arXiv:2603.02405v1 [cs.PL] 2 Mar 2026

# 高度增量式：一种面向多种目标的简单程序化方法（扩展版）

Philipp Schr¨oer[1]和 Joost-Pieter Katoen[1]

德国亚琛工业大学，phisch@cs.rwth-aachen.de，
katoen@cs.rwth-aachen.de

**摘要**。我们提出一种通用的程序化方法，用于推理概率程序中的各种目标。首先，在语言中加入奖励语句。然后，我们定义一种程序变换，对程序的累积奖励应用单调函数。其核心思想是，该变换利用奖励的增量差分。这种简单而优雅的方法能够表达例如高阶矩、奖励的阈值概率、超出预算的期望值以及矩生成函数等目标。现在，所有这些目标都可以通过一种现有方法进行分析：概率 wp 推理。我们使用 Caesar 演绎验证器实现了自动化验证，并报告了该变换在若干示例中的应用。

## 1 引言

概率程序。 概率程序将循环和条件语句等常规编程结构与概率选择结合起来。它们是功能强大的建模工具，可用于各种应用，包括随机算法、安全机制和概率数据库。由于其内在的随机性，概率程序的运行时间服从某种概率分布。图 1a 展示了一个 Web 服务器不断重试数据库调用直至成功的过程。变量 done 用于记录是否成功；每次循环迭代都会产生 奖励1（1 秒）。每次迭代成功的概率为 $1/2$，且程序以概率 1 终止（几乎必然终止）。其期望运行时间为 2（正几乎必然终止）。

期望运行时间相同，方差不同。我们添加一个缓存（图 1b），将成功概率提高到 $2/3$，初始化成本为 $1/2$ 秒。新程序也几乎必然终止，而且或许更令人惊讶的是，其期望运行时间也为 2。尽管期望运行时间相同，它们的运行时间分布却有所不同（图 2）。图 1a 更早终止的概率较高，而图 1b 的运行时间则更集中在 2 秒附近。两者的方差分别为 2 秒$^2$（无缓存）和 $3/4$秒$^2$（有缓存），因此缓存版本的运行时间更可预测。这个例子表明，期望值——一个平均值——并不总是足以体现概率程序的行为。

```
done : =假; while循环(
¬done) {奖励1; {done : =真
} [1/2] {空操作}
}
probabilistic choice
```

(a) 对重试数据库调用的 Web 服务器进行建模的程序，成功概率为 1/2。每次数据库调用耗时 1 秒。

```
奖励 1/2; done : =假;
while循环 (¬done) {奖励
1; {done : =真} [2/3] {空操
作}}
probabilistic choice
```

(b) 使用缓存的变体。它会初始化缓存（耗时 1/2 秒），然后重试数据库调用，该调用成功的概率为 2/3。每次数据库调用耗时 1 秒。

图 1：用于建模 Web 服务器重试数据库调用的示例程序。

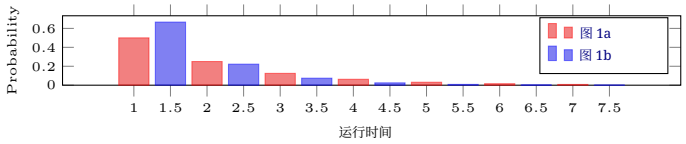



图 2：程序运行时间的概率分布，程序见图 1。

概率程序的演绎验证。针对概率程序的不同目标，人们已开发出专门的技术。最弱前期望（wp）语义 [27] 是一种成熟的形式体系，用于推理程序终止时的期望值。人们已开发出详尽的证明规则目录，包括 [5, 28, 14, 4, 11]。wp 演算已扩展，可用于推理期望运行时间 [21, 19] 和一般奖励 [8]。基于 wp 的工具辅助验证包括*HOL*机械化 [16] 以及*Caesar* 演绎验证器 [35]。人们还针对协方差 [20], 、奖励的高阶矩 [24, 37], 以及摊还复杂度分析 [7]提出了专门的变体。

程序化奖励作为统一的抽象。 不同的目标目前需要各自独立的最弱前置条件演算，阻碍了复用。我们的目标是采用一种适用于所有目标的方法，从而能够将现有的最弱前置条件语义、证明规则和工具复用于广泛的目标。为此，我们在编程语言中加入一条奖励语句，灵感来自 [3, 7], ，用于在程序执行期间收集（非负）奖励。随后，期望累积奖励就是一次程序运行期间收集的所有奖励之和的期望。我们展示了如何使用奖励语句对不同目标进行建模，包括期望运行时间、期望访问时间和折扣奖励。

奖励变换的增量差分。 要处理期望奖励的高阶矩等目标，我们需要对变换后的程序的期望奖励 $S$，例如$\mathsf{E}(\mathsf{rew}(S)^2)$。我们的关键思路是利用奖励的增量差分，灵感来自aert 演算 [7]。我们定义了一种简单的程序变换，将对复杂期望奖励的推理化简为对标准期望奖励的推理，从而能够复用现有的最弱前置条件理论、技术和工具。

贡献与概述。 我们仅用一种编程语言和一种最弱前期望语义，以统一的方式对概率程序的广泛定量目标进行推理。 第2节介绍概率程序的语法和语义。我们的主要贡献如下：

1. 面向多个定量目标的程序化编码（第3节），2. 一项程序变换（第 4.1 节），其相对于马尔可夫链变换（第 4.2 节）的可靠性已得到证明，3. 应用于不同目标，包括高阶矩。（第 5 节），

最后，我们在第6节简要讨论自动化，在第7节比较相关工作，并在第8节作结。

## 2 基础：概率程序的语义

在本节中，我们介绍带有奖励语句的概率程序的语法和语义。我们的阐述基于 [8], ，并有两项关键调整：（1）我们允许任意表达式 $a$ 出现在奖励 $a$语句中，而不仅限于常量；（2）我们允许程序变量取值于集合 $\mathbb{R}^{\infty}_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$。我们形式化地定义概率守卫命令语言（pGCL）程序的语法，通过马尔可夫链描述其操作语义，并给出最弱前期望语义。

### 2.1 概率程序的语法

状态与表达式。 我们考虑一个有限的变量集合 Vars。一个程序状态 $\sigma$ 是从变量到取值的映射，即 $\sigma \in$States $=$ Vars $\to$ Vals。在本文中，我们将使用 Vals $= \mathbb{R}^{\infty}_{\geq 0}$。我们用 $\sigma[x \mapsto e]$ 表示状态 $\sigma$，该状态经过更新后使 $x$ 映射到 $\sigma$ 中的 $e$的取值，而所有其他变量保持不变，即当 $y = x$ 时， $\sigma[x \mapsto e](y) = e(\sigma)$；否则为 $\sigma(y)$。算术表达式$a \in$ AExp 是从程序状态到取值的映射，即 AExp $=$ States $\to$ Vals。布尔表达式 $b \in$ BExp 将程序状态映射到 $\{$ 真， 假 $\}$。

程序。pGCL程序的 $S$ 语法由以下文法定义，其中 $x \in$ 变量集、 $a \in$ 算术表达式、$p \in [0,1]$和 $b \in$ 布尔表达式：

$$S ::= \texttt{skip} \mid x := a \mid \texttt{reward}\ a \mid S_1\,;\, S_2 \\ \mid \{S_1\}\ [p]\ \{S_2\} \mid \texttt{if}\ (b)\ \{S_1\}\ \texttt{else}\ \{S_2\} \mid \texttt{while}\ (b)\ \{\ S\ \}\ .$$

skip语句不执行任何操作。赋值 $x{:=}a$ 将表达式 $a$ 的值赋给变量 $x$。奖励 $a$ 语句在执行时会收集非负奖励 $a$。请注意， $a$ 是一个可能包含程序变量的表达式。顺序组合 $S_1$; $S_2$ 先执行 $S_1$ ，再执行 $S_2$。概率选择 $\{S_1\}\ [p]\ \{S_2\}$ 以概率 $p$ 执行 $S_1$ ，并以概率$1-p$执行 $S_2$ 。条件语句 if ($b$) $\{S_1\}$ else $\{S_2\}$ 在布尔表达式 $b$ 的值为 true 时执行 $S_1$，否则执行 $S_2$ 。while循环 while ($b$) { $S$ } 只要布尔表达式 $b$的值为 true，就会反复执行 $S$ 。

### 2.2 操作语义

程序的操作语义由带奖励的马尔可夫链给出，我们简要回顾其基础。

一个马尔可夫链是一个元组 $\mathcal{M} = (\mathfrak{S},\ \mathsf{Prob}\ \sigma_1,\ \mathsf{rew})$，其中 $\mathfrak{S}$ 是配置的（可数）集合，$\mathsf{Prob}\colon \mathfrak{S} \to \mathsf{Dist}(\mathfrak{S})$ 是转移函数， $\sigma_1 \in \mathfrak{S}$ 是初始配置，$\mathsf{rew}: \mathfrak{S} \to \mathbb{R}^{\infty}_{\geq 0}$ 是奖励函数。

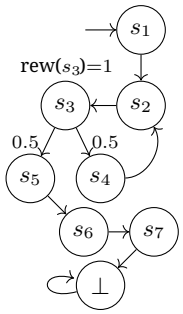


图 3：操作马尔可夫链，针对 图 1a。

长度为 $n$、属于 $\mathcal{M}$的（有限）路径 $\pi \in \overline{\ }^{n}_{\mathcal{M}}$ 是由配置组成的序列 $\pi = s_1 s_2 \ldots s_n$，且对于所有 $1 \leq i < n$，$\mathsf{Prob}(s_i)(s_{i+1}) > 0$。我们通过 $\mathsf{Prob}(s_1 \ldots s_n) = \prod_{1 \leq i < n} \mathsf{Prob}(s_i)(s_{i+1})$ 和 $\mathsf{rew}(s_1 \ldots s_n) = \sum_{1 \leq i \leq n} \mathsf{rew}(s_i)$，将 Prob 和 rew 扩展到有限路径上。令 $\mathsf{E}(\mathsf{rew}(\mathcal{M}))$ 表示 $\mathcal{M}$的期望累积奖励，其定义为 $\sup_{n \in \mathbb{N}} \sum_{\pi \in \text{路径}^{n}_{\mathcal{M}}(\sigma)} \mathsf{Prob}(\pi) \cdot \mathsf{rew}(\pi)$。更一般地，由函数 $f\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 变换后的奖励定义为——义为 $\mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) = \sup_{n \in \mathbb{N}} \sum_{\pi \in \text{路径}^{n}_{\mathcal{M}}(\sigma)} \mathsf{Prob}(\pi) \cdot f(\mathsf{rew}(\pi))$。

该程序的操作马尔可夫链 $\mathcal{M}^{\sigma_1}_{S}$ =（$\mathfrak{S}$、概率 $s_1$和奖励函数）如下定义： $S$ 其状态为可达[1] 配置 $\mathfrak{S} \subseteq (\mathsf{pGCL} \cup \{\Downarrow\}) \times$，转移概率为 概率(见图7），初始配置为 $s_1 = (S,\ \sigma_1)$，奖励函数为奖励函数，定义如下：

$$\mathsf{rew}((S,\sigma)) = \begin{cases} r & \text{if } (\exists S' \in \mathsf{pGCL}.\ S = \texttt{reward}\ r\texttt{;}\ S') \vee (S = \texttt{reward}\ r) \\ 0 & \text{otherwise} \end{cases}$$

图 1a 中程序的操作马尔可夫链 如图 3所示。记$\mathsf{E}(\text{奖励}^{\sigma_1}(S))$ 表示该操作马尔可夫链的期望累积奖励$\mathcal{M}^{\sigma_1}_{S}$，即$\mathsf{E}(\text{奖励}(\mathcal{M}^{\sigma_1}_{S}))$。令$\mathsf{E}(\text{奖励}(S)) = \lambda\sigma.\ \mathsf{E}(\text{奖励}^{\sigma}(S))$。

### 2.3 最弱前期期望语义

一个期望是一个函数 $f$：状态 $\to \mathbb{R}^{\infty}_{\geq 0}$。令 $\mathbb{E}$ 表示所有期望的集合。期望构成一个完备格（$\mathbb{E}$，$\preceq$），其逐点序$X \preceq Y$ 当且仅当 $X$（$\sigma$） $\leq Y$（$\sigma$）对所有 $\sigma \in$ 状态成立。因此，所有下确界和上确界都存在，

[1] 推理规则从每个配置导出的转换数量有限，因此可达配置集合是可数的。

| $S$ | 最弱前置条件 $[\![S]\!](X)$ | $S$ | 最弱前置条件 $[\![S]\!](X)$ |
|---|---|---|---|
| skip | $X$ | $\{S_1\}\ [p]\ \{S_2\}$ | $p\cdot$最弱前置条件$[\![S_1]\!](X)+(1-p)\cdot$最弱前置条件$[\![S_2]\!](X)$ |
| $x \coloneqq a$ | $X[x \mapsto a]$ | 如果 $(b)$ { $S_1$ } 否则 { $S_2$ } | $[b]\cdot$最弱前置条件$[\![S_1]\!](X)+[\neg b]\cdot$最弱前置条件$[\![S_2]\!](X)$ |
| 奖励 $a$ | $a + X$ | while循环 $(b)$ { $S'$ } | 最小不动点 $Y.\ [b]\cdot$最弱前置条件$[\![S']\!](Y) + [\neg b]\cdot X$ |
| $S_1$; $S_2$ | 最弱前置条件$[\![S_1]\!]$(最弱前置条件$[\![S_2]\!](X)$) | | |

图 4：程序的最弱前期望语义的归纳定义$S$，最弱前置条件$[\![S]\!]$: $\mathbb{E} \to \mathbb{E}$ 其中 $X \in \mathbb{E}$ 是后期望。

并且根据克莱尼不动点定理，最小不动点lfp $Y.\ \Phi\ (Y)$ （即单调函数 $\Phi$: $\mathbb{E} \to \mathbb{E}$ 的最小不动点）存在。令 $[b] \in \mathbb{E}$ 将布尔表达式 $b$映射为 $[b]\ (\sigma) = 1$，当且仅当 $b\ (\sigma) =$ 为真，否则映射为0。替换 $X[x \mapsto a]$ （将变量 $x$ 替换为表达式 $a$，作用于 $X \in \mathbb{E}$ ）由 $X[x \mapsto a]\ (\sigma) = X\ (\sigma[x \mapsto a])$ 给出，对所有 $\sigma \in$ 状态成立。期望上的算术运算按逐点方式提升，例如 $(X + Y)\ (\sigma) = X\ (\sigma) + Y\ (\sigma)$ 。

程序 $S$ 的最弱前期望是一个函数 wp$[\![S]\!]$: $\mathbb{E} \to \mathbb{E}$，它通过对 $S$进行结构归纳来定义；见图 4。直观地说，wp$[\![S]\!](X)$ 是程序 $S$ 从初始状态 $\sigma$开始运行时的期望累积奖励，假设终止时获得的奖励为 $X$。语义是标准的[2]，但我们重点讨论奖励和 while 循环。对于奖励语句 reward $a$，有 wp$[\![$reward $a]\!](X)$ $= a + X$，也就是说，对当前状态下的 $a$ 求值所得的结果会加到 $X$上。对于循环 while $(b)$ { $S$ }，前期望由泛函 $Y \mapsto [b] \cdot$ wp$[\![S]\!](Y) + [\neg b] \cdot X$的最小不动点给出，该泛函称为相对于后期望 $X$的循环特征泛函。

例如，图 1a 中循环的最小不动点，在后置条件为 $X = 0$时，由 (lfp $Y.\ [\neg done] \cdot (1 + 0.5 \cdot Y[done \mapsto \texttt{true}] + 0.5 \cdot Y)) = [\neg done] \cdot 2$给出。对于完整程序 $S$，我们有 最弱前置条件$[\![S]\!](0) = ([\neg done] \cdot 2)[done \mapsto \texttt{false}] = 2$。

可靠性。表达式 wp$[\![S]\!](X)(\sigma)$ 的求值结果，是从$\sigma$ [8,开始时，与 $S$ 的语义相对应的马尔可夫链中的期望累积奖励。定理 6],即 wp$[\![S]\!](X)(\sigma) = \mathrm{E}(\mathrm{rew}^{\sigma}_{X}(S))$。这里，$\mathrm{rew}^{\sigma}_{X}(S)$ 是由 $X$ 在操作马尔可夫链 $\mathcal{M}^{\sigma}_{S}$上诱导的奖励函数，它在终止时收集奖励 $X(\sigma)$，否则则收集根据图7为配置 $(S, \sigma)$ 中的每条语句 $S$ 定义的奖励。形式化地，定义为 $\mathrm{rew}_X((\Downarrow, \sigma)) = X(\sigma)$，而 $\mathrm{rew}_X((S, \sigma)) = \mathrm{rew}((S, \sigma))$。我们使用缩写 $\mathrm{rew}^{\sigma}_{X}(S) = \mathrm{rew}_X(\mathcal{M}^{\sigma}_{S})$ 和 $\mathrm{E}(\mathrm{rew}_X(S)) = \lambda\sigma.\ \mathrm{E}(\mathrm{rew}^{\sigma}_{X}(S))$。

健康性条件。与期望值的性质相对应，最弱前置条件满足若干良好性质，通常称为健康性条件 [27,17, 8]。设 $S \in$ pGCL 中的程序为 $X$， $Y \in \mathbb{E}$ 为期望。则，最弱前置条件 $[\![S]\!]$ 是单调的，即对于 $X \preceq Y$，有 wp$[\![S]\!](X) \preceq$ wp$[\![S]\!](Y)$。此外，对于无奖励程序 $S$，最弱前置条件 $[\![S]\!]$ 是线性的，即对于所有 $X$、$Y \in \mathbb{E}$ 和 $a \in \mathbb{R}_{\geq 0}$，有 wp$[\![S]\!](a \cdot X + Y) = a \cdot$ wp$[\![S]\!](X) +$ wp$[\![S]\!](Y)$。

[2] 该最弱前置条件演算最初定义时并不包含奖励语句 [27, 17]；我们使用的扩展演算来自 [8] ，并带有奖励语句。

## 3 程序化奖励建模

我们通过程序语句程序化奖励建模，为概率程序引入奖励。通过用奖励来表达奖励，我们可以将不同的奖励结构直接纳入程序文本中。

### 3.1 终止时奖励

一种常见的奖励结构只在终止时收集奖励，正如原始的最弱前期望语义所考虑的那样 最弱前置条件 [27, 13, 17] (不含奖励 语句）。这里，对于程序 $S$ 和 $f \in \mathbb{E}$，最弱前置条件$[\![S]\!](f)$ 是 $f$ 在 $S$终止时的期望值。利用 最弱前置条件，我们也可以轻松查询经变换后的终止时奖励的期望值，例如 最弱前置条件$[\![S]\!](x^2)$，以获得 $x$ 在 $S$终止时取值的二阶矩。

为了采用更程序化的方法，我们将经典思想推广：通过 断言 语句将后置条件嵌入程序中，从而无需单独指定后置条件。我们不必显式地用某个后期望来指定 $S$，而可以使用 $X$语句来收集终止时的奖励，即使用序列 奖励，并使用 $S$; 奖励 $X$。相应的可靠性陈述也得到了简化，将 $X = 0$替换为奖励$_X$ ，再替换为奖励。这里，关键在于允许 $X$ 中的任意算术函数，包括期望值奖励语句（参见 $\mathbb{E}$ （参见 第2节）。[3]

**引理 1（程序终止时的程序化奖励）**。设 $S \in$pGCL 和$X \in \mathbb{E}$。则，最弱前置条件$[\![S]\!](X)$ = 最弱前置条件$[\![S\texttt{; reward } X]\!](0)$ = $\mathsf{E}(\mathsf{rew}(S\texttt{; reward } X))$。

正如程序化视角所表明的，该模型仅在程序终止时收集奖励。特别是，发散执行不会收集任何奖励。

### 3.2 运行时间与资源消耗

期望运行时间的形式化验证，以及更一般意义上的资源消耗验证，是文献中一个经过广泛研究的主题 [21, 22, 24, 19, 37]。该分析颇为微妙。我们已经考察了 图 1，其中尽管存在一条无限执行路径（数据库调用从未成功），期望运行时间仍然是有限的。下面，我们考察图 5，该图展示了针对一个由单个 while 循环组成的示意程序、用于跟踪运行时间的三种不同奖励模型。

使用计数变量时的不可靠性。 在图 5a中，运行时间被记录在变量 $\tau$ 中，并且 $\tau$ 的奖励会在程序终止时收集。然而，例如对于无穷循环（$b = \texttt{true}$），奖励为零，因为执行永远不会到达 奖励 $\tau$ 语句。这种技术是不可靠的 [22]。

[3]虽然这对于我们的设置并非必要，但我们推测，可以扩展 [6] 中的断言语言，以支持 奖励语句，同时仍保持 *expressive*，也就是说，对于所有语法上的 $X \in \mathbb{E}$，都可以用语法方式表示前期望 最弱前置条件$[\![S]\!](X)$。

```
τ := 0;
while循环 (b) {
  τ := τ + 1;
  S′
}
奖励 τ
```

(a) 跟踪运行时间，通过终止时奖励，对应于 最弱前置条件 $[\![S]\!](\tau)$。

```
τ := 0;
while循环 (b) {
  τ := τ + 1;
  S′
}
奖励 τ²
```

(b) 跟踪平方值 通过奖励跟踪运行时间终止，对应于 最弱前置条件 $[\![S]\!](\tau^2)$。

```
while循环 (b) {
  奖励 1;
  S′
}
```

(c) 通过跟踪运行时间增量奖励 ，对应于 ert$[\![S]\!](0)$。

图 5：针对一个由单个 while 循环组成的示意程序，用于跟踪运行时间的三种不同奖励模型。我们关注的运行时间是循环体 $S'$ 执行的次数。只有图 5c 能够正确跟踪运行时间，而图 5a 和5b 在发散循环（$b = \texttt{true}$）中得到的结果为零。

用于跟踪运行时间二阶矩的改编方法同样存在缺陷。 图 5b 是 rt 演算中一种方法的程序化版本，该方法由 [20],提出，用于分析运行时间的协方差。[4] 它会在终止时收集奖励$\tau^2$ 。但例如对于无穷循环，奖励再次为零。

已有研究 [39]证明，计数器变量方法（针对运行时间的一阶矩）可靠的一个充分条件是几乎必然终止（AST），即以概率 1 终止。证明几乎必然终止是不可判定的 [18], ，因此我们希望采用一种避免这种强附加条件的方法。

**增量运行时间收集。** ert 演算 [21, 22] 是对最弱前置条件的扩展，用于推理所有程序（包括非几乎必然终止的程序）的期望运行时间。[5]核心思想是在每次循环迭代中增量收集奖励。因此，奖励会不断增加，发散路径将累积无限奖励。为了避免再使用另一种演算，我们采用程序化奖励建模，如图 5c所示。这种方法在终止方面无需附加条件即可保证可靠性。

**引理 2（增量运行时间收集）**。设无循环 $S' \in$ pGCL和 $b \in$布尔表达式。那么，最弱前置条件$[\![\texttt{while}\,(b)\ \{\ \texttt{reward}\ 1;\ S'\ \}]\!] =$ 期望运行时间演算$[\![\texttt{while}\,(b)\ \{\ S'\ \}]\!]$。

因此，为了正确跟踪程序的期望运行时间，应避免使用计数变量，而应改用奖励语句来增量地收集奖励。将此方法扩展到其他目标（例如运行时间的高阶矩）是第 4 节的目标。

[4] 来自 [20] 的 rt 演算不同于其后继者——期望运行时间演算 [21, 22]。[5] 为简单起见，我们考虑统计循环迭代次数的运行时模型。

### 3.3 用于奖励建模的幽灵变量

我们现在说明如何使用幽灵变量，通过奖励语句编码各种奖励结构。幽灵变量是不影响程序函数行为、但用于辅助验证的程序变量。术语幽灵变量在演绎验证中是标准术语；它们也被称为辅助（或观测）变量。前文刚刚介绍的计数器变量方法就是这样一个例子。我们针对不同的定量目标给出若干构件，表示为程序变换 $S \rightsquigarrow S'$。假设幽灵变量初始化为 0；则最弱前置条件$[\![S']\!](0)$ 的结果即为所需的奖励。

**折扣奖励。** 在某些应用中，未来奖励被认为不如即时奖励有价值。这被称为 折扣 [34]。我们使用一个幽灵变量来编码折扣 $\tau$，以跟踪程序步数，并收集每个时间步按因子 $\gamma \in [0,1]$ 折扣后的奖励。

$$\texttt{reward}\ a \quad \rightsquigarrow \quad \texttt{reward}\ \gamma^{\tau} \cdot a;\ \tau := \tau + 1\ .$$

**步索引期望值。** 我们得到步索引 期望值，例如由 [30], 使用一个幽灵变量 $\tau$ 来跟踪程序步数，并使用一个逻辑变量 $N$ 表示所关注的步：

$$\texttt{reward}\ a \quad \rightsquigarrow \quad \texttt{if}\ (\tau = N)\ \{\ \texttt{reward}\ a\ \};\ \tau := \tau + 1\ .$$

而当使用条件 $\tau \leq N$ 时，我们会跟踪截至步骤 $N$的累积期望值。这使得我们能够对长期平均奖励进行推理，即截至步骤 $N$的平均奖励在 $N$趋于无穷时的极限：$\limsup_{N\to\infty} {}^{1}/_{N}$·最弱前置条件$[\![S]\!](0)$。

**期望访问时间。** wp 演算允许使用符号条件 $b$ ∈布尔表达式。这使得我们能够对期望访问时间 （又称占用测度）进行推理，即在程序执行期间，满足 $b$ 的状态被访问的期望次数。

$$S \quad \rightsquigarrow \quad S;\ \texttt{if}\ (b)\ \{\ \texttt{reward}\ 1\ \}$$

计算完成后，随机变量 $X$ 的进一步期望值可简化为 $X$ 乘以期望访问时间 [36]。这种方法已成功应用于概率模型检测 [29]。

**首次访问时间与返回时间。**同样，也可以构造用于跟踪满足条件 $b \in$ 布尔表达式 的状态的首次访问时间和首次返回时间的程序构件。幽灵变量 $\phi$用于记录这是否为首次访问/返回。

- *First Visit:* $S \rightsquigarrow S;\ \texttt{if}\ (b)\ \{\ \texttt{reward}\ [\phi = 0];\ \phi := 1\ \}$
- *First Return:* $S \rightsquigarrow S;\ \texttt{if}\ (b)\ \{\ \texttt{reward}\ [\phi = 1];\ \phi := \min(\phi + 1, 2)\ \}$

## 4 奖励变换

现在我们考虑 $f$-变换后的累积奖励，即对累积奖励应用某个函数 $f$ 所得值的期望。使用最弱前期望演算（第 3.1 节），对终止时奖励的期望应用 $f$ 很简单；但对于累积期望奖励（第 3.2 节），情况并非如此。我们不再定义另一种 wp 变体，而是提出一种程序变换 $\mathcal{T}_f(S)$，适用于单调函数 $f\colon \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$。将期望奖励为 E(rew($S$)) 的程序 $S$ 变换为程序 $\mathcal{T}_f(S)$，其期望奖励为经 $f$变换后的期望奖励，即 E(rew($\mathcal{T}_f(S)$)) $=$E($f$(rew($S$)))。

增量奖励更新。 奖励变换的核心思想是逐步收集 $f$变换后的奖励，这一思想受到微积分基本 [7]的启发。一个 奖励 $a$ 语句只需将 $a$ 加到累积奖励中；如果当前累积奖励为 $\tau$，则新的累积奖励为 $\tau + a$。对于$f$变换后的累积奖励，收集到的奖励等于 $f(\tau)$，新的累积奖励等于 $f(\tau{+}a)$。关键思路是：(1) 减去先前的累积奖励 $f(\tau)$，然后 (2) 加上新的累积奖励 $f(\tau{+}a)$。

考虑期望运行时间的建模，其中每一步收集 1 个奖励。于是，以 $f(\tau)$ 开始的奖励序列具有望远镜性质，并归结为最终累积奖励 $f(\tau + N)$：[6]

$$\underbrace{f(\tau)}_{(0)\text{ previous}} + \sum_{i=0}^{N-1} \underbrace{-f(\tau+i)}_{(1)\text{ reset}} + \underbrace{f(\tau+i+1)}_{(2)\text{ set}} \;=\; f(\tau+N)\,. \tag{1}$$

由于 $f$是单调的，差值 $f(\tau + a) - f(\tau)$ 非负，因此我们可以通过一条奖励 语句在程序中编码这一变化。

### 4.1 程序变换

程序变换很简单：我们引入一个幽灵变量 $\tau$ 来跟踪原程序的累积奖励 $S$，并将每个奖励 $a$语句变换为收集差值 $f(\tau{+}a) - f(\tau)$。下文中，我们用$f$: $\mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 表示一个单调函数，以确保奖励差值非负。

**定义 1（奖励程序变换）**。设 $S \in$ pGCL 和 $\tau$是一个在 $S$中未出现的新变量。 $f$-变换后的程序 $\mathcal{T}_f(S)$ 为

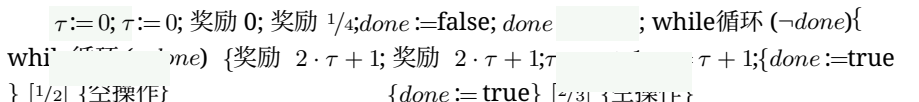
$$\mathcal{T}_f(S) \;=\; \tau \coloneqq 0\texttt{; reward } f(0)\texttt{; } \mathcal{T}'_f(S)\,,$$

其中，辅助变换 $\mathcal{T}'_f(S)$ 将 `reward` 语句替换为：

$$\mathcal{T}'_f(\texttt{reward } a) \;=\; \texttt{reward } (f(\tau + a) - f(\tau))\texttt{; } \tau \coloneqq \tau + a\,,$$

其他语句保持不变。

[6] 我们对所有 $x \in \mathbb{R}^\infty_{\geq 0}$ 定义 $\infty - x = \infty$（特别是 $\infty - \infty = \infty$），使得只要任意一个 $f$（$\tau + i$）为无穷大，总和就为无穷大。




(a) 图 1a 的变换版本。(b) 图 1b 的变换版本。

图 6：程序的变换 $\mathcal{T}_f(S)$ $S$将图 1中的 $f(x) = x^2$。

图6展示了使用 $f(x) = x^2$对图 1中的示例程序进行的变换，突出显示的语句源自此次变换。例如，图 1a中奖励1 语句被变换为奖励$((\tau+1)^2 - \tau^2)$；$\tau := \tau + 1$ 见图 6a。

变换后程序 $\mathcal{T}_f(S)$的期望奖励等于将 $f$ 应用于原程序 $S$的奖励所得的期望值。

**定理 1（可靠性）**。对于 $S \in$ pGCL 以及单调的 $f\colon\ \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$:

$$\mathsf{E}(\mathsf{rew}(\mathcal{T}_f(S))) = \mathsf{E}(f(\mathsf{rew}(S)))\ .$$

用最弱前期望语义表示，我们可以将定理 1表述为$\mathsf{E}(\mathsf{rew}(\mathcal{T}_f(S))) =$最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) = \mathsf{E}(f(\mathsf{rew}(S)))$。特别是，最弱前置条件语义中的辅助变换，针对奖励语句体现了公式（1）中的望远镜求和思想。对于期望 $X \in \mathbb{E}$，我们有:

$$\mathsf{wp}[\![\mathcal{T}'_f(\texttt{reward}\ a)]\!](X) \ = \ \lambda\sigma.\ \underbrace{f(\sigma(\tau)+a(\sigma))}_{(2)\ \text{set}} - \underbrace{f(\sigma(\tau))}_{(1)\ \text{reset}} + \underbrace{X(\sigma[\tau \mapsto \tau + a])}_{\text{next}}\ .$$

该程序变换保持与最弱前置条件类似的健康性条件。

**定理 2（单调性）**。设 $S \in$为 pGCL，并设 $f$，$g\colon\ \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$单调且 $f \preceq g$。对所有 $X \in \mathbb{E}$，我们有 最弱前置条件$[\![\mathcal{T}_f(S)]\!](X) \preceq$最弱前置条件$[\![\mathcal{T}_g(S)]\!](X)$。

该变换关于缩放和平移是线性的。

**定理 3（线性）**。设 $S \in$为 pGCL，并设 $f\colon\ \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$单调。则对所有 $\alpha$，$\beta \in \mathbb{R}_{\geq 0}$，我们有 最弱前置条件$[\![\mathcal{T}_{\alpha\cdot f+\beta}(S)]\!](0) = \alpha \cdot$最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) + \beta$。

对于线性变换函数，我们可以完全避免在变换后的程序 $\mathcal{T}_f(S)$ 中使用幽灵变量 $\tau$。

**定理 4（幽灵消除器）**。令 $f\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 具有 $f(x) = \alpha \cdot x + \beta$ 的形式，其中 $\alpha$、$\beta \in \mathbb{R}_{\geq 0}$为某些值。那么，

$$\mathcal{T}_f(S) \;=\; \texttt{reward}\ \beta\texttt{;}\ \mathcal{T}'_f(S) \qquad \textit{and} \qquad \mathcal{T}'_f(\texttt{reward}\ a) \;=\; \texttt{reward}\ \alpha \cdot a\ .$$

该变换对于变换函数的复合具有良好的性质。本质上：$\mathcal{T}'_g(\mathcal{T}'_f(S)) = \mathcal{T}'_{g\circ f}(S$，其中等式是在忽略变换所引入的变量后理解的。

**定理 5（变换的复合）**。设 $S \in \mathsf{pGCL}$，且$f, g\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$是单调的。假设 $\mathcal{T}'_f(\cdot)$和 $\mathcal{T}'_{g\circ f}(\cdot)$使用同一个变量 $\tau$，而 $\mathcal{T}'_g(\cdot)$使用不同的变量 $\tau'$。那么，

$$\mathsf{wp}[\![\tau' \coloneqq f(\tau)\texttt{;}\ \mathcal{T}'_g(\mathcal{T}'_f(S))]\!] = \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S)\texttt{;}\ \tau' \coloneqq f(\tau)]\!]\ .$$

### 4.2 马尔可夫链的奖励变换

为证明程序变换的可靠性（定理 1），我们直接为马尔可夫链定义奖励变换。通过增加一个维度 $\mathbb{R}^{\infty}_{\geq 0}$ 来跟踪原始马尔可夫链的累积奖励，该奖励在初始状态 $\sigma'_1$中为零，并调整奖励函数以累积增量差分。这里，为简洁起见，我们限制为满足 $f$ 且 $f(0) = 0$的单调函数。

**定义 2（马尔可夫链变换）**。设马尔可夫链 $\mathcal{M}$ =(状态，概率 $\sigma_1$，奖励以及单调函数 $f\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 满足 $f(0) = 0$。[7]
我们定义 $\mathcal{T}_f(\mathcal{M})$ =(状态 $\times\, \mathbb{R}^{\infty}_{\geq 0}$，概率′，$\sigma'_1$，奖励′)，其中

- $\sigma'_1 = (\sigma_1, 0)$,
- $\mathsf{Prob}'((\sigma,\alpha),(\sigma',\alpha')) = \begin{cases} \mathsf{Prob}(\sigma)(\sigma') & \textit{if } \alpha + \mathsf{rew}(\sigma) = \alpha', \\ 0 & \textit{otherwise} \end{cases}$
- $\mathsf{rew}'((\sigma,\alpha)) = f(\alpha + \mathsf{rew}(\sigma)) - f(\alpha)$.

请注意，该变换不会保留可达状态空间的大小，因为新增维度 $\mathbb{R}^{\infty}_{\geq 0}$ 记录了奖励累积。

例 *1*（可达状态爆炸）。设马尔可夫链 $\mathcal{M}$ =(状态，概率 $\sigma_1$，奖励)满足 States $= \{\sigma_1\}$, $\mathsf{Prob}(\sigma_1)(\sigma_1) = 1$，$\mathsf{rew}(\sigma_1) = 1$。设 $f(x) = x$。变换后的马尔可夫链 $\mathcal{T}_f(\mathcal{M})$ 存在一条无限路径 $(\sigma_1, 0)$，$(\sigma_1, 1)$，$(\sigma_1, 2)$……，其发生概率为 1。可达状态集合为 $\{(\sigma_1, \alpha) \mid \alpha \in \mathbb{N}\}$，这是一个无限集合。

变换后的马尔可夫链 $\mathcal{T}_f(\mathcal{M})$ 的路径与原始 $\mathcal{M}$的路径一一对应，由此可得可靠性定理。

[7] 限制 $f(0) = 0$在 $\mathcal{T}_f(\mathcal{M})$ 可以很容易地通过使用函数 $f\ x\ -f\ 0$ $\mathcal{T}_g\ \mathcal{M}$ $\sigma'_0$ $g\ x\ =$ $f\ 0$ ()()()（针对 ()）来解除，并在前面添加一个奖励为 () 的新初始状态。完整构造见 附录 A.3。

**引理 3.** 设马尔可夫链 $\mathcal{M} = (\text{状态}, \text{概率}\ \sigma_1, \text{重写})$ 和 $f\colon\ \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 是单调的，且 $f(0) = 0$。设 $n \in \mathbb{N}$。对于每个 $\sigma_1 \ldots \sigma_n \in \text{路径}^{=n}_{\mathcal{M}}$，都存在一个从 $(\sigma_1, \alpha_1)\ldots(\sigma_n, \alpha_n) \in \text{路径}^{=n}_{\mathcal{T}_f(\mathcal{M})}$ 到 $\mathcal{T}_f(\mathcal{M})$ 的一一映射，使得：
*1.* 相同的路径概率：$\mathsf{Prob}(\sigma_1 \ldots \sigma_n) = \mathsf{Prob}'((\sigma_1, \alpha_1)\ldots(\sigma_n, \alpha_n))$，*2.* $\alpha_i$ 跟踪先前的未变换奖励：$\forall 1 \leq i \leq n.\ \sum_{j<i} \mathsf{rew}(\sigma_j) = \alpha_i$，3 $= \ldots$。
变换后奖励：$f(\mathsf{rew}(\sigma_1 \ldots \sigma_n))\ \mathsf{rew}'((\sigma_1, \alpha_1)\,(\sigma_n, \alpha_n))$。

由此可知，原始马尔可夫链$\mathcal{M}$ 的期望 $f$变换奖励等于变换后马尔可夫链 $\mathcal{T}_f(\mathcal{M})$的期望奖励。

**定理 6（可靠性）。** 对于每个马尔可夫链 $\mathcal{M}$ 以及单调的 $f$：

$$\mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) = \mathsf{E}(\mathsf{rew}'(\mathcal{T}_f(\mathcal{M})))\ .$$

请注意，可以放宽函数 $f$的单调性要求。对于该变换的可靠性，只需对马尔可夫链中所有实际出现的值 $\tau$ 与 $a$，$f(\tau+a)-f(\tau) \geq 0$成立即可。

## 5 奖励变换函数

我们使用不同的变换函数 $f$ 来应用程序变换。

### 5.1 奖励的高阶矩

第 $k$ 阶矩随机变量 $X$ 的期望值为$\mathsf{E}(X^k)$。对于某个 $f(x) = x^k$，选取变换函数 $k \in \mathbb{N}_{>0}$ 即可得到累积奖励的高阶矩。根据 定理 1，有 $\mathsf{E}(\text{奖励}(S)^k) = \text{最弱前置条件}[\![\mathcal{T}_f(S)]\!](0)$。

示例 *2*（*Web* 服务器运行时间的方差）。我们回到图6，其中展示了使用 $f(x) = x^2$对图 1 中的 Web 服务器示例进行变换，以获得期望运行时间的二阶矩。为计算期望奖励，即 最弱前置条件$[\![S]\!](0)$，对于图6a，我们先处理循环。循环关于后置条件 0 的 最弱前置条件为

$$\mathrm{lfp}\, X.\ [\neg \mathit{done}] \cdot ((2 \cdot \tau + 1) + 0.5 \cdot X[\mathit{done} \mapsto \mathtt{true}] + 0.5 \cdot X) + [\mathit{done}] \cdot 0.$$

其值为 $[\neg \mathit{done}] \cdot (4 \cdot \tau + 6)$。对剩余的前三行应用最弱前置条件，得到最弱前置条件$[\![\mathcal{T}_f(S)]\!] = 6$。因此，图 1a 的期望奖励的二阶矩为 6。由于一阶矩为 2（参见第 2.3 节），方差为 $\mathsf{Var}(\text{奖励}(S)) = \mathsf{E}(\text{奖励}(S)^2) - (\mathsf{E}(\text{奖励}(S)))^2 = 6 - 2^2 = 2$。

同样，可以证明 最弱前置条件$[\![S_{\text{loop}}]\!]$在 图 6b 中循环的 (0) 等于 $[\neg \mathit{done}] \cdot (3 \cdot (\tau - 0.5) + 4.5)$。将 最弱前置条件 应用于前三行，可得最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) = 4.75$，即 图 1b中奖励的期望奖励的二阶矩。期望奖励的方差为 $\mathsf{Var}(\text{奖励}(S)) = 4.75 - 2^2 = 0.75$。

例 *3*（有偏随机游走）。考虑一个从位置$x > 1$ 开始的随机游走，每步以 0.75 的概率向左移动两步，以 0.25 的概率向右移动两步，直到到达位置 $x \leq 1$。二阶矩的上界为 $x^2 + 3 \cdot x$，我们通过程序变换和最弱前置条件推理验证了这一点。

### 5.2 累积分布函数

我们可以使用 $f(x) = [x \geq N]$ 针对某个 $N \in \mathbb{N}$ 计算期望奖励的累积分布函数（CDF）。

例*4*（*Web* 服务器运行时间的累积分布函数）。为了获得图 1a的累积分布函数，我们用 $f(x) = [x \geq N]$对其进行转换：

$\tau \coloneqq 0$; reward $[0 \geq N]$; $done \coloneqq$ false;
while循环$(\neg done)$ {
  reward $[\tau + 1 \geq N] - [\tau \geq N]$; $\tau \coloneqq \tau + 1$; $\{done \coloneqq \texttt{true}\}$ $[1/2]$ {skip} }

注意，可以将奖励语句简化为奖励 $[\tau + 1 = N]$。对于累积分布函数，我们得到最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) = 0.5^{N \ominus 1}$，其中 $a \ominus b = \max(0, a - b)$。

### 5.3 期望超额

考虑一个固定预算 $N \in \mathbb{R}_{\geq 0}$，目标是确定累积奖励在期望意义上超出该预算多少。下面的例子展示了如何利用我们的程序变换来计算循环运行时间的期望超额，即对所有 $N \in \mathbb{R}_{\geq 0}$ 求 E((rew($S$) $\ominus N$))。

例*5*（用于计算期望超额的循环拆分）。我们根据数据库调用的成功概率 $p \in [0, 1]$ 和预算 $N \in \mathbb{N}$，计算图 1a中 Web服务器程序的期望超额。应用变换，其中 $f(x) = x \ominus N$，化简后得到[8]，如下结果：

$\tau :=0$; $done :=$假;
while $(\neg done)$ { reward $[\tau \geq N]$; $\tau \coloneqq \tau + 1$; $\{done \coloneqq \texttt{true}\}$ $[p]$ {skip} }

现在不难看出，该循环分两个阶段运行。第一阶段一直运行到运行时间低于 $N$，不产生任何奖励；第二阶段则收集运行时间超出的部分。将循环拆分为两个循环后得到：

$\tau \coloneqq 0$; $done \coloneqq$ false;
while $(\neg done \wedge (\tau < N))$ { $\tau \coloneqq \tau + 1$; $\{done \coloneqq \texttt{true}\}$ $[p]$ {skip} };
while $(\neg done)$ { reward 1; $\tau \coloneqq \tau + 1$; $\{done \coloneqq \texttt{true}\}$ $[p]$ {skip} }

第二个循环的期望奖励为 $1/p$ 当 $done =$ false时，否则为 0。形式化地：最弱前置条件 $[\![loop2]\!](0) = [\neg done] \cdot 1/p$。第一个循环以 $done =$ false 终止的概率为 $(1-p)^N$。形式化地：最弱前置条件$[\![S]\!](0) = \text{wp}[\![loop1]\!](\text{wp}[\![loop2]\!](0)) = (1-p)^N/p$，这就是运行时间超过阈值 $N$ 的期望超额。例如，如果 $p = 0.1$ 且 $N = 10$，期望超额为 $(1-0.1)^{10}/0.1 \approx 3.486$。

[8] We use that $(((\tau + 1) \ominus N) - (\tau \ominus N))$ is equal to $[\tau \geq N]$.

### 5.4 矩母函数

随机变量 $X$ 的矩母函数$\mathsf{E}(e^{t\cdot X})$ 定义为$t \in \mathbb{R}_{\geq 0}$ [12]。通过对矩母函数在 $t=0$ 处求导，可以得到 $X$ 的所有矩。对于 $f(x) = e^{t\cdot x}$，我们的奖励变换可得矩母函数。[9]

示例 6 $\{x=1\}\ [p]\ \{x=0\}$ $S =$（随机赋值的矩母函数）。考虑程序 $x$ ;; 奖励 。令 $f\ x = e^{t\cdot x}$ ()。可以证明 最弱前置条件$[\![\mathcal{T}_f\ S\ ]\!]\ 0 = [\tau \mapsto 0] = p\cdot e^t + (1-p)$。 $n$阶导数在 $t=0$处给出 $n$阶矩： $\mathsf{E}(X^n) = p$ 对所有 $n \in \mathbb{N}_{>0}$。

### 5.5 多个奖励

有时，同时跟踪多个奖励并将它们汇总为单个奖励很有用。例如，在一些云计算平台中，定价基于运行时间与内存消耗 [2]的乘积。

通过引入多个奖励变量，我们的变换可以推广到多奖励设置 $\tau_i$。推广后的奖励语句可以一次性分配多个奖励，例如 奖励 $(a_1,\ a_2)$ 将奖励 $a_1$ 分配给奖励 1，并将奖励$a_2$ 分配给奖励 2。变换 $\mathcal{T}'_f(S)$ 可推广为 $f\colon\ (\mathbb{R}^{\infty}_{\geq 0})^{\ n} \to \mathbb{R}^{\infty}_{\geq 0}$，将 $n$多个奖励映射为单个奖励：

$$\mathcal{T}'_f(\texttt{reward}\ (a_1,\ldots,a_n)) = \texttt{reward}\ f(\tau_1+a_1,\ldots,\tau_n+a_n) - f(\tau_1,\ldots,\tau_n);\\ \tau_1 \coloneqq \tau_1 + a_1\texttt{;}\ \ldots\texttt{;}\ \tau_n \coloneqq \tau_n + a_n$$

示例 7（运行时间和内存消耗下的期望成本）。考虑一个程序，其初始拥有两个单位的内存和计算资源。以 $p$ 的概率，它需要更多计算资源；为此，它以 $q$ 的概率请求额外的一个单位内存。我们可以将其写成一个程序，使用 $f(\tau_1,\ \tau_2) = \tau_1\cdot\tau_2$ 应用我们的变换（将成本建模为运行时间与内存的乘积），并计算期望成本。结果为 最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) = 4+(p\cdot(2+(q\cdot 3)\ )$ 。

## 6 自动化

现成的 *Caesar* 概率程序演绎验证器 [35]被用于支持 奖励 语句，并实现了多种证明规则，用于推理期望奖励的下界和上界。一个例子是 归纳证明规则 [33, 17], ，用于验证 上界 $I \in X$ 关于 最弱前置条件$[\![S]\!](X)$，对于$S$ =while循环 $(b)\ \{\ S\ \}$ 和 $X \in \mathbb{E}$，其中 $I$ 是用户提供的循环不变式：

$$[b]\cdot \mathsf{wp}[\![S]\!](I) + [\neg b]\cdot X \sqsubseteq I \quad \text{implies} \quad \mathsf{wp}[\![\texttt{while}\ (b)\ \{\ S\ \}]\!](X) \sqsubseteq I\ .$$

[9] 矩母函数类似于 *probability-generating functions*（PGF），最近用于概率程序中的推断 [10, 40, 25]。然而， $f\ (x)\ = X^x$ 对于 PGF 并非单调，因此 $X<1$ 我们的变换不适用。

我们在第 5 节 中附录 B提供了示例的 Caesar 输入文件，使用归纳证明规则验证期望奖励的上界。此外，Caesar 支持通过概率模型检查器*Storm* [15]分析带整数变量的（有限）操作语义。对于无限状态操作语义（参见 图 3），Stor m 可以对期望值进行下近似。

## 7 相关工作

期望运行时间分析。在第 3.2 节中，我们讨论了例如对ert 演算 [21, 22]的高阶矩缺乏支持。同样的问题也适用于摊销期望运行时间演算 [7], ，它扩展了ert演算，以对势函数进行推理，详见附录 C。对ert风格运行时间的自动推断已在例如 [32]中得到研究。我们假设非负奖励；针对混合符号奖励的分析包括 [38, 9]。作为一种互补方法，[1]提出以渐近界而非期望值进行运行时间分析。

高阶矩。Kaminski 等人旨在推理概率程序的协方差 [20]。他们定义了一种称为rt 的语义，该语义将专用运行时间变量 $\tau$ 对每条已执行语句增加一，并考虑 $\tau^k$ 在终止时的期望值。然而，如第 3.2 节所述，该方法对发散程序存在缺陷。Kura 等人 [24] 关注运行时间的尾概率，并利用高阶矩来获得其界。他们综合出 $k$阶矩的上界，采用基于模板的不变式。在 [37], Wang 等人提出了一种自动方法，用于推断概率程序期望运行时间的高阶矩。与先前方法一样，必须专门针对其语义证明相应的证明规则。下界也被考虑，并用于推导中心矩。他们的方法是自动化的，并使用基于模板的不变式综合来得到区间界。针对步索引期望值高阶矩的代数方法已用于 [30]。在满足某些限制的程序上，例如无环的非线性变量依赖，且条件中的变量仅取有限多个值时，他们可以为所有矩计算闭式解。关于步索引期望值高阶矩的困难性结果发表于 [31]。

## 8 结论

我们介绍了带奖励语句的概率程序，从而为分析多个目标提供了一种通用方法，例如期望运行时间、期望访问时间、期望奖励的高阶矩，或超过阈值的期望超额。所有这些问题都可归结为“经典”的 wp 演算，无需针对每个目标分别设计专用演算或语义。因此，现有的 wp 证明规则和工具都可以在我们的框架中重用。未来的工作包括纳入具有非确定性的程序；程序化奖励建模同样适用于这种情形，但由于计数变量 $\tau$ 引入了额外信息，奖励变换也因此带来了一些额外的微妙问题。

**致谢**。本研究部分获 ERC POC 项目 VERIPROB（项目编号 101158076）以及欧盟“地平线 2020”研究与创新计划资助，资助形式为玛丽·斯克沃多夫斯卡-居里项目 MISSION（项目编号 101008233）。

## 参考文献

1. Ajdar´ow, M., Ku˘cera, A., Novotn´y, P.：《概率程序的渐近分析：当期望不符合我们的预期》。载于：《验证原理：探索概率领域——献给 Joost-Pieter Katoen 六十岁生日的论文》，第一部分。编者：N. Jansen、S. Junges、B.L. Kaminski、C. Matheja、T. Noll、T. Quatmann、M. Stoelinga 和 M. Volk。沙姆：施普林格·自然瑞士，2025年，页码 85–97。 ISBN：978-3-031-75783-9。 https://doi.org/10.1007/978-3-031-75783-9_4。2. 亚马逊云科技，AWS Lambda 定价，访问日期：2025年12月1日。https: //aws.amazon.com/lambda/pricing/ 3. Avanzini, M., Moser, G., Schaper, M.：概率程序的模块化成本分析。《ACM 编程语言论文集》4(OOPSLA)，172:1–172:30 (2020)。https: //doi.org/10 3428240 4 1145. / . Batz, K., Chen, M., Kaminski, B.L., Katoen, J., Matheja, C., Schr¨oer, P.：格上的 k 归纳及其在概率程序中的应用。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》。LNCS，第12760卷，页码 524–549。施普林格，海德堡（2021）。 https://doi.org/10.1007/978- 3- 030-81688-9_255。Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.：贝叶斯网络啊，我还要采样多久？——从程序分析视角看期望采样时间。载于：Ahmed, A.（编）《编程语言与系统——第27届欧洲编程研讨会，ESOP 2018，作为欧洲软件理论与实践联合会议 ETAPS 2018 的一部分举行，希腊塞萨洛尼基，2018年4月14–20日，论文集》。LNCS，第10801卷，页码 186–213。施普林格，海德堡（ 2018 10 1007 978 3 319 89884 1_7 ）。 6 https://doi.org/ . / - - - - . Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.：概率程序的相对完备验证：一种用于基于期望推理的表达性语言。《ACM 编程语言论文集》5(POPL)，1–30 (2021)。https://doi.org/10.1145/ 3434320 7. Batz, K., Kaminski, B.L., Katoen, J., Matheja, C., Verscht, L.：摊销期望运行时间演算。《ACM 编程语言论文集》7(POPL)，1957–1986 (2023 3571260 8 10 1145 )。https://doi.org/ . / . Batz, K., Kaminski, B.L., Matheja, C., Winkler, T.：J-P: MDP。FP。PP——将马尔可夫决策过程中的总期望奖励刻画为最小不动点，并将其应用于概率程序的操作语义。载于：Jansen, N., Junges, S., Kaminski, B.L., Matheja, C., Noll, T., Quatmann, T., Stoelinga, M., Volk, M.（编）《验证原理：探索概率领域——献给 Joost-Pieter Katoen 六十岁生日的论文》，第一部分。LNCS，第15260卷，页码 255–302。施普林格，海德堡（2024）。https://doi.org/10.1007/978-3-031-75783-9_11

9. Chatterjee, K., Goharshady,A.K., Meggendorfer, T., ˇZikeli´c, u.：概率程序资源使用的定量界限。《ACM 编程语言论文集》8(OOPSLA 1)
(20243649824 10101145). https://doi.org/./. Chen, M., Katoen, J., Klinkenberg, L., Winkler, T.：程序是否产生正确的分布？——通过生成函数验证概率程序。载于：Shoham, S., Vizel, Y.（编）计算机辅助验证——第34届国际会议，CAV 2022，以色列海法，2022年8月7-10日，论文集，第一部分。LNCS，第13371卷，第79–101页。施普林格，海德堡（2022）。https://doi.org/10.1007/978-3-031-13185-1_511. Feng, S., Chen, M., Su, H., Kaminski, B.L., Katoen, J., Zhan, N.：可能不终止的概率程序的下界。ACM程序语言汇刊 7(OOPSLA1 ),
696 3586051 12 726 2023 10 1145 – ( )。https://doi.org/ 。/ 。Fristedt, B., Gray, L.：“概率论现代导论”。载于：《概率及其应用》。瑞士巴塞尔：Birkhauser Verlag AG，1996年8月。第13章。13. Gretz, F., Katoen, J., McIver, A.：概率守卫命令语言的操作语义与最弱前期望语义。Perform. Evaluation 73，110 132 2014 10 1016 2013 11 – ( )。https://doi.org/ 。/J.PEVA. 。.004 14. Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.：瞄准下界更难：概率程序验证中的下界归纳法。Proc. ACM Program. Lang. 4 3371105 15 37 1 37 28 2020 10 1145 (POPL)，：–：( )。https://doi.org/ 。/ 。Hensel, C., Junges, S., Katoen, J., Quatmann, T., Volk, M.：概率模型检查器 Storm。Int. J. Softw. Tools Technol. Transf. 24(4)，589–610（2022）。https://doi.org/10.1007/S10009-021-00633-Z 16. Hurd, J., McIver, A., Morgan, C.：在*HOL*中机械化概率守卫命令。Theor. Comput. Sci. 346(1)，96–112（2005）。https://doi.org/10.1016/J. TCS.2005.08.005 17. Kaminski, B.L.：概率程序的高级最弱前置条件演算。博士论文，亚琛工业大学，德国（2019）。https://doi.org/10. 18154 / RWTH - 2019- 01829。http: // publications. rwth - aachen. de/ record / 755408。18. Kaminski, B.L., Katoen, J.：论几乎必然终止的困难性。载于：Italiano, G.F., Pighizzini, G., Sannella, D.（编）：计算机科学数学基础2015——第40届国际研讨会，MFCS 2015，意大利米兰，2015年8月24日至28日，论文集，第I部分。LNCS，第9234卷，第307–318页。施普林格，海德堡（
2015 10 1007 978 3 662 48057 1_24 ）。
19
https://doi.org/ 。/ - - - - 。Kaminski, B.L., Katoen, J., Matheja, C.：通过程序验证分析期望运行时间。载于：《概率编程基础》。G. Barthe、J. Katoen和A. Silva编，第185–220页。剑桥大学出版社（2020）。https: //doi.org/10 007 20 1017 9781108770750 。/ 。. Kaminski, B.L., Katoen, J., Matheja, C.：推断概率程序的协方差。载于：Agha, G., Houdt, B.V.（编）：系统定量评估——第13届国际会议，QEST 2016，加拿大魁北克市，2016年8月23日至25日，论文集。LNCS，第9826卷，第191–206页。施普林格，海德堡（2016）。https://doi.org/ 10 1007 978 3 319 43425 4_14 。/
21
- - - - 。Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.：对概率程序期望运行时间的最弱前置条件推理。载于：Thiemann, P.（编）：编程语言与系统——第25届欧洲编程研讨会，ESOP 2016，作为欧洲软件理论与实践联合会议ETAPS 2016的一部分举行，荷兰埃因霍温，2016年4月2日至8日，

2016，论文集。LNCS，第9632卷，第364–389页。施普林格，海德堡（2016）。https://doi.org/ 10 1007 978 3 662 49498 1_15 。/ - - - - 。
22 Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.：随机化算法期望运行时间的最弱前置条件推理。《ACM学报》65(5)，30:1–30 3208102 23 68 2018 10 1145：( )。https://doi.org/ 。/ 。Kemeny, J.G., Snell, J.L.：有限马尔可夫链。施普林格，纽约，NY（1976）24. Kura, S., Urabe, N., Hasuo, I.：通过高阶矩鞅分析随机化程序运行时间的尾概率。载于：Vojnar, T., Zhang, L.（编）：系统构造与分析工具和算法——第25届国际会议，TACAS 2019，作为欧洲软件理论与实践联合会议ETAPS 2019的一部分举行，2019年11月，6 11428，135 153布拉格，捷克共和国，4月——论文集，第II部分。LNCS，第 卷，第页–。施普林格，海德堡（ 2019 10 1007 978 3 030 17465 1_8 ）。https://doi.org/ 。/
25 - - - - 。Li, J., Zhang, Y.：使用生成函数进行编译。Proc. ACM Program. Lang. 9 3747534 26 2025 10 1145 (ICFP) ( )。https://doi.org/ 。/ 。Lumbroso, J.O.：从抛硬币中进行最优离散均匀生成及其应用。CoRR abs/1304.1916（2013）27. McIver, A., Morgan, C.：概率系统的抽象、精化与证明。施普林格（2005）28. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.：几乎必然终止的新证明规则。Proc. ACM Program. Lang. 2(POPL)，33:1–33:28（2018）。https: //doi.org/10 3158121 29 1145。/ 。Mertens, H., Katoen, J., Quatmann, T., Winkler, T.：计算马尔可夫链中的期望访问时间和平稳分布：快速且准确。《自动化推理杂志》 **69** 3 23 2025 10 1007 10817 025 09736 7
30 ( )，( )。https://doi.org/ 。/S - - - 。Moosbrugger, M., Stankovic, M., Bartocci, E., Kovacs, L.：概率循环的时刻。Proc. ACM Program. Lang. 6(OOPSLA2)，1497–1525（2022）。https://doi.org/ 10 1145 3563341 。/ 。Mullner, J., Moosbrugger, M., Kovacs, L.：强不变式很难：论（概率）程序最强多项式不变式的困难性。Proc. ACM Program. Lang. 8(POPL)，882–910（2024）。
31 https://doi.org/10.1145/3632872 32. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.：有界期望：概率程序的资源分析。载于：Foster, J.S., Grossman, D.（编）：第39届ACM SIGPLAN编程语言设计与实现会议论文集，PLDI 2018，美国宾夕法尼亚州费城，2018年6月18日至22日，第496–512页。ACM（2018 3192394 33 10 1145 3192366）。https://doi.org/ 。/ 。. Park, D.：不动点归纳与程序性质证明。Machine Intelligence 5（1969）34. Puterman, M.L.：“马尔可夫决策过程：离散随机动态规划”。载于：《威利概率与统计丛书》。威利，1994年。第6章。ISBN: 978-0-47161977-2。https://doi.org/10.1002/9780470316887。35. Schr¨oer, P., Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.：概率程序的演绎验证基础设施。Proc. ACM Program. Lang. 7 3622870 36 2 2052 2082 2023 10 1145 (OOPSLA )，–（）。https://doi.org/ 。/ 。Sharir, M., Pnueli, A., Hart, S.：概率程序验证。SIAM J. Comput. 13 0213021 37 2 292 314 1984 10 1137（），–（）。https://doi.org/ 。/ 。Wang, D., Hoffmann, J., Reps, T.W.：概率程序成本累加器的中心矩分析。载于：Freund, S.N., Yahav, E.（编）：PLDI ’21：第42届ACM SIGPLAN编程语言设计与实现国际会议，

编程语言设计与实现大会、虚拟活动、加拿大，2021年6月20日至25日，第559–573页。ACM（2021 3454062.3481011453453483）。https://doi.org/./.. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.：非确定性概率程序的成本分析。载于：第40届 ACM SIGPLAN 编程语言设计与实现大会论文集。PLDI 2019，第204–220页。计算机协会，美国亚利桑那州凤凰城（2019 3314581.3391011453314221）。https://doi.org/./.. Winkler, T.：离散概率 while 循环的期望运行时间（个人笔记）。（2023）40. Zaiser, F., Murawski, A.S., Ong, C.L.：通过概率生成函数对离散模型进行精确贝叶斯推断：一种概率编程方法。载于：Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S.（编）：神经信息处理系统进展第36卷：2023年神经信息处理系统年会，NeurIPS 2023，美国路易斯安那州新奥尔良，2023年12月10日至16日（2023）。http://papers.nips.cc/paper_f i les/paper/2023/hash/0747af6f877c0cb555fea595f01b0e83-Abstract-Conference.html

$$\frac{}{(\Downarrow,\sigma) \xrightarrow{1} \bot} \qquad \frac{}{\bot \xrightarrow{1} \bot}$$

$$\frac{}{(\texttt{skip},\sigma) \xrightarrow{1} (\Downarrow,\sigma)} \qquad \frac{}{(x \coloneqq a,\sigma) \xrightarrow{1} (\Downarrow,\sigma[x \mapsto a])} \qquad \frac{}{(\texttt{reward}\ a,\sigma) \xrightarrow{1} (\Downarrow,\sigma)}$$

$$\frac{(S_1,\sigma) \xrightarrow{p} (\Downarrow,\sigma')}{(S_1\texttt{;}\ S_2,\sigma) \xrightarrow{p} (S_2,\sigma')} \qquad \frac{(S_1,\sigma) \xrightarrow{p} (S_1{}',\sigma')}{(S_1\texttt{;}\ S_2,\sigma) \xrightarrow{p} (S_1{}'\texttt{;}\ S_2,\sigma')}$$

$$\frac{}{(\{S_1\}\ [p]\ \{S_2\},\sigma) \xrightarrow{p} (S_1,\sigma)} \qquad \frac{}{(\{S_1\}\ [p]\ \{S_2\},\sigma) \xrightarrow{1-p} (S_2,\sigma)}$$

$$\frac{[\![b]\!]_\sigma = \texttt{true}}{(\texttt{if}\ (b)\ \{S_1\}\ \texttt{else}\ \{S_2\},\sigma) \xrightarrow{1} (S_1,\sigma)} \qquad \frac{[\![b]\!]_\sigma = \texttt{false}}{(\texttt{if}\ (b)\ \{S_1\}\ \texttt{else}\ \{S_2\},\sigma) \xrightarrow{1} (S_2,\sigma)}$$

$$\frac{[\![b]\!]_\sigma = \texttt{true}}{(\texttt{while}\ (b)\ \{\ S\ \},\sigma) \xrightarrow{1} (S\texttt{; while}\ (b)\ \{\ S\ \},\sigma)} \qquad \frac{[\![b]\!]_\sigma = \texttt{false}}{(\texttt{while}\ (b)\ \{\ S\ \},\sigma) \xrightarrow{1} (\Downarrow,\sigma)}$$

图 7：pGCL 程序的操作语义，其中 $(S_1,\sigma) \xrightarrow{p} (S_2,\sigma')$ 当且仅当 $\mathsf{Prob}((S_1,\sigma))((S_2,\sigma')) = p$，对于 $\sigma \in$ 状态以及 $S, S_1, S_2 \in$ pGCL。

# A 证明

## A.1 第 2 节的定义

对于程序 $S$，其操作马尔可夫链 $\mathcal{M}_S^{\sigma_1} =$（$\mathfrak{S}$，概率 $s_1$，奖励）由可达配置 $\mathfrak{S} \subseteq$（pGCL $\cup \{\Downarrow\}$）$\times$ 状态 $\cup \{\bot\}$ 及转移概率概率构成，其中概率是由图 7中的推理规则定义的最小关系，初始配置 $s_1 =$（$S$，$\sigma_1$），以及奖励函数奖励定义如下：

$$\mathsf{rew}((S,\sigma)) = \begin{cases} r & \text{if } (\exists S' \in \mathsf{pGCL}.\ S = \texttt{reward}\ r\texttt{;}\ S') \vee (S = \texttt{reward}\ r) \\ 0 & \text{otherwise} \end{cases}$$

对于 图 1a，图8 展示了从任意初始状态出发的 Web 服务器程序的操作马尔可夫链 $\sigma$，其中显式列出了配置；这是正文中 图 3 的完整版本。

## A.2 第 3 节的证明

**引理 1（程序终止时的程序化奖励）**。设 $S \in$pGCL 与 $X \in \mathbb{E}$。则，最弱前置条件 $[\![S]\!](X)$ = 最弱前置条件$[\![S\texttt{;}\ \texttt{reward}\ X]\!](0)$ = $\mathsf{E}(\mathsf{rew}(S\texttt{;}\ \texttt{reward}\ X))$。

证明。设 $S \in$ pGCL、$X \in \mathbb{E}$ 和 $\sigma \in$ 状态。根据 wp语义，有 $\mathsf{wp}[\![S]\!](X)(\sigma) = \mathsf{wp}[\![S]\!](\mathsf{wp}[\![\text{奖励}\ X]\!](0))(\sigma)$。根据 wp可靠性，这等于 $\mathsf{wp}[\![S\texttt{;}\ \text{奖励}\ X]\!](0)(\sigma) = \mathsf{E}(\text{奖励}_0^\sigma(S\texttt{;}\ \text{奖励}\ X)) = \mathsf{E}(\text{奖励}^\sigma(S\texttt{;}\ \text{奖励}\ X))$。

□

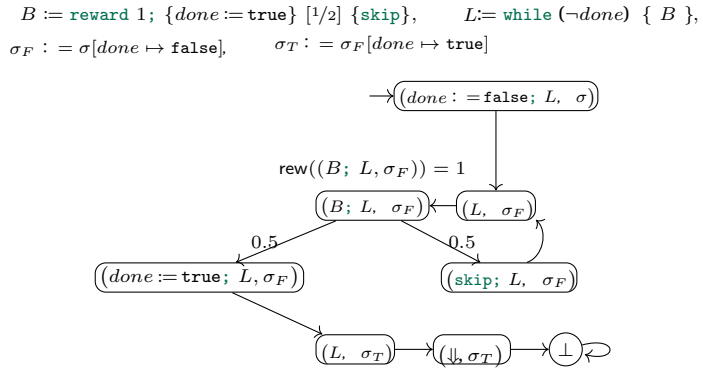



图 8：来自 图 1a 的 Web服务器程序的配置级操作马尔可夫链，由任意初始状态 $\sigma$ 实例化；这是图 3的完整版本。

| $S$ | 最弱前置条件 $[\![S]\!](X)$ |
|---|---|
| $\texttt{skip}$ | $X$ |
| $x \coloneqq a$ | $X[x \mapsto a]$ |
| $S_1;\ S_2$ | 最弱前置条件$[\![S_1]\!]$(最弱前置条件$[\![S_2]\!](X)$) |
| $\{S_1\}\ [p]\ \{S_2\}$ | $p \cdot$最弱前置条件$[\![S_1]\!](X)\ + (1-p) \cdot$最弱前置条件$[\![S_2]\!](X)$ |
| 如果 $(b)\ \{S_1\}$ 否则 $\{S_2\}$ | $[b] \cdot$最弱前置条件$[\![S_1]\!](X)\ + [\neg b] \cdot$最弱前置条件$[\![S_2]\!](X)$ |
| while循环 $(b)\ \{\ S'\ \}$ | 最小不动点$Y.\ [b] \cdot (1 +$最弱前置条件$[\![S']\!](Y)) + [\neg b] \cdot X$ |

图9：程序的期望运行时间变换器的归纳定义$S$，期望运行时间$[\![S]\!]$：$\mathbb{E} \to \mathbb{E}$ 其中 $X \in \mathbb{E}$ 是后运行时间。与标准最弱前置条件语义（图4）的唯一区别在于 while 循环中的“$+1$”以及缺少奖励语句。

**引理 2（增量运行时间收集）**。设无循环 $S' \in$ pGCL和 $b \in$布尔表达式。则，最弱前置条件$[\![\texttt{while}\ (b)\ \{\ \texttt{reward}\ 1;\ S'\ \}]\!]$ = 运行时间$[\![\texttt{while}\ (b)\ \{\ S'\ \}]\!]$。

证明。 我们考虑 ert$[\![S]\!]$ 中采用仅统计循环迭代次数的运行时模型的期望运行时语义。图 9 给出了 ert$[\![S]\!]$的归纳定义，它与标准的最弱前置条件语义仅在 while 循环情形中多了“$+1$”，且没有 奖励语句。

考虑循环 $\texttt{while}\ (b)\ \{\ S'\ \}$。我们证明对于所有期望 $X \in \mathbb{E}$:

$$\mathsf{wp}[\![\texttt{while}\ (b)\ \{\ \texttt{reward}\ 1;\ S'\ \}]\!](X)\ =\ \mathsf{ert}[\![\texttt{while}\ (b)\ \{\ S'\ \}]\!](X)\ .$$

对于程序化奖励循环，我们有:

$$\mathsf{wp}[\![\texttt{while}\ (b)\ \{\ \texttt{reward}\ 1;\ S'\ \}]\!](X)$$

| $S$ | $\mathcal{T}_f(S)$ |
|---|---|
| $S$ | $\tau :=0;$ 奖励 $f(0);\ \ \mathcal{T}'_f(S)$ |

| $S$ | $\mathcal{T}'_f(S)$ |
|---|---|
| `skip` | `skip` |
| 奖励 $a$ | 奖励 $(f(\tau+a)-f(\tau));\ \ \tau :=\tau+a$ |
| $x :=a$ | $x :=a$ |
| $\{S_1\}\ [p]\ \{S_2\}$ | $\{\mathcal{T}'_f(S_1)\}\ [p]\ \{\mathcal{T}'_f(S_2)\}$ |
| 如果 $(b)\ \{S_1\}$ 否则 $\{S_2\}$ | 如果 $(b)\ \{\mathcal{T}'_f(S_1)\}$ 否则 $\{\mathcal{T}'_f(S_2)\}$ |
| $S_1;\ S_2$ | $\mathcal{T}'_f\ (S_1)\ ;\ \mathcal{T}'_f\ (S_2)$ |
| while循环 $(b)\ \{\ S'\ \}$ | while循环 $(b)\ \{\ \mathcal{T}'_f(S')\ \}$ |

图 10：针对单调性 $f$ 的程序变换 $\mathcal{T}_f(S)$ 和辅助变换 $\mathcal{T}'_f(S)$：$\mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 以及在 $S$ 中未出现的新变量 $\tau$。

$$
\begin{aligned}
&= \mathrm{lfp}\, Y.\ [b]\cdot \mathsf{wp}[\![\texttt{reward } 1;\ S']\!](Y) + [\neg b]\cdot X \\
&= \mathrm{lfp}\, Y.\ [b]\cdot \mathsf{wp}[\![\texttt{reward } 1]\!](\mathsf{wp}[\![S']\!](Y)) + [\neg b]\cdot X \\
&= \mathrm{lfp}\, Y.\ [b]\cdot (1+\mathsf{wp}[\![S']\!](Y)) + [\neg b]\cdot X\ ,
\end{aligned}
$$

这等于期望运行时语义：

$$\mathsf{ert}[\![\texttt{while}\ (b)\ \{\ S'\ \}]\!](X)$$

最后一个等式成立是因为 $\mathsf{ert}[\![S']\!]$ =最弱前置条件$[\![S']\!]$ 因为 $S'$ 无循环。

通过对程序结构进行归纳，并将 奖励 语句加入每个循环体中，可以将此引理扩展到嵌套循环。□

### A.3 第 4 节证明

这里更详细地陈述定义 1 中的程序变换函数，以便于下文的证明。图 10 定义了程序变换 $\mathcal{T}_f(S)$ 和辅助变换 $\mathcal{T}'_f(S)$。（外层）变换 $\mathcal{T}_f(S)$ 仅添加变量 $\tau$，将其初始化为 0，前置奖励 $f(0)$，并调用辅助变换 $\mathcal{T}'_f(S)$。后者负责变换奖励语句，并递归地将该变换应用于 $S$ 的子语句。每个奖励语句 $a$ 都会被变换为奖励 $(f(\tau+a)-f(\tau))$，而变量 $\tau$ 则更新为 $\tau :=\tau+a$。其他语句保持不变。

定理 1 指出，变换 $\mathcal{T}_f(S)$ 能够根据 $f$正确变换 $S$ 的期望奖励。定理 1 的证明大多是技术性的；主要的概念性论证见定理 6 (参见 第 4.2 节)。

**定理 1（可靠性）**。对于 $S \in$ pGCL 和单调 $f$：$\mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$:

$$\mathsf{E}(\mathsf{rew}(\mathcal{T}_f(S))) = \mathsf{E}(f(\mathsf{rew}(S)))\ .$$

证明。令 $\sigma_1 \in$状态为任意初始状态。定义

$$\mathcal{M}_{\mathsf{prog}} \coloneqq \mathcal{M}^{\sigma_1}_{\mathcal{T}_f(S)}, \qquad \mathcal{M}_{\mathsf{mc}} \coloneqq \mathcal{T}_f(\mathcal{M}^{\sigma_1}_{S}) \ .$$

我们比较这两个马尔可夫链，其中 $\mathcal{M}_{\mathsf{mc}}$ 是通过定理 6中的马尔可夫链变换从 $f(0) = 0$ 得到的。定理 6要求我们假设 $f(0) = 0$ 成立，但引理 4 表明，可以通过另一个简单的变换解除这一限制。马尔可夫链 $\mathcal{M}_{\mathsf{prog}}$ 和 $\mathcal{M}_{\mathsf{mc}}$ 本质上是相同的，区别仅在于以下几点：

– $\mathcal{M}_{\mathsf{prog}}$ 的状态是程序状态 $\sigma \in$ 状态（扩展了新变量 $\tau$），而 $\mathcal{M}_{\mathsf{mc}}$的状态是对（$\sigma$， $\tau$） $\in$状态 $\times \mathbb{R}^{\infty}_{\geq 0}$。这些是对相同状态的等价表示。– 奖励语句 $a$ 被变换为奖励（$f$（$\tau + a$ $-$ $f$（$\tau$））） 和 $\tau$：$= \tau + a$的顺序组合。设 $\sigma_i$是执行 $S$ 过程中某个时刻$i$ 的状态，设 $\alpha_i$ 是截至该时刻收集的累积奖励。在奖励语句处， $\mathcal{M}_{\mathsf{mc}}$ 将执行一步：

$$((\texttt{reward}\ (f(\tau + a) - f(\tau)), \sigma_i), \alpha_i) \ .$$

另一方面， $\mathcal{M}_{\mathsf{prog}}$中有三个步骤：

$$\begin{aligned} &(\texttt{reward}\ (f(\tau + a) - f(\tau))\texttt{;}\ \tau \coloneqq \tau + a, \sigma_i') \\ &\xrightarrow{1} (\Downarrow\texttt{;}\ \tau \coloneqq \tau + a, \sigma_i') \\ &\xrightarrow{1} (\tau \coloneqq \tau + a, \sigma_i') \ . \end{aligned}$$

然而，这些路径的概率和累积奖励相等。

因此，我们可以得出结论：两个马尔可夫链的期望奖励相等：

$$\mathsf{E}(\mathsf{rew}(\mathcal{M}_{\mathsf{prog}})) \ = \ \mathsf{E}(\mathsf{rew}(\mathcal{M}_{\mathsf{mc}})) \ .$$

根据定理 6， $\mathcal{M}_{\mathsf{mc}}$ 满足

$$\mathsf{E}(\mathsf{rew}(\mathcal{M}_{\mathsf{mc}})) \ = \ \mathsf{E}(f(\mathsf{rew}(\mathcal{M}^{\sigma_1}_{S}))) \ .$$

因此，对于所有$\sigma_1 \in$状态，我们有 $\mathsf{E}(f(\mathsf{rew}^{\sigma_1}(S))) = \mathsf{E}(f(\mathsf{rew}(\mathcal{M}^{\sigma_1}_{S}))) = \mathsf{E}(\mathsf{rew}(\mathcal{M}^{\sigma_1}_{\mathcal{T}_f(S)}))$，这等价于 $\mathsf{E}(\mathsf{rew}(\mathcal{T}_f(S))) = \mathsf{E}(f(\mathsf{rew}(S)))$。□

**定理 2（单调性）**。设 $S \in$pGCL 并设 $f$， $g$： $\mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$是单调的，且 $f \preceq g$。对于所有 $X \in \mathbb{E}$，我们有最弱前置条件$[\![\mathcal{T}_f(S)]\!](X) \preceq$最弱前置条件$[\![\mathcal{T}_g(S)]\!](X)$。

证明。 设 $S \in$pGCL 且设 $f, g$： $\mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$为单调函数，且满足$f(x) \leq g(x)$ 对所有 $x \in \mathbb{R}^{\infty}_{\geq 0}$。我们对 $S$ 进行结构归纳，证明

$$\forall X \in \mathbb{E}. \quad \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X) \leq \mathsf{wp}[\![\mathcal{T}'_g(S)]\!](X) \ . \tag{2}$$

基本情形:

- $S$ =空操作:

$$\text{最弱前置条件}[\![\mathcal{T}_f'(\text{空操作})]\!](X) = \text{最弱前置条件}[\![\text{空操作}]\!](X) = X$$

$$= \text{最弱前置条件}[\![\text{空操作}]\!](X) = \text{最弱前置条件}[\![\mathcal{T}_g'(\text{空操作})]\!](X)$$

- $S = x :\!= a$:

$$\text{最弱前置条件}[\![\mathcal{T}_f'(x :\!= a)]\!](X) = \text{最弱前置条件}[\![x :\!= a]\!](X) = X[x \mapsto a]$$

$$= \text{最弱前置条件}[\![x :\!= a]\!](X) = \text{最弱前置条件}[\![\mathcal{T}_g'(x :\!= a)]\!](X)$$

- $S$ =奖励 $a$:

$\text{最弱前置条件}[\![\mathcal{T}'_f(\text{奖励 } a)]\!](X)$
$=$最弱前置条件$[\![$奖励 $(f(\tau+a) - f(\tau))$; $\tau :\!= \tau + a]\!](X)$（$\mathcal{T}_f'$（奖励 $a$）的定义）$=(f(\tau+a) - f(\tau)) + X[\tau \mapsto \tau + a]$（最弱前置条件的定义）$\leq (g(\tau+a) - g(\tau)) + X[\tau \mapsto \tau + a]$（$f$ 和 $g$ 的单调性）$=$ 最弱前置条件 $[\![$ 奖励 $(g(\tau+a) - g(\tau))$; $\tau :\!= \tau + a]\!](X)$（最弱前置条件的定义）$=$最弱前置条件$[\![\mathcal{T}_g'($奖励 $a)]\!](X)$（$\mathcal{T}_g'$（奖励 $a$）的定义）

现在假设归纳假设对 $S_1$、 $S_2$和 $S' \in$ pGCL成立。

- $S = S_1 ; S_2$:

$$\begin{aligned}
&\text{最弱前置条件}[\![\mathcal{T}_f'(S_1 ; S_2)]\!](X) \\
&= \text{最弱前置条件}[\![\mathcal{T}_f'(S_1) ; \mathcal{T}_f'(S_2)]\!](X) && (\mathcal{T}_f'(\cdot)\text{的定义}) \\
&= \text{最弱前置条件}[\![\mathcal{T}_f'(S_1)]\!](\text{最弱前置条件}[\![\mathcal{T}_f'(S_2)]\!](X)) && (\text{最弱前置条件的定义}) \\
&\leq \mathsf{wp}[\![\mathcal{T}_g'(S_1)]\!](\mathsf{wp}[\![\mathcal{T}_g'(S_2)]\!](X)) && (\text{归纳假设}) \\
&= \mathsf{wp}[\![\mathcal{T}_g'(S_1) ; \mathcal{T}_g'(S_2)]\!](X) && (\text{最弱前置条件的定义}) \\
&= \mathsf{wp}[\![\mathcal{T}_g'(S_1 ; S_2)]\!](X) && (\mathcal{T}_g'(\cdot)\text{的定义})
\end{aligned}$$

- $S = \{S_1\}\ [p]\ \{S_2\}$:

$$\begin{aligned}
&\text{最弱前置条件}[\![\mathcal{T}_f'(\{S_1\}\ [p]\ \{S_2\})]\!](X) \\
&= \text{最弱前置条件}[\![\{\mathcal{T}_f'(S_1)\}\ [p]\ \{\mathcal{T}_f'(S_2)\}]\!](X) && (\mathcal{T}_f'(\cdot)\text{的定义}) \\
&= p \cdot \text{最弱前置条件}[\![\mathcal{T}_f'(S_1)]\!](X) + (1-p) \cdot \text{最弱前置条件}[\![\mathcal{T}_f'(S_2)]\!](X) && (\text{最弱前置条件的定义}) \\
&\leq p \cdot \text{最弱前置条件}[\![\mathcal{T}_g'(S_1)]\!](X) + (1-p) \cdot \text{最弱前置条件}[\![\mathcal{T}_g'(S_2)]\!](X) && (\text{归纳假设}) \\
&= \text{最弱前置条件}[\![\{\mathcal{T}_g'(S_1)\}\ [p]\ \{\mathcal{T}_g'(S_2)\}]\!](X) && (\text{最弱前置条件的定义}) \\
&= \text{最弱前置条件}[\![\mathcal{T}_g'(\{S_1\}\ [p]\ \{S_2\})]\!](X) && (\mathcal{T}_g'(\cdot)\text{ 的定义})
\end{aligned}$$

- $S$ =如果 $(b)$ $\{S_1\}$ 否则 $\{S_2\}$:

$$\text{最弱前置条件}[\![\mathcal{T}_f'(\text{如果 } (b)\ \{S_1\}\ \text{否则}\ \{S_2\})]\!](X)$$

$$
\begin{aligned}
&= \mathsf{wp}[\![\texttt{if } (b) \ \{\mathcal{T}_f'(S_1)\} \texttt{ else } \{\mathcal{T}_f'(S_2)\}]\!](X) && \text{(definition of } \mathcal{T}_f'(\cdot)) \\
&= [b] \cdot \mathsf{wp}[\![\mathcal{T}_f'(S_1)]\!](X) + [\neg b] \cdot \mathsf{wp}[\![\mathcal{T}_f'(S_2)]\!](X) && \text{(definition of wp)} \\
&\leq [b] \cdot \mathsf{wp}[\![\mathcal{T}_g'(S_1)]\!](X) + [\neg b] \cdot \mathsf{wp}[\![\mathcal{T}_g'(S_2)]\!](X) && \text{(归纳假设)} \\
&= \text{最弱前置条件}[\![\text{如果 } (b) \ \{\mathcal{T}_g'(S_1)\} \text{ 否则 } \{\mathcal{T}_g'(S_2)\}]\!](X) && \text{(definition of wp)} \\
&= \mathsf{wp}[\![\mathcal{T}_g'(\texttt{if } (b) \ \{S_1\} \texttt{ else } \{S_2\})]\!](X) && \text{(definition of } \mathcal{T}_g'(\cdot))
\end{aligned}
$$

- $S = \text{while 循环 } (b) \ \{ \ S' \ \}$：回顾一下，最弱前置条件$[\![\text{while 循环 } (b) \ \{ \ S' \ \}]\!](X)$ 被定义为泛函的最小不动点

$$\Phi_X(Y) = [b] \cdot \mathsf{wp}[\![S']\!](Y) + [\neg b] \cdot X \ .$$

根据克莱尼不动点定理，我们有

$$\mathsf{wp}[\![\mathcal{T}_f'(\texttt{while } (b) \ \{ \ S' \ \})]\!](X) = \sup_{n \in \mathbb{N}} {\Phi_X}^n(0) \ .$$

记 $\Phi_X$ 为变换后 while 循环 $\mathcal{T}_f'(\texttt{while}\,(b) \ \{ \ S' \ \})$ 的泛函，并记 $\Psi_X$ 为 $\mathcal{T}_g'(\texttt{while}\,(b) \ \{ \ S' \ \})$ 的泛函。

我们对 $n$ 进行归纳法证明，证明 ${\Phi_X}^n(0) \leq {\Psi_X}^n(0)$ 对所有 $n \in \mathbb{N}$均成立。基础情形 $n = 0$成立，因为 ${\Phi_X}^0(0) = 0 = {\Psi_X}^0(0)$。对于归纳步骤，假设 ${\Phi_X}^n(0) \leq {\Psi_X}^n(0)$。于是，

$$
\begin{aligned}
&{\Phi_X}^{n+1}(0) = \Phi_X({\Phi_X}^n(0)) \\
&= [b] \cdot \mathsf{wp}[\![\mathcal{T}_f'(S')]\!]({\Phi_X}^n(0)) + [\neg b] \cdot X && \text{(definition of } \Phi_X) \\
&\leq [b] \cdot \mathsf{wp}[\![\mathcal{T}_g'(S')]\!]({\Psi_X}^n(0)) + [\neg b] \cdot X \\
& && \text{(induction hypothesis and monotonicity of } \mathsf{wp}[\![\cdot]\!]) \\
&= \Psi_X(\Psi_X{}^n(0)) && \text{(definition of } \Psi_X) \\
&= {\Psi_X}^{n+1}(0) \ .
\end{aligned}
$$

因此，根据归纳法，对于所有 $n \in \mathbb{N}$，我们有 ${\Phi_X}^n(0) \leq {\Psi_X}^n(0)$。由此可得

$$
\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}_f'(\texttt{while } (b) \ \{ \ S' \ \})]\!](X) = \sup_{n \in \mathbb{N}} {\Phi_X}^n(0) \\
&\leq \sup_{n \in \mathbb{N}} {\Psi_X}^n(0) = \mathsf{wp}[\![\mathcal{T}_g'(\texttt{while } (b) \ \{ \ S' \ \})]\!](X) \ .
\end{aligned}
$$

归纳法至此结束。最后，对于完整变换 $\mathcal{T}_f(S)$，我们有

$$
\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}_f(S)]\!](X) \\
&= \mathsf{wp}[\![\tau \coloneqq 0\texttt{; reward } f(0)\texttt{; } \mathcal{T}_f'(S)]\!](X) && \text{(definition of } \mathcal{T}_f(S)) \\
&= f(0) + \mathsf{wp}[\![\mathcal{T}_f'(S)]\!](X)[\tau \mapsto 0] && \text{(definition of wp)} \\
&\leq g(0) + \mathsf{wp}[\![\mathcal{T}_f'(S)]\!](X)[\tau \mapsto 0] && (f(0) \leq g(0)) \\
&\leq g(0) + \mathsf{wp}[\![\mathcal{T}_g'(S)]\!](X)[\tau \mapsto 0] && \text{(Equation (2))} \\
&= \mathsf{wp}[\![\tau \coloneqq 0\texttt{; reward } g(0)\texttt{; } \mathcal{T}_g'(S)]\!](X) && \text{(definition of wp)} \\
&= \text{最弱前置条件}[\![\mathcal{T}_g(S)]\!](X) && \text{(定义 } \mathcal{T}_g(S))
\end{aligned}
$$

□

**定理 3（线性）**。设 $S \in \mathsf{pGCL}$ 且设 $f\colon \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 为单调的。那么，对于所有 $\alpha$，$\beta \in \mathbb{R}_{\geq 0}$，我们有 最弱前置条件$[\![\mathcal{T}_{\alpha\cdot f+\beta}(S)]\!](0) = \alpha \cdot$ 最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) + \beta$。

证明。设 $f\colon \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 单调且 $\alpha$，$\beta \in \mathbb{R}_{\geq 0}$。记作 $g = \alpha \cdot f + \beta$。

首先，我们证明辅助变换 $\mathcal{T}'_g(S)$ 保持标量乘法。对于所有 $S \in \mathsf{pGCL}$ 和 $X \in \mathbb{E}$，有

$$\mathsf{wp}[\![\mathcal{T}'_g(S)]\!](\alpha \cdot X) = \alpha \cdot \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X)\ .$$

证明通过对 $S$进行结构归纳法得出。有趣的情况是 $S =$奖励 $a$：

$$\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}'_g(\texttt{reward}\ a)]\!](\alpha \cdot X)\\
&= \mathsf{wp}[\![\texttt{reward}\ g(\tau+a) - g(\tau);\ \tau \coloneqq \tau + a]\!](\alpha \cdot X) && \text{(def. } \mathcal{T}'_g(S)\text{)}\\
&= (g(\tau+a) - g(\tau)) + \alpha \cdot X[\tau \mapsto \tau + a] && \text{(def. wp)}\\
&= ((\alpha \cdot f(\tau+a) + \beta) - (\alpha \cdot f(\tau) + \beta)) + \alpha \cdot X[\tau \mapsto \tau + a] && \text{(def. } g\text{)}\\
&= \alpha \cdot (f(\tau+a) - f(\tau)) + \alpha \cdot X[\tau \mapsto \tau + a] && \text{(cancellation of } \beta\text{)}\\
&= \alpha \cdot (f(\tau+a) - f(\tau) + X[\tau \mapsto \tau + a]) && \text{(linearity)}\\
&= \alpha \cdot \mathsf{wp}[\![\mathcal{T}'_f(\texttt{reward}\ a)]\!](X) && \text{(def. of wp)}
\end{aligned}$$

其余情况的证明过程显而易见。

对于完整变换 $\mathcal{T}_g(S)$，有

$$\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}_g(S)]\!](\alpha \cdot X)\\
&= \mathsf{wp}[\![\tau \coloneqq 0;\ \texttt{reward}\ g(0);\ \mathcal{T}'_g(S)]\!](\alpha \cdot X) && \text{(def. } \mathcal{T}_g(S)\text{)}\\
&= g(0) + \mathsf{wp}[\![\mathcal{T}'_g(S)]\!](\alpha \cdot X)[\tau \mapsto 0] && \text{(def. wp)}\\
&= \alpha \cdot f(0) + \beta + \mathsf{wp}[\![\mathcal{T}'_g(S)]\!](\alpha \cdot X)[\tau \mapsto 0] && \text{(def. } g\text{)}\\
&= \alpha \cdot f(0) + \alpha \cdot \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X)[\tau \mapsto 0] + \beta && \text{(above)}\\
&= \alpha \cdot \mathsf{wp}[\![\mathcal{T}_f(S)]\!](X) + \beta && \text{(def. } \mathcal{T}'_f(S)\text{)}
\end{aligned}$$

使用 $X = 0$，陈述 最弱前置条件$[\![\mathcal{T}_g(S)]\!](0) = \alpha \cdot$ 最弱前置条件$[\![\mathcal{T}_f(S)]\!](0) + \beta$ 即可得出。 □

**定理 4（幽灵消除器）**。令 $f\colon \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 具有形式 $f(x) = \alpha \cdot x + \beta$，其中 $\alpha$、$\beta \in \mathbb{R}_{\geq 0}$为任意项。那么，

$$\mathcal{T}_f(S) = \texttt{reward}\ \beta;\ \mathcal{T}'_f(S) \qquad \text{and} \qquad \mathcal{T}'_f(\texttt{reward}\ a) = \texttt{reward}\ \alpha \cdot a\ .$$

证明。令 $S \in \mathsf{pGCL}$，$f\colon \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 使得 $f(x) = \alpha \cdot x + \beta$ 对于某个$\alpha$，$\beta \in \mathbb{R}_{\geq 0}$。令 $X \in \mathbb{E}$ 使得 $\tau$不出现在 $X$中，即 $\forall \sigma \in$ 状态 $v \in$取值. $X[\tau \mapsto v] = X$。

对于 $\mathcal{T}'_S(f)$ 对 奖励 语句的变换，我们有：

$$
\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}'_f(\texttt{reward}\ a)]\!](X) \\
&= \mathsf{wp}[\![\texttt{reward}\ (f(\tau+a)-f(\tau));\ \tau \coloneqq \tau + a]\!](X) && \text{(definition of } \mathcal{T}'_f(\texttt{reward}\ a)) \\
&= (f(\tau+a)-f(\tau)) + X[\tau \mapsto \tau + a] && \text{(definition of wp)} \\
&= (\alpha\cdot(\tau+a)+\beta-(\alpha\cdot\tau+\beta)) + X[\tau\mapsto\tau+a] && \text{(definition of } f) \\
&= \alpha\cdot a + X[\tau\mapsto\tau+a] && \text{(cancellation of } \beta) \\
&= \alpha\cdot a + X && (\tau \text{ not in } X) \\
&= \mathsf{wp}[\![\texttt{reward}\ \alpha\cdot a]\!](X) && \text{(definition of wp)}
\end{aligned}
$$

对于完整变换 $\mathcal{T}_f(S)$，我们有：

$$
\begin{aligned}
&\text{最弱前置条件}[\![\mathcal{T}_f(S)]\!](X)(\sigma) \\
&= \text{最弱前置条件}[\![\tau\coloneqq 0;\ \text{奖励}\ \ f(0);\ \mathcal{T}'_f(S)]\!](X)(\sigma) && \text{(definition of } \mathcal{T}_f(S)) \\
&= f(0) + \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X)[\tau\mapsto 0](\sigma) && \text{(definition of wp)} \\
&= f(0) + \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X) && (\tau\text{不在 最弱前置条件}[\![\mathcal{T}'_f(S)]\!](X)) \\
&= (\alpha\cdot 0+\beta) + \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X) && \text{(definition of } f) \\
&= \beta + \mathsf{wp}[\![\mathcal{T}'_f(S)]\!](X) && \text{(simplification)} \\
&= \text{最弱前置条件}[\![\text{奖励}\ \ \beta;\ \mathcal{T}'_f(S)]\!](X) && \text{(definition of wp)}
\end{aligned}
$$

因此，该简化是有效的，并且对于线性变换可以省略 $\tau$ 。□

**定理 5（变换的复合）**。设 $S \in \mathsf{pGCL}$，并设$f$，$g$：$\mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$是单调的。假设 $\mathcal{T}'_f(\cdot)$ 和 $\mathcal{T}'_{g\circ f}(\cdot)$ 使用相同的变量 $\tau$，而 $\mathcal{T}'_g(\cdot)$ 使用不同的变量 $\tau'$。则，

$$\mathsf{wp}[\![\tau' \coloneqq f(\tau);\ \mathcal{T}'_g(\mathcal{T}'_f(S))]\!] = \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S);\ \tau' \coloneqq f(\tau)]\!]\ .$$

证明。根据最弱前置条件的定义，只需证明

$$\mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S))]\!](X)[\tau'\mapsto f(\tau)] = \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S)]\!](X[\tau'\mapsto f(\tau)])\ .$$

我们对 $S$ 的结构进行归纳法。

在基本情形中，$S$ =奖励 $a$ 是值得关注的：

$$
\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(\texttt{reward}\ a))]\!](X)[\tau'\mapsto f(\tau)] \\
&= \mathsf{wp}[\![\mathcal{T}'_g(\texttt{reward}\ (f(\tau+a)-f(\tau));\ \tau\coloneqq\tau+a)]\!](X)[\tau'\mapsto f(\tau)] \\
&\qquad\qquad \text{(definition of } \mathcal{T}'_f(\cdot)) \\
&= \mathsf{wp}\left[\!\!\left[\begin{array}{l}\texttt{reward}\ (g(\tau'+(f(\tau+a)-f(\tau)))-g(\tau'));\\ \tau'\coloneqq\tau'+(f(\tau+a)-f(\tau));\\ \tau\coloneqq\tau+a\end{array}\right]\!\!\right](X)[\tau'\mapsto f(\tau)] \\
&\qquad\qquad \text{(definition of } \mathcal{T}'_g(\cdot))
\end{aligned}
$$

$$
\begin{aligned}
&= (g(\tau' + (f(\tau+a) - f(\tau))) - g(\tau') \\
&\quad + X[\tau \mapsto \tau + a][\tau' \mapsto \tau' + (f(\tau+a) - f(\tau))][\tau' \mapsto f(\tau)] && \text{(definition of wp)} \\
&= g(f(\tau) + (f(\tau+a) - f(\tau))) - g(f(\tau)) \\
&\quad + X[\tau \mapsto \tau + a][\tau' \mapsto f(\tau) + (f(\tau+a) - f(\tau))] && \text{(substitution)} \\
&= g(f(\tau+a)) - g(f(\tau)) + X[\tau \mapsto \tau + a][\tau' \mapsto f(\tau+a)] \\
& && (f(\tau) + (f(\tau+a) - f(\tau)) = f(\tau+a)) \\
&= \mathsf{wp}[\![\texttt{reward}\ (g(f(\tau+a)) - g(f(\tau)));\ \tau' \coloneqq f(\tau+a);\ \tau \coloneqq \tau + a]\!](X) \\
& && \text{(definition of wp)} \\
&= \mathsf{wp}[\![\texttt{reward}\ (g(f(\tau+a)) - g(f(\tau)));\ \tau \coloneqq \tau + a;\ \tau' \coloneqq f(\tau)]\!](X) \\
& && \text{(exchange assignments)} \\
&= \mathsf{wp}[\![\texttt{reward}\ (g(f(\tau+a)) - g(f(\tau)));\ \tau \coloneqq \tau + a]\!](X[\tau' \mapsto f(\tau)]) \\
& && \text{(definition of wp)} \\
&= \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(\texttt{reward}\ a)]\!](X[\tau' \mapsto f(\tau)]) && \text{(definition of } \mathcal{T}'_{g\circ f}(\cdot))
\end{aligned}
$$

其他基本情形都很简单。对于归纳情形，假设该命题对 $S_1$ 和 $S_2$均成立。我们来证明顺序组合 $S = S_1;\ \ S_2$:

$$
\begin{aligned}
&\mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_1;\ S_2))]\!](X)[\tau' \mapsto f(\tau)] \\
&= \mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_1);\ \mathcal{T}'_f(S_2))]\!](X)[\tau' \mapsto f(\tau)] && \text{(definition of } \mathcal{T}'_f(\cdot)) \\
&= \mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_1));\ \mathcal{T}'_g(\mathcal{T}'_f(S_2))]\!](X)[\tau' \mapsto f(\tau)] && \text{(definition of } \mathcal{T}'_g(\cdot)) \\
&= \mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_1))]\!](\mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_2))]\!](X))[\tau' \mapsto f(\tau)] && \text{(definition of wp)} \\
&= \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S)]\!](\mathsf{wp}[\![\mathcal{T}'_g(\mathcal{T}'_f(S_2))]\!](X)[\tau' \mapsto f(\tau)]) && \text{(induction hypothesis for } S_1) \\
&= \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S)]\!](\mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S_2)]\!](X[\tau' \mapsto f(\tau)])) && \text{(induction hypothesis for } S_2) \\
&= \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S_1);\ \mathcal{T}'_{g\circ f}(S_2)]\!](X[\tau' \mapsto f(\tau)]) && \text{(definition of wp)} \\
&= \mathsf{wp}[\![\mathcal{T}'_{g\circ f}(S)]\!](X[\tau' \mapsto f(\tau)]) && \text{(definition of } \mathcal{T}'_{g\circ f}(\cdot))
\end{aligned}
$$

其他归纳情形都很直接。因此，该结论由归纳法得出。□

**定理 6（可靠性）**。对于每个马尔可夫链 $\mathcal{M}$ 和单调的 $f$:

$$\mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) = \mathsf{E}(\mathsf{rew}'(\mathcal{T}_f(\mathcal{M}))) \ .$$

证明。我们将原始马尔可夫链中每条有限路径 $\pi = \sigma_1 \ldots \sigma_n$ $\mathcal{M}$ 与一条有限路径 $\pi' = \sigma'_1 \ldots \sigma'_n$ 建立一一对应关系，该路径位于变换后的马尔可夫链中 $\mathcal{T}_f(\mathcal{M})$ 。我们利用下文引理 3 的两个性质来证明这两条路径的期望奖励相等。

(1) $\pi$ 和 $\pi'$ 具有相同的概率：$\mathsf{Prob}(\pi) = \mathsf{Prob}'(\pi')$。(3) 经过 $f$ 变换后的奖励等于 $\pi'$ 的奖励： $f(\mathsf{rew}(\pi)) = \mathsf{rew}'(\pi')$。

我们得到：

$$\begin{aligned}\mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) &= \sup_{n\in\mathbb{N}} \sum_{\pi\in\mathsf{Paths}^{=n}_{\mathcal{M}}(\sigma)} \mathsf{Prob}(\pi)\cdot f(\mathsf{rew}(\pi)) \\ = \mathsf{E}(\mathsf{rew}'(\mathcal{T}_f(\mathcal{M}))) &= \sup_{n\in\mathbb{N}} \sum_{\pi'\in\mathsf{Paths}^{=n}_{\mathcal{T}_f(\mathcal{M})}(\sigma)} \mathsf{Prob}'(\pi')\cdot \mathsf{rew}'(\pi')\ . \qquad \square\end{aligned}$$

**引理 3.** 设马尔可夫链 $\mathcal{M} = (\text{状态}, \text{概率}\ \sigma_1, \text{奖励})$ 且 $f\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 为单调的，满足 $f(0) = 0$。设 $n \in \mathbb{N}$。对于每个 $\sigma_1 \dots \sigma_n \in \text{路径}^{=n}_{\mathcal{M}}$，都存在一个一一映射至 $(\sigma_1, \alpha_1)\dots(\sigma_n, \alpha_n) \in \text{路径}^{=n}_{\mathcal{T}_f(\mathcal{M})}$ 的映射，且该映射属于 $\mathcal{T}_f(\mathcal{M})$，使得：

*1.* 路径概率相同：$\text{概率}(\sigma_1\dots\sigma_n) = \text{概率}'((\sigma_1,\alpha_1)\dots(\sigma_n,\alpha_n))$，*2.* $\alpha_i$ 追踪先前的未变换奖励：$\forall 1 \le i \le n.\ \sum_{j<i} \text{奖励}(\sigma_j) = \alpha_i$，*3.* $= \dots$。变换后奖励：$f(\text{奖励}(\sigma_1\dots\sigma_n)) \text{奖励}'((\sigma_1,\alpha_1)(\sigma_n,\alpha_n))$。

证明。设 $\mathcal{M} = (\mathsf{States}, \mathsf{Prob}\ \sigma_1, \mathsf{rew})$ 是一个马尔可夫链，且设 $f\colon \mathbb{R}^{\infty}_{\geq 0} \to \mathbb{R}^{\infty}_{\geq 0}$ 是一个单调函数，满足 $f(0) = 0$。

对每个 $n \geq 1$，定义

$$\xi_n\colon \mathsf{Paths}^{=n}_{\mathcal{M}} \to \mathsf{Paths}^{=n}_{\mathcal{T}_f(\mathcal{M})},$$

$$\xi_n(\sigma_1\dots\sigma_n) \coloneqq (\sigma_1,\alpha_1)\dots(\sigma_n,\alpha_n),$$

其中 $\alpha_1 = 0$，且对所有 $1 \le i \le n$，$\sigma_1\dots\sigma_n \in \mathsf{Paths}^{=n}_{\mathcal{M}}$，$\mathsf{rew}(\xi_n\ \pi) = \pi = \alpha_i = \sum_{j<i}$。

对于每条路径 Paths，我们对 作归纳法，证明 ( ) 定义良好且满足性质 (1)–(3)。

基础情形。对于 $n = 1$，设 $\sigma_1 \in \mathsf{Paths}^{=1}_{\mathcal{M}}$。那么，$\xi_1(\sigma_1) = (\sigma_1,\ 0) \in \mathsf{Paths}^{=1}_{\mathcal{T}_f(\mathcal{M})}$，并且性质 (1)–(3) 成立：

1. $\mathsf{Prob}'((\sigma_1,0)) = 1 = \mathsf{Prob}(\sigma_1)$, 2. $\alpha_1 = 0 = \sum_{j<1} \mathsf{rew}(\sigma_j)$,
3. $\mathsf{rew}'((\sigma_1,0)) = f(0 + \mathsf{rew}(\sigma_1)) - f(0) = f(\mathsf{rew}(\sigma_1))$,

归纳步骤。现在假设该结论对长度为 $n$ 的情形成立，并令 $\pi = \sigma_1\dots\sigma_{n+1} \in \text{路径}^{=n+1}_{\mathcal{M}}$。我们将截至 $n$ 步的截断路径 $\pi$ 定义为 $\pi^- = \sigma_1\dots\sigma_n$。于是，$\xi_n(\pi^-) = (\sigma_1,\ \alpha_1)\dots(\sigma_n,\ \alpha_n)$。由 $\alpha_{n+1} \coloneqq \alpha_n + \mathsf{rew}(\sigma_n)$，我们有

$$\xi_{n+1}(\pi) = (\sigma_1,\alpha_1)\dots(\sigma_n,\alpha_n)(\sigma_{n+1},\alpha_{n+1}).$$

路径 $\xi_{n+1}(\pi)$ 是 $\mathcal{T}_f(\mathcal{M})$ 中的一条有效路径，因为根据 $\mathsf{Prob}'$ 的定义，$(\sigma_n,\ \alpha_n)$ 唯一具有非零概率的后继状态恰好是 $(\sigma_{n+1},\ \alpha_{n+1})$。此外，我们证明 $\xi_{n+1}(\pi)$ 满足性质 (1)–(3)：

1. 相同的路径概率：

$$\begin{aligned}\mathsf{Prob}'(\xi_{n+1}(\pi)) &= \mathsf{Prob}'(\xi_n(\pi^-))\cdot\mathsf{Prob}'((\sigma_n,\alpha_n),(\sigma_{n+1},\alpha_{n+1})) \\ &= \mathsf{Prob}(\pi^-)\cdot\mathsf{Prob}'((\sigma_n,\alpha_n),(\sigma_{n+1},\alpha_{n+1})) \qquad (\text{IH } (1)) \\ &= \mathsf{Prob}(\pi^-)\cdot\mathsf{Prob}(\sigma_n)(\sigma_{n+1}) \qquad (\text{def. } \mathsf{Prob}') \\ &= \mathsf{Prob}(\pi)\ .\end{aligned}$$

2. $\alpha_i$ 跟踪未变换奖励：根据归纳假设 (2)，我们有 $\alpha_n = \sum_{j<n} \mathsf{rew}(\sigma_j)$。因此，我们有 $\alpha_{n+1} = \left(\sum_{j<n} \mathsf{rew}(\sigma_j)\right) + \mathsf{rew}(\sigma_n) = \sum_{j<n+1} \mathsf{rew}(\sigma_j)$。3. 变换后奖励：

$$
\begin{aligned}
&\mathsf{rew}'(\xi_{n+1}(\pi)) \\
&= \mathsf{rew}'(\xi_n(\pi^-)\,(\sigma_{n+1}, \alpha_{n+1})) && (\text{def. } \xi_{n+1}) \\
&= \mathsf{rew}'(\xi_n(\pi^-)) + \mathsf{rew}'((\sigma_{n+1}, \alpha_{n+1})) && (\text{def. } \mathsf{rew}') \\
&= f(\mathsf{rew}(\pi^-)) + \mathsf{rew}'((\sigma_{n+1}, \alpha_{n+1})) && (\text{IH } (3)) \\
&= f(\mathsf{rew}(\pi^-)) + (f(\alpha_{n+1} + \mathsf{rew}(\sigma_{n+1})) - f(\alpha_{n+1})) && (\text{def. } \mathsf{rew}') \\
&= f(\text{反向}(\pi^-)) + (f((\alpha_n + \text{反向}(\sigma_n)) + \text{反向}(\sigma_{n+1})) - f(\alpha_n + \text{反向}(\sigma_n))) \\
& && (\text{def. } \alpha_{n+1}) \\
&= f(\mathsf{rew}(\pi^-)) + \left(f(\mathsf{rew}(\pi^-) + \mathsf{rew}(\sigma_{n+1})) - f(\mathsf{rew}(\pi^-))\right) && (\text{IH } (2)) \\
&= f(\mathsf{rew}(\pi^-)) + \left(f(\mathsf{rew}(\pi)) - f(\mathsf{rew}(\pi^-))\right) && (\text{def. } \mathsf{rew}(\pi)) \\
&= f(\mathsf{rew}(\pi))\ 。
\end{aligned}
$$

因此，$\xi_{n+1}$ 是良定义的，并且满足(1)–(3)。此外，$\xi_{n+1}$ 是一个双射。为证明单射性，设 $\pi$，$\pi' \in$ 路径$^{=n+1}_{\mathcal{M}}$ 满足 $\xi_{n+1}(\pi) = \xi_{n+1}(\pi')$。根据$\xi_{n+1}$的定义，可得 $\pi = \pi'$。对于满射性，设 $\rho = (\sigma_1,\ \beta_1) \ldots (\sigma_{n+1},\ \beta_{n+1}) \in$路径$^{=n+1}_{\mathcal{T}_f\ \mathcal{M}()}$ 为任意的。有 $\xi_{n+1}(\sigma_1 \ldots \sigma_{n+1}) = \rho$。
□

为了取消限制 $f(0) = 0$ 于 定理 6中，我们使用以下引理。它使用平移函数 $g(x) = f(x) - f(0)$ 进行变换，并前置一个奖励为 $f(0)$ 的初始状态到变换后的马尔可夫链。

引理 *4*（取消限制 $f(0) = 0$）。考虑马尔可夫链$\mathcal{M} = (\text{状态}, \mathit{Prob}\ \sigma_1, \mathit{rew})$，并令 $f\colon\ \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 为单调函数。令$g\colon\ \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ 定义为 $g(x) = f(x) - f(0)$。由定理 *6*，我们得到 $g$-变换后的马尔可夫链 $\mathcal{T}_g(\mathcal{M}) = (\text{状态} \times \mathbb{R}^\infty_{\geq 0}, \mathit{Prob}', \sigma_1', \mathit{rew}')$。为了插入一个奖励为 $f(0)$ 的初始状态，令 $\mathcal{M}'' = (\text{状态}'', \mathit{Prob}'', \sigma_0'', \mathit{rew}'')$ 如下：

- $\mathsf{States} \times \mathbb{R}^\infty_{\geq 0} \cup \{\ \sigma_0''\ \}$,
- $\mathsf{Prob}''(\sigma, \sigma') = \begin{cases} 1 & \mathit{if}\ \sigma = \sigma_0' \wedge \sigma' = (\sigma_1, 0) \\ \mathsf{Prob}'(\sigma, \sigma') & \mathit{otherwise} \end{cases}$
- $\mathsf{rew}''(\sigma_0'') = f(0)$ *and otherwise* $\mathsf{rew}''((\sigma, \alpha)) = \mathsf{rew}'((\sigma, \alpha))$.

于是，有

$$\mathsf{E}(\mathsf{rew}(\mathcal{M}'')) = \mathsf{E}(f(\mathsf{rew}(\mathcal{M})))\ .$$

证明。由于 $g$是单调的，且 $g(0) = f(0) - f(0) = 0$，我们可以应用 定理 6，得到 $g$-变换后的马尔可夫链 $\mathcal{T}_g(\mathcal{M})$。在扩展马尔可夫链中，每条

路径的奖励序列前面都加上了 $f(0)$。因此，

$$\begin{aligned}\mathsf{E}(\mathsf{rew}(\mathcal{M}'')) &= \mathsf{E}(f(0) + \mathsf{rew}'(\mathcal{T}_g(\mathcal{M}))) \\ &= \mathsf{E}(f(0) + g(\mathsf{rew}(\mathcal{M}))) && \text{(by Theorem 6)} \\ &= \mathsf{E}(f(0) + f(\mathsf{rew}(\mathcal{M})) - f(0)) && \text{(definition of } g\text{)} \\ &= \mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) && \square\end{aligned}$$

### A.4 推广到多个奖励（第 5.5 节）

我们概述结果如何推广到多个奖励，对应于第 5.5 节。固定 $n \geq 1$ 和单调的 $f$: $(\mathbb{R}^{\infty}_{\geq 0})^{n} \to \mathbb{R}^{\infty}_{\geq 0}$。

多奖励概率程序和马尔可夫链。在操作语义和最弱前期望语义中（第2节），奖励语句变为元组值形式：奖励（$a_1$，..., $a_n$）。因此，马尔可夫链带有元组值奖励：

$$\mathcal{M} = (\mathsf{States}, \mathsf{Prob}, \sigma_1, \mathsf{rew}), \qquad \mathsf{rew} \colon \mathsf{States} \to (\mathbb{R}^{\infty}_{\geq 0})^{n}.$$

有限路径 $\pi = \sigma_1 \dots \sigma_k$ 的奖励为 $\mathsf{rew}(\pi) := \sum_{i=1}^{k} \mathsf{rew}(\sigma_i) \in (\mathbb{R}^{\infty}_{\geq 0})^{n}$。对于每个 $i$，令 $S_i$ 由 $S$ 替换每个元组奖励为其第 $i$ 个分量得到。于是，元组值最弱前置期望变换器按分量定义如下：

$$\begin{gathered}\mathsf{wp}^{n} \colon \mathsf{pGCL} \times \mathbb{E}^{n} \to \mathbb{E}^{n}, \\ \mathsf{wp}^{n}[\![S]\!](X_1, \dots, X_n) := (\mathsf{wp}[\![S_1]\!](X_1), \dots, \mathsf{wp}[\![S_n]\!](X_n)).\end{gathered}$$

推广该变换。回想一下，奖励程序变换（定义 1）按第 5.5 节所述进行扩展：我们引入新的幽灵变量 $\tau_1$、.....、$\tau_n$ 并定义

$$\mathcal{T}_f(S) := \tau_1 := 0;\ \cdots;\ \tau_n := 0;\ \texttt{reward}\ f(0, \dots, 0);\ \mathcal{T}'_f(S)\ ,$$

其中，对于奖励语句，内部变换定义如下：

$$\begin{aligned}\mathcal{T}'_f(\texttt{reward}\ (a_1, \dots, a_n)) := \texttt{reward}\ f(\tau_1 + a_1, \dots, \tau_n + a_n) - f(\tau_1, \dots, \tau_n); \\ \tau_1 := \tau_1 + a_1;\ \cdots;\ \tau_n := \tau_n + a_n\ .\end{aligned}$$

其他情况的定义与单奖励情况类似（参见 图 10），通过递归应用变换即可得到。同样，马尔可夫链变换（定义 2）可扩展如下，假设 $f(0, \dots, 0) = 0$:

$$\begin{aligned}\mathcal{T}_f(\mathcal{M}) &= (\mathsf{States} \times (\mathbb{R}^{\infty}_{\geq 0})^{n},\ \mathsf{Prob}',\ (\sigma_1, (0, \dots, 0)),\ \mathsf{rew}'), \\ \mathsf{Prob}'((\sigma, \overline{\tau}), (\sigma', \overline{\tau}')) &= \begin{cases}\mathsf{Prob}(\sigma)(\sigma') & \text{if } \overline{\tau}' = \overline{\tau} + \mathsf{rew}(\sigma), \\ 0 & \text{otherwise,}\end{cases} \\ \mathsf{rew}'((\sigma, \overline{\tau})) &= f(\overline{\tau} + \mathsf{rew}(\sigma)) - f(\overline{\tau})\ ,\end{aligned}$$

马尔可夫链变换的可靠性（引理 3 和 4）可直接推广到元组值累积奖励，并沿用标量情形中的相同路径耦合和望远镜求和论证，由此得到

$$\mathsf{E}(f(\mathsf{rew}(\mathcal{M}))) = \mathsf{E}(\mathsf{rew}'(\mathcal{T}_f(\mathcal{M}))).$$

定理 1，即 $\mathsf{E}(f(\mathsf{rew}(S))) = \mathsf{E}(\mathsf{rew}(\mathcal{T}_f(S)))$，由相同的论证可得。

## B 案例研究

我们在此补充介绍正文中提到的示例。对于使用 *Caesar* 验证器 [35]，验证的示例，我们提供了相应的 *HeyVL* 语言代码。这些示例在未经修改的 Caesar 版本（3.0.0 版）上运行，并利用其通过 最弱前置条件-based reasoning about 期望奖励 with奖励语句的支持。

***HeyVL* 语法。** 下面简要介绍以下示例中使用的 HeyVL 语法。更多详细信息可参见相关论文 [35] 以及在线文档（ `https://caesarverifier.org/docs/heyvl/`）。

在下面的示例中，我们使用 Caesar 的语法在 HeyVL 中声明两类过程，过程 和 协程。这两类声明均由名称、输入和输出变量列表、前置条件 $X$、后置条件 $Y$和主体 $S$组成。 过程 的语法如下，其中 `x` 是类型为 整数 的输入变量，`y` 是类型为 整数的输出变量：

```
过程 示例过程 (x: 整数) -> (y: 整数)
        前置条
件 X₃         后
置条件 Y₄ {
        S₆ }
```

唯一的区别在于过程与 协程之间：验证过程满足关于期望值的下界 $X$，即 $Y$。

$$X \preceq \mathsf{wp}[\![S]\!](Y)\ ,$$

而且 协程 则必须满足对偶的上界 性质，即

$$\mathsf{wp}[\![S]\!](Y) \preceq X\ .$$

为了实现概率选择，HeyVL 使用普通赋值的语法，并搭配一个分布表达式 翻转($p$)，其以真的概率为 $p$，并以假的概率为 $1-p$。

### B.1 快速掷骰算法的期望访问时间（第 3.3 节）

为了说明第 3.3 节中期望访问时间的编码，我们考虑快速掷骰算法(FDR)，这是一种运行时间最优算法，用于从 $[1, N]$ [26]中进行均匀采样。

对于 $N = 6$，Mertens等人 [29] 分析了图 11所示马尔可夫链上的期望访问时间。在该示例中，他们计算到达使算法终止的状态 $s_3$、 $s_4$、 $s_6$ 的期望访问时间，并由此推导各骰面的期望可达概率。我们使用 Caesar 验证期望访问时间的相同界，方法是将马尔可夫链编码为简单的while循环，其中包含概率选择和奖励语句。

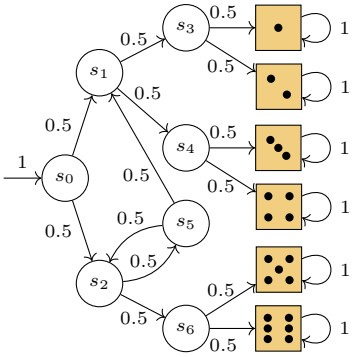



图 11：来自 [29]的快速掷骰算法（FDR）的马尔可夫链。

图 12展示了该编码的 HeyVL 代码。变量`s` 表示状态索引，`done` 指示是否已到达终止状态，`res` 表示采样得到的骰面。为查询期望访问时间

对于特定状态，我们使用幽灵变量`query_s` 和`query_done`，并使用`reward` [`query s` $= s \wedge$ `query done` $=$ `done`]收集奖励。协程声明位于图 12 中，配合`pre` $1/3$ 表明，我们验证上界 $1/3$，即到达状态 $s \in \{0, \ldots, 6\}$ 与 $done \in \{$`true`、`false`$\}$的期望访问时间。

根据这些界，我们可以得到终止状态的期望可达概率界（参见 [29, 第 6.1 节。1]）。由于相关状态$s_3$， $s_4$， $s_6$ 的期望访问时间至多为 $1/3$，且从这些状态转移到终止状态的概率均为 $1/2$，因此每个终止状态的期望可达概率至多为 $1/6$。还可以通过添加更多幽灵变量来指示骰子点数的具体值（结果变量），并相应地修改奖励语句，来验证这一结论。

为简洁起见，图 12 中省略了不变式，但在图 13中单独给出。它针对每个查询状态和每个当前状态，规定了从当前状态到查询状态的期望访问时间上界。它对应于基本矩阵，其定义见 [23, 定义 3.2。2], 该矩阵最初被定义为马尔可夫链中期望访问时间的确定方法。

我们还通过 Caesar 确认，我们的不变式是最优的。这是利用 Caesar 内置的、与用于证明上界的 wp Park 归纳证明规则对偶的规则完成的；现在我们验证下界。由于下界和上界相等，因此该不变式是最优的。

### B.2 Web 服务器方差（示例 2）

对于图 1中的 Web 服务器示例，我们对响应时间的期望值和方差作出了断言。关于计算的一些细节已在示例 2中给出，我们在此补充一些精选的计算过程。

我们关注未修改的示例，即图 1a。对于while循环，相对于后置条件 0，其循环特征泛函为

$$\Phi_0(X) \;=\; [\neg done]\cdot((2\cdot\tau+1)+0.5\cdot X[done\mapsto \texttt{true}]+0.5\cdot X)+[done]\cdot 0.$$

为了计算最小不动点，我们使用克莱尼不动点定理：

$$\mathsf{wp}[\![\texttt{while}\ (\neg done)\ \{\ \ldots\ \}]\!](0) \;=\; \sup_{n\in\mathbb{N}}\Phi_0^n(0).$$

前几次迭代得出以下结果：

$$\begin{aligned}
\Phi_0^0(0) &= 0,\\
\Phi_0^1(0) &= [\neg done]\cdot(2\cdot\tau+1),\\
\Phi_0^2(0) &= [\neg done]\cdot(1/2\cdot(6\cdot\tau+5)),\\
\Phi_0^3(0) &= [\neg done]\cdot(1/4\cdot(14\cdot\tau+15)),\\
\Phi_0^4(0) &= [\neg done]\cdot(1/8\cdot(30\cdot\tau+37))\ .
\end{aligned}$$

由此，我们可以推导出以下闭式：

$$\Phi_0^n(0) \;=\; [\neg done]\cdot\left(\frac{1}{2^{n-1}}\cdot((2^{n+1}-2)\cdot\tau+3\cdot(2^n-1)-2\cdot n)\right)\ .$$

我们利用单调收敛定理计算上确界：

$$\begin{aligned}
\sup_{n\in\mathbb{N}}\Phi_0^n(0) &= \lim_{n\to\infty}\Phi_0^n(0)\\
&= [\neg done]\cdot(4\cdot\tau+6)\ .
\end{aligned}$$

因此，我们证明了 wp[while $(\neg done)\ \{\ \ldots\ \}$](0) = $[\neg done]\cdot(4\cdot\tau+6)$。完整程序 $S$ 以 $\tau=0$ 和 $done=$ false 开始，因此 wp[$\mathcal{T}_{x^2}(S)$](0) $=6$。对于随机变量 $X$，我们有 Var($X$) $=$E($X^2$) $-$E($X$)$^2$。因此，我们计算得 wp[$\mathcal{T}_{x^2}(S)$](0) $-$wp[$S$](0)$^2 = 6-2^2=2$。

对于带缓存的示例程序（图 1b），记作 $S'$，我们得到最小不动点 $[\neg done]\cdot(3\cdot(\tau-0.5)+4.5)$，并得到二阶矩 wp[$\mathcal{T}_{x^2}(S')$](0) $=0.25+4.5=4.75$。由于 wp[$S'$](0) $=2$，我们得到方差 $4.75-2^2=0.75$。

相应的 HeyVL 代码如图 14所示。图 14a 和 14b分别验证了未修改示例和修改后示例的期望运行时间上界 2。图 14c和 14d验证了二阶矩上界 6 和 4。75，分别。

### B.3 有偏随机游走（例 3）

回顾一下示例 3 （见第5.1节），其中我们考虑一个从某个初始值 init x 开始、定义在非负整数上的有偏随机游走。以 0。75 的概率，将变量 x 减小 2；以 0。25 的概率，将其增加 2。每一步之后，我们将奖励增加 1。当x达到零时，过程停止。

累积奖励一阶矩的 HeyVL 代码如图 16所示。给出的上界是`init_x - (init_x \% 2)`，这意味着步数的期望值不超过x的初始值向下取整到最接近的偶数。

对于二阶矩，我们将变换 $\mathcal{T}'_f(\cdot)$ 应用于原始程序，其中 $f(x) = x^2$。变换后的程序如图 16所示。与之前一样，原始的奖励语句被替换为奖励$(\texttt{tau}+1)^2 - \texttt{tau}^2$，并且赋值$\texttt{tau} := \texttt{tau} + 1$。我们使用不变式 $x^2 + 2 \cdot x \cdot \texttt{tau} + 3 \cdot x$来验证 $x^2 + 3 \cdot x$的上界。

### B.4 累积分布函数（例4）

对于图 1a中的 Web 服务器示例，第 5.2 节中的例 4验证了运行时间累积分布函数的一个上界。我们通过应用参数为 $f(x) = [x \geq N]$的变换来编码这一目标。在变换后的程序（图 17）中， $\tau$跟踪运行时间，并且每次奖励增量为

$$[\tau + 1 \geq N] - [\tau \geq N] \quad = \quad [\tau + 1 = N] \ ,$$

因此，只有当运行首次到达步骤 $N$ 时，才会获得奖励。我们使用以下循环不变式来验证 Pr(runtime $\geq N$) 至多为 $0.5^{N \ominus 1}$:

$$I \ = \ [\neg \mathit{done} \wedge \tau + 1 \leq N] \cdot \left(\frac{1}{2}\right)^{N-(\tau+1)} \ .$$

请注意，这个不变式需要用到指数函数，而 Caesar 开箱即用的功能并不支持指数函数，但可以通过用户定义函数来编码。具体做法是使用`domain` 指数函数声明，其中包含 `pow` 的递归定义，用于模拟函数 $a^b$ 对 $a \in \mathbb{R}_{\geq 0}$ 和 $b \in \mathbb{N}$。

### B.5 期望超额（例5）

回顾第 5.3 节中的例 5：对于图 1a中的 Web 服务器循环，每轮成功概率为 $p$，预算为 $N$，目标是E(运行时间 $\ominus N$）。

图 18 编码原始程序，图 19 编码变换后的单循环变体，其中 $f(x) = x \ominus N$（因此增量为 $[\tau \geq N]$），而图 20编码拆分后的两阶段变体。两种变换后的编码都证明了相同的上界$\overline{(1-p)_p}^{\,N}$，对所有 $0 < p < 1$ 均成立。同样，变换后程序的不变式需要用指数函数，这些指数函数通过用户定义函数编码，具体如前一个示例所述。

### B.6 抛硬币的矩母函数（示例 6）

回顾 示例 6 中的 第 5.4 节：对于 $S = \{x : = 1\}\ [p]\ \{x : = 0\}$；奖励 $x$ 和 $f(x) = e^{t \cdot x}$，目标是将矩母函数的上界设为 $p \cdot e^t + (1 - p)$。

图 21 给出了相应的 Caesar 证明在 HeyVL 中的编码。其 域 矩母函数声明引入了一个无解释的 `mgf` ，其定义域为 $e^{t \cdot x}$ ，并满足公理`mgf(0)=1` 以及严格单调性（`x < y ==> mgf(x)< mgf(y)`）。这些公理固定了初始奖励，并排除了诸如 `mgf(0)=1, mgf(1)=0` 这样的反模型，其中`mgf(tau + x)- mgf(tau)` 在 `UReal` 中被解释为`max(0, mgf(tau + x)-mgf(tau))` ，并且增量被截断为0。

### B.7 多个奖励（示例7）

回顾一下，示例 7 在第 5.5 节中：我们跟踪运行时间和内存，二者初始值均为 2；随后以概率 $p$ 增加运行时间，并以概率 $q$增加额外内存。图 22 给出了相应的 Caesar 证明在 HeyVL 中的编码。使用 $f(memory,\ runtime) = memory \cdot runtime$，每次更新都会产生乘积增量差 $f(memory',\ runtime') - f(memory,\ runtime)$。该文件验证了预期综合成本的上界 $4 + (p \cdot (2 + (q \cdot 3)))$。

```
1 协程 fdr6 ( query_s : 无符号整数 , query_done : 布尔值 ) -> (res :
    无符号整数 ) 2 前置 !?( 查询_完成 == 真 && 查询_s <= 6) 3 前置
1/3 4 后置 0 5 { 6 变量 s: 无符号整数 = 07 变量 完成: 布尔值 = 假 8
@不变式 (...) 9 while循环 ! 完成 { 10 奖励 [ 查询_s ==s && 查询_完成
 ==完成 ]11 变量 选择 : 布尔值 ; 变量 旧_s : 无符号整数 =s 12 如果 旧_s
 == 0{ 13 选择 = 翻转 (0.5) 14 如果 选择 { s = 1} 否则 { s = 2 }
15 } 否则 {}; 如果 旧_s == 1{ 16 选择 = 翻转 (0.5) 17 如果 选择 { s = 3}
否则 { s = 4 } 18 } 否则 {}; 如果 旧 _s == 2{ 19 选择 = 翻转 (0.5)
20 如果 选择 { s = 5} 否则 { s = 6 } 21 } 否则 {}; 如果 旧_s == 3{
22 选择 = 翻转 (0.5) 23 如果 选择 { 结果 = 1 } 否则 { 结果 = 2 }
24 完成 =真 25 } 否则 {}; 如果 旧_s == 4{ 26 选择 = 翻转 (0.5) 27 如
果 选择 { 结果 = 3 } 否则 { 结果 = 4 } 28 完成 =真 29 } 否则 {}; 如
果 旧_s == 5{ 30 选择 = 翻转 (0.5) 31 如果 选择 { s = 1} 否则 { s
 = 2 } 32 } 否则 {}; 如果 旧_s == 6{ 33 选择 = 翻转 (0.5) 34 如果 选
择 { 结果 = 5 } 否则 { 结果 = 6 } 35 完成 =真 36 } 否则 {} 37 } 38
奖励 [ 查询_s ==s && 查询_完成 ==完成 ]39 }
```

图 12：用于验证 FDR示例中期望访问时间的 HeyVL 代码。不变式在图 13中给出。

```
1 条件表达式 ( 2 完成 ,3       /* then */ [(( 查询_s  == s) && ( 查询_完成
  == 真 ))] ,4       /* else */5 条件表达式 ( 6 ( 查询 _ 完成  == 假 ) ,
7       /* then */8            [ 查询_s ==0] *  [s==0]9            + [查询_
s ==1] * (  [s ==0]*(2/3)  + [s ==1]*(1) +[s ==2]*(1/3)  + [s==5]*(2/3) )
10            + [查询_s ==2] * (  [s ==0]*(2/3)  + [s ==2]*(4/3)  +[s==5]
*(2/3) )11            + [查询_s ==3] * (  [s ==0]*(1/3)  + [s ==1]*(1/2)  +[s
==2]*(1/6)  + [s==3]*1  + [s ==5]*(1/3) )12            + [查询_s ==4] * (  [
s ==0]*(1/3)  + [s ==1]*(1/2)  +[s ==2]*(1/6)  + [s==4]*1  + [s ==5]*(1/3) )
13            + [查询_s ==5] * (  [s ==0]*(1/3)  + [s ==2]*(2/3)  +[s==5]
*(4/3) )14            + [查询_s ==6] * (  [s ==0]*(1/3)  + [s ==2]*(2/3)  +[s
==5]*(1/3)  + [s ==6]*1 ),15       /* else */16            [查询_s ==3] * (  [
s ==0]*(1/3)  + [s ==1]*(1/2)  + [s==2]*(1/6)  + [s==3]*1  + [s ==5]
*(1/3) )17            + [查询_s ==4] * (  [s ==0]*(1/3)  + [s ==1]*(1/2)  +[s
==2]*(1/6)  + [s==4]*1  + [s ==5]*(1/3) )18            + [查询_s ==6] * (  [
s ==0]*(1/3)  + [s ==2]*(2/3)  +[s ==5]*(1/3)  + [s==6]*1 ) 19 ) 20 )
```

图13：用于 图 12 中 FDR 期望访问时间验证的不变式。

```
1协程 original_moment1 () ->( done : 布尔值) 2 pre 2 3 post 0 4 { 5 done = false 6 @不变式 ( [! done ] * 2) 7 while循环 ! done { 8 奖励 1 9 done = 翻转 (0.5) 10 } 11 }
```

(a) HeyVL 代码，用于验证未修改的 Web 服务器示例的期望运行时间（图 1a）的上界为 2。

```
1 协程 caching_moment1 () -> ( done : 布尔值) 2 pre 2 3 post 0 4 { 5 done = false 6 奖励 0.5 7 @不变式 ( [! done ] * 1.5) 8 while循环 ! done { 9 奖励 1 10 done = 翻转 (2/3) 11 } 12 }
```

(b) 用于验证修改后的 Web 服务器示例（图 1b）的预期运行时间上限为 2 的 HeyVL 代码。

```
1协程 original_moment2 () ->( done : 布尔值 ) 2 pre 6 3 post 0 4 { 5 done = false 6 var tau: 无符号整数 = 07 @不变式 ( [! done ] * ( 6 + 4 * tau)) 8 while循环 ! done { 9 奖励 2 * tau +1 10 tau = tau + 111 done = 翻转 (0.5) 12 } 13 }
```

(c) 用于验证未修改的 Web 服务器示例（图 1a）的运行时间二阶矩不超过 6 的 HeyVL 代码。

```
1 协程 caching_moment2 () -> ( done : 布尔值 ) 2 前置条件 4.75 3 后置条件 0 4 { 5 done = false 6 变量 tau: UReal = 0.5 7 奖励 0.25 8 @不变式 ( [! done ] * (4.5 + 3 * ( tau - 0.5) )) 9 while循环 ! done { 10 奖励 2 * tau +1 11 tau = tau + 112 done = 翻转 (2/3) 13 } 14 }
```

（d）用于验证修改后的 Web 服务器示例的运行时间二阶矩（图 1b）上界为 4。75。

图 14：用于验证有关以下内容的 HeyVL 代码 图 1。

```
1 协程 random_walk（ init_x：无符号整数）->（x：无符号整数）
2 pre ite（ init_x > 1, init_x -（ init_x % 2）, 0）3 post 0 4 { 5 x
= init_x 6 @不变式（ite（x > 1, x -（x % 2）, 0））7 while循
环 x > 1 { 8 var prob_choice：布尔值 =翻转（0.75）9 if prob_
choice { 10 x = x - 2 11 } else { 12 x =x + 213 } 14 奖励 1 15 } 16 }
```

图 15：表示初始位置为 init x 的有偏随机游走的 HeyVL 代码。

```
1 协程 random_walk（ init_x：无符号整数）->（x：无符号
整数）2 前置条件 init_x * init_x + 3* init_x 3 后置条件 0 4 { 5
x = init_x 6 变量 tau：无符号整数 = 07 @不变式（x*x + 2
*x* tau + 3* x）8 while循环 x > 1 { 9 变量 prob_choice：
布尔值 =翻转（0.75）10 如果 prob_choice { 11 x = x - 2 12 }
否则 { 13 x =x + 214 } 15 奖励（ tau +1） *( tau +1) - tau *
tau 16 tau = tau + 117 } 18 }
```

图 16：应用变换函数 $f(x) = x^2$ 后，用于计算到达 $x \leq 1$ 的时间的二阶矩的 HeyVL 代码。

```
1Exponentials域 { 2 函数 pow（base：UReal，exponent：无符号整数）：
UReal =ite（exponent == 0，1，base * pow（base，exponent - 1））3 }
4 5 协程 original_server_cdf（N：无符号整数）->（done：布尔值）6 前置条件
pow（5/10，N - 1）7 后置条件 0 8 { 9 变量 tau：无符号整数 = 010 奖励 [0 >=
N]11 12 done = false 13 @不变式（14 ite（! done && tau + 1 <=N，15
pow（0.5，N -（tau + 1））， 16 0 17）18）19 while循环 ! done { 20 奖励 [
tau + 1 == N]21 tau = tau + 122 done = 翻转（0.5）23 } 24 }
```

图 17：用于验证以下示例中 Web 服务器累积分布函数上界的 HeyVL 代码：例4，该示例来自图 1a。

```
1Exponentials域 { 2 func pow ( base : UReal , exponent : UInt ): UReal = ite ( exponent == 0, 1, base * pow ( base , exponent - 1) ) 3 } 4 5 协程 expected_excess (p: UReal ) -> () 6 pre !?( p > 0 && p < 1) 7 pre 1/p 8 post 0 9 { 10 var done : 布尔值 = false 11 @不变式 ( ite (! done , 1/p , 0) ) 12 while循环 ! done { 13 done = 翻转 (p) 14 奖励 1 15 } 16 }
```

图18：表示原始程序的 HeyVL 代码，来自 例5。

```
1Exponentials域 { 2 func pow ( base : UReal , exponent : UInt ): UReal = ite ( exponent == 0, 1, base * pow ( base , exponent - 1)) 3 } 4 5 协程 expected_excess (N: UInt , p: UReal ) -> () 6 pre !?( p > 0 && p < 1) 7 pre pow (1 -p, N)/p 8 post 0 9 { 10 var tau : UInt = 011 var done : 布尔值 = false 12 @不变式 ( ite (! done , pow (1 -p , N - tau )/p , 0) ) 13 while循环 ! done { 14 奖励 [tau >= N ]15 tau = tau + 116 done = 翻转 (p) 17 } 18 }
```

图19：应用变换函数 $f(x) = x \ominus N$ 后的 HeyVL 代码。

```
Exponentials域 { func pow ( base : UReal , exponent : UInt ): UReal = ite ( exponent == 0, 1, base * pow ( base , exponent - 1)) } 协程 expected_excess (N: UInt , p: UReal ) ->() pre !?( p > 0 && p < 1) pre pow (1 -p, N)/p post 0 { var tau : UInt = 0 var done : 布尔值 = false @不变式 ( ite (! done && tau <N , pow (1 -p , N - tau )/p , ite (! done , 1/p , 0) )) while循环 ! done && ( tau <N) { done = 翻转 (p) 奖励 0 tau = tau + 1 } @不变式 ( ite (! done , 1/p , 0) ) while循环 ! done { done = 翻转 (p) 奖励 1 tau = tau + 1 } }
```

图 20：拆分循环并应用变换函数 $f(x) = x \ominus N$ 后的 HeyVL 代码。

```
域 矩母函数 {
    // uninterpreted function representing e^{t·x}3 函数 矩母函数 (x: UReal ): UReal 4 公理 矩母函数_零 矩母函数 (0) == 15 公理 矩母函数_递增 对所有 x: UReal , y: UReal . (x <y) ==>( 矩母函数 (x) <矩母函数 (y)) 6 } 7 8 协程 随机_赋值_矩母函数 (p: UReal ) ->(x: 无符号整数 ) 9 前置条件 !?( p <= 1) 10 前置条件 p * 矩母函数 (1) +(1 -p) 11 { 12 变量 tau : 无符号整数 = 013 奖励 矩母函数 (0) 14 15 变量 概率 _ 选择 : 布尔值 = 翻转 (p) 16 如果 概率 _ 选择 { 17 x = 118 } 否则 { 19 x = 020 } 21 22 奖励 矩母函数 ( tau +x) - 矩母函数 ( tau ) 23 tau = tau +x 24 }
```

图 21：应用变换函数 $f(x) = e^{t \cdot x}$ 后的 HeyVL 代码，用于计算概率为 $p$ 的抛硬币的矩母函数。

```
协程 多个_奖励（p: UReal , q: UReal）-> ()
    pre !?( p <= 1 && q <= 1)
    pre （4 + （p * （2 + （q * 3) )))
{
    变量 tau_内存: 无符号整数 = 2
    变量 tau _运行时间: 无符号整数 = 2
    奖励 tau_内存 * tau_运行时间

    var 概率_选项: 布尔值 =翻转(p)
    if 概率_选项 {

        // extra computation
        奖励 (tau_内存 + 0) *( tau_运行时间 + 1) -(
            tau_内存 * tau_运行时间 )
        tau_内存 = tau_内存
        tau_运行时间 =tau _运行时间 + 1

        概率_选择 = 翻转（q）
        if 概率_选择 {

            // extra memory
            奖励 （ tau_内存 + 1) *( tau_运行时间 + 0) -(
                tau_内存 * tau_运行时间 )
            tau_内存 = tau_内存 + 1
            tau_运行时间 = tau _运行时间

        } else {}
    } else {}
}
```

图 22：通过乘法组合两个奖励结构的程序的 HeyVL 代码。

# C 相关工作详述

## C.1 与摊销期望运行时间演算的关系

文献 [7] 提出了一种称为摊销期望运行时间演算的期望运行时间演算扩展。其 aert变换器计算势函数 $\pi$ 的期望变化：状态 $\to \mathbb{R}^{\infty}_{\geq 0}$，即一个期望。根据 [7, 定理 5.4]：$\mathsf{aert}_{\tau}[\![S]\!](X) =$ 最弱前置条件$[\![S]\!](X+\pi)-\pi$。因此，aert 在摊销运行时间的格 $\mathbb{A}$ 上运算，这些运行时间可以是负期望，但其下界为势函数的相反数，即 $\mathbb{A} = \{\, X\colon$ 状态 $\to (\mathbb{R} \cup \infty) \mid X \geq -\pi \,\}$。

然而，aert 变换器对势函数的分析只考虑势函数的期望值在终止时。这由他们的定理5.4 推出：势函数 $\pi$ 仅在终止时才加到期望中（参见 第 3.2 节）。

进一步说明这一点，考虑无穷循环 $\mathsf{while\ (true)}\ \{\, \tau \coloneqq \tau + 1 \,\}$，它使用赋值 $\tau \coloneqq \tau + 1$ 在每次迭代中将运行时间变量 $\tau$ 增加 1。我们尝试分析计数变量 $\tau$的势能变化，即选择 $\pi = \tau$。我们有 $\mathsf{aert}_{\tau}[\![\mathsf{while\ (true)}\ \{\, \tau \coloneqq \tau + 1 \,\}]\!](0) = -\tau$，这意味着无穷循环不会导致期望运行时间发生任何变化。要理解为什么该循环的语义求值为 $-\pi$，只需注意，while 循环的 aert语义被定义为函数 $\Phi_0(X) = X[\tau \mapsto \tau + 1] + ((\tau + 1) - \tau) = X[\tau \mapsto \tau + 1] + 1$的最小不动点。格 $\mathbb{A}_{\pi}$ 的最小元素是 $-\tau$，但它已经是一个不动点，因为$\Phi_0(-\tau) = (-(\tau + 1)) + 1 = -\tau$。

因此，势函数 $\pi$ （即 aert 变换器所用的势函数）无法在没有奖励语句的情况下用于跟踪运行时间，而人们可以改用平凡势函数 $\pi = 0$ 来应用我们的变换。