DisLog: A Separation Logic for Disentanglement

Alexandre Moine[1] Sam Westrick[2] Stephanie Balzer[2]

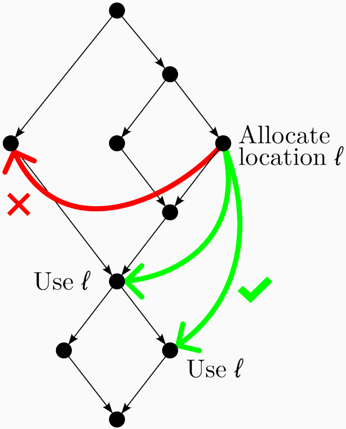
Cambium Seminar - 27/11/2023

[1] Inria

[2] Carnegie Mellon University

Disentanglement: A Property of Parallel Programs

Concurrently-executing tasks remain oblivious to each other's allocations



Disentanglement broadly occurs:

- pure programs are disentangled.
- race-free programs are disentangled.

Disentanglement guarantees locality.

The MaPLe Compiler

https://github.com/MPLLang/mpl

- StandardML with `par : (unit -> 'a) * (unit -> 'b) -> ('a * 'b)`
- Fast memory management based on disentanglement.

MPL detects and manages entanglement at runtime.

- Detection: Westrick, Arora, and Acar (ICFP'22)
- Management: Arora, Westrick, and Acar (PLDI'23)

Contribution: The First Static Analysis for Disentanglement

DisLog: A separation logic for disentanglement, with timestamps and clocks.

DisLog+: A standard 2010-era separation logic for race-free programs.

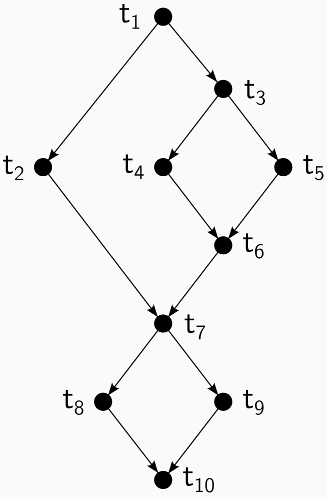
- No invariants.
- Disentanglement proof for free.
- Extensions for benign races

  write-write races and read-write races on previously allocated data.

DisLog 🤝 DisLog+

Theory and examples are fully mechanized in Coq on top of Iris.

Formal Semantics of Disentanglement



- Tasks are identified by timestamps.
- If $\ell \mapsto \ell'$ and a task t dereferences ℓ then ℓ' must have been allocated by a task preceding t.

DisLog First Steps: Timestamps and Clocks

A new weakest precondition $\mathsf{wp}\,\langle t,\ e\rangle\ \{\lambda t'\, v.\ \Psi\}$

- t is the current timestamp
- t' is the end-timestamp

The clock assertion $\ell \mathbin{\text{🕒}} t$

" ℓ was allocated by a task preceding t "

✓

Persistent

DisLog's LOAD rule

$$\frac{\ell \mapsto \ell' \qquad \ell' \mathbin{\text{🕒}} t}{\mathsf{wp}\,\langle t,\ !\ell\rangle\ \{\lambda t'\, v.\ \ulcorner t' = t \land v = \ell' \urcorner \ast\ \ell \mapsto \ell'\}}$$

Winding Clocks with the Precedence Assertion

$$\ell \,\text{🕒}\, t \ \ast\ ??? \ \twoheadrightarrow\ \ell \,\text{🕒}\, t'$$

The precedence assertion $\quad t \preccurlyeq t'$

" t precedes t' "

✓ Persistent

The clock assertion is monotonic w.r.t. the precedence pre-order.

$$\ell \,\text{🕒}\, t \ \ast\ t \preccurlyeq t' \ \twoheadrightarrow\ \ell \,\text{🕒}\, t'$$

Time Flies

Precedence and clock assertions can be generated on the fly.

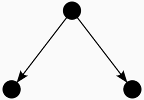
$$\frac{\mathsf{wp}\,\langle t,\ e\rangle\,\{\lambda t'\,v.\ t \preccurlyeq t' \mathrel{-\!\!*} \Psi\,t'\,v\}}{\mathsf{wp}\,\langle t,\ e\rangle\,\{\Psi\}} \qquad\qquad \frac{\ulcorner \ell \in \mathit{locs}(e) \urcorner \qquad \ell \mathrel{🕒} t \mathrel{-\!\!*} \mathsf{wp}\,\langle t,\ e\rangle\,\{\Psi\}}{\mathsf{wp}\,\langle t,\ e\rangle\,\{\Psi\}}$$

The Par rule:

$$\frac{\forall t_1\ t_2. \qquad t \preccurlyeq t_1 \mathrel{-\!\!*} \mathsf{wp}\,\langle t_1,\ e_1\rangle\,\{\Psi_1\} \qquad t \preccurlyeq t_2 \mathrel{-\!\!*} \mathsf{wp}\,\langle t_2,\ e_2\rangle\,\{\Psi_2\}}{\mathsf{wp}\,\langle t,\ e_1 \mathbin{||} e_2\rangle \left\{\lambda t'\,\ell.\ \begin{array}{l} \exists t_1'\ v_1\ t_2'\ v_2.\ \Psi_1\,t_1'\,v_1 \ * \ \Psi_2\,t_2'\,v_2 \\ t_1' \preccurlyeq t' \ * \ t_2' \preccurlyeq t' \ * \ \ell \mapsto (v_1, v_2) \end{array}\right\}}$$

Simple Programs Should Have Simple Proofs: DisLog+

- Entanglement results from a race.
- To reason about a race in separation logic, one needs an invariant.



- A proof in a 2010-era separation logic, without invariants, yields disentanglement.
- DisLog+ is such a separation logic, but encoded on top of DisLog.

Two benefits:

- Foundational proof of disentanglement for race-free programs.
- The user can switch to DisLog in a DisLog+ proof, and vice-versa.

Monotonicity to the Rescue

DisLog+ assertions are monotonic DisLog predicates over an ambient timestamp.

The monotonicity trick appeared in prior work on weak-memory models:

iGPS [Kaiser et al., 2017]   iRC11 [Dang et al., 2020]   Cosmo [Mével et al., 2020]

Key idea: the points-to assertion of DisLog+ guarantees disentanglement

$$
\begin{aligned}
\mathrm{DisLog+} \quad &\triangleq \quad \mathsf{Timestamp} \xrightarrow{\mathit{mono}} \mathrm{DisLog} \\
\ell \mapsto v \quad &\triangleq \quad \lambda t.\ \ell \mapsto v \ * \ v \mathbin{\text{🕒}} t
\end{aligned}
$$

The points-to assertion of DisLog+ cannot occur inside an invariant!

Conversions Between DisLog+ and DisLog

The weakest precondition of DisLog+

$$\mathsf{wpm}\, e\, \{Q\} \;\triangleq\; \lambda t.\ \forall t'.\ t \preccurlyeq t' \mathrel{-\!\!*} \mathsf{wp}\,\langle t',\, e\rangle\,\{\lambda t''\, v.\ (Q\, v) t''\}$$

We can convert between DisLog and DisLog+.

$$(\, P \vdash_{\mathrm{DisLog+}} \mathsf{wpm}\, e\, \{Q\}\,) \iff (\forall t.\ \ P\, t \vdash_{\mathrm{DisLog}} \mathsf{wp}\,\langle t,\, e\rangle\,\{\lambda t'\, v.\ (Q\, v)\, t'\}\,)$$

- We can verify a DisLog+ interface using DisLog.
- During a DisLog proof, we can use a DisLog+ specification.

DisLog+ is a Standard Separation Logic

Load does not require a clock assertion: it is bundled inside the points-to.

$$\frac{\ell \mapsto v}{\mathsf{wpm}\,(!\ell)\,\{\lambda v'.\, \ulcorner v' = v \urcorner \ast \ell \mapsto v\}}$$

Thanks to monotonicity, the Par rule is standard!

$$\frac{\mathsf{wpm}\; e_1\,\{Q_1\} \qquad \mathsf{wpm}\; e_2\,\{Q_2\}}{\mathsf{wpm}\,(e_1 \,||\, e_2)\,\{\lambda \ell.\, \exists v_1\ v_2.\, \ell \mapsto (v_1, v_2) \ast Q_1\, v_1 \ast Q_2\, v_2\}}$$

Read the Paper for Details

Extensions to DisLog+ for benign races:

- Write-only points-to assertions for write-write races.
- Objectivity lemmas for reasoning on races on previously allocated data.

The soundness proof and the mechanization.

Case studies:

- Spin-lock.
- Parallel lookup in a lazy collection.
- A fast lock-free hash-set for previously allocated data.
- A slow lock-free hash-set for arbitrary data.

Conclusion & Future Work

We present:

- **DisLog**, the first program logic for disentanglement.
- **DisLog+**, a high-level logic for race-free programs.

`https://gitlab.inria.fr/amoine/dislog`

Future work:

- A type system in between DisLog and DisLog+, proved sound with semantic typing.
- Arora et al. [2024] add futures to disentanglement. How to adapt DisLog?

Cambium's Special

- DisLog+ extensions and case studies.
- A bit of semantics.
- The soundness theorem.

Write-Only Points-to Assertions

Trivially disentangled: write-write races

We introduce assertions to reason about write-write races within DisLog+.

The write-only points-to $\quad \ell \Mapsto^{\delta}_{p} X$ $\qquad (\, p \in (0;1],\, X \in \wp(\mathcal{V})\,)$

- " ℓ perhaps stores a value of X ".
- If $p = 1$ and $X \neq \emptyset$, then ℓ stores a value of X.

The orig assertion $\quad \mathrm{orig}^{\delta}\ v$ $\qquad (v \in \mathcal{V})$

- " The location originally stored v ".

Write-Only Points-to API

$$\textsc{WOBegin}$$
$$\ell \mapsto v \;\Rrightarrow\; \exists \delta.\, \mathsf{orig}^{\delta}\, v \;*\; \ell \Mapsto^{\delta}_{1} \emptyset$$

$$\textsc{WOFrac}$$
$$\ell \Mapsto^{\delta}_{(p_1+p_2)} (X_1 \cup X_2) \;\dashv\vdash\; \ell \Mapsto^{\delta}_{p_1} X_1 \;*\; \ell \Mapsto^{\delta}_{p_2} X_2$$

$$\textsc{WOStore}$$
$$\frac{\ell \Mapsto^{\delta}_{p} X}{\mathsf{wpm}\, (\ell := v)\, \{\lambda\_.\, \ell \Mapsto^{\delta}_{p} \{v\}\}}$$

$$\textsc{WOCancel}$$
$$\mathsf{orig}^{\delta}\, v \;*\; \ell \Mapsto^{\delta}_{1} \emptyset \;\Rrightarrow\; \ell \mapsto v$$

$$\textsc{WOEnd}$$
$$\frac{X \neq \emptyset}{\ell \Mapsto^{\delta}_{1} X \;\Rrightarrow\; \exists v.\, \ulcorner v \in X \urcorner \;*\; \ell \mapsto v}$$

Case Study: Parallel Lookup in a Lazy Collection

- Problem: find an element inside a lazy collection.
- Solution: in parallel, search for the element, and write it inside a shared location.
- **Entanglement hazard**: the shared location must not be read!

```
let lookup (p:'a -> bool) (k:int -> 'a) (n:int) : 'a option =
  let r = ref None in
  let f i =
    let x = k i in
    if p x then r := (Some x) else () in
  parfor 0 n f; !r
```

Objectivity Lemmas — The Easy Part

Trivially disentangled: read-write races on non-location values

Supported out-of-the-box by DisLog+.

If v is not a location:

$$\begin{aligned} & \ell \mapsto v \\ \dashv\vdash\quad & \lambda t.\, \ell \mapsto v \;*\; v \text{ 🕑 } t \\ \dashv\vdash\quad & \lambda \_.\, \ell \mapsto v \end{aligned}$$

- $\ell \mapsto v$ is objective: it does not depend on the ambient timestamp.
- We can install invariants for objective assertions!
- Typical example: a spin-lock.

Objectivity Lemmas — Races on Previously Allocated Data

Trivially disentangled: read-write races on previously allocated data

Idea: unveil "just enough" DisLog in DisLog+.

- The witness $\uparrow t \triangleq \lambda t'.\, t \preccurlyeq t'$
- The embedding $\lceil \Phi \rceil \triangleq \lambda\_.\, \Phi$

The SplitSubjectiveObjective rule of Cosmo [Mével et al., 2020].

$$P \dashv\vdash \exists t.\ \uparrow t * \lceil P\ t \rceil$$

$$\begin{aligned} \ell \mapsto v &\dashv\vdash \exists t.\ \uparrow t * \lceil (\ell \mapsto v)\ t \rceil \\ &\dashv\vdash \exists t.\ \uparrow t * \lceil \ell \mapsto v * v \text{ 🕒 } t \rceil \end{aligned}$$

Clocks in DisLog+

Clocks can appear in DisLog+ $\quad \ell \,\text{◷}\, \mathit{now} \triangleq \lambda t.\, \ell \,\text{◷}\, t$

$$\frac{\ell \in \mathit{locs}(e) \qquad \ell \,\text{◷}\, \mathit{now} \mathrel{-\!\!*} \mathsf{wpm}\; e\,\{Q\}}{\mathsf{wpm}\; e\,\{Q\}}$$

The general recipe for read-write race on previously allocated data:

- Generate clocks of the values that will be involved.
- Use the SplitSubjectiveObjective rule on the points-to and clocks.
- Install an invariant storing everything.

Case Study: Deduplication with The World's Simplest Lock-Free Hash Set

- Problem: remove duplicates from an array.
- Solution: in parallel, insert elements in a hash-set (without duplicates).
  Then retrieve the elements.

The hash-set is inspired by the 3rd problem of VerifyThis [2022].

- Implemented as an array, uses open addressing and linear probing for collision.
- Insertion proceeds by a CAS loop.
- **Entanglement hazard**: the CAS must not see concurrently-allocated data.

We restrict the insertion to previously allocated data.

Some Semantics

At runtime, the semantics maintains a task tree $T \triangleq t \mid T \otimes T$.

Fork

$$\frac{t_1 \text{ and } t_2 \text{ fresh}}{t \ / \ e_1 \,||\, e_2 \ / \ \sigma \rightarrow t_1 \otimes t_2 \ / \ e_1 \,||\, e_2 \ / \ \sigma}$$

Join

$$\frac{t \text{ and } \ell \text{ fresh} \qquad \sigma' = [\ell := (v_1, v2)]\sigma}{t_1 \otimes t_2 \ / \ v_1 \,||\, v_2 \ / \ \sigma \rightarrow t \ / \ \ell \ / \ \sigma'}$$

ParL

$$\frac{T_1 \ / \ e_1 \ / \ \sigma \rightarrow T_1' \ / \ e_1' \ / \ \sigma}{T_1 \otimes T_2 \ / \ e_1 \,||\, e_2 \ / \ \sigma \rightarrow T_1' \otimes T_2 \ / \ e_1' \,||\, e_2 \ / \ \sigma'}$$

ParR

$$\frac{T_2 \ / \ e_2 \ / \ \sigma \rightarrow T_2' \ / \ e_2' \ / \ \sigma}{T_1 \otimes T_2 \ / \ e_1 \,||\, e_2 \ / \ \sigma \rightarrow T_1 \otimes T_2' \ / \ e_1 \,||\, e_2' \ / \ \sigma'}$$

The Truth About WP

- DisLog's WP is defined in terms of a more general WP wpg.
- wpg is parameterized by a task-tree.
- Generalization needed for various induction to succeed.
- At the end, we define wp as a specialization of wpg on leaves.

  ⇝ reasoning is only needed happens at leaves!

The Soundness Theorem

- We encode disentanglement to a safety condition.
- The semantics detects and prevents entanglement.

An expression is stuck if one of its tasks cannot reduce.

Soundness Theorem

If $\forall t.\ \mathsf{wpg}\,\langle t,\ e\rangle\,\{\lambda\_\,\_.\ \ulcorner \mathit{True} \urcorner\}$ holds, then e cannot reach a stuck configuration.

Corollaries:

- If $\forall t.\ \mathsf{wp}\,\langle t,\ e\rangle\,\{\lambda\_\,\_.\ \ulcorner \mathit{True} \urcorner\}$ holds, then e cannot reach a stuck configuration.
- If $\mathsf{wpm}\, e\,\{\lambda\_.\ \ulcorner \mathit{True} \urcorner\}$ holds, then e cannot reach a stuck configuration.

Thank you for your attention!

alexandre.moine [at] inria.fr
swestric [at] cs.cmu.edu
balzers [at] cs.cmu.edu

References i

Jatin Arora, Sam Westrick, and Umut A. Acar. Efficient parallel functional programming with effects. Proc. ACM Program. Lang., 7(PLDI), jun 2023. doi: 10.1145/3591284. URL https://doi.org/10.1145/3591284.

Jatin Arora, Stephen K. Muller, and Umut A. Acar. Disentanglement with futures, state, and interaction. Proc. ACM Program. Lang., 8(POPL), 2024. URL https://www.cs.cmu.edu/afs/cs.cmu.edu/user/jatina/www/public_html/_site/assets/docs/POPL24.pdf.

Hoang-Hai Dang, Jacques-Henri Jourdan, Jan-Oliver Kaiser, and Derek Dreyer. RustBelt meets relaxed memory. Proceedings of the ACM on Programming Languages, 4 (POPL):34:1–34:29, 2020. URL https://hal.inria.fr/hal-02351793/.

References ii

Jan-Oliver Kaiser, Hoang-Hai Dang, Derek Dreyer, Ori Lahav, and Viktor Vafeiadis. Strong logic for weak memory: Reasoning about release-acquire consistency in Iris. In European Conference on Object-Oriented Programming (ECOOP), pages 17:1–17:29, June 2017. URL `https://people.mpi-sws.org/~dreyer/papers/iris-weak/paper.pdf`.

Glen Mével, Jacques-Henri Jourdan, and François Pottier. Cosmo: A concurrent separation logic for Multicore OCaml. Proceedings of the ACM on Programming Languages, 4(ICFP), June 2020. URL `http://cambium.inria.fr/~fpottier/publis/mevel-jourdan-pottier-cosmo-2020.pdf`.

References iii

VerifyThis. Challenge 3 - the world's simplest lock-free hash set, 2022. URL https://ethz.ch/content/dam/ethz/special-interest/infk/chair-program-method/pm/documents/Verify%20This/Challenges2022/verifyThis2022-challenge3.pdf.

Sam Westrick, Jatin Arora, and Umut A. Acar. Entanglement detection with near-zero cost. Proc. ACM Program. Lang., 6(ICFP), aug 2022. doi: 10.1145/3547646. URL https://doi.org/10.1145/3547646.