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*:

```
type 'α queue = 'α list × 'α list

let push (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
```

The “banker’s method” (Tarjan, 1985) gives constant **amortized** costs:

- *push* costs $\mathcal{O}(1)$:
  - we spend $\mathcal{O}(1)$ for cons-ing this element
  - we **save** $\mathcal{O}(1)$, covering for this element’s future reversal
- *pop* costs $\mathcal{O}(1)$:
  - we spend $\mathcal{O}(1)$ for the call to *pop* itself
  - reversal is pre-paid by past *push*es

Issue: we can't spend time savings twice

let $q = push\,(push\,(push\; nil\; 1)\; 2)\; 3$ **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**!

The front sequence is a stream, i.e., a list computed on-demand:

```
type 'α stream = 'α cell thunk
and 'α cell = Nil | Cons of 'α × 'α stream

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 push (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
```

Amortized complexity of the banker's queue

Reversing $|r|$ elements is costly, but is done after $|f| \geq |r|$ calls to *pop*

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

$\Longrightarrow$ Constant amortized costs:

- *rebalance* costs $\mathcal{O}(1)$
- *push* costs $\mathcal{O}(1)$
- *pop* costs $\mathcal{O}(1)$

Persistence: credit vs. debit

Key idea: time is a resource, $\$n$ ("n time credits") allow taking n steps

- The non-lazy queue saves **credit** for a yet unknown computation
  $\Longrightarrow$ Not duplicable (cannot forge money)
- The banker's queue repays a **debit** for an already known computation
  $\Longrightarrow$ Duplicable (can waste money)
  $\Longrightarrow$ The banker's queue can be used **persistently**
  - Remark: the value is computed only once the debit is repaid

Building blocks:

- A **thunk** is a suspended computation, it holds a debit:

$$\mathit{isThunk}\ t\ m\ \varphi \qquad (m \in \mathbb{N})$$

  Ownership of a thunk is duplicable:

$$\mathit{isThunk}\ t\ m\ \varphi \quad \twoheadrightarrow \quad \mathit{isThunk}\ t\ m\ \varphi \star \mathit{isThunk}\ t\ m\ \varphi$$

- A **stream** is a chain of nested thunks, it holds a list of debits:

$$\begin{aligned}
&\mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \triangleq \\
&\quad \mathit{isThunk}\ s\ m_1\ (\lambda c_1.\ \exists s_2.\ c_1 = \mathit{Cons}(v_1, s_2) \star \\
&\qquad \mathit{isThunk}\ s_2\ m_2\ (\lambda c_2.\ \exists s_3.\ c_2 = \mathit{Cons}(v_2, s_3) \star \\
&\qquad\quad \ddots \\
&\qquad\qquad \mathit{isThunk}\ s_{n+1}\ 0\ (\lambda c_{n+1}.\ c_{n+1} = \mathit{Nil})\ldots))
\end{aligned}$$

  Ownership of a stream is duplicable

We can anticipate an inner thunk's debit:

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

$\Longrightarrow$ We can anticipate debits in a stream:

$$\text{e.g.}\quad \frac{\mathit{isStream}\ s\ [\overbrace{A, \ldots, A}^{n \text{ times}}, (n{+}1)B, \overbrace{0, \ldots, 0}^{n \text{ times}}]\ [f_1, \ldots, f_n, r_{n+1}, \ldots, r_1]}{\mathit{isStream}\ s\ [A{+}B, \ldots, A{+}B, B, 0, \ldots, 0]\ [f_1, \ldots, f_n, r_{n+1}, \ldots, r_1]}$$

This is needed in the proof of the banker's queue

Formal proof?

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

- semi-formal guarantee
- no ghost operations: must insert them in code, manually

  must conform to a strict discipline, must balance branches' costs, payment creates a thunk, in-depth payment needs special care…
- ad-hoc type system, not a general-purpose program logic

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

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

This talk: thunks, streams

1. Introduction
2. Iris$^\$$ in a nutshell
3. Specification and proof, without anticipation
4. Anticipation

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

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

We can throw credits away, but not forge or duplicate them

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

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

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)

Soundness of Iris$^{\$}$

Theorem (Soundness)

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

Implementation of thunks

```
type 'α thunk = 'α thunk_contents ref
and 'α thunk_contents =
  | Future of (unit → 'α)
  | Busy
  | Done of 'α

let create f =
  ref (Future f)

let force t =
  match ! t with
  | Future f →
      if not (compare_and_set t (Future f) Busy)    – forbid concurrent forcing
        then exit () ;
      let v = f () in                               – evaluate the thunk...
      t := Done v ;                                 – ...and memoize the result
      v
  | Busy → exit ()                                  – forbid reentrancy
  | Done v → v
```

Specification of thunks

$$\{\$K_{cr} \star (\$n \mathrel{-\!\!*} wp\ f()\ \{\square\, \varphi\})\}\quad create\ f \quad \{\lambda t.\ isThunk\ t\ n\ \varphi\}$$

$$\{\$K_{frc} \star isThunk\ t\ 0\ \varphi\}\quad force\ t \quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

$$\text{persistent}(isThunk\ t\ m\ \varphi)$$

OVERESTIMATE

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

PAY

$$\frac{isThunk\ t\ m\ \varphi \qquad \$p}{\mathrel{|\!\!\Rrightarrow} isThunk\ t\ (m - p)\ \varphi}$$

ANTICIPATE

$$\frac{isThunk\ t\ m\ \varphi \qquad \forall v.\ \$n \star \varphi\ v \Rrightarrow \square\, \psi\ v}{\mathrel{|\!\!\Rrightarrow} isThunk\ t\ (m + n)\ \psi}$$

Specification of thunks

$$\{\$K_{\mathrm{cr}} \star (\$n \twoheadrightarrow\!\!* \; wp\; f() \;\{\Box\, \varphi\})\}$$
$$create\; f$$
$$\{\lambda t.\; isThunk\; t\; n\; \varphi\}$$

$$\{\$K_{\mathrm{frc}} \star isThunk\; t\; 0\; \varphi\}$$
$$force\; t$$
$$\{\lambda v.\; \varphi\; v\}$$

PERSIST

A thunk can be forced twice:
its postcond must be duplicable

OVERESTIMATE

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

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

ANTICIPATE

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

$$\{\$K_{cr} \star (\$n \twoheadrightarrow wp\ f()\ \{\Box\ \varphi\})\}\quad create\ f\quad \{\lambda t.\ isThunk\ t\ n\ \varphi\}$$

$$\{\$K_{frc} \star isThunk\ t\ 0\ \varphi\}\quad force\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

A thunk can be forced twice:
its postcond must be persistent

OVERESTIMATE

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

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

ANTICIPATE

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

$$\{\$K_{\mathrm{cr}} \star (\$n \twoheadrightarrow\!\!\ast\; wp\; f()\,\{\,\square\,\varphi\})\}\quad create\ f\quad \{\lambda t.\ isThunk\ t\ n\ \varphi\}$$

$$\{\$K_{\mathrm{frc}} \star isThunk\ t\ 0\ \varphi\}\quad force\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

$$\mathrm{persistent}(isThunk\ t\ m\ \varphi)$$

OVERESTIMATE

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

PAY

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

ANTICIPATE

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

Simplified proof

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

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

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\textsc{Auth}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

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

$$\mathit{thunkInv}\ t\ \varphi \triangleq \exists n.\ \boxed{\bullet\, n}^{\gamma_t} \star \bigvee\begin{cases}\exists f.\ t \mapsto \mathit{Future}\ f \star (\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\square\,\varphi\}) \\ \quad t \mapsto \mathit{Busy} \\ \exists v.\ t \mapsto \mathit{Done}\ v \star \square\,\varphi\ v\end{cases}$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\textsc{Auth}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

- $\boxed{\bullet\, n}^{\gamma}$ asserts that the remaining cost is exactly n credits

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

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

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\ m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\textsc{Auth}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

- $\boxed{\bullet\ n}^{\gamma}$ asserts that the remaining cost is exactly n credits
- $\boxed{\circ\ m}^{\gamma}$ witnesses that the remaining cost is at most m credits

Simplified proof

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

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

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\textsc{Auth}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

- $\boxed{\bullet\, n}^{\gamma}$ asserts that the remaining cost is exactly n credits
- $\boxed{\circ\, m}^{\gamma}$ witnesses that the remaining cost is **at most** m credits
  $\Longrightarrow$ persistent ✔

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

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

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\text{AUTH}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

- $\boxed{\bullet\, n}^{\gamma}$ asserts that the remaining cost is exactly n credits
- $\boxed{\circ\, m}^{\gamma}$ witnesses that the remaining cost is **at most** m credits
  $\Longrightarrow$ persistent ✔

OVERESTIMATE: $\boxed{\circ\, m_1}^{\gamma} \twoheadrightarrow \boxed{\circ\, m_2}^{\gamma}$ if $m_1 \leq m_2$ ✔

PAY: $\boxed{\bullet\, n}^{\gamma} \Rrightarrow \boxed{\bullet\, (n-p)}^{\gamma} \star \boxed{\circ\, (n-p)}^{\gamma}$ ✔

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

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

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_t} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

Ghost state in $\textsc{Auth}(\bar{\mathbb{N}}, \min)$ reflects the remaining cost:

- $\boxed{\bullet\, n}^{\gamma}$ asserts that the remaining cost is exactly n credits
- $\boxed{\circ\, m}^{\gamma}$ witnesses that the remaining cost is **at most** m credits
  $\Longrightarrow$ persistent ✔

OVERESTIMATE: $\boxed{\circ\, m_1}^{\gamma} \mathrel{-\!\!*} \boxed{\circ\, m_2}^{\gamma}$ if $m_1 \leq m_2$ ✔

PAY: $\boxed{\bullet\, n}^{\gamma} \Rrightarrow \boxed{\bullet\, (n-p)}^{\gamma} \star \boxed{\circ\, (n-p)}^{\gamma}$ ✔

spec of *create*: ✔

spec of *force*: $(m = 0) \Rightarrow (n = 0) \Rightarrow (\$n \equiv \mathit{emp})$ ✔

A stream is a thunk which computes an element (its head) and another thunk (its tail):

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

A stream has a list of debits, one before each element:

$$
\begin{aligned}
&\mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \triangleq \\
&\quad \mathit{isThunk}\ s\ m_1\ (\lambda c_1.\ \exists s_2.\ c_1 = \mathit{Cons}(v_1, s_2) \star \\
&\qquad \mathit{isThunk}\ s_2\ m_2\ (\lambda c_2.\ \exists s_3.\ c_2 = \mathit{Cons}(v_2, s_3) \star \\
&\qquad\quad \ddots \\
&\qquad\qquad \mathit{isThunk}\ s_{n+1}\ 0\ (\lambda c_{n+1}.\ c_{n+1} = \mathit{Nil})\ldots))
\end{aligned}
$$

$$\{\$K_{\mathrm{ap}} \star \mathit{isStream}\ s\ [m_1, \dots, m_n]\ [v_1, \dots, v_n] \star \mathit{isStream}\ s'\ [m'_1, \dots, m'_{n'}]\ [v'_1, \dots, v'_{n'}]\}$$
$$\mathit{append}\ s\ s'$$
$$\{\lambda t.\ \mathit{isStream}\ t\ [A + m_1, \dots, A + m_n, m'_1, \dots, m'_{n'}]\ [v_1, \dots, v_n, v'_1, \dots, v'_{n'}]\}$$

$$\{\$K_{\mathrm{rv}} \star \mathit{isList}\ \ell\ [v_1, \dots, v_n]\}$$
$$\mathit{rev\_of\_list}\ \ell$$
$$\{\lambda s.\ \mathit{isStream}\ s\ [B \cdot n, 0, \dots, 0]\ [v_n, \dots, v_1]\}$$

PAYSTREAM

$$\frac{\mathit{isStream}\ s\ [m_1, m_2, \dots, m_n]\ [v_1, \dots, v_n] \qquad \$p}{\Rrightarrow \mathit{isStream}\ s\ [m_1 - p, m_2, \dots, m_n]\ [v_1, \dots, v_n]}$$

ANTICIPATE+OVERESTIMATESTREAM

$$\frac{\mathit{isStream}\ s\ [m_1, \dots, m_n]\ [v_1, \dots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow \mathit{isStream}\ s\ [m'_1, \dots, m'_n]\ [v_1, \dots, v_n]}$$

The banker's queue needs anticipation of debits in streams...

$$\frac{\textsc{anticipate+overestimateStream} \quad isStream\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow isStream\ s\ [m'_1, \ldots, m'_n]\ [v_1, \ldots, v_n]}$$

...therefore in thunks:

$$\frac{\textsc{anticipate} \quad isThunk\ t\ m\ \varphi}{\Rrightarrow isThunk\ t\ (m + n)\ (\$n \star \varphi)}$$

The banker's queue needs anticipation of debits in streams...

$$\frac{\textsc{anticipate+overestimateStream} \quad \mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow \mathit{isStream}\ s\ [m'_1, \ldots, m'_n]\ [v_1, \ldots, v_n]}$$

...therefore in thunks:

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

nonsensical, thunk postconditions must be persistent

The banker's queue needs anticipation of debits in streams...

$$\frac{\text{isStream } s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow \text{isStream } s\ [m'_1, \ldots, m'_n]\ [v_1, \ldots, v_n]}\ \textsc{anticipate+overestimateStream}$$

...therefore in thunks:

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

The banker's queue needs anticipation of debits in streams...

$$\frac{\text{isStream } s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \leq k} m_i \leq \sum_{i \leq k} m'_i}{\Rrightarrow \text{isStream } s\ [m'_1, \ldots, m'_n]\ [v_1, \ldots, v_n]}\ \textsc{anticipate+overestimateStream}$$

...therefore in thunks:

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

Example: from rules PAY and ANTICIPATE we can derive:

$$\frac{\text{isThunk } t_1\ m_1\ (\lambda t_2.\ \text{isThunk } t_2\ m_2\ \varphi) \qquad \dfrac{}{\$n \star \text{isThunk } t_2\ m_2\ \varphi \Rrightarrow \text{isThunk } t_2\ (m_2 - n)\ \varphi}\ (\textsc{pay})}{\Rrightarrow \text{isThunk } t_1\ (m_1 + n)\ (\lambda t_2.\ \text{isThunk } t_2\ (m_2 - n)\ \varphi)}\ (\textsc{anticipate})$$

Problems:

- known upper bounds $\boxed{\circ\, m}$ must remain valid $\Longrightarrow$ can't increase $\boxed{\bullet\, n}$
- φ is fixed in the invariant $\Longrightarrow$ can't change it

Solution: stack a new debit, with a new invariant, on top of the old one!

Example scenario:

create a thunk with debit 5
and postcondition A

$isThunk\ t\ 5\ A$

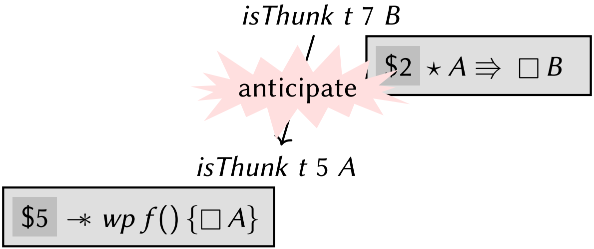
$\boxed{\$5 \mathrel{-\!\!*} wp\ f()\ \{\Box A\}}$

Example scenario:

$isThunk\ t\ 5\ A$

$\$5 \mathrel{-\!\!*} wp\ f()\ \{\Box A\}$

Example scenario:



A stack of summand debits

Example scenario:

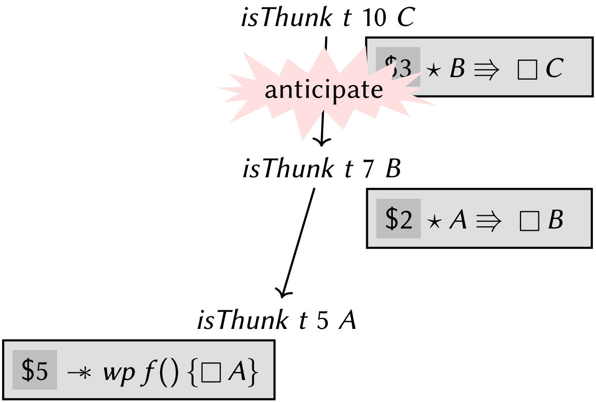
$$\mathit{isThunk}\ t\ 7\ B$$

$$\$2 \star A \Rrightarrow \Box B$$

$$\mathit{isThunk}\ t\ 5\ A$$

$$\$5 \twoheadrightarrow \mathit{wp}\ f()\ \{\Box A\}$$

Example scenario:



A stack of summand debits

Example scenario:

$isThunk\ t\ 10\ C$

$\$3 \star B \Rrightarrow \square C$

$isThunk\ t\ 7\ B$

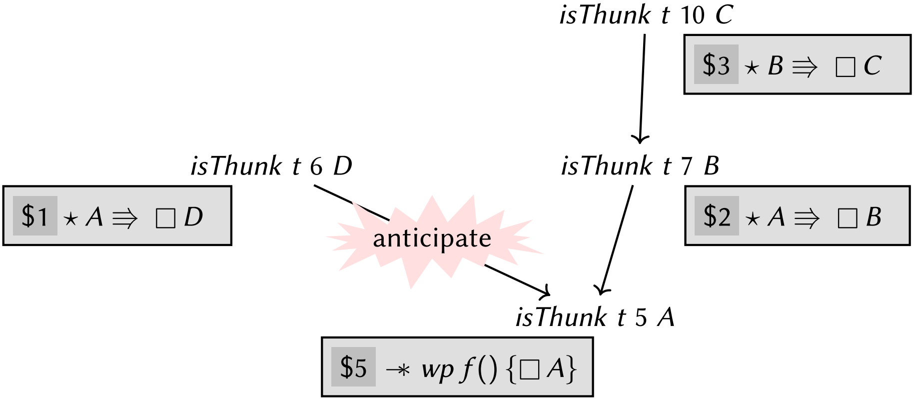
$\$2 \star A \Rrightarrow \square B$

$isThunk\ t\ 5\ A$

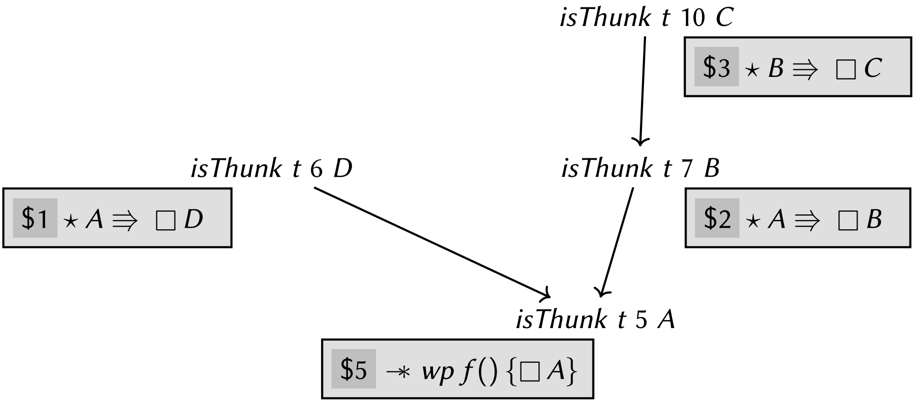
$\$5 \mathrel{-\!\!*} wp\ f()\ \{\square A\}$

A tree of summand debits

Example scenario:

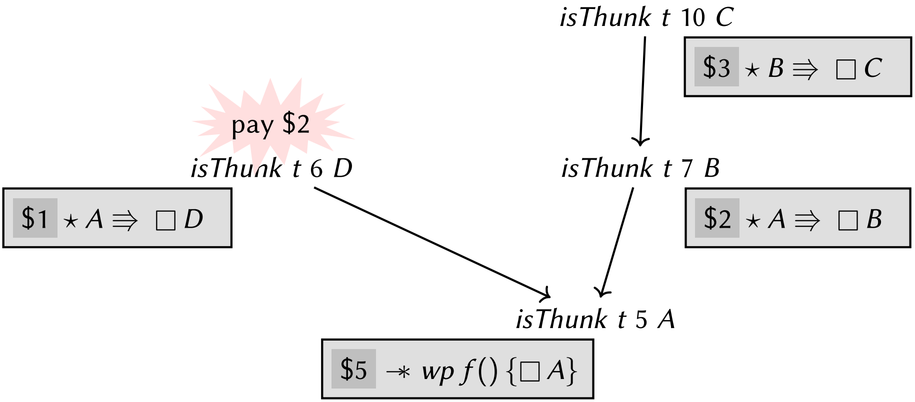


Example scenario:

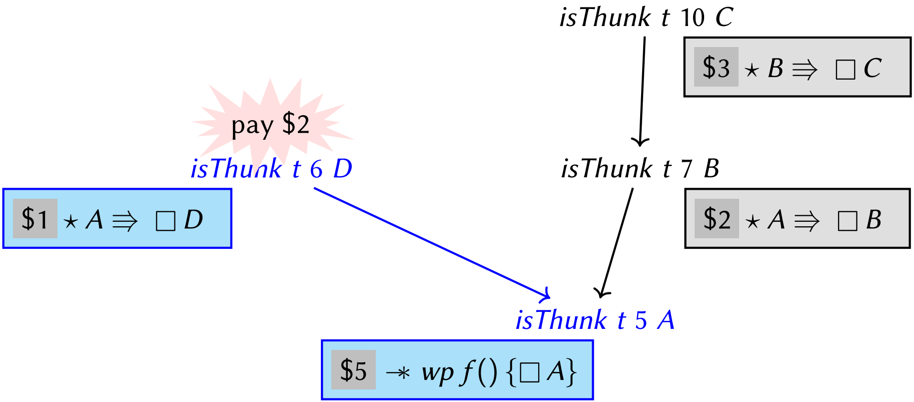


A tree of summand debits

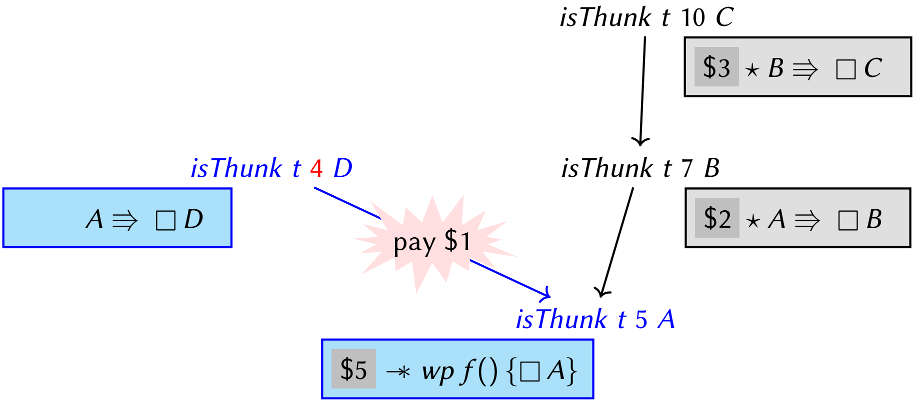
Example scenario:



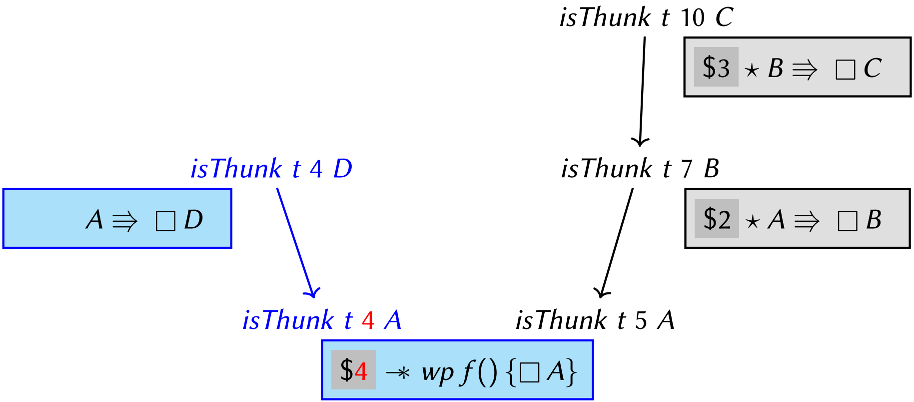
Example scenario:



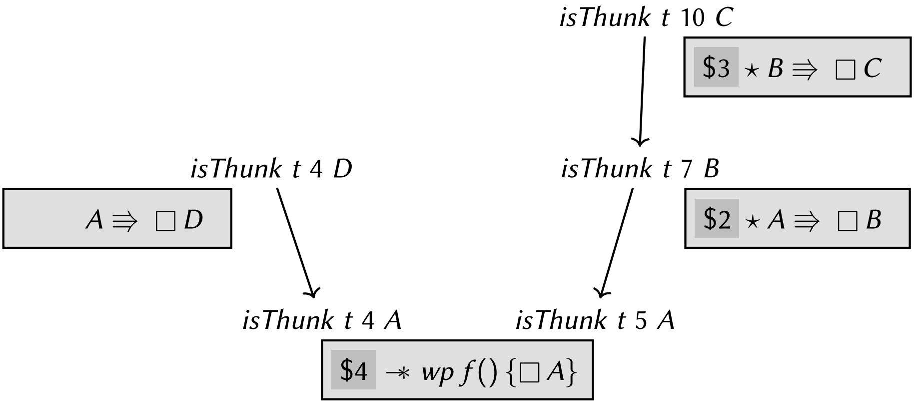
Example scenario:



Example scenario:

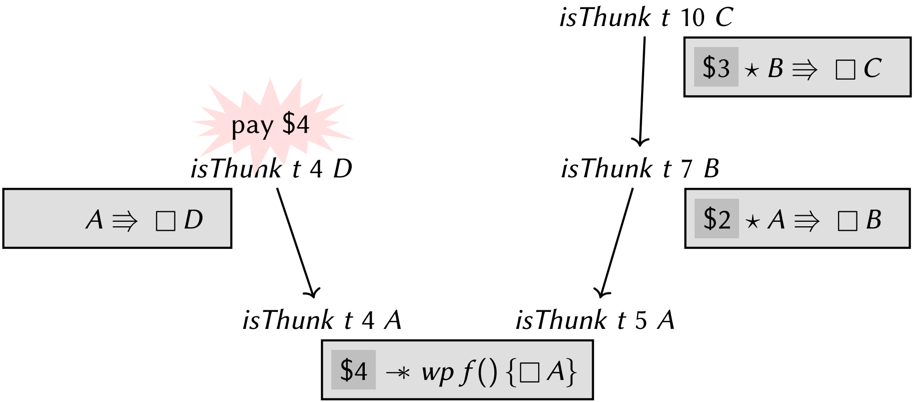


Example scenario:



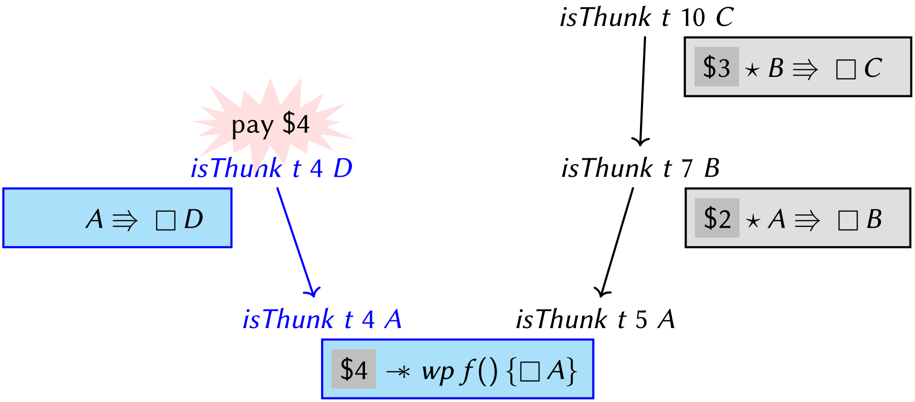
A tree of summand debits

Example scenario:



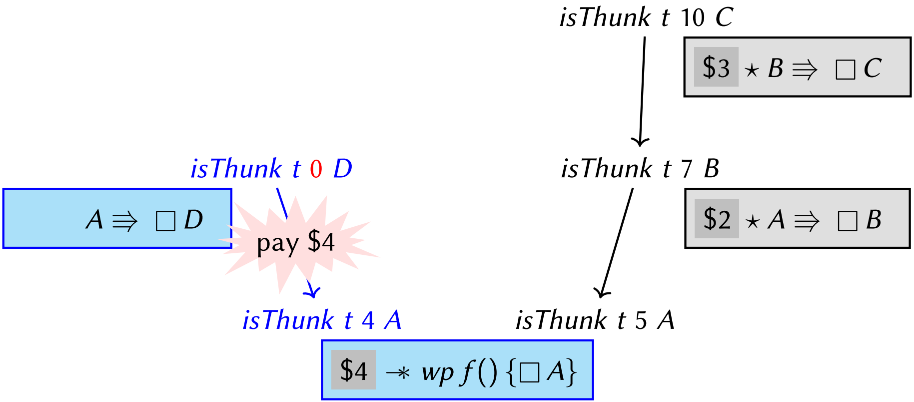
A tree of summand debits

Example scenario:

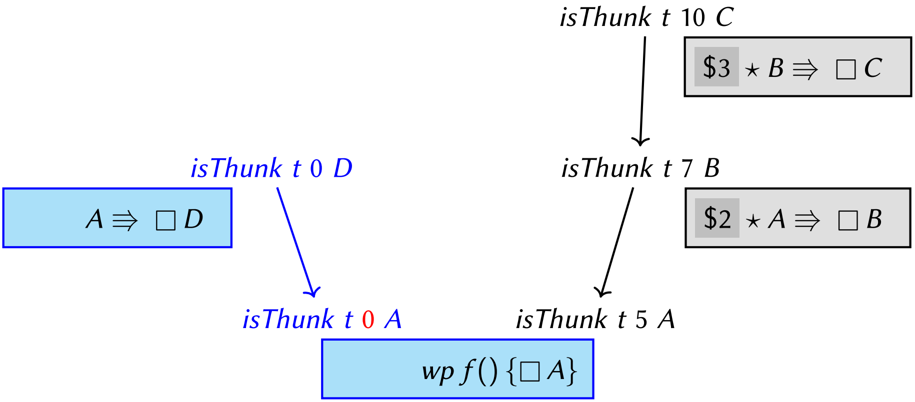


A tree of summand debits

Example scenario:

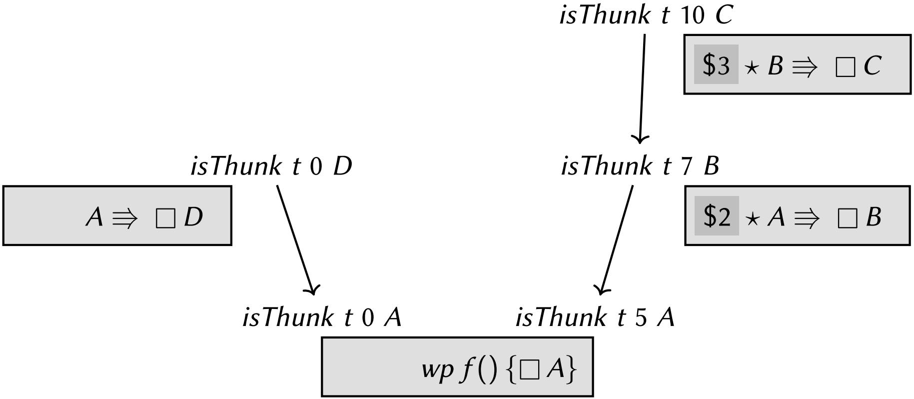


Example scenario:



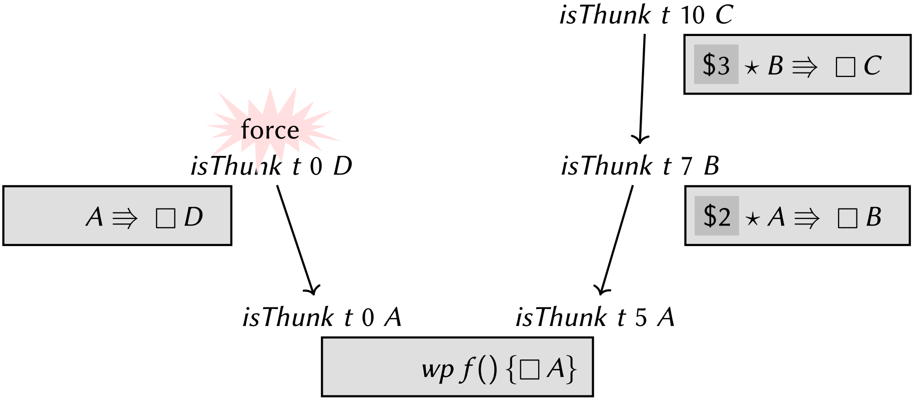
A tree of summand debits

Example scenario:

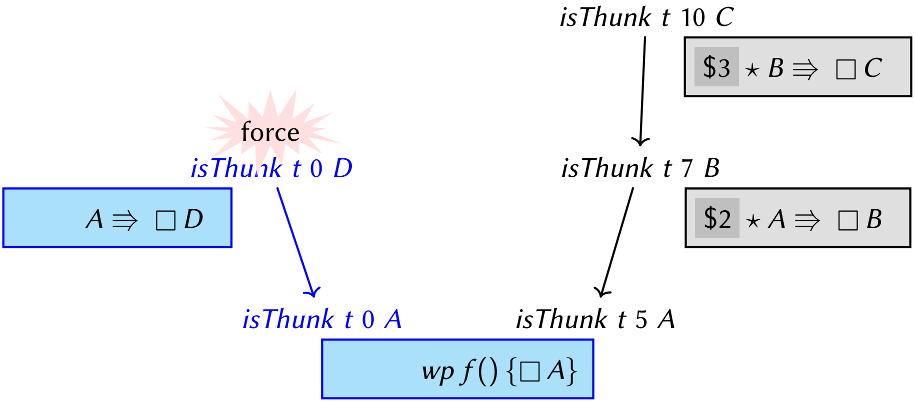


A tree of summand debits

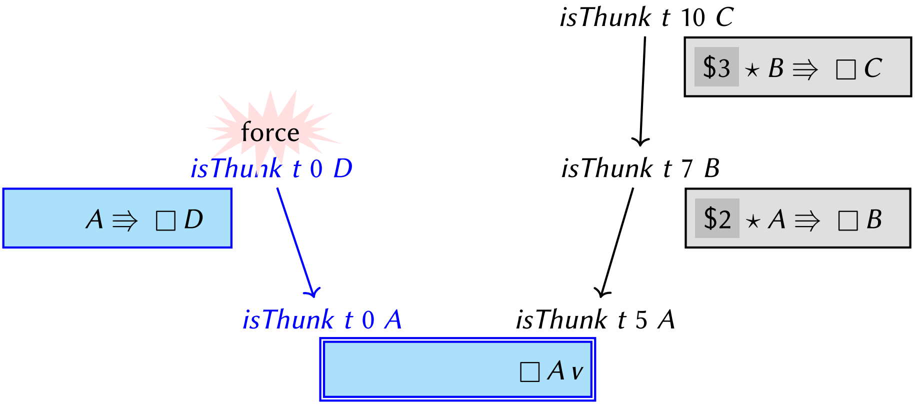
Example scenario:



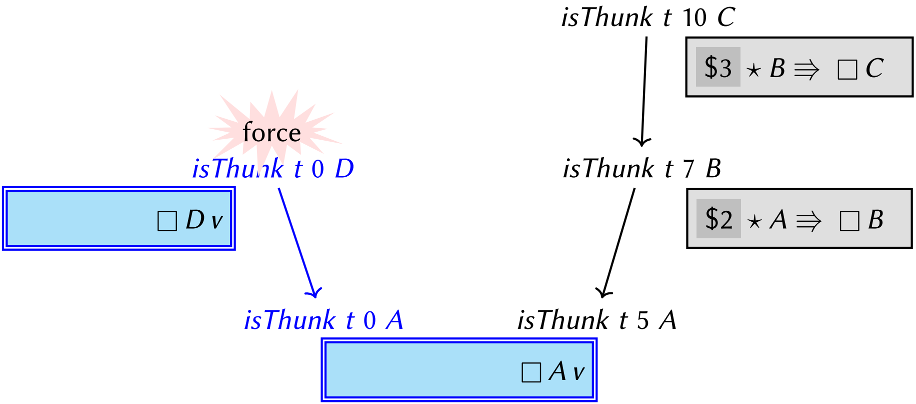
Example scenario:



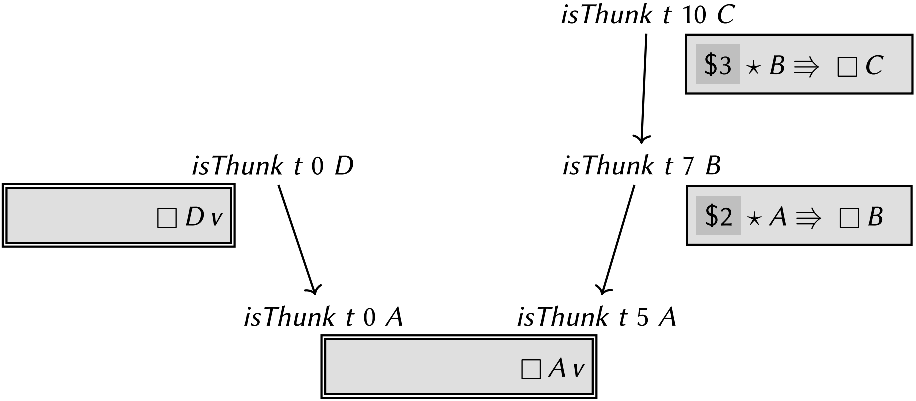
Example scenario:



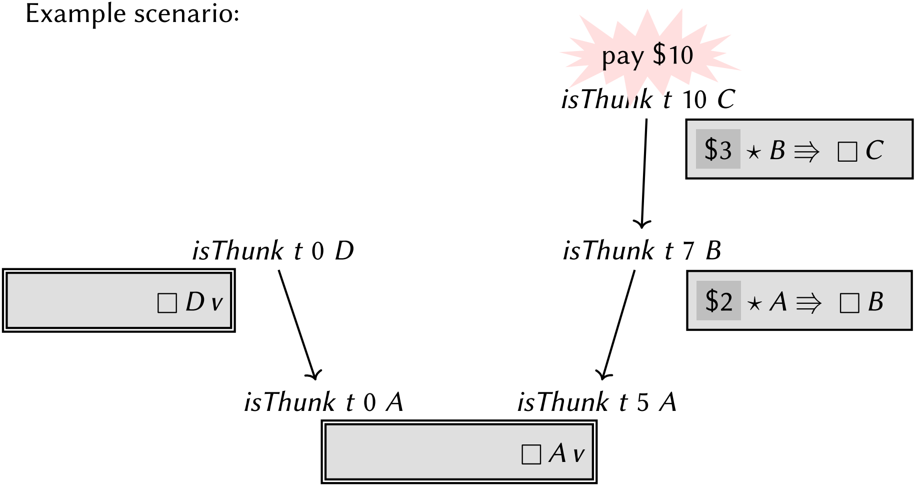
Example scenario:



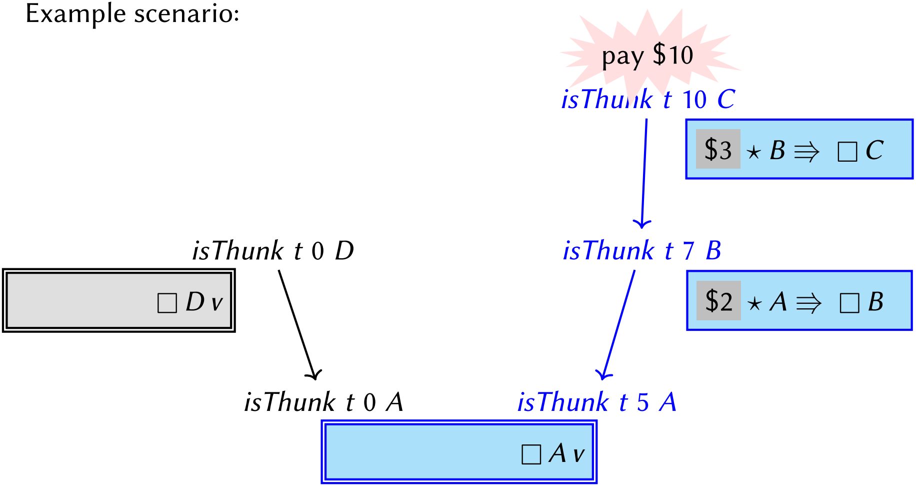
Example scenario:



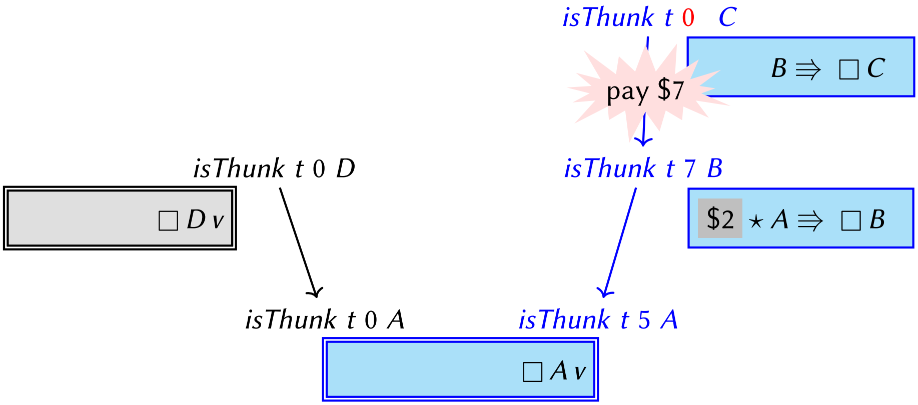
Example scenario:



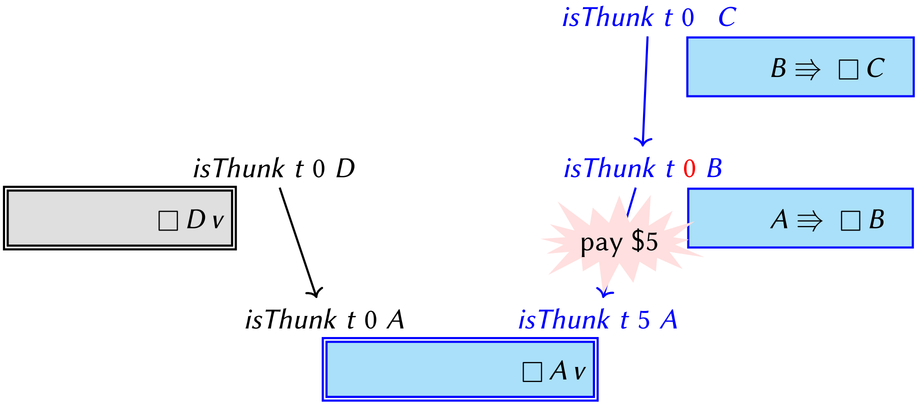
Example scenario:



Example scenario:

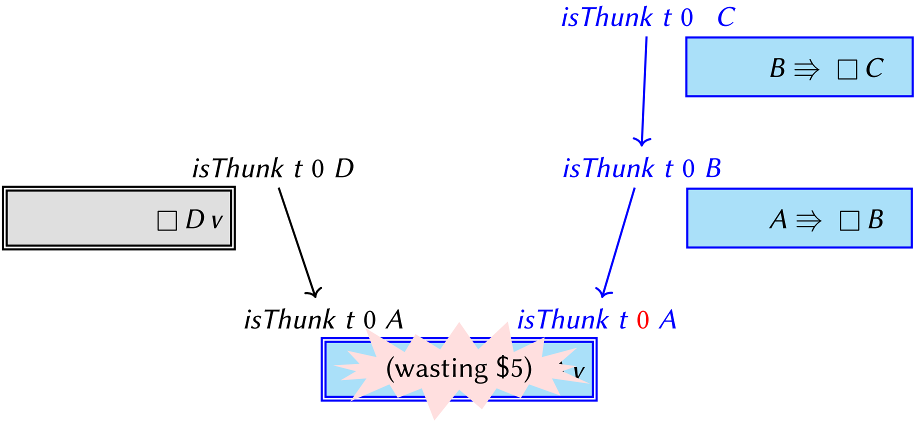


Example scenario:

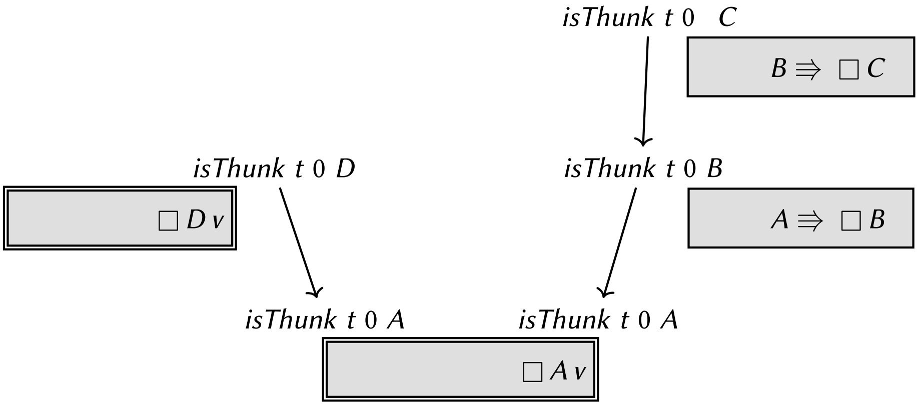


A tree of summand debits

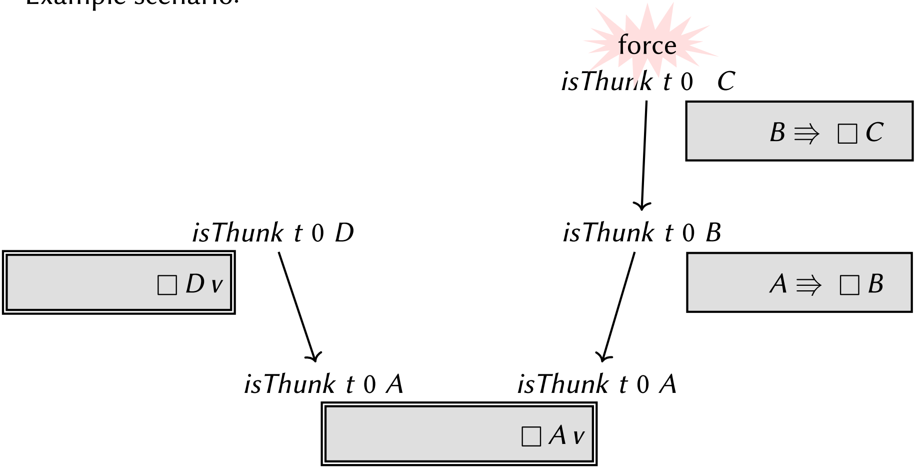
Example scenario:



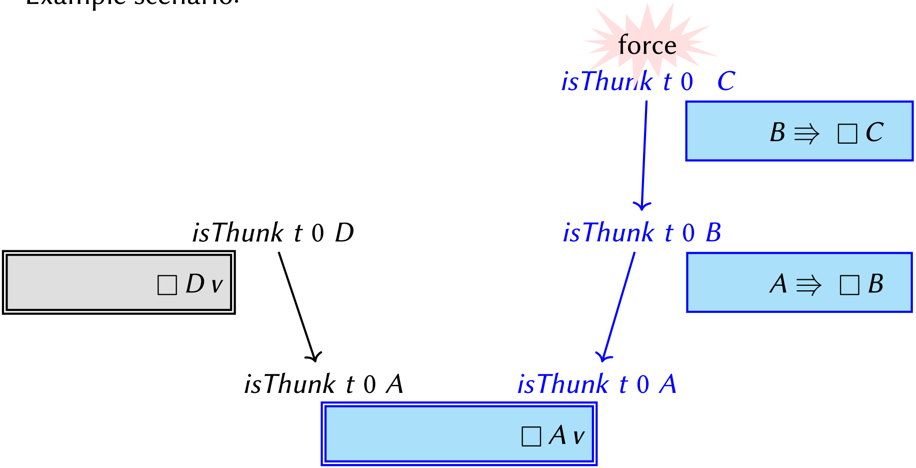
Example scenario:



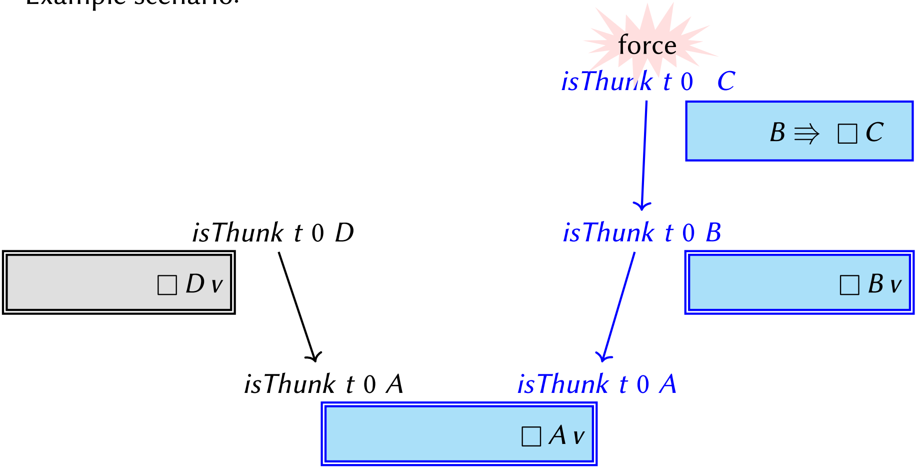
Example scenario:



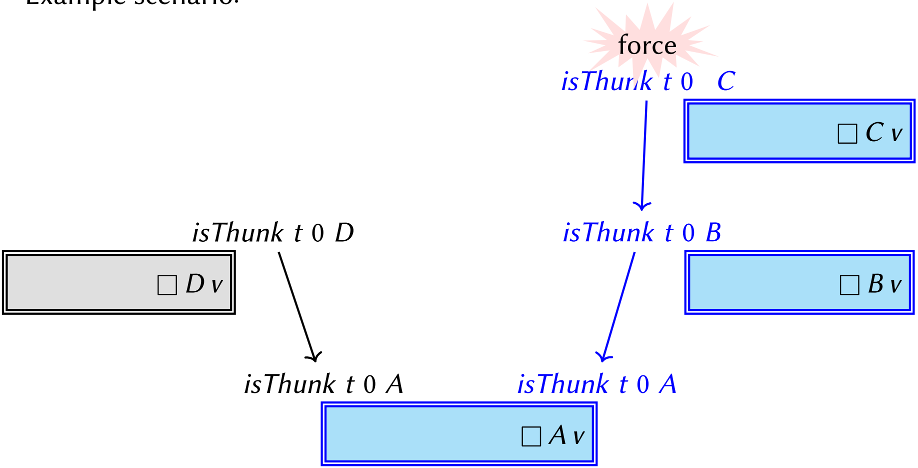
Example scenario:



Example scenario:



Example scenario:



We stack a new invariant and ghost state each time ANTICIPATE is used

Each height $h \in \mathbb{N}$ has its own debit $\gamma_{t,h}$

$$\mathit{thunkInv}\ t\ \varphi \triangleq \exists n.\ \boxed{\bullet\, n}^{\gamma_{t,0}} \star \bigvee \begin{cases} \exists f.\ t \mapsto \mathit{Future}\ f \star (\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\Box\,\varphi\}) \\ \quad t \mapsto \mathit{Busy} \\ \exists v.\ t \mapsto \mathit{Done}\ v \star \Box\,\varphi\ v \end{cases}$$

$$\mathit{csqInv}_h\ t\ \varphi\ \psi \triangleq \exists n.\ \boxed{\bullet\, n}^{\gamma_{t,h}} \star \bigvee \begin{cases} \forall v.\ \$n \star \varphi\ v \Rrightarrow \Box\,\psi\ v \\ \Box\,\psi\ v \end{cases}$$

$$\mathit{isThunk}_0\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_{t,0}} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

$$\mathit{isThunk}_h\ t\ m\ \varphi \triangleq \exists m', \psi.\ m' \leq m \star \boxed{\circ\, m'}^{\gamma_{t,h}} \star \boxed{\mathit{csqInv}_h\ t\ \psi\ \varphi} \star \mathit{isThunk}_{h-1}\ t\ (m - m')\ \psi$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists h.\ \mathit{isThunk}_h\ t\ m\ \varphi$$

We stack a new invariant and ghost state each time ANTICIPATE is used

Each height $h \in \mathbb{N}$ has its own debit $\gamma_{t,h}$

$$\mathit{thunkInv}\ t\ \varphi \triangleq \exists n.\ \boxed{\bullet\, n}^{\gamma_{t,0}} \star \bigvee \begin{cases} \exists f.\ t \mapsto \mathit{Future}\ f \ \star\ (\$n \mathrel{-\!\!*} \mathit{wp}\ f()\ \{\Box\,\varphi\}) \\ \quad t \mapsto \mathit{Busy} \\ \exists v.\ t \mapsto \mathit{Done}\ v \ \star\ \Box\,\varphi\ v \end{cases}$$

$$\mathit{csqInv}_h\ t\ \varphi\ \psi \triangleq \exists n.\ \boxed{\bullet\, n}^{\gamma_{t,h}} \star \bigvee \begin{cases} \forall v.\ \$n \star \varphi\ v \Rrightarrow \Box\,\psi\ v \\ \Box\,\psi\ v \end{cases}$$

$$\mathit{isThunk}_0\ t\ m\ \varphi \triangleq \boxed{\circ\, m}^{\gamma_{t,0}} \star \boxed{\mathit{thunkInv}\ t\ \varphi}$$

$$\mathit{isThunk}_h\ t\ m\ \varphi \triangleq \exists m', \psi.\ m' \leq m \star \boxed{\circ\, m'}^{\gamma_{t,h}} \star \boxed{\mathit{csqInv}_h\ t\ \psi\ \varphi} \star \mathit{isThunk}_{h-1}\ t\ (m - m')\ \psi$$

$$\mathit{isThunk}\ t\ m\ \varphi \triangleq \exists h.\ \mathit{isThunk}_h\ t\ m\ \varphi$$

Omitted: ghost state in AUTH(EX() + AG(VAL)) for remembering the value computed

Conclusion

Three library layers: thunks (proven), streams (proven), queues (WIP)

In this talk:

- **anticipation** of debit
  - we overlooked it at first
  - non-trivial proof: tree of debits, many invariants
- streams are chains of nested thunks

Not in this talk:

- **reentrancy** forbidden statically
  - non-atomic invariants $\Longrightarrow$ thunks have **namespaces**
  - avoid reentrant streams $\Longrightarrow$ streams have **generations** (internally)
- full proof of the banker's queue
- **ghost debits**! (WIP)

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.

Ghost debits?

$$\frac{\$m \Rrightarrow \Box Q}{\Rrightarrow \mathit{debit}\ m\ Q}\ \textsc{createDebit}$$

$$\frac{\mathit{debit}\ 0\ Q}{\Rrightarrow \triangleright Q}\ \textsc{forceDebit}$$

$$\textsc{persistDebit}\quad \text{persistent}(\mathit{debit}\ m\ Q)$$

$$\frac{\mathit{debit}\ m_1\ Q \qquad m_1 \leq m_2}{\mathit{debit}\ m_2\ Q}\ \textsc{overestimateDebit}$$

$$\frac{\mathit{debit}\ m\ Q \qquad \$p}{\Rrightarrow \mathit{debit}\ (m - p)\ Q}\ \textsc{payDebit}$$

$$\frac{\mathit{debit}\ m\ Q \qquad \$n \star Q \Rrightarrow \Box Q'}{\Rrightarrow \mathit{debit}\ (m + n)\ Q'}\ \textsc{anticipateDebit}$$

Actual implementation of thunks

```
type 'α thunk = 'α thunk_contents ref
and 'α thunk_contents =
  | Future of (unit → 'α)
  | Done of 'α

let create f =
  ref (Future f)

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

No reentrancy detection (2 states only) $\Longrightarrow$ static proof obligations

Specification of thunks

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

$$\{\$K_{frc} \star isThunk\ t\ 0\ \varphi\}\quad force\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

$$\text{persistent}(isThunk\ t\ m\ \varphi)$$

OVERESTIMATE

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

PAY

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

ANTICIPATE

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

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

$$\{\$K_{frc} \star isThunk\ t\ 0\ \varphi\}\quad force\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

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

OVERESTIMATE

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

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

ANTICIPATE

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

$$\{\$K_{cr} \star (\$n \twoheadrightarrow wp\ f()\ \{\Box\ \varphi\})\}\quad create\ f\quad \{\lambda t.\ isThunk\ t\ n\ \varphi\}$$

$$\{\$K_{frc} \star isThunk\ t\ 0\ \varphi\}\quad force\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

$$\text{persistent}(isThunk\ t\ m\ \varphi)$$

OVERESTIMATE

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

PAY

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

ANTICIPATE

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

$$\{\$K_{\mathrm{cr}} \star (\$n \twoheadrightarrow\!\!\!* \ \mathit{wp}\ f()\ \{\ \Box\,\varphi\})\}\quad \mathit{create}\ f\quad \{\lambda t.\ \mathit{isThunk}\ t\ n\ \varphi\}$$

$$\{\$K_{\mathrm{frc}} \star \mathit{isThunk}\ t\ 0\ \varphi\}\quad \mathit{force}\ t\quad \{\lambda v.\ \varphi\ v\}$$

PERSIST

$$\mathrm{persistent}(\mathit{isThunk}\ t\ m\ \varphi)$$

Reentrancy?

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

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

Specification of thunks

One *canForce* token exists
at the beginning of the world

canForceExcl

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

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

$$\{\$K_{\mathrm{frc}} \star \mathit{isThunk}\ t\ 0\ \varphi \star \mathit{canForce}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v \star \mathit{canForce}\}$$

persist

$$\mathrm{persistent}(\mathit{isThunk}\ t\ m\ \varphi)$$

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

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

One *canForce* token exists
at the beginning of the world

CANFORCEEXCL

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

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

$$\{\$K_{\mathrm{frc}} \star \mathit{isThunk}\ t\ 0\ \varphi \star \mathit{canForce}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v \star \mathit{canForce}\}$$

How to force a thunk from another thunk?

PERSIST

$$\text{persistent}(\mathit{isThunk}\ t\ m\ \varphi)$$

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

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

One *canForce* $\top$ token exists at the beginning of the world

CANFORCEEXCL

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

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

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

PERSIST

$$\mathrm{persistent}(\mathit{isThunk}\ t\ \mathcal{N}\ m\ \varphi)$$

OVERESTIMATE

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

PAY

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

ANTICIPATE

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

Specification of thunks

One *canForce* $\top$ token exists at the beginning of the world

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

$$\{\$K_{\mathrm{cr}} \star (\$n \twoheadrightarrow \mathit{wp}\ f()\ \{\square\,\varphi\})\}\ \ \mathit{create}\ f\ \ \{\lambda t.\ \mathit{isThunk}\ t\ \mathcal{N}\ \ n\ \varphi\}$$

$$\{\$K_{\mathrm{frc}} \star \mathit{isThunk}\ t\ \mathcal{N}\ \ 0\ \varphi \star \mathit{canForce}\ \mathcal{N}\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v \star \mathit{canForce}\ \mathcal{N}\}$$

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

PERSIST
$$\text{persistent}(\mathit{isThunk}\ t\ \mathcal{N}\ \ m\ \varphi)$$

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

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

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

Specification of thunks

One *canForce* $\top$ token exists at the beginning of the world

CANFORCEEXCL

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

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

$$\{\$K_{\mathrm{frc}} \star \mathit{isThunk}\ t\ \mathcal{N}\ 0\ R\ \varphi \star \mathit{canForce}\ \mathcal{N} \star R\}\ \ \mathit{force}\ t\ \ \{\lambda v.\ \varphi\ v \star \mathit{canForce}\ \mathcal{N} \star R\}$$

PERSIST

$$\mathrm{persistent}(\mathit{isThunk}\ t\ \mathcal{N}\ m\ R\ \varphi)$$

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

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

```
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 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

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
```

(Selected rules) Specification of streams

$$\{\$K_{\mathrm{ap}} \star \mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \star \mathit{isStream}\ s'\ [m'_1, \ldots, m'_{n'}]\ [v'_1, \ldots, v'_{n'}]\}$$
$$\mathit{append}\ s\ s'$$
$$\{\lambda t.\ \mathit{isStream}\ t\ [A + m_1, \ldots, A + m_n, m'_1, \ldots, m'_{n'}]\ [v_1, \ldots, v_n, v'_1, \ldots, v'_{n'}]\}$$

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

PAYSTREAM

$$\frac{\mathit{isStream}\ s\ [m_1, m_2, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \$p}{\Rrightarrow \mathit{isStream}\ s\ [m_1 - p, m_2, \ldots, m_n]\ [v_1, \ldots, v_n]}$$

ANTICIPATE+OVERESTIMATESTREAM

$$\frac{\mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \qquad \forall k.\ \sum_{i \le k} m_i \le \sum_{i \le k} m'_i}{\Rrightarrow \mathit{isStream}\ s\ [m'_1, \ldots, m'_n]\ [v_1, \ldots, v_n]}$$

We forbid recursive streams by using **generations** $g \in \mathbb{N}$:

$$
\begin{array}{l}
\mathit{isStream}\ s\ [m_1, \ldots, m_n]\ [v_1, \ldots, v_n] \triangleq \\
\quad \exists g_1.\ \mathit{isThunk}\ s\ \mathcal{N}_{g_1}\ m_1\ (\mathit{naInvTok}\ \mathcal{E}_{g_1})\ (\lambda c_1.\ \exists s_2.\ c_1 = \mathit{Cons}(v_1, s_2) \star \\
\qquad \exists g_2 \leq g_1.\ \mathit{isThunk}\ s_2\ \mathcal{N}_{g_2}\ m_2\ (\mathit{naInvTok}\ \mathcal{E}_{g_2})\ (\lambda c_2.\ \exists s_3.\ c_2 = \mathit{Cons}(v_2, s_3) \star \\
\qquad\quad \ddots \\
\qquad\qquad \exists g_{n+1} \leq g_n.\ \mathit{isThunk}\ s_{n+1}\ \mathcal{N}_{g_{n+1}}\ 0\ (\mathit{naInvTok}\ \mathcal{E}_{g_{n+1}})\ (\lambda c_{n+1}.\ c_{n+1} =
\end{array}
$$

where:

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