Reasoning about Amortized Cost: From Syntax to Semantics

FPDag 2026 @ Nijmegen

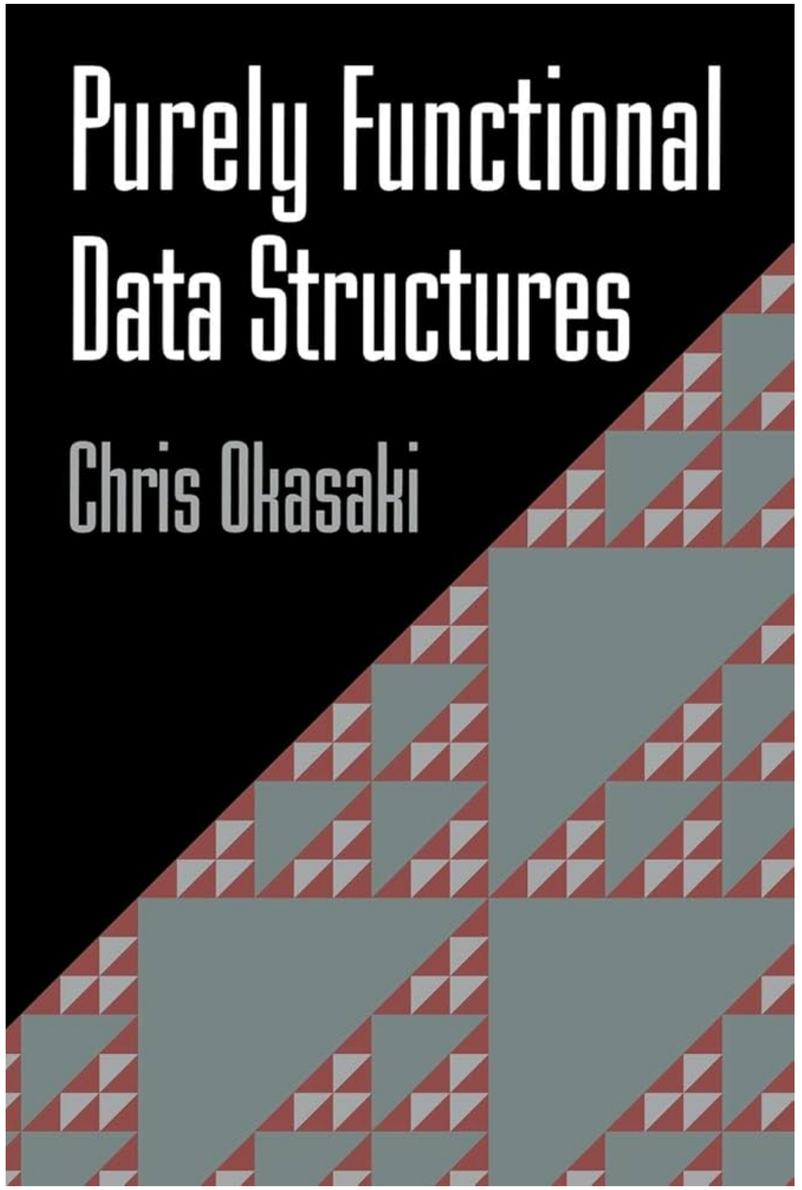
David Binder, University of Kent (joint work with Dominic Orchard, Vineet Rajani and David Corfield)

Part I

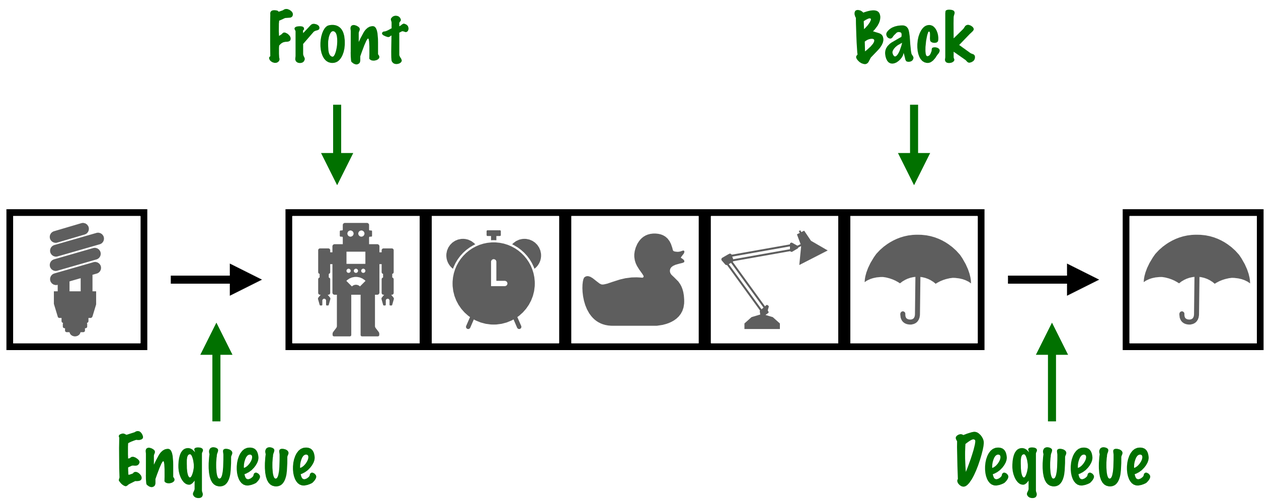
Fundamentals of Amortized Cost Analysis

Okasaki: Purely Functional Data Structures

Introduces key techniques for analyzing the complexity of algorithms on functional data structures

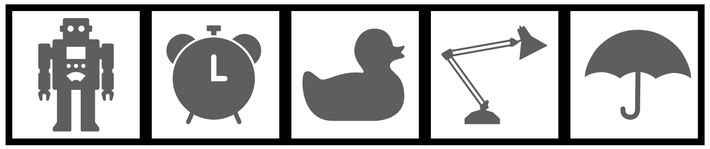


FIFO Queues

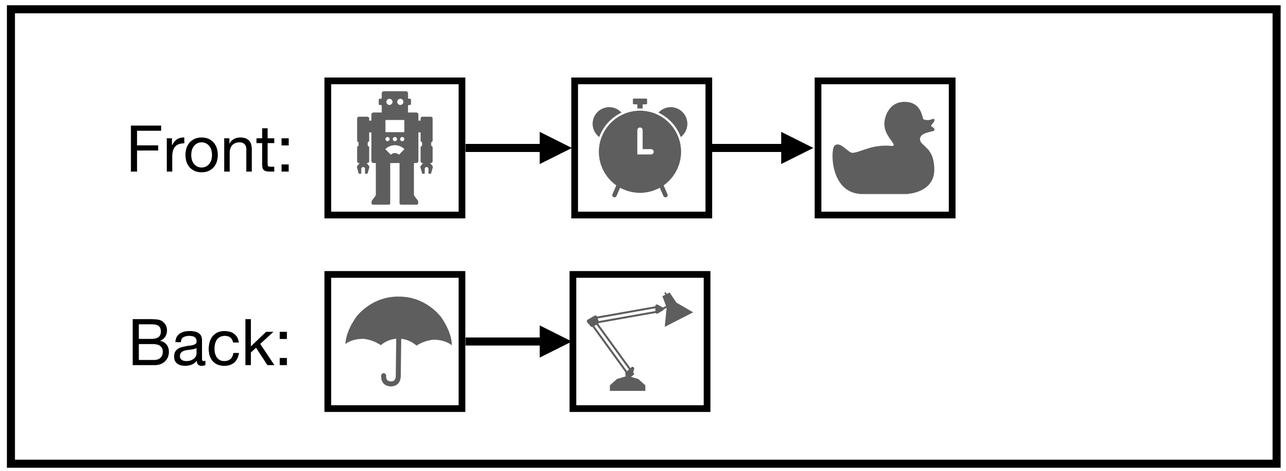


The Two-List Implementation

External View:



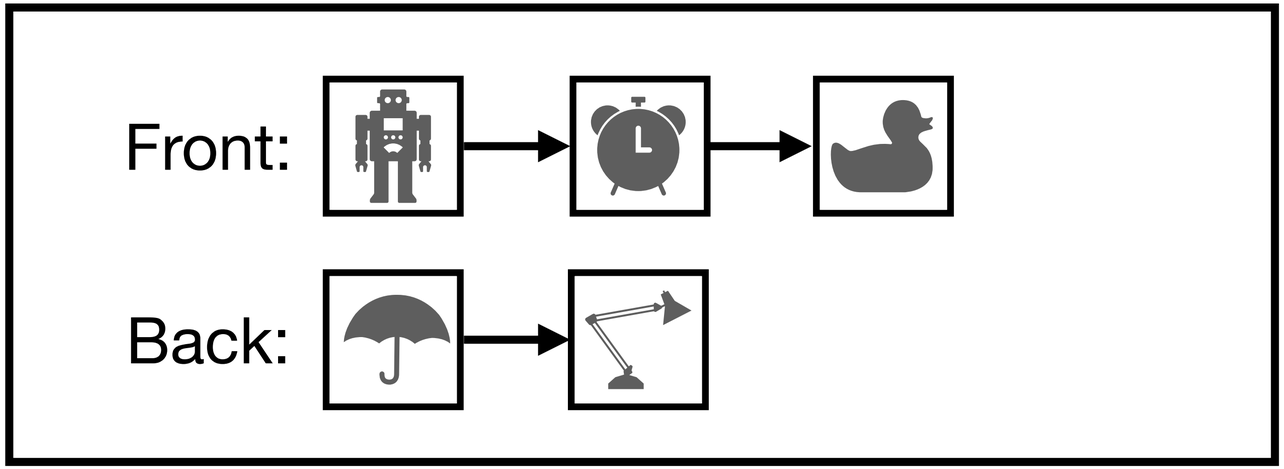
Internal Representation:



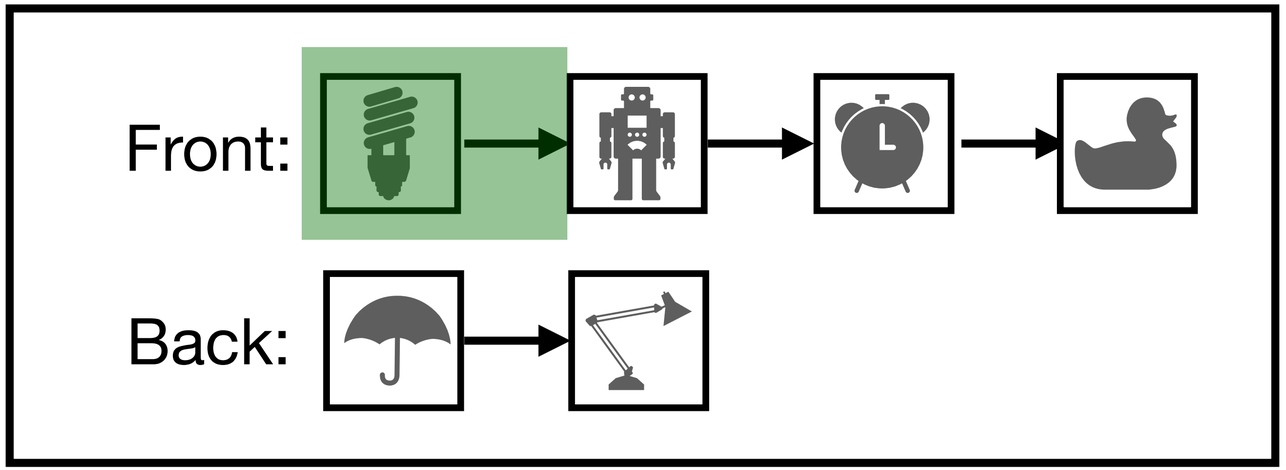
We represent a queue by two singly-linked lists. The back of the queue is stored in reverse order.

Enqueue

Before:

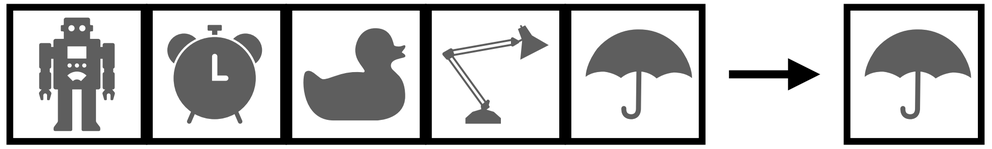


After:

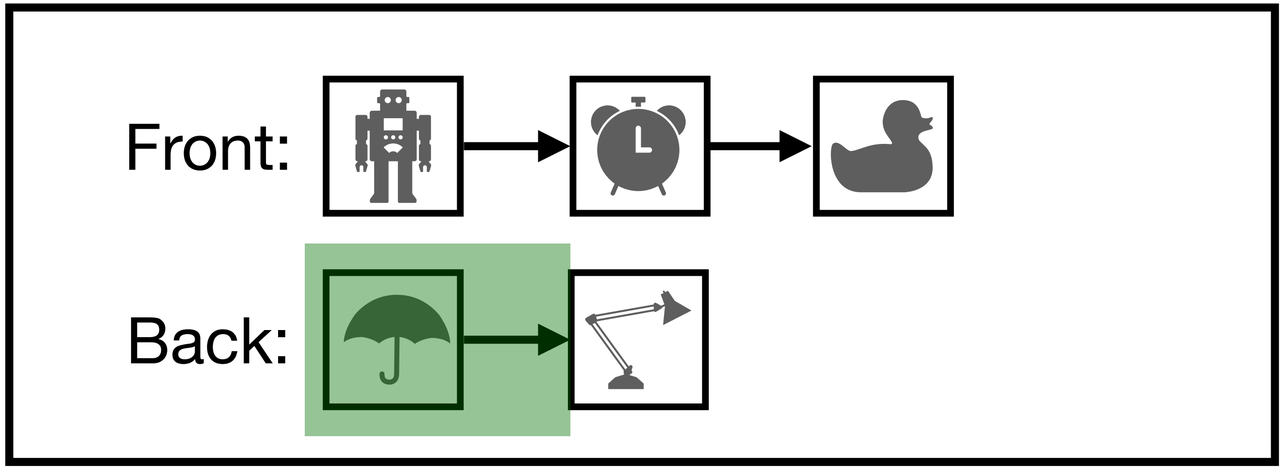


Worst Case: O(1)

Dequeue

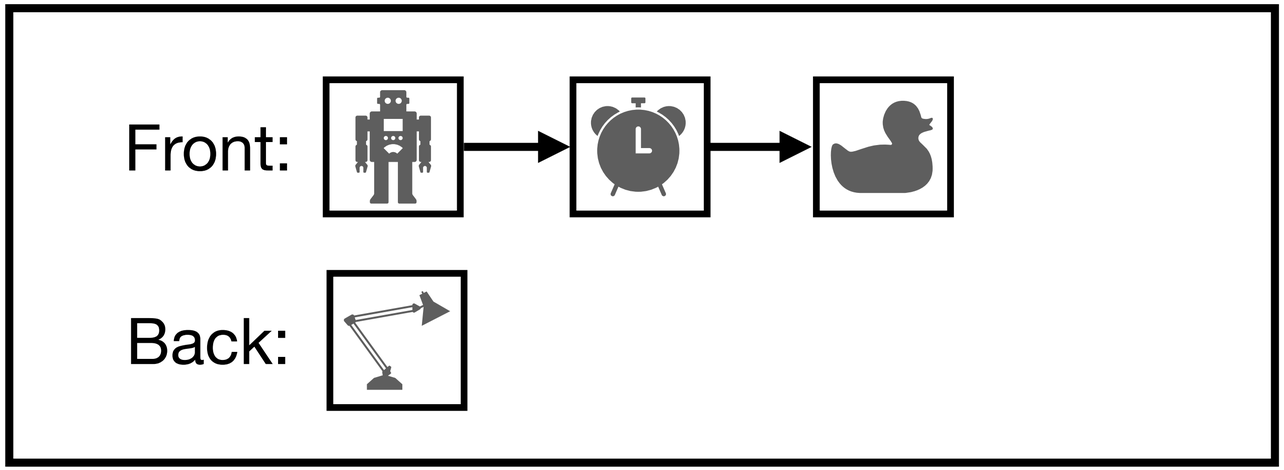


Before:

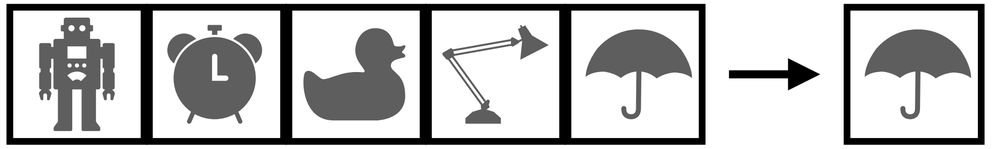


Worst Case: O(1)

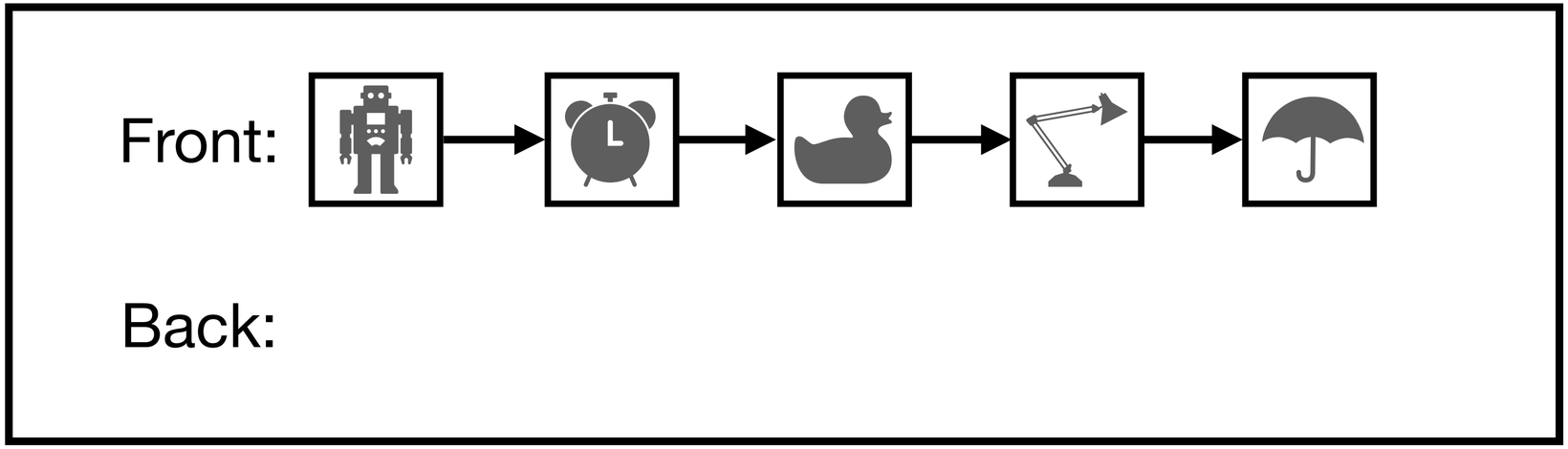
After:



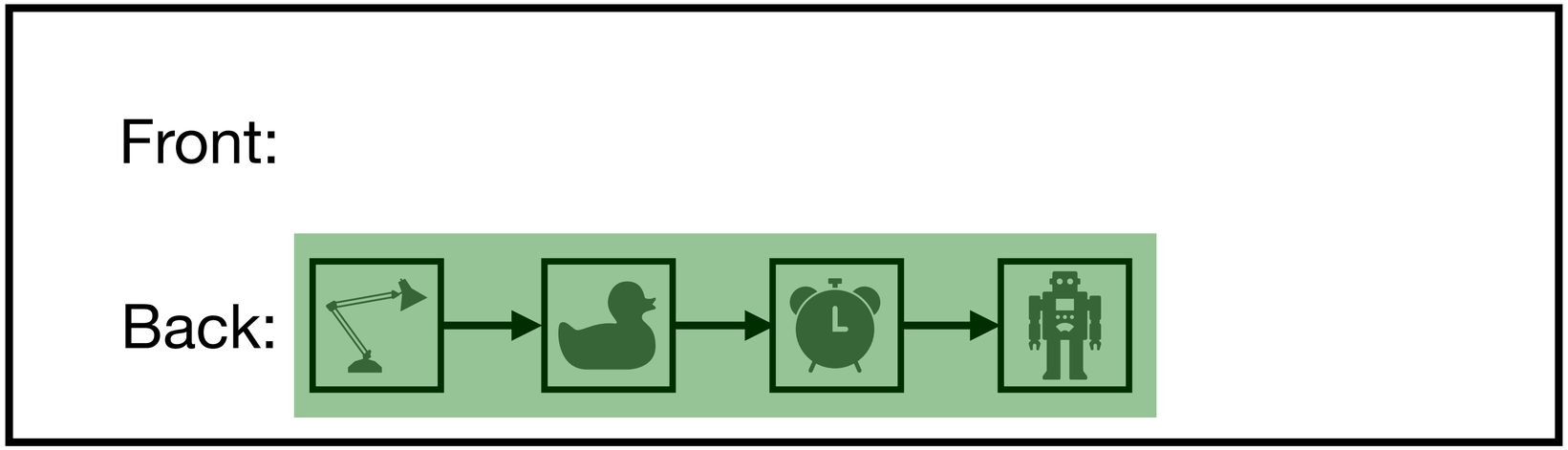
Dequeue



Before:



After:

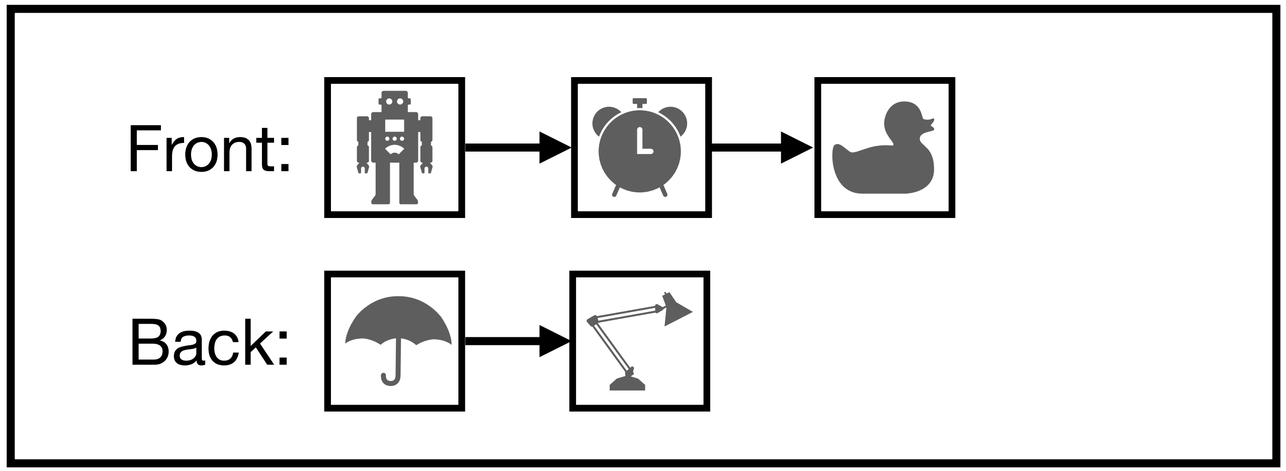


Worst Case: O(n)

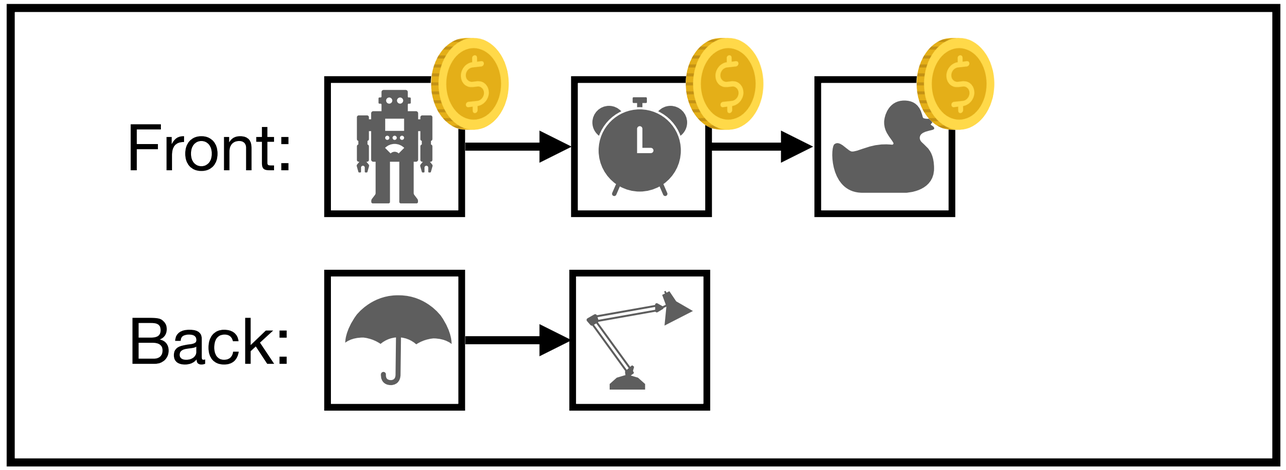
The Gist of the Banker's Method: Store Potential in the Data Structure

Amortization with Potential

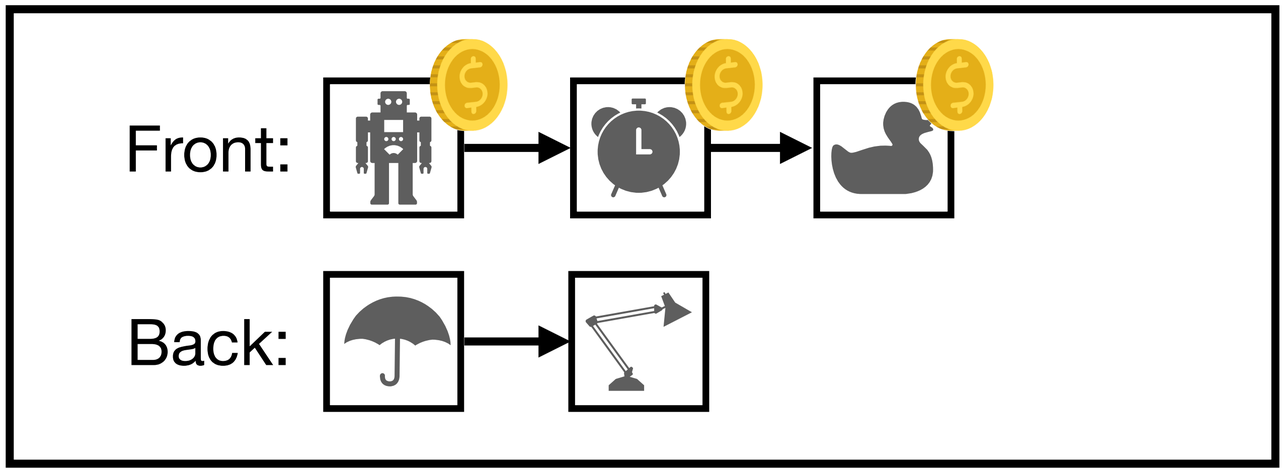
Without Potential:



With Potential:



Cost of Enqueue and Dequeue



- Enqueue: Cost 1 for cons-ing an element to the front list plus cost 1 for storing potential.
- Dequeue (simple): Cost 1 for removing the head of the back list.
- Dequeue (reverse): Use the stored potential to pay for the cost of reversing the list which results in a cost of 0.
- Together ensures amortized cost of O(1): Any sequence of n enqueue and dequeue operations costs at most O(n).

Takeaway: We need a way to store potential within a data structure

Part II

A Type System for Amortized Cost

From Whiteboard Reasoning to Type Theory

- To track cost we use graded monads
- To represent stored potential we use a phantom type $[p]\tau$
- The type system is affine because we are not allowed to duplicate potential
- Based on V. Rajani, M. Gaboardi, D. Garg, and J. Hoffmann:
  A unifying type-theory for higher-order (amortized) cost analysis

Graded Monads

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{ret}\ t : \mathbb{M}\ \blacksquare\ \tau}\ \textsc{Return}$$

$$\frac{\Gamma \vdash t_1 : \mathbb{M}\ \blacksquare\ \tau_1 \qquad \Gamma, x : \tau_1 \vdash t_2 : \mathbb{M}\ \blacksquare\ \tau_2}{\Gamma \vdash \mathsf{bind}\ x = t_1\ \mathsf{in}\ t_2 : \mathbb{M}\ \blacksquare\blacksquare\blacksquare\ \tau_2}\ \textsc{Bind}$$

Grades are elements of a monoid with unit 0 and operation +

Annotating Cost

$$\frac{}{\Gamma \vdash \uparrow^{\kappa} : \mathbb{M}\ \kappa\ \mathbf{1}}\ \textsc{Tick}$$

Unit type

The user annotates those parts of the program that should incur costs.

Honest Graded Monads

$$\frac{\Gamma \vdash e : \mathbb{M}\ 0\ \tau}{\Gamma \vdash \mathsf{run}(e) : \tau}\ \textsc{Run}$$

Any possible effect is reflected in the grade.

Overapproximation Through Subtyping

Order relation on costs

Subtyping

$$\frac{p_1 \leq p_2 \qquad \tau_1 <: \tau_2}{\mathbb{M}\ p_1\ \tau_1 <: \mathbb{M}\ p_2\ \tau_2}\ \textsc{S-Mon}$$

Costs should form an ordered monoid.

This order allows for over approximation in the analysis.

How to model amortization?

Storing Potential

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \texttt{store}\ t : \mathbb{M}\ \kappa\ ([\kappa]\tau)}\ \textsc{Store}$$

Potential type

Pay k cost now to store k potential for later use.

Releasing Potential

This value has κ_1 potential attached to it

This computation has cost $\kappa_1 + \kappa_2$

$$\frac{\Gamma \vdash t_1 : [\kappa_1]\tau_1 \qquad \Gamma, x : \tau_1 \vdash t_2 : \mathbb{M}\ (\kappa_1 + \kappa_2)\ \tau_2}{\Gamma \vdash \texttt{release}\ x = t_1\ \texttt{in}\ t_2 : \mathbb{M}\ \kappa_2\ \tau_2}\ \textsc{Release}$$

We have reduced the cost of the computation by κ_1

Summary

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{ret}\ t : \mathbb{M}\ 0\ \tau}\ \textsc{Return} \qquad \frac{\Gamma \vdash t_1 : \mathbb{M}\ p_1\ \tau_1 \qquad \Gamma, x : \tau_1 \vdash t_2 : \mathbb{M}\ p_2\ \tau_2}{\Gamma \vdash \mathsf{bind}\ x = t_1\ \mathsf{in}\ t_2 : \mathbb{M}\ (p_1 + p_2)\ \tau_2}\ \textsc{Bind}$$

$$\frac{}{\Gamma \vdash \uparrow^{\kappa} : \mathbb{M}\ \kappa\ \mathbf{1}}\ \textsc{Tick} \qquad \frac{\Gamma \vdash e : \mathbb{M}\ 0\ \tau}{\Gamma \vdash \mathsf{run}(e) : \tau}\ \textsc{Run}$$

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{store}\ t : \mathbb{M}\ \kappa\ ([\kappa]\tau)}\ \textsc{Store}$$

$$\frac{\Gamma \vdash t_1 : [\kappa_1]\tau_1 \qquad \Gamma, x : \tau_1 \vdash t_2 : \mathbb{M}\ (\kappa_1 + \kappa_2)\ \tau_2}{\Gamma \vdash \mathsf{release}\ x = t_1\ \mathsf{in}\ t_2 : \mathbb{M}\ \kappa_2\ \tau_2}\ \textsc{Release}$$

Part III

A Categorical Semantics of Cost

Existing Kripke Semantics

$$
\begin{array}{lcl}
[\![\mathbf{1}]\!] & \triangleq & \{(p, T, ())\} \\
[\![\mathrm{b}]\!] & \triangleq & \{(p, T, v) \mid v \in [\![\mathrm{b}]\!]\} \\
[\![L^0\tau]\!] & \triangleq & \{(p, T, nil)\} \\
[\![L^{s+1}\tau]\!] & \triangleq & \{(p, T, v :: l) | \exists p_1, p_2 . p_1 + p_2 \leq p \wedge (p_1, T, v) \in [\![\tau]\!] \wedge (p_2, T, l) \in [\![L^s\tau]\!]\} \\
[\![\tau_1 \otimes \tau_2]\!] & \triangleq & \{(p, T, \langle\!\langle v_1, v_2 \rangle\!\rangle) \mid \exists p_1, p_2 . p_1 + p_2 \leq p \wedge (p_1, T, v_1) \in [\![\tau_1]\!] \wedge (p_2, T, v_2) \in [\![\tau_2]\!]\} \\
[\![\tau_1 \,\&\, \tau_2]\!] & \triangleq & \{(p, T, \langle v_1, v_2 \rangle) \mid (p, T, v_1) \in [\![\tau_1]\!] \wedge (p, T, v_2) \in [\![\tau_2]\!]\} \\
[\![\tau_1 \oplus \tau_2]\!] & \triangleq & \{(p, T, \mathsf{inl}(v)) \mid (p, T, v) \in [\![\tau_1]\!]\} \cup \{(p, T, \mathsf{inr}(v)) \mid (p, T, v) \in [\![\tau_2]\!]\} \\
[\![!\tau]\!] & \triangleq & \{(p, T, !e) \mid (0, T, e) \in [\![\tau]\!]_{\mathcal{E}}\} \\
[\![\tau_1 \multimap \tau_2]\!] & \triangleq & \{(p, T, \lambda x.e) \mid \forall p', e', T' {<} T \,.(p', T', e') \in [\![\tau_1]\!]_{\mathcal{E}} \implies (p + p', T', e[e'/x]) \in [\![\tau_2]\!]_{\mathcal{E}}\} \\
[\![[n]\,\tau]\!] & \triangleq & \{(p, T, v) \mid \exists p' . p' + n \leq p \wedge (p', T, v) \in [\![\tau]\!]\}\} \\
[\![\mathbb{M}\,\kappa\,\tau]\!] & \triangleq & \{(p, T, v) \mid \forall \kappa', v', T' {<} T \,. v \Downarrow^{\kappa'}_{T'} v' \implies \exists p' . \kappa' + p' \leq p + \kappa \wedge (p', T - T', v') \in [\![\tau]\!]\} \\
[\![\forall \alpha.\tau]\!] & \triangleq & \{(p, T, \Lambda.e) \mid \forall \tau', T' {<} T \,.(p, T', e) \in [\![\tau[\tau'/\alpha]]\!]_{\mathcal{E}}\} \\
[\![\forall i.\tau]\!] & \triangleq & \{(p, T, \Lambda.e) \mid \forall I, T' {<} T \,.(p, T', e) \in [\![\tau[I/i]]\!]_{\mathcal{E}}\} \\
[\![C \Rightarrow \tau]\!] & \triangleq & \{(p, T, \Lambda.e) \mid . \models C \implies (p, T, e) \in [\![\tau]\!]_{\mathcal{E}}\} \\
[\![C\&\tau]\!] & \triangleq & \{(p, T, v) \mid . \models C \wedge (p, T, v) \in [\![\tau]\!]\} \\
[\![\exists s.\tau]\!] & \triangleq & \{(p, T, v) \mid \exists s' . (p, T, v) \in [\![\tau[s'/s]]\!]\} \\
[\![\lambda_t i.\tau]\!] & \triangleq & f \text{ where } \forall I.\, f\ I = [\![\tau[I/i]]\!] \\
[\![\tau\ I]\!] & \triangleq & [\![\tau]\!]\ I
\end{array}
$$

Operational forcing semantics for monad

Rajani et al: A unifying type-theory for higher-order (amortized) cost analysis

An interesting observation

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \mathtt{store}\ t : \mathbb{M}\ \kappa\ ([\kappa]\tau)}\ \textsc{Store}$$

$$\mathsf{pay} : [p](\mathbb{M}\ p\ \tau) \multimap \tau$$

$$\mathsf{pay} = \lambda x.\mathsf{run}(\mathtt{release}\ y = x\ \mathtt{in}\ y)$$

$$\epsilon : FG \to 1 \qquad \eta : 1 \to GF$$

Unit and counit of an adjunction

Categorical Models

Morphism in a category

Objects of a category

Typing Derivation

$$If\ \mathcal{D} : \Gamma \vdash e : \tau\ is\ derivable,\ then\ [\![\Gamma]\!] \overset{[\![\mathcal{D}]\!]}{\longrightarrow} [\![\tau]\!]$$

$$If\ \mathcal{D}_1 \equiv \mathcal{D}_2,\ then\ [\![\mathcal{D}_1]\!] = [\![\mathcal{D}_2]\!]$$

Equational Theory

Equality of Morphisms

The Cost Category $\mathbb{C}$

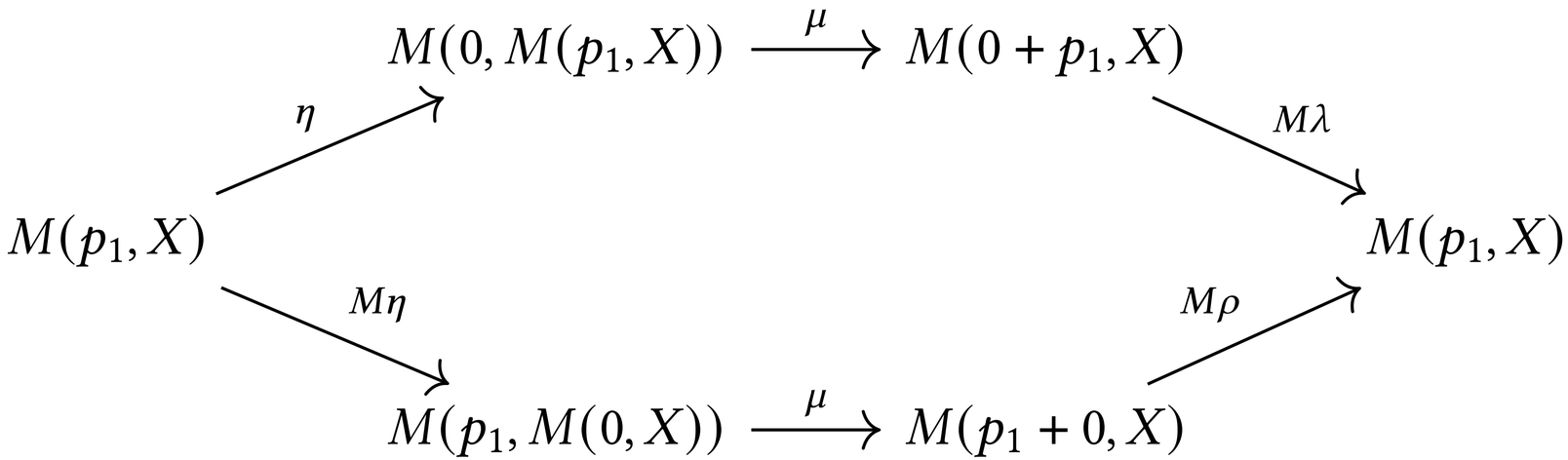
- Previously we required that costs form an ordered monoid
- We can turn this into a category $\mathbb{C}$
- Objects of $\mathbb{C}$ are costs 0,1,42,...
- Morphisms of $\mathbb{C}$ witness the order between costs
- The unit 0 and operation + form a monoidal structure on $\mathbb{C}$

The Model Category $\mathbb{X}$

- The model category $\mathbb{X}$ needs enough structure to interpret the usual linear logic connectives &, $\oplus$, $\otimes$ and $\multimap$, as well as their units.
- We need a strong graded monad $M : \mathbb{C} \times \mathbb{X} \to \mathbb{X}$
- We need a strong functor $P : \mathbb{C} \times \mathbb{X} \to \mathbb{X}$ (contravariant in first argument)
- For every grade p there is an adjunction $P(p, -) \dashv M(p, -)$

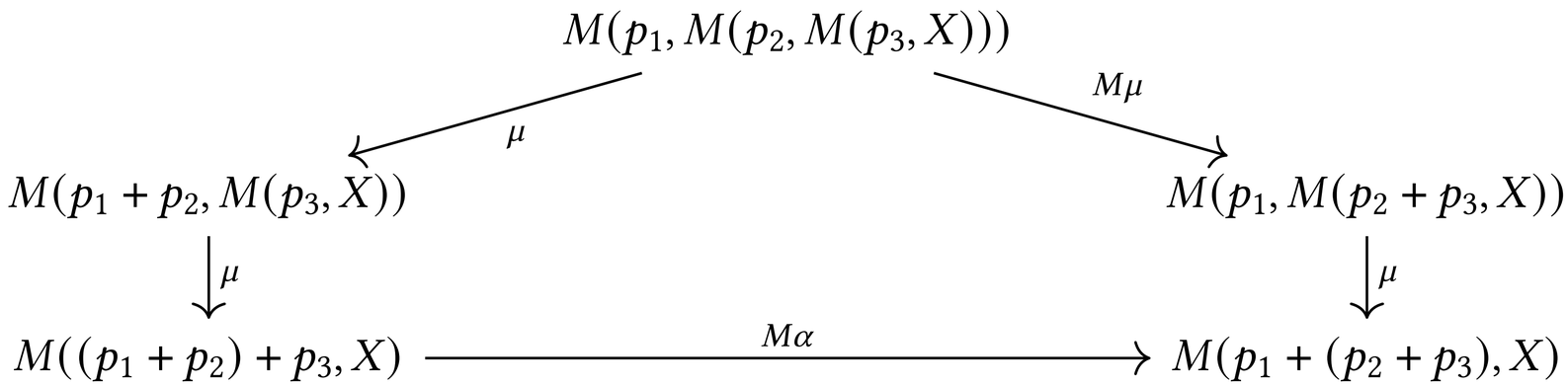
Graded Monads

- A functor $M : \mathbb{C} \times \mathbb{X} \to \mathbb{X}$
- A natural transformation $\eta_X : X \to M(0,X)$
- A natural transformation $\mu_{X,p_1,p_2} : M(p_1, M(p_2, X)) \to M(p_1 + p_2, X)$



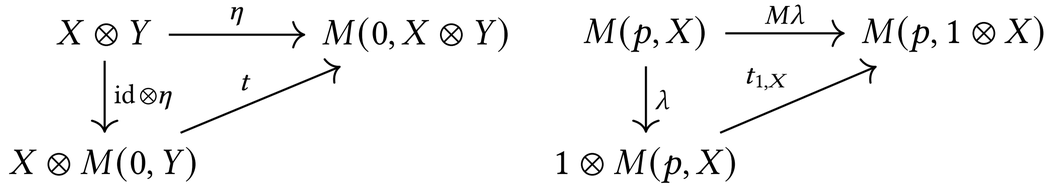
Graded Monads

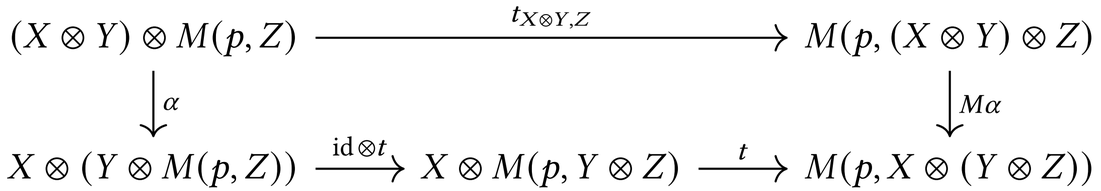
- A functor $M : \mathbb{C} \times \mathbb{X} \to \mathbb{X}$
- A natural transformation $\eta_X : X \to M(0,X)$
- A natural transformation $\mu_{X,p_1,p_2} : M(p_1, M(p_2, X)) \to M(p_1 + p_2, X)$

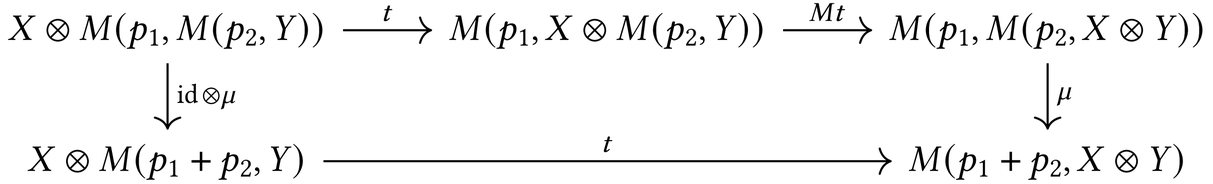


Strong Graded Monads

- Strength: $t_{X,Y} : X \otimes M(p, Y) \to M(p, X \otimes Y)$







Instances of this Model

- The degenerate set-theoretic model
- The Kripke model of Vineet et al. (Proven in Lean)
- A covariant presheaf model $\mathbb{X} = [\mathbb{C}, \mathsf{Set}]$ which interprets $\otimes$ as Day convolution

Future Work

- Modelling linear logic exponentials $!\tau$ and subexponentials $!_{\leq n}\tau$
- Modelling recursion with a fixpoint operator
- Proving completeness of the categorical model

Questions?