者_位置(标识符) 9 生成，使用 入侵者_标识符 10 求值，当 标识符 = 入侵者_标识符，使用 (入侵者_纬度, 入侵者_经度) 11 关闭@true，当 过期(标识符).保持(or: false) 12 输出距离(标识符) 13 生成，使用 入侵者_标识符 14 求值 @((入侵者_标识符 && 入侵者_纬度 && 入侵者 _ 经度 ) || ( 纬度 && 经度 )) 15 使用 sqrt(( 入侵者 _ 位置 ( 标识符 ). 保持 ().0. 默认值 (to: 0.0) - 纬度 .保持(or: 0.0))**2.0 + (入侵者_位置(标识符).保持().1.默认值(to: 0.0) - 经度.保持(or: 0.0))**2.0) 16 关闭@true，当 过期(标识符).保持(or: false) 17 输出更近(标识符) 18 生成，使用 入侵者_标识符 19 求值，使用 距离(标识符).offset(by: -1).默认值(to: 距离(标识符)) >= 距离(标识符)
```

20 关闭@true，当 过期(标识符).保持(or: false) 21 输出过期(标识符) 22 生成，使用 入侵者_标识符 23 10 _ 10 = 0 求值 @ s, 24 使用 入侵者位置(标识符).聚合(over: s, using: count) 关闭@true，当 过期(标识符).保持(or: false) 25 26 触发器(标识符) 27 生成，当 距离(入侵者_标识符).保持(or: 1.0) < 0.1，使用 入侵者_标识符 28 求值 @1Hz，当 更近(标识符).聚合(over_exactly: 5s, using: forall).默认值(to: false)，使用 "检测到入侵者 {{}}".format(标识符) 29 关闭@true，当 过期(标识符).保持(or: false)

G.2 航点任务

本示例展示了如何在 RTLola 中实现包含任意多个航路点的航点任务。为此，流以航路点的 X 坐标和 Y 坐标进行参数化，从而使单个流实例负责单个航点。

1 输入 pos: (浮点数, 浮点数) 2 输入 航点: (浮点数, 浮点数) 3 4 输出 航点_距离(wx: 浮点数, wy: 浮点数) 5 以 (航点.0, 航点.1) 生成 6 以 sqrt((pos.0 - wx)**2.0 + (pos.1 - wy)**2.0) 求值 7 当 航点_已到达(wx, wy) 时关闭 8 9 输出 航点_正在接近(wx:浮点数, wy: 浮点数) 10 以 (航点.0, 航点.1) 生成 11 以 航点_距离(wx, wy) <= 航点_距离(wx, wy).Offset(按: -1).默认值(为: 0.0) 求值 12 当 航点_已到达(wx, wy) 时关闭 13 14 输出 航点_已到达(wx: 浮点数, wy: 浮点数) 15 以 (航点.0, 航点.1) 生成 16 以 航点_距离(wx, wy) < 5.0 求值 17 当 航点_已到达(wx, wy) 时关闭

假设输入流 pos 包含无人机的当前位置，以 (x, y) 坐标表示；输入流 waypoint 则包含待监控的下一个航点的 (x, y) 坐标。每个输出流都通过相同的生成子句，以这些航点坐标为参数。到达相应航点后，实例便会关闭；这是通过关闭子句中的 waypoint_reached 输出流访问实现的。为此，waypoint_distance 流会计算无人机当前位置与航点之间的欧几里得距离。当该距离低于某个阈值时，waypoint_reached 输出流的相应实例就会求值为 true。此外，waypoint_approaching 流还会针对每个航点，通过时间偏移监控无人机是否正逐渐接近航点。具体而言，它会比较无人机当前与航点之间的距离和上一步的距离，并断言后者更大。对前一个值的访问通过偏移访问实现，并附带一个默认值，因为在监控器执行的第一步中不存在前一个值。

G.3 看门狗

该示例实现了一个监控器，用于监督飞机内电子控制单元（ECU）的响应性。

```
1 输入 ping：整数型 2 输入 pong：整数型 3 4 输出 pong_of_node(node：整数型) 5 使用
pong 生成 6 使用 pong  =node 求值 7 8 输出 is_alive(node：整数型 ) 9 使用 ping 生成 10 使
用 pong_of_node(node) 求值 @1min.aggregate(over: 1min, using: exists) 11 当 is_
alive(node)  =  is_alive(node) 时关闭
```

两个输入流 `ping` 和 `pong` 的值表示 ECU 的数字标识符。如果在 `ping` 流上收到一个值，就表示已向该 ECU 发出存活请求，需要由通过 `pong`流实现的本单元进行响应。参数化输出流 `pong_of_node` 会按 ECU 对 pong 响应进行分组，为每个 ECU 生成一个新实例，并且只有收到来自同一 ECU 的 pong 时才会求值。`is_alive` 输出流为每个 ping 请求实现一个看门狗。该看门狗会检查是否在规定时间内收到响应。同样，我们需要一个随每个 ping 生成的参数化输出流。每个实例从创建时起，一分钟后进行求值。求值会聚合本单元收到的 pong 响应，以检查该单元是否在最近一分钟内作出了响应。最后，关闭声明会在首次求值后从内存中删除该实例，以便为来自该 ECU 的下一次 ping 做好监控准备。

G.4 RCC

示例取自 [5]，用于监控无人飞机的遥控单元。

```
1 输入 seq_number：64 位整数 2 3 /// 辅助流 4 输出 lost_connection_to_master @true：=
false 5 输出 switch_to_secondary @true：=false 6 输出 both_rc_disconnected @true：=f
alse 7 8 /// 性质 1：日志消息递增 9 输出 valid_seq_number：=seq_number  =seq_
number.offset(by: -1, or: -1)  +  1 10 /// 性质 7：遥控器回退测试 11 outputmain_fallback_vali
d_dyn 12 生成when lost_connection_to_master 13 关闭when switch_to_secondary  ||
both_rc_disconnected 14 求值 @200ms with false 15 outputmain_fallback_valid @true：=
main_fallback_valid_dyn.hold(or: true)
```

G.5 FFD

此示例取自 [5]，用于监控无人驾驶航空器的飞行阶段。

```
1 import math 2 3 常量 ROTOR_1: UInt8 := 14 常量 ROTOR_2: UInt8 := 25 常量 EPSILON_RPM_
ON : Float64 := 1.0 6 常量 EPSILON_RPM : Float64 := 0.5 7 8 9 输入 rpm : Int64 10 输入 src : UInt8 11
12 输出 rpm_1 求值 when src == ROTOR_1 with abs(cast<Int64, Float64>(rpm)) 13 输出 rpm_2 求值
when src == ROTOR_2 with abs(cast<Int64, Float64>(rpm)) 14 15 输出 rpm_on_check @rpm :=
ifrpm_1.hold(or: 0.0) + rpm_2.hold(or: 0.0) / 2.0 > EPSILON
                    _   _          1 0       0 0 RPM ON then . else . 输出 rpm_on @ s : rpm_on_
16                      1   =
check.aggregate(over: 1s, using: avg).defaults(to: 0.0) >EPSILON_RPM 17 输出 take_off @1s := false
18 输出 已着陆 @1s := false 19 输出 rpm_in_air @1s := false 20 21 22 输出 phase_1 := !take_off &&!
landed && rpm_on && !rpm_in_air
```

G.6 地理围栏

此示例展示了一个由四个点构成的地理围栏多边形，自主飞机不得离开该区域。

```
1 import math 2 /// 位置 3 输入 gps__纬度 :Float64 4 输入 gps__经度 :Float64 5 输入 gps__高度 :
Float64 6 /// 速度 7 输入 gps__速度_x :Float64 8 输入 gps__速度_y :Float64 9 输入 gps__速度_z :Float64
10 /// 精度检查 11 输入 gps__水平_精度 :Float64 12 输入 gps__垂直_精度 :Float64 13 14 ///
****************/// 15 ///****地理围栏****/// 16 ///****************///
17              _                64  = 0 0000001 常量 c epsilon : Float : . 常量 close
18                                   64  = 3
                    _   _                        00000000000000000000 to 地理围栏 : Float : . 常量
19 min_time_to : Float : . 20 常量 inside_g1: Bool :=true 21 常量 inside_g2: Bool :=true 22 常量 inside_g3:
Bool :=true 23 24 // 计算飞行器轨迹
25              _   _                  _                    --          3 14159265359   180 0
26                                                                       3 14159265359   180 0
输出lat in rad 求值 when GPS条件 with gps latitude * . / . 输出lon_in_rad 求值 when gps_条件 with
gps__longitude * . / . 27 输出filtered_height 求值 when gps_条件 with gps__height 28 输出velocity_xy
求值 when gps_条件 with sqrt(gps__speed_x**2.0 + gps__speed_y**2.0) 29 输出initial_height 求值
when gps_条件 with initial_height.offset(by: -1).defaults(to: gps__height) 30 输出x 求值 when gps_条件
with lat_in_rad 31 输出 y 求值 when gps_ 条件 with lon_in_rad 32 输出 delta_x 求值 when gps_ 条件
with x - x.last(or: x) 33 输出delta_y 求值 when gps_条件 with y - y.last(or: x)
```

类型安全的参数化流监控 33

34 输出 isFnc 求值 when gps_条件 with x != x.last(or: x) 35 输出 梯度 求值 when gps_条件 with if isFnc then delta_y / delta_x else 0.0 36 输出 y_截距 求值 when gps_条件 with y - (梯度 * x) 37 输出 dstToPnt 求值 when gps_条件 with sqrt(delta_x**2.0 + delta_y**2.0) 38 输出 方向_x 求值 when gps_条件 with if abs(delta_x) < c_epsilon then false else delta_x < 0.0 39 输出 方向_y 求值 when gps_条件 with if abs(delta_y) < c_epsilon then false else delta_y < 0.0 40 41 常量 height_违规 : Bool :=false 42 43 ///******************/// 44 ///****条件****/// 45 ///******************/// 46 输出 condition_horizontal_accuracy := ((gps_._horizontal_accuracy) < 0.01) 47 输出 condition_vertical_accuracy := ((gps_._vertical_accuracy) < 0.01) 48 输出 gps_条件 := condition_horizontal_accuracy && condition_vertical_accuracy 49 50 // 多边形折线 g0 89655156587213, 52 76 07671638200421, 17 139048054333795 73 81763088322348: (. .) to (. .)

```
51                      _ 1_0        64  =  0 06110570104726112    常量 gradient g :Float : - . 常量 y
52                        1_0        64  = 21 787769142229816
```

intercept g :Float : . 53 输出 g1_0_is_intersecting 求值 when gps_条件 with gradient != gradient_g1_0 or y_intercept =y_intercept_g1_054 输出 g1_0_x 求值 when gps_条件 with if isFnc and g1_0_is_intersecting then (y_intercept - y_intercept_g1_0) /(gradient_g1_0 - gradient) else x

```
55          1_0_                  _                        _ 1_0     1_0_  +   _           _ 1_0  输出 g y
56          1_0_                                        1_0_     180 0    3                14159265359
```

eval when gps condition with gradient g * g x y intercept g output g x deg eval when gps condition with g x * . /

```
.57         1_0_ _                _                   1_0_     180 0    3                output g y deg
```

eval when gps condition with g y * . / .

```
                                                                                  14159265359
58          1_0_                                 _                          1_0_            output g
```

orientation eval when gps condition with ((if abs(g x - x.last(or: x)) < c_epsilon then false else g1_0_x - x.last(or: x) < 0.0) = 方向_x) and ((if abs(g1_0_y - y.last(or: y)) < c_epsilon then false else g1_0_y - y.last(or: y) < 0.0) = 方向_y) 59 输出 g1_0_violated_cnd 求值 when gps_条件 with (!isFnc or g1_0_is_intersecting) and g 1_0_ 1_0_ < 73 89655156587213 1_0_ > orientation and 17 139048054333795 1_0_ < 52 81763088322348 1_0_ > 17 (g x . and g x .)and (g y . and g y .139048054333795) 60 输出 g1_0_is_violated 求值 when gps_条件 with g1_0_violated_cnd and (sqrt((x.last(or: x) - g1_0_x)**2.0 + (y.last(or: y) - g1_0_y)**2.0) <= dstToPnt) 61 输出 g1_0_距离_to 求值 when gps_条件 with if g1_0_violated_cnd then 6371.0 * 2.0 * arcsin(sqrt((sin((x - g1_0_x) / 2.0) ** 2.0) + cos(x) * cos(g1_0_x) * (sin((y - g1_0_y) / 2.0) ** 2.0))) * 1000.0 else 1000.0 62 输出 g1_0_time_to 求值 when gps_条件 with g1_0_距离_to / 速度_xy 63 触发器 if gps_条件 then g1_0_is_violated.hold(or: false) else false " 违规: g1_0!" 64 65 // 多边形折线 g1 16368257693344, 76 56 286685329564456, 37 16234411824784 55 79656929555932: (. .) to (. .)

```
66                        _ 1_1        64  =  0 02833416693796948    常量
67                          1_1        64  = 38 757180456760 67
```

gradient g :Float : - . 常量 y intercept g :Float : . 68 输出 g1_1_is_intersecting 求值 when gps_条件 with gradient != gradient_g1_1 or y_intercept =y_intercept_g1_169 输出 g1_1_x 求值 when gps_条件 with if isFnc and g1_1_is_intersecting then (y_intercept - y_intercept_g1_1) /(gradient_g1_1 - gradient) else x

```
70          1_1_                  _                        _ 1_1     1_1_  +   _           _ 1_1  output g
71          1_1_                                        1_1_     180 0    3                14159265359
```

y eval when gps condition with gradient g * g x y intercept g output g x deg eval when gps condition with g x * . /

```
.72         1_1_ _                _                   1_1_     180 0    3                output g y
```

deg eval when gps condition with g y * . / .

```
                                                                                  14159265359
73          1_1_                                 _                          1_1_            output g
```

orientation eval when gps condition with ((if abs(g x - x.last(or: x)) < c_epsilon then false else g1_1_x - x.last(or: x) < 0.0) = 方向_x) and ((if abs(g1_1_y - y.last(or: y)) < c_epsilon then false else g1_1_y - y.last(or: y) < 0.0) = 方向_y) 74 输出 g1_1_violated_cnd 求值 when gps_条件 with (!isFnc or g1_1_is_intersecting) and g 1_1_ 1_1_ < 55 16368257693344 1_1_ > orientation and 37 16234411824784 1_1_ < 76 79656929555932 1_1_ > 37 (g x . and g x .)and (g y . and g y .16234411824784) 75 输出 g1_1_is_violated 求值 when gps_条件 with g1_1_violated_cnd and (sqrt((x.last(or: x) - g1_1_x)**2.0 + (y.last(or: y) - g1_1_y)**2.0) <= dstToPnt) 76 输出 g1_1_距离_to 求值 when gps_条件 with if g1_1_violated_cnd then 6371.0 * 2.0 * arcsin(sqrt((sin((x - g1_1_x) / 2.0) ** 2.0) + cos(x) * cos(g1_1_x) * (sin((y - g1_1_y) / 2.0) ** 2.0))) * 1000.0 else 1000.0

77 输出 g1_1_时间_to 求值，当全球定位系统_条件满足且 g1_1_距离_to / 速度_xy 时 78 如果全球定位系统_条件满足，则触发器：若 g1_1_is_违规则保持（或：false），否则 false "违规：g1_1!" 79 80 // 多边形折线 g2 602330928870806, 44 87 35977750932717, 64 42663705638026 60 93835038321772: (. .) 至 (. .)

```
81                      _ 1_2       64  = 1 373001486954943 常量 梯度 g :浮点数 : . 常量 y 截距 g :浮点数 : - .
82                        _ 1_2       64  =  55
```

518467363978933 83 输出 g1_2_is_相交 求值，当全球定位系统_条件满足且 梯度 != 梯度_g1_2 或 y_截距 =y_截距_g1_284 输出 g1_2_X 坐标 求值，当全球定位系统_条件满足且若 isFnc 且 g1_2_is_相交，则 (y_截距 - y_截距_g1_2) /(梯度_g1_2 - 梯度)，否则为 X 坐标

```
85          1_2_                _                       _ 1_2    1_2_  +  _          _ 1_2 输出 g y 求
86          1_2                                      1_2_    180 0   3
```

值，当全球定位系统条件满足且 梯度 g * g x̄ y 截距 g 输出 g x deg 求值，当全球定位系统条件满足且 g x * . / .

```
87          1_2_ _                  _                1_2_    180 0   3              输出 g y deg 求值，
```

当全球定位系统条件满足且 g y * . / .

```
88          1_2_                            _                        1_2_          输出 g 方向 求值，
```

当全球定位系统条件满足且 ((if abs(g x - x.last(or: x)) < c_epsilon then false else g1_2_x - x.last(or: x) < 0.0) = 方向_x) and ((if abs(g1_2_y - y.last(or: y)) < c_epsilon then false else g1_2_y - y.last(or: y) < 0.0) = 方向_y) 89 输出 g1_2_违规_cnd 求值，当全球定位系统_条件满足且 (!isFnc or g1_2_is_相交) and g

```
1_2_                  1_2_  < 60 602330928870806      1_2_  > 方向 and (g x . and g x . )and (g y .
```

64 42663705638026 1_2_ < 44 93835038321772 1_2_ > 64 and g y .42663705638026) 90 输出 g1_2_is_违规 求值，当全球定位系统_条件满足且 g1_2_违规_cnd and (sqrt((x.last(or: x) - g1_2_x)**2.0 + (y.last(or: y) - g1_2_y)**2.0) <= dstToPnt) 91 输出 g1_2_距离_to 求值，当全球定位系统_条件满足且若 g1_2_违规_cnd，则 6371.0 * 2.0 * arcsin(sqrt((sin((x - g1_2_x) / 2.0) ** 2.0) + cos(x) * cos(g1_2_x) * (sin((y - g1_2_y) / 2.0) ** 2.0))) * 1000.0，否则为 1000.0 92 输出 g1_2_时间_to 求值，当全球定位系统_条件满足且 g1_2_距离_to / 速度_xy 93 如果全球定位系统_条件满足，则触发器：若 g1_2_is_违规则保持（或：false），否则 false "违规：g1_2!" 94 95 // 多边形折线 g3 335291068837705, 6 0 8691110136063163, 30 889711873136804 39 808748494096017: (. .) 至 (. .)

```
96                      _ 1_3       64  =  1 597368819916523  常量 梯度
97                        _ 1_3       64  = 32 27800270731757 6
```

g :浮点数 : - . 常量 y 截距 g :浮点数 : . 98 输出 g1_3_is_相交 求值，当全球定位系统_条件满足且 梯度 != 梯度_g1_3 或 y_截距 =y_截距_g1_399 输出 g1_3_X 坐标 求值，当全球定位系统_条件满足且若 isFnc 且 g1_3_is_相交，则 (y_截距 - y_截距_g1_3) /(梯度_g1_3 - 梯度)，否则为 X 坐标

```
100         1_3_                _                       _ 1_3    1_3_  +  _          _ 1_3 输出 g y
101         1_3                                      1_3_    180 0   3
```

求值，当全球定位系统条件满足且 梯度 g * g x y 截距 g 输出 g x deg 求值，当全球定位系统条件满足且 g x * . / .

```
102         1_3_ _                  _                1_3_    180 0   3              输出 g y deg 求值，
```

当全球定位系统条件满足且 g y * . / .

```
103         1_3_                            _                        1_3_          输出 g 方向 求值，
```

当全球定位系统条件满足且 ((if abs(g x - x.last(or: x)) < c_epsilon then false else g1_3_x - x.last(or: x) < 0.0) =方向_x) and ((if abs(g1_3_y - y.last(or: y)) <c_epsilon then false else g1_3_y - y.last(or: y) < 0.0) = 方向_y) 104 输出 g1_3_违规 _cnd 求值，当全球定位系统 _ 条件满足且 (!isFnc org1_3_is_ 相交) and g

```
1_3_                  1_3_  < 39 335291068837705      1_3_  > 方向 and (g x . and g x . )and (g y .
```

30 889711873136804 1_3_ < 6 808748494096017 1_3_ > 30 and g y .889711873136804) 105 输出 g1_3_is_违规 求值，当全球定位系统_条件满足且 g1_3_违规_cnd and (sqrt((x.last(or: x) - g1_3_x)**2.0 + (y.last(or: y) - g1_3_y)**2.0) <= dstToPnt) 106 输出 g1_3_距离_to 求值，当全球定位系统_条件满足且若 g1_3_违规_cnd，则 6371.0 * 2.0 * arcsin(sqrt((sin((x - g1_3_x) / 2.0) ** 2.0) + cos(x) * cos(g1_3_x) * (sin((y - g1_3_y) / 2.0) ** 2.0))) * 1000.0，否则为 1000.0 107 输出 g1_3_时间_to 求值，当全球定位系统_条件满足且 g1_3_距离_to / 速度_xy 108 如果全球定位系统_条件满足，则触发器：若 g1_3_is_违规则保持（或：false），否则 false "违规：g1_3!"

G.7 人工基准测试

```
输入 i1: 布尔值
输入 i2: 布尔值
输入 i3: 布尔值
输出 s1
    在 i1 && i2 && i3 时求值, 使用 s2
输出 s2
    当 i1 && i2 时求值, 使用 s3
输出 s3
    当 i1 时求值, 使用 i1
```

(a) 合取项基准测试

```
输入 bench: 整数型
输出s1(p1:整数型,p2: 整数型,p3:整数型)
    生成并使用(bench,bench, bench)
    使用以下内容求值: s2(p1, p2)
输出s2(p1:整数型,p2: 整数型)
    生成并使用 (bench, bench)
    使用以下内容求值: s3(p1)
输出 s3(p1: 整数型)
    生成, 使用 (bench)
求值, 使用 bench
```

(b) 参数基准测试

```
输入 bench:整数型
输出 s1 := s2
输出 s2: = s3
输出 s3: = 基准测试
```

(c) 同步访问基准测试

图 10：带有 $n = 3$ 的基准测试规范示例