Australian National University

COMP3430 / 8430
Data wrangling

Lecture 11: Schema mapping and matching (Lecturer: Peter Christen)

Based on slides by Prof Erhard Rahm (University of Leipzig and ScaDS, Germany)

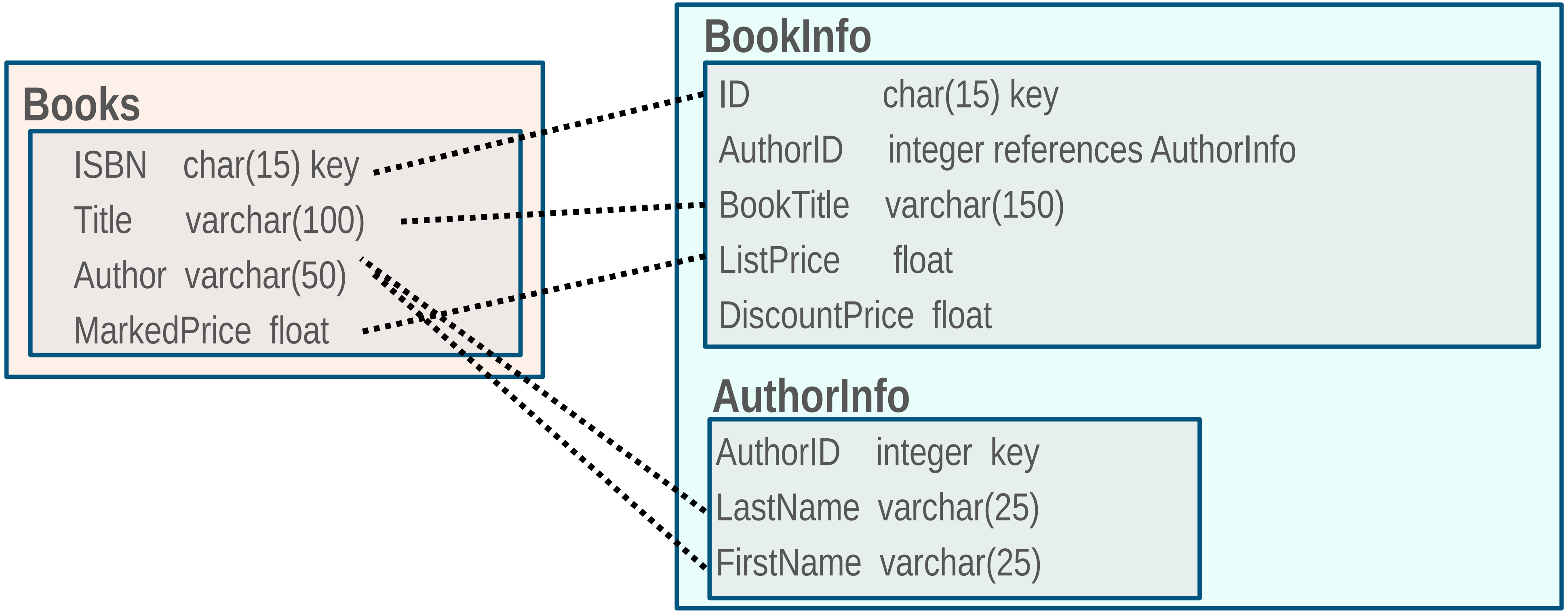
Lecture outline

- The schema matching problem
- Examples of schema matching applications
- Schema matching techniques

Note: The terms schema *matching* and *mapping* are often used interchangeably

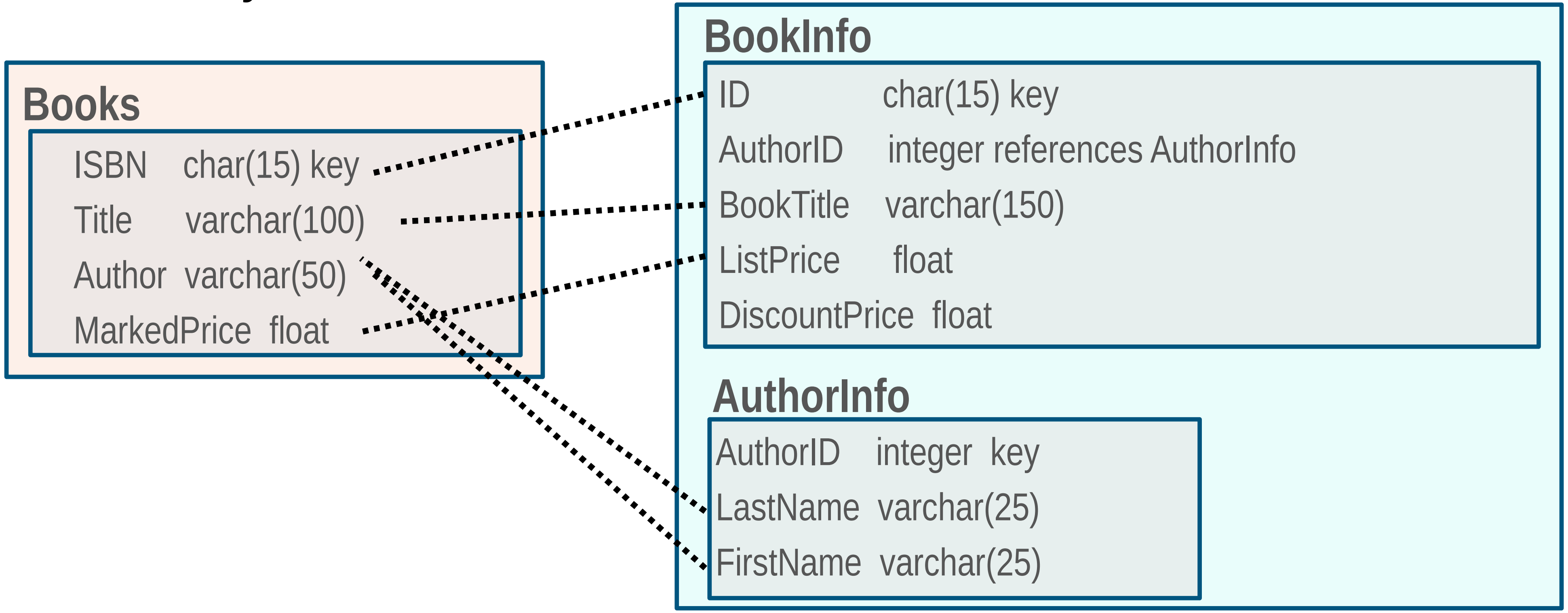
The schema matching problem

- The problem of generating correspondences between elements of two database schemas



Basic input to matching techniques

- Schema structures; element (attribute) names; and constraints such as data types and keys



Other inputs to basic schema matching

- **Synonyms**
  Code = Id = Num = No
  Zip = Postal [code]

- **Acronyms**
  PO = Purchase Order
  UOM = Unit of Measure
  SS# = Social Security Number

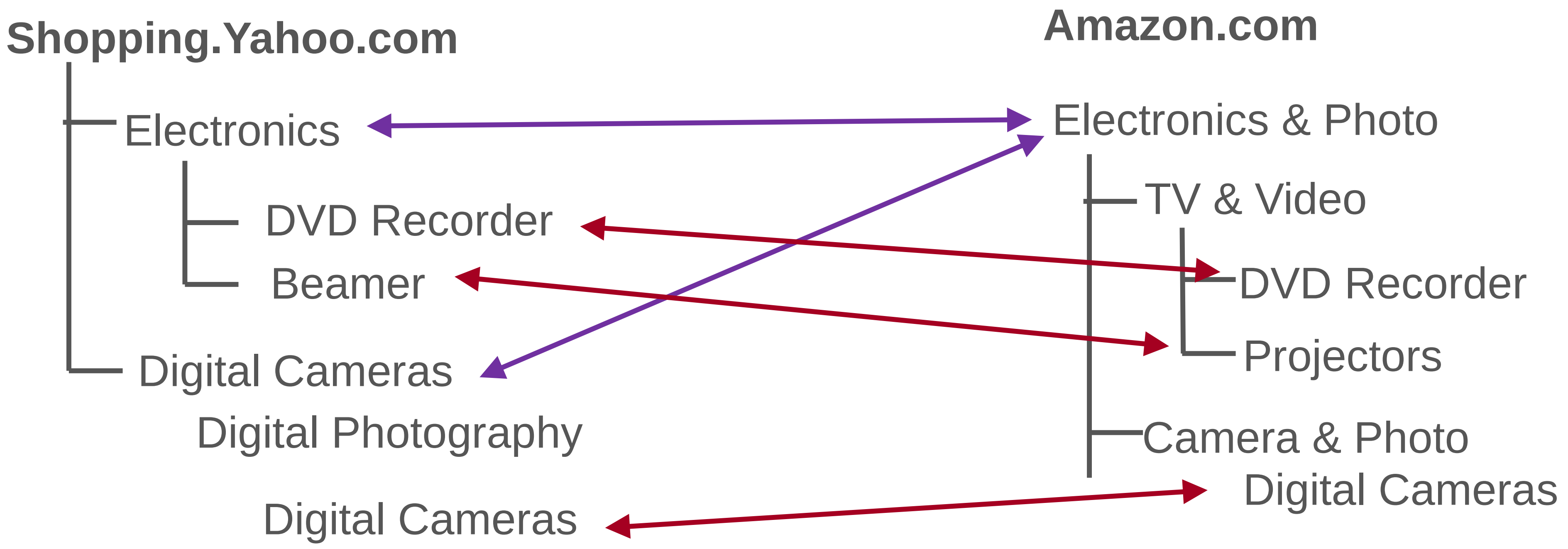
- **Data instances** (attribute values)
  Key insight: *Elements match if they have similar instances or value distributions*

Many applications need correspondences

- **Data translation**
  - Object-to-relational mapping
  - XML message translation (for example between different applications)
  - Data warehouse loading (ETL)

- **Data integration**
  - ER (entity relationship) design tools
  - Schema evolution (temporal changes)
  - Record linkage *(next lecture)*

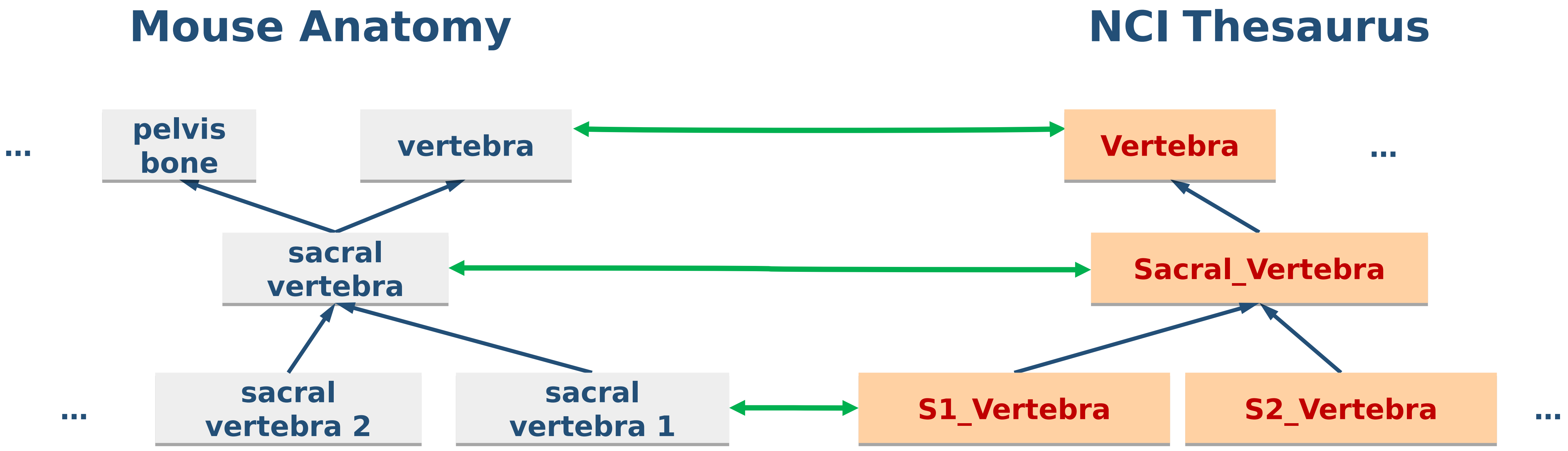
Example: matching product catalogues



- Mapping is useful for improving query results, for example to find a specific product across Web sites, or merging catalogues

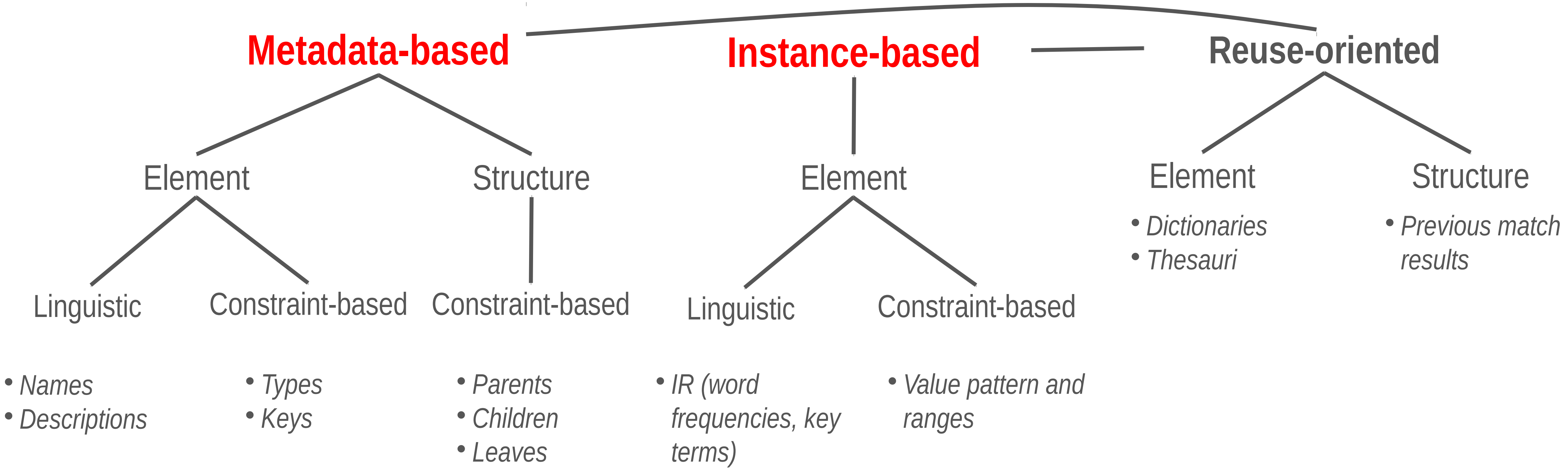
Example: matching life science ontologies

- There are many large biomedical ontologies, used to annotate or enrich objects (genes, proteins, etc.) or documents (publications, electronic health records, etc.)



Taxonomy of automatic match techniques

- Matcher combinations are either *hybrid* matches (that consider for example name and type similarity), or *composite* matches



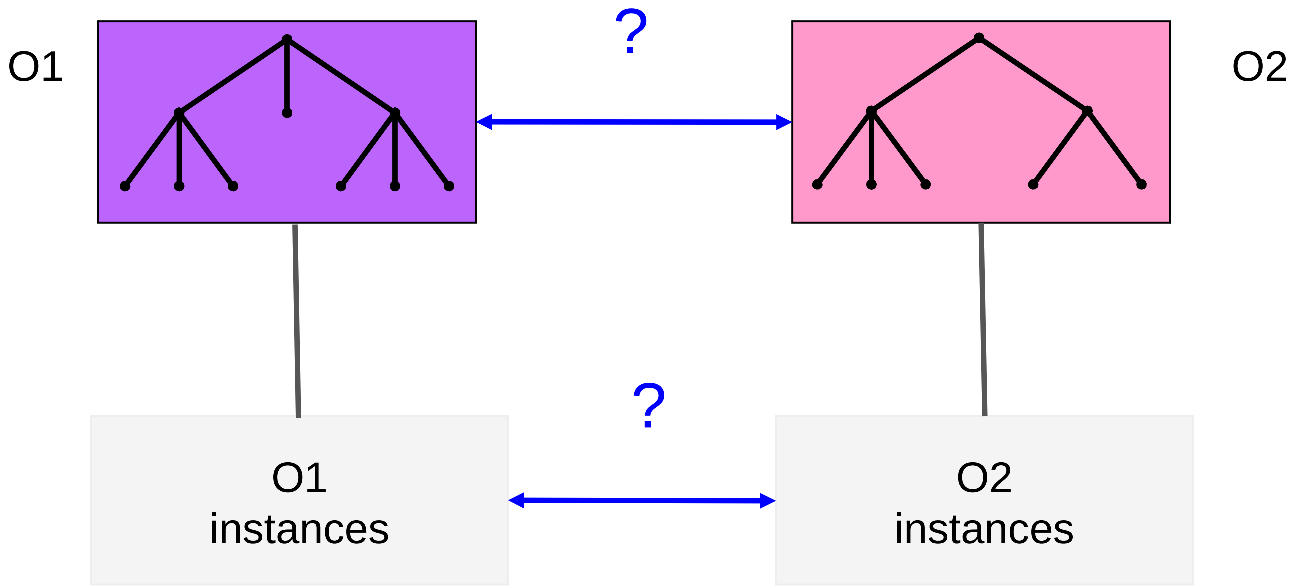
E. Rahm and P.A. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4), 2001

Match techniques

- **Linguistic matchers**
  - (String) similarity of concept/element names
  - Based on dictionaries or thesauri, such as WordNet / UMLS
- **Structure-based matchers**
  - Consider similarity of ancestors/descendants
  - Graph-based matching such as *Similarity Flooding* (Melnik et al., ICDE 2002)
- **Instance-based matchers**
  - Concepts with similar instances/annotated objects should match
  - Consider all instances of a concept as a document and utilise document similarity (such as TF-IDF) to find matching concepts

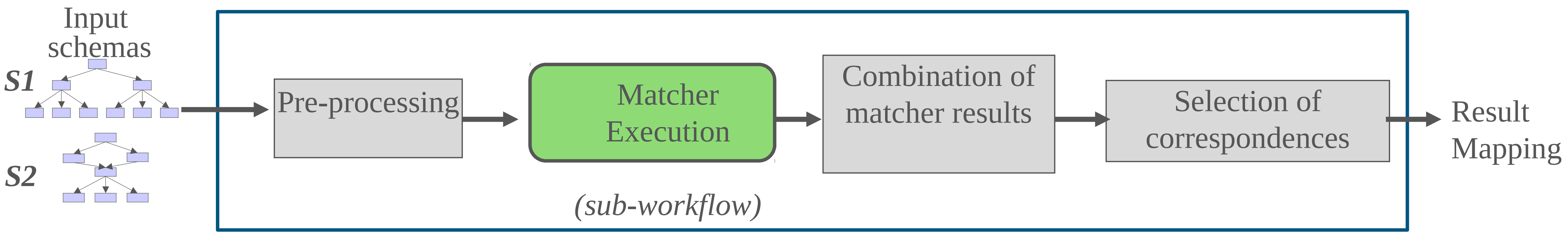
Instance based ontology matching

- Concepts with most similar instances should match (requires shared / similar instances for most concepts)
- Mutual treatment of *entity resolution* (instance matching) and ontology matching
- Promising for link discovery in the Linked Open Web of Data

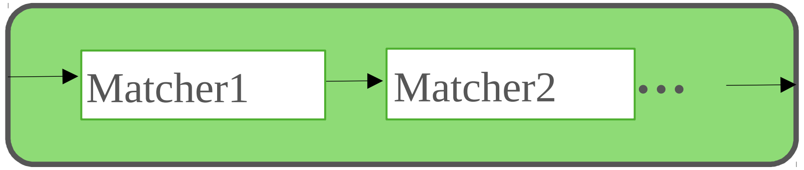


Schema matching is a multi-step process

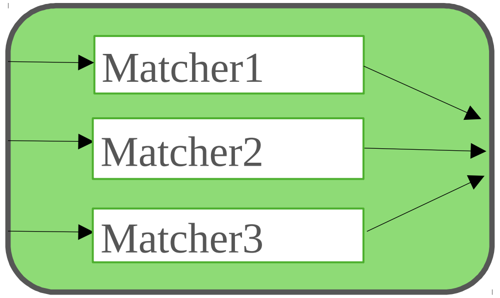
General workflow



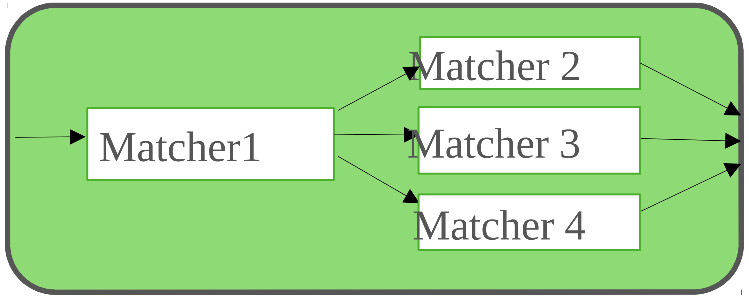
Matcher sub-workflow



Sequential matchers



Parallel (independent) matchers



Mixed strategy

Large-scale matching

- Very large ontologies / schemas (>10,000 elements)
  - Quadratic complexity of evaluating the Cartesian product (match efficiency)
  - Difficult to find all right correspondences (match quality)
  - Support for user interaction

- Many (>>2) ontologies/schemas
  - Holistic ontology/schema matching
  - Clustering of equivalent concepts/elements or linking to some hubs

Self-tuning match workflows (1)

- Semi-automatic configuration
  - Selection and ordering of matchers
  - Combination of match results
  - Selection of correspondences (top-k, threshold, …)

- Prototype tuning frameworks (Apfel, eTuner, YAM)
  - Use of supervised machine learning
  - Need previously solved match problems for training
  - Difficult to support large schemas

Self-tuning match workflows (2)

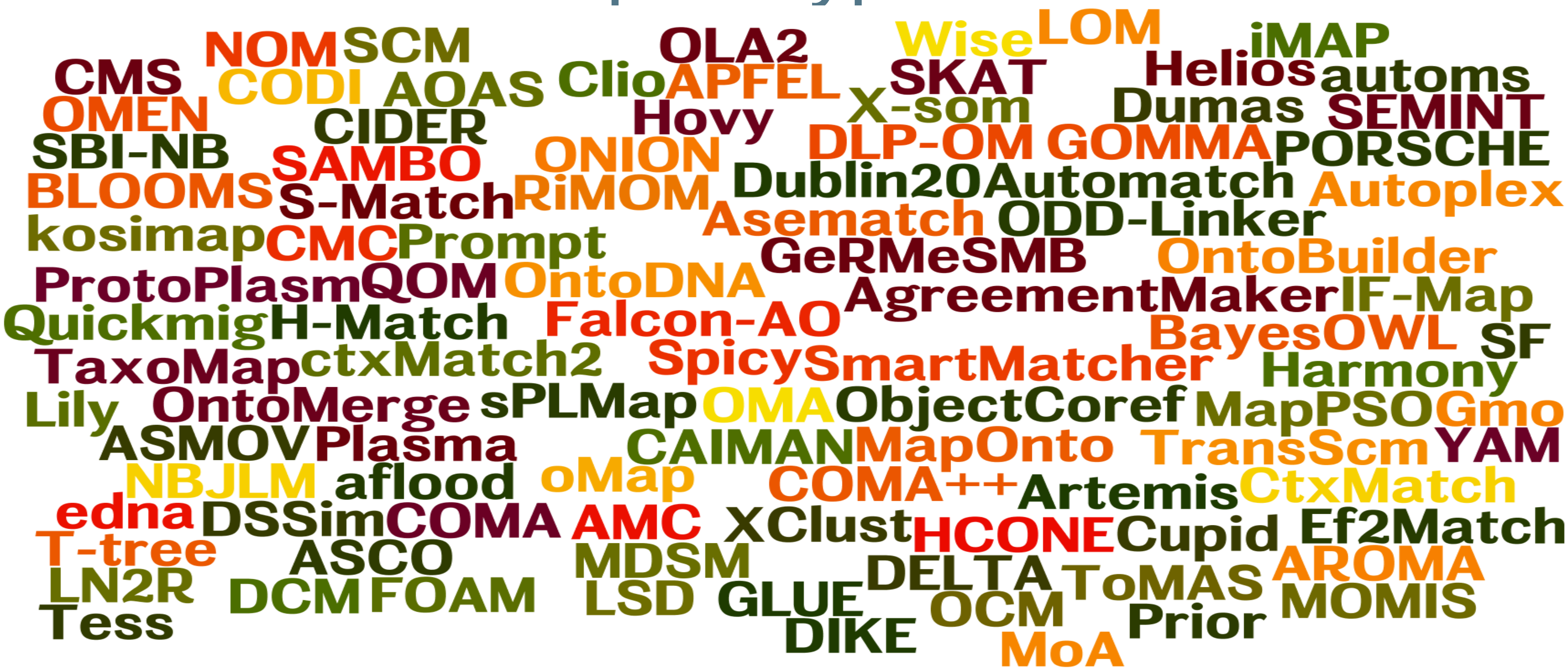
- Heuristic approaches
  - Use linguistic and structural similarity of input schemas to select matchers and their weights
  - Favour matchers that give higher similarity values in the combination of matcher results

- Rule-based approach
  - Comprehensive rule set to determine and tune match workflow
  - Use of schema features and intermediate match results

Re-use oriented matching

- Many similar match tasks, therefore reuse previous matches
  - Can improve both efficiency and match quality
- Repository of match tasks is needed
  - Store previously matched schemas / ontologies and obtained mappings
  - Identify and apply reusable correspondences
- First proposals for reuse at three mapping granularities
  1) Reuse *individual element correspondences,* such as synonyms
  2) Reuse *complete mappings*, for example after schema/ontology evolution
  3) Reuse *mappings between schema/ontology fragments* (such as common data elements)

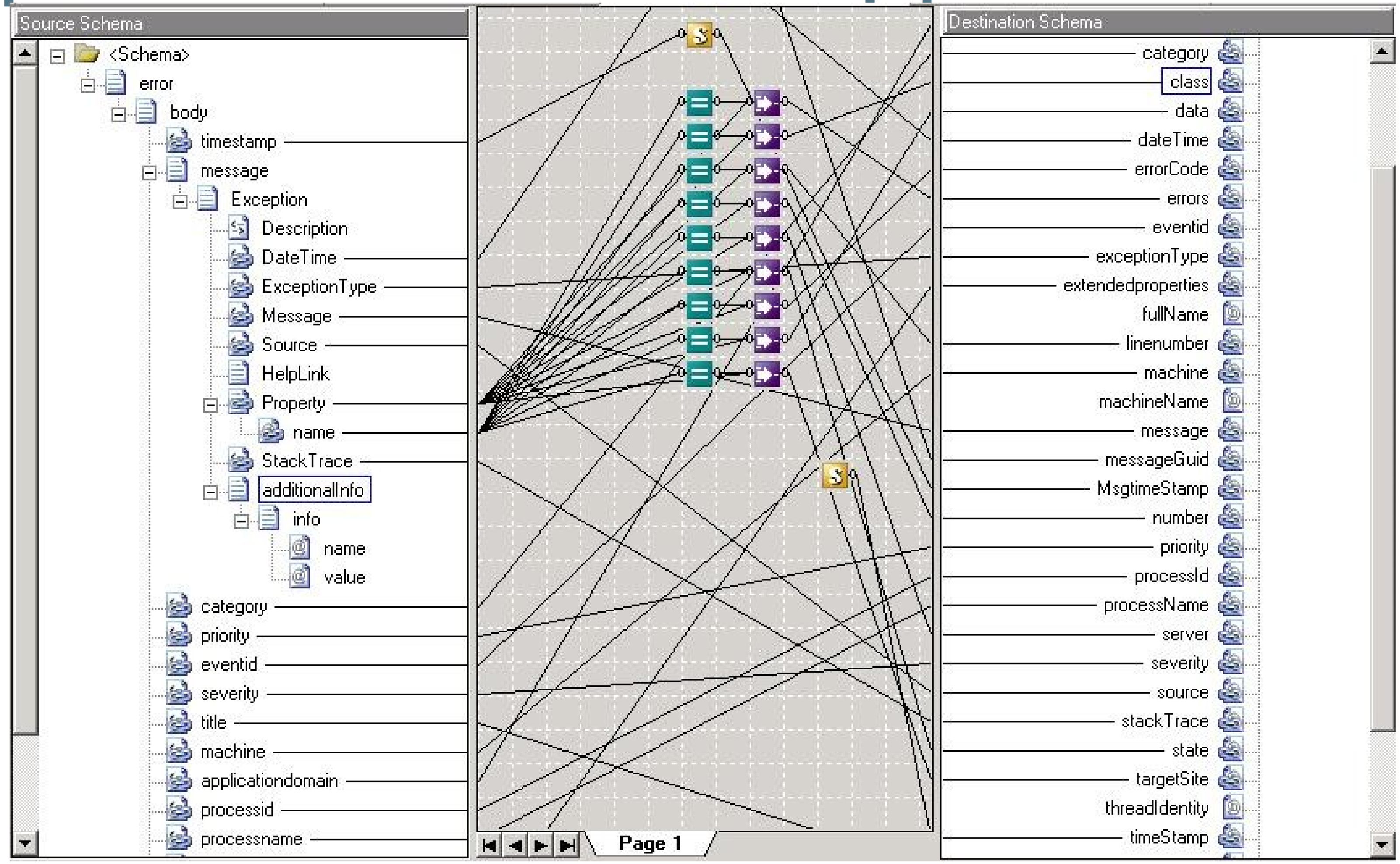
Research match prototypes



Commercial oriented matching tools

- Many GUI-based mapping editors to manually specify correspondences and mappings
- Initial support for automatic matching, in particular linguistic matching
  - Altova MapForce
  - MS BizTalk Server
  - SAP Netweaver
  - IBM Infosphere
- Many further improvements possible
  - Structural / instance-based matching
  - Advanced techniques for large schemas

Example tool: Biztalk mapper



Example tool: Altova MapForce