Collaborative Data Access between Enterprise Clouds*

Krishna Kant

Center for Secure Information Systems

George Mason University

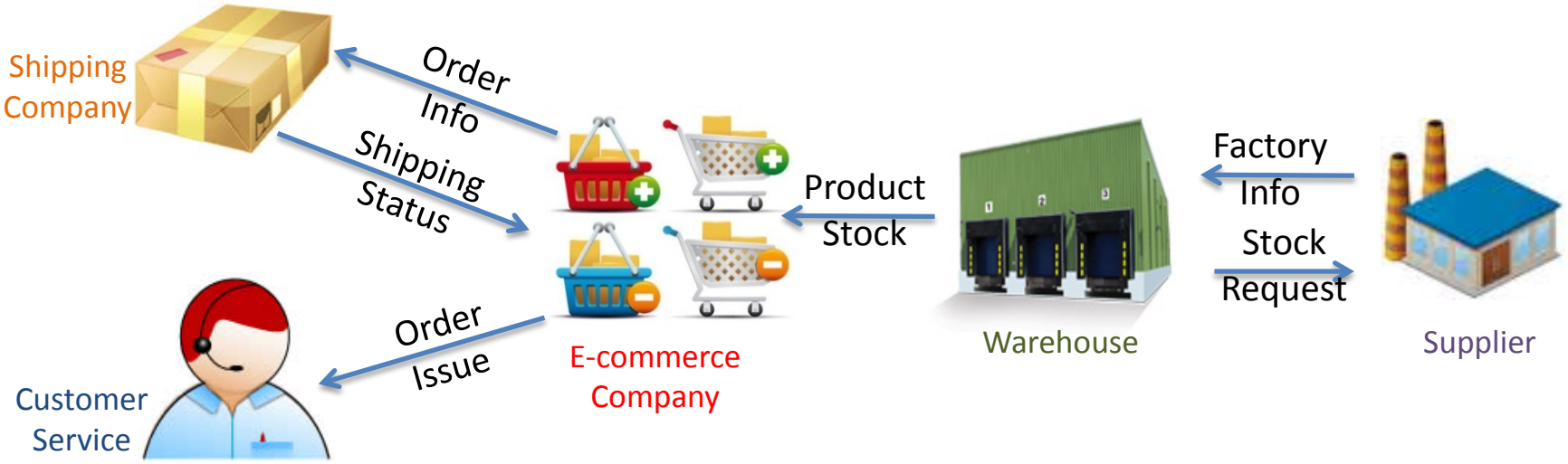
***With Meixing Le and Sushil Jajodia**

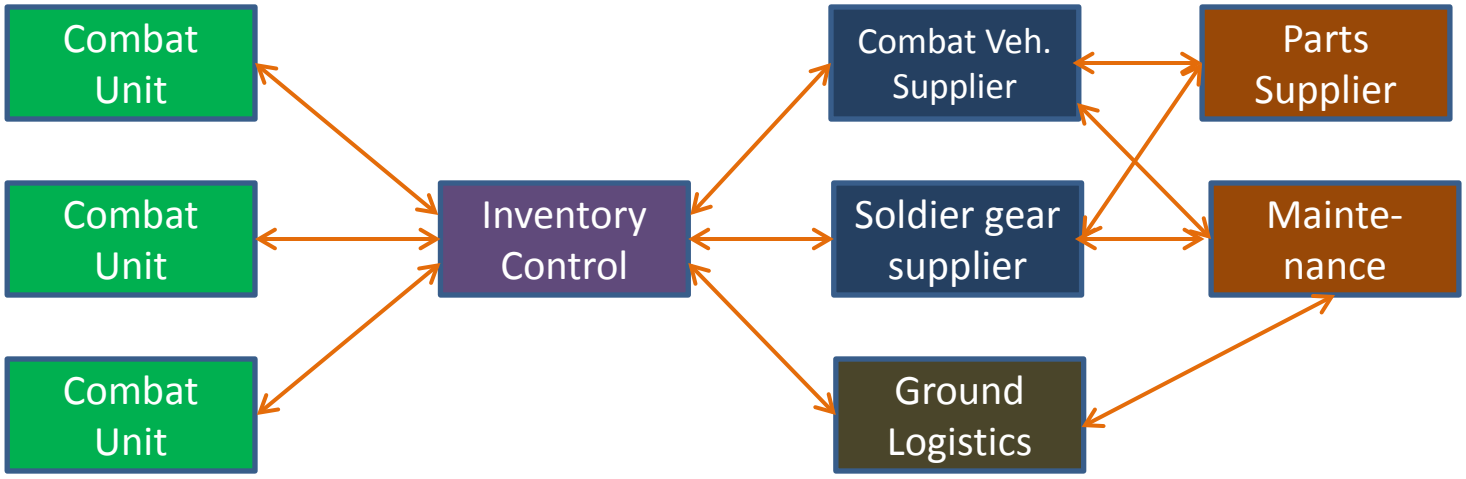
ARO Workshop on Cloud Computing Security

March 11-12, 2013

George Mason University, Fairfax, VA

Collaborative Sharing





Model

- A set of parties, each "owns" some data.
  - Data: Relations in standard (BCNF) form.
  - Access rules: Mutually agreed & visible to all parties
  - Rules: Joins & projections, but no selections.
    - Assuming joins only on key attributes
- Trust Model
  - No subversion of access rules or malicious query processing.
  - Accessible data may be manipulated further (e.g., further joins)
  - Cloud may not be entirely trusted (future)

Problems

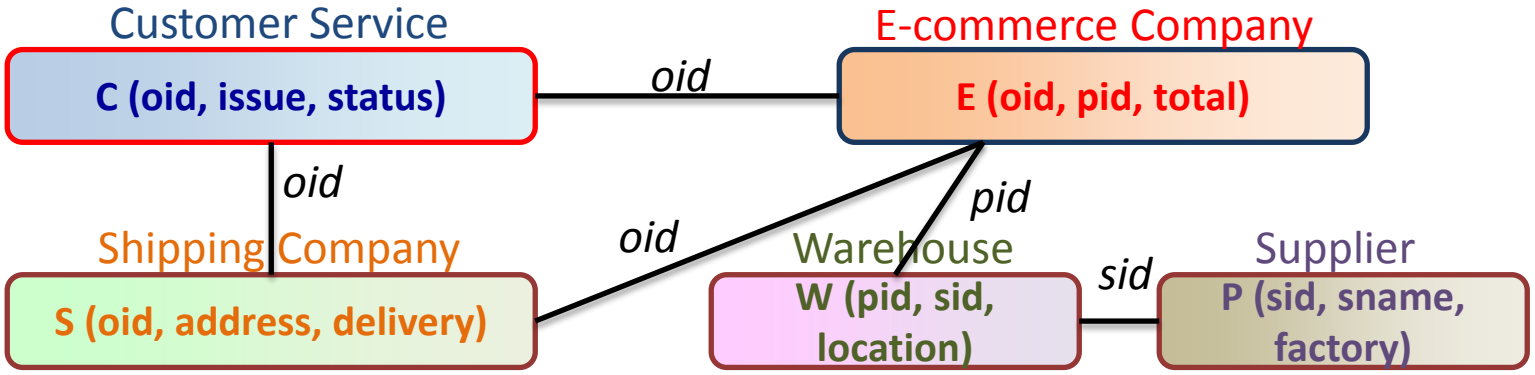
- Consistency [Le1'12]
  - Some data accessible but not explicitly authorized
- Enforceability
  - Access explicitly authorized, but not possible
  - Enforcement via third parties
- Query planning with collaborative access [Le2'12]
- Rule Changes
  - Efficient checking of consistency and enforceability
- Other Issues
  - Trust, Granular access control, …

Related Work

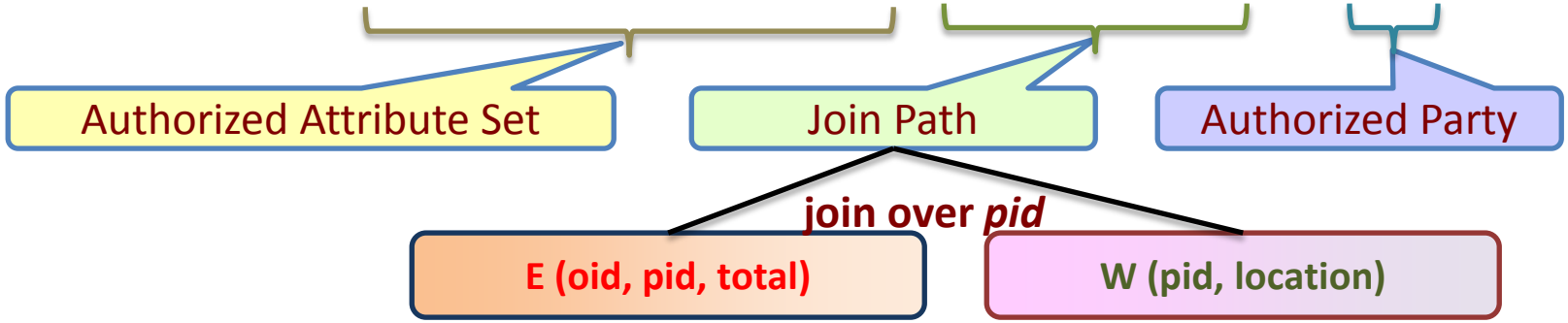
- The authorization model[D.C.Vimercati'08,CCS'08]
  - Similar model, but assumes rules consistency
- Classical distributed query processing [Kossmann'00]
  - Access constraints on data not addressed
- Distributed query processing under protections [Cali'08]
  - Binding patterns, no collaborative parties
  - Views based authorization[Goldstein'01, Halevy'01]
  - Conjunctive queries, do not consider rule consistency
- Collaborative access control[Tolone'05]
  - RBAC, different authorization models

A Running Example

- An e-commerce scenario with five parties



- An access rule is a triple[At, Jt, Pt]
  - E.g., {oid, pid, location}, (E ⋈pid W) -> PS



Example Rule Set

Rule No.	Authorized Attribute Set	Join Path	Party
1	{oid, pid, total}	E	P_E
2	{oid, issue, address}	S $\bowtie_{oid}$ **C**	P_E
3	{oid, pid, total, issue}	E $\bowtie_{oid}$ **C**	P_E
4	{oid, pid, sid, location, total}	E $\bowtie_{pid}$ **W**	P_E
5	{pid, sid, factory}	W $\bowtie_{sid}$ **P**	P_E

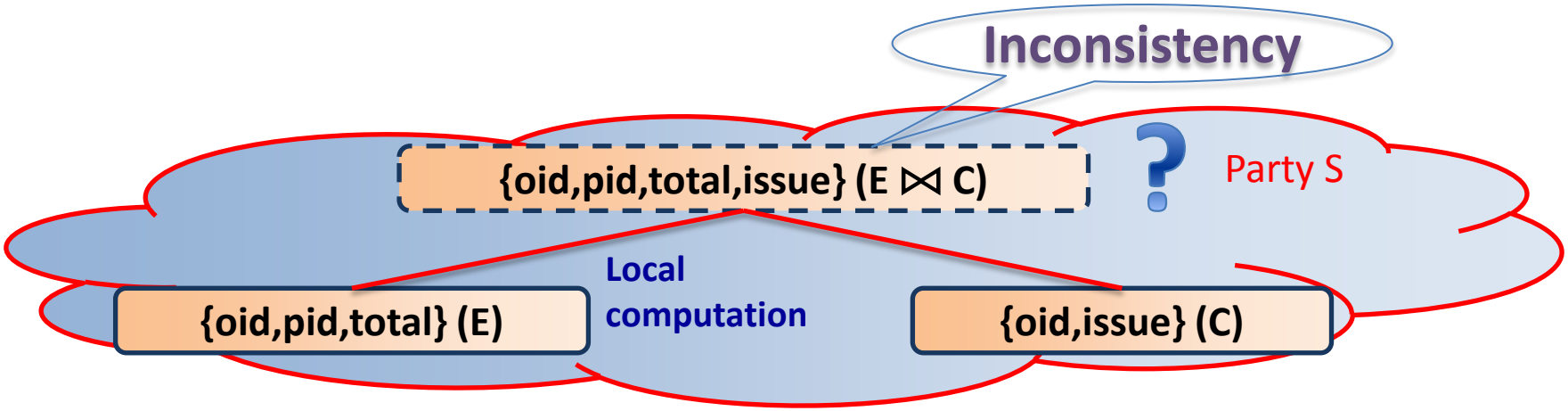
Example query authorized by rule r2:

SELECT *oid, issue* FROM JOIN(S, C) ON S.*oid* = C.*oid*

WHERE address = “Pittsburgh”

Supplier

Rule Inconsistency

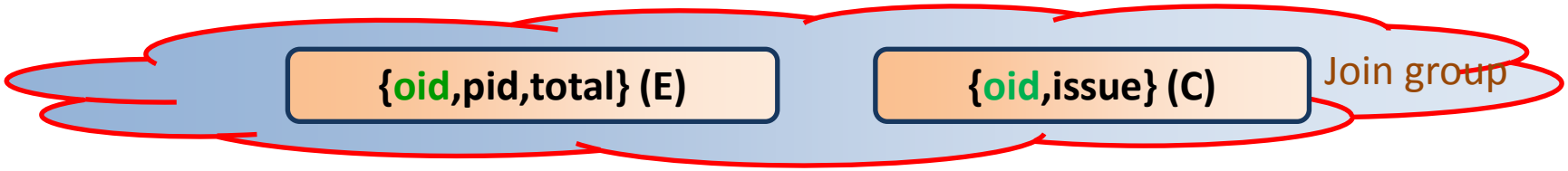


- Query
  - SELECT oid, total, issue FROM JOIN(E,C) ON E.oid = C.oid is allowed but not authorized
- Two key questions:
  - Discover all possible inconsistencies
  - Remove inconsistency by adding/removing rules or using third parties

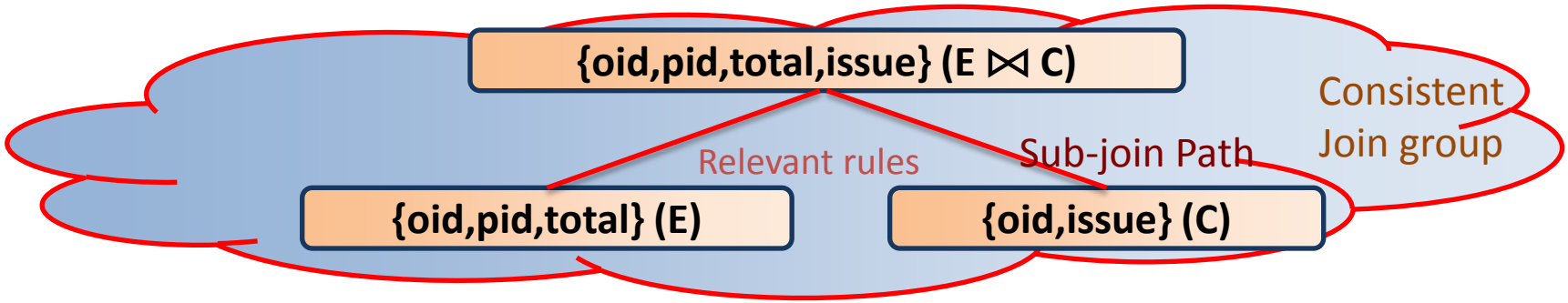
Consistency Checking

Join Group & Graph

- A group of Rules having identical key attributes



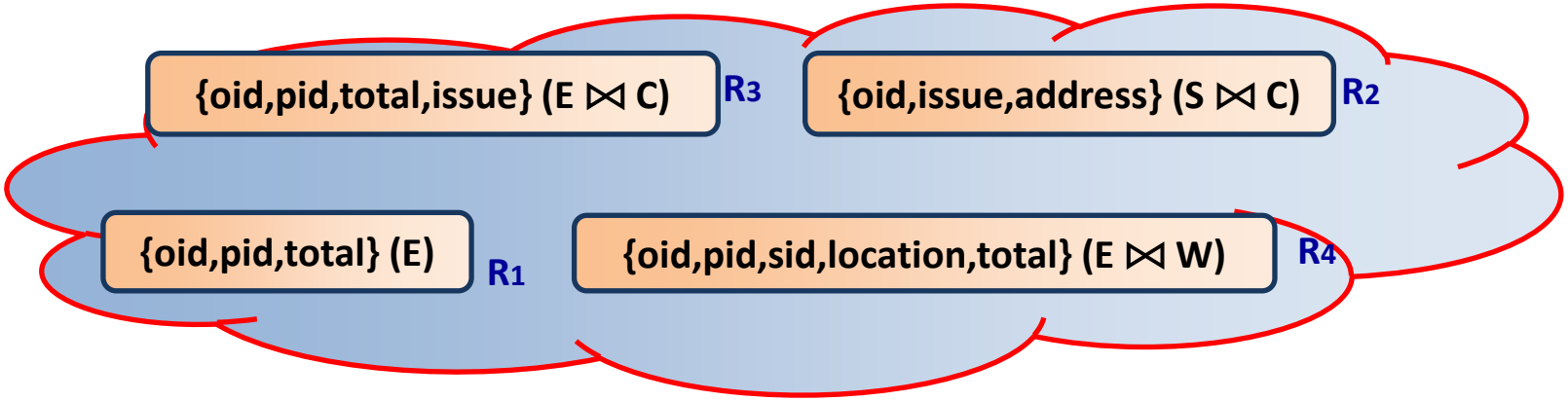
- Consistent Join Group (CJG): Closure of join group



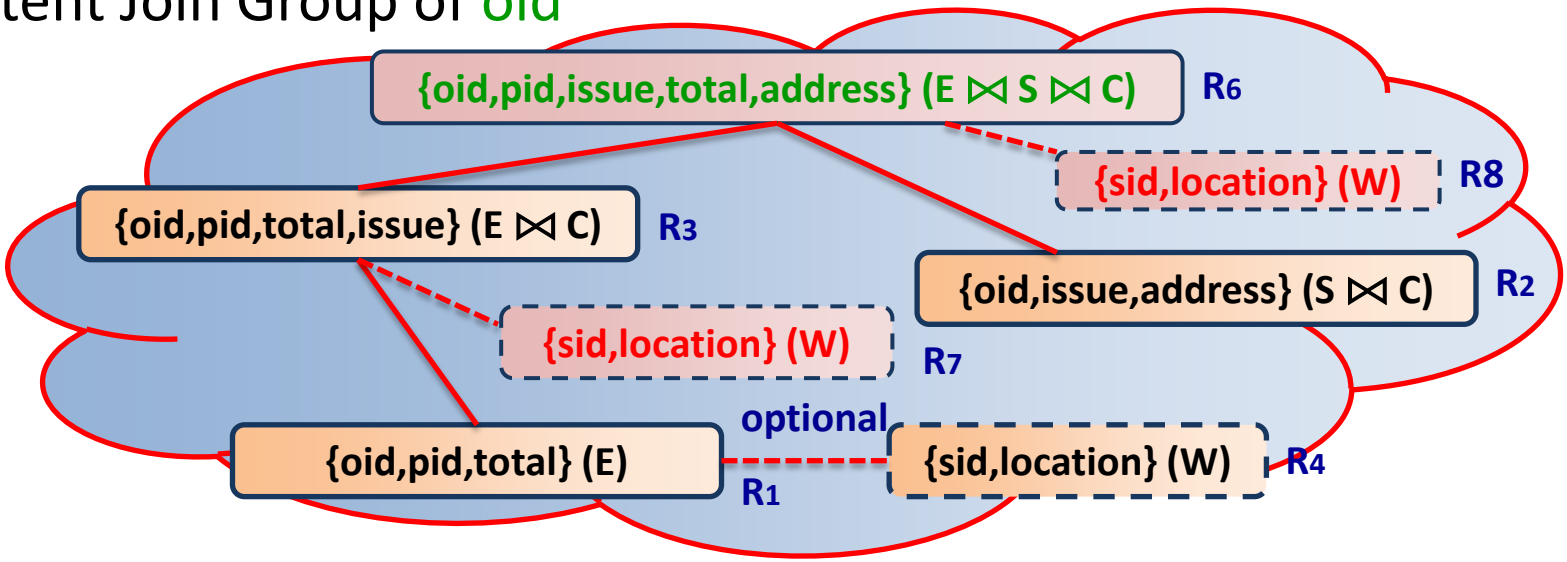
- A rule may be part of two distinct CGJ's
  - Rules are connected if they can be joined
  - Creates a join graph involving multiple CJG's

Generating CJG

Original Join Group of oid



Consistent Join Group of oid



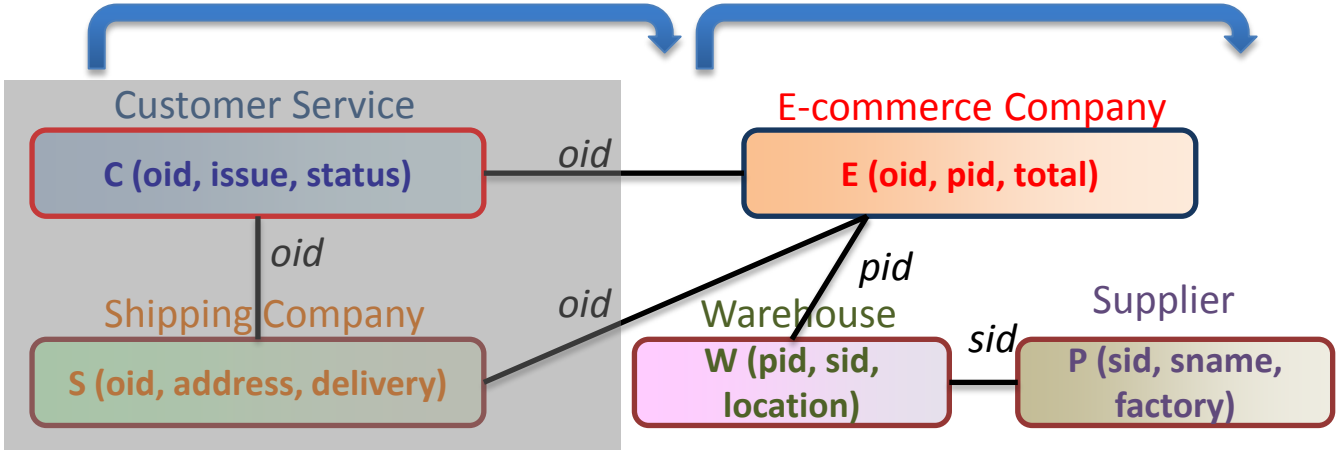
CJG of oid

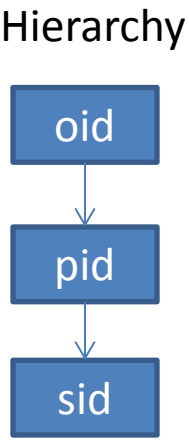
Rule No.	Authorized Attribute Set	Join Path	Party
1	{oid, pid, total}	E	P_E
2	{oid, issue, address}	S $\bowtie_{oid}$ C	P_E
3	{oid, pid, total, issue}	E $\bowtie_{oid}$ C	P_E
4	{oid, pid, sid, location, total}	E $\bowtie_{pid}$ W	P_E
5	{pid, sid, factory}	W $\bowtie_{sid}$ P	P_E
6	{oid, pid, total, issue, address}	E $\bowtie_{oid}$ S $\bowtie_{oid}$ C	P_E
7	{oid, pid, total, issue, location, sid}	C $\bowtie_{oid}$ E $\bowtie_{pid}$ W	P_E
8	{oid, pid, total, issue, location, sid, address}	S $\bowtie_{oid}$ C $\bowtie_{oid}$ E $\bowtie_{pid}$ W	P_E

Key Attributes Hierarchy

oid	total	pid	location	sid	sname
3	248.23	56	NY	23	Alpha
5	854.21	41	CA	12	Sigma

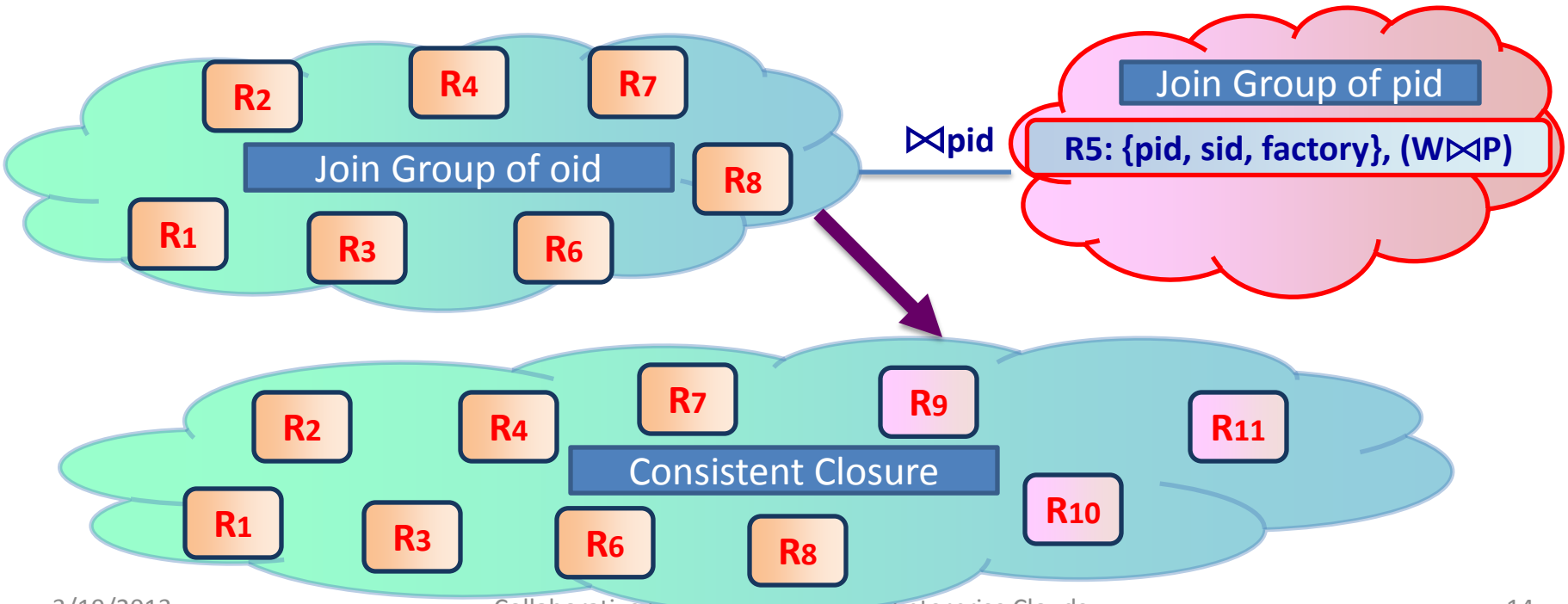
- Join attributes are key attributes of basic relations
  - Key of a join path is a key attribute of a relation
  - Key attributes of the relations form hierarchy in a join path





Iteration over Key Attributes

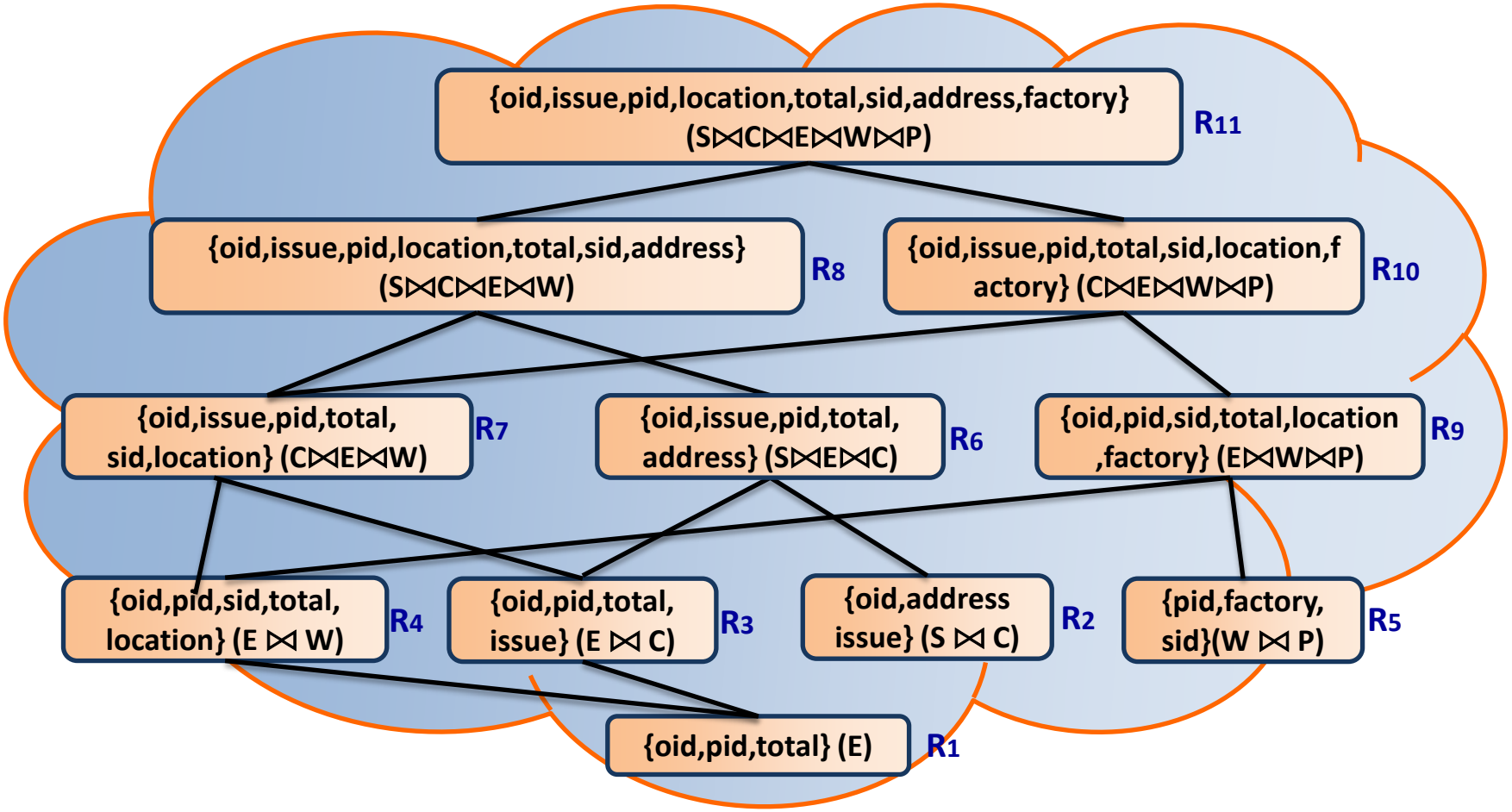
- Iterate join groups based on key hierarchy
- Add generated rules into Target rule set
  - Check if the rule includes the key of the next join group
  - If so, add the generated rules or merge with existing rules



Consistent Rule Closure

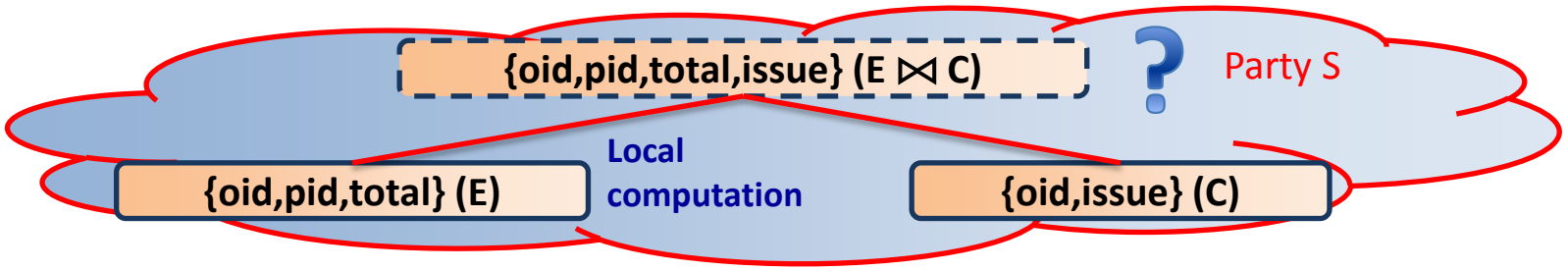
Rule No.	Authorized Attribute Set	Join Path	Party
1	{oid, pid, total}	E	P_E
2	{oid, issue, address}	S $\bowtie_{oid}$ C	P_E
3	{oid, pid, total, issue}	E $\bowtie_{oid}$ C	P_E
4	{oid, pid, sid, location, total}	E $\bowtie_{pid}$ W	P_E
5	{pid, sid, factory}	W $\bowtie_{sid}$ P	P_E
6	{oid, pid, total, issue, address}	E $\bowtie_{oid}$ S $\bowtie_{oid}$ C	P_E
7	{oid, pid, total, issue, location, sid}	C $\bowtie_{oid}$ E $\bowtie_{pid}$ W	P_E
8	{oid, pid, total, issue, location, sid, address}	S $\bowtie_{oid}$ C $\bowtie_{oid}$ E $\bowtie_{pid}$ W	P_E
9	{oid, pid, sid, factory, location, total}	E $\bowtie_{pid}$ W $\bowtie_{sid}$ P	P_E
10	{oid, pid, total, issue, sid, location, factory}	C $\bowtie_{oid}$ E $\bowtie_{pid}$ W $\bowtie_{sid}$ P	P_E
11	{oid, pid, total, issue, location, sid, factory, address}	S $\bowtie_{oid}$ C $\bowtie_{oid}$ E $\bowtie_{pid}$ W $\bowtie_{sid}$ P	P_E

Closure Graph



So, for consistency, we need to add several new rules. Is that desirable?

Respecting Inconsistency

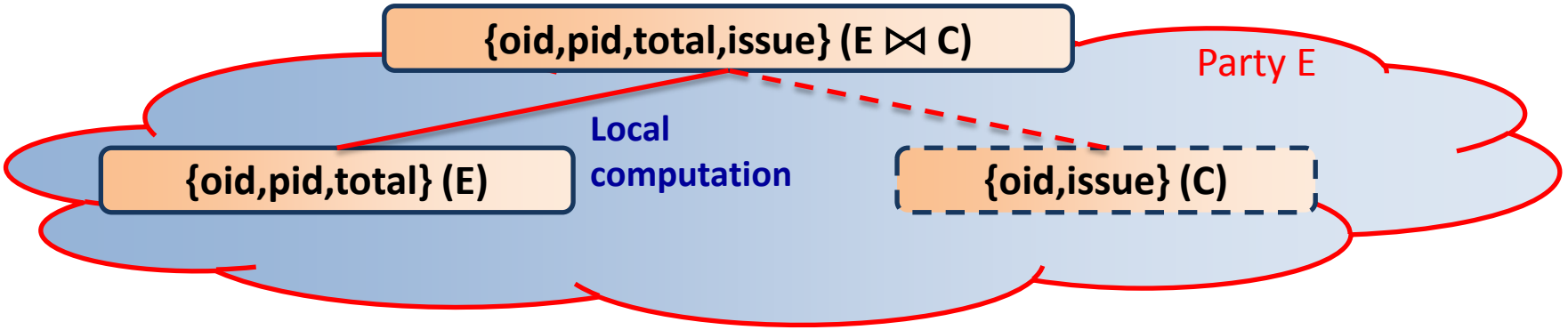


- Is access to E ⋈ C okay?
  - Yes: The party already has access to both E & C
  - No: Association between E & C is more sensitive than either.
- Potential resolution
  - Limit #tuples for queries➔ Limited leakage of E ⋈ C
  - Queries involving C or E ⋈ C executed via a 3rd party

Enforceability Checking

Rule Enforceability

- A given authorization rule is not necessarily enforceable



- No access to C ➔ Cannot do join ➔
  - Queries involving E ⋈ C are authorized but cannot be answered
- Different from consistency

Rule Enforcement checking

- Recursion wrt join path length (JPL)
  - JPL=1 ➔ Rules always enforceable
  - At step JPL=n
    - Path enforcement: Can we generate the desired join path? E.g., (E ⋈ C)
    - Attribute enforcement: Can we provide access to the desired attributes? E.g., {oid, pid}, (E ⋈ C)
- Cooperative enforcement
  - May need to transmit parts of relations to a party that has access but doesn't own it.

Cooperative Enforcement

- Steps
  - C can access {oid, pid}(E), but needs its transmission from E
  - Generates {oid,pid,issue} (E ⋈ C), and sent to E to generate the desired data

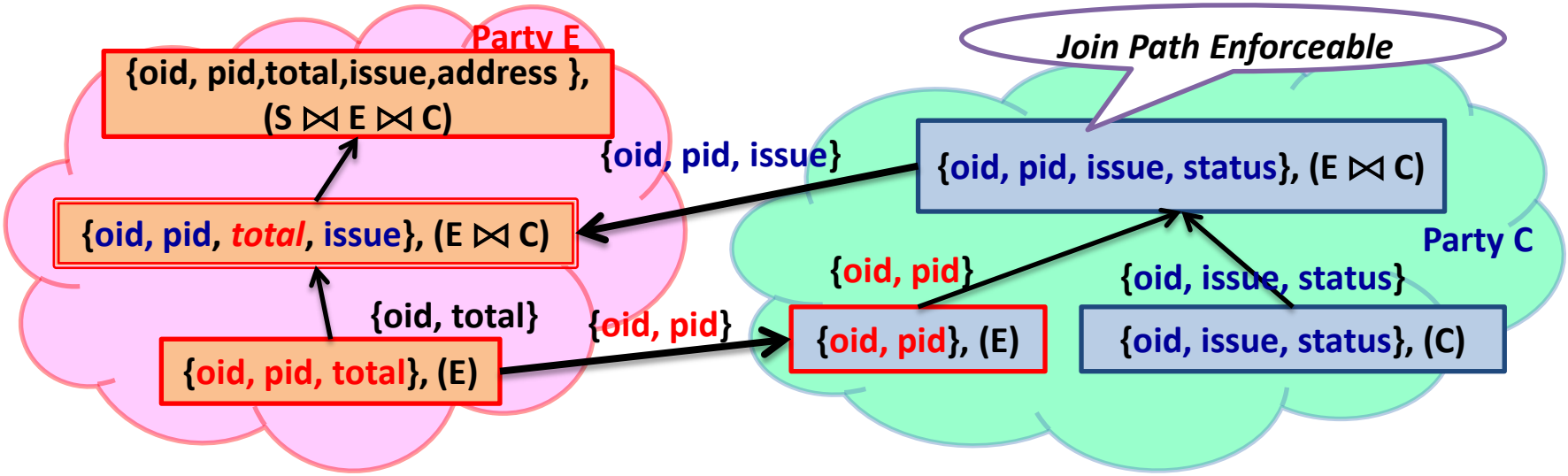
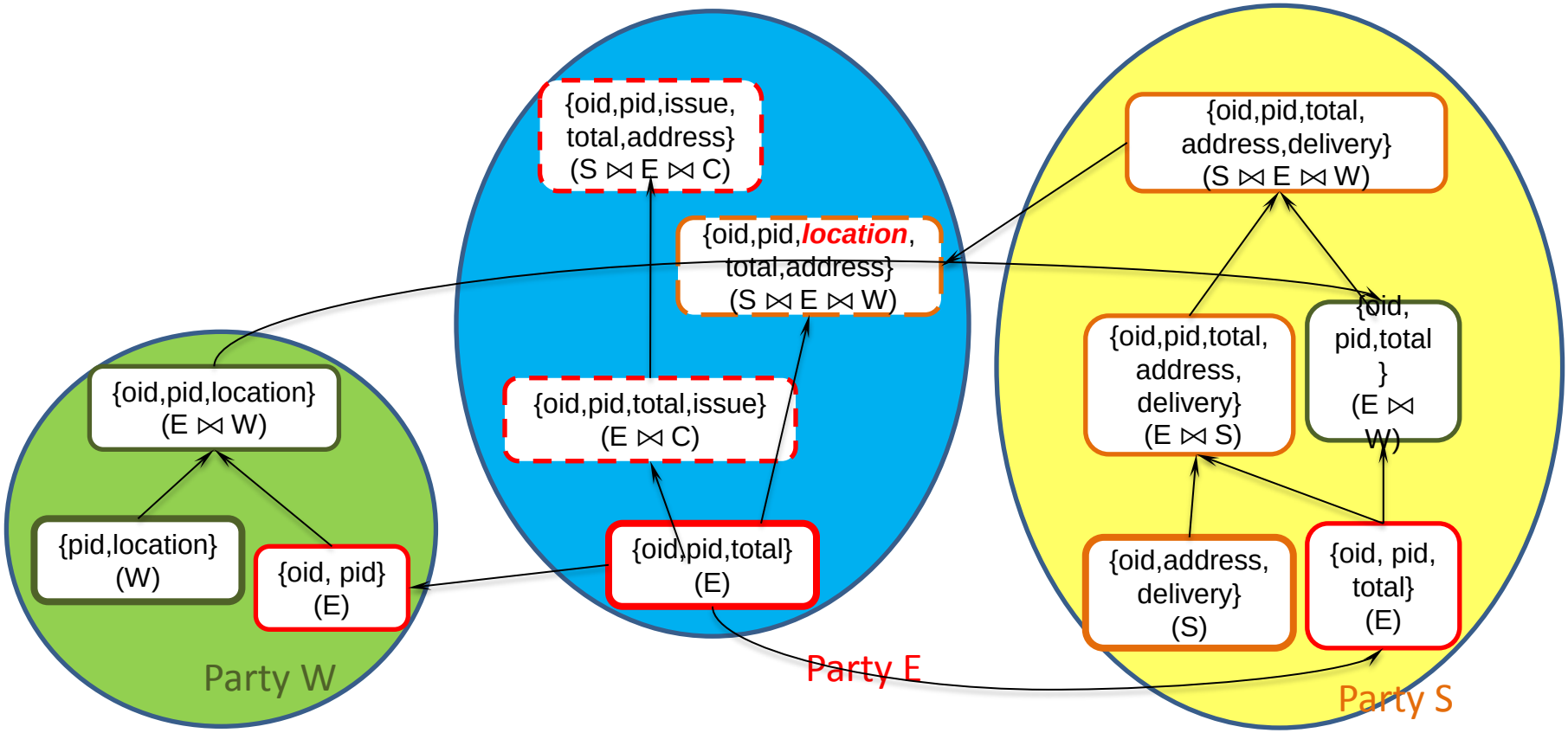


Illustration with example

- Start bottom up at each party
- Collaborative enforcement as needed

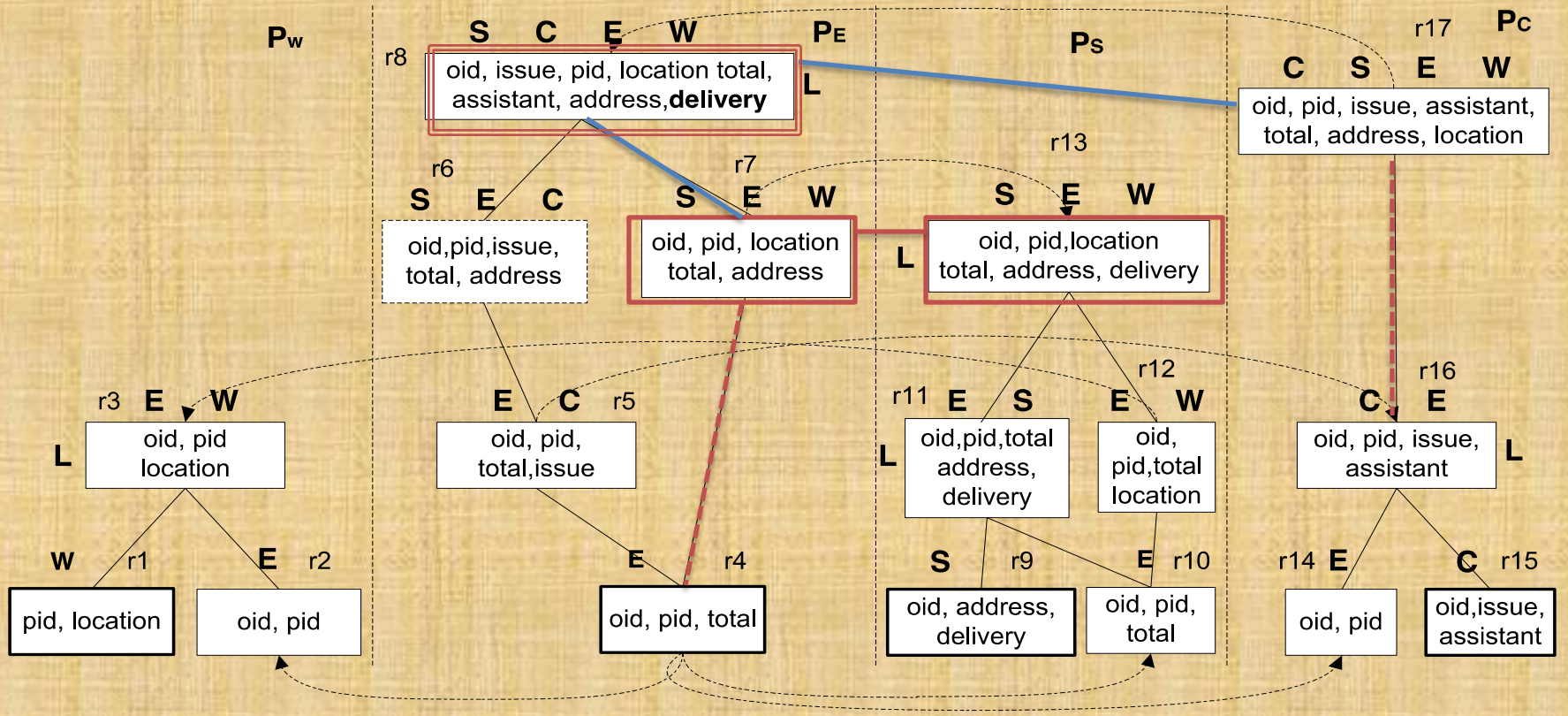


Enforceability Check Outcomes

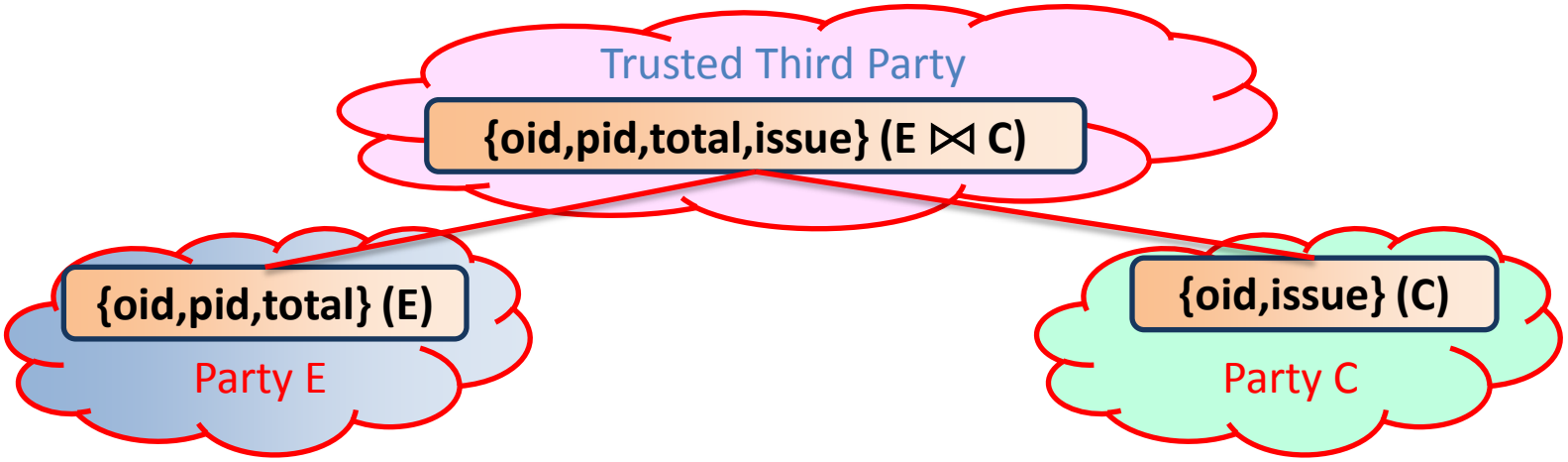
- Case 1: Totally enforceable
  - All rules can be enforced by existing parties
- Case 2: Partially enforceable
  - Can do all joins, but some attributes unavailable
  - Remedy: Change rules to grant more attributes
- Case 3: Unenforceable
  - Can't even do certain joins.
  - Remedy 1: Enforcement via trusted third party
  - Remedy 2: Suggest addition of new rules → may be undesirable.

Adding Attributes for Full Enforceability

- Add at the top and do top-down breadth first search
- Objective: Minimal number of parties affected



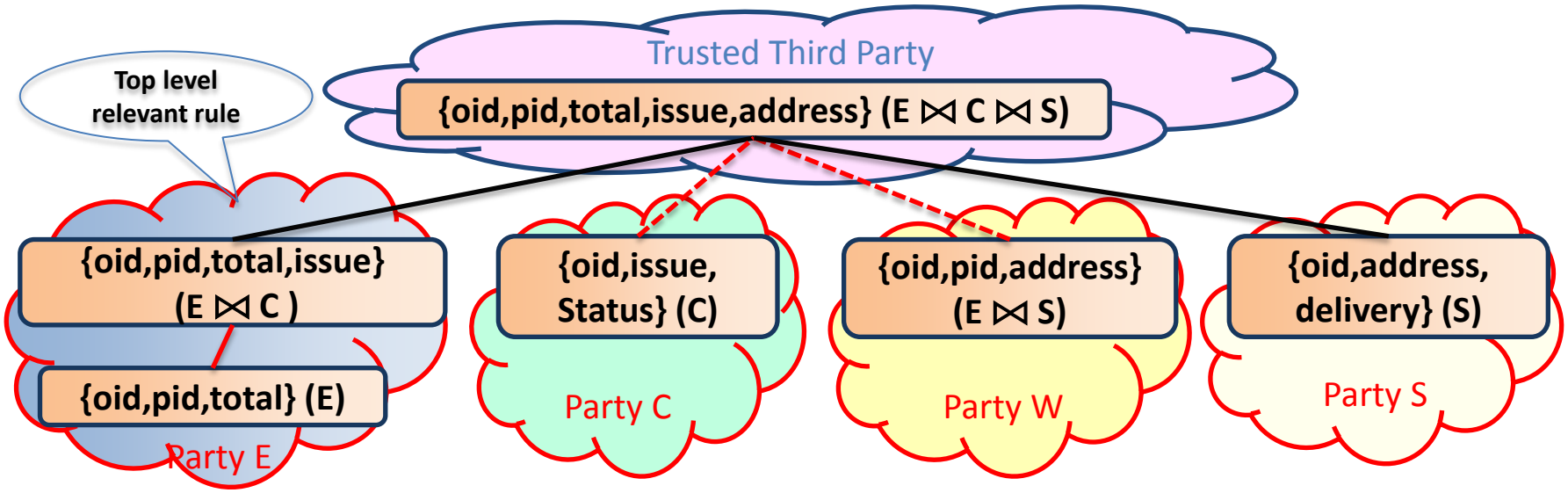
Enforcement via third parties



- Can always do it, but …
  - Overhead of sending information
  - Latency/expense of third party computation
  - Potential exposure of data to the third party
- Minimal involvement of third party

Minimizing Communication costs

- Communication cost
  - The cost of transferring data to third party
- Cost minimization is NP-hard
  - Set covering problem in its simplest form



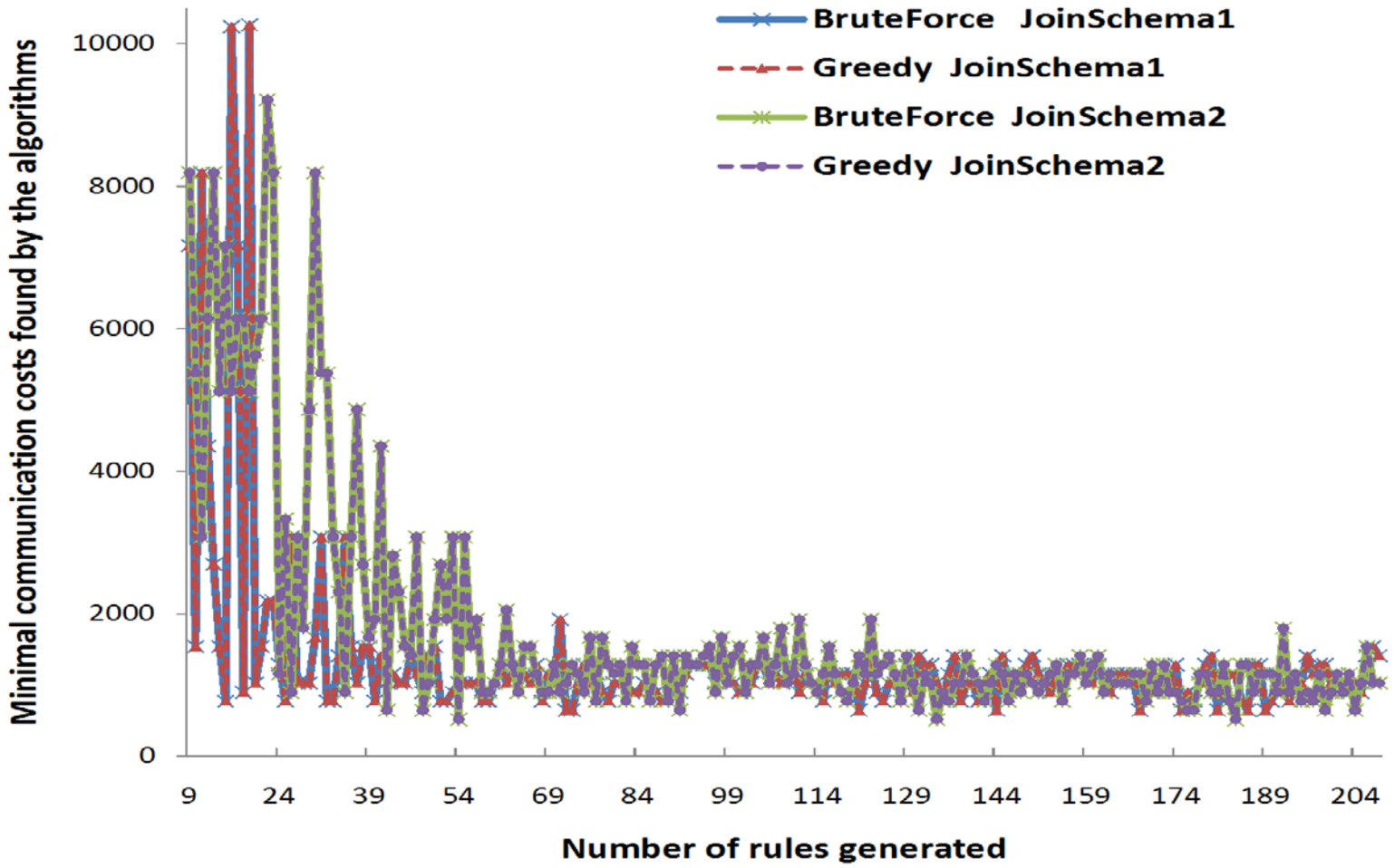
Minimizing Computation Costs

- Computation cost
  - Join costs primarily (w/ or w/o index)
- Minimization problem is NP-hard

- Greedy algorithm to find solutions
  - Select the rule with minimal relative cost (MRC)
  - MRC: Cost of selecting the rule / No. of attributes being covered
  - The greedy algorithm is a 2-approximation
- Works extremely well
  - max difference 5%, mostly identical results

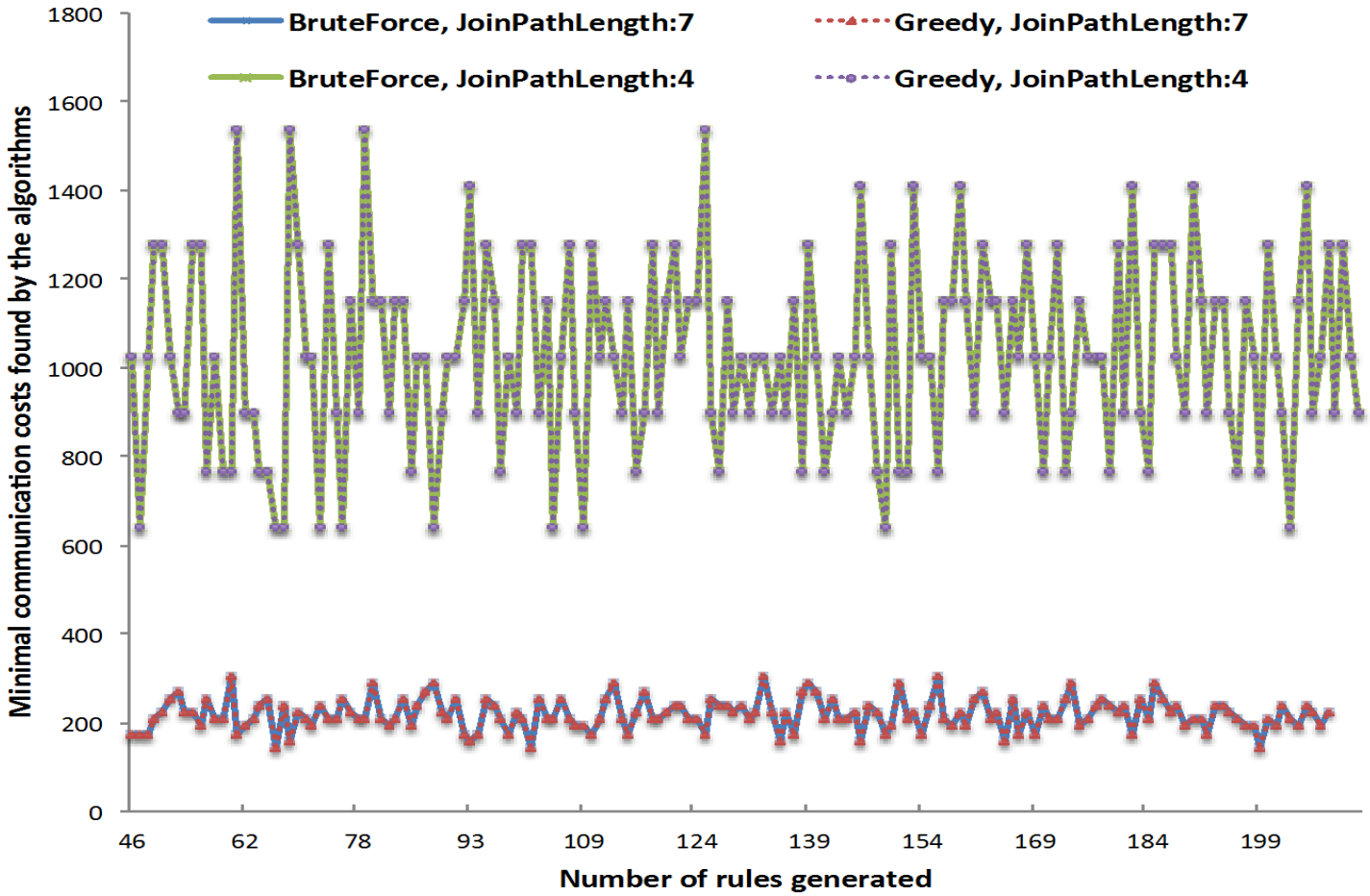
Evaluation – Accuracy

- Results with different join schemas



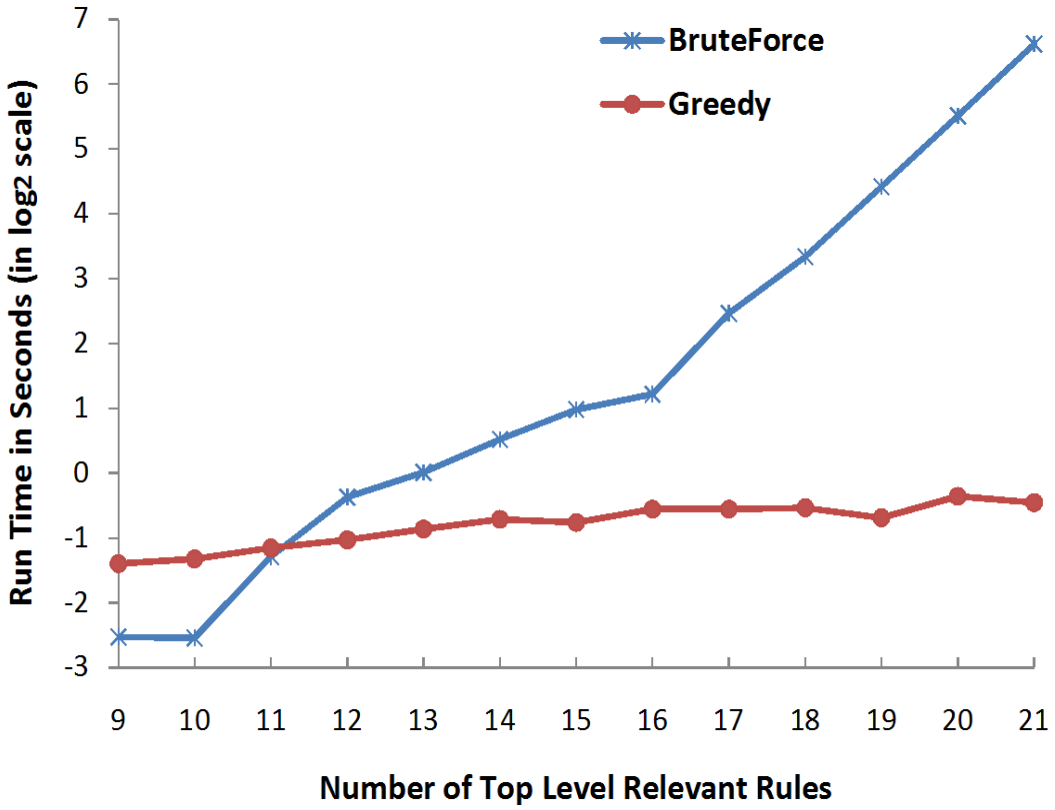
Evaluation – Accuracy

- Results with different join path lengths



Evaluations – Performance

- Function of # top level relevant rules
- Need greedy for >20 top level relevant rules

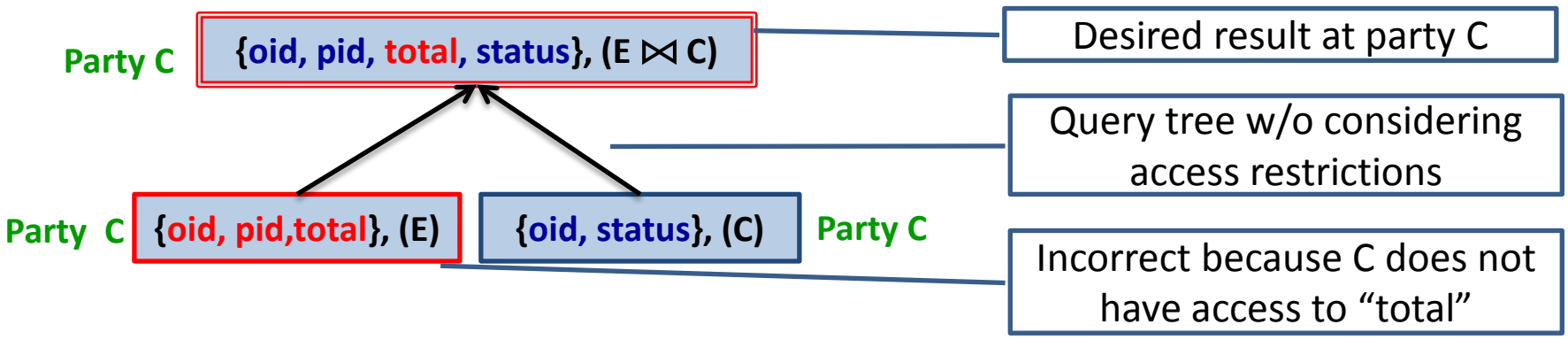


Query Planning

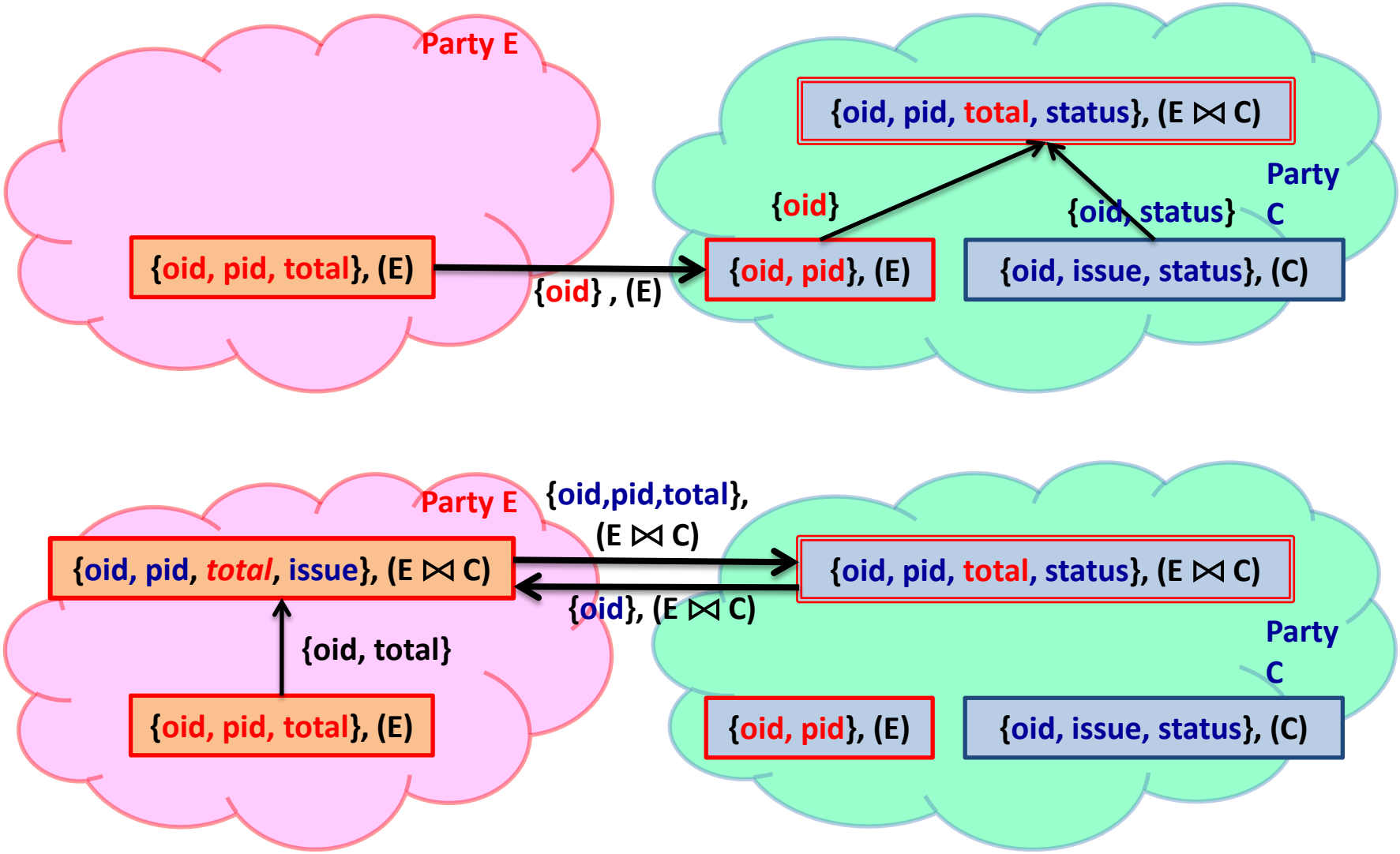
What's New?

- Classical distributed query planning does not consider access restrictions

Rule No.	Authorized Attribute Set	Join Path	Party
1	{oid, pid, total}	E	P_E
2	{oid, pid, total, issue}	E $\bowtie_{oid}$ **C**	P_E
3	{oid, pid}	E	P_C
4	{oid, issue, status}	**C**	P_C
5	{oid, pid, total, status}	E $\bowtie_{oid}$ **C**	P_C



Actual Query Plan is Quite Complex



Difficulty of query planning

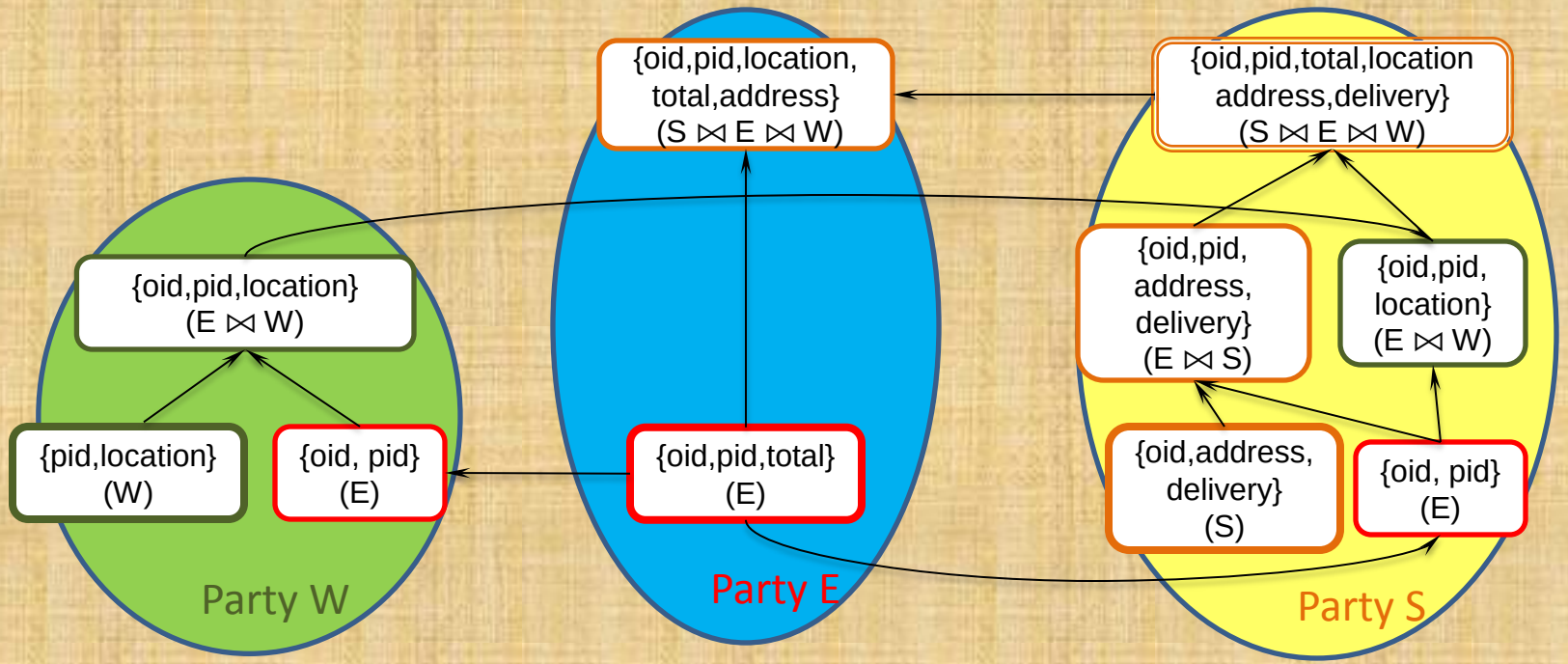
- Query optimization is NP-complete
- Additional Complexity
  - Need to consider all possible ways of obtaining the join path
  - Need to consider all possible ways of covering the desired set of attributes.
- Have an algorithm to generate a "good" query plan, not optimal
  - Properties wrt optimal not known

Proposed approach

- Record the optimal way of enforcing a join path when doing the rule enforcement checking
- Decompose the missing attributes to relevant rules on cooperative parties
  - Get attributes from basic relations by semi-joins
  - Decompose to fewest rules
  - Add corresponding operations to the query plan

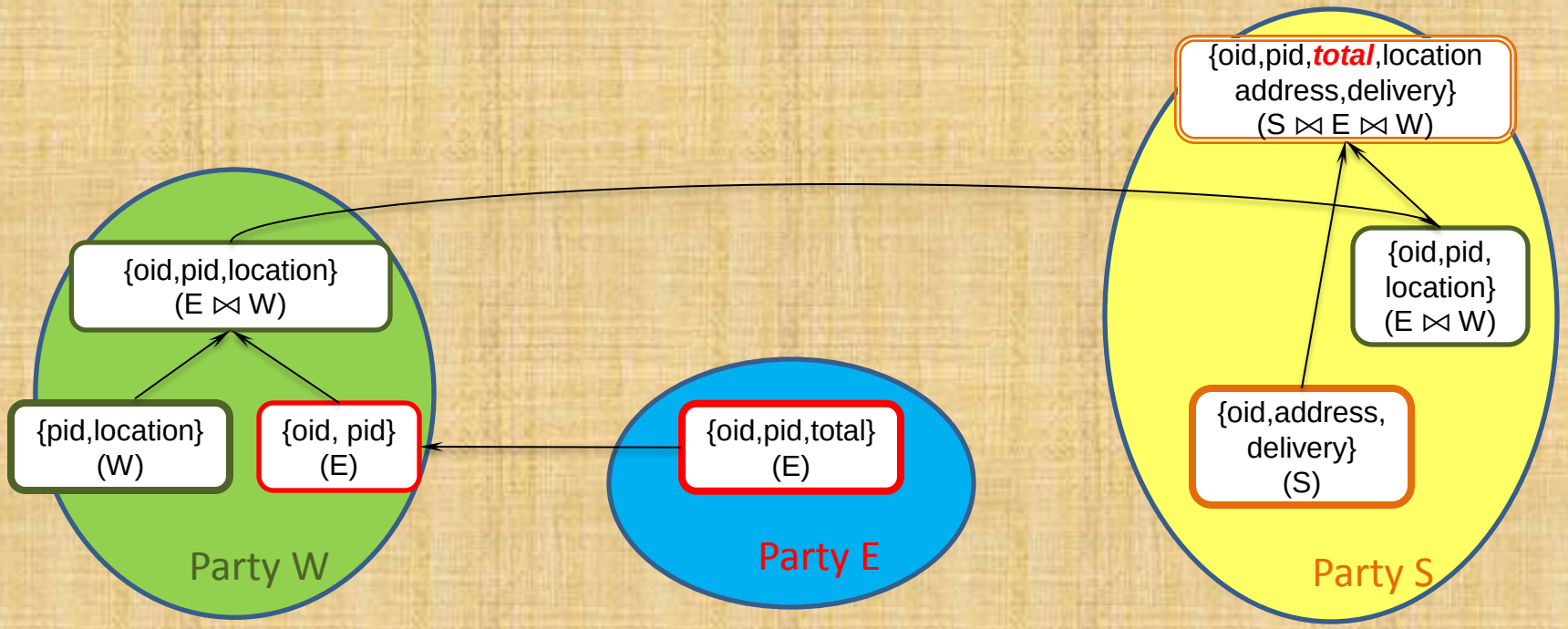
Illustration with example

- Join path enforcement plan
- Retrieving missing attributes



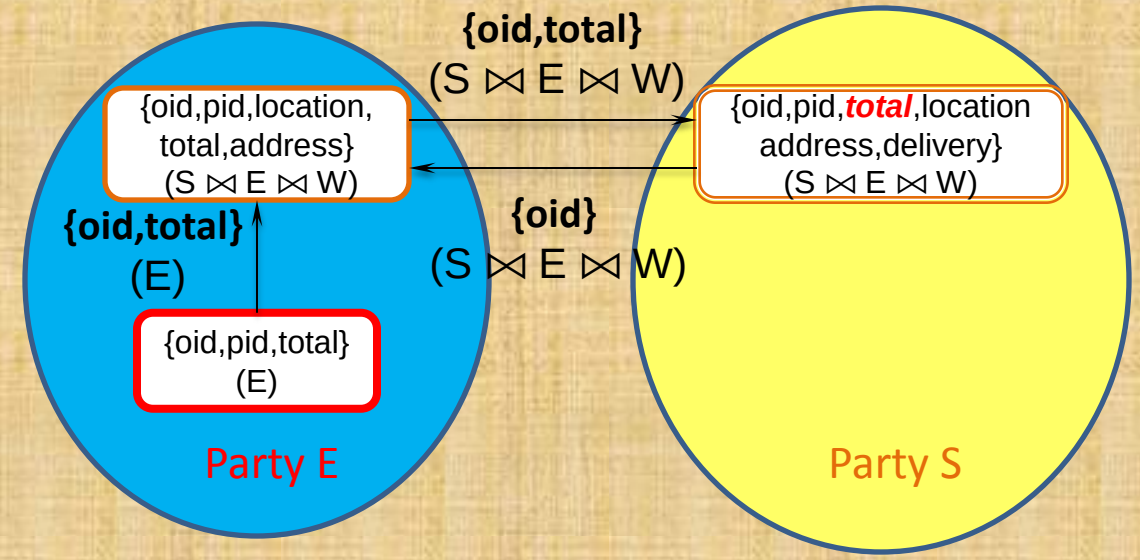
Join path enforcement plan

- The plan enforces most attributes but has a missing attribute “total”



Retrieving missing attributes

- Party E and S has the rules on the equivalent join paths
- A semi-join can enforce the missing attribute “total”



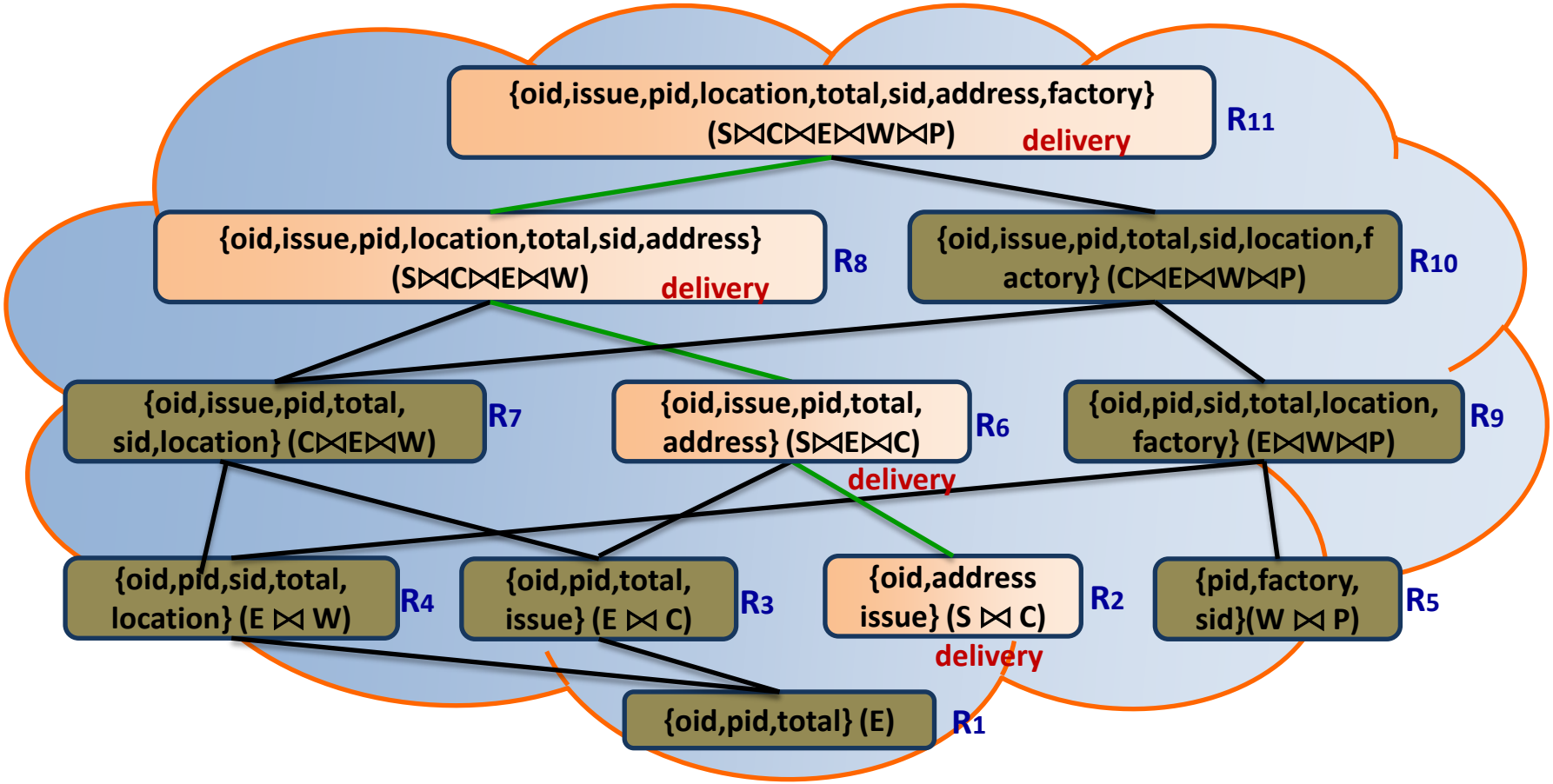
Rule Modification

Rule Modification

- Need for Changes
  - Evolving business needs ➔ Occasional change
  - A distinct set of rules for each mission/workflow
  - Changes based on reciprocal actions of parties
- Issues to consider
  - Efficient change evaluation for consistency & enforceability
  - Treatment of ongoing queries: do they see change?
- Types of Changes
  - Cases 1 & 2: Addition/removal of attributes
  - Cases 3 & 4: Addition/removal of rules

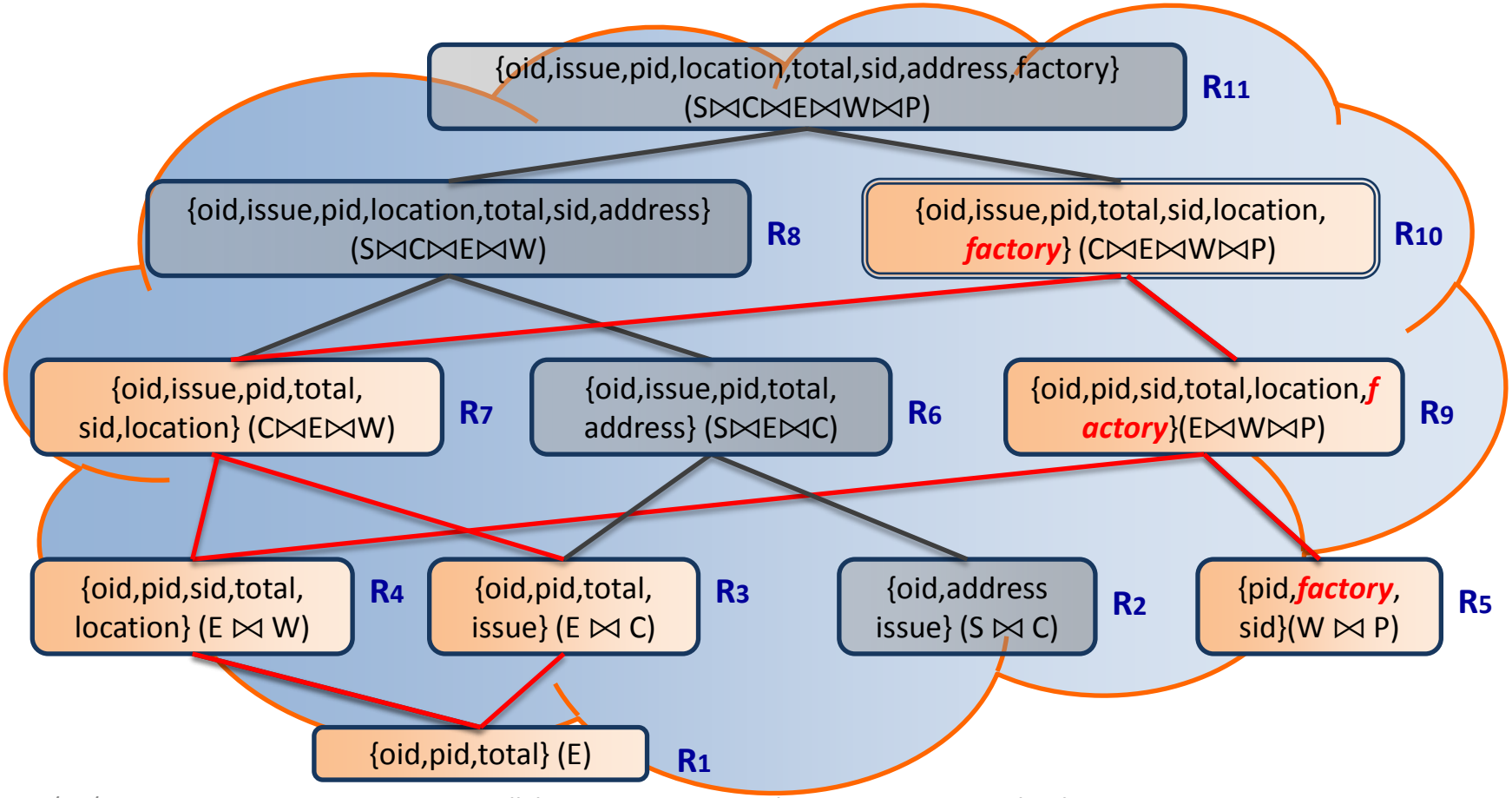
Case 1: Grant New Attributes

- Bottom up propagation of new attr. to higher level rules
- Example: Add “delivery” attribute



Case 2: Revocation of Attributes

- May cause inconsistency
  - Search downwards on relevant rules, revoke such attributes

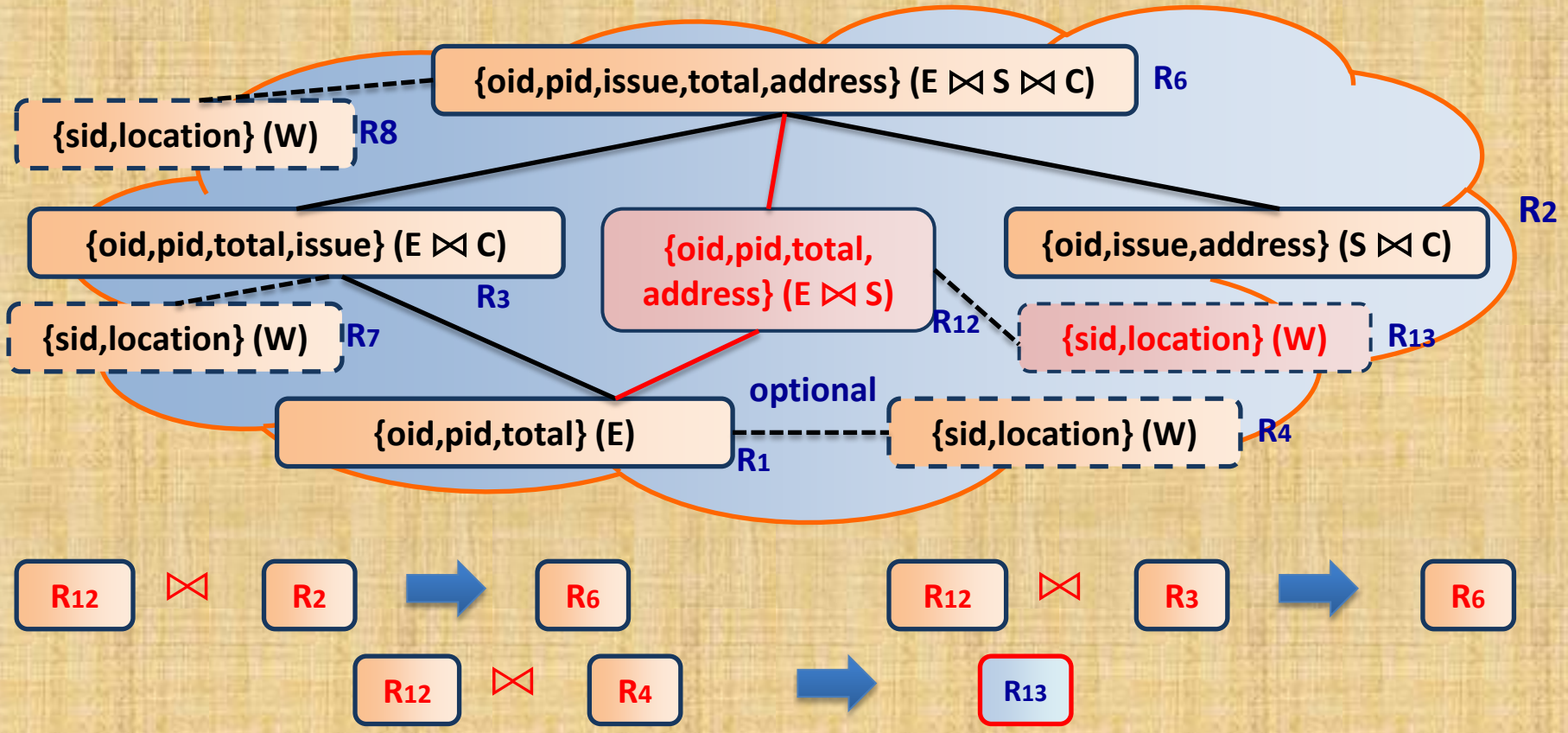


Case 3: Adding New Rule

- A rule with a new join path is added
  - Update CJG for the key attribute of the new rule
    - Consider joins with existing rules
  - Compose with other join groups systematically

Rule No.	Authorized Attribute Set	Join Path	Party
12	{oid, pid, total, address}	$E \bowtie_{oid} S$	P_E

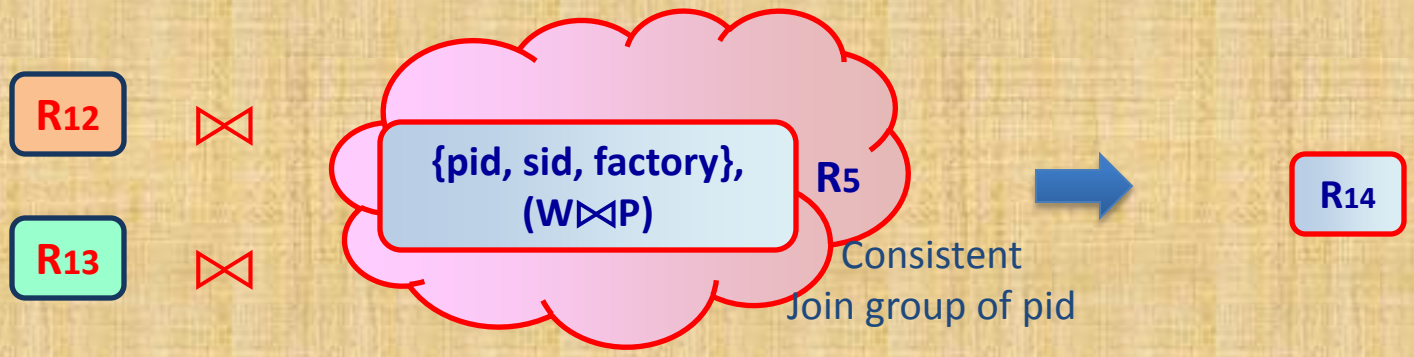
Recompute CJG



Rule No.	Authorized Attribute Set	Join Path	Party
13	{oid, pid, total, address, sid, location}	S ⋈oid E ⋈pid W	P_E

Compose with other Join Groups

Rule No.	Authorized Attribute Set	Join Path	Party
12	{oid, ***pid***, total, address}	E ⋈oid S	P_E
13	{oid, ***pid***, total, address, ***sid***, location}	S ⋈oid E ⋈pid W	P_E

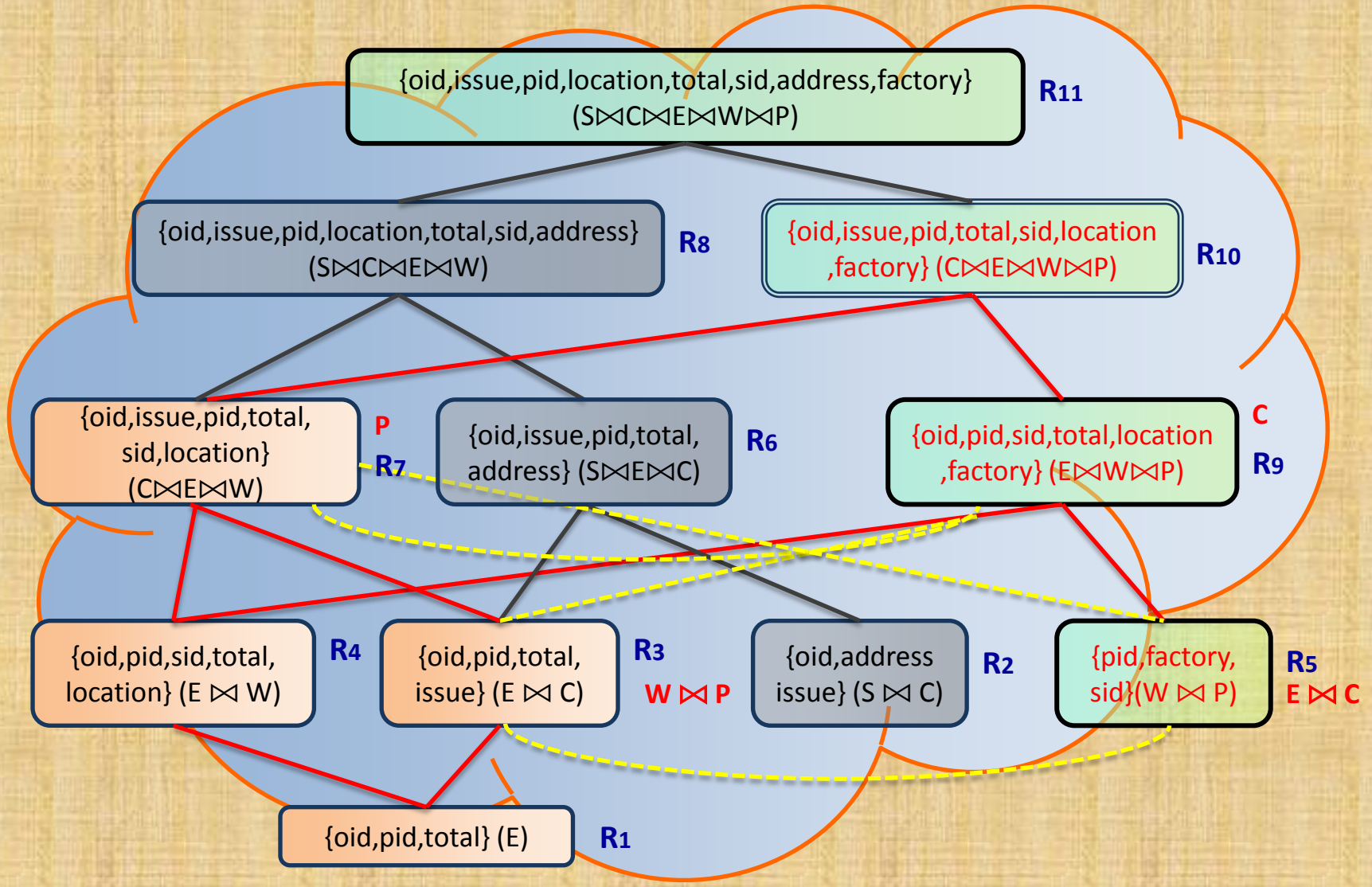


14	{oid, pid, total, address, sid, location, factory}	S ⋈oid E ⋈pid W ⋈sid P	P_E

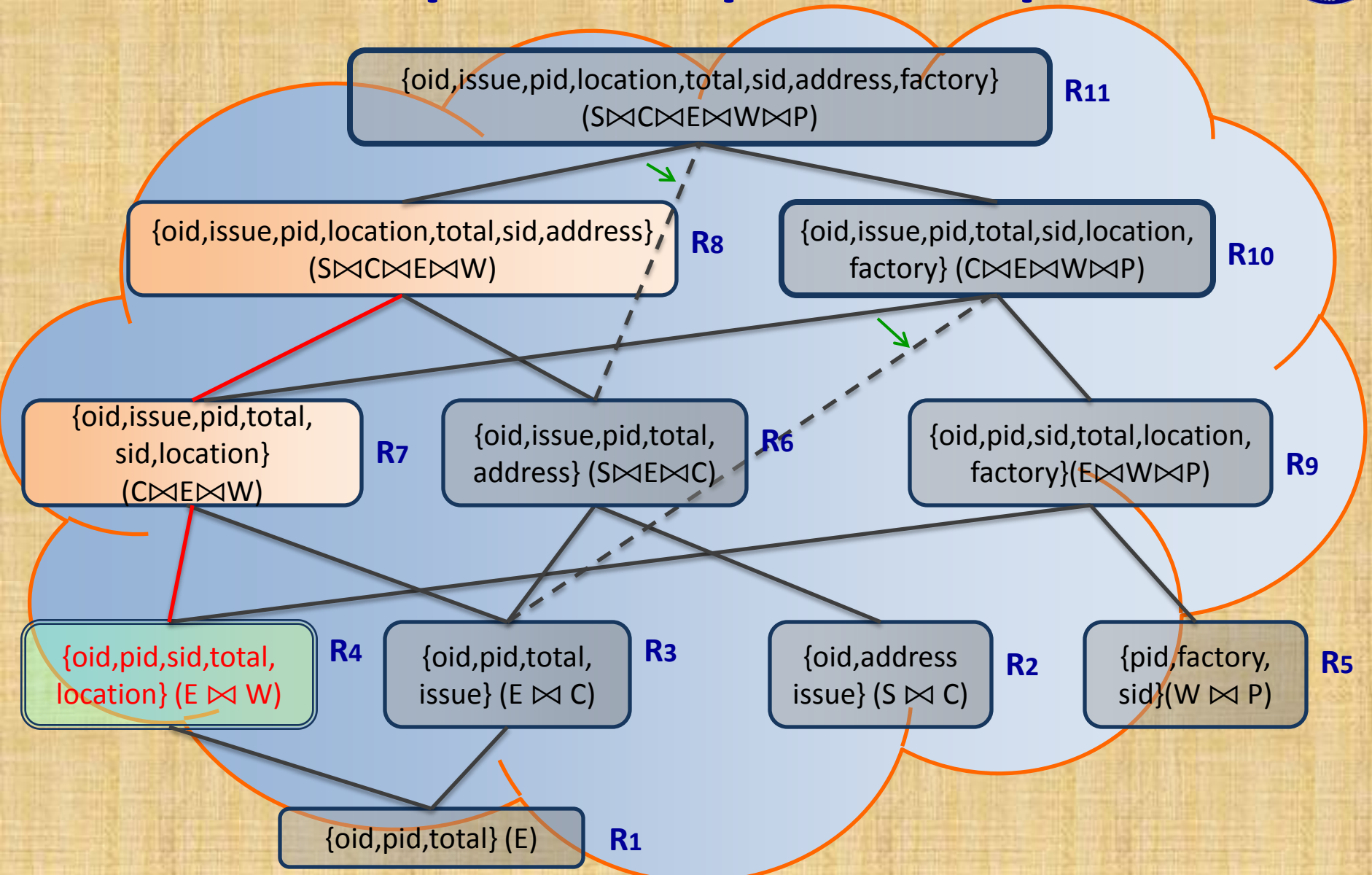
Case 4: Rule Revocation

- Simply removing the rule is inadequate since the rule may be composable from others
- Search the graph & remove rules from each possible join pair
- Fix up: Revoke related rules that are added in consistency checking process
- Split possible join pairs
  - Find relevant rules
  - Search rule pairs that can recover the join path
    - Matching join path
    - Top-down search
  - Revoke one rule from each found pairs
    - Rules with most appearances are preferred
    - Rules have fewer connections with others are preferred

Step 1: Remove Join Paths



Step 2: Fix Up the Graph



Other Issues

- Untrusted Cloud Provider
  - Cooperative verification of results returned by a query.
    - Exploit multiple ways of executing queries
    - Collaborative water-marking, etc.
  - Queries over encrypted data
- Rules with limited forms of selection
- Lack of trust between parties
  - Verification of results

Thank you!

References

- [D.C.Vimercati'08] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Controlled information sharing in collaborative distributed query processing. In ICDCS 2008, Beijing, China, June 2008.
- [CCS'08] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Assessing query privileges via safe and efficient permission composition. In CCS 2008, Virginia, USA, October 27-31, 2008.
- [Kossmann'00] D. Kossmann. The state of the art in distributed query processing. ACM Comput. Survey, 32(4):422–469, 2000.
- [Cali'08] A. Cali and D. Martinenghi. Querying data under access limitations. In ICDE 2008, April 7-12, 2008, Cancun, Mexico, pages 50–59, 2008.
- [Goldstein'01] J. Goldstein and P. Larson. Optimizing queries using materialized views: A practical, scalable solution. In SIGMOD 2001, pages 331–342.
- [Halevy'01] A. Y. Halevy. Answering queries using views: A survey. VLDB Journal, 10(4):270–294, 2001.
- [Tolone'05] Tolone, Ahn, Pai, and Hong. Access control in collaborative systems. CSURV: Computing Surveys, 37, 2005.
- [Le1'12] M. Le, K. Kant and S. Jajodia, "Consistent Query Plan Generation in Secure Cooperative Data Access", Proc. of SafeConfig 2012, Baltimore, MD, Oct 3-4, 2012.
- [Le2'12] M. Le, K. Kant and S. Jajodia, "Access Rule Consistency in Cooperative Data Access Environment", Proc. of CollaborateCom conference, Oct 15-17, 2012, Pittsburgh, PA. An extended version to appear in Elsevier Computers & Security Journal.