Entity Resolution in SERF

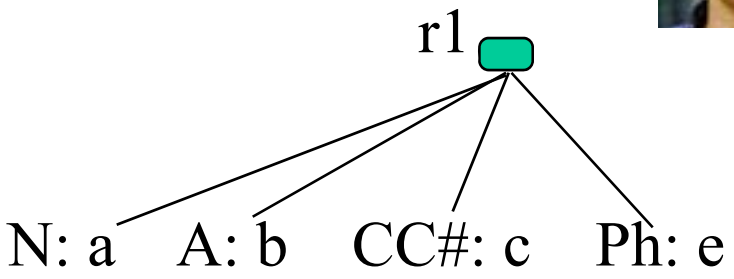
Omar Benjelloun

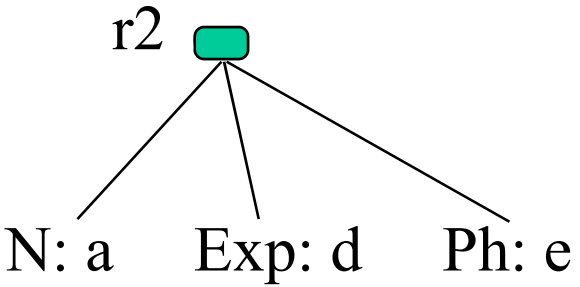
Stanford University

Joint work with:

Hector Garcia-Molina, Hideki Kawai, Tait E. Larson, David Menestrina, Qi Su, Sutthipong Thavisomboon, Jennifer Widom

Entity Resolution (ER)



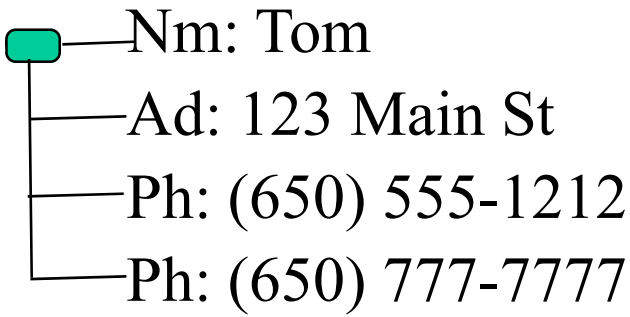


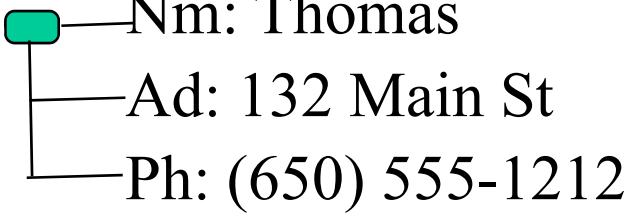
- Many applications:
  - customer files,
  - counter-terrorism,
  - comparison shopping...

- Aka: deduplication, record linkage, object co-identification, reference reconciliation, …

Challenges (1)

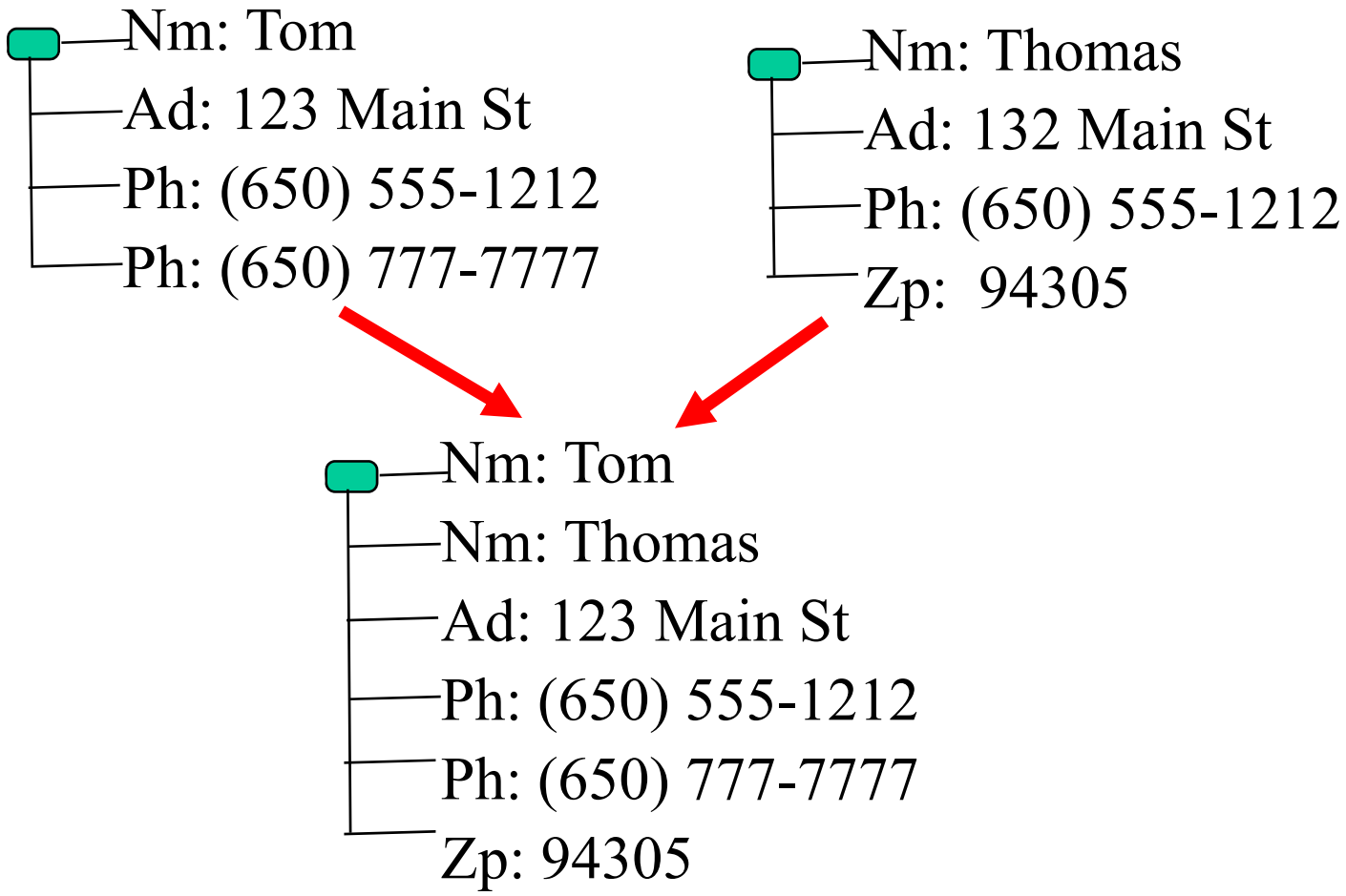
- No keys!
- Value matching
  - "Kaddafi", "Qaddafi", "Kadafi", "Kaddaffi"...
  - Many techniques developed
- Record matching





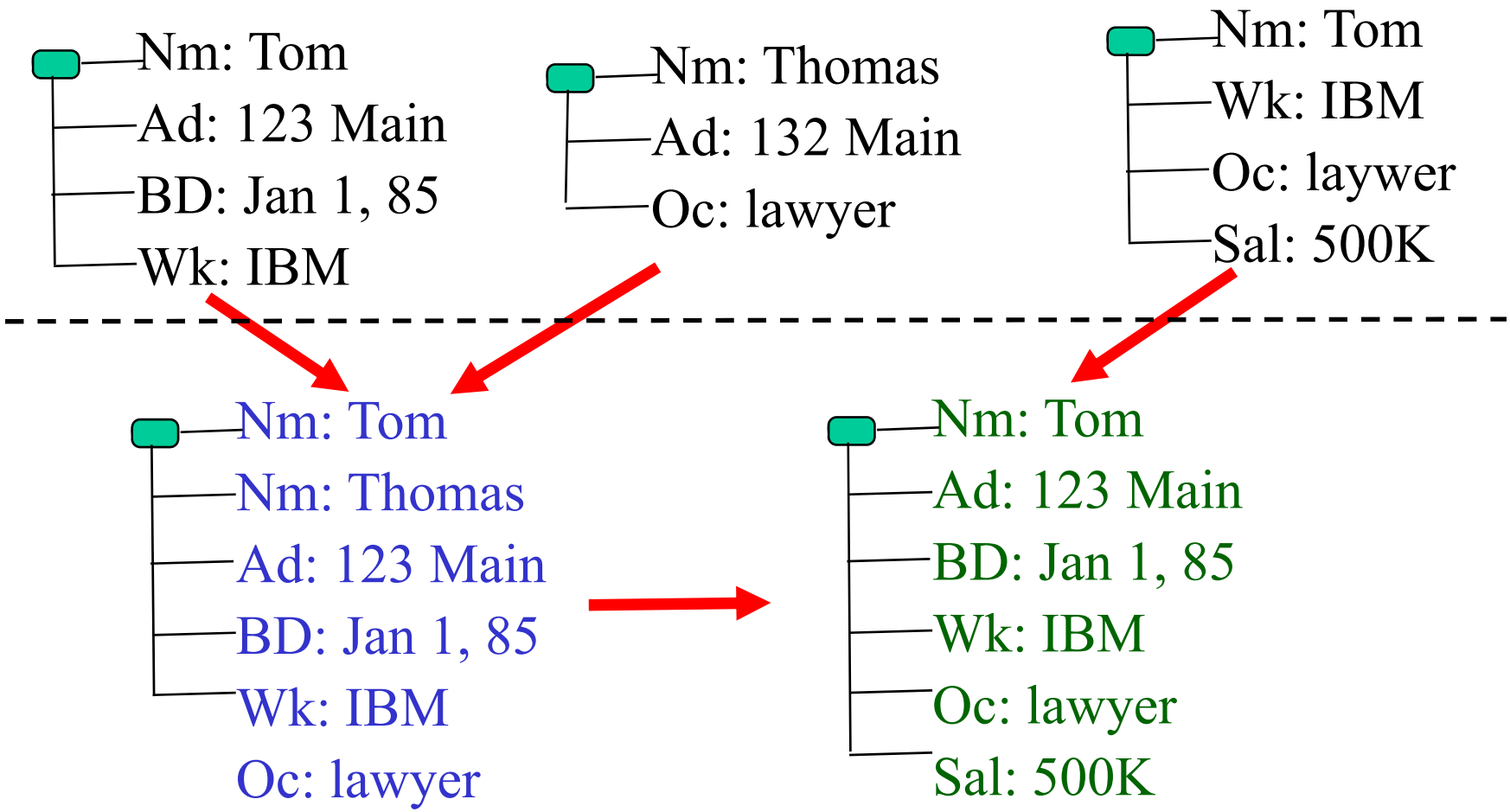
Challenges (2)

- Merging records



Challenges (3)

- Chaining



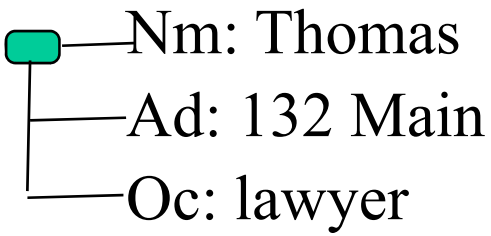
Generic Entity Resolution

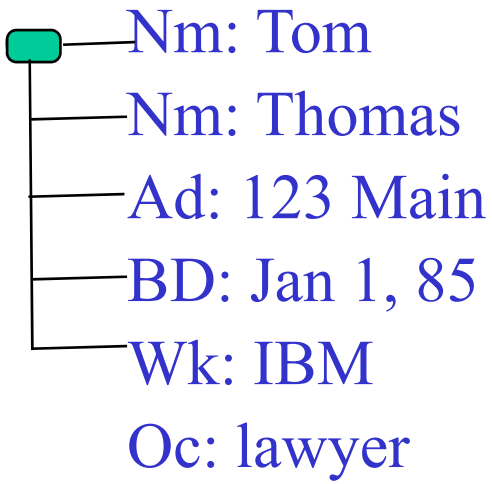
- Set of records: R (from domain $\mathcal{R}$)
- Match function: $\mathcal{R} \times \mathcal{R} \rightarrow \text{Boolean}$
  - M(r1,r2) = true if r1,r2 represent the same entity
- Merge function: $\mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$
  - r3 = <r1,r2> (exists if M(r1,r2)=true)

- We view match and merge as black boxes
- Focus on performance rather than accuracy

Domination

- Some records are less informative than others





- Record r1 is dominated by record r2 if <r1,r2>=r2
- Dominated records should be discarded

The Entity Resolution problem

- Given a set of records R,
  the Entity Resolution of R:
  - Has only records derived from R
  - Dominates all records derivable from R
  - Contains no matching or dominated records
- We provide simple and natural conditions to
  - Make ER "consistent" (finite and unique)
  - Enable efficient computation strategies

Conditions

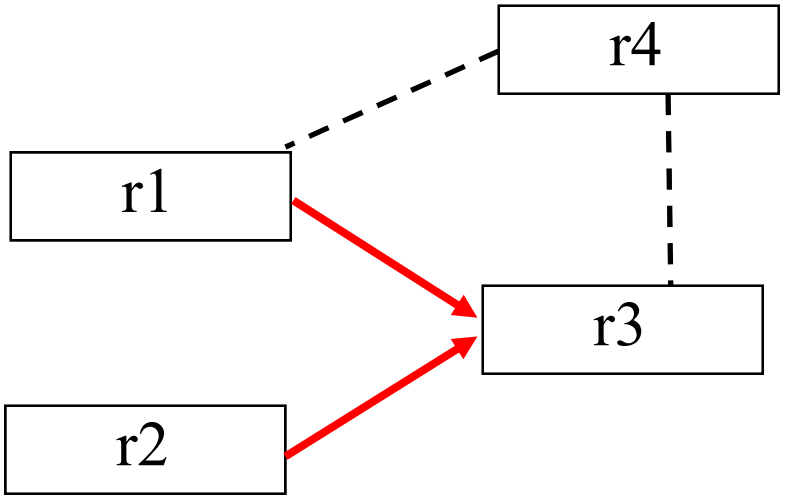
- Commutativity:
  - M(r1, r2) = M(r2, r1)
  - <r1, r2> = <r2, r1>
- Idempotence:
  - M(r1, r1) = true; <r1, r1> = r1
- Merge associativity:
  - <r1, <r2, r3>> = <<r1, r2>, r3> (if they exist)

Conditions (2)

- Representativity
  - r3 = <r1, r2>

    for any r4 such that M(r1, r4) = true

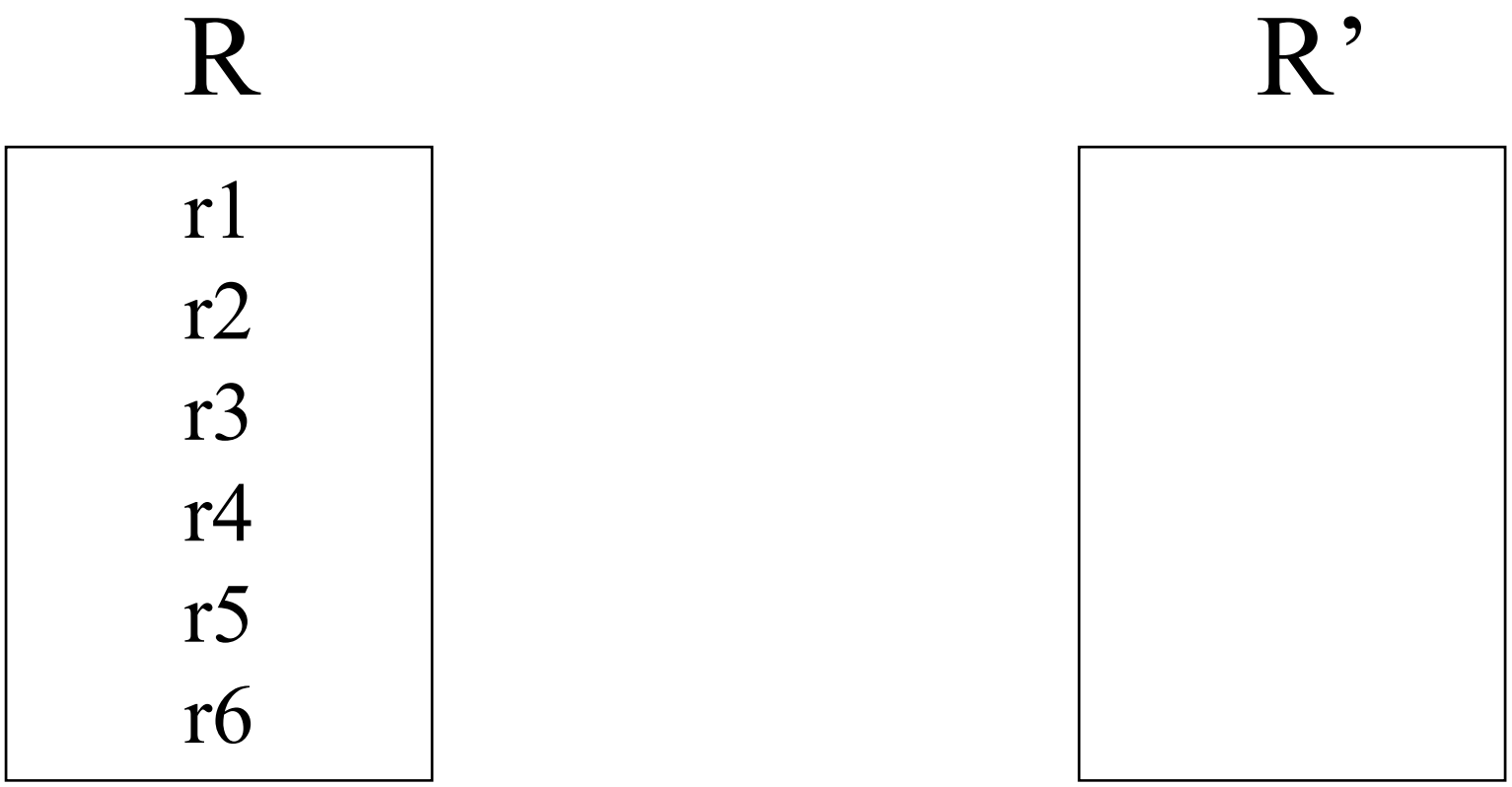
    we also have M(r3, r4) = true.



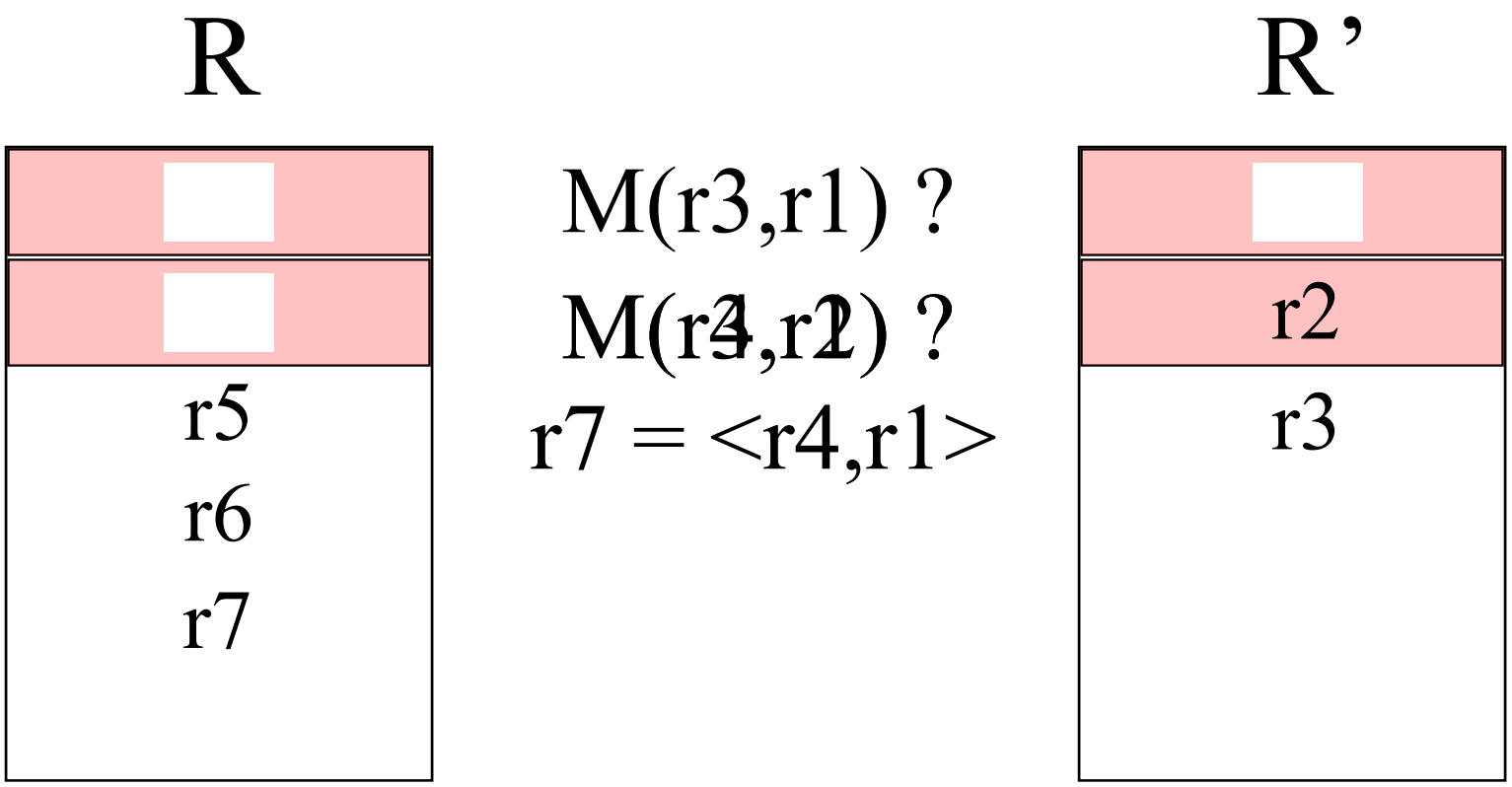
Algorithms

- These conditions enable flexible computation of ER(R)
  - Starting from R…
  - Find matches, add merged records
  - Find and delete dominated records
  - …in any order
- Optimal algorithm: R-Swoosh
  - Merges records and deletes dominated records early
  - No algorithm performs fewer record comparisons in the worst case

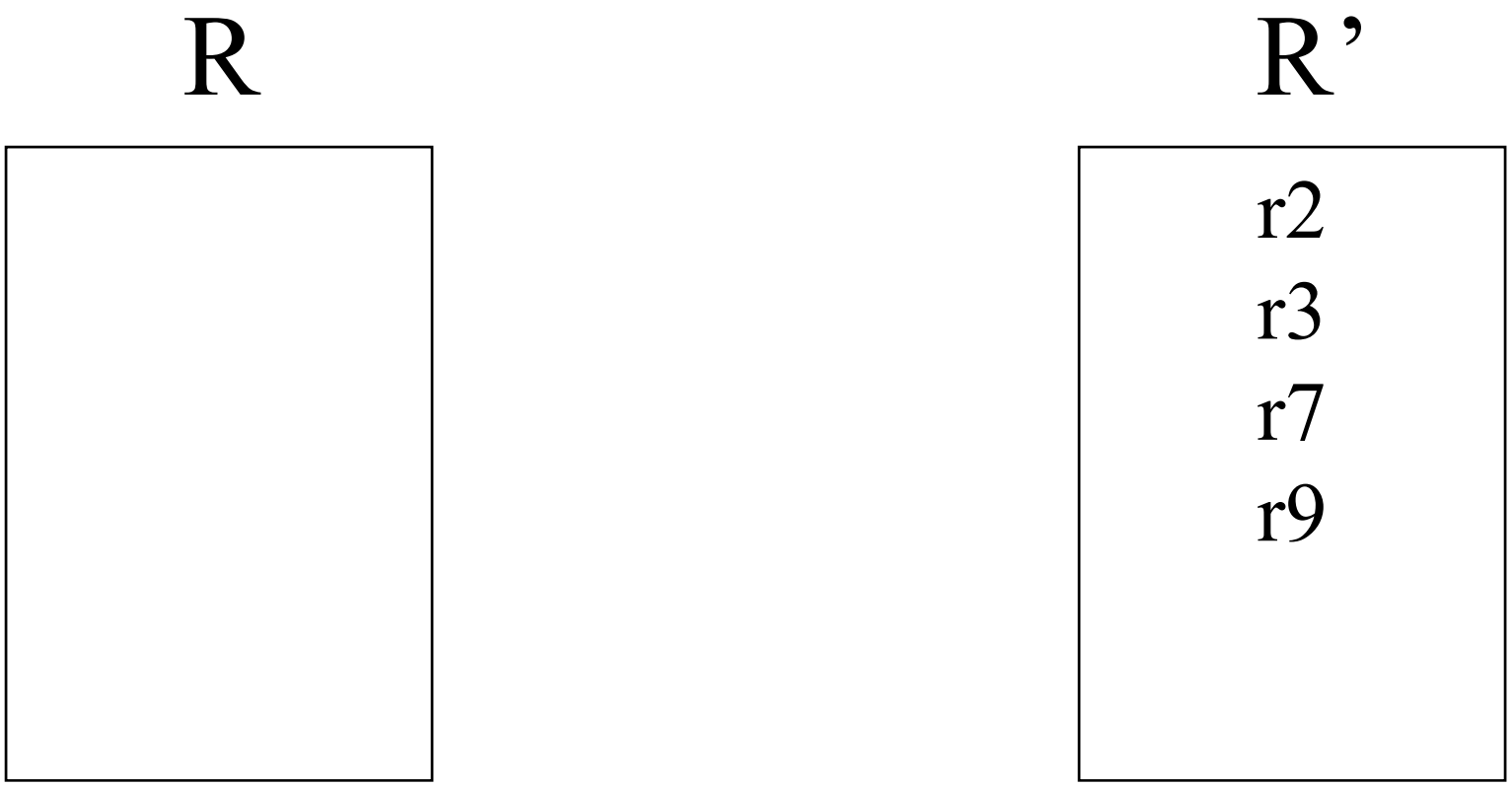
R-Swoosh



R-Swoosh



R-Swoosh



Also F-Swoosh, a variant that efficiently caches results of value comparisons

Example

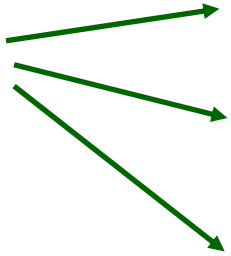
- [a: v1, b: w1]
- [a: v2, b: w2]
- [a: v3, b: w3]
- ...
- [a: vn, b: wn]

Match: M(ri, rj) = True
Merge: Union of values

answer: [a:{v1, ...,vn}, b:{w1, ..., wn}]

Naïve strategy

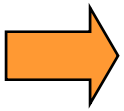
- [a: v1, b: w1]
- [a: v2, b: w2]
- [a: v3, b: w3]
- [a: v4, b: w4]



- [a:{v1,v2}, b:{w1,w2}]
- [a:{v1,v3}, b:{w1,w3}]
- [a:{v1,v4}, b:{w1,w4}]
- [a:{v2,v3}, b:{w2,w3}]
- [a:{v2,v4}, b:{w2,w4}]
- [a:{v3,v4}, b:{w3,w4}]

Naïve strategy (2)

- [a:{v1,v2}, ...]
- [a:{v1,v3}, ...]
- [a:{v1,v4}, ...]
- [a:{v2,v3}, ...]
- [a:{v2,v4}, ...]
- [a:{v3,v4}, ...]



- [a:{v1,v2,v3}, ...]
- [a:{v1,v2,v4}, ...]
- [a:{v2,v3,v4}, ...]
- [a:{v1,v2,v4}, ...]

- [a:{v1,v2,v3,v4}, ...]

... A lot of useless work!

R-Swoosh

- [a: v1, b: w1]
- [a: v2, b: w2]
- [a: v3, b: w3]
- [a: v4, b: w4]

- M(r1,r2) ®

  [a:{v1,v2}, ...]
- M(r3, r12) ®

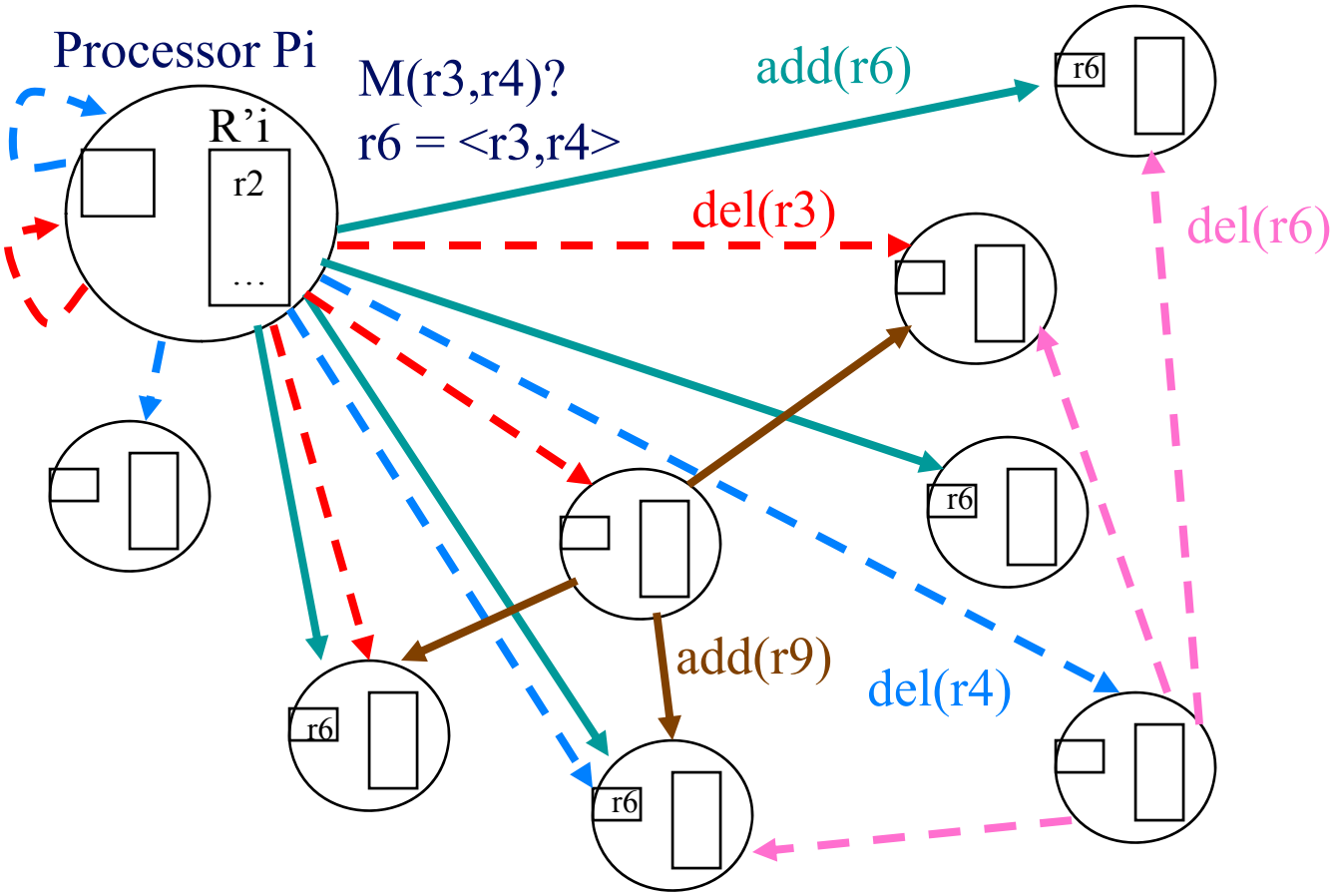
  [a:{v1,v2,v3}, ...]
- M(r4, r123) ®

  [a: v1, a: v2, a: v3, a: v4, ...]

Distributed ER

- ER is expensive:
  - Many records
  - Match comparisons are costly
- Distribute the work across multiple processors
  - Make sure no matches are missed
  - Minimize computation, communications and storage
- Use domain knowledge when available
  - E.g., DOB within 5 years, same product category

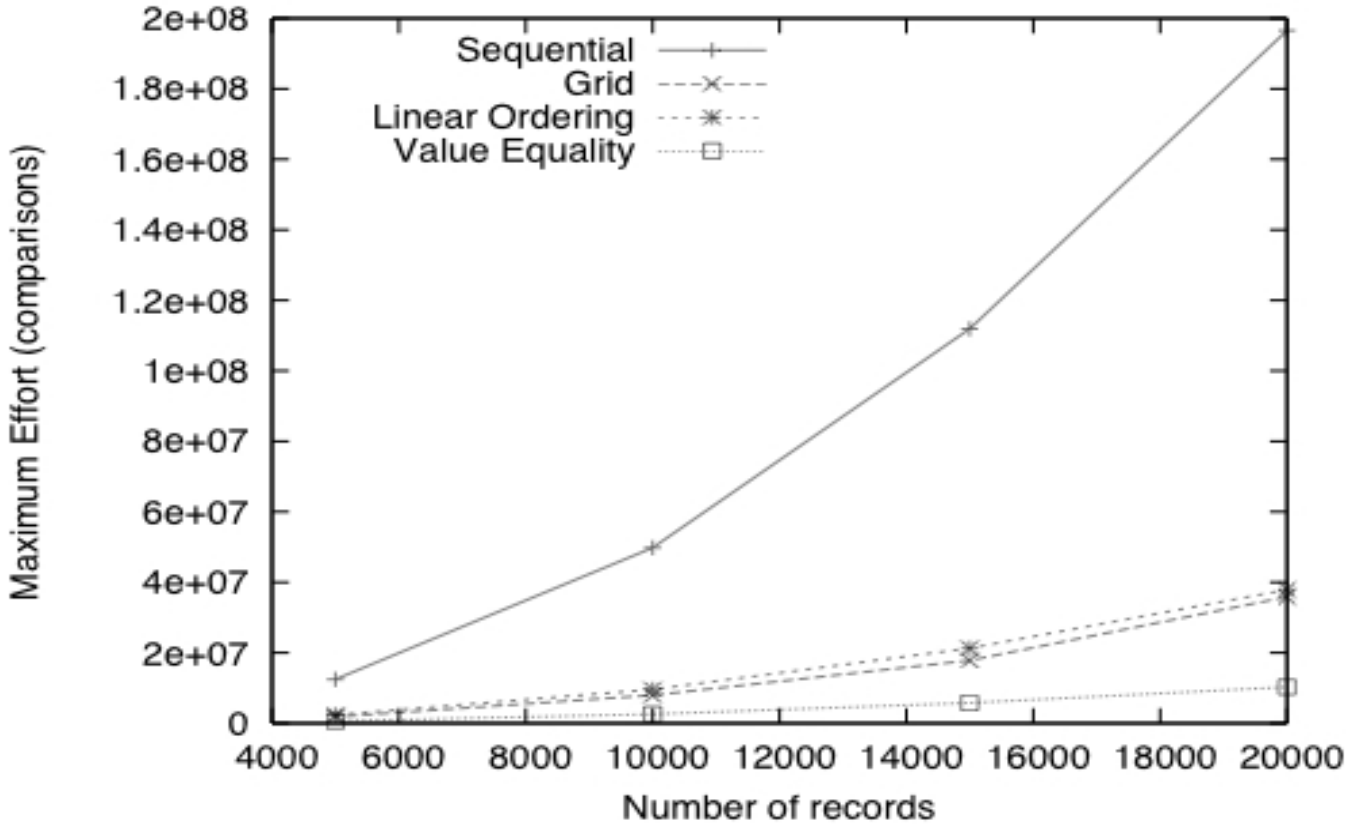
D-Swoosh


Processor Pi
M(r3,r4)?
r6 = <r3,r4>
R'i
r2
...
add(r6)
del(r3)
del(r6)
add(r9)
del(r4)
r6
r6
r6
r6

D-Swoosh

- Where to send records?
  - scope function (e.g., scope(r2)={P2,P5,P7})
- Who is responsible for comparisons?
  - resp predicate (e.g., resp(P6,r3,r5)= true)
- scope and resp must satisfy coverage property
  (related to mutual exclusion problem -- coteries)
- Schemes without domain knowledge
  - Majority, grid
- Schemes with domain knowledge
  - Value equality, linear ordering, hierarchies

D-Swoosh performance



- Computation cost per processor (10 processors)
- Experiments on Yahoo! comparison shopping data

ER with confidences

- Each record has a "confidence" ($0 \leq c \leq 1$)
  - Not tied to specific interpretation (e.g., probabilistic)
  - Match function may exploit confidences
  - Merge function propagates confidences
- Some conditions do not hold anymore:
  - Representativity: Confidence decreases with merges
  - Associativity: Different derivations produce different confidences
- More costly algorithm is required (Koosh)
  - Optimizations: early detection of domination, thresholds

Summary

- Entity resolution is critical
- Generic approach yields reusable techniques
- Efficient resolution is important
- Currently working on
  - Large scale distributed ER
  - Negative information
  - Uncertainty and lineage in ER

Thank you.