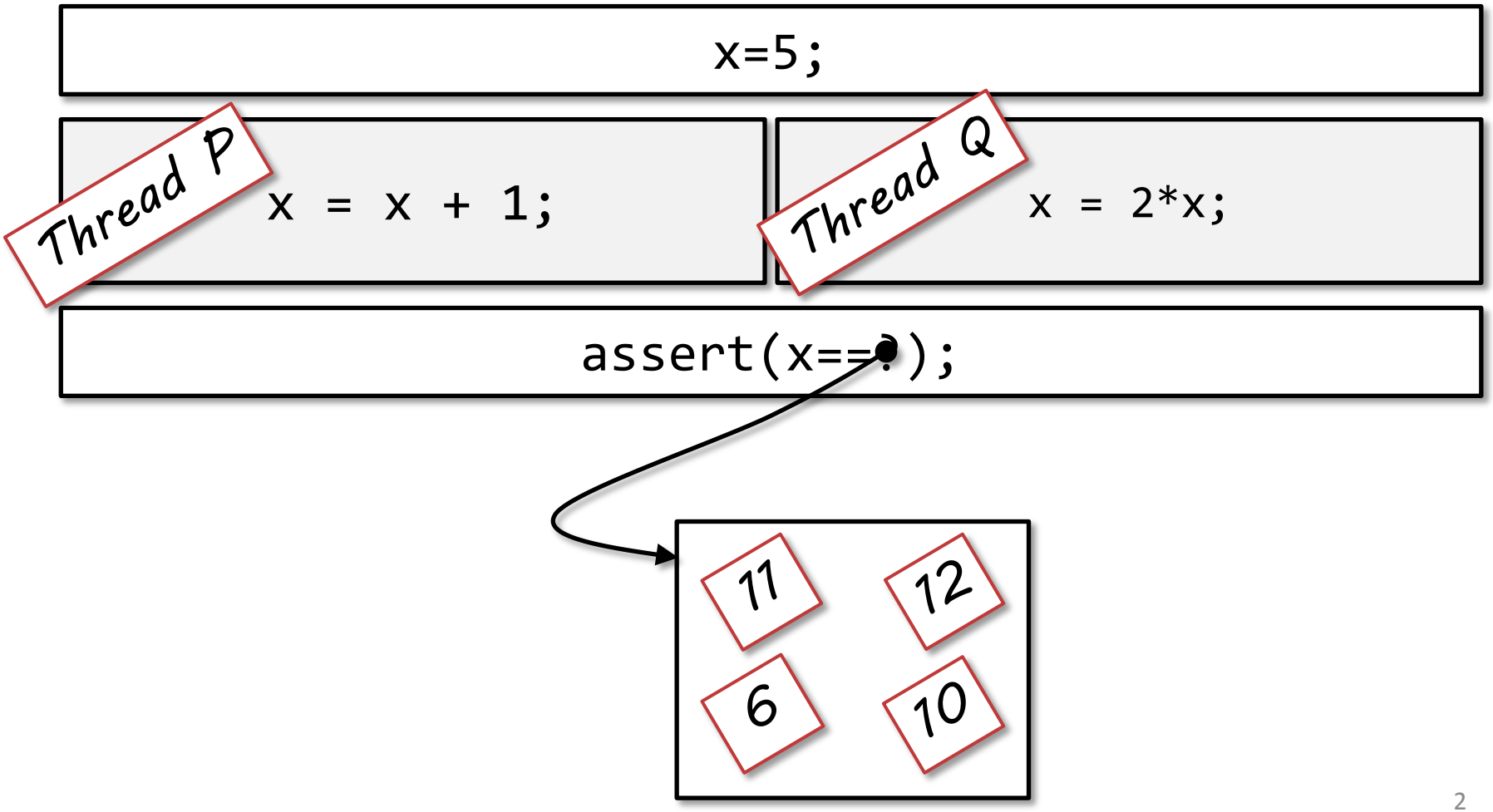
Concurrent Programming with Revisions and Isolation Types

2010

Authors:
Sebastian Burckhardt (Microsoft Research)
Alexandro Baldassin (State University of Campinas)
Daan Leijen (Microsoft Research)

Presented by: Yves Bonjour

Motivation

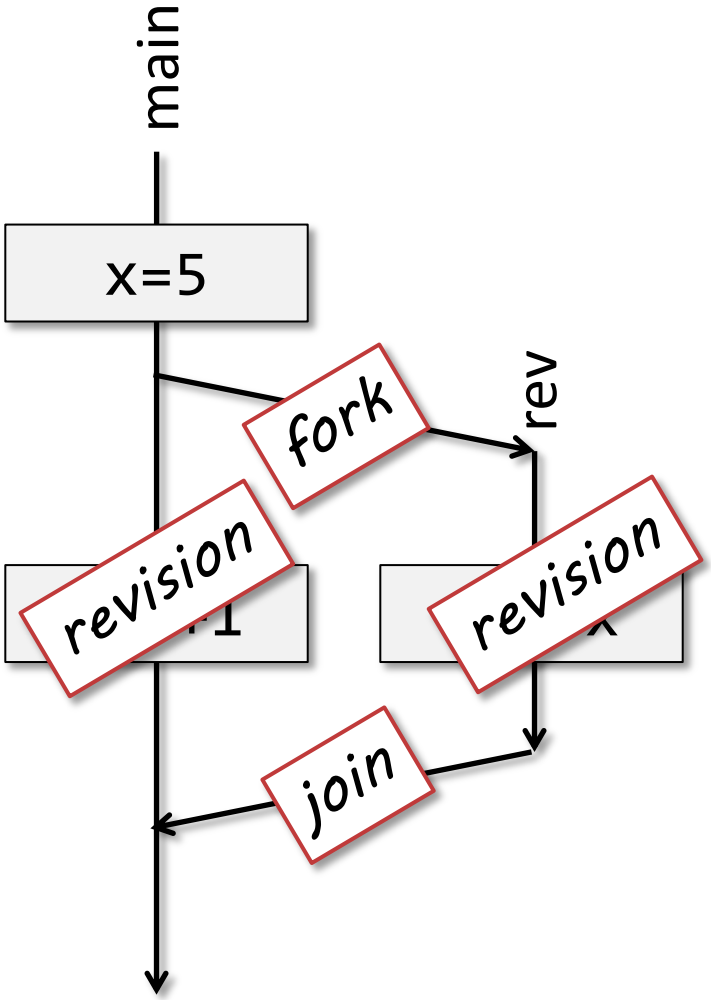


Concurrent Revisions

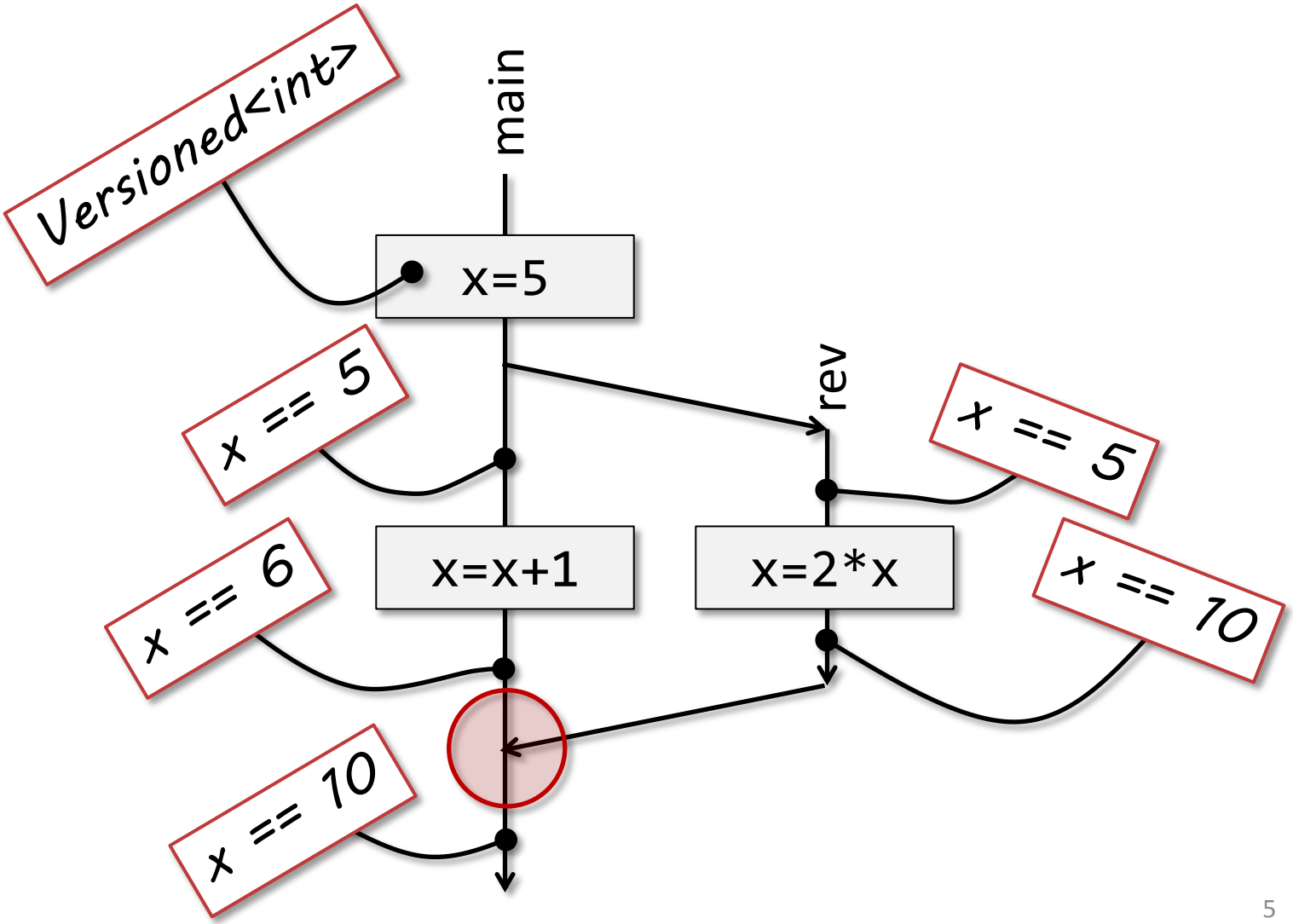
- Concept from Source Control Systems

- Revisions
  - Run in isolation
  - Operate on copy of shared state
  - Resolve write conflicts

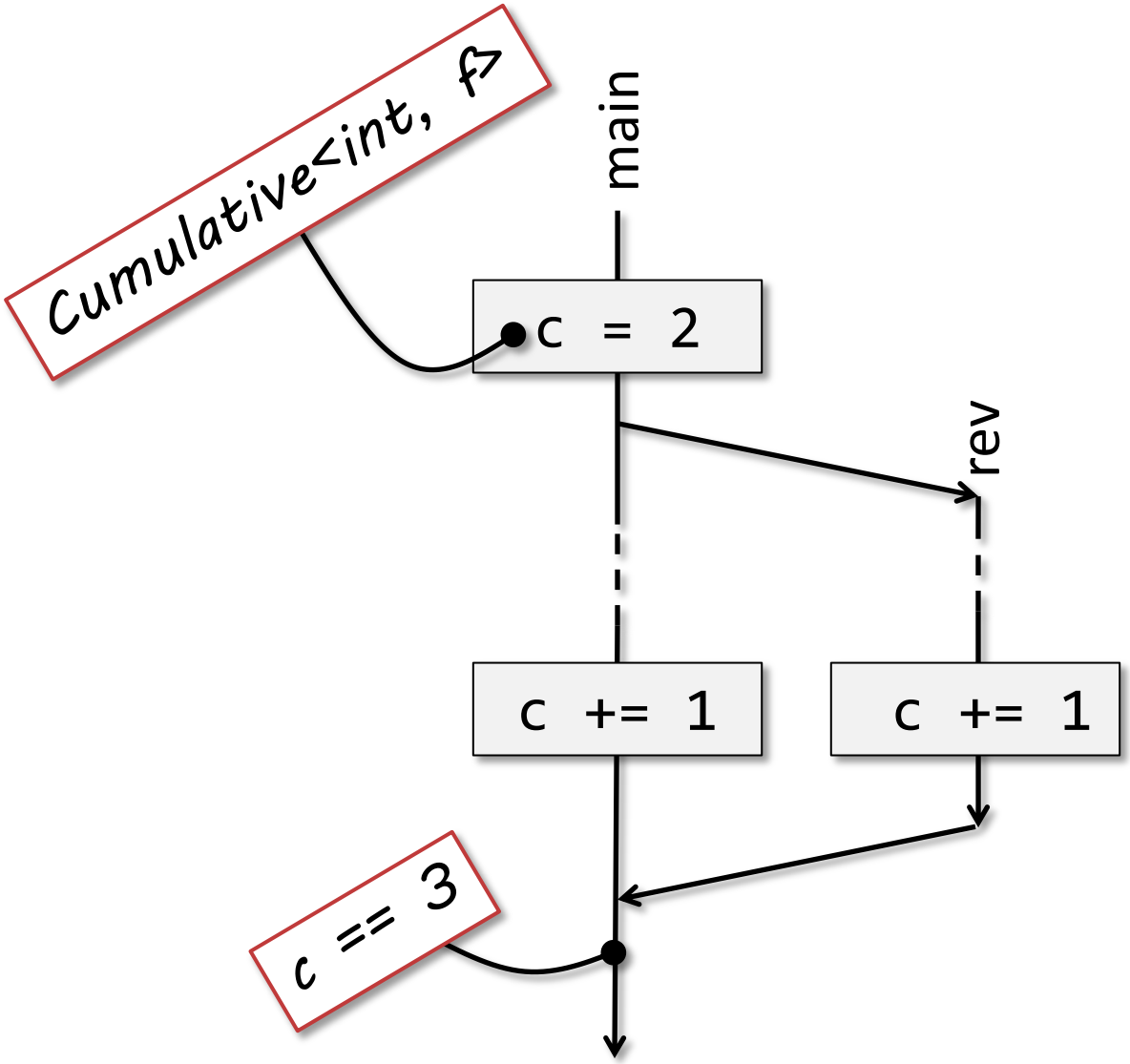
Revision diagrams



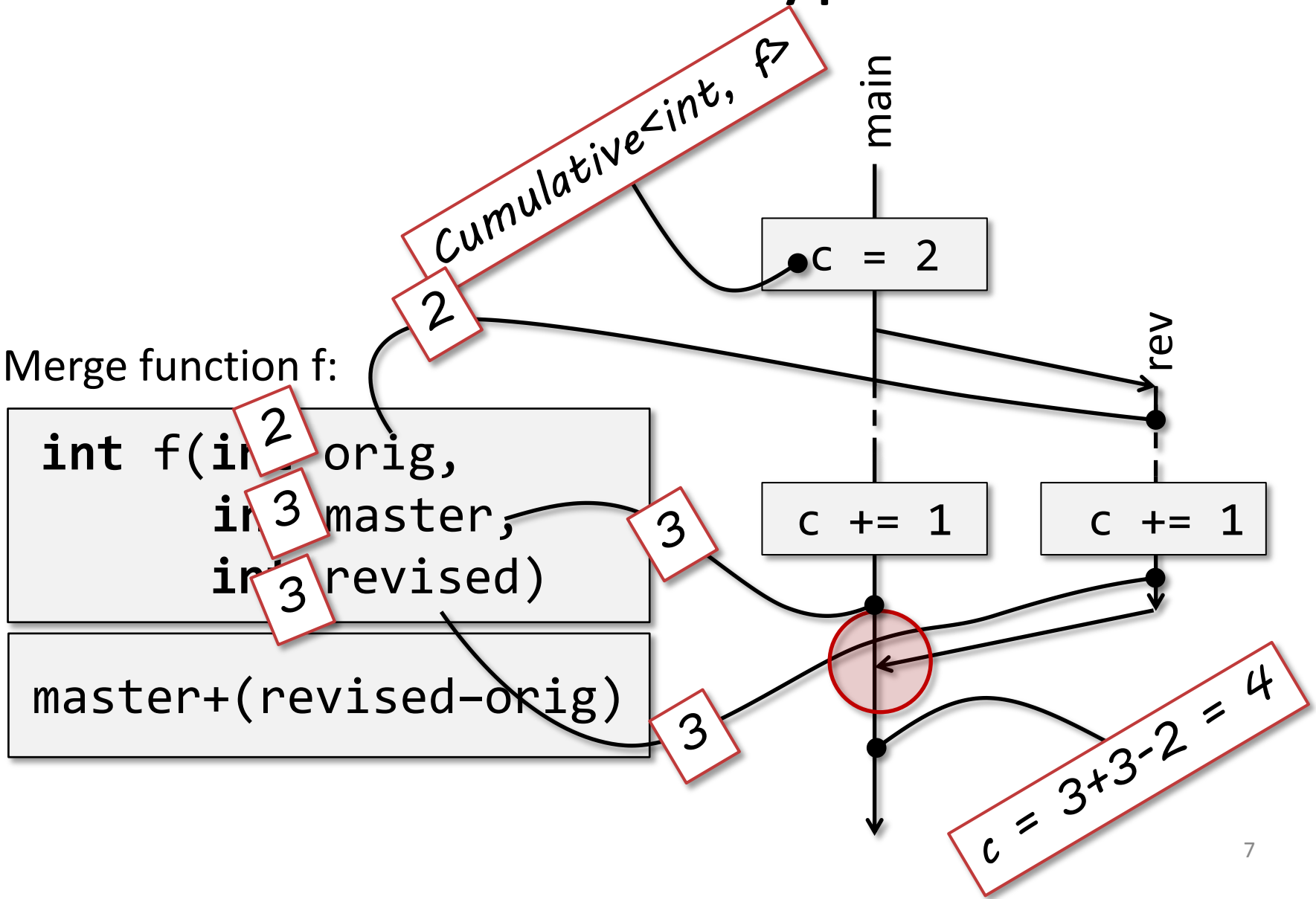
Example - Versioned



Example - Counter



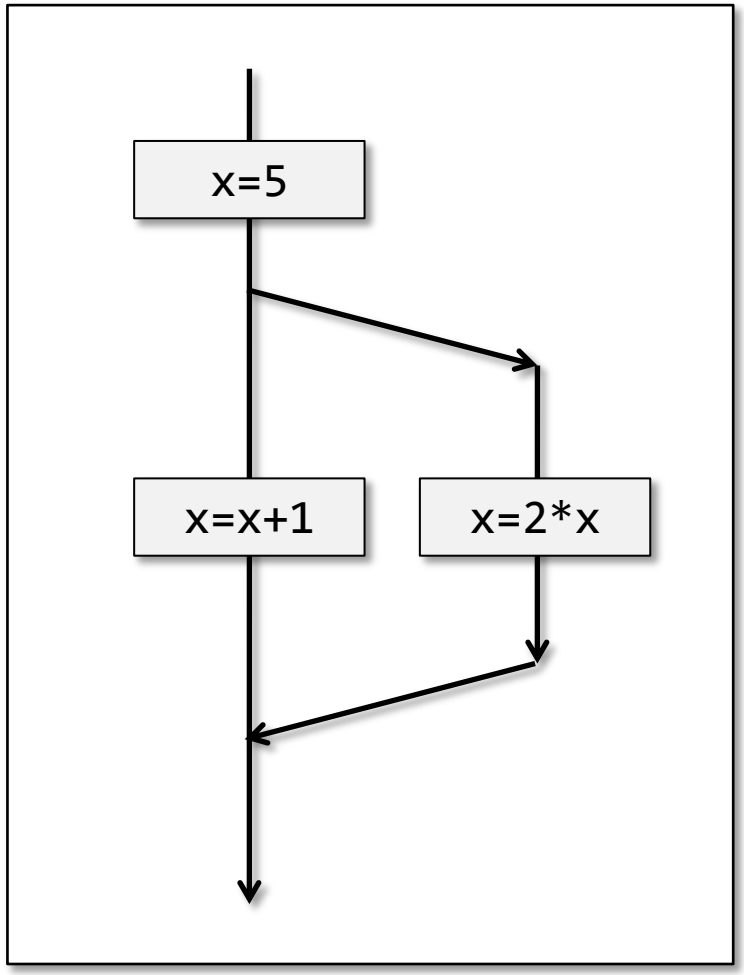
Cumulative Types



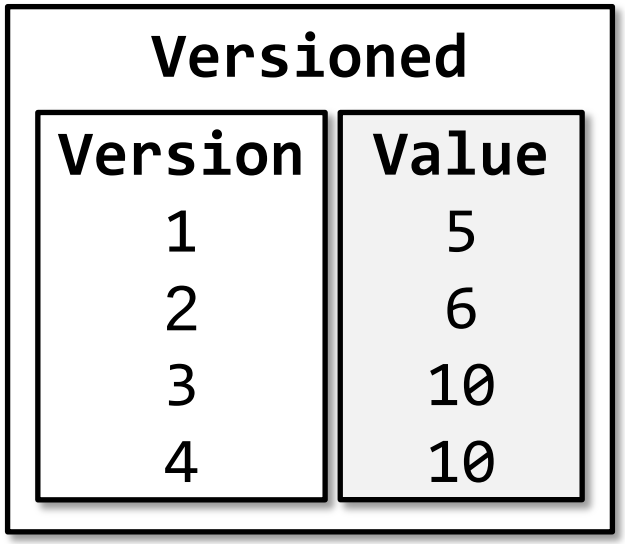
Code

Annotation

```
…
[Versioned]
int x=5;
…
RevisionTask r =
CurrentRevision.Fork(() =>
    {
        x = 2*x;
    }
);
x=x+1
CurrentRevision.Join(r);
…
```

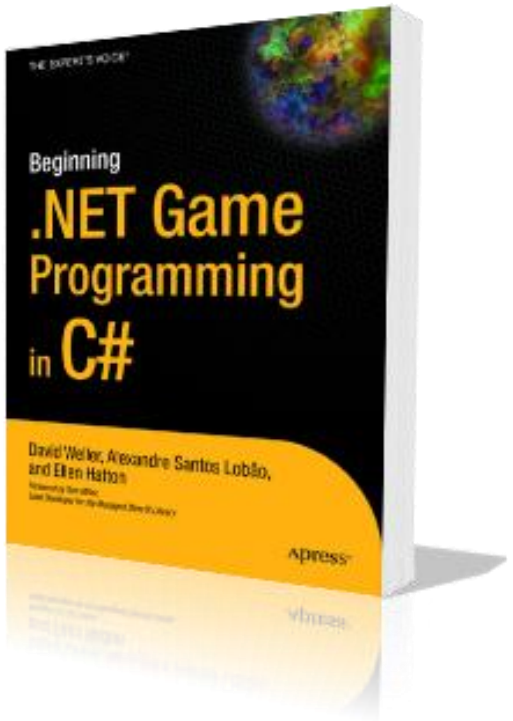


Implementation



- Lazy copy
- Release unused values
- Lock free

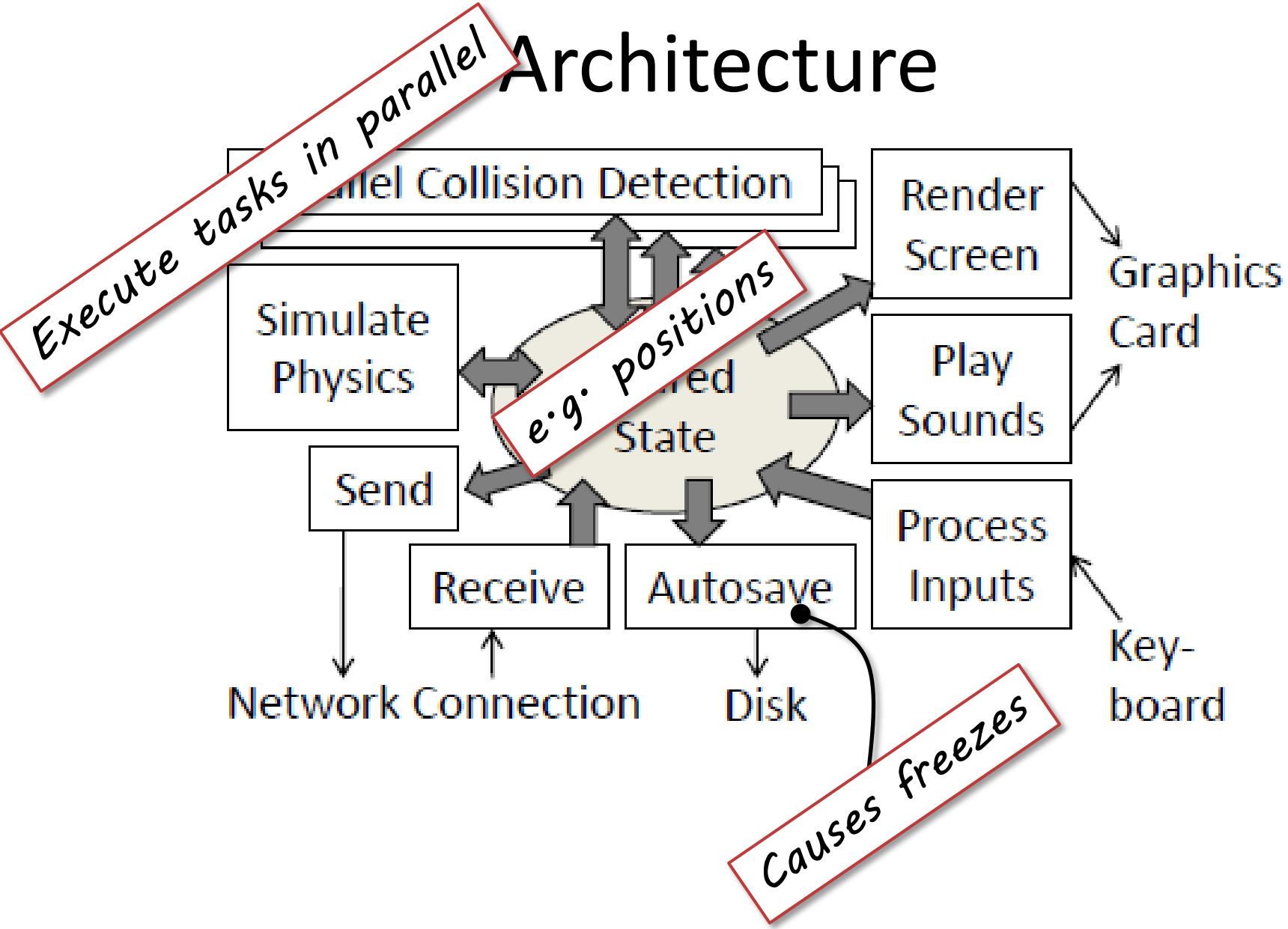
Case Study – SpaceWars3D



12’000 lines of code

[apress.com]

Architecture

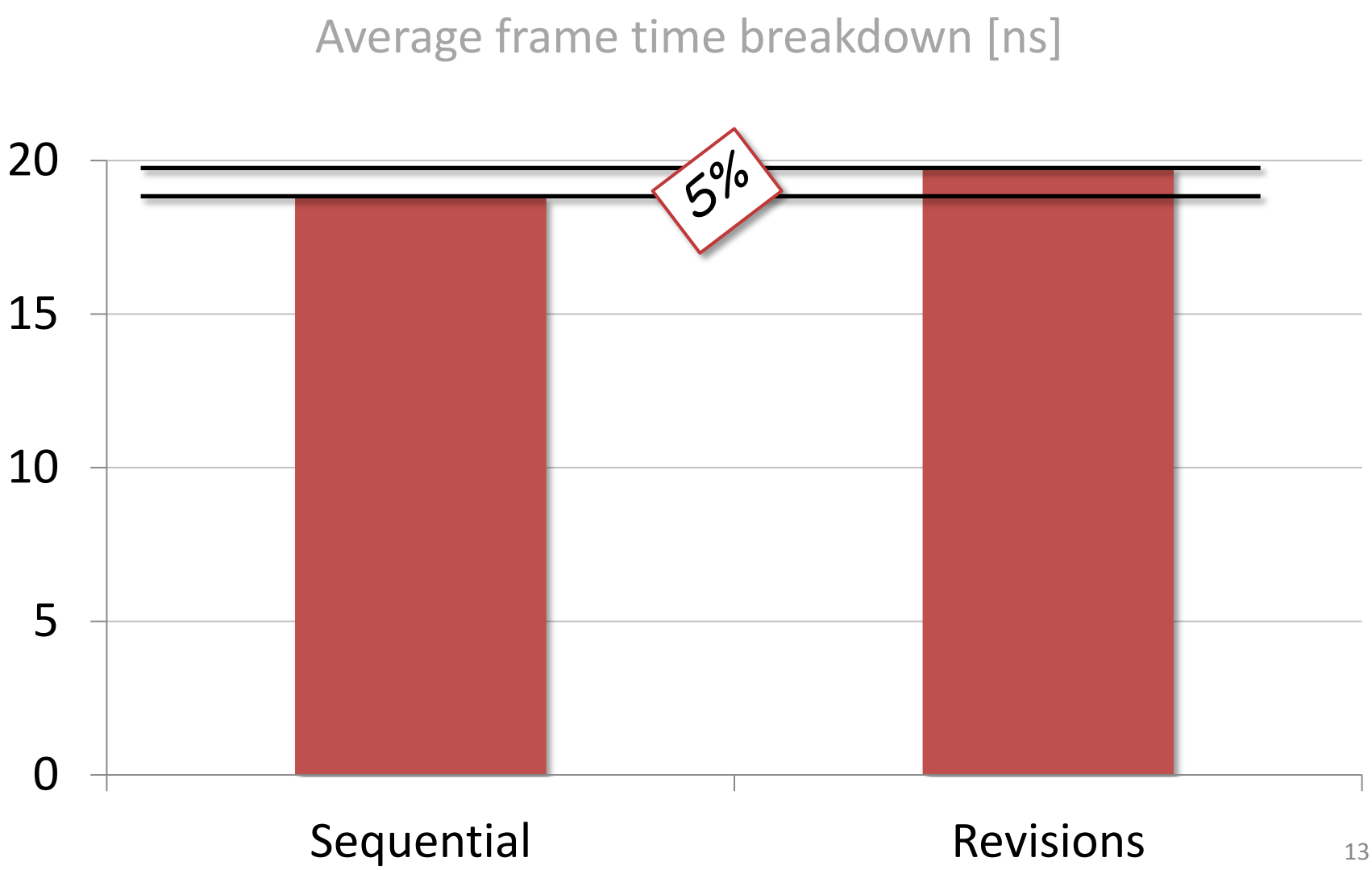


[Concurrent Programming with Revisions and Isolation Types, S. Burckhardt et. al., 2010]

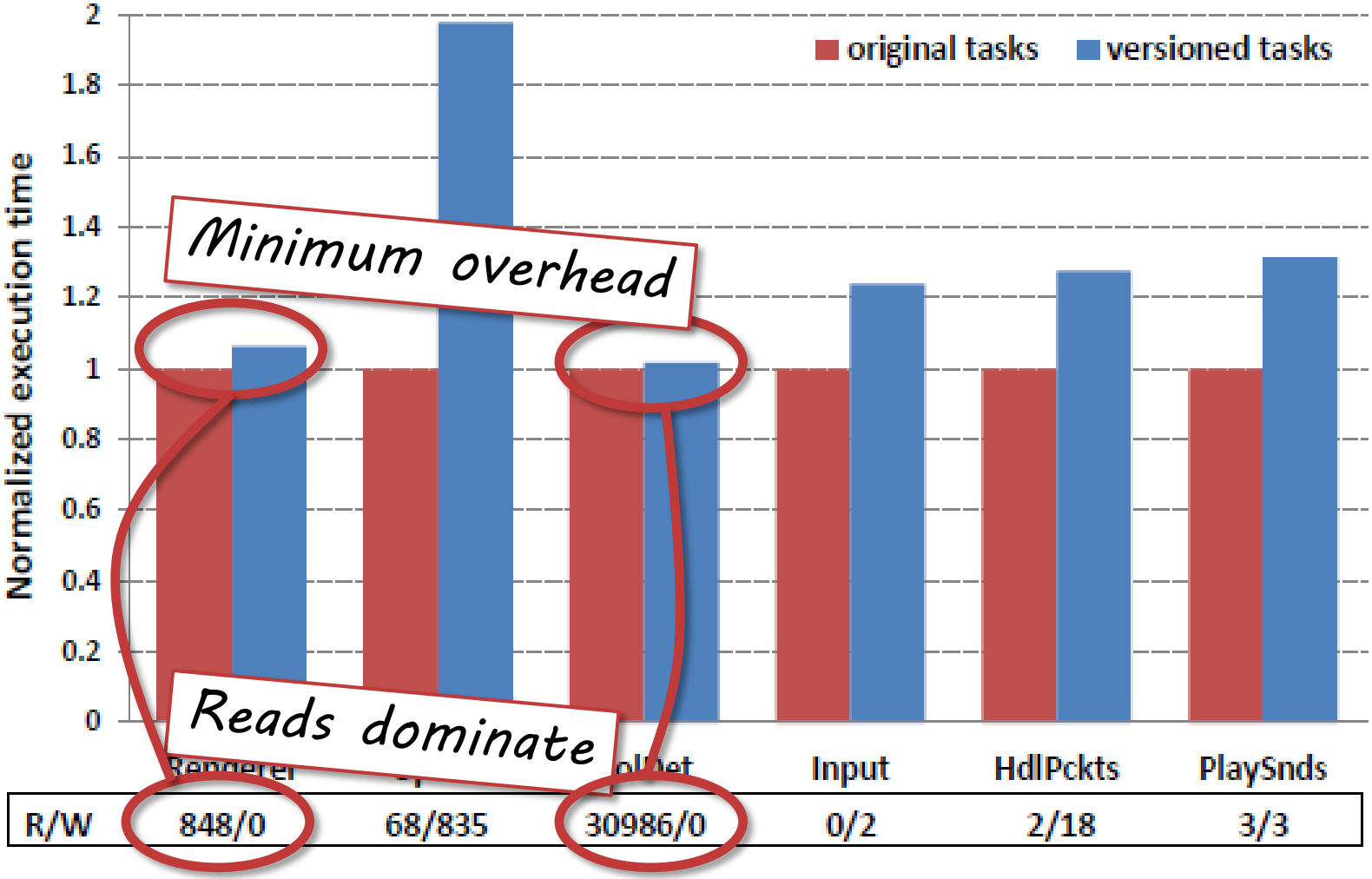
Methodology

- Typical game
  - 2 players
  - 800 asteroids
- #asteroids $\longrightarrow$ proportion of parallelizable computation
- Record and Replay
- 2000 frames

Revision Overhead

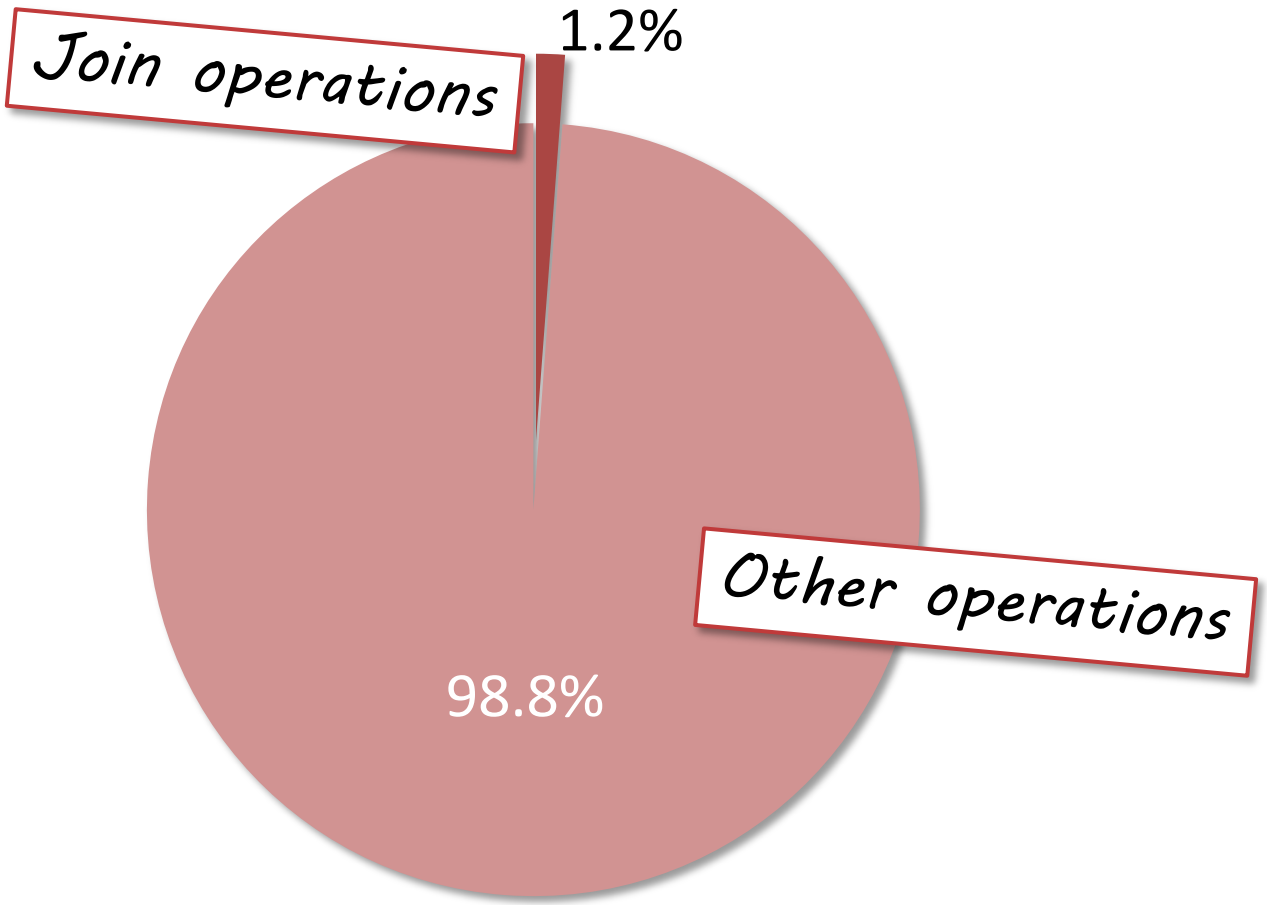


Revision Overhead (2)

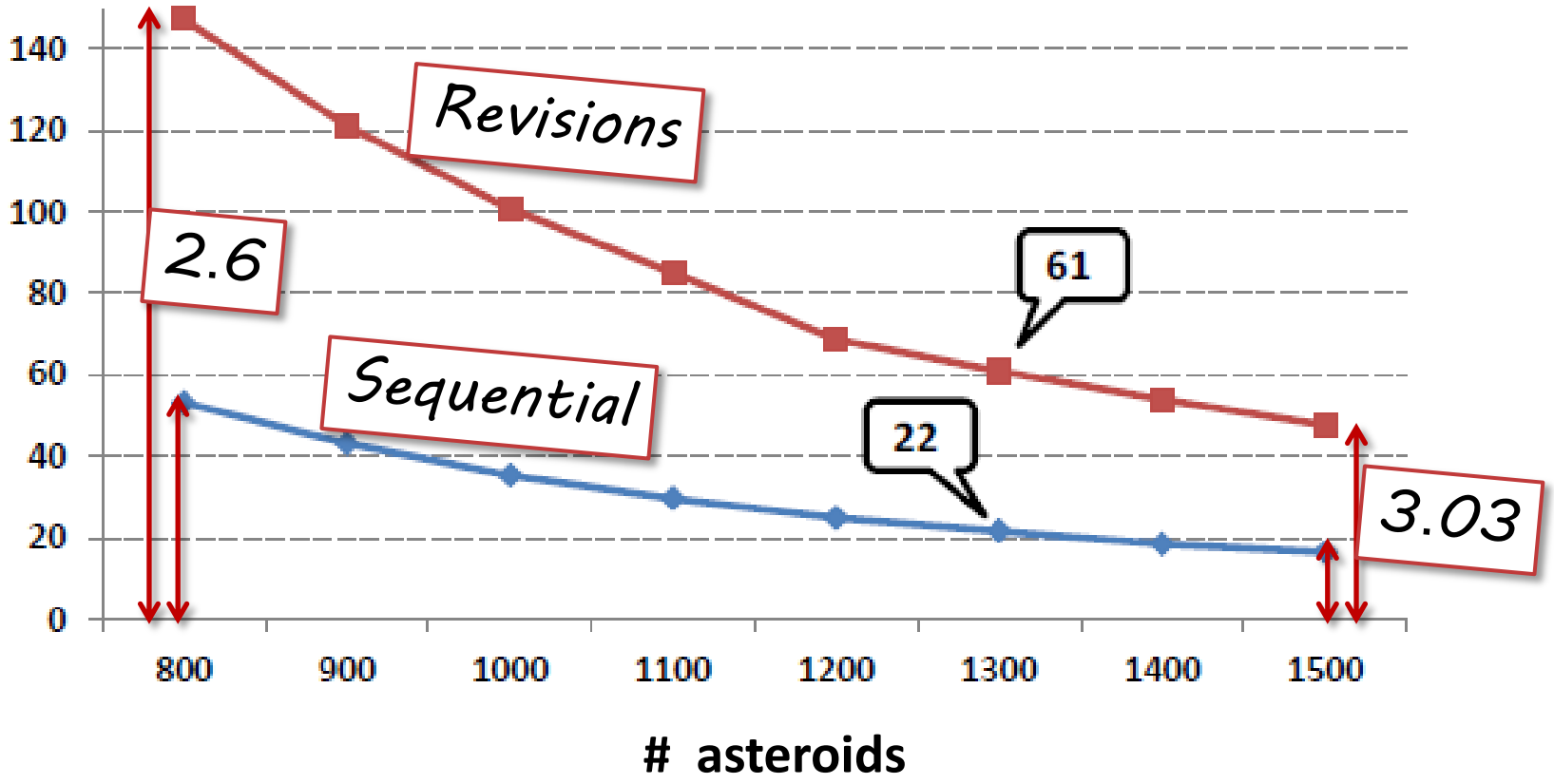


[Concurrent Programming with Revisions and Isolation Types, S. Burckhardt et. al., 2010]

Runtime Of Join Operations



Parallel Speed Up (4 cores)



[Concurrent Programming with Revisions and Isolation Types, S. Burckhardt et. al., 2010]

Conclusion

My opinion…

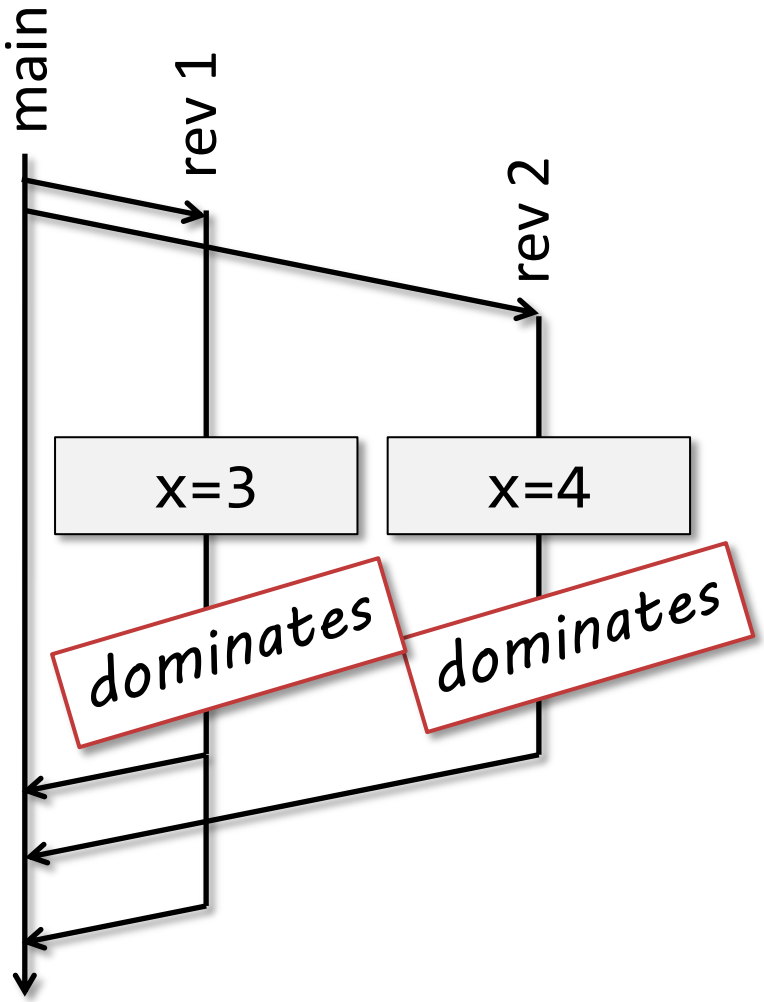
- Simple
- Data centric
- Lock free
- Reasoning with revision diagrams
  - Deterministic

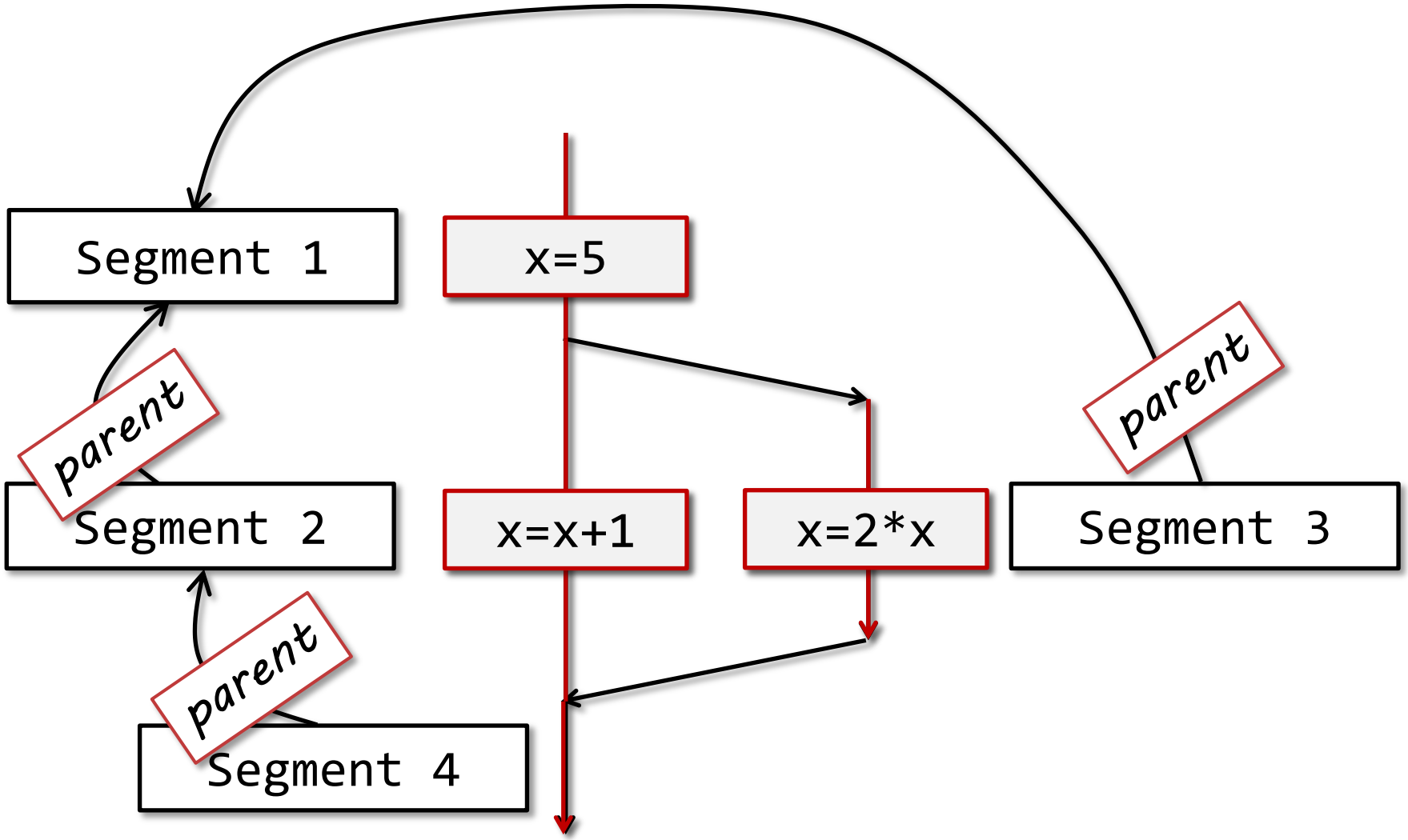
- Performance with more writes?
- Join time accuracy?
  - More complex data structures

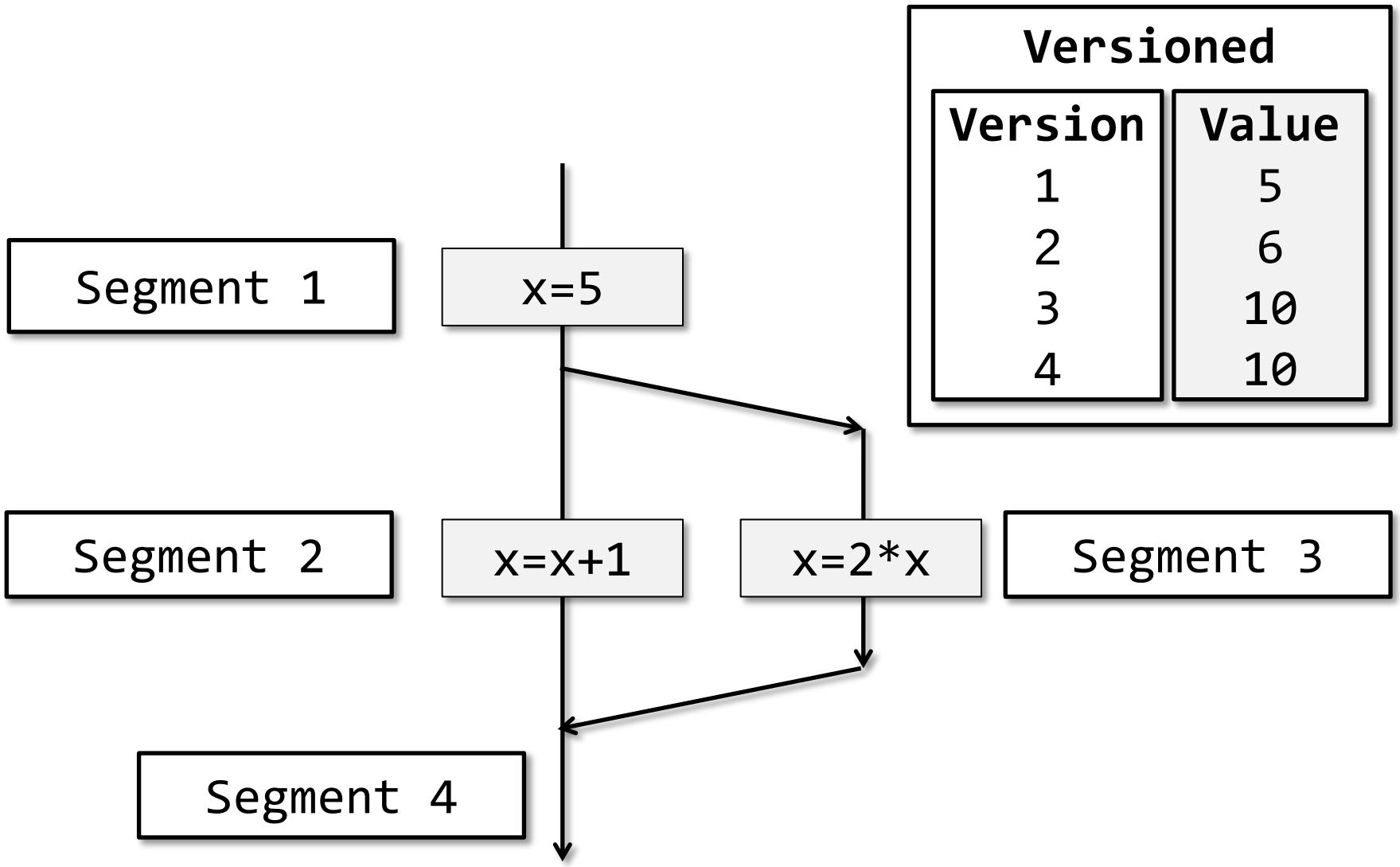
rise4fun.com/Revisions

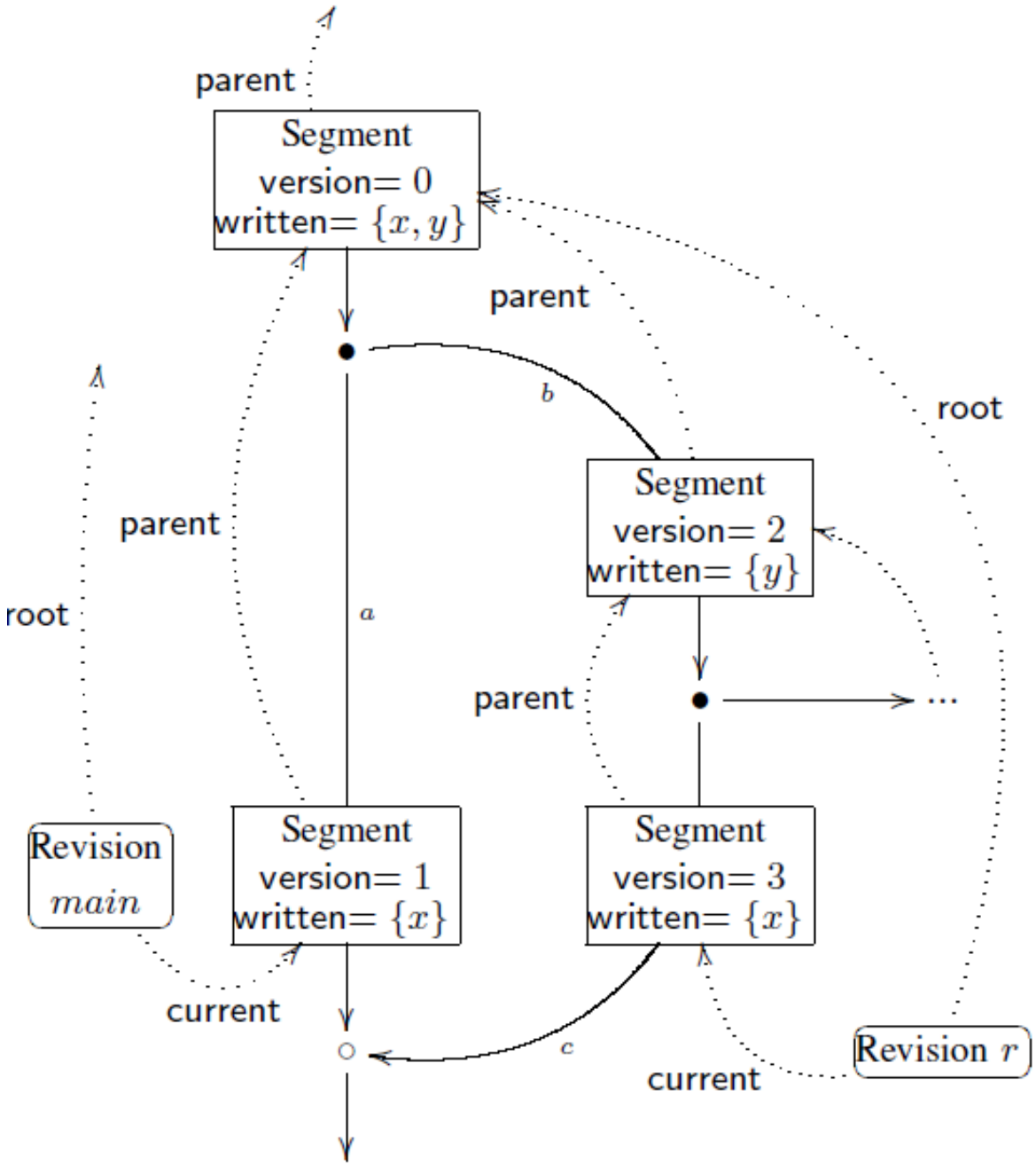
Additional slides

Control The Sideffects

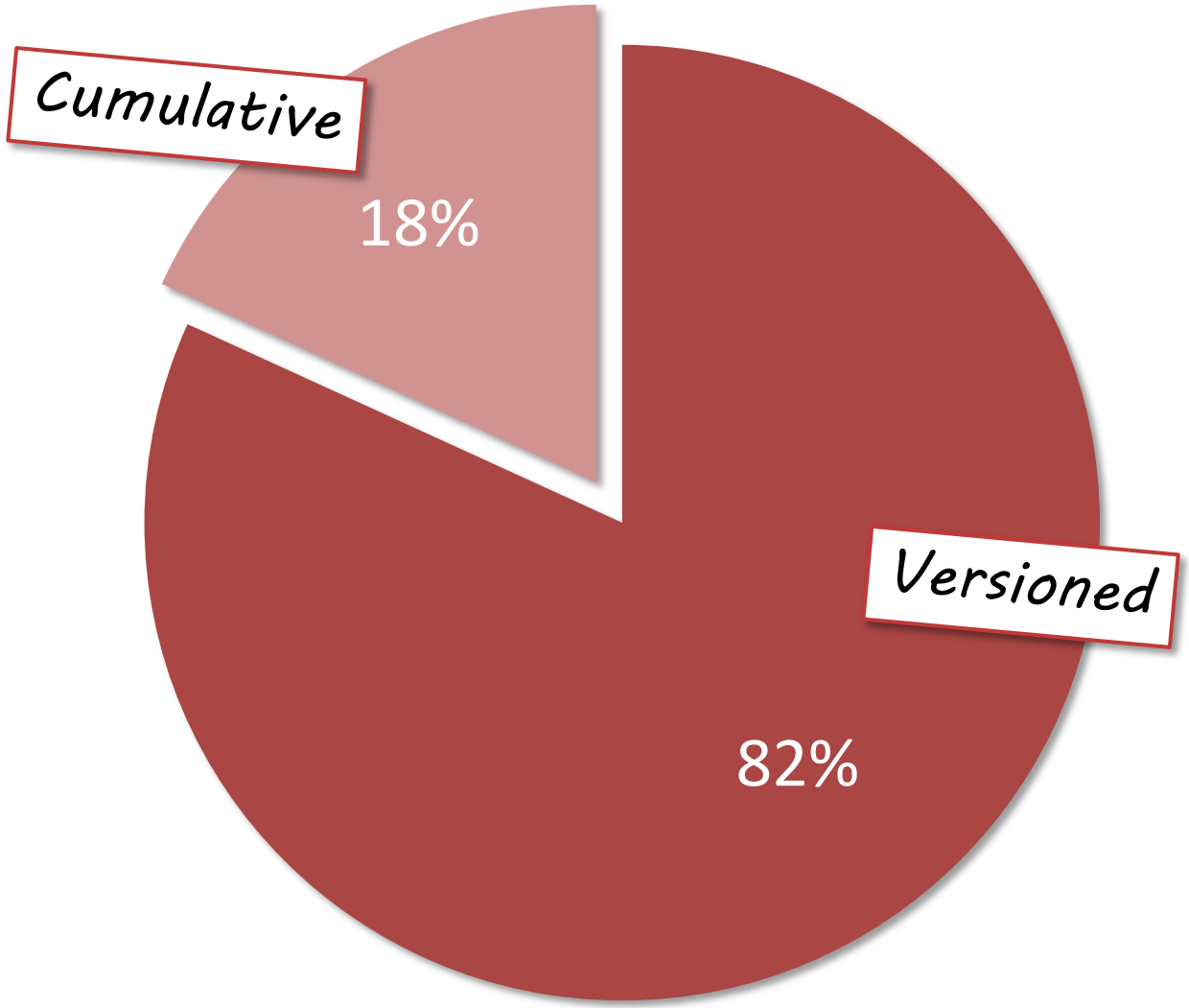



Segment 1
x=5
parent
parent
Segment 2
x=x+1
x=2*x
Segment 3
parent
Segment 4


Versioned
Version
Value
1
5
2
6
3
10
4
10
Segment 1
x=5
Segment 2
x=x+1
x=2*x
Segment 3
Segment 4


parent
Segment
version= 0
written= {x, y}
parent
b
root
parent
Segment
version= 2
written= {y}
root
a
parent
...
Revision
main
Segment
version= 1
written= {x}
Segment
version= 3
written= {x}
current
c
current
Revision r

Proportion Of Isolation Types



Memory Overhead