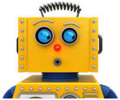

When Programs Have to Watch Paint Dry

Danel Ahman

Faculty of Mathematics and Physics (FMF)
University of Ljubljana

FoSSaCS 2023, Paris, 24.04.2023

Safe usage of resources in programming

Safe usage of resources in programming

- Let us consider **controlling a robot arm** on a production line:

```
...

let (body', left-door', right-door') =
  paint (body, left-door, right-door) in

assemble (body', left-door', right-door');


...
```

where the **resources** are the various **car parts** (body, doors, . . .)

Safe usage of resources in programming

- Let us consider **controlling a robot arm** on a production line:

```
...

let (body', left-door', right-door') =
  paint (body, left-door, right-door) in

assemble (body', left-door', right-door');


...
```

  where the **resources** are the various **car parts** (body, doors, . . .)

- Much of **existing work** has focused on **how** such res. are used
  - **linear types** can be used to **avoid discarding and duplication**
  - **session types** can be used to **enforce order of operations**
  - **runners of alg. effs.** can be used to **ensure proper finalisation**
  - . . .

Safe usage of temporal resources in prog.

- Let us consider **controlling a robot arm** on a production line:

```
...

let (body', left-door', right-door') =
  paint (body, left-door, right-door) in

assemble (body', left-door', right-door');


...
```

  where the **resources** are the various **car parts** (body, doors, . . .)

- **In this paper**, we instead focus on **when** resources are used!

Safe usage of temporal resources in prog.

- Let us consider **controlling a robot arm** on a production line:

```
...

let (body', left-door', right-door') =
  paint (body, left-door, right-door) in
                                              ← τ_dry time needs to pass
assemble (body', left-door', right-door');

...
```

- **Correctness** relies on the **parts given enough time to dry**:
  - **(a)** a **scheduler** could **dynamically block execution**, or
  - **(b)** a **compiler** could **insert enough time delay** between op. calls, or
  - **(c)** the **robot arm** could meanwhile **do other useful work**
- But **how to reason** about the result being **temporally correct**?

What's in the paper

- **Temporal resources** via **time-graded modal types**
- A **core calculus** $\lambda_{[\tau]}$ for safe programming with temp. resources
  - Fitch-style **time-graded modal types** (for temporal resources)
  - **temporally aware graded algebraic effects** (for time passage)
  - **temporally aware effect handlers** (for redefining operations)
  - with an **FGCBV-style equational presentation**
- A natural **denotational semantics** justifying the proposed design
  - **adjoint strong monoidal functors** (for modalities)
  - $[-]$**-strong time-graded monad** (for effectful computations)
  - a **presheaf example** (for concreteness and intuition)

Temporal resources via time-graded modal types

A naive solution attempt

- What if **we stay in a simply typed effectful language** and simply make paint **return the desired drying time**?

```
let (τ_dry, body', left-door', right-door') =
  paint (body, left-door, right-door) in

delay τ_dry;

assemble (body', left-door', right-door')
```

- So, **are we done?**

A **naive** solution attempt

- What if **we stay in a simply typed effectful language** and simply make paint **return the desired drying time**?

```
let (τ_dry, body', left-door', right-door') =
  paint (body, left-door, right-door) in

delay τ_dry;                                  ← τ_dry time now passes

assemble (body', left-door', right-door')
```

- So, **are we done?**
- **No!**
  - all the burden for correctness is on the programmer's shoulders
  - typechecker saying yes does not guarantee that `delay` happens, or that it happens where/when it is supposed to happen

Our solution: temporal resource types and $\lambda_{[\tau]}$

Our solution: temporal resource types and $\lambda_{[\tau]}$

- We use a **time-graded modal type** to capture temporal resources

$$X, Y, Z \quad ::= \quad \ldots \quad | \quad [\tau]\, X \qquad\qquad (\tau \in \mathbb{N})$$

- **Intuition 1:** $[\tau]\, X$ denotes that an X-typed resource **becomes usable in at most τ time units** (and remains so afterwards)

- **Intuition 2:** **at least τ time units need to pass** before a program is allowed to access the underlying X-typed resource

Our solution: temporal resource types and $\lambda_{[\tau]}$

- We use a **time-graded modal type** to capture temporal resources

$$X, Y, Z \quad ::= \quad \ldots \quad | \quad [\tau]\, X \qquad\qquad (\tau \in \mathbb{N})$$

- **Intuition 1:** $[\tau]\, X$ denotes that an X-typed resource **becomes usable in at most τ time units** (and remains so afterwards)

- **Intuition 2: at least τ time units need to pass** before a program is allowed to access the underlying X-typed resource

- This allows us to work with **resource values** such as

$$\mathsf{body}' : [\tau_{\mathsf{dry\text{-}body}}]\, \mathsf{Body} \qquad \mathsf{left\text{-}door}' : [\tau_{\mathsf{dry\text{-}door}}]\, \mathsf{Door} \qquad \ldots$$

Time-graded Fitch-style presentation

Time-graded Fitch-style presentation

- We also include **context modalities** (modelling time passage)

$$\Gamma \quad ::= \quad \cdot \;\mid\; \Gamma, x : X \;\mid\; \Gamma, \langle \tau \rangle$$

Time-graded Fitch-style presentation

- We also include **context modalities** (modelling time passage)

$$\Gamma \quad ::= \quad \cdot \ \mid \ \Gamma, x : X \ \mid \ \Gamma, \langle \tau \rangle$$

- **Introduction form** is given by **boxing up a temp. resource**

$$\frac{\Gamma, \langle \tau \rangle \vdash V : X}{\Gamma \vdash \mathsf{box}_\tau \ V : [\tau] X}$$

Time-graded Fitch-style presentation

- We also include **context modalities** (modelling time passage)

$$\Gamma \quad ::= \quad \cdot \quad | \quad \Gamma, x : X \quad | \quad \Gamma, \langle \tau \rangle$$

- **Introduction form** is given by **boxing up a temp. resource**

$$\frac{\Gamma, \langle \tau \rangle \vdash V : X}{\Gamma \vdash \mathsf{box}_\tau \ V : [\tau] X}$$

- **Elimination rule** is given by **unboxing a temporal resource**

$$\frac{\tau \leqslant \mathsf{time}\, \Gamma \qquad \lfloor \Gamma \rfloor_\tau \vdash V : [\tau] X \qquad \Gamma, x : X \vdash N : Y \,!\, \tau'}{\Gamma \vdash \mathsf{unbox}_\tau \ V \ \mathsf{as} \ x \ \mathsf{in} \ N : Y \,!\, \tau'}$$

where $\lfloor \Gamma \rfloor_\tau$ takes Γ to a τ **time units earlier state**[1], e.g., as in

$$\lfloor \Gamma, x : X, \langle 4 \rangle, y : Y, \langle 1 \rangle, z : Z \rfloor_3 \equiv \Gamma, x : X, \langle 2 \rangle$$

[1] We have $\lfloor - \rfloor_\tau \dashv \langle \tau \rangle$ for Γs with $\tau \leqslant \mathsf{time}\, \Gamma$, i.e., $\langle \tau \rangle$ is **param. r. adj.** (Gratzer et al. '22)

Equational theory and admissible typ. rules

- The **computational behaviour** of box & unbox is unsurprising

$$\Gamma \vdash \mathsf{unbox}_{\tau}\ (\mathsf{box}_{\tau}\ V)\ \mathsf{as}\ x\ \mathsf{in}\ N \equiv N[V/x] : Y \mathbin{!} \tau' \qquad (\beta)$$

$$\Gamma \vdash \mathsf{unbox}_{\tau}\ V\ \mathsf{as}\ x\ \mathsf{in}\ N[(\mathsf{box}_{\tau}\ x)/y] \equiv N[V/y] : Y \mathbin{!} \tau' \qquad (\eta)$$

with the rest of the eq. theory also fairly standard for FGCBV

Equational theory and admissible typ. rules

- The **computational behaviour** of box & unbox is unsurprising

$$\Gamma \vdash \mathsf{unbox}_\tau\ (\mathsf{box}_\tau\ V)\ \mathsf{as}\ x\ \mathsf{in}\ N \equiv N[V/x] : Y \mathrel{!} \tau' \qquad (\beta)$$

$$\Gamma \vdash \mathsf{unbox}_\tau\ V\ \mathsf{as}\ x\ \mathsf{in}\ N[(\mathsf{box}_\tau\ x)/y] \equiv N[V/y] : Y \mathrel{!} \tau' \qquad (\eta)$$

with the rest of the eq. theory also fairly standard for FGCBV

- The type system admits **standard structural rules** (wk, ...)

- It also admits **temporal rules** for **context modalities**

$$\frac{\Gamma, \langle 0 \rangle \vdash J}{\Gamma \vdash J} \qquad \frac{\Gamma, \langle \tau_1 + \tau_2 \rangle \vdash J}{\Gamma, \langle \tau_1 \rangle, \langle \tau_2 \rangle \vdash J} \qquad \frac{\Gamma, \langle \tau \rangle \vdash J \quad \tau \leqslant \tau'}{\Gamma, \langle \tau' \rangle \vdash J} \qquad \frac{\Gamma, \langle \tau \rangle, x : X \vdash J}{\Gamma, x : X, \langle \tau \rangle \vdash J}$$

i.e., $\langle - \rangle$ is contravariant **strong monoidal functor** (with co-str.)

Temporally aware graded algebraic effects

- Given by **temporal operation signatures**, such as

$$\mathsf{paint} \quad : \quad \overrightarrow{\mathsf{Part}} \quad \rightsquigarrow \quad \overrightarrow{[\tau_{\mathsf{dry}}]\,\mathsf{Part}} \quad ! \quad \tau_{\mathsf{paint}}$$

giving rise to **operation calls** with **temporal awareness**, e.g.,

$$\frac{\Gamma \vdash V : \mathsf{Body} \times \mathsf{Door} \times \mathsf{Door} \qquad \Gamma\,,\ \langle\, \tau_{\mathsf{paint}} \,\rangle\,,\ y : [\tau_{\mathsf{dry}}]\,\mathsf{Body} \times [\tau_{\mathsf{dry}}]\,\mathsf{Door} \times [\tau_{\mathsf{dry}}]\,\mathsf{Door} \vdash M : X \,!\, \tau}{\Gamma \vdash \mathsf{paint}\ V\ (y\,.\,M) : X \,!\, \tau_{\mathsf{paint}} + \tau}$$

where M can assume that τ_{paint} **additional time has passed**

Temporally aware graded algebraic effects

- Given by **temporal operation signatures**, such as

$$\mathsf{paint} \;:\; \overrightarrow{\mathsf{Part}} \;\rightsquigarrow\; \overrightarrow{[\tau_{\mathsf{dry}}]\,\mathsf{Part}} \;!\; \tau_{\mathsf{paint}}$$

  giving rise to **operation calls** with **temporal awareness**, e.g.,

$$\frac{\Gamma \vdash V : \mathsf{Body} \times \mathsf{Door} \times \mathsf{Door} \qquad \Gamma \,,\, \langle\, \tau_{\mathsf{paint}} \,\rangle \,,\; y : [\tau_{\mathsf{dry}}]\,\mathsf{Body} \times [\tau_{\mathsf{dry}}]\,\mathsf{Door} \times [\tau_{\mathsf{dry}}]\,\mathsf{Door} \vdash M : X \;!\; \tau}{\Gamma \vdash \mathsf{paint}\ V\ (y \,.\, M) : X \;!\; \tau_{\mathsf{paint}} + \tau}$$

  where M can assume that τ_{paint} **additional time has passed**

- This **temporal awareness** also happens in **seq. composition**

$$\frac{\Gamma \vdash M : X \;!\; \tau \qquad \Gamma \,,\, \langle\, \tau \,\rangle \,,\; x : X \vdash N : Y \;!\; \tau'}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : Y \;!\; \tau + \tau'}$$

Temporally aware effect handlers

- Allow us to **redefine the operations**
  - e.g., to split complex assembly tasks into smaller ones
- **Effect handlers** and **effect handling**[2]

$$\frac{\Gamma \vdash M : X \,!\, \tau \qquad \Gamma, \langle \tau \rangle, y : X \vdash N : Y \,!\, \tau' \qquad \Big(\forall \tau'' .\ \Gamma, x : A_{\mathsf{op}}, k : [\tau_{\mathsf{op}}](B_{\mathsf{op}} \to Y \,!\, \tau'') \vdash M_{\mathsf{op}} : Y \,!\, \tau_{\mathsf{op}} + \tau''\Big)_{\mathsf{op} \in \mathcal{O}}}{\Gamma \vdash \mathsf{handle}\ M\ \mathsf{with}\ (x.k.M_{\mathsf{op}})_{\mathsf{op} \in \mathcal{O}}\ \mathsf{to}\ y\ \mathsf{in}\ N : Y \,!\, \tau + \tau'}$$

  have to adhere to the **temporal discipline**
  - **op. cases M_{op} require τ_{op}-time** to pass before resuming cont. k
  - **continuation N** can still **safely assume τ-time** has passed

[2] We assume being given a set $\mathcal{O}$ of typed operation symbols $\mathsf{op} : A_{\mathsf{op}} \to B_{\mathsf{op}}$.

Back to **controlling the robot arm**

Back to controlling the robot arm

- Using the above, we can now **rewrite our example in $\lambda_{[\tau]}$** as

```
let (body', left-door', right-door') =           ← resource-typed variables
  paint (body, left-door, right-door) in

delay τ_dry;                                     ← forces τ_dry time to pass

unbox body' as body'' in   ← context:  Γ , body′ : [τ_dry] Body , … , ⟨ τ_dry ⟩
unbox left-door' as left-door'' in
unbox right-door' as right-door'' in

assemble (body'', left-door'', right-door'')  ← non-resource-typed variables
```

Back to controlling the robot arm

- Using the above, we can now **rewrite our example in $\lambda_{[\tau]}$** as

```
let (body', left-door', right-door') =              ← resource-typed variables
  paint (body, left-door, right-door) in

delay τ_dry;                                         ← forces τ_dry time to pass

unbox body' as body'' in   ← context:  Γ , body' : [τ_dry] Body , ... , ⟨ τ_dry ⟩
unbox left-door' as left-door'' in
unbox right-door' as right-door'' in

assemble (body'', left-door'', right-door'')  ← non-resource-typed variables
```

- This is **remarkably similar to the naive attempt** from earlier!
  - The only difference is **some additional calls** to `unbox`
  - But we have gained **strong static temporal guarantees**!

Back to controlling the robot arm

- **Alternatively**, **instead of blocking execution** with `delay`, we could have equally well **called other useful alg. operations**

```
let (body', left-door', right-door') =          ← resource-typed variables
  paint (body, left-door, right-door) in

! op₁ v₁; . . . opₙ vₙ;          ← as long as they collectively take ⩾ τ_dry time

unbox body' as body'' in  ← context:  Γ , body′ : [τ_dry] Body , ... , ⟨τ_dry⟩
unbox left-door' as left-door'' in
unbox right-door' as right-door'' in

assemble (body'', left-door'', right-door'')  ← non-resource-typed variables
```

A glimpse into the denotational semantics

Denotational semantics: category $\mathbb{C}$

- Want $\mathbb{C}$ to have **binary products** $(\mathbb{1}, A \times B)$
- Want $\mathbb{C}$ to have **exponentials** $A \Rightarrow B$
  - for most of the development, Kleisli exps. $A \Rightarrow T\,\tau\,B$ suffice
- **Example: presheaf category** $\mathrm{Set}^{(\mathbb{N},\leqslant)}$ (of time-varying sets)
  - gives Kripke's **possible worlds style semantics**
  - but with **all types being monotone** (resources do not expire)

given $A \in \mathrm{Set}^{(\mathbb{N},\leqslant)}$, then

$$t_1 \leqslant t_2 \qquad \text{implies} \qquad A(t_1 \leqslant t_2) : A(t_1) \longrightarrow A(t_2)$$

Denotational semantics: modal types $[\tau]\, X$

- Want there to be **strong monoidal functor**

$$[-] : (\mathbb{N}, \leqslant) \longrightarrow [\mathbb{C}, \mathbb{C}]$$

with the **strong monoidality** witnessed by the natural isos.[3]

$$\varepsilon_A : [0]\, A \xrightarrow{\cong} A \qquad \delta_{A,\tau_1,\tau_2} : [\tau_1 + \tau_2]\, A \xrightarrow{\cong} [\tau_1]\, ([\tau_2]\, A)$$

- In the **presheaf example**, we define $[-]$ as

$$([\tau]\, A)(t) \quad \stackrel{\text{def}}{=} \quad A(t + \tau)$$

[3] In Fitch-style, the S4 modality $\Box$ is interpreted by an **idempotent comonad**

Denotational semantics: context modality

- Want there to be (contravariant) **strong monoidal functor**

$$\langle - \rangle : (\mathbb{N}, \leqslant)^{\mathbf{op}} \longrightarrow [\mathbb{C}, \mathbb{C}]$$

with the **strong monoidality** witnessed by the natural isos.[4]

$$\eta_A : A \xrightarrow{\cong} \langle 0 \rangle A \qquad \mu_{A,\tau_1,\tau_2} : \langle \tau_1 \rangle (\langle \tau_2 \rangle A) \xrightarrow{\cong} \langle \tau_1 + \tau_2 \rangle A$$

- In the **presheaf example**, we define $\langle - \rangle$ as

$$(\langle \tau \rangle A)(t) \quad \stackrel{\text{def}}{=} \quad (\tau \leqslant t) \times A(t \dot{-} \tau)$$

[4] In Fitch-style, the ctx. modality for S4 is interpreted by an **idempotent monad**

Denotational semantics: mod. interaction

- Also want there to be a **family of adjunctions**[5]

$$\langle \tau \rangle \dashv [\tau]$$

witnessed by natural transformations

$$\eta^{\dashv}_{A,\tau} : A \longrightarrow [\tau]\,(\langle \tau \rangle\, A) \qquad \varepsilon^{\dashv}_{A,\tau} : \langle \tau \rangle\,([\tau]\, A) \longrightarrow A$$

  - required to interact well with the two **strong mon. structures**
  - they allow values/resources to be **pushed forward in time**

[5] In Fitch-style modal λ-calculi, one also requires an **adjunction between mods.**

Denotational semantics: mod. interaction

- Also want there to be a **family of adjunctions**[5]

$$\langle \tau \rangle \dashv [\tau]$$

witnessed by natural transformations

$$\eta^{\dashv}_{A,\tau} : A \longrightarrow [\tau]\,(\langle \tau \rangle A) \qquad \varepsilon^{\dashv}_{A,\tau} : \langle \tau \rangle ([\tau]\, A) \longrightarrow A$$

  - required to interact well with the two **strong mon. structures**
  - they allow values/resources to be **pushed forward in time**

- In the **presheaf example**,
  - $\eta^{\dashv}_{A,\tau}$ and $\varepsilon^{\dashv}_{A,\tau}$ are given by **id. on A-values**, plus by $\leqslant$-reasoning
  - $\varepsilon^{\dashv}_{A,\tau}$ is definable because of the $(\tau \leqslant t)$ **condition** in $(\langle \tau \rangle A)(t)$

[5] In Fitch-style modal λ-calculi, one also requires an **adjunction between mods.**

Denotational semantics: comp. effects

- Want there to be a **graded monad** (disc.-graded as no sub-eff.)

$$T : \mathbb{N} \longrightarrow [\mathbb{C}, \mathbb{C}]$$

with **unit** and **multiplication** (satisfying standard g. m. laws)

$$\eta^T_A : A \longrightarrow T\,0\,A \qquad \mu^T_{A,\tau_1,\tau_2} : T\,\tau_1\,(T\,\tau_2\,A) \longrightarrow T\,(\tau_1 + \tau_2)\,A$$

and with a $[-]$-**strength**[6] (satisfying variants of std. str. laws)

$$\mathrm{str}^T_{A,B,\tau} : [\tau]\,A \times T\,\tau\,B \longrightarrow T\,\tau\,(A \times B)$$

[6] Terminology follows the parlance of Bierman and de Paiva ($\Diamond$ was $\Box$-strong)

Denotational semantics: comp. effects

- Want there to be a **graded monad** (disc.-graded as no sub-eff.)

$$T : \mathbb{N} \longrightarrow [\mathbb{C}, \mathbb{C}]$$

with **unit** and **multiplication** (satisfying standard g. m. laws)

$$\eta^T_A : A \longrightarrow T\,0\,A \qquad \mu^T_{A,\tau_1,\tau_2} : T\,\tau_1\,(T\,\tau_2\,A) \longrightarrow T\,(\tau_1 + \tau_2)\,A$$

and with a **$[-]$-strength**[6] (satisfying variants of std. str. laws)

$$\mathsf{str}^T_{A,B,\tau} : [\tau]\,A \times T\,\tau\,B \longrightarrow T\,\tau\,(A \times B)$$

- $\mathsf{str}^T_{A,B,\tau}$ is the same as **$[-]$-variant of enrichment of T**, i.e.,

$$[\tau]\,(A \Rightarrow B) \longrightarrow (T\,\tau\,A \Rightarrow T\,\tau\,B)$$

[6] Terminology follows the parlance of Bierman and de Paiva ($\Diamond$ was $\Box$-strong)

Denotational semantics: comp. effects

- Want there to be a **graded monad** (disc.-graded as no sub-eff.)

$$T : \mathbb{N} \longrightarrow [\mathbb{C}, \mathbb{C}]$$

  with **unit** and **multiplication** (satisfying standard g. m. laws)

$$\eta^T_A : A \longrightarrow T\,0\,A \qquad \mu^T_{A,\tau_1,\tau_2} : T\,\tau_1\,(T\,\tau_2\,A) \longrightarrow T\,(\tau_1 + \tau_2)\,A$$

  and with a **$[-]$-strength**[6] (satisfying variants of std. str. laws)

$$\mathrm{str}^T_{A,B,\tau} : [\tau]\,A \times T\,\tau\,B \longrightarrow T\,\tau\,(A \times B)$$

- $\mathrm{str}^T_{A,B,\tau}$ is the same as **$[-]$-variant of enrichment of T**, i.e.,

$$[\tau]\,(A \Rightarrow B) \longrightarrow (T\,\tau\,A \Rightarrow T\,\tau\,B)$$

- We also require T to have **alg. ops.** and support **eff. handling**

[6] Terminology follows the parlance of Bierman and de Paiva ($\Diamond$ was $\Box$-strong)

Denotational semantics: comp. effects

- In the **presheaf example**, the **graded monad**[7] is given by cases

$$\frac{a \in A(t)}{\mathsf{ret}\, a \in (T\, 0\, A)(t)}$$

$$\frac{a \in [\![A_{\mathsf{op}}]\!](t) \qquad k \in \big([\tau_{\mathsf{op}}]\, ([\![B_{\mathsf{op}}]\!] \Rightarrow T\, \tau\, A)\big)(t)}{\mathsf{op}\, a\, k \in (T\, (\tau_{\mathsf{op}} + \tau)\, A)(t)}$$

$$\frac{k \in [\tau]\, (T\, \tau'\, A)(t)}{\mathsf{delay}\, \tau\, k \in (T\, (\tau + \tau')\, A)(t)}$$

with the graded-monadic structure given by unsurprising recursion

[7] This T is for the setting where there are **no delay-equations** in the calculus

Denotational semantics: comp. effects

- In the **presheaf example**, the **graded monad**[7] is given by cases

$$\frac{a \in A(t)}{\mathrm{ret}\, a \in (T\, 0\, A)(t)}$$

$$\frac{a \in [\![A_{\mathrm{op}}]\!](t) \qquad k \in \big([\tau_{\mathrm{op}}]\, ([\![B_{\mathrm{op}}]\!] \Rightarrow T\, \tau\, A)\big)(t)}{\mathrm{op}\, a\, k \in (T\, (\tau_{\mathrm{op}} + \tau)\, A)(t)}$$

$$\frac{k \in [\tau]\, (T\, \tau'\, A)(t)}{\mathrm{delay}\, \tau\, k \in (T\, (\tau + \tau')\, A)(t)}$$

  with the graded-monadic structure given by unsurprising recursion

- Direct def. in the **Agda** formalisation uses **induction-recursion**
  - IR needed so that k is natural for continuations in effect handling

[7] This T is for the setting where there are **no delay-equations** in the calculus

Let’s wrap it up

Conclusion

- **Temporal resources** can be naturally captured using
  - **modal temporal resource type** $[\tau]\,X$
  - with a **time-graded Fitch-style presentation**
    - using a **temporal context modality** $\Gamma, \langle \tau \rangle$
    - a time-graded instance of **param. r. adjs. (Gratzer et al. '22)**
  - with a **temporally aware type-and-effect system**
  - with a **natural category-th. semantics** (based on $\langle \tau \rangle \dashv [\tau]$)
- The paper is also accompanied by an **Agda formalisation**

  https://github.com/danelahman/temporal-resources

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0024.

Some ongoing/future work directions

- **Operational semantics**
  - modelling delay and alg. effs. as actually progressing time
- **Sub-effecting**
  - as sub-effecting M = all-possible-ways-to-insert-delays-into-M?
- **(Primitive) recursion**
  - grade of $\quad \mathsf{rec}\ V\ M_z\ x.k.M_s \quad$ computed by iteration/recursion
  - M_z and M_s being temporally aware depending on iteration count
- **Generalising gradings**
  - other $(\mathbb{N}, 0, +, \dot{-}, \leqslant)$-like structures, e.g., (sets of) traces or states
  - different structures, e.g., as $\quad \Gamma, \langle\, \tau(\mathsf{trace}) \,\rangle, x : X \vdash N : Y \ !\ \mathsf{trace}'$
- **Expiring resources**
  - where resources are usable only for an interval, e.g., as $[\,\tau, \tau'\,]\, X$