Time debits in nested thunks: a proof of Okasaki's banker's queue

Glen Mével, François Pottier, Jacques-Henri Jourdan

Inria Paris & LMF Paris-Saclay, France

2nd Iris workshop, May 2–6, 2022, Nijmegen

We can implement an immutable queue using two lists *front* and *rear*:

$$\textbf{type}\ '\alpha\ \mathit{queue} = '\alpha\ \mathit{list} \times '\alpha\ \mathit{list}$$

```
let snoc (front, rear) x =
  (front, x :: rear)                          – insert into rear list

let pop (front, rear) =
  match front with                            – if front is non-empty...
  | x :: front′ → Some (x, (front′, rear))    – ...pop its head
  | [] →                                      – otherwise...
      match List.rev rear with                – ...reverse rear to front (costly)...
      | x :: front′ → Some (x, (front′, []))  – ...and pop head
      | [] → None
```

Time cost of the purely functional queue

- *snoc*: worst time $\mathcal{O}(1)$
- *pop*: worst time $\mathcal{O}(n)$

Say that each function call "costs" \$1; then:

- *snoc* costs \$1 at worst
- *pop* costs $\$(n+1)$ at worst

Too pessimistic! The "banker's method" (Tarjan, 1985) gives constant amortized costs:

- *snoc* costs \$2

  we save \$1 *extra for each inserted element, covering for that element's reversal*

- *pop* costs \$1

  reversal is pre-paid by snocs, so we only need \$1 *more for the call to pop itself*

Issue: we can't spend time credits twice

let q = *snoc* (*snoc* (*snoc nil* 1) 2) 3 **in**
let (x_1, q_1) = *pop q* **in** – *we spend our savings here*
let (x_2, q_2) = *pop q* **in** – *wrong! we don't have any savings anymore*
...

$\Longrightarrow$ Amortized complexity breaks if an old version of the queue is used

Idea (Okasaki, 1999):

1. compute reversals once $\Longrightarrow$ memoize them
2. share reversals among futures $\Longrightarrow$ suspend them ahead of time

$\Longrightarrow$ **Laziness**! We use a **stream**, i.e., a list computed on-demand

type $'\alpha$ *stream* = $'\alpha$ *cell thunk*
and $'\alpha$ *cell* = *Nil* | *Cons* **of** $'\alpha \times {}'\alpha$ *stream*

Tradeoff: suspending too early would create too many thunks

```
type 'α queue = int × 'α stream × int × 'α list
```

We enforce that $|f| \geq |r|$:

```
let rebalance ((lenf, f, lenr, r) as q) =
  assert (lenf+1 ≥ lenr) ;
  if lenf ≥ lenr then q else     – re-establish inv. when r grows larger than f:
    (lenf+lenr, Stream.append f (Stream.rev_of_list r), 0, [])
                                 – ↑ create a thunk that will reverse r when forced

let snoc (lenf, f, lenr, r) x =
  rebalance (...)                – rebalance with element inserted into r

let pop (lenf, f, lenr, r) =
  match Stream.pop f with        – force the head thunk of f
  ... rebalance (...) ...        – rebalance with head removed from f
```

Time cost of the banker's queue

Reversing $|r|$ elements is costly, but is done after $|f|$ elements are popped

$\Longrightarrow$ We can **anticipate** the cost of reversal on that of the previous *pop*s

$\Longrightarrow$ Because $|f| \geq |r|$, each *pop* absorbs a constant cost

$\Longrightarrow$ Everything is in constant amortized time:

- *rebalance* costs \$1 at worst
- *snoc* costs \$2 at worst
- *pop* costs (e.g.) \$5 amortized

The banker's queue can be used **persistently**

Key idea:

- the non-lazy queue saves **credit** for an unknown future computation
  $\Longrightarrow$ not duplicable (cannot forge money)
- the banker's queue repays a **debit** for a known past computation
  $\Longrightarrow$ duplicable (can waste money)

Soundness: you get nothing until you are done repaying (debit $\neq$ loan)

Basic building blocks: **thunks**, holding debits:

$$\mathit{isThunk}\ t\ m\ \varphi$$

Ownership is duplicable:

$$\mathit{isThunk}\ t\ m\ \varphi \quad -\!\!*\quad \mathit{isThunk}\ t\ m\ \varphi \star \mathit{isThunk}\ t\ m\ \varphi$$

We can anticipate a debit:

$$\text{e.g.}\quad \frac{\mathit{isThunk}\ t_1\ m_1\ (\lambda t_2.\ \mathit{isThunk}\ t_2\ m_2\ \varphi)}{\mathit{isThunk}\ t_1\ (m_1 + m)\ (\lambda t_2.\ \mathit{isThunk}\ t_2\ (m_2 - m)\ \varphi)}$$

Formal proof?

Danielsson (2008) gives a dependent type system (in Agda) for specifying and verifying amortized costs of programs with thunks

- ad-hoc type system, not a general-purpose program logic
- explicit credit-consuming operations must be inserted in code

Mével et al. (2019) extend Iris with time credits ⇒ Iris$^{\$}$

Our contribution: thunks, streams and the banker's queue (WIP) in Iris$^{\$}$

This talk: thunks

Iris extended with an assertion $\$n$ ($n \in \mathbb{N}$) satisfying a few laws:

$$\begin{aligned} &\vdash \$0 \\ \$(m+n) &\equiv \$m \star \$n \end{aligned}$$

We can throw credits away, but not forge nor duplicate them

Each execution step **consumes** $\$1$:

$$\{\$1 \star \ell \mapsto v\}\ !\ell\ \{\lambda v'.\ v' = v\}$$

Realized as ghost state: $\$n \triangleq \boxed{\circ\, n}^{\gamma_{\mathrm{TC}}}$ in the monoid $\textsc{Auth}(\mathbb{N}, +)$

$\Longrightarrow \boxed{\bullet\, N}^{\gamma_{\mathrm{TC}}}$ gives the total number of time credits in existence (kept in an Iris invariant)

Theorem (Soundness)

If $\{\$n\}\ e\ \{True\}$ is derivable in Iris$^\$$, then program e is safe and terminates in at most n steps.

```
type 'α thunk = 'α thunk_contents ref
and 'α thunk_contents =
  | Unevaluated of (unit → 'α)
  | Evaluated of 'α

let create f =
  ref (Unevaluated f)

let force t =
  match ! t with
  | Unevaluated f →
      let v = f () in        – evaluate the thunk
      t := Evaluated v ;     – memoize the result
      v
  | Evaluated v →
      v
```

Note: no re-entrency detection (2 states only)

$\Longrightarrow$ a static proof obligation will be needed

We want a persistent assertion $\mathit{isThunk}\ t\ m\ \varphi$ such that:

OVERESTIMATE
$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

PAY
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m - p)\ \varphi}$$

ANTICIPATE+DEDUCE
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \varphi\ v \Rrightarrow \psi\ v \qquad \forall v.\ \mathrm{persistent}(\psi\ v)}{\Rrightarrow \mathit{isThunk}\ t\ (m + n)\ \psi}$$

$$\{(\$n \mathrel{-\!\!*} \mathit{wp}\ f()\ \{\varphi\}) \star \forall v.\ \mathrm{persistent}(\varphi\ v)\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v\}$$

We want a persistent assertion $\mathit{isThunk}\ t\ m\ \varphi$ such that:

$$\textsc{Overestimate}\quad \frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

$$\textsc{Pay}\quad \frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m-p)\ \varphi}$$

$$\textsc{Anticipate+Deduce}\quad \frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \varphi\ v \Rrightarrow \psi\ v \qquad \forall v.\ \mathrm{persistent}(\psi\ v)}{\Rrightarrow \mathit{isThunk}\ t\ (m+n)\ \psi}$$

$$\{(\$n \mathrel{-\!\!*} \mathit{wp}\ f\,()\ \{\varphi\}) \star \forall v.\ \mathrm{persistent}(\varphi\ v)\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v\}$$

We want a persistent assertion $\mathit{isThunk}\ t\ m\ \varphi$ such that:

OVERESTIMATE

$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

PAY

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m - p)\ \varphi}$$

ANTICIPATE+DEDUCE

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \Box\, \varphi\ v \Rrightarrow \Box\, \psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ (m + n)\ \psi}$$

$$\begin{array}{c} \{\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\, \varphi\}\} \\ \mathit{create}\ f \\ \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\} \end{array} \qquad \begin{array}{c} \{\mathit{isThunk}\ t\ 0\ \varphi\} \\ \mathit{force}\ t \\ \{\lambda v.\ \Box\, \varphi\ v\} \end{array}$$

We want a persistent assertion *isThunk t m* φ such that:

A thunk is evaluated only once:
these arrows need not be persistent

OVERESTIMATE

$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \le m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m - p)\ \varphi}$$

ANTICIPATE+DEDUCE

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \Box\,\varphi\ v \Rrightarrow \Box\,\psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ (m + n)\ \psi}$$

$$\{\$n \mathrel{-\!\!*} \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\quad \mathit{create}\ f\quad \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi\}\quad \mathit{force}\ t\quad \{\lambda v.\ \Box\,\varphi\ v\}$$

We want a persistent assertion *isThunk* $t\ m\ \varphi$ such that:

OVERESTIMATE
$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

PAY
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m-p)\ \varphi}$$

ANTICIPATE+DED Re-entrancy?
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \ast \Box\,\varphi\ v \Rrightarrow \Box\,\psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ (m+n)\ \psi}$$

$$\{\$n \mathrel{-\!\!*} \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \Box\,\varphi\ v\}$$

We want a persistent assertion *isThunk t m φ* such that:

OVERESTIMATE
$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \le m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

PAY
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m-p)\ \varphi}$$

ANTICIPATE+DEDUCE
$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \Box\,\varphi\,v \Rrightarrow \Box\,\psi\,v}{\Rrightarrow \mathit{isThunk}\ t\ (m+n)\ \psi}$$

CANFORCEEXCL
$$\frac{\mathit{canForce} \qquad \mathit{canForce}}{\mathit{False}}$$

$$\{\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi \star \mathit{canForce}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \Box\,\varphi\,v \star \mathit{canForce}\}$$

where *canForce* is owned at the beginning of the world

Desired logical API

We want a persistent assertion $\mathit{isThunk}\ t\ m\ \varphi$ such that:

OVERESTIMATE

$$\frac{\mathit{isThunk}\ t\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ m_2\ \varphi}$$

PAY

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ (m - p)\ \varphi}$$

ANTICIPATE+D

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \Box\,\varphi\ v \Rrightarrow \Box\,\psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ (m + n)\ \psi}$$

How to force a thunk from another thunk?

CL

$$\frac{\mathit{canForce} \qquad \mathit{canForce}}{\mathit{False}}$$

$$\{\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\quad \mathit{create}\ f \quad \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ 0\ \varphi\ \star\ \mathit{canForce}\}\quad \mathit{force}\ t \quad \{\lambda v.\ \Box\,\varphi\ v\ \star\ \mathit{canForce}\}$$

where *canForce* is owned at the beginning of the world

We want a persistent assertion $\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi$ such that:

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m_1\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ \mathcal{N}\ m_2\ \varphi}\ \text{OVERESTIMATE}$$

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m-p)\ \varphi}\ \text{PAY}$$

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \qquad \forall v.\ \$n \star \Box\,\varphi\ v \Rrightarrow \Box\,\psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m+n)\ \psi}\ \text{ANTICIPATE+DEDUCE}$$

$$\frac{\mathit{canForce}\ \mathcal{N}_1 \qquad \mathit{canForce}\ \mathcal{N}_2}{(\uparrow\mathcal{N}_1) \cap (\uparrow\mathcal{N}_2) = \varnothing}\ \text{CANFORCEEXCL}$$

$$\{\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ \mathcal{N}\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ \mathcal{N}\ 0\ \varphi \star \mathit{canForce}\ \mathcal{N}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \Box\,\varphi\ v \star \mathit{canForce}\ \mathcal{N}\}$$

where $\mathit{canForce}\ \top$ is owned at the beginning of the world

We want a persistent assertion $\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi$ such that:

OVERESTIMATE

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m_1\ \varphi \qquad m_1 \le m_2}{\mathit{isThunk}\ t\ \mathcal{N}\ m_2\ \varphi}$$

PAY

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m-p)\ \varphi}$$

ANTICIPATE+D

...But how to pass the token to the inner thunk?

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \qquad \forall v.\ \$n \star \Box\,\varphi\, v \Rrightarrow \Box\,\psi\, v}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m+n)\ \psi}$$

$$\frac{\mathit{canForce}\ \mathcal{N}_1 \qquad \mathit{canForce}\ \mathcal{N}_2}{(\uparrow\mathcal{N}_1) \cap (\uparrow\mathcal{N}_2) = \varnothing}$$

$$\{\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\,\varphi\}\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ \mathcal{N}\ n\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ \mathcal{N}\ 0\ \varphi \ \star\ \mathit{canForce}\ \mathcal{N}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \Box\,\varphi\, v \ \star\ \mathit{canForce}\ \mathcal{N}\}$$

where $\mathit{canForce}\ \top$ is owned at the beginning of the world

We want a persistent assertion $\mathit{isThunk}\ t\ \mathcal{N}\ m\ R\ \varphi$ such that:

OVERESTIMATE

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m_1\ R\ \varphi \qquad m_1 \leq m_2}{\mathit{isThunk}\ t\ \mathcal{N}\ m_2\ R\ \varphi}$$

PAY

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ R\ \varphi \qquad \$p}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m - p)\ R\ \varphi}$$

ANTICIPATE+DEDUCE

$$\frac{\mathit{isThunk}\ t\ \mathcal{N}\ m\ R\ \varphi \qquad \forall v.\ \$n \star \Box\, \varphi\ v \Rrightarrow \Box\, \psi\ v}{\Rrightarrow \mathit{isThunk}\ t\ \mathcal{N}\ (m + n)\ R\ \psi}$$

CANFORCEEXCL

$$\frac{\mathit{canForce}\ \mathcal{N}_1 \qquad \mathit{canForce}\ \mathcal{N}}{(\uparrow\mathcal{N}_1) \cap (\uparrow\mathcal{N}_2) = \varnothing}$$

$$\{\$n \star R \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\, \varphi \star R\}\}$$
$$\mathit{create}\ f$$
$$\{\lambda t.\ \mathit{isThunk}\ t\ \mathcal{N}\ n\ R\ \varphi\}$$

$$\{\mathit{isThunk}\ t\ \mathcal{N}\ 0\ R\ \varphi\ \star\ \mathit{canForce}\ \mathcal{N}\ \star\ R\}$$
$$\mathit{force}\ t$$
$$\{\lambda v.\ \Box\, \varphi\ v\ \star\ \mathit{canForce}\ \mathcal{N}\ \star\ R\}$$

where $\mathit{canForce}\ \top$ is owned at the beginning of the world

ANTICIPATE+DEDUCE

$$\frac{\mathit{isThunk}\ t\ m\ \varphi \qquad \forall v.\ \$n \star \Box\, \varphi\, v \Rrightarrow \Box\, \psi\, v}{\Rrightarrow \mathit{isThunk}\ t\ (m+n)\ \psi}$$

The following is nonsensical, the thunk's post-cond. must be persistent:

ANTICIPATE

$$\frac{\mathit{isThunk}\ t\ m\ \varphi}{\Rrightarrow \mathit{isThunk}\ t\ (m+n)\ (\$n \star \varphi)}$$

$\Longrightarrow$ We bake deduction with anticipation

$\Longrightarrow$ $n = 0$ gives a deduction rule which allows ghost updates

Rules PAY and ANTICIPATE+DEDUCE allow to derive e.g.:

$$\frac{\mathit{isThunk}\ t_1\ m_1\ (\lambda t_2.\ \mathit{isThunk}\ t_2\ m_2\ \varphi)}{\Rrightarrow \mathit{isThunk}\ t_1\ (m_1+m)\ (\lambda t_2.\ \mathit{isThunk}\ t_2\ (m_2-m)\ \varphi)}$$

Tentative invariant

(assuming a ghost name γ_t for each location t, by convenience)

$$\mathit{thunkInv}\ t\ n\ \varphi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_t} \star \bigvee \begin{cases} \exists f.\ t \mapsto U\ f\ \star\ \$p\ \star\ (\$n \twoheadrightarrow wp\ f()\ \{\Box\, \varphi\}) \\ \exists v.\ t \mapsto E\ v\ \star\ \Box\, \varphi\ v \end{cases}$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\, (n - m)}^{\gamma_t}\ \star\ \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$ (slightly wrong, see next slide)

Ghost state in $\text{AUTH}(\mathbb{N}, \max)$ records the number of accumulated credits:

Tentative invariant

(assuming a ghost name γ_t for each location t, by convenience)

$$\mathit{thunkInv}\ t\ n\ \varphi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_t} \star \bigvee \begin{cases} \exists f.\ t \mapsto U\, f \ \star\ \$p \ \star\ (\$n \mathrel{-\!\!*} \mathit{wp}\ f()\ \{\Box\,\varphi\}) \\ \exists v.\ t \mapsto E\ v\ \star\ \Box\,\varphi\ v \end{cases}$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\,(n-m)}^{\gamma_t} \ \star\ \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$

(slightly wrong, see next slide)

Ghost state in $\textsc{Auth}(\mathbb{N}, \max)$ records the number of accumulated credits:

- $\boxed{\bullet\, p}^{\gamma_t}$ asserts that exactly p credits have been **p**aid already

(assuming a ghost name γ_t for each location t, by convenience)

$$\mathit{thunkInv}\ t\ n\ \varphi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_t} \star \bigvee \begin{cases} \exists f.\ t \mapsto U\, f \ \star\ \$p \ \star\ (\$n \mathrel{-\!\!*} \mathit{wp}\, f()\, \{\Box\, \varphi\}) \\ \exists v.\ t \mapsto E\ v \ \star\ \Box\, \varphi\, v \end{cases}$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\, (n - m)}^{\gamma_t} \ \star\ \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$

(slightly wrong, see next slide)

Ghost state in $\text{Auth}(\mathbb{N}, \max)$ records the number of accumulated credits:

- $\boxed{\bullet\, p}^{\gamma_t}$ asserts that exactly p credits have been **p**aid already
- $\boxed{\circ\, (n - m)}^{\gamma_t}$ witnesses that **at least** $n - m$ credits have been paid
  - n credits are **n**eeded in total
  - m credits are apparently **m**issing (our debit)

(assuming a ghost name γ_t for each location t, by convenience)

$$\mathit{thunkInv}\ t\ n\ \varphi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_t} \star \bigvee \begin{cases} \exists f.\ t \mapsto U\ f\ \star\ \$p\ \star\ (\$n \twoheadrightarrow wp\ f()\ \{\Box\, \varphi\}) \\ \exists v.\ t \mapsto E\ v\ \star\ \Box\, \varphi\ v \end{cases}$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\, (n - m)}^{\gamma_t}\ \star\ \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$ (slightly wrong, see next slide)

Ghost state in AUTH($\mathbb{N}$, max) records the number of accumulated credits:

- $\boxed{\bullet\, p}^{\gamma_t}$ asserts that exactly p credits have been **p**aid already
- $\boxed{\circ\, (n-m)}^{\gamma_t}$ witnesses that **at least** $n-m$ credits have been paid
  - n credits are **n**eeded in total
  - m credits are apparently **m**issing (our debit)

OVERESTIMATE: $\boxed{\circ\, (n - m_1)}^{\gamma} \twoheadrightarrow \boxed{\circ\, (n - m_2)}^{\gamma}$ if $m_1 \leq m_2$

PAY: $\boxed{\bullet\, p}^{\gamma} \Rrightarrow \boxed{\bullet\, (p + p')}^{\gamma} \star \boxed{\circ\, (p + p')}^{\gamma}$

Problem: an Iris invariant can only stay opened around one atomic step

$$\mathit{isThunk}\ t \quad m\ \varphi \triangleq \exists n.\ \boxed{\circ\,(n - m)}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$

Re-entrency and non-atomic invariants

Problem: an Iris invariant can only stay opened around one atomic step

$$\mathit{isThunk}\ t \quad m\ \varphi \triangleq \exists n.\ \boxed{\circ\,(n-m)}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ n\ \varphi}$$

Solution: use a "**non-atomic invariant**" (Iris' convenience library)

Problem: an Iris invariant can only stay opened around one atomic step

$$\begin{aligned} \mathit{isThunk}\ t\quad m\ \varphi &\triangleq \exists n.\ \boxed{\circ\,(n-m)}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ n\ \varphi} \\ \mathit{canForce} &\triangleq \mathit{naInvTok} \end{aligned}$$

Solution: use a "**non-atomic invariant**" (Iris' convenience library)

A NA invariant is guarded by an **exclusive token**

Re-entrency and non-atomic invariants

Problem: an Iris invariant can only stay opened around one atomic step

$$\mathcal{N} \;:\; \text{namespace}$$

$$\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\,(n - m)}^{\gamma_t} \ast \boxed{\mathit{thunkInv}\ t\ n\ \varphi}^{\mathcal{N}}$$

$$\mathit{canForce}\ \mathcal{N} \triangleq \mathit{naInvTok}\ (\uparrow\mathcal{N})$$

Solution: use a "**non-atomic invariant**" (Iris' convenience library)

A NA invariant is guarded by an **exclusive token**

$$\vdash \quad \mathit{naInvTok}\ \varnothing$$

$$\mathit{naInvTok}\ (\mathcal{E}_1 \uplus \mathcal{E}_2) \quad \equiv \quad \mathit{naInvTok}\ \mathcal{E}_1 \ast \mathit{naInvTok}\ \mathcal{E}_2$$

Problem: an Iris invariant can only stay opened around one atomic step

$$\mathcal{N} \ : \ \text{namespace}$$

$$\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi \triangleq \exists n.\ \boxed{\circ\,(n-m)}^{\gamma_t} \ \star\ \boxed{\boxed{\mathit{thunkInv}\ t\ n\ \varphi}}^{\mathcal{N}}$$

$$\mathit{canForce}\ \mathcal{N} \triangleq \mathit{naInvTok}\ (\uparrow\mathcal{N})$$

Solution: use a "**non-atomic invariant**" (Iris' convenience library)

A NA invariant is guarded by an **exclusive token**

$$\vdash \quad \mathit{naInvTok}\ \emptyset$$

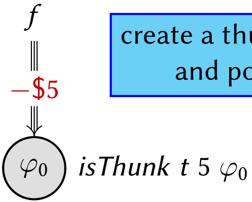
$$\mathit{naInvTok}\ (\mathcal{E}_1 \uplus \mathcal{E}_2) \quad \equiv \quad \mathit{naInvTok}\ \mathcal{E}_1\ \star\ \mathit{naInvTok}\ \mathcal{E}_2$$

Thus, it can stay opened for as long as we want:

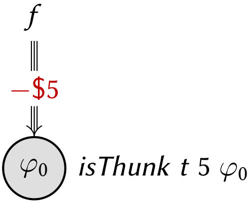
NA-INV-ACC

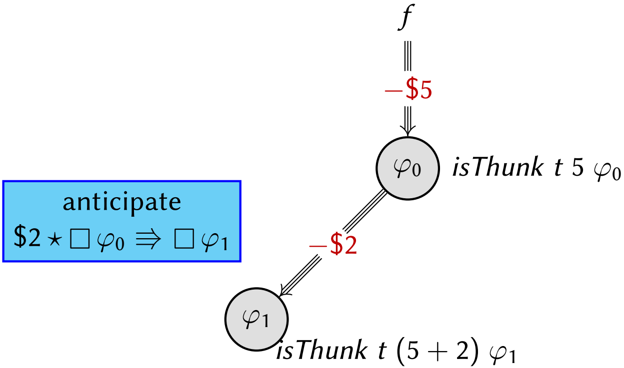
$$\frac{\boxed{\boxed{P}}^{\mathcal{N}} \qquad \mathit{naInvTok}\ (\uparrow\mathcal{N}) \qquad \uparrow\mathcal{N} \subseteq \mathcal{E}}{{}^{\mathcal{E}}{\Rrightarrow}^{\mathcal{E}}\ \triangleright P\ \star\ \left(\triangleright P\ {}^{\mathcal{E}}{\Rrightarrow}^{\mathcal{E}}\ \mathit{naInvTok}\ (\uparrow\mathcal{N})\right)}$$

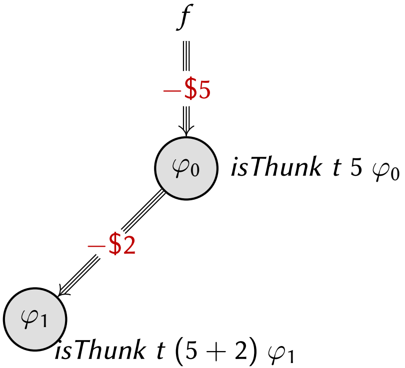
- persistence ✔
- OVERESTIMATE ✔
- PAY ✔
- ANTICIPATE+DEDUCE ✘
- CANFORCEEXCL ✔
- spec of *create* ✔
- spec of *force* ✔

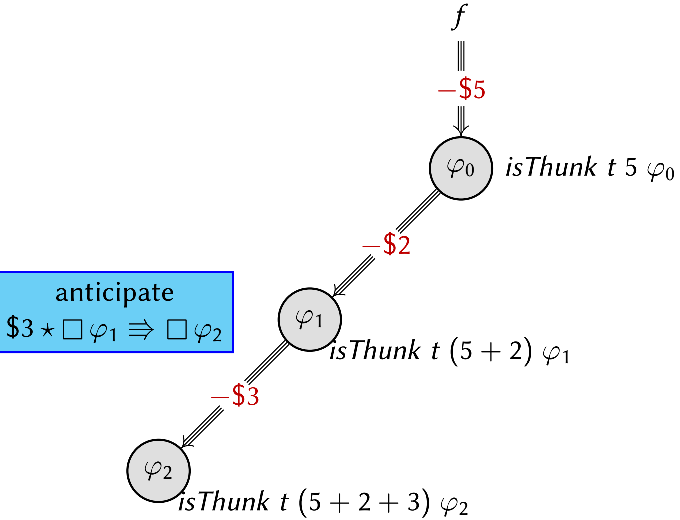


create a thunk with debit 5 and post-cond. φ_0

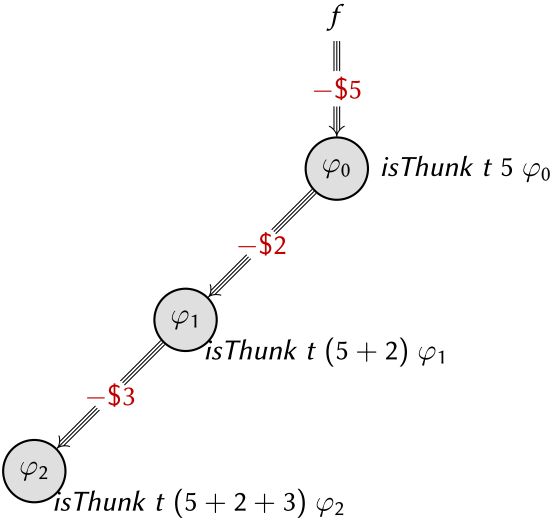

f
−$5
φ0
isThunk t 5 φ0


f
−$5
φ0
isThunk t 5 φ0
anticipate
$2 ⋆ □ φ0 ⇛ □ φ1
−$2
φ1
isThunk t (5 + 2) φ1

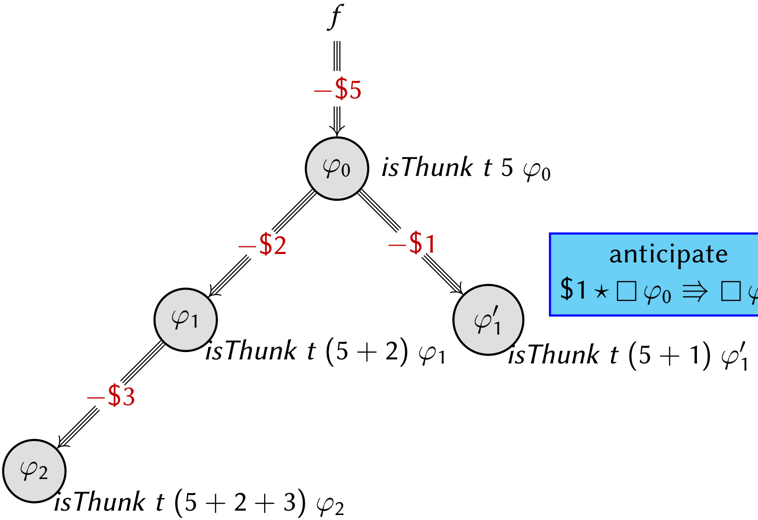

f
−$5
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1



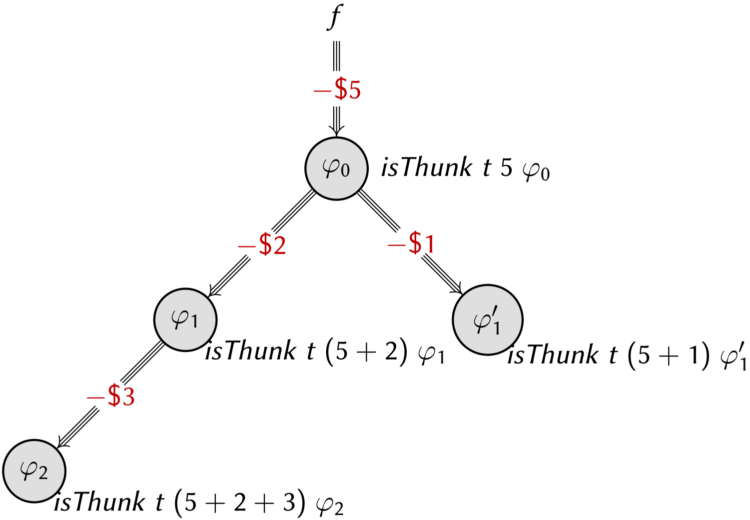
f
$-\$5$
φ_0
$isThunk\ t\ 5\ \varphi_0$
$-\$2$
anticipate
$\$3 \star \Box\,\varphi_1 \Rrightarrow \Box\,\varphi_2$
φ_1
$isThunk\ t\ (5+2)\ \varphi_1$
$-\$3$
φ_2
$isThunk\ t\ (5+2+3)\ \varphi_2$

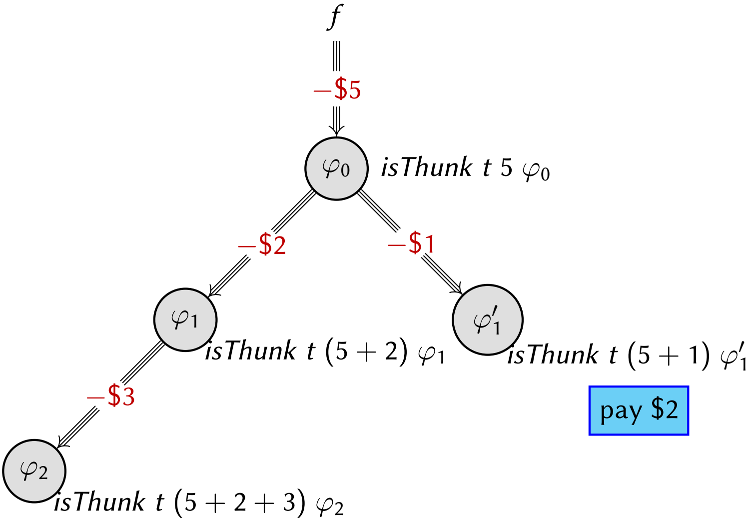


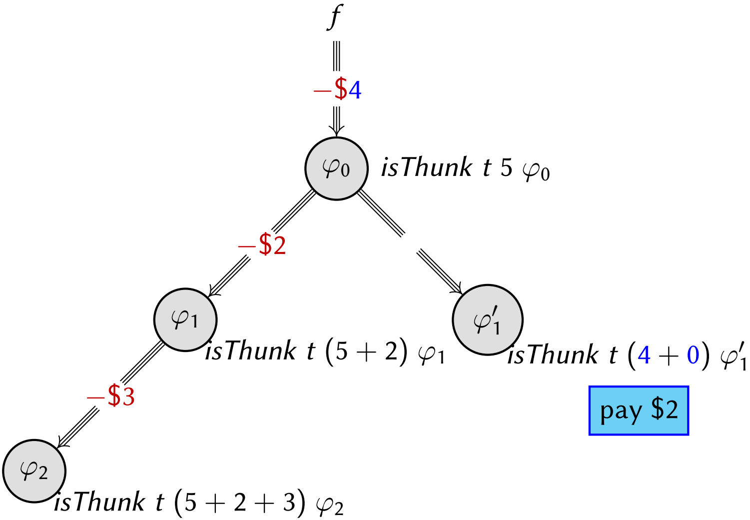
f
$-\$5$
φ_0
$isThunk\ t\ 5\ \varphi_0$
$-\$2$
φ_1
$isThunk\ t\ (5+2)\ \varphi_1$
$-\$3$
φ_2
$isThunk\ t\ (5+2+3)\ \varphi_2$

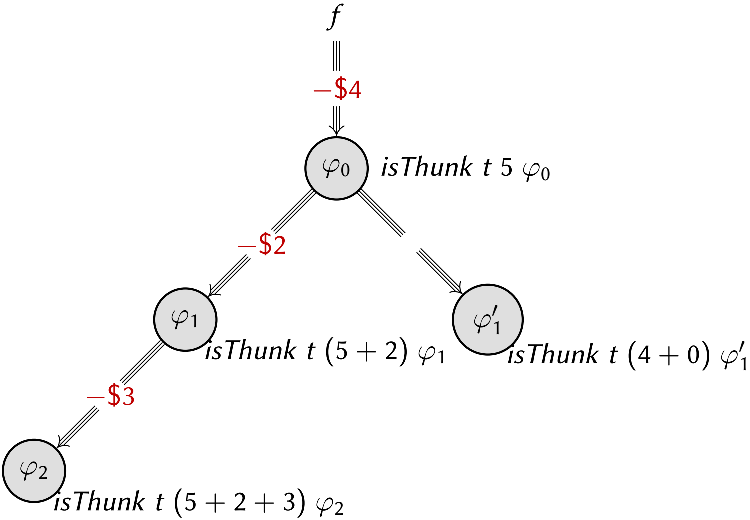


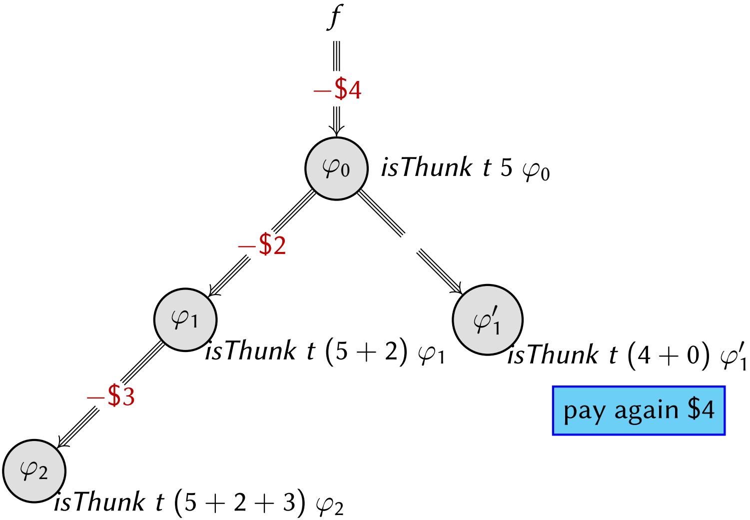
anticipate
$\$1 \star \Box\,\varphi_0 \Rrightarrow \Box\,\varphi'_1$

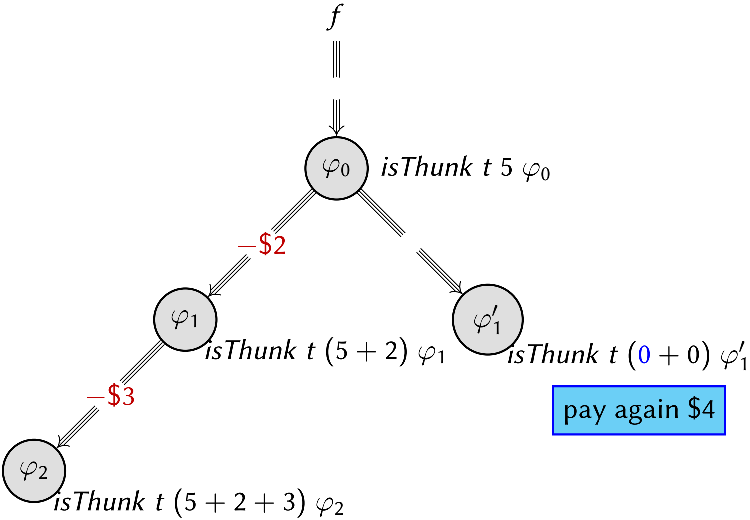

f
−$5
φ0
isThunk t 5 φ0
−$2
−$1
φ1
φ′1
isThunk t (5 + 2) φ1
isThunk t (5 + 1) φ′1
−$3
φ2
isThunk t (5 + 2 + 3) φ2

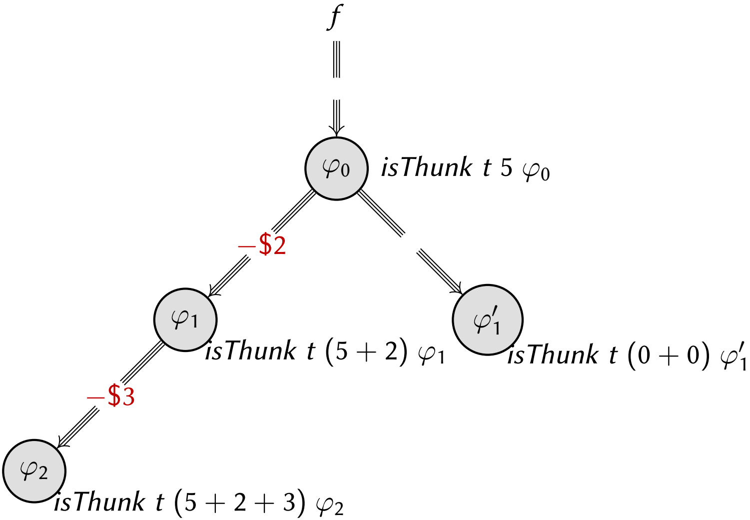

f
$-\$5$
φ_0
$isThunk\ t\ 5\ \varphi_0$
$-\$2$
$-\$1$
φ_1
φ_1'
$isThunk\ t\ (5+2)\ \varphi_1$
$isThunk\ t\ (5+1)\ \varphi_1'$
$-\$3$
pay \$2
φ_2
$isThunk\ t\ (5+2+3)\ \varphi_2$

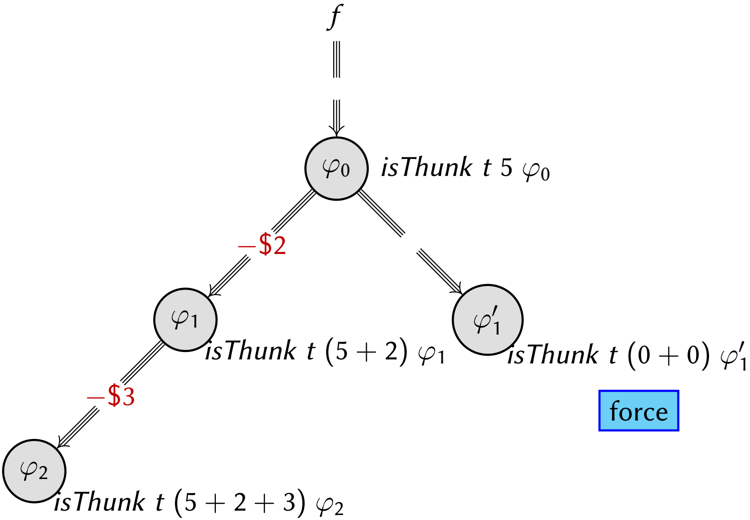

f
−$4
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1
φ'1
isThunk t (4 + 0) φ'1
pay $2
−$3
φ2
isThunk t (5 + 2 + 3) φ2

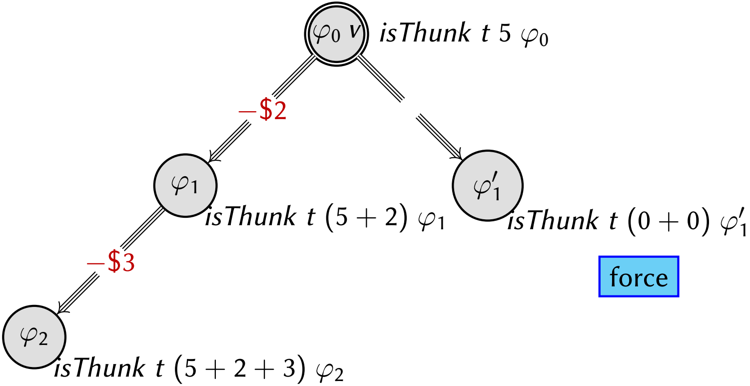

f
−$4
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1
φ′1
isThunk t (4 + 0) φ′1
−$3
φ2
isThunk t (5 + 2 + 3) φ2


f
−$4
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1
φ'1
isThunk t (4 + 0) φ'1
pay again $4
−$3
φ2
isThunk t (5 + 2 + 3) φ2

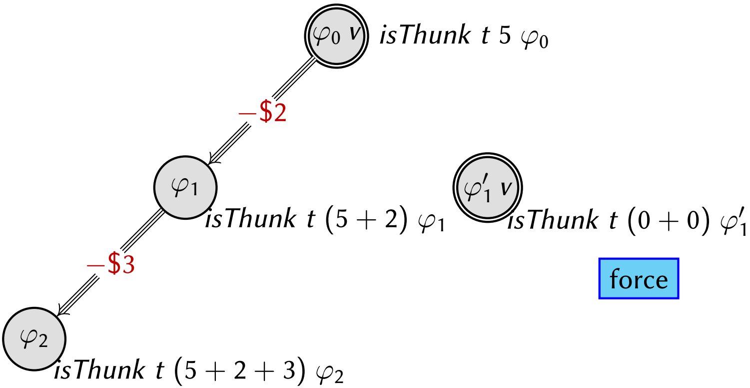

f
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1
φ′1
isThunk t (0 + 0) φ′1
pay again $4
−$3
φ2
isThunk t (5 + 2 + 3) φ2


f
φ_0
$\mathit{isThunk}\ t\ 5\ \varphi_0$
$-\$2$
φ_1
$\mathit{isThunk}\ t\ (5+2)\ \varphi_1$
φ'_1
$\mathit{isThunk}\ t\ (0+0)\ \varphi'_1$
$-\$3$
φ_2
$\mathit{isThunk}\ t\ (5+2+3)\ \varphi_2$

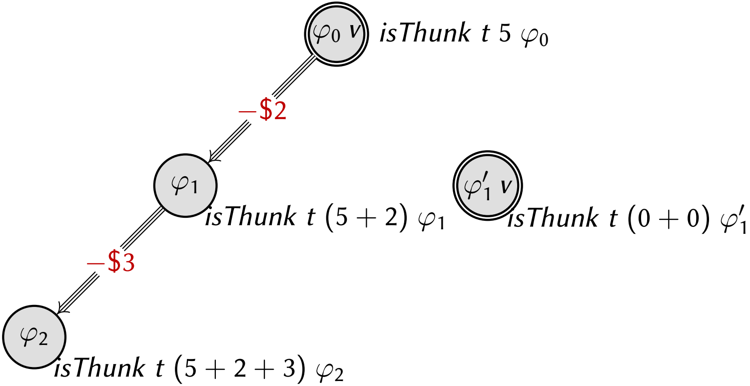

f
φ0
isThunk t 5 φ0
−$2
φ1
isThunk t (5 + 2) φ1
φ′1
isThunk t (0 + 0) φ′1
force
−$3
φ2
isThunk t (5 + 2 + 3) φ2

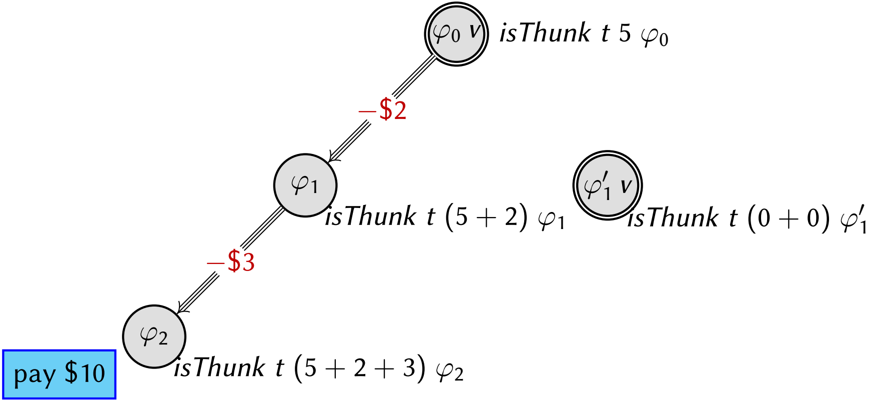


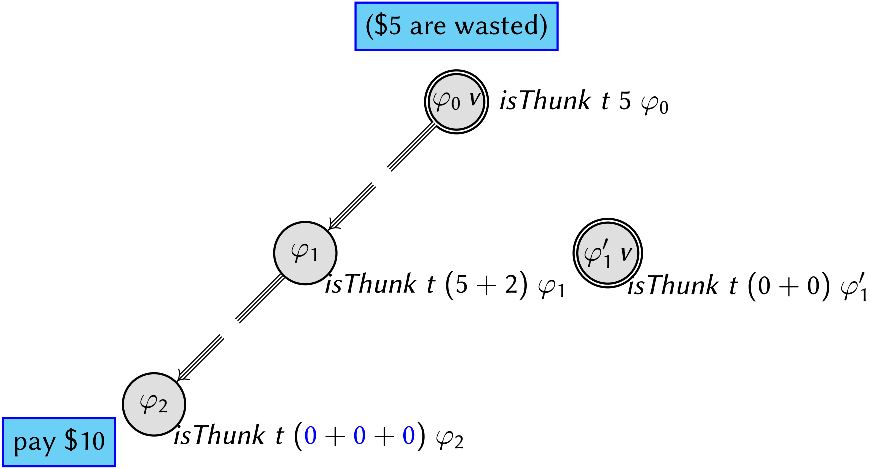
φ_0 v
isThunk t 5 φ_0
$-\$2$
φ_1
isThunk t (5 + 2) φ_1
φ'_1
isThunk t (0 + 0) φ'_1
$-\$3$
φ_2
isThunk t (5 + 2 + 3) φ_2
force

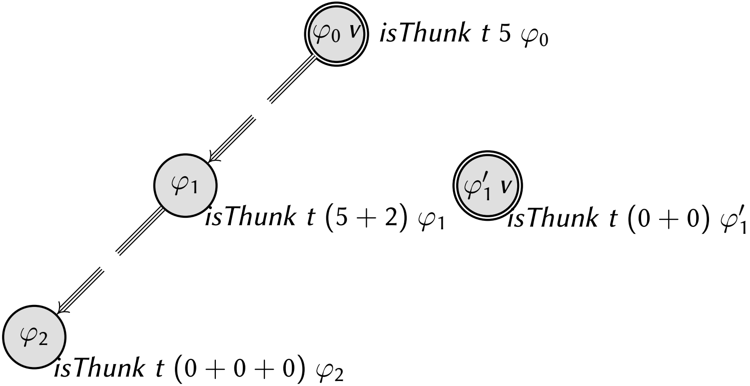


φ_0 v
isThunk t 5 φ_0
$-\$2$
φ_1
isThunk t $(5+2)$ φ_1
φ'_1 v
isThunk t $(0+0)$ φ'_1
force
$-\$3$
φ_2
isThunk t $(5+2+3)$ φ_2

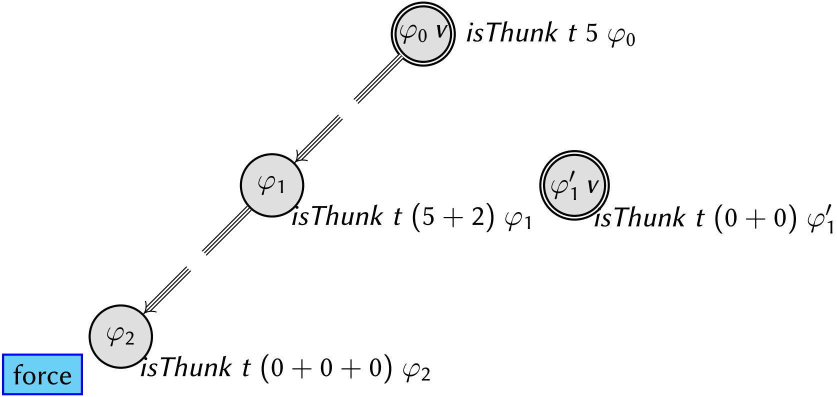

φ_0 v
isThunk t 5 φ_0
$-\$2$
φ_1
isThunk t (5 + 2) φ_1
φ'_1 v
isThunk t (0 + 0) φ'_1
$-\$3$
φ_2
isThunk t (5 + 2 + 3) φ_2


φ_0 v
isThunk t 5 φ_0
$-\$2$
φ_1
isThunk t (5 + 2) φ_1
φ'_1 v
isThunk t (0 + 0) φ'_1
$-\$3$
φ_2
isThunk t (5 + 2 + 3) φ_2
pay \$10

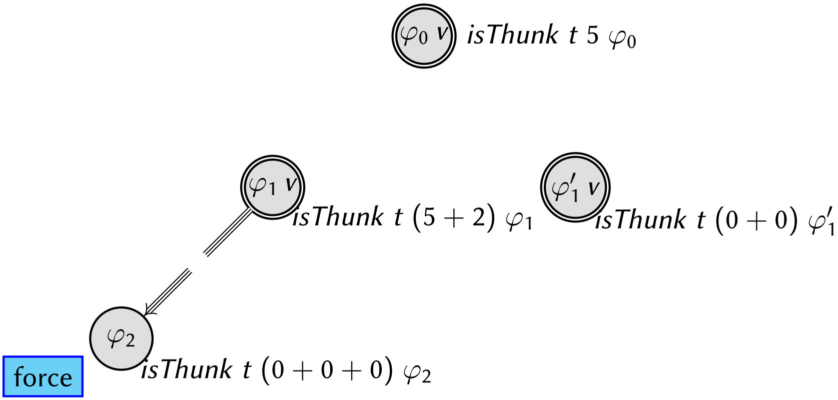

($5 are wasted)
φ0 v
isThunk t 5 φ0
φ1
isThunk t (5 + 2) φ1
φ'1 v
isThunk t (0 + 0) φ'1
φ2
isThunk t (0 + 0 + 0) φ2
pay $10



φ0 v
isThunk t 5 φ0
φ1
isThunk t (5 + 2) φ1
φ1′ v
isThunk t (0 + 0) φ1′
φ2
isThunk t (0 + 0 + 0) φ2



φ_0 v
isThunk t 5 φ_0
φ_1
isThunk t (5 + 2) φ_1
φ'_1 v
isThunk t (0 + 0) φ'_1
φ_2
isThunk t (0 + 0 + 0) φ_2
force



φ_0 v
isThunk t 5 φ_0
φ_1 v
isThunk t $(5+2)$ φ_1
φ'_1 v
isThunk t $(0+0)$ φ'_1
φ_2
isThunk t $(0+0+0)$ φ_2
force

(φ_0 v) $\mathit{isThunk}\ t\ 5\ \varphi_0$

(φ_1 v) $\mathit{isThunk}\ t\ (5+2)\ \varphi_1$ (φ_1' v) $\mathit{isThunk}\ t\ (0+0)\ \varphi_1'$

force (φ_2 v) $\mathit{isThunk}\ t\ (0+0+0)\ \varphi_2$

Tower of invariants

Idea: stack a new invariant each time ANTICIPATE+DEDUCE is used

$\Longrightarrow$ Tower of invariants

(assuming a ghost name $\gamma_{t,d}$ for each location t, and integer d, by convenience)

$$\mathit{thunkInv}\ t\ n\ \varphi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_{t,0}} \star \bigvee\begin{cases}\exists f.\ t \mapsto U\ f\ \star\ \$p\ \star\ (\$n \mathrel{-\!\!*} wp\ f()\ \{\square\varphi \\ \exists v.\ t \mapsto E\ v\ \star\ \square\,\varphi\ v\end{cases}$$

$$\mathit{csqInv}\ t\ d\ n\ \varphi\ \psi \triangleq \exists p.\ \boxed{\bullet\, p}^{\gamma_{t,d}} \star \bigvee\begin{cases}\$p\ \star\ (\forall v.\ \$n \star \square\,\varphi\ v \Rrightarrow \square\,\psi\ v) \\ \square\,\psi\ v\end{cases}$$

$$\mathit{isThunk}\ t\ 0\ m\ \varphi \triangleq \exists p, n.\ 0 \le m{-}p \le n\ \star\ \boxed{\circ\, p}^{\gamma_{t,0}}\ \star\ \boxed{\boxed{\mathit{thunkInv}\ t\ n\ \varphi}}$$

$$\mathit{isThunk}\ t\ d\ m\ \varphi \triangleq \exists p, n, \psi.\ 0 \le m{-}p \le n\ \star\ \boxed{\circ\, p}^{\gamma_{t,d}}\ \star\ \boxed{\boxed{\mathit{csqInv}\ t\ n\ \psi\ \varphi}}$$
$$\star\ \mathit{isThunk}\ t\ (d{-}1)\ (m{-}n{+}p)\ \psi$$

Each level ($d \in \mathbb{N}$) has its own vault ($\gamma_{t,d}$) for filling a debit

```
type 'α stream = 'α cell thunk                       – a stream is computed on-demand
and 'α cell = Nil | Cons of 'α × 'α stream

let pop (xs : 'α stream) =
  match Thunk.force xs with
  | Cons (x, xs′) → Some (x, xs′)
  | Nil → None

let rev_of_list (xs : 'α list) : 'α stream =
  let rec rev_app (xs : 'α list) (ys : 'α cell) =    – rev_app reverses the list eagerly
    match xs with                                    – ↓ these new thunks have cost 0
    | x :: xs′ → rev_app xs′ (Cons (x, Thunk.create@@fun()→ ys))
    | [] → ys in
  Thunk.create@@fun()→ rev_app xs Nil                – this leading thunk is costly

let rec append (xs : 'α stream) (ys : 'α stream) =
  Thunk.create@@fun()→                               – this thunk has a constant overhead
    match Thunk.force xs with
    | Cons (x, xs′) → Cons (x, append xs′ ys)
    | Nil → Thunk.force ys
```

A stream is a thunk which computes a value and another thunk (its tail)

A stream has a list of debits, those required for computing the successive elements of the stream:

$$
\begin{aligned}
&\mathit{isStream}\ s\ [m_0, \ldots, m_n]\ [v_1, \ldots, v_n] \triangleq \\
&\quad \mathit{isThunk}\ s\ m_0\ (\lambda c_1.\ \exists s_1.\ c_1 = \mathit{Cons}(v_1, s_1) \star \\
&\qquad \mathit{isThunk}\ s_1\ m_1\ (\lambda c_2.\ \exists s_2.\ c_2 = \mathit{Cons}(v_2, s_2) \star \\
&\qquad\quad \ldots \\
&\qquad\qquad \mathit{isThunk}\ s_n\ m_n\ (\lambda c_{n+1}.\ c_{n+1} = \mathit{Nil})\ldots))
\end{aligned}
$$

Logical API of streams

PAYSTREAM

$$\frac{\mathit{isStream}\ s\ [m_0, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \$p}{\mathit{isStream}\ s\ [m_0, \ldots, m_i - p, \ldots, m_n]\ [v_1, \ldots, v_n]}$$

ANTICIPATE+OVERESTIMATESTREAM

$$\frac{\mathit{isStream}\ s\ [m_0, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow \mathit{isStream}\ s\ [m'_0, \ldots, m'_n]\ [v_1, \ldots, v_n]}$$

$$\{\mathit{isStream}\ s\ [m_0, \ldots, m_n]\ [v_1, \ldots, v_n] \star \mathit{isStream}\ s'\ [m'_0, \ldots, m'_{n'}]\ [v'_1, \ldots, v'_{n'}]\}$$
$$\mathit{append}\ s\ s'$$
$$\{\lambda t.\ \mathit{isStream}\ t\ [A + m_0, \ldots, A + m_n + m'_0, m'_1, \ldots, m'_{n'}]\ [v_1, \ldots, v_n, v'_1, \ldots, v'_{n'}]\}$$

$$\{\mathit{isList}\ \ell\ [v_1, \ldots, v_n]\}$$
$$\mathit{rev\_of\_list}\ \ell$$
$$\{\lambda s.\ \mathit{isStream}\ s\ [B \cdot n, 0, \ldots, 0]\ [v_n, \ldots, v_1]\}$$

We address nested thunks with **generations** $g \in \mathbb{N}$:

$$
\begin{aligned}
&\mathit{isStream}\ s\ [m_0, \ldots, m_n]\ [v_1, \ldots, v_n] \triangleq \\
&\quad \exists g_0.\ \mathit{isThunk}\ s\ \mathcal{N}(g_0)\ m_0\ (\mathit{naInvTok}\ \mathcal{E}(g_0))\ (\lambda c_1.\ \exists s_1.\ c_1 = \mathit{Cons}(v_1, s_1) \star \\
&\qquad \exists g_1 \leq g_0.\ \mathit{isThunk}\ s_1\ \mathcal{N}(g_1)\ m_1\ (\mathit{naInvTok}\ \mathcal{E}(g_1))\ (\lambda c_2.\ \exists s_2.\ c_2 = \mathit{Cons}(v_2 \\
&\qquad\quad \cdots \\
&\qquad\qquad \exists g_n \leq g_{n-1}.\ \mathit{isThunk}\ s_n\ \mathcal{N}(g_n)\ m_n\ (\mathit{naInvTok}\ \mathcal{E}(g_n))\ (\lambda c_{n+1}.\ c_{n+1} =
\end{aligned}
$$

where:

$$
\begin{aligned}
\mathcal{E}(g) &\triangleq \top \setminus \uparrow\mathcal{N}(g) \\
\mathcal{E}(g) &\subseteq \mathcal{E}(g+1) \\
\uparrow\mathcal{N}(g+1) &\subseteq \uparrow\mathcal{N}(g)
\end{aligned}
$$

Highlights of the proof of the banker's queue:

- anticipation of debit
  not obvious to state
  even less obvious to ensure
- generations for nested thunks

https://gitlab.inria.fr/gmevel/iris-time-proofs

DANIELSSON, N. A. 2008. Lightweight semiformal time complexity analysis for purely functional data structures. In *Principles of Programming Languages (POPL)*.

MÉVEL, G., JOURDAN, J.-H., ET POTTIER, F. 2019. Time credits and time receipts in Iris. In *European Symposium on Programming (ESOP)*. Lecture Notes in Computer Science, vol. 11423. Springer, 1–27.

OKASAKI, C. 1999. *Purely Functional Data Structures*. Cambridge University Press.

TARJAN, R. E. 1985. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods 6*, 2, 306–318.