Incremental Updates for Graph Queries

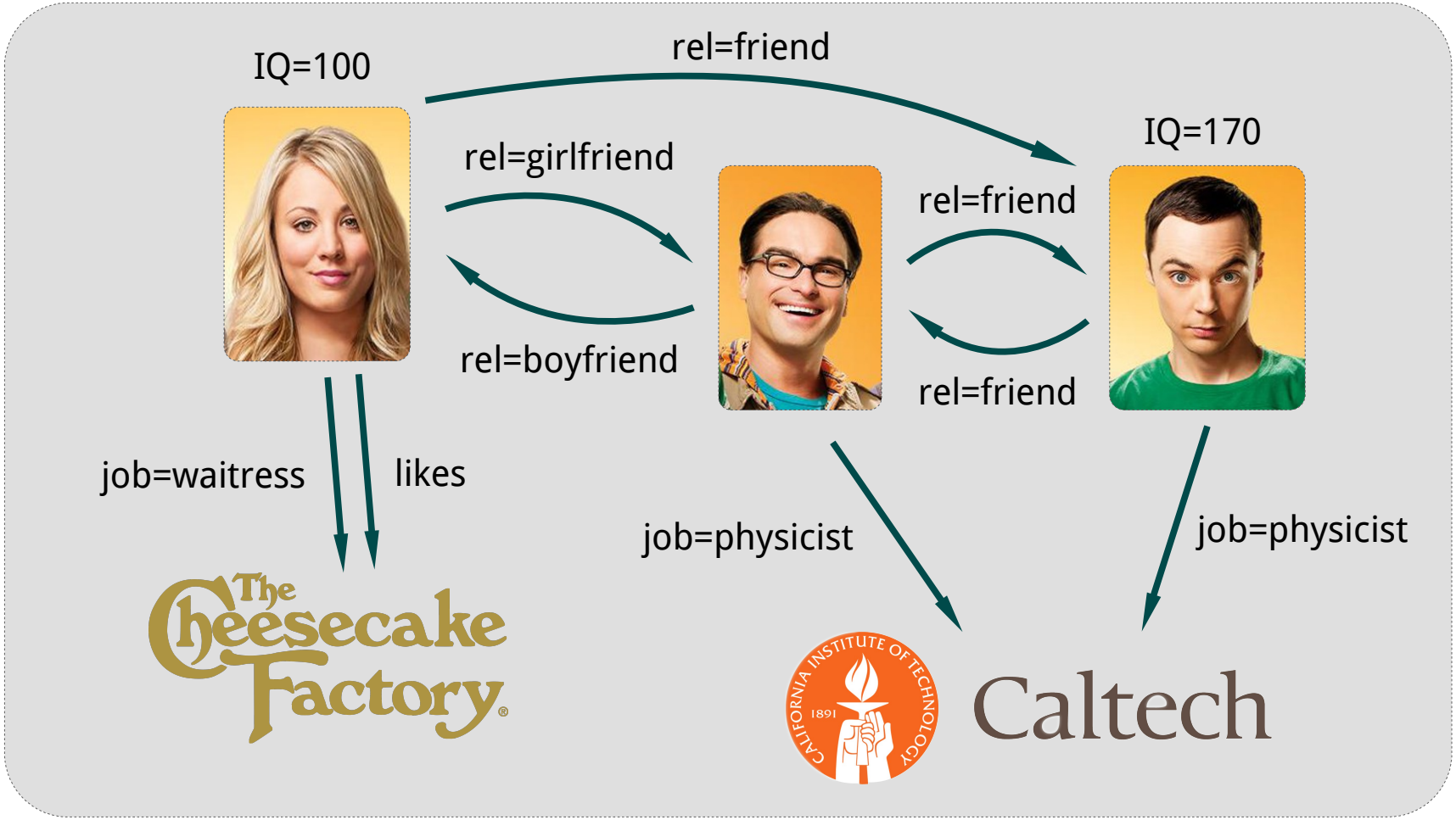
Oracle VM meetup – September 2014

Sandro Stucki, Oracle Labs

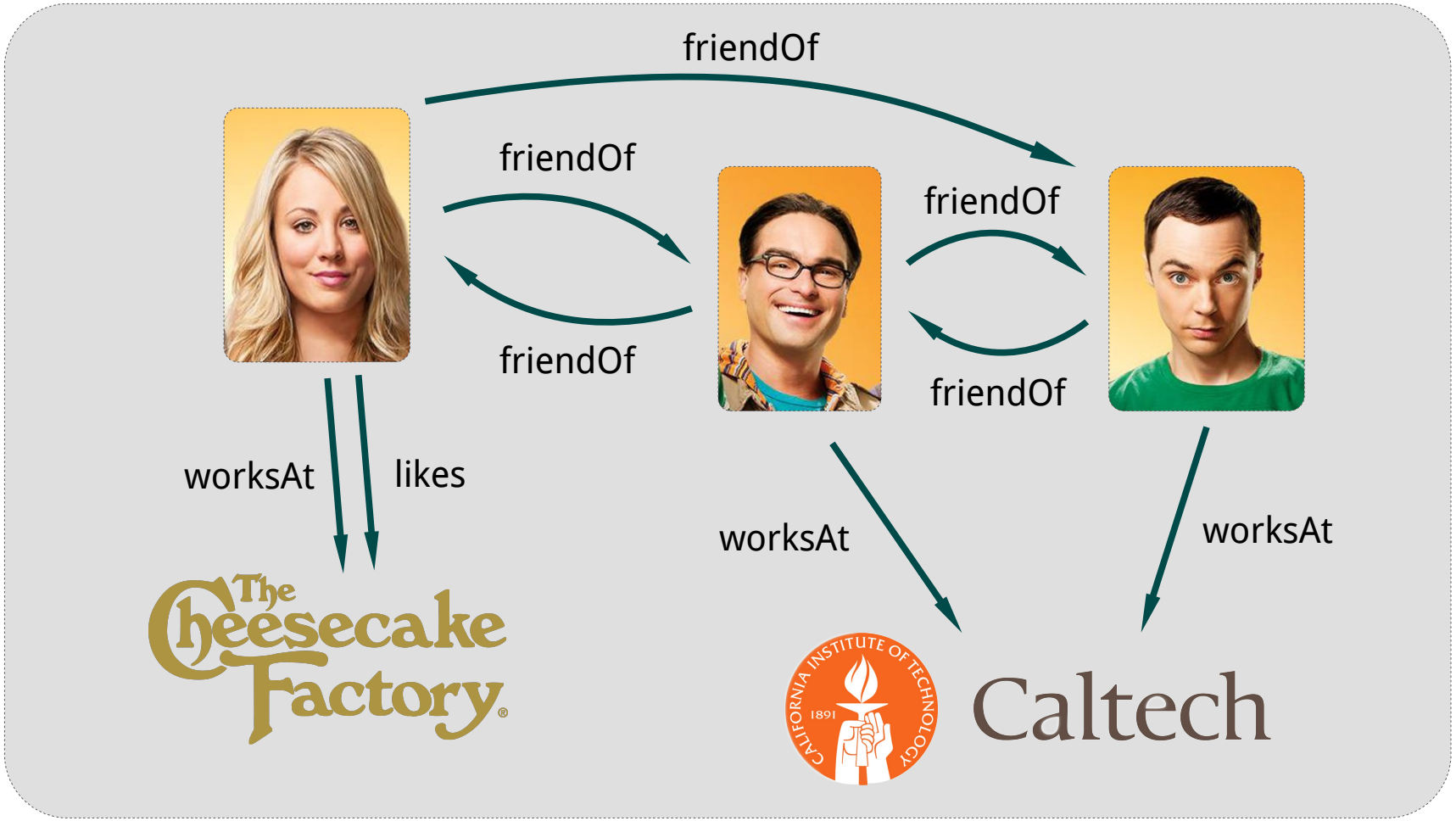
Disclaimer

The following is intended to outline our research activities and/or general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decision. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Graph Databases



Graph Databases



PGX.ISO

- Parallel in-memory graph query engine
- Developed by Oracle Labs
- Based on Green-Marl runtime

PGX.ISO: Parallel and Efficient In-Memory Engine for Subgraph Isomorphism

Raghavan Raman
Oracle Labs
raghavan.raman@oracle.com

Oskar van Rest
Oracle Labs
oskar.van.rest@oracle.com

Sungpack Hong
Oracle Labs
sungpack.hong@oracle.com

Zhe Wu
Oracle
alan.wu@oracle.com

Hassan Chafi
Oracle Labs
hassan.chafi@oracle.com

Jay Banerjee
Oracle
jayanta.banerjee@oracle.com

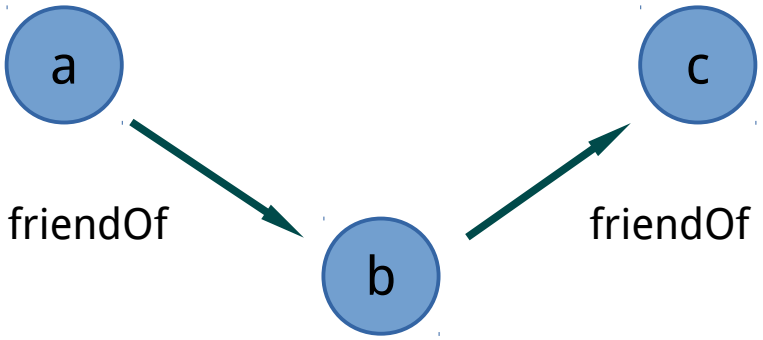
ABSTRACT

Subgraph isomorphism, or finding matching patterns in a graph, is a classic graph problem that has many practical use

Consequently, many solutions have been proposed for this problem, including commercialized ones. First, there are graph databases [9, 3] which adopt the classic RDF graph

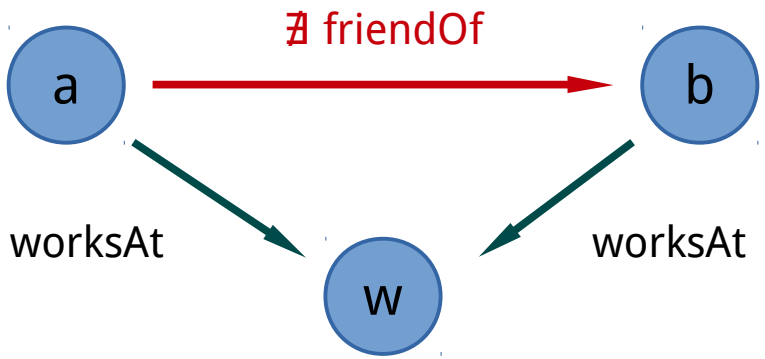
Graph Queries

```
SELECT ?a ?c
WHERE {
     ?a friendOf ?b .
     ?b friendOf ?c .
}
```



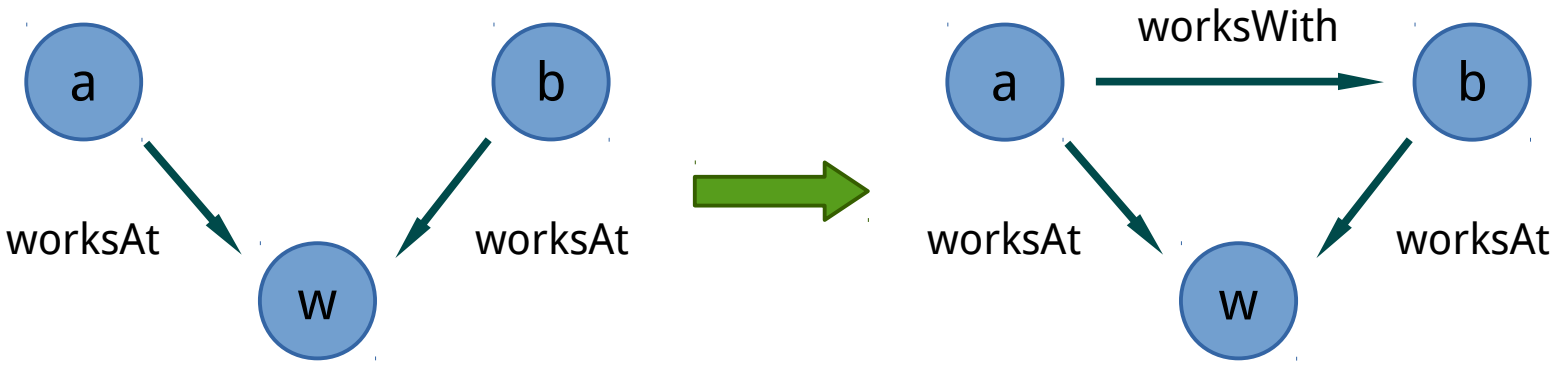
Filters

```
SELECT ?a ?b
WHERE {
     ?a worksAt ?w .
     ?b worksAt ?w .
     FILTER NOT EXISTS { ?a friendOf ?b }
}
```



Updates

```
INSERT { ?a worksWith ?b }
WHERE {
     ?a worksAt ?w .
     ?b worksAt ?w .
}
```



Updates

- Caused by
  - update queries
  - “external” changes to the DB
- Performance issues
  - Data structures allowing fast in-memory updates
  - Update query results after updates to DB

Updates

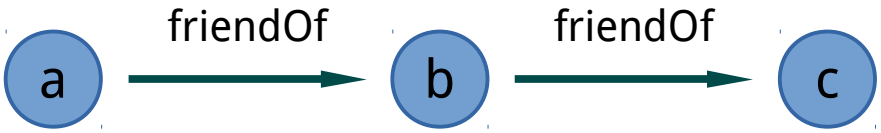
- Caused by
  - update queries
  - “external” changes to the DB
- Performance issues
  - Data structures allowing fast in-memory updates
  - Update query results after updates to DB

Why should I care?

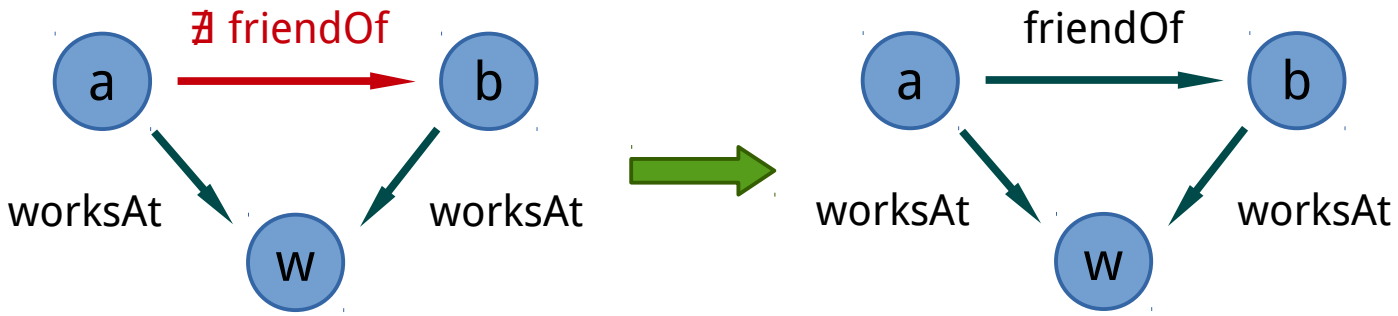
- We want graph analysis to be fast
- But also...
  - ... enables fast, large-scale graph rewriting
  - useful for some types of program analysis
  - used in “VMs” for modeling/simulating complex systems
    - biochemical system (Kappa/BNGL)
    - statistical physics
    - social networks
    - etc.

Updating query results

Example query



Example update

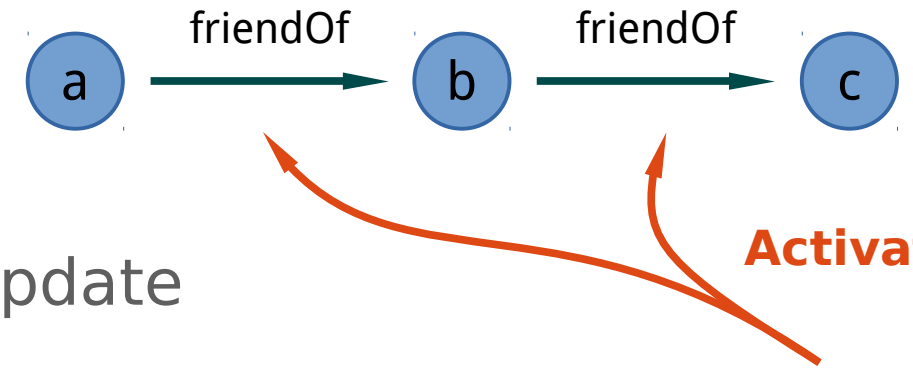


Updating query results

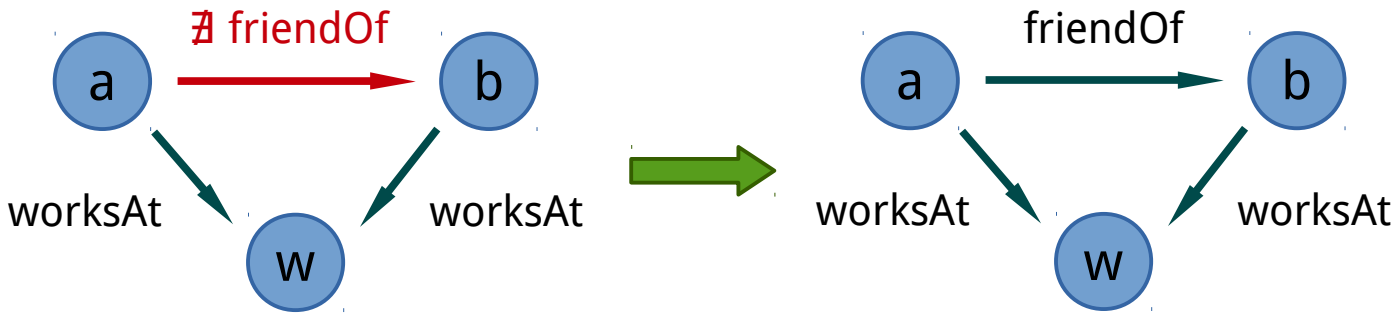
- Naive approach
  - Just re-run all queries upon updates
  - Inefficient, especially for small changes
- Which queries need updating?
- What type of update?
  - Addition/deletion of result
  - Location of result
- We'll need a bit of static analysis...

Dependencies

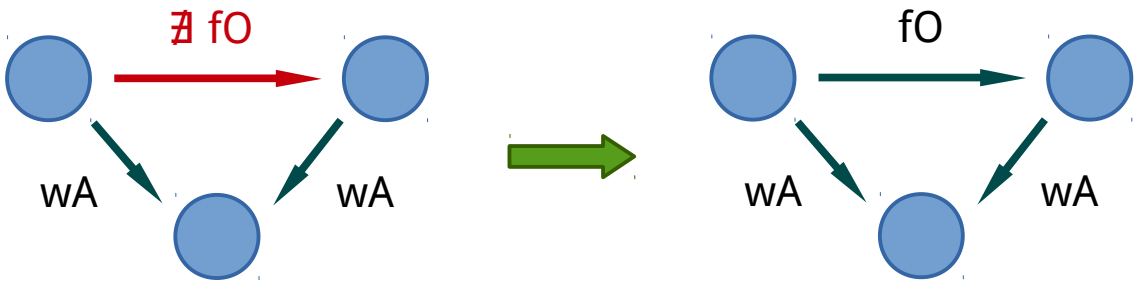
Example query



Example update



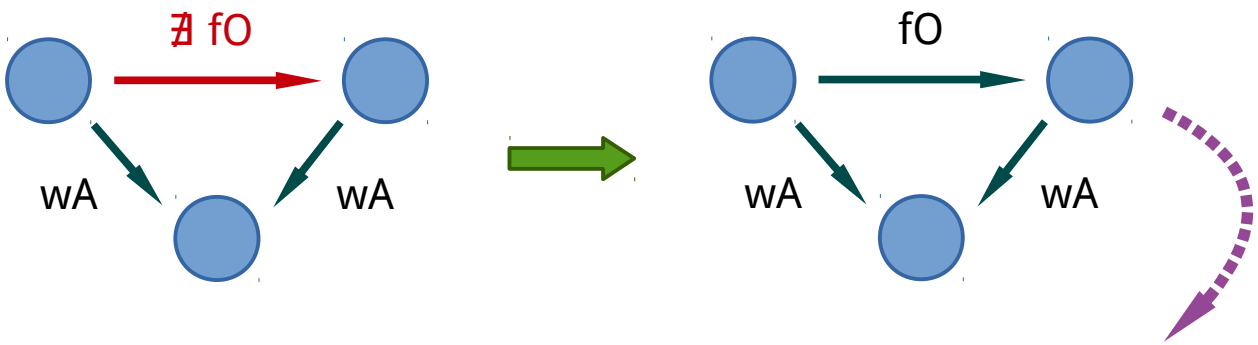
Overlaps



Observations

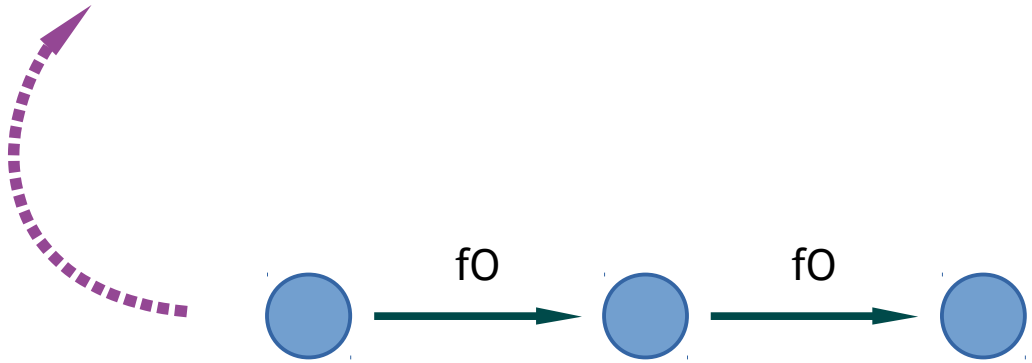
- dependencies correspond to overlaps

Overlaps

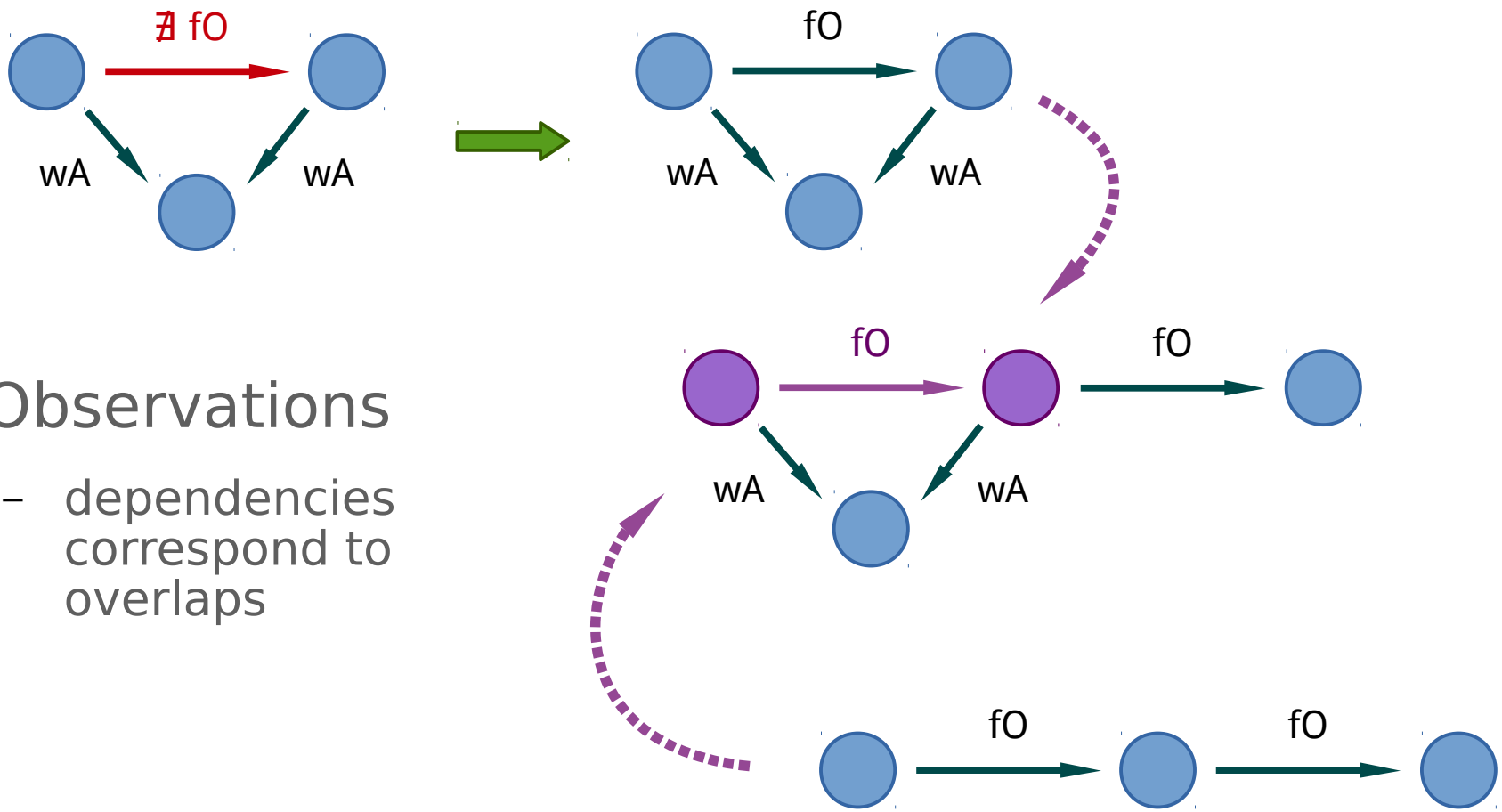


Observations

- dependencies correspond to overlaps



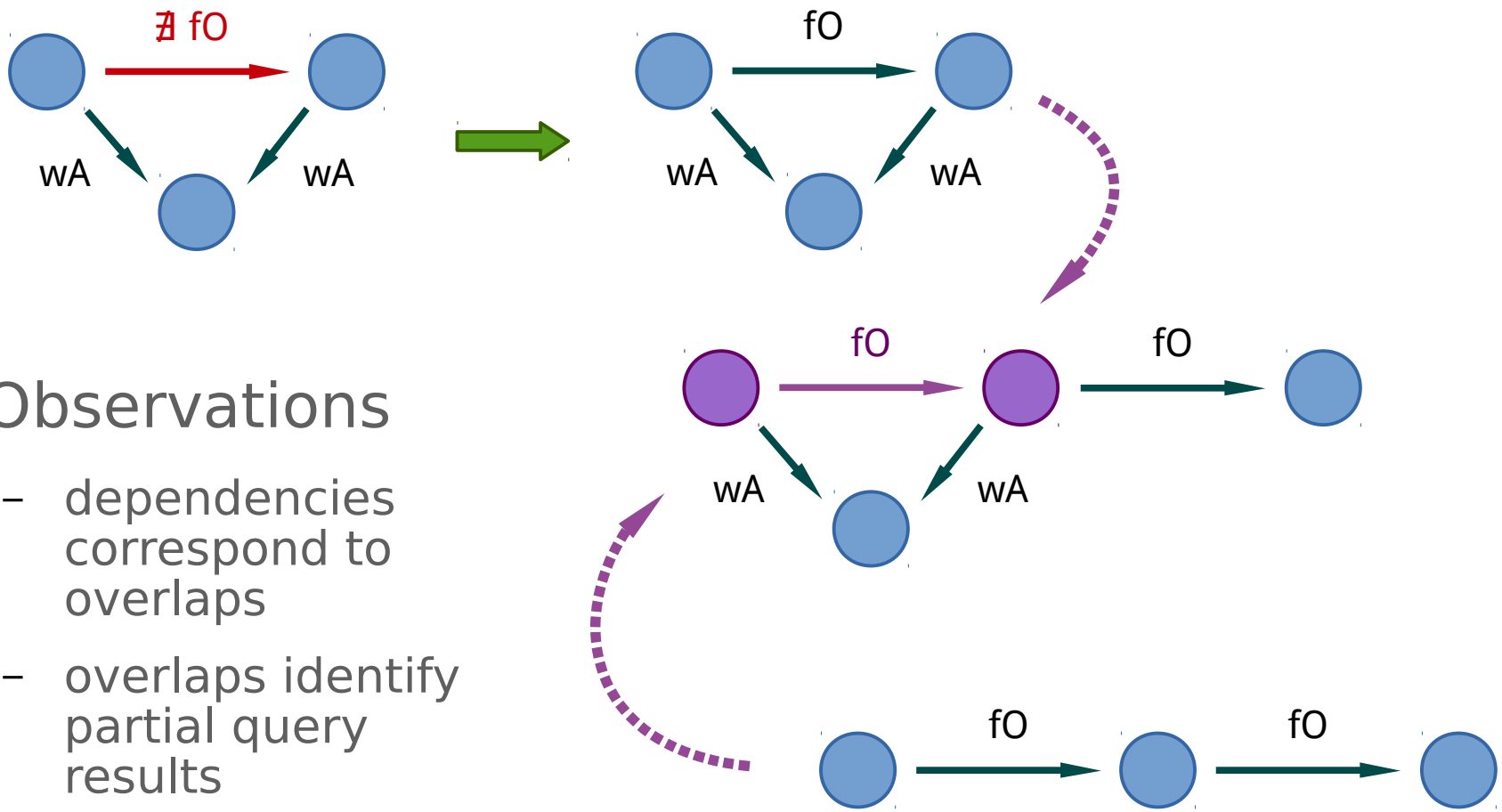
Overlaps



Observations

- dependencies correspond to overlaps

Overlaps



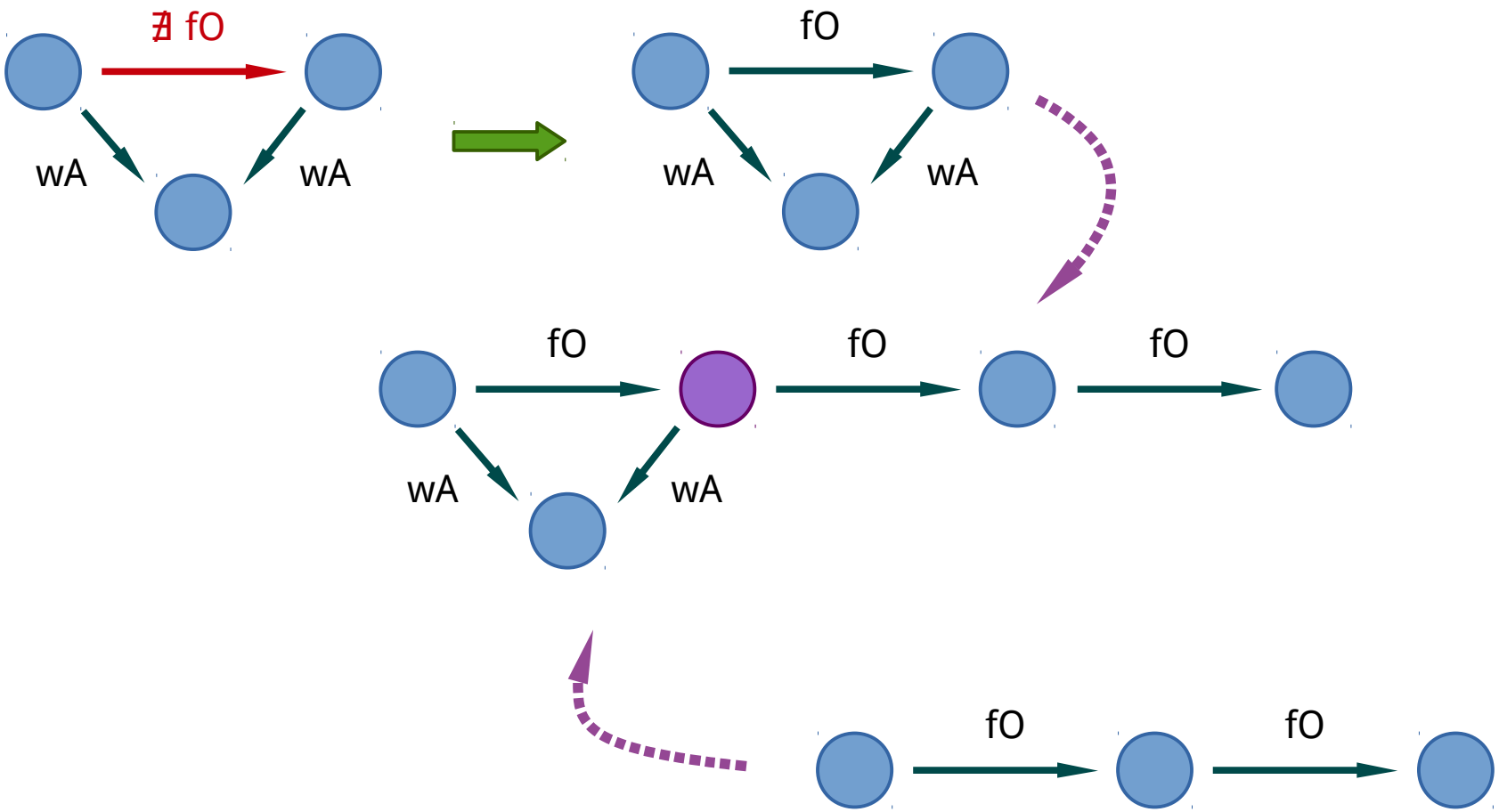
Observations

- dependencies correspond to overlaps
- overlaps identify partial query results

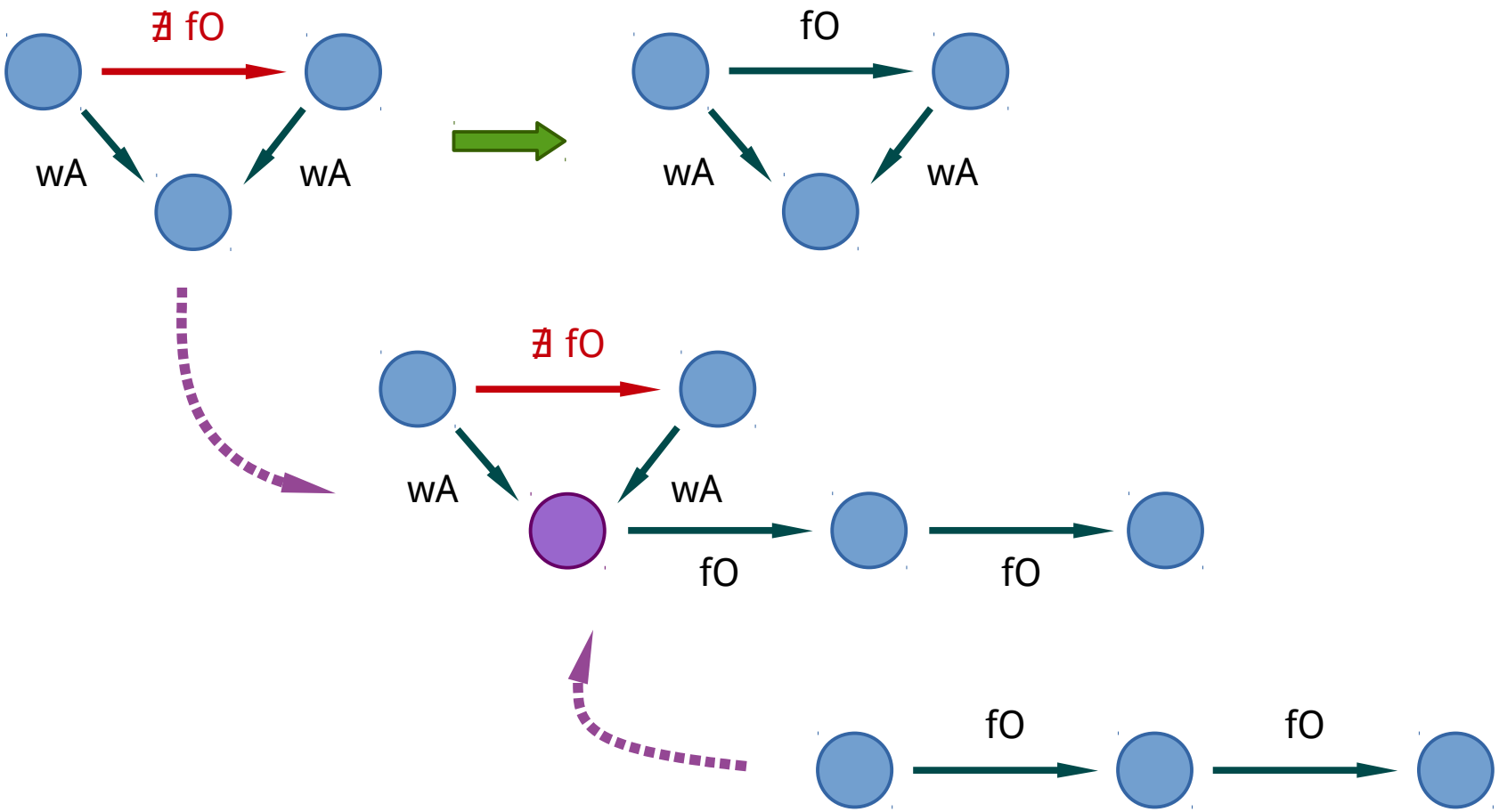
Dependency analysis

- Idea: find overlaps of queries and updates
  - Overlaps on left-hand sides → inhibition
  - Overlaps on right-hand sides → activation
  - Use overlaps as “seeds” for query engine
  - Basic theory dates back to the 70ies (GTS, parallel/sequential dependence, critical pairs, Rete networks)
- Issues…
  - There are lots of overlaps, most are irrelevant
  - How to deal with “external” updates?

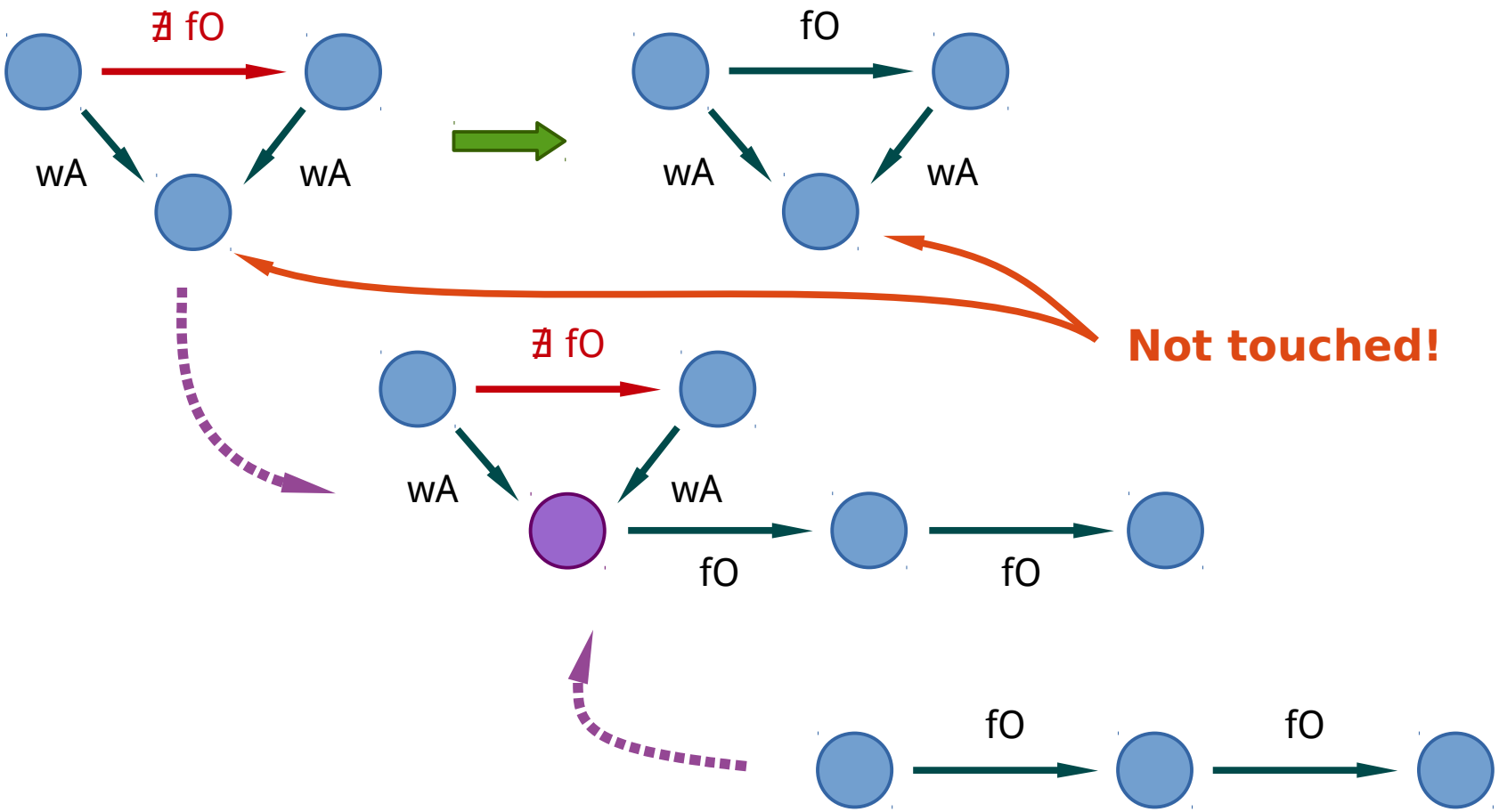
Irrelevant overlaps



Irrelevant overlaps



Irrelevant overlaps



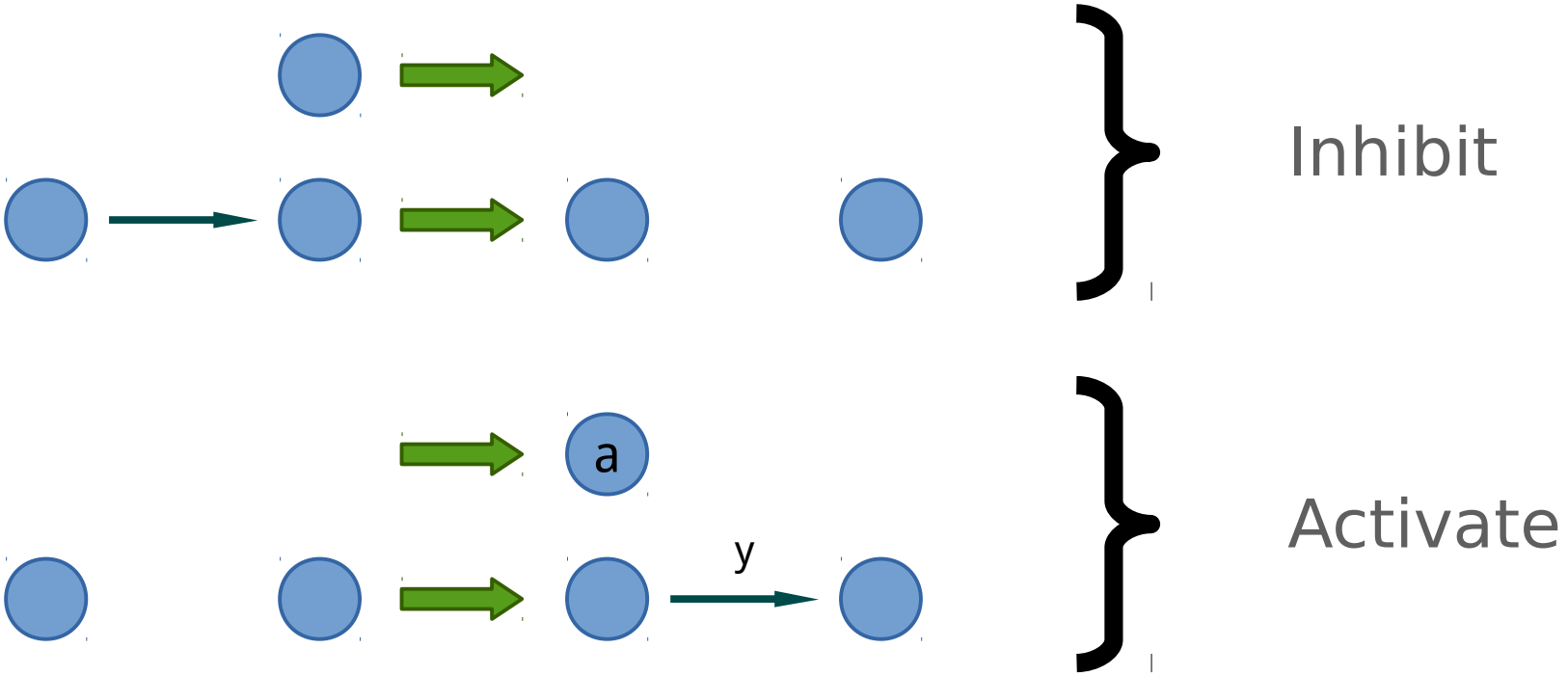
Improving dependency analysis

1. Only consider “relevant” overlaps

Basic updates

Observation

- treat updates as sequences of “atomic” updates

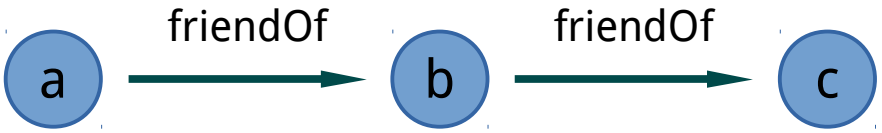


Improving dependency analysis

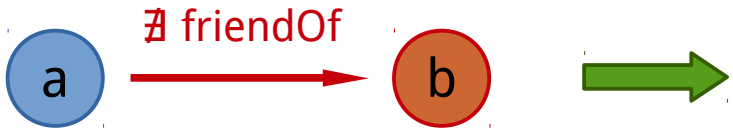
1. Only consider “relevant” overlaps
2. Analyze only basic updates
   - Break updates into basics and find overlaps
   - No dependency unless at least on basic update overlaps
   - Handles “external” updates too

Filter dependencies

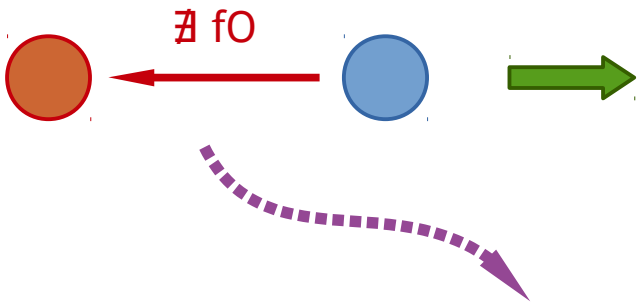
Example query

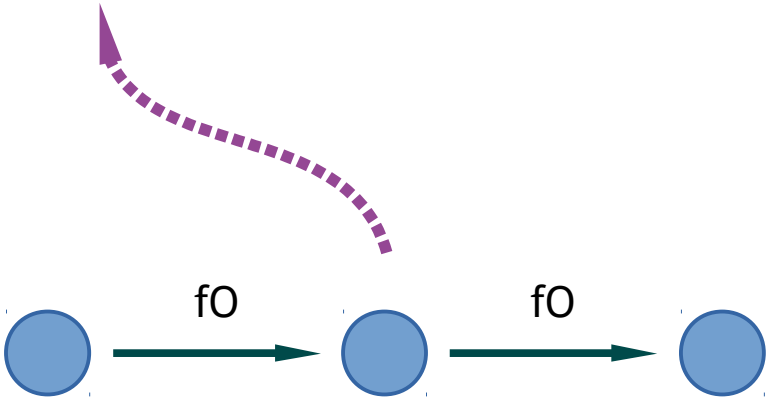


Example update

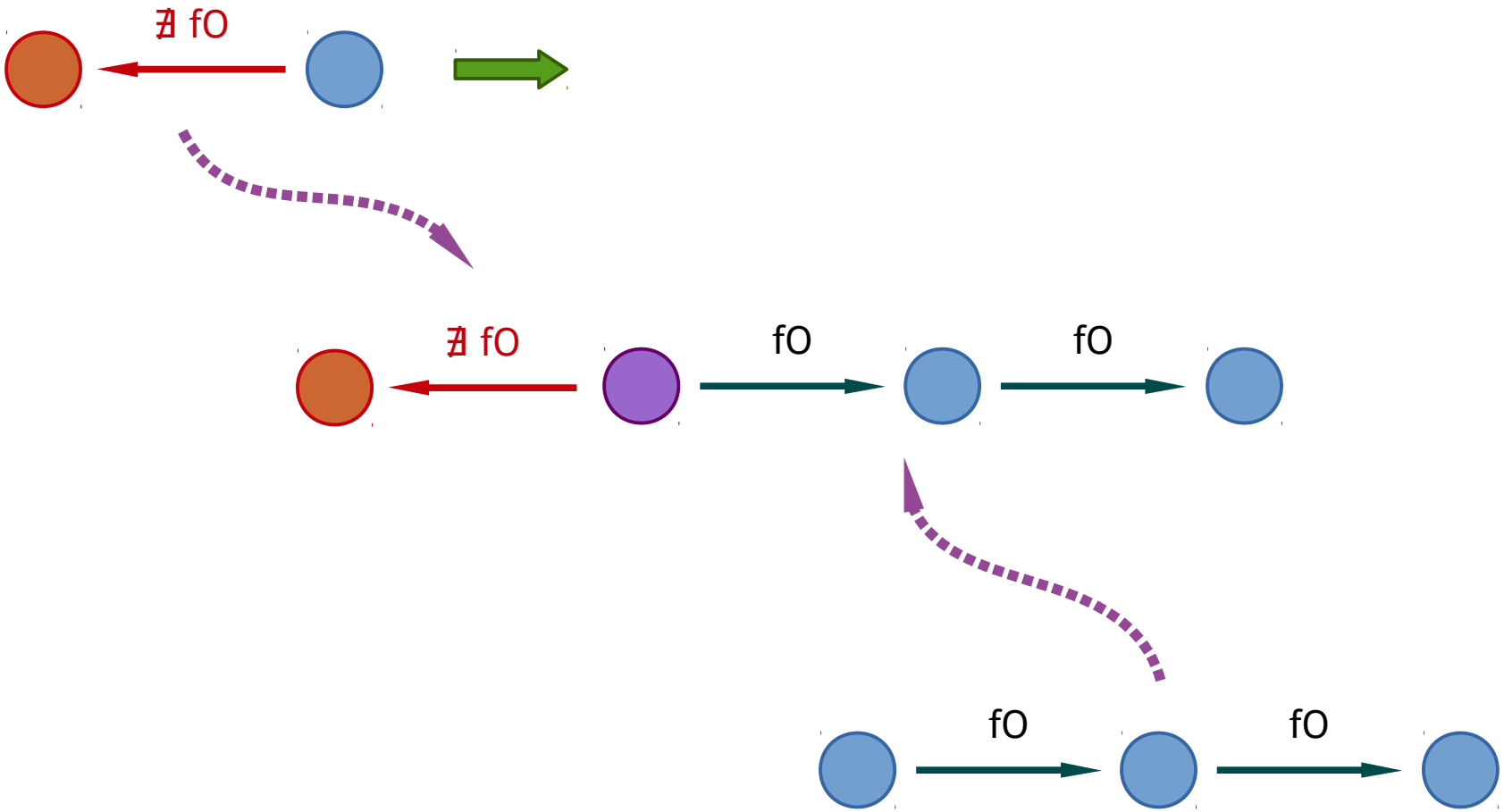


Filter overlaps

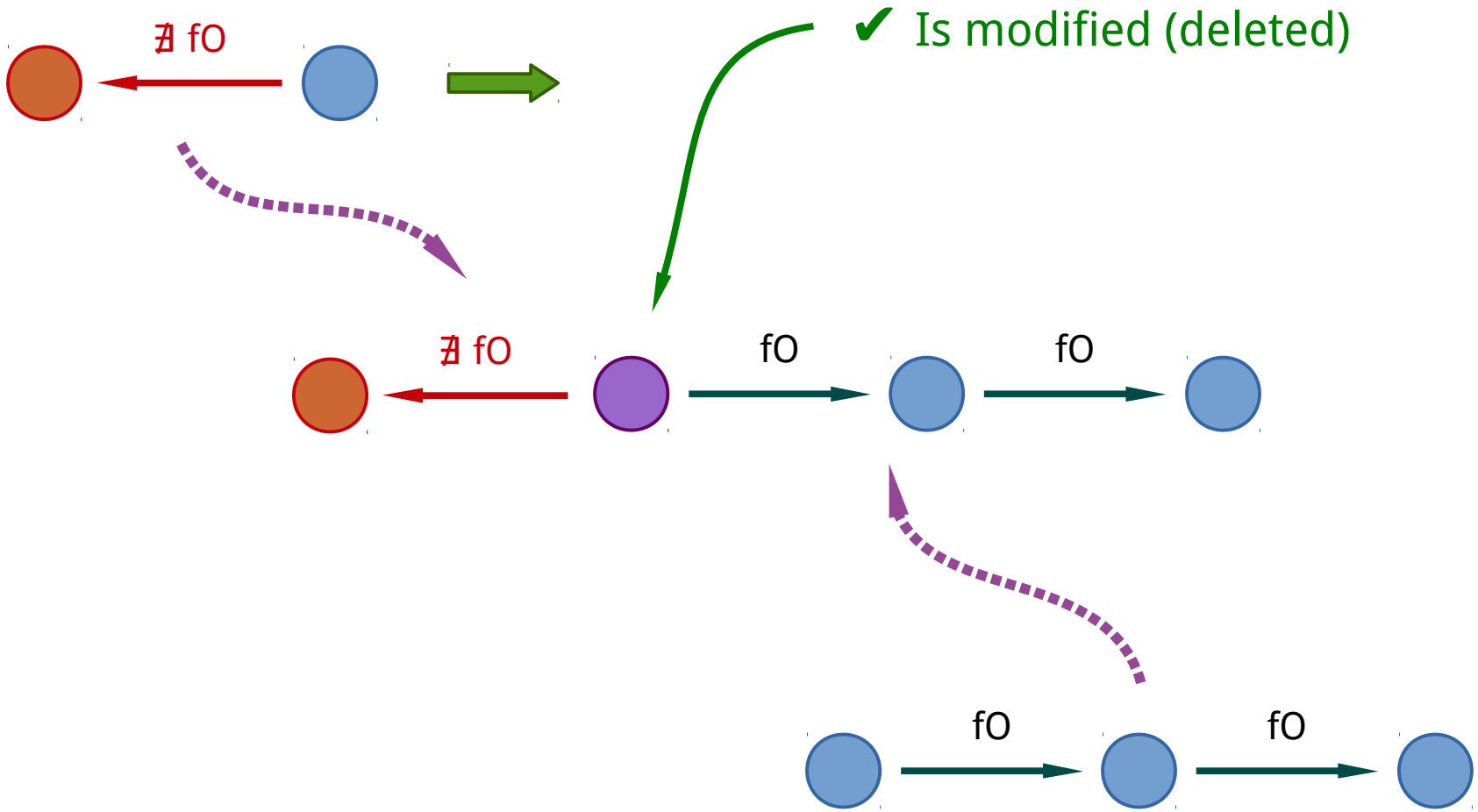




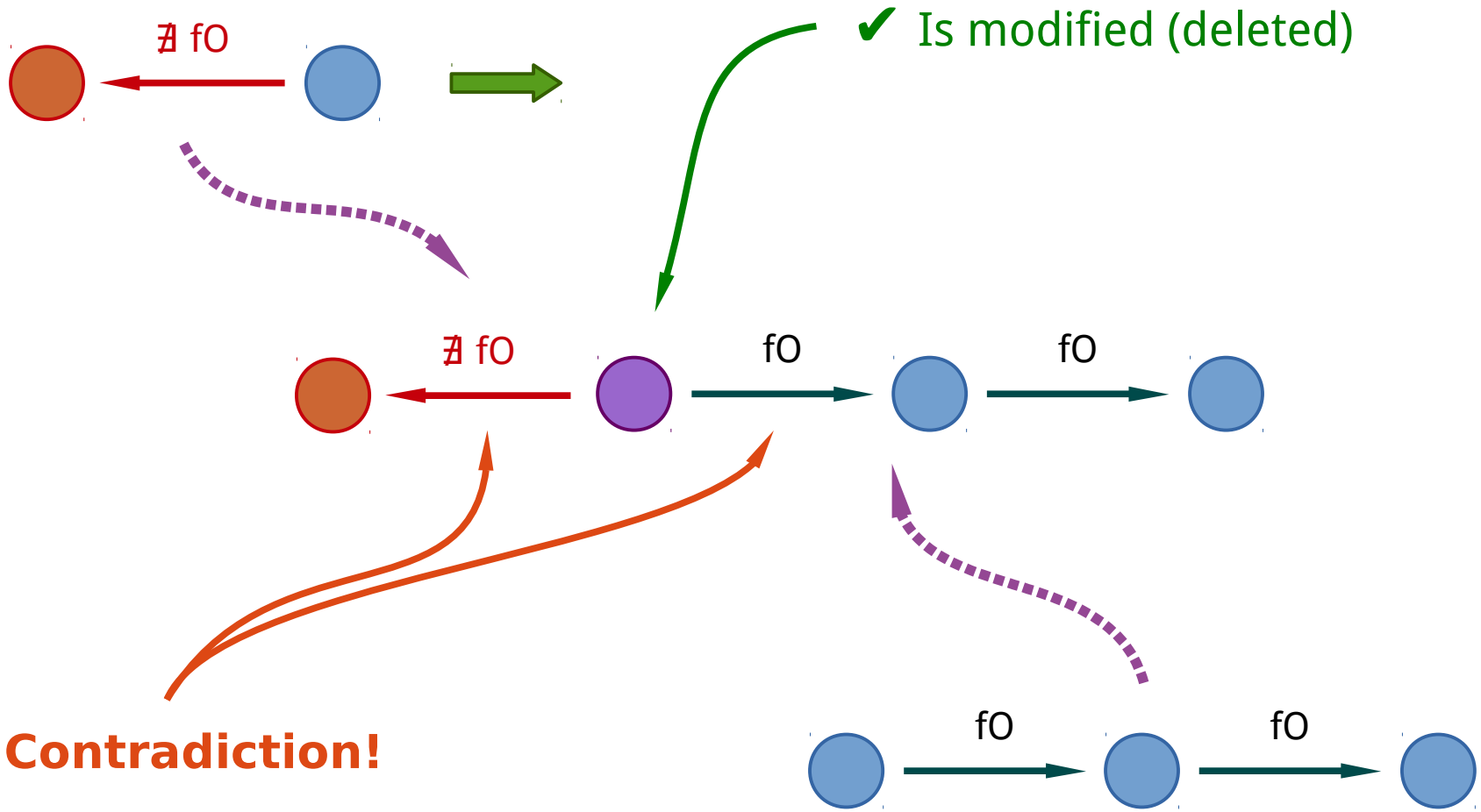
Filter overlaps



Filter overlaps



Filter overlaps



Improving dependency analysis

1. Only consider “relevant” overlaps
2. Analyze only basic updates
   - Break updates into basics and find overlaps
   - No dependency unless at least on basic update overlaps
   - Handles “external” updates too
3. Check consistency of filter overlaps
   - Expensive or even undecidable for nested filters
   - Check only to a given depth (over approximation)


ORACLE®