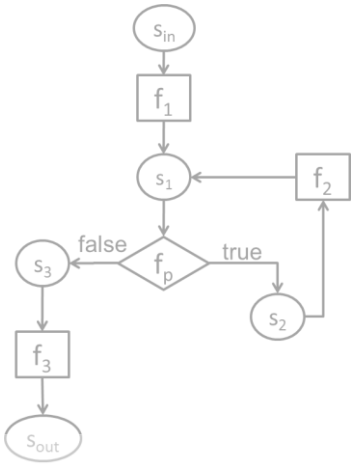
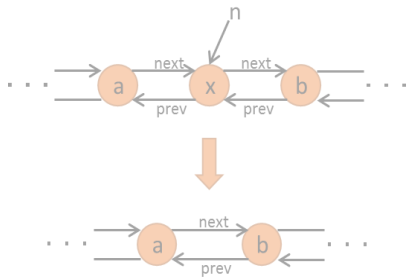
$$\exists c \forall in\ Q(c, in)$$

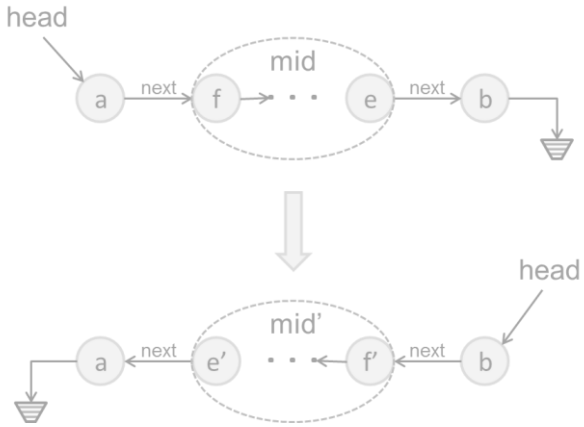
```
/* Average of x and y without using x+y (avoid overflow)*/
int avg(int x, int y){
  int t = expr({x/2, y/2, x%2, y%2, 2 }, {PLUS, DIV});
  assert t == (x+y)/2;
  return t;
}
```

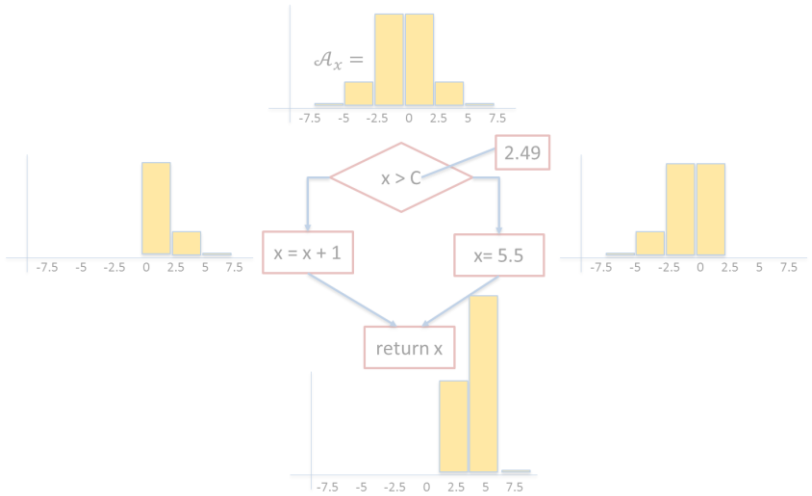





```
{
  s  = n.succ;
  p = n.pred;
  p.succ = s;
  s.pred = p;
}
```

Module III: Applications of Synthesis





$$\varphi(p)$$

$$Sk[c](in)$$

Logistics

Project presentations

- Monday Dec 10, 3-6pm; room TBD
- 20 min per team (15 min presentation + questions)
- Structure: motivation, **demo**, technique, evaluation

Project reports

- Due on Dec 14 (start working on them now!)
- Format: see course organization page (3-5 pages, SIGPLAN format)

Applications of synthesis

- Superoptimization
- Custom data structures
- Data extraction and data wrangling
- Cryptographic implementations
- SQL queries

Scaling up superoptimization

[Phothimilthana et al. ASPLOS'16]

What is the problem the paper is trying to solve?

- Why is it important?
- What are existing approaches to this problem?

- Other approaches to optimization: manual or local compiler optimization
- Other superoptimization tools: STOKE

Contributions

Lens: bidirectional enumerative search with selective refinement

- More scalable than traditional enumerative search

Context-aware window decomposition

- Trades off optimization quality and scalability

Cooperative superoptimizer

- combines strengths of enumerative, symbolic, and stochastic search
- what are they?

Limitations

Straight-line code only

- Why? How would you extend Lens to handle jumps?

Bit-width reduction is hacky

- Not all correct reduced bit-width programs correspond to correct full bit-width programs

Imprecise cost model

Pruning techniques in Lens

Bidirectional search

- roughly combination of bottom-up and top-down (with example propagation)
- no technique we've seen so far has combined them

Reduced bit width

- What are the two reasons they do it?
- Sketch also finitizes its variables to small bit-widths
- Motivated by the Small Scope hypothesis

Pruning techniques in Lens

Incremental observational equivalence / selective refinement

- what does the graph representation remind you of?
- EUSolver has an optimization where it stores the equivalence classes and doesn't have to restart on new examples; is this the same?

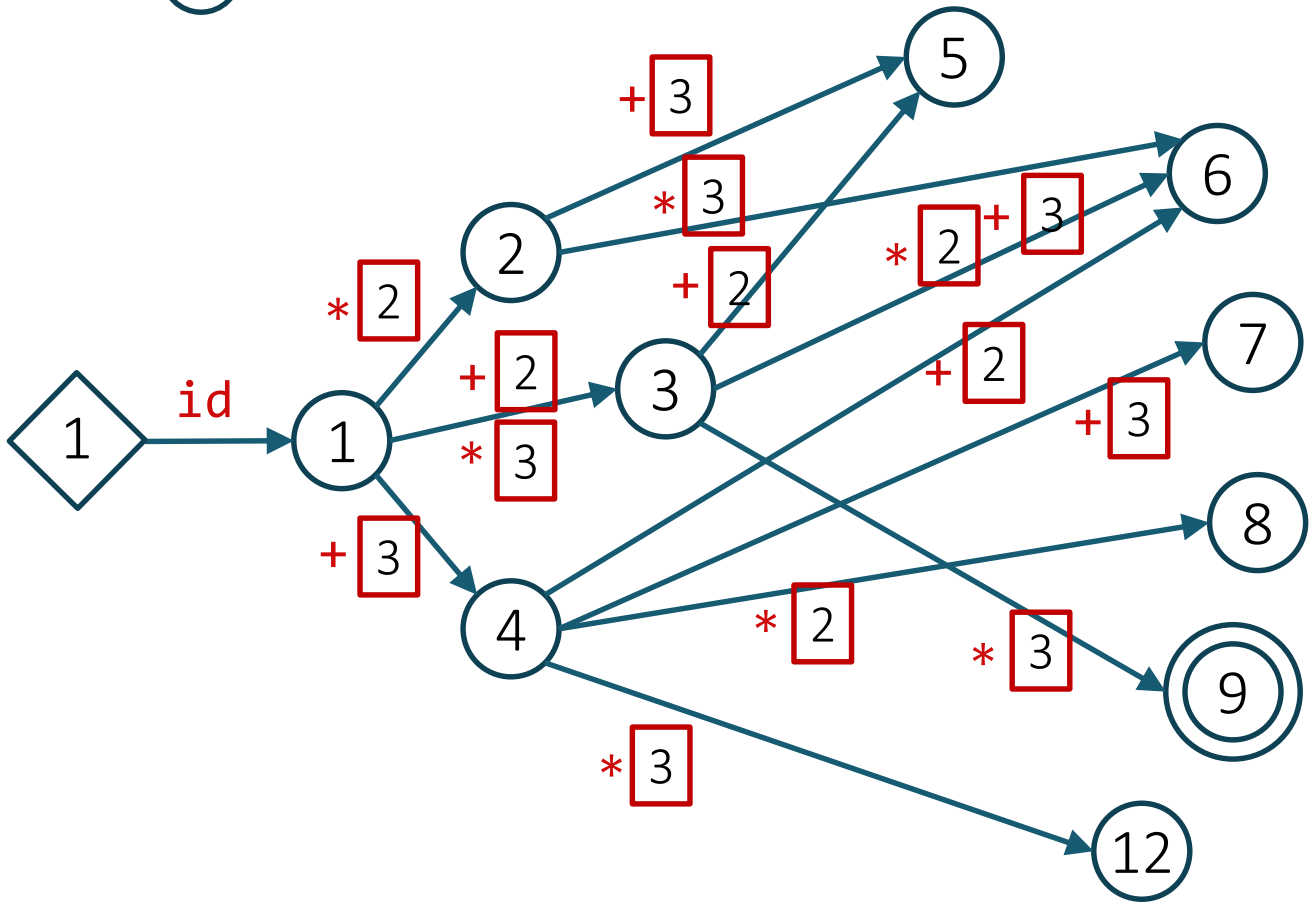
- graph structure similar to FlashFill's VSA and to Finite Tree Automata
- instead of eagerly building VSAs for all examples and merging, Lens builds for one example and "merges" incrementally
- FlashFill can represent all programs in a finite structure; here they have to restrict length
- EUSolver cannot determine if a program prefix cannot reach the goal (no length limitation); so it cannot use eq classes to reject programs

Finite Tree Automata

[Wang, Dillig, Singh '17]

```
N ::= id(V) | N + T | N * T
T ::= 2 | 3
V ::= x
```

1 → 9



Applications of synthesis

Superoptimization

→ Custom data structures

Data extraction and data wrangling

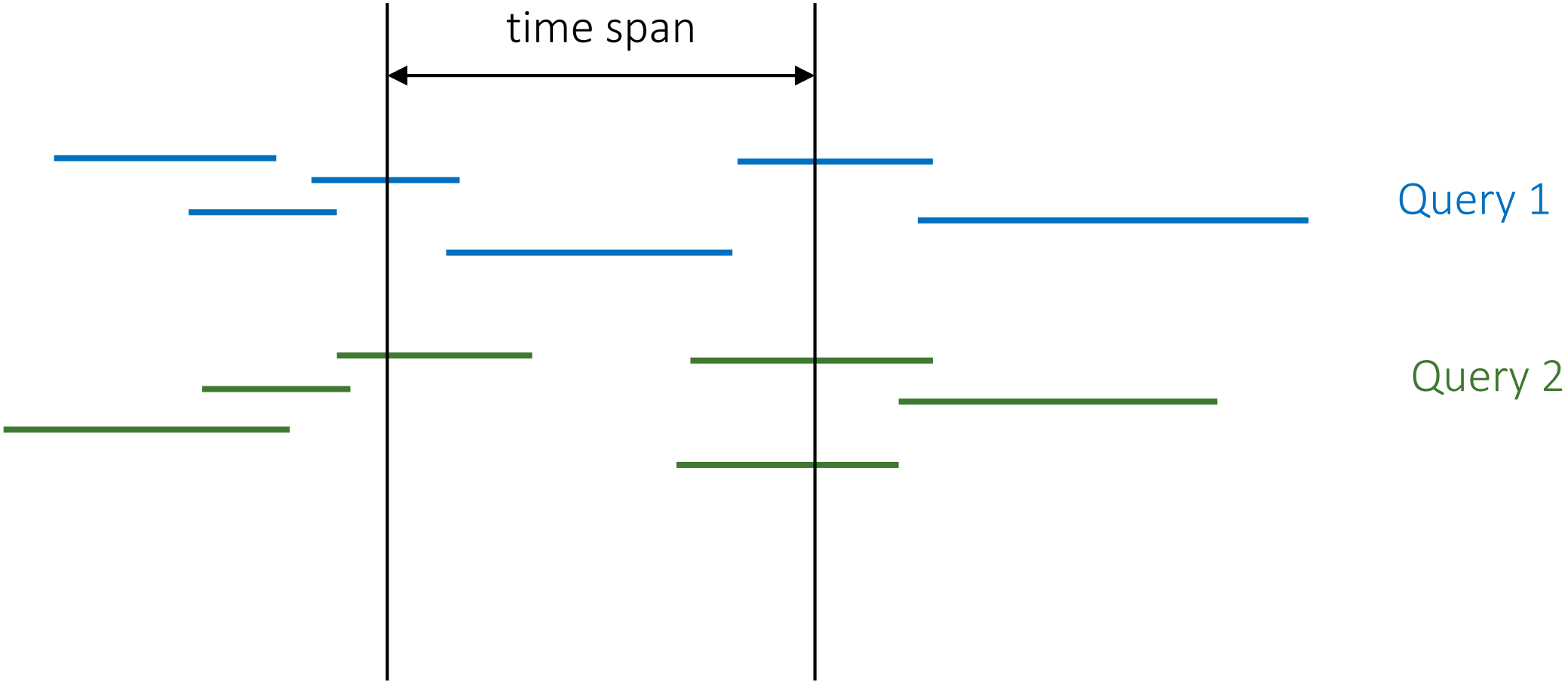
Cryptographic implementations

SQL queries

Fast Synthesis of Fast Collections

[Loncaric et al. PLDI'16]

Myria distributed database: needs to retrieve all queries in a given timespan



Specification

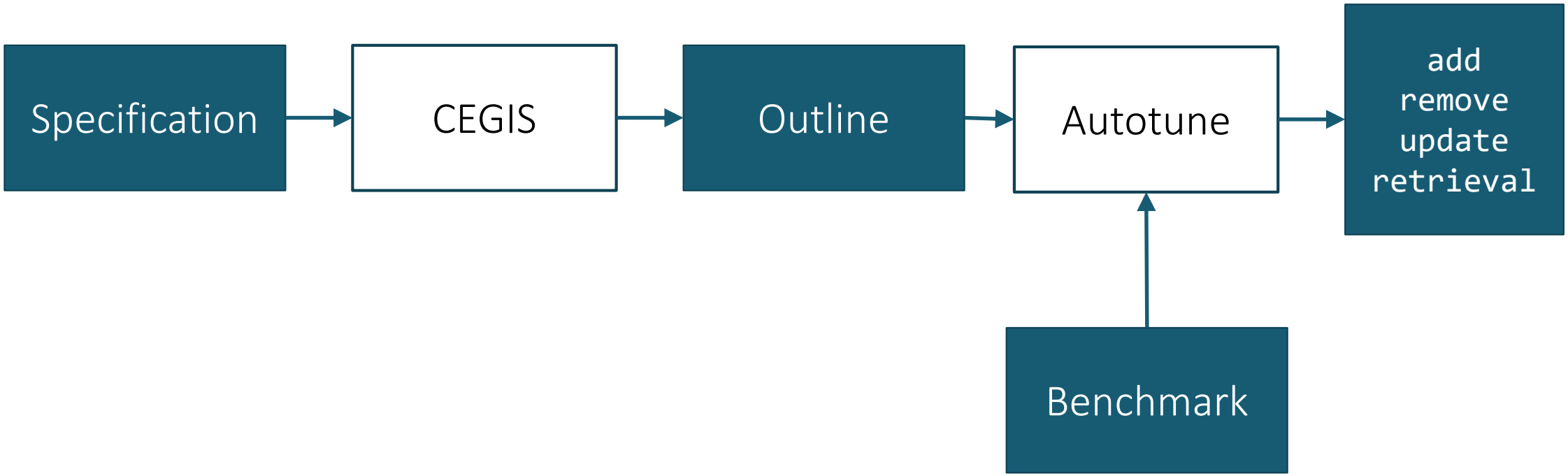
```
fields
  queryId:long, subqueryId:long,
  fragmentId:int, opId:int,
  startTime:long, endTime:long,

  assume startTime <= endTime

query getAnalyticsInTimespan(
    v_queryId:long, v_subqueryId:long,
    v_fragmentId:int,
    v_start:long, v_end:long)
  assume v_start <= v_end
  queryId == v_queryId and
  subqueryId == v_subqueryId and
  fragmentId == v_fragmentId and
  startTime < v_end and
  endTime >= v_start

costmodel myria-cost.java
```

The Cozy workflow



Results

Match performance of hand-written code on four real-world applications:

- Myria (a distributed data-base)
- Bullet (a physics simulation library)
- ZTopo (a topographic map viewer)
- Sat4J (a SAT solver)