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:

```markdown
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:

```plaintext
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
  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)
  - **$[\tau]$-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?

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

delay τdry;

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?

```plaintext
let (τ\text{dry}, \text{body}', \text{left-door}', \text{right-door}') =
paint (\text{body}, \text{left-door}, \text{right-door}) \text{ in }

delay τ\text{dry};

assemble (\text{body}', \text{left-door}', \text{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

  $$X, Y, Z ::= \ldots | [\tau] X$$

  ($\tau \in \mathbb{N}$)

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

- **Intuition 2**: at least $\tau$ 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

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

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

- **Intuition 2:** **at least $\tau$ 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

\[
\text{body}': [\tau_{\text{dry-body}}] \text{Body} \quad \text{left-door}': [\tau_{\text{dry-door}}] \text{Door} \quad \ldots
\]
Introduction form is given by boxing up a temporal resource $\Gamma, \tau \vdash V : \Sigma$:

$$\Gamma \vdash \tau V : \Sigma$$

Elimination rule is given by unboxing a temporal resource $\tau \vdash \Gamma$:

$$\Gamma \vdash \tau \vdash \tau V : \Sigma$$

where $\tau \vdash \Gamma$ takes $\Gamma$ to a $\tau$ time units earlier state, e.g., as in $\Gamma, x : X, x : X, y : Y, y : Y, x : Z \vdash 3$.

We have $\tau \vdash \tau \vdash \tau V : \Sigma$ for $\Gamma$s with $\tau \vdash \Gamma$, i.e., $\tau V : \Sigma$ is param. r. adj. (Gratzer et al. '22).
Time-graded Fitch-style presentation

- We also include context modalities (modelling time passage)
  \[ \Gamma ::= \cdot \mid \Gamma, x:X \mid \Gamma, \langle \tau \rangle \]
We also include **context modalities** (modelling time passage)

\[ \Gamma ::= \cdot \mid \Gamma, x : X \mid \Gamma, \langle \tau \rangle \]

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

\[
\begin{align*}
\Gamma, \langle \tau \rangle &\vdash V : X \\
\Gamma &\vdash \text{box}_\tau V : [\tau]X
\end{align*}
\]
Time-graded Fitch-style presentation

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

  \[ \Gamma ::= \cdot \mid \Gamma, x : X \mid \Gamma, \langle \tau \rangle \]

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

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

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

  \[
  \frac{\tau \leq \text{time}\Gamma \quad |\Gamma|_\tau \vdash V : [\tau]X \quad \Gamma, x : X \vdash N : Y ! \tau'}{\Gamma \vdash \text{unbox}_\tau V \text{ as } x \text{ in } N : Y ! \tau'}
  \]

  where \(|\Gamma|_\tau\) takes \(\Gamma\) to a \(\tau\) time units earlier state\(^1\), e.g., as in

  \[ |\Gamma, x : X, \langle 4 \rangle, y : Y, \langle 1 \rangle, z : Z |_3 \equiv \Gamma, x : X, \langle 2 \rangle \]

---

\(^1\) We have \(|\cdot|_\tau \rightarrow \langle \tau \rangle\) for \(\Gamma\)s with \(\tau \leq \text{time}\Gamma\), 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

\[ \Gamma \vdash \text{unbox}_\tau (\text{box}_\tau V) \text{ as } x \text{ in } N \equiv N[V/x] : Y!\tau' \]  \hspace{1cm} (\beta)

\[ \Gamma \vdash \text{unbox}_\tau V \text{ as } x \text{ in } N[(\text{box}_\tau x)/y] \equiv N[V/y] : Y!\tau' \]  \hspace{1cm} (\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

\[ \Gamma \vdash \text{unbox}_\tau (\text{box}_\tau V) \text{ as } x \text{ in } N \equiv N[V/x] : Y ! \tau' \]  
\[ \Gamma \vdash \text{unbox}_\tau V \text{ as } x \text{ in } N[(\text{box}_\tau x)/y] \equiv N[V/y] : Y ! \tau' \]  
with the rest of the eq. theory also fairly standard for FGCBV

- The type system admits standard structural rules \((wk, \ldots)\)

- It also admits temporal rules for context modalities

\[ \frac{\Gamma, \langle 0 \rangle \vdash J}{\Gamma \vdash J} \quad \frac{\Gamma, \langle \tau_1 + \tau_2 \rangle \vdash J}{\Gamma, \langle \tau_1 \rangle, \langle \tau_2 \rangle \vdash J} \quad \frac{\Gamma, \langle \tau \rangle \vdash J \quad \tau \leq \tau'}{\Gamma, \langle \tau' \rangle \vdash J} \quad \frac{\Gamma, \langle \tau \rangle, x : X \vdash J}{\Gamma, x : X, \langle \tau \rangle \vdash J} \]

i.e., \(\langle - \rangle\) is contravariant strong monoidal functor (with co-str.)
Temporally aware graded algebraic effects

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

\[
\text{paint} : \overset{\rightarrow}{\text{Part}} \sim \overset{[\tau_{\text{dry}}]}{\text{Part}} ! \tau_{\text{paint}}
\]

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

\[
\Gamma \vdash V : \text{Body} \times \text{Door} \times \text{Door}
\]

\[
\Gamma, \langle \tau_{\text{paint}} \rangle, y : [\tau_{\text{dry}}] \text{Body} \times [\tau_{\text{dry}}] \text{Door} \times [\tau_{\text{dry}}] \text{Door} \vdash M : X ! \tau
\]

\[
\Gamma \vdash \text{paint } V (y . M) : X ! \tau_{\text{paint}} + \tau
\]

where \( M \) can assume that \( \tau_{\text{paint}} \) **additional time has passed**
Temporally aware graded algebraic effects

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

\[
paint : \text{Part} \rightsquigarrow \tau_{\text{dry}} \text{Part} ! \tau_{\text{paint}}
\]

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

\[
\Gamma \vdash V : \text{Body} \times \text{Door} \times \text{Door}
\]

\[
\Gamma, \langle \tau_{\text{paint}} \rangle, \ y : \tau_{\text{dry}} \text{Body} \times \tau_{\text{dry}} \text{Door} \times \tau_{\text{dry}} \text{Door} \vdash M : X ! \tau
\]

\[
\Gamma \vdash \text{paint} \ V \ (y \ M) : X ! \tau_{\text{paint}} + \tau
\]

where \( M \) can assume that \( \tau_{\text{paint}} \) **additional time has passed**

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

\[
\Gamma \vdash M : X ! \tau \quad \Gamma, \langle \tau \rangle, \ x : X \vdash N : Y ! \tau'
\]

\[
\Gamma \vdash \text{let} \ x = M \text{ in } N : Y ! \tau + \tau'
\]
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\(^2\)

\[
\begin{align*}
\Gamma & \vdash M : X \rightarrow \tau \\
\Gamma, \langle \tau \rangle, y : X & \vdash N : Y \rightarrow \tau' \\
(\forall \tau''. \Gamma, x : A_{\text{op}}, k : [\tau_{\text{op}}](B_{\text{op}} \rightarrow Y \rightarrow \tau'') & \vdash M_{\text{op}} : Y \rightarrow \tau_{\text{op}} + \tau'')_{\text{op} \in \mathcal{O}} \\
\Gamma & \vdash \text{handle } M \text{ with } (x.k.M_{\text{op}})_{\text{op} \in \mathcal{O}} \text{ to } y \text{ in } N : Y \rightarrow \tau + \tau'
\end{align*}
\]

have to adhere to the temporal discipline

- op. cases \(M_{\text{op}}\) require \(\tau_{\text{op}}\)-time to pass before resuming cont. \(k\)

- continuation \(N\) can still safely assume \(\tau\)-time has passed

\(^2\)We assume being given a set \(\mathcal{O}\) of typed operation symbols \(\text{op} : A_{\text{op}} \rightarrow B_{\text{op}}\).
Back to **controlling the robot arm**

Using the above, we can now rewrite our example in $\lambda$ as

```plaintext
let (body', left-door', right-door') = \resource-typed variables
  paint (body, left-door, right-door)
  in delay \dry; \forces \dry time to pass
  unbox body' as body'' in \context: Γ, body_1: \dry τ, ...,
  x\dry y
  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

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

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

delay $\tau_{\text{dry}}$;  \hspace{1cm} \text{← resource-typed variables}

unbox body' as body'' in \hspace{1cm} \text{← context: $\Gamma$, body':$[\tau_{\text{dry}}]$Body , ..., $\langle \tau_{\text{dry}} \rangle$}
unbox left-door' as left-door'' in
unbox right-door' as right-door'' in

assemble (body'', left-door'', right-door'') \hspace{1cm} \text{← non-resource-typed variables}
```
Back to controlling the robot arm

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

\[
\text{let } (\text{body}', \text{left-door}', \text{right-door}') = \hspace{1cm} \leftarrow \text{resource-typed variables}
\]
\[\text{paint } (\text{body}, \text{left-door}, \text{right-door}) \text{ in}
\]
\[\text{delay } \tau_{\text{dry}}; \hspace{1cm} \leftarrow \text{forces } \tau_{\text{dry}} \text{ time to pass}
\]
\[\text{unbox } \text{body}' \text{ as } \text{body}'' \text{ in } \hspace{1cm} \leftarrow \text{context: } \Gamma, \text{body}' : [\tau_{\text{dry}}] \text{Body}, \ldots, \langle \tau_{\text{dry}} \rangle
\]
\[\text{unbox } \text{left-door}' \text{ as } \text{left-door}'' \text{ in}
\]
\[\text{unbox } \text{right-door}' \text{ as } \text{right-door}'' \text{ in}
\]
\[\text{assemble } (\text{body}'', \text{left-door}'', \text{right-door}'') \hspace{1cm} \leftarrow \text{non-resource-typed variables}
\]

• This is remarkably similar to the naive attempt from earlier!

• The only difference is some additional calls to \texttt{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.

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

! op_1 \nu_1; \ldots \ op_n \nu_n;  

unbox body' as body'' in  
context: \Gamma, body':[\tau_{dry}]Body, ... , \langle \tau_{dry} \rangle
unbox left-door' as left-door'' in
unbox right-door' as right-door'' in

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

← resource-typed variables

← as long as they collectively take \( \geq \tau_{dry} \) time

← non-resource-typed variables
A glimpse into the denotational semantics
Denotational semantics: category $\mathbb{C}$

- Want $\mathbb{C}$ to have **binary products** $(1, A \times B)$
- 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 $\text{Set}^{(\mathbb{N}, \leq)}$ (of time-varying sets)
  - gives Kripke’s **possible worlds style semantics**
  - but with **all types being monotone** (resources do not expire)

  given $A \in \text{Set}^{(\mathbb{N}, \leq)}$, then

  $t_1 \leq t_2$ implies $A(t_1 \leq t_2) : A(t_1) \rightarrow A(t_2)$
Denotational semantics: modal types $[\tau] X$

- Want there to be strong monoidal functor

  \[ [-] : (\mathbb{N}, \leq) \longrightarrow [\mathcal{C}, \mathcal{C}] \]

  with the strong monoidality witnessed by the natural isos.\(^3\)

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

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

\[ ([\tau] A)(t) \overset{\text{def}}{=} A(t + \tau) \]

\(^3\)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**

\[ \langle - \rangle : (\mathbb{N}, \leq)^{\text{op}} \longrightarrow [\mathbb{C}, \mathbb{C}] \]

with the **strong monoidality** witnessed by the natural isos.\(^4\)

\[ \eta_A : A \xrightarrow{\simeq} \langle 0 \rangle A \quad \mu_{A,\tau_1,\tau_2} : \langle \tau_1 \rangle (\langle \tau_2 \rangle A) \xrightarrow{\simeq} \langle \tau_1 + \tau_2 \rangle A \]

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

\[ (\langle \tau \rangle A)(t) \overset{\text{def}}{=} (\tau \leq t) \times A(t - \tau) \]

---

\(^4\)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

\[ \langle \tau \rangle \vdash [\tau] \]

witnessed by natural transformations

\[ \eta_{A,\tau} : A \rightarrow [\tau] (\langle \tau \rangle A) \quad \varepsilon_{A,\tau} : \langle \tau \rangle ([\tau] A) \rightarrow A \]

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

\[ ^5 \text{In Fitch-style modal } \lambda\text{-calculi, one also requires an adjunction between mods.} \]
Denotational semantics: mod. interaction

- Also want there to be a family of adjunctions\(^5\)

\[ \langle \tau \rangle \rightarrow [\tau] \]

witnessed by natural transformations

\[ \eta_{A,\tau}^{-1} : A \rightarrow [\tau] (\langle \tau \rangle A) \quad \varepsilon_{A,\tau}^{-1} : \langle \tau \rangle ([\tau] A) \rightarrow A \]

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

- In the presheaf example,
  - \( \eta_{A,\tau}^{-1} \) and \( \varepsilon_{A,\tau}^{-1} \) are given by id. on \( A \)-values, plus by \( \leq \)-reasoning
  - \( \varepsilon_{A,\tau}^{-1} \) is definable because of the \( (\tau \leq t) \) condition in \( (\langle \tau \rangle A)(t) \)

\(^5\) In Fitch-style modal \( \lambda \)-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.)
  \[ T : \mathbb{N} \rightarrow [\mathbb{C}, \mathbb{C}] \]

  with **unit** and **multiplication** (satisfying standard g. m. laws)
  \[ \eta^T_A : A \rightarrow T 0 A \quad \mu^T_{A,\tau_1,\tau_2} : T\tau_1 (T\tau_2 A) \rightarrow T (\tau_1 + \tau_2) A \]

  and with a **[-]-strength**\(^6\) (satisfying variants of std. str. laws)
  \[ \text{str}^T_{A,B,\tau} : [\tau] A \times T\tau B \rightarrow T\tau (A \times B) \]

---

\(^6\)Terminology follows the parlance of Bierman and de Paiva (◊ was □-strong)
Denotational semantics: comp. effects

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

\[ T : \mathbb{N} \rightarrow [\mathbb{C}, \mathbb{C}] \]

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

\[ \eta^T_A : A \rightarrow T \circ A \quad \mu^T_{A, \tau_1, \tau_2} : T \tau_1 (T \tau_2 A) \rightarrow T (\tau_1 + \tau_2) A \]

and with a \([-\cdot]-strength^6\) (satisfying variants of std. str. laws)

\[ \text{str}^T_{A, B, \tau} : [\tau] A \times T \tau B \rightarrow T \tau (A \times B) \]

- \(\text{str}^T_{A, B, \tau}\) is the same as \([-\cdot]-variant of enrichment of \(T\), i.e.,

\[ [\tau] (A \Rightarrow B) \rightarrow (T \tau A \Rightarrow T \tau B) \]

---

^6 Terminology follows the parlance of Bierman and de Paiva (\(\diamond\) was \(\square\)-strong)
Denotational semantics: comp. effects

- Want there to be a **graded monad** (disc.-graded as no sub-eff.)
  \[ T : \mathbb{N} \rightarrow [\mathbb{C}, \mathbb{C}] \]

  with **unit** and **multiplication** (satisfying standard g. m. laws)
  \[
  \eta^T_A : A \rightarrow T 0 A \quad \mu^T_{A,\tau_1,\tau_2} : T \tau_1 (T \tau_2 A) \rightarrow T (\tau_1 + \tau_2) A
  \]

  and with a \([-\text{-strength}]^6\) (satisfying variants of std. str. laws)
  \[
  \text{str}^T_{A,B,\tau} : [\tau] A \times T \tau B \rightarrow T \tau (A \times B)
  \]

- \(\text{str}^T_{A,B,\tau}\) is the same as \([-\text{-variant of enrichment of } T, \text{ i.e., }\]
  \[
  [\tau] (A \Rightarrow B) \rightarrow (T \tau A \Rightarrow T \tau B)
  \]

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

---

\(^6\)Terminology follows the parlance of Bierman and de Paiva (◇ was □-strong)
Denotational semantics: **comp. effects**

- In the **presheaf example**, the **graded monad**\(^7\) is given by cases

\[
\frac{a \in A(t)}{\text{ret } a \in (T \circ A)(t)}
\]

\[
\frac{a \in [A_{op}](t) \quad k \in ([\tau_{op}]([B_{op}] \Rightarrow T \tau A))(t)}{\text{op } a \, k \in (T \tau_{op} + \tau) \circ A(t)}
\]

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

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

\[
\frac{a \in A(t)}{\text{ret } a \in (T \ 0 \ A)(t)}
\]

\[
\frac{\begin{array}{c} a \in \llbracket A_{\text{op}} \rrbracket(t) \\ k \in (\llbracket \tau_{\text{op}} \rrbracket (\llbracket B_{\text{op}} \rrbracket \Rightarrow T \ 	au \ A))(t) \end{array}}{\text{op } a \ k \in (T \ (\tau_{\text{op}} + \tau) \ A)(t)}
\]

\[
\frac{k \in \llbracket \tau \rrbracket (T \ \tau' \ A)(t)}{\text{delay } \tau \ k \in (T \ (\tau + \tau') \ A)(t)}
\]

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 \(\langle \tau \rangle \vdash [\tau]\))

- 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 $M = \text{all-possible-ways-to-insert-delays-into-}M$?

- **(Primitive) recursion**
  - grade of $\text{rec } V M_z x.k.M_s$ computed by iteration/recursion
  - $M_z$ and $M_s$ being temporally aware depending on iteration count

- **Generalising gradings**
  - other $(\mathbb{N}, 0, +, \div, \leq)$-like structures, e.g., (sets of) traces or states
  - different structures, e.g., as $\Gamma, \langle \tau(\text{trace}) \rangle, x: X \vdash N : Y \! \triangleright \! \text{trace'}$

- **Expanding resources**
  - where resources are usable only for an interval, e.g., as $[\tau, \tau'] X$