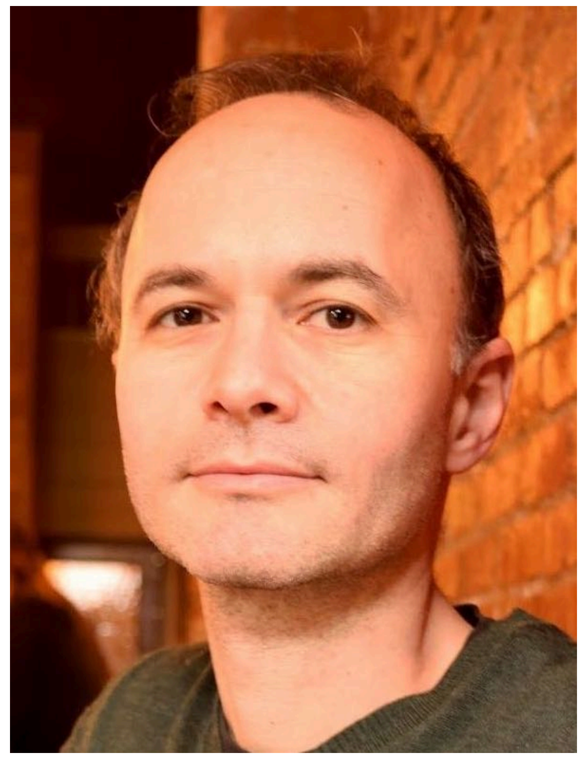
Entanglement Detection With Near-Zero Cost

Sam Westrick
Carnegie Mellon University

ICFP 2022
Ljubljana, Slovenia

joint work with:

Jatin Arora



Umut Acar

can parallel functional programming be efficient and scalable ?

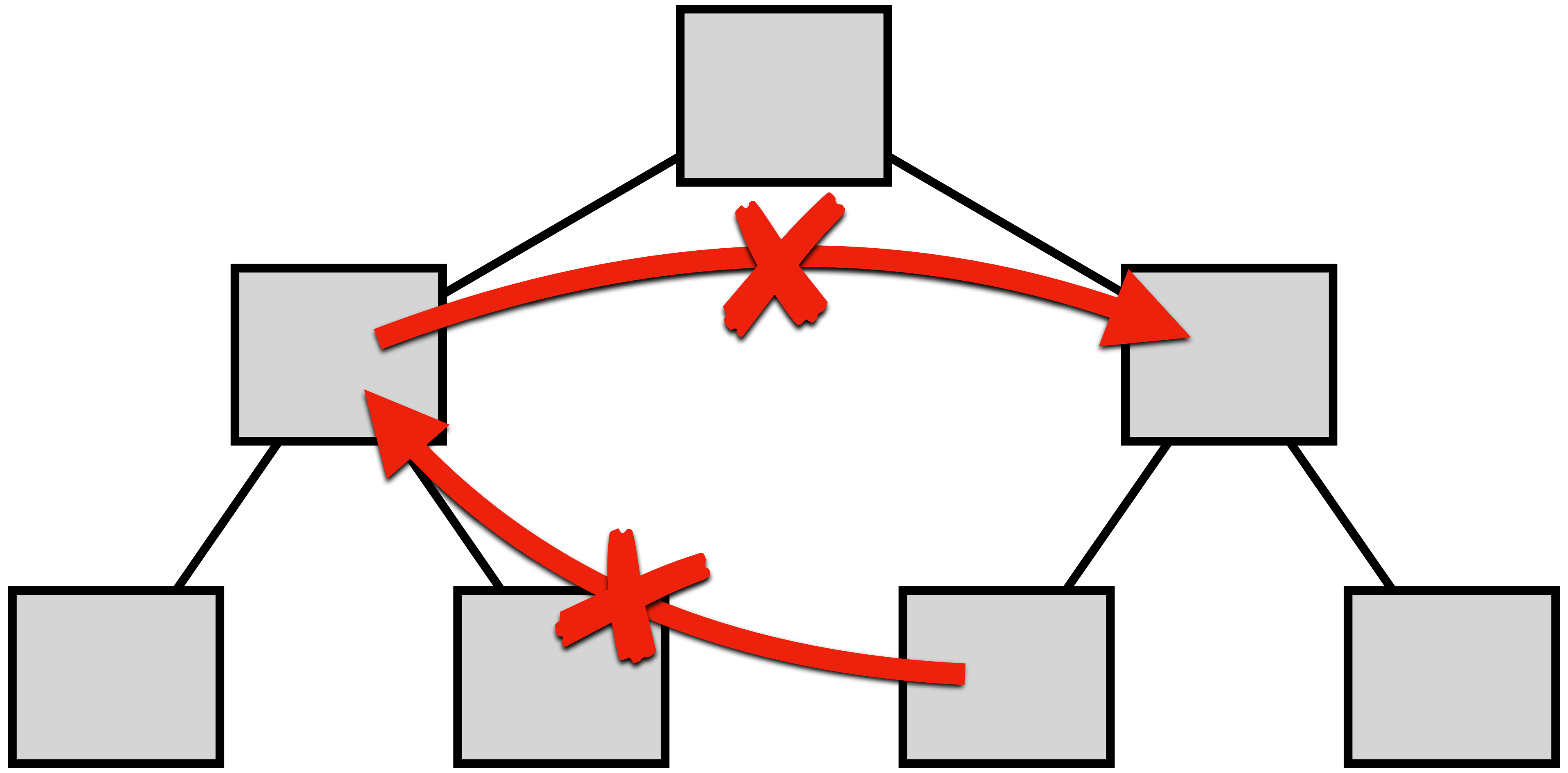
challenges

- high rate of allocation
- heavy reliance on GC

YES with parallel memory management based on **disentanglement**

Disentanglement from 10000’

- informal defn: **“concurrent tasks remain oblivious to each other’s allocations”**
- **broadly applicable**: occurs naturally in deterministic (e.g. functional!) programs
- enables efficient and scalable automatic memory management
  - **no cross-pointers**



MaPLe Compiler

github.com/mpllang/mpl

- based on MLton, **full Standard ML language**, extended with

```
val par: (unit -> ‘a) * (unit -> ‘b) -> ‘a * ‘b
```

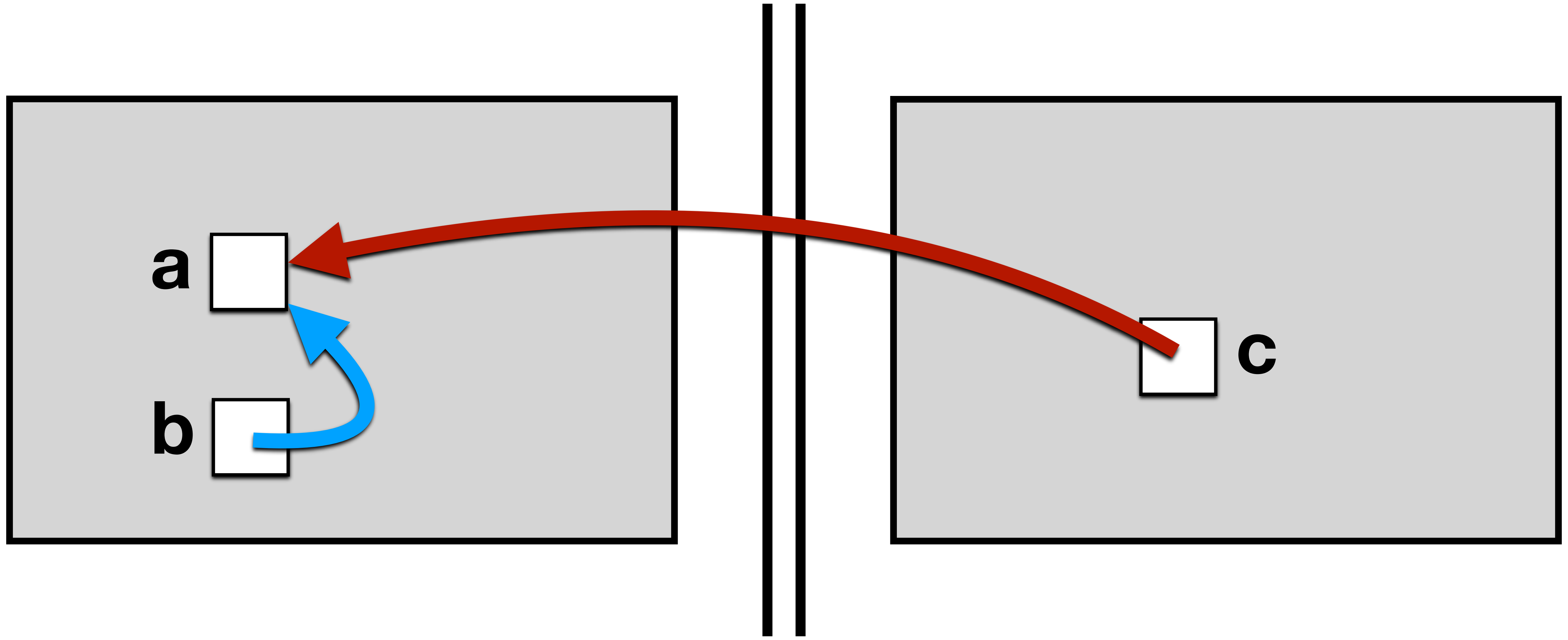
- used by 500+ students at CMU each year
- parallel memory management based on **disentanglement**
- in practice: fast, scalable, and low space usage
- competitive performance vs low-level parallel C/C++ code

MPL vs Java:
~3x faster, ~4x less space
MPL vs Go:
~2x faster, ~30% less space
MPL vs multicore OCaml:
~2x faster, ~2x less space

(averages on 72 processors)

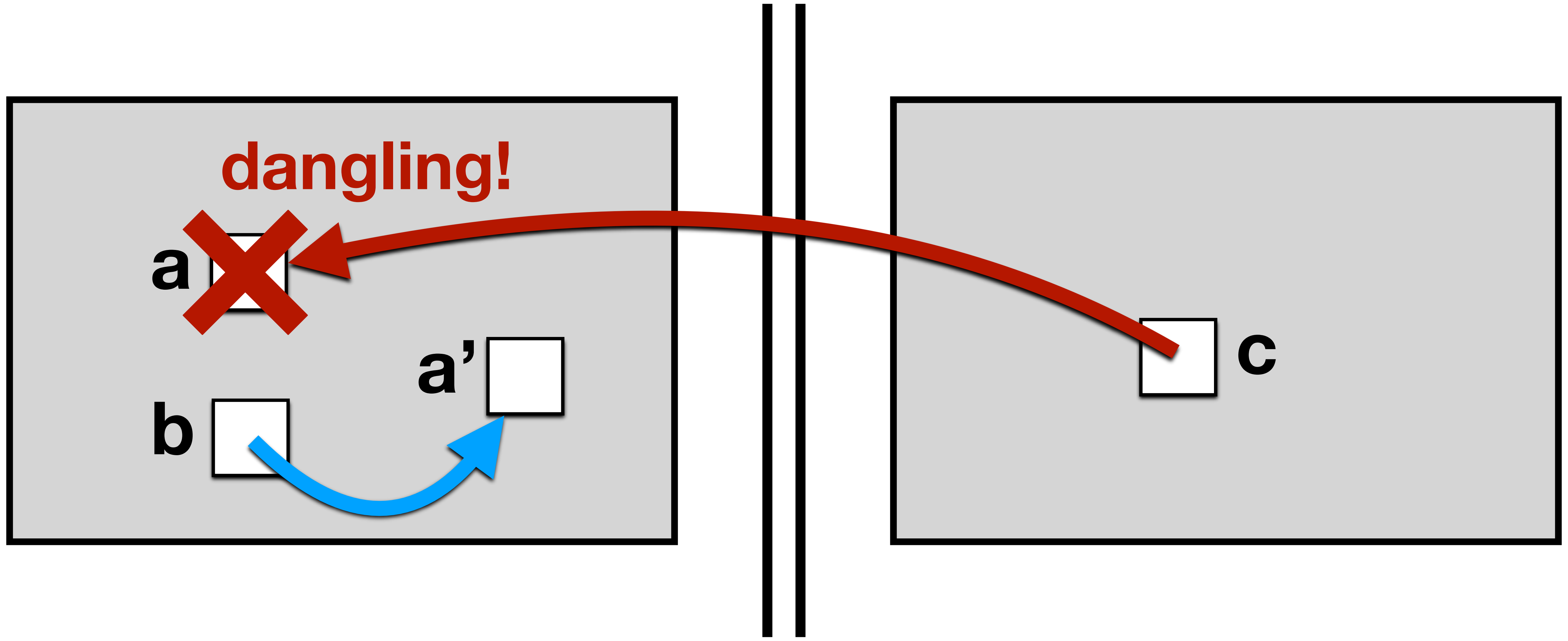
The Problem

- not all programs are disentangled
- if GC assumes disentanglement, **entangled** programs might crash (or worse)



The Problem

- not all programs are disentangled
- if GC assumes disentanglement, **entangled** programs might crash (or worse)



disentanglement needs to be enforced

Enforce Disentanglement Statically?

- disallow in-place updates? **inefficient**
- type+effect system?
    - enforce determinism? **too conservative**
    - enforce disentanglement directly? **tricky!**

Challenge Cases:
algorithms with “a little bit” of non-determinism

Our Approach: Entanglement Detection

- enforce disentanglement dynamically
  - monitor memory reads and writes
  - if entanglement detected, terminate with error message
- like race detection, except **almost zero overhead** in practice (average: ~1% for both time and space. max ~10%)

sound (“no missed alarms”)

safe for disentanglement

complete (“no false alarms”)

permits all disentangled programs

Details

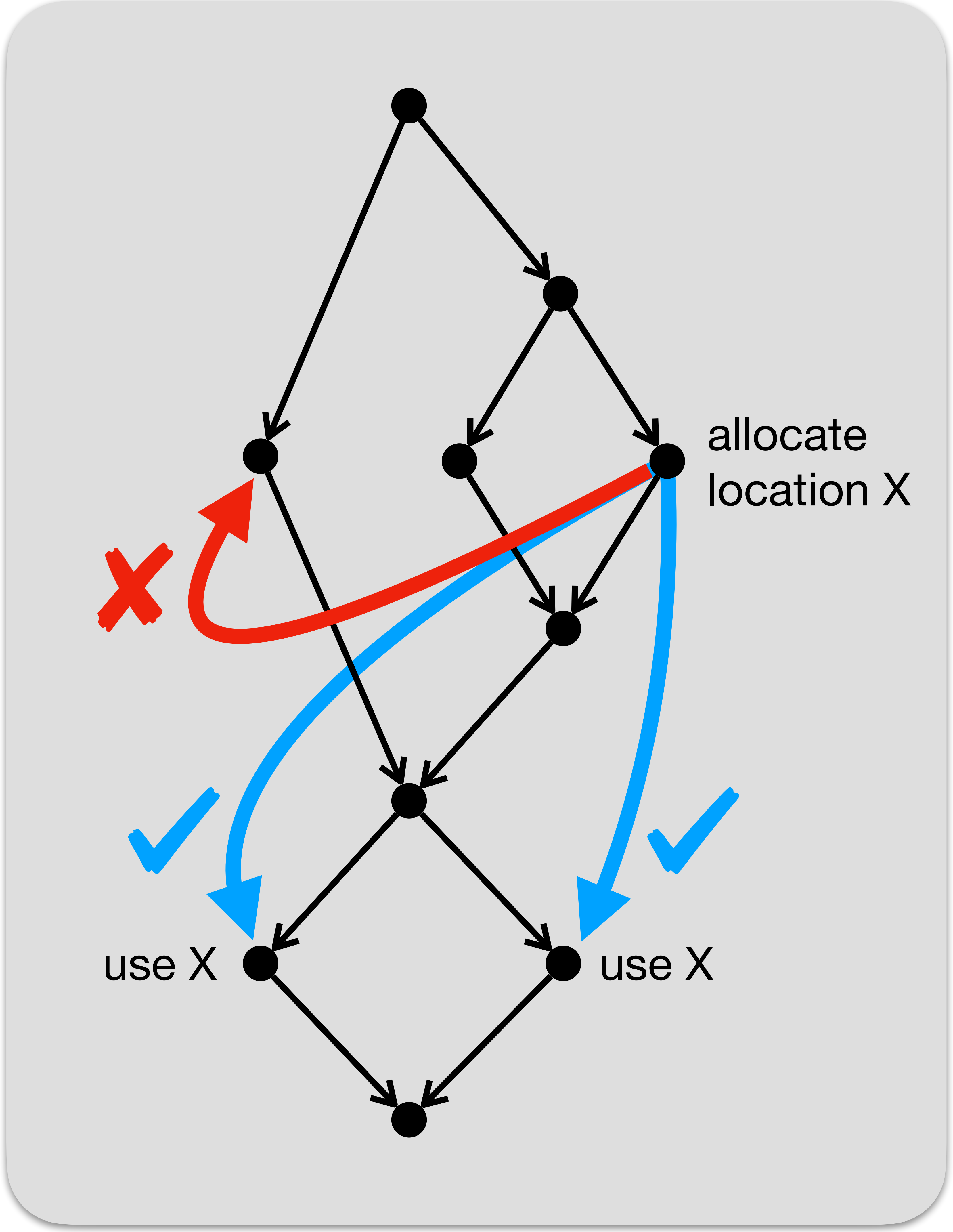
disentanglement = **allocation precedes use**

Algorithm

- build **computation graph** during execution
- annotate allocated locations with current vertex
- check **results of memory reads**
  - disentangled: result allocated before current vertex ✓
  - otherwise, **entanglement detected** ✗

Implementation Notes:

- SP-order maintenance
- read-barrier on mutable pointers only (with a **very effective fast-path**)
- closely integrated with memory management



Summary

disentanglement

- common and natural property
- important for efficient automatic memory management
- **can be checked dynamically with nearly zero overhead (this paper)**

MaPLe implementation

- fast, scalable, and space-efficient!
- competitive with low-level imperative code

Future / Ongoing work

- dynamic “entanglement management”

`github.com/mpllang/mpl`

Come see my
ML Workshop keynote!
(Thursday, 9:00am)