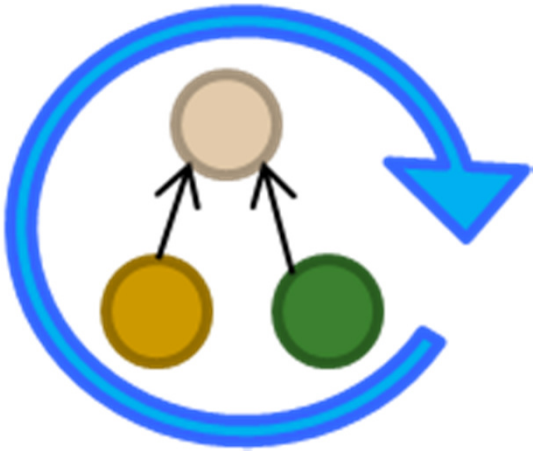
Ontologie-Management

Kapitel 5: Matching und Mapping

Dr. Michael Hartung

Wintersemester 2012/13

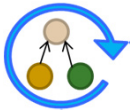
Universität Leipzig
Institut für Informatik
http://dbs.uni-leipzig.de



Inhalt

- **Ontology Matching**
  - Warum?
  - Definition, Probleme
  - Klassifikation von Match-Ansätzen
  - Diskussion verschiedener Ansätze
  - Evaluierung

- **Large-Scale Ontology Matching**
  - Probleme bei großen Ontologien
  - Ansätze zum zeiteffizienten Matching von Ontologien

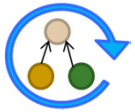


Ontology Matching – Motivation

Warum?

- Verschiedene Applikationsszenarien erfordern einen Abgleich von Ontologien
  - Datentransformation
    - Integration von Daten (siehe VL Datenintegration)
  - Anfrageverarbeitung (Umformulierung von Anfragen)
    - P2P Netzwerke, Web Service Komposition, ...
  - Datenmigration bei Ontologieänderungen
    - Ontology Evolution
  - Navigation / Browsing / Analysen
    - Navigation zwischen Datenquellen (z.B. Linked Data)
    - Übertragung von Analyseergebnissen (z.B. Maus → Mensch)

→ Überall: "Wie stehen zwei Ontologien in Beziehung?"

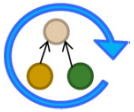


Ontology Matching – Definition

Definition

Ontology Matching is the process of finding relationships or correspondences between entities (concepts, categories) of two different ontologies *(Euzenat, Shvaiko: Ontology Matching, 2007)*

- Identifikation semantischer Korrespondenzen zwischen verschiedenen Ontologien
- *Wichtig:* Prozess (d.h. Algorithmus, Methode, Strategie) nicht das Ergebnis selbst

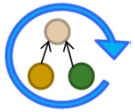


Ontology Matching – Definition

Eingabe / Ausgabe

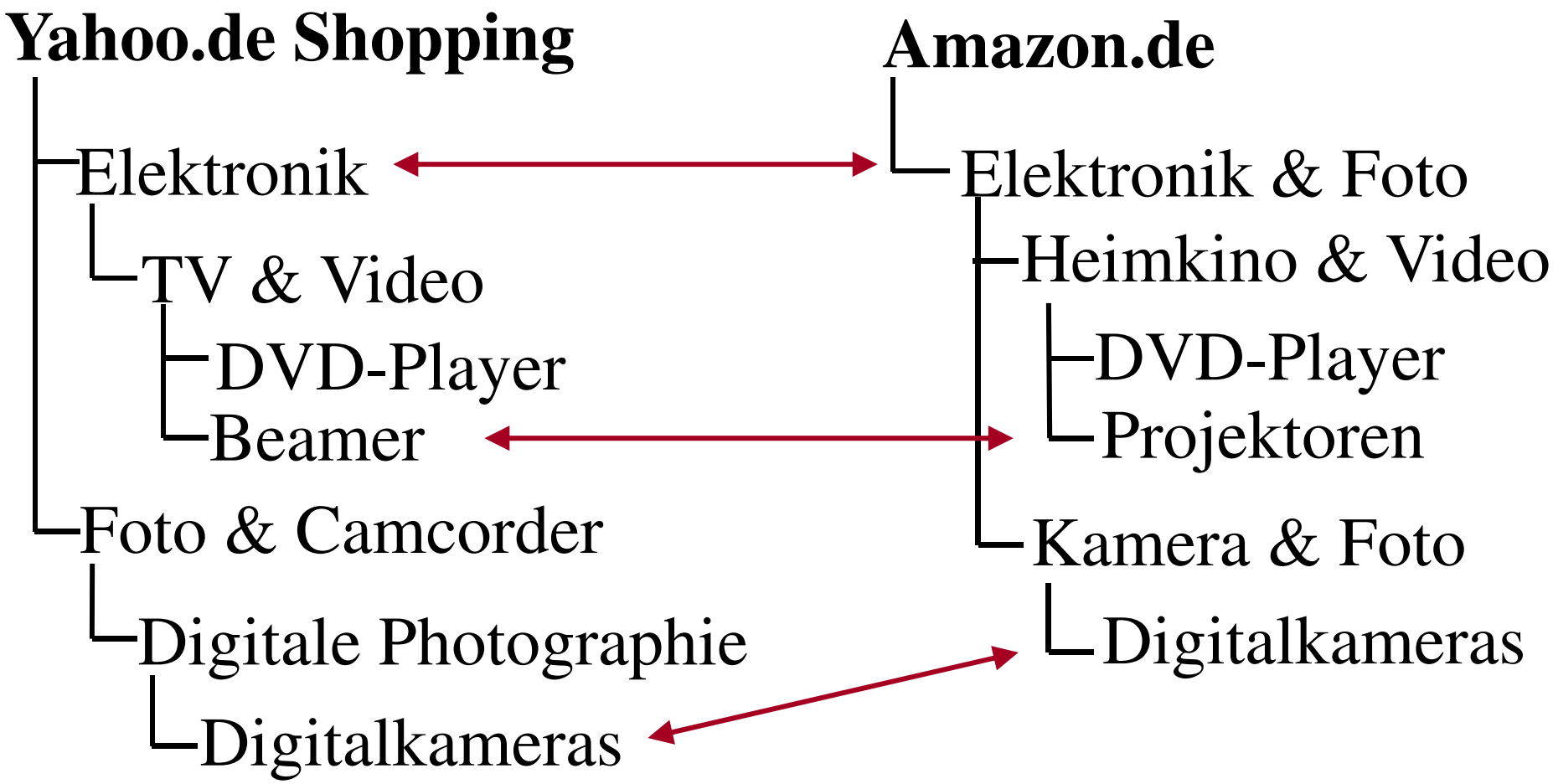
- *Eingabe*
  - Zwei Ontologien O1 und O2
  - Evtl. Instanzen, Annotationen zu O1/O2
  - Evtl. weiteres Hintergrundwissen
    - Domänenwissen wie Wörterbücher, Abkürzungsverzeichnisse, ...
    - Andere Terminologien in der gleichen Domäne
- *Ausgabe*
  - **Mapping** (Alignment) zwischen O1 und O2, d.h. eine Menge von Korrespondenzen zwischen semantisch gleichen Konzepten / Kategorien

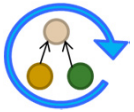
$$M_{O1,O2} = \{(c1,c2) \mid c1 \in O1,\ c2 \in O2\}$$



Ontology Matching – Beispiele

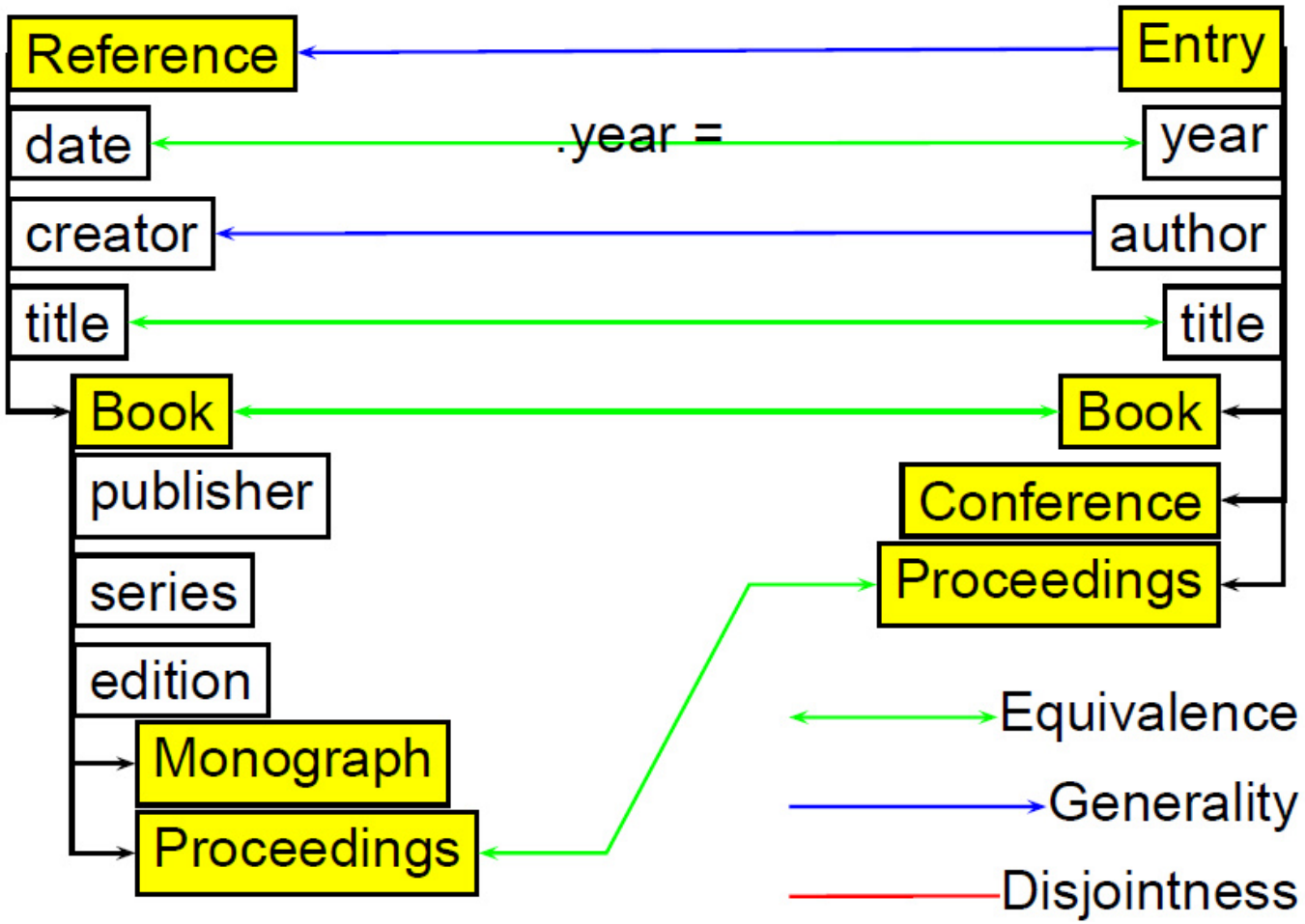
Produktkataloge





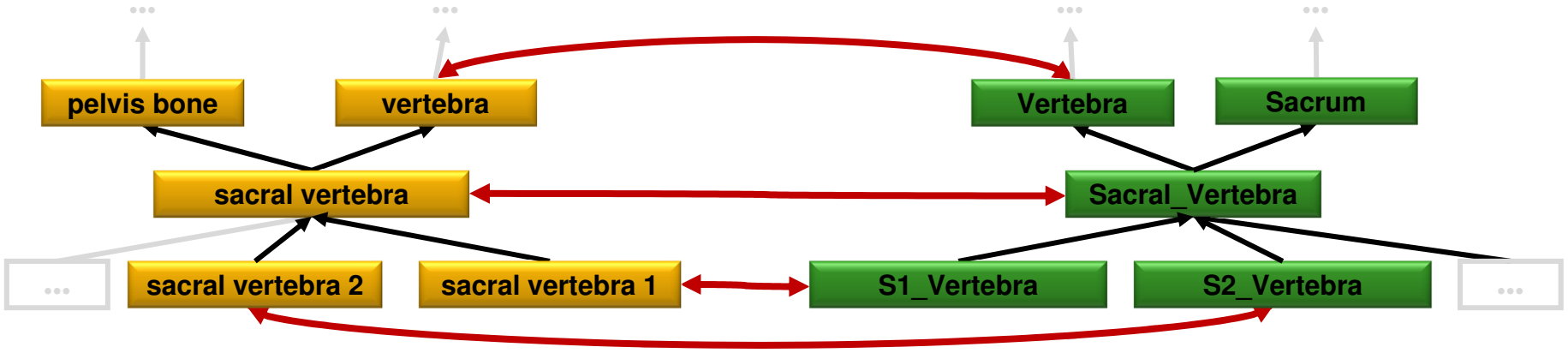
Ontology Matching – Beispiele

Publikationsverwaltung



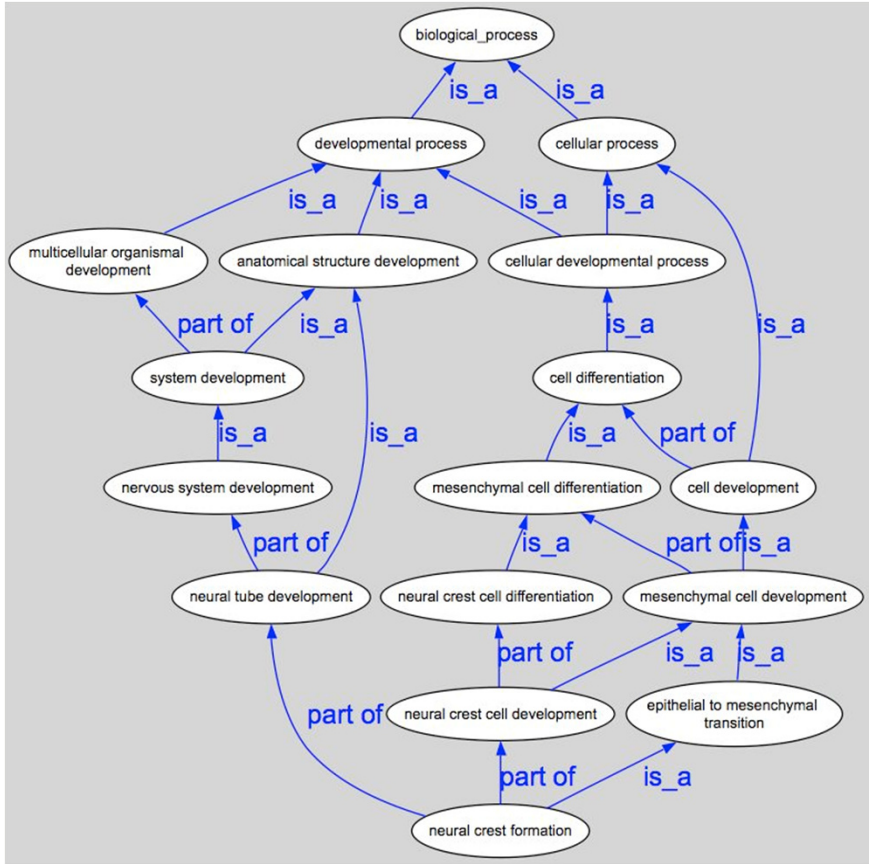
Ontology Matching – Beispiele

Anatomie-Ontologien



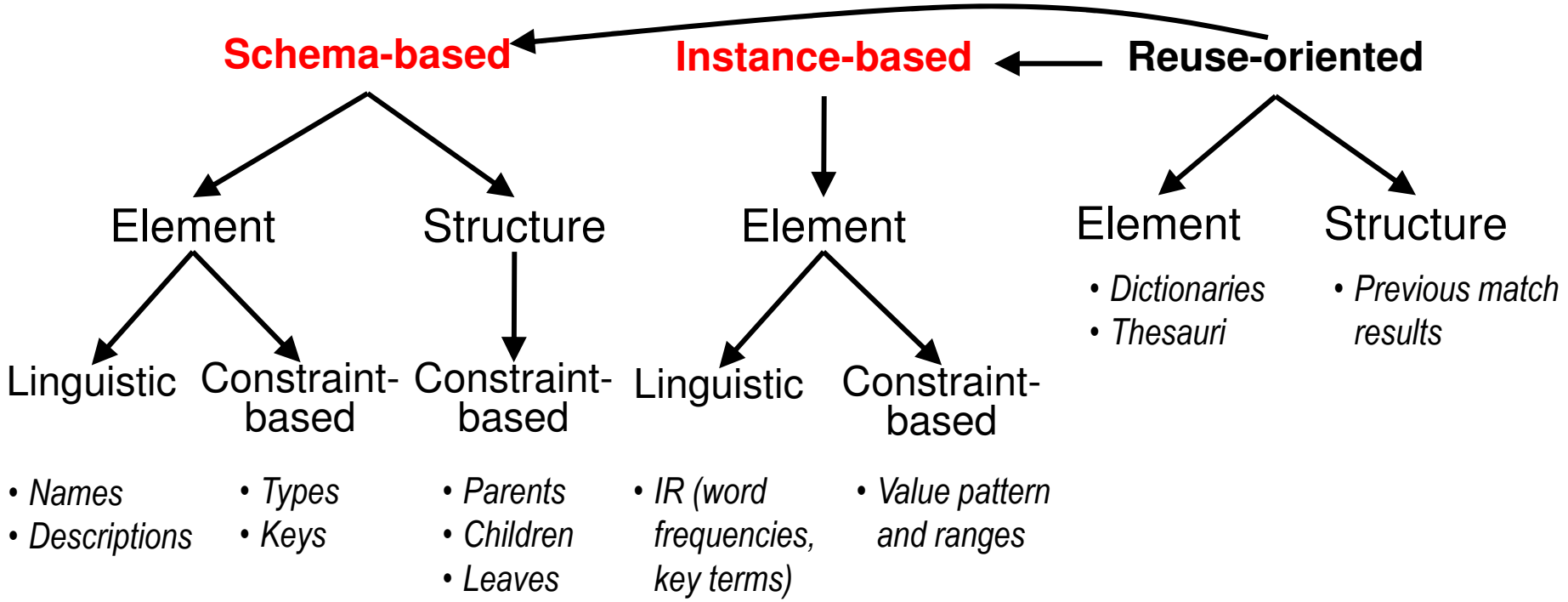
Ontology Matching – Probleme

- **Groß und unübersichtlich**
  - > 10.000 Konzepte
  - Tiefe Hierarchien
- **Unabhängige Entwickelung**
  - Vers. Arten der Heterogenität
    - Fremdsprachenproblem
    - Unbekannte Synonyme, Homonyme
    - Verwendete Abkürzungen
- **Schwierigkeit**
  - Alle Korrespondenzen finden
  - Vermeidung falscher Korrespondenzen



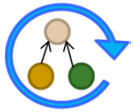
Automatische Ansätze zum Schema Matching

- Klassifikation nach Rahm/Bernstein

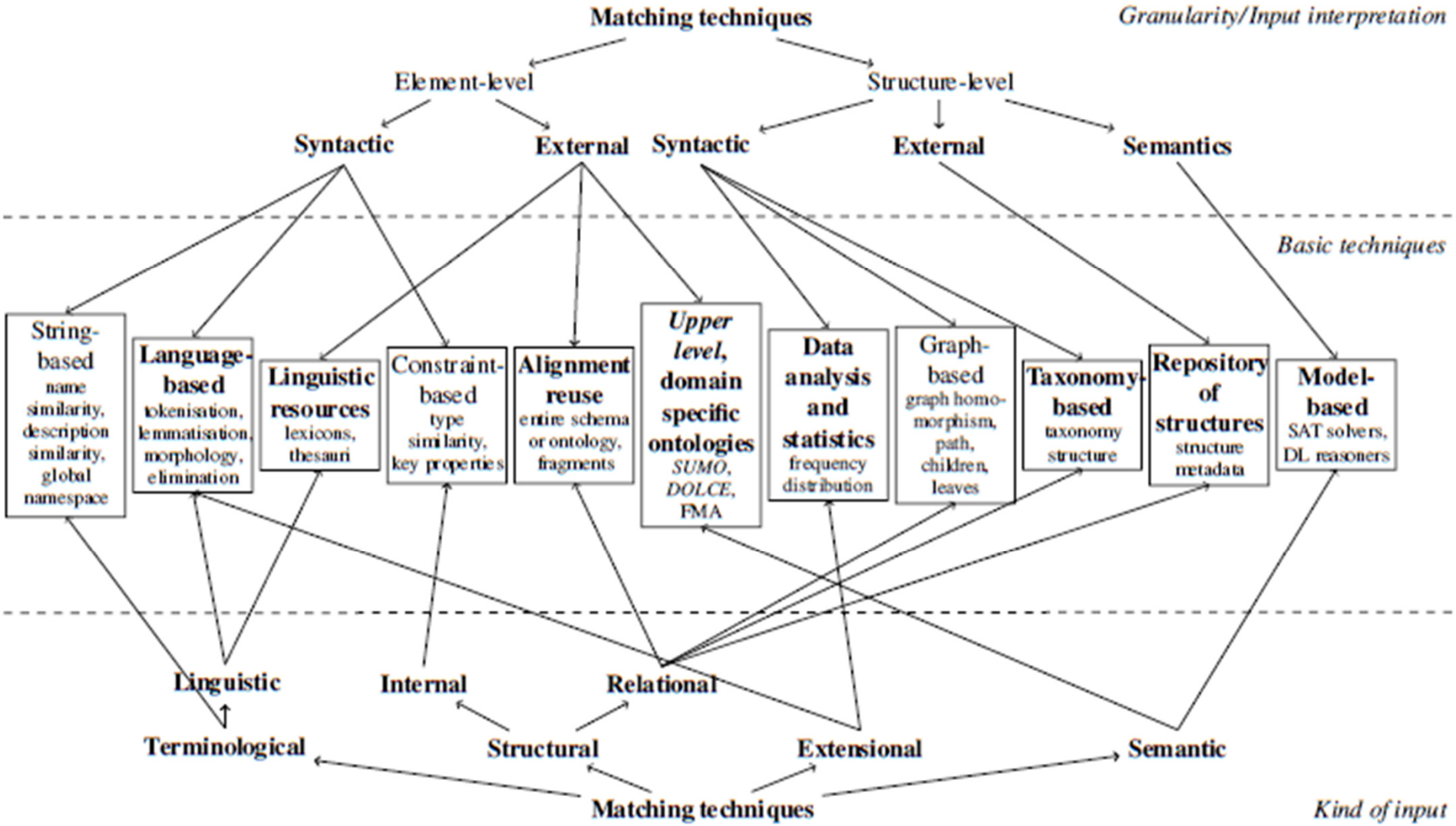


- Kombinierte Ansätze: *Composite* vs. *Hybrid*
  - *Hybrid*: ein Algorithmus wendet mehrere Verfahren an
  - *Composite*: Kombination der Ergebnisse mehrerer Verfahren

Rahm, Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal, 2001

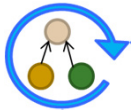


Weitere Klassifikation nach Euzenat, Shvaiko



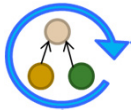
Ontology Matching: Element-based

- **Gegeben:** zwei Ontologien O1 / O2 mit ihren Konzepten sowie zugehörigen Attributen (Eigenschaften)
- **Kernidee:**
  - Bilde Kreuzprodukt aller Konzepte aus O1 und O2
  - Für jedes Paar (c1,c2) c1 aus O1 / c2 aus O2 bestimme deren Ähnlichkeit
    - Nutzung der Attributinformationen, z.B. Label, Synonyme
    - Verwendung von Ähnlichkeitsfunktionen, z.B. Edit-Distance
  - „Ähnlichste“ Paare bilden Korrespondenzen
- **Probleme:**
  - Effizienz
  - Auswahl der besten Matches (Selektionsstrategie)



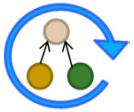
Ähnlichkeitsbestimmung über Ähnlichkeitsmaße

- **Ähnlichkeitsmaß: *sim*(e1, e2) → [0,1]**
- **Generische Ähnlichkeitsmaße**
  - Weit verbreitet, da universell einsetzbar
  - Üblicherweise auf Zeichenketten (Strings) operierend
    - Soundex, Edit-Distance, q-gram, Jaccard, Jaro-Winkler, ...
    - Kein bestes Ähnlichkeitsmaß
      - Qualität abhängig vom Match-Problem
      - Eigenschaften wie Stringlänge usw. sind zu beachten
- **Domänenspezifische Ähnlichkeitsmaße**
  - Spezielle Funktionen, welche Kontextwissen ausnutzen
    - Abstand von zwei geographischen Orten mittels Lat/Long
    - Zeitlicher Abstand basierend auf Datum
    - Einsatz von Hintergrundwissen (z.B. Synonymlisten)



Ähnlichkeitsmaße für Strings: Soundex

- Entwickelt 1918 zur Volkszählung in den USA
- Idee: Gleichklingende Wörter werden in identische Zeichenfolge übersetzt
  - Soundex-Code:
    - Erster Buchstabe gefolgt von Codes für die nächsten drei Konsonanten
    - Ähnliche Konsonanten besitzen den gleichen Code (B, F, P, V erhalten „1“)
    - Beispiel: Soundex („Robert “) = Soundex („Rupert “) = R163
- Gut
  - Beachtet Lautähnlichkeiten, oft implementiert
- Schlecht
  - Sehr heuristisch (fehlende Konsonanten? Warum nur 3? Warum keine Vokale?)
  - Nur für Namen geeignet, Fehlermodell ist „verhören“
  - Abhängigkeit von Sprache



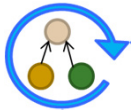
Ähnlichkeitsmaße für Strings: Edit-Distance

- Gegeben seien zwei Strings *a* und *b* (|*a*|=n, |*b*|=m)
- **Editabstand** = Anzahl der Edit-Operationen, um *a* in *b* zu konvertieren
  - Edit-Operationen: Einfügen, Löschen oder Ersetzen eines Zeichens
- Ähnlichkeit ist normierter Editabstand
  - auch: Levenshtein-Abstand

$$sim_{edit}(a,b) = 1 - \frac{dist(a,b)}{\max(|a|,|b|)}$$

- Editabstandsfunktion d(i,j) mit 0≤i≤n und 0≤j≤m
  - berechne den Editabstand zwischen *a*[1..i] und *b*[1..j]
  - d(0,j) = j (j Einfügeoperationen) und d(i, 0) = i (i Löschoperationen)
  - d(n,m) = dist(*a*,*b*) = Editabstand zwischen *a* und *b*

$$d(i,j) = \min\begin{Bmatrix} d(i,j-1)+1 \\ d(i-1,j)+1 \\ d(i-1,j-1)+t(i,j) \end{Bmatrix} \qquad t(i,j) = \begin{cases} 1 : wenn\ a[i] \neq b[j] \\ 0 : sonst \end{cases}$$

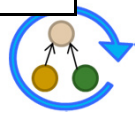


Ähnlichkeitsmaße für Strings: Edit-Distance

- Beispiel: RASEN und HASE
  - RASEN → RASE → HASE oder RASEN → HASEN → HASE
  - Edit-Abstand = 2 → String-Ähnlichkeit = 1 – 2 / max(5;4) = 0,6
- Berechnung des kürzesten Editabstandes mittels Abstandsmatrix
  - Speichern der Teillösungen (für rekursive Berechnung)
  - Initialisierung mit festen Werten d(i,0) und d(0,j)
  - Sukzessive Berechnung von d(i,j) mit steigenden i, j

$$d(i,j) = \min\begin{Bmatrix} d(i,j-1)+1 \\ d(i-1,j)+1 \\ d(i-1,j-1)+t(i,j) \end{Bmatrix}$$

		H	A	S	E
R					
A					
S					
E					
N					



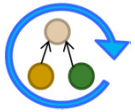
Ähnlichkeitsmaße für Strings: q-gram

- Gegeben seien zwei Strings *a* und *b* (|*a*|=n, |*b*|=m)
- Idee: Wie viele gleiche Substrings der Länge q enthalten *a* und *b*?
  - Häufig q = 3 (Trigram)
- Ähnlichkeit mittels Dice-Koeffizient

$$sim_{qgram}(a,b) = \frac{2 \cdot |Q(a) \cap Q(b)|}{|Q(a)| + |Q(b)|}$$

  - Q(*a*) = Menge der q-Gramme von *a*
  - zu vergleichende Strings erhalten Präfix und Suffix mit je q-1 Füllzeichen

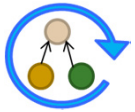
- Beispiel: RASEN und HASE mit q=3
  - RASEN →
  - HASE →
  - Ähnlichkeit =



Ontology Matching: Structure-based

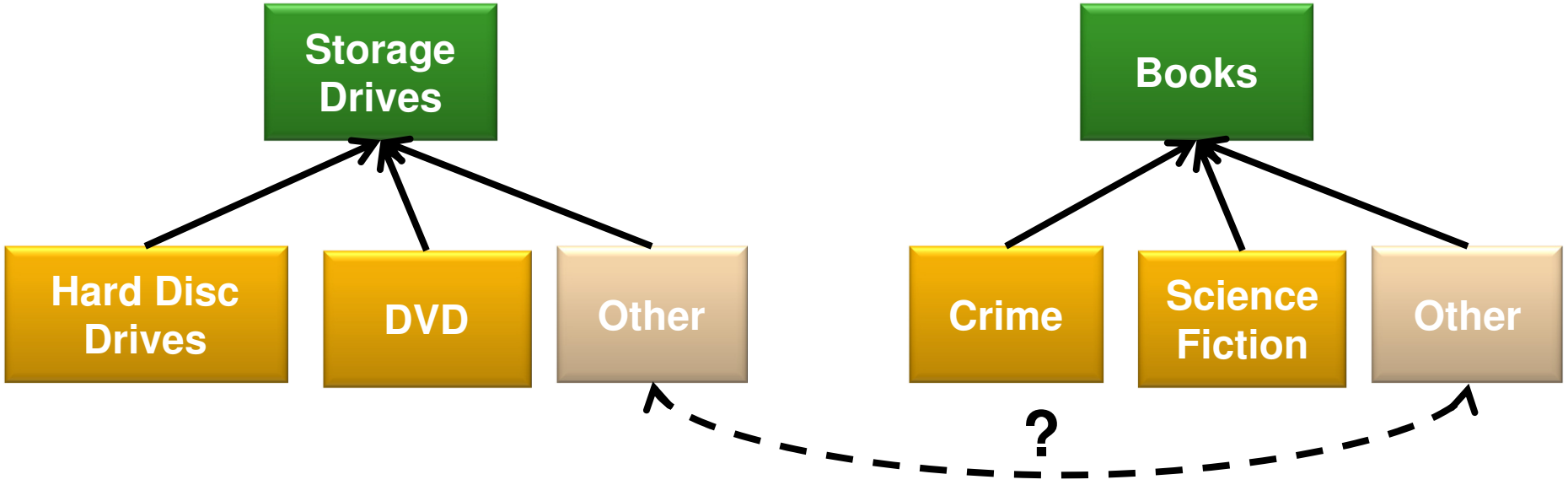
- **Gegeben:** zwei Ontologien O1 / O2 mit ihren Konzepten sowie ihre strukturelle Anordnung (is_a, part_of, ...)
- **Kernidee:** Ausnutzung der Struktur
  - Hierarchieebenen, Pfade (→ siehe Namepath)
  - Nachbarschaftsbeziehungen
- **Komplexes Beispiel: Similarity Flooding ***
  - Gegeben initiale Ähnlichkeit zwischen Schemaelementen (z.B. durch Element-based Matcher)
  - Lasse Ähnlichkeiten „abfärben“ auf die Nachbarn (über Struktur)
  - Intuition: „Sind alle / viele Nachbarn von x und y ähnlich zueinander, sind (vielleicht) auch x und y ein match.“
  - Analogie: Man „flutet“ das Netzwerk der Ähnlichkeiten bis ein Gleichgewicht erreicht ist.

** Melnik, Garcia-Molina, Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. ICDE 2002: 117-128*

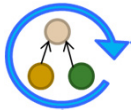


Structure: Namepath

- Oftmals Kontext nötig, um korrekte Matches zu finden (und falsche auszuschließen)
- **Idee**: Beziehe Label der Vorgänger in Berechnung ein

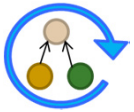


- Ohne Kontext: *sim*(Other,Other) = 1
- Namepath: *sim*(Storage Drives / Other, Books / Other) = ?

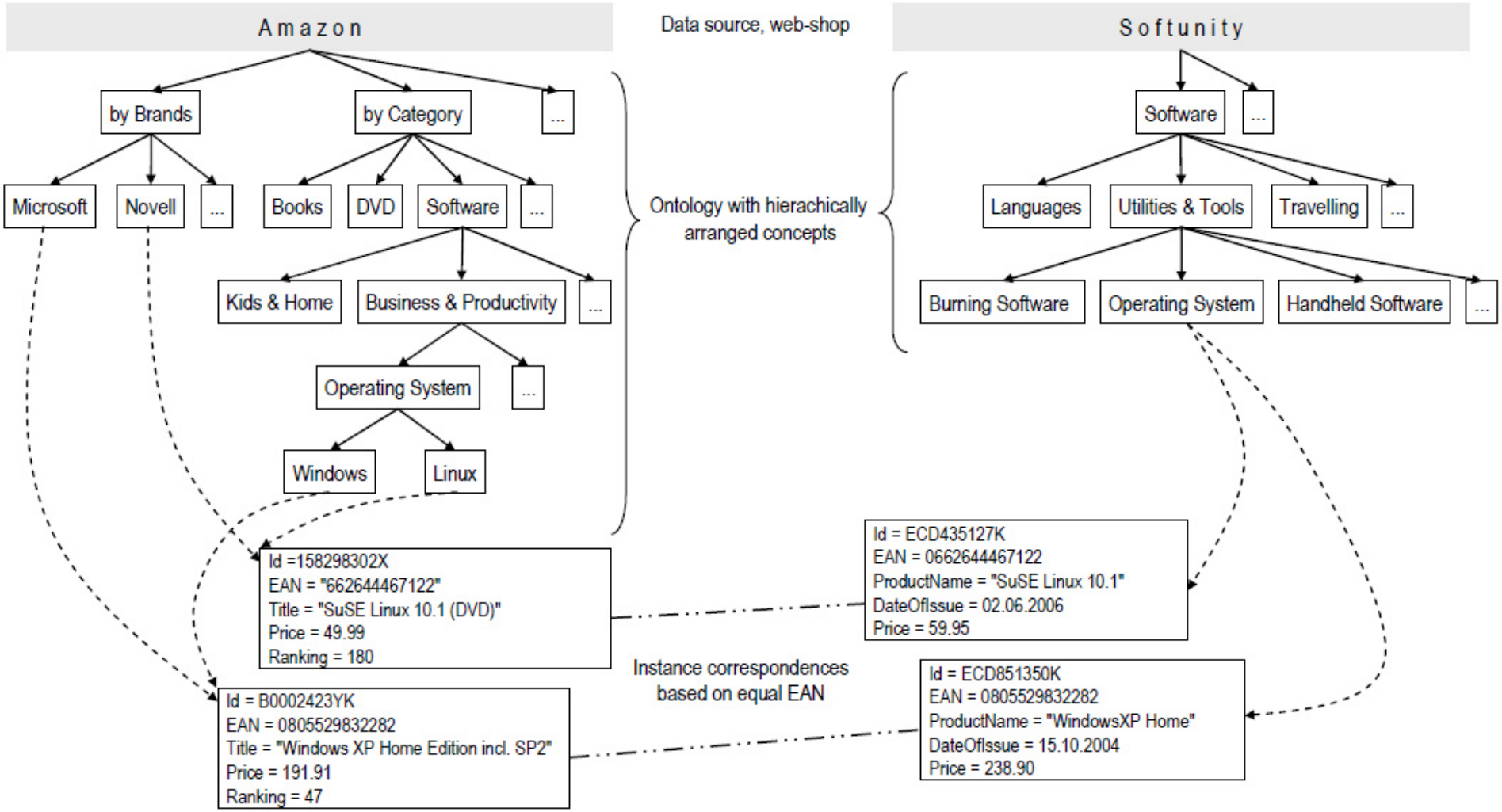


Ontology Matching: Instance-based

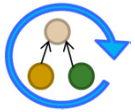
- **Gegeben:** zwei Ontologien O1 / O2 mit ihren Konzepten sowie zugehörige Instanzen oder Annotationen
- **Kernidee:** direkt Instanzen zu Konzepten verfügbar
  - Für jedes Attribut extrahiere interessante Eigenschaften der Daten
    - Buchstabenverteilung, Länge, etc.
  - Bilde Kreuzprodukt aller Konzepte
  - Für jedes Paar vergleiche Ähnlichkeit bzgl. der Eigenschaften
- **Probleme**
  - Auswahl der Eigenschaften
  - Ganze Datenmenge? (Sampling)



Instance-based mittels Annotationen



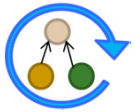
Thor, Kirsten, Rahm: Instance-based matching of hierarchical ontologies. BTW, 2007



Instance-based mittels Annotationen

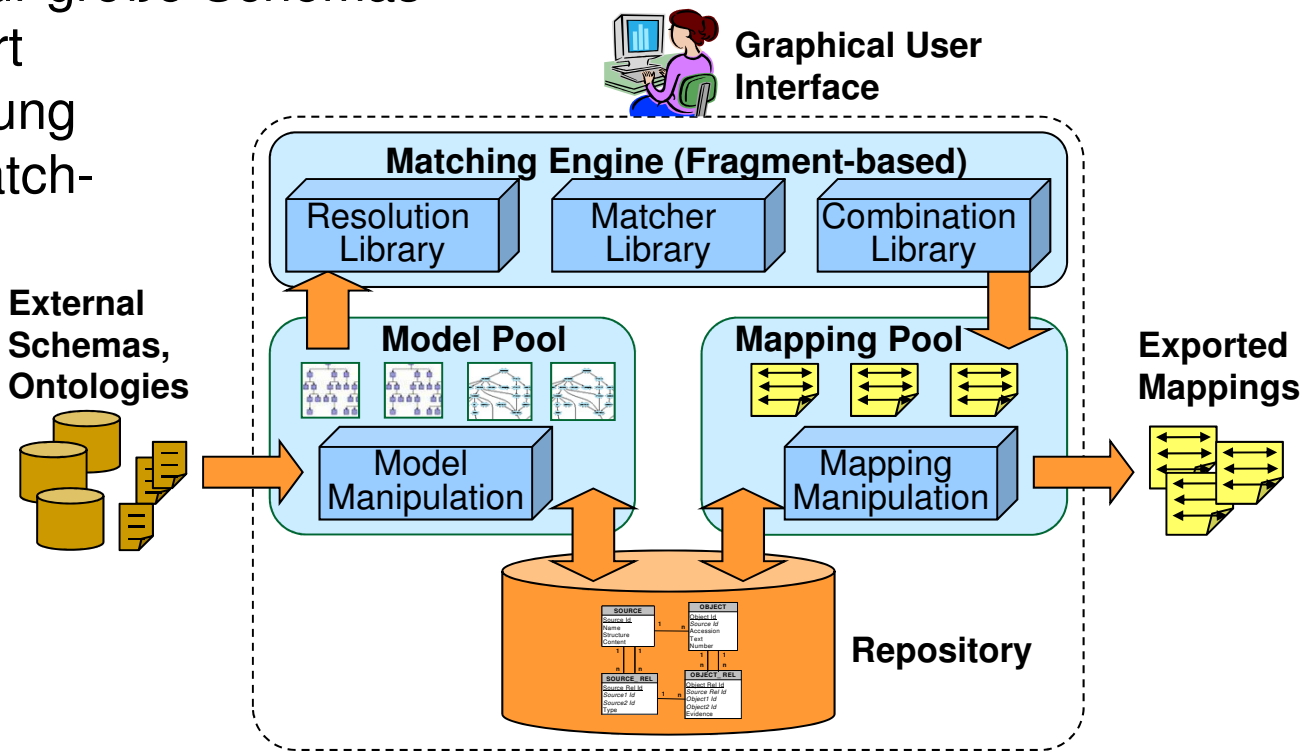
- Vers. Metriken zur Ähnlichkeitserkennung
- **Gegeben:** zwei Konzepte c1 und c2 aus O1 bzw. O2 sowie deren zugeordnete Instanzen I1 / I2

- Dice-Metrik: $$sim_{DICE}(c1,c2) = \frac{2 \cdot |I1 \cap I2|}{|I1| + |I2|}$$
- Min-Metrik $$sim_{MIN}(c1,c2) = \frac{|I1 \cap I2|}{\min(|I1|,|I2|)}$$
- Base-Metrik: $$sim_{BASE}(c1,c2) = 1, if\ |I1 \cap I2| > 0$$



COMA++ als generische Match-Plattform

- Zunächst Fokus auf Schemas (XML, relationale Schemas)
  → Ausbau auf Ontologien
- Composite-Ansatz mit flexibler Konfiguration von Matchern
- Match-Strategien für große Schemas
  - Fragment-basiert
  - Wiederverwendung vorhandener Match-Ergebnisse
- Umfassende GUI
- Neueste Version COMA 3.0

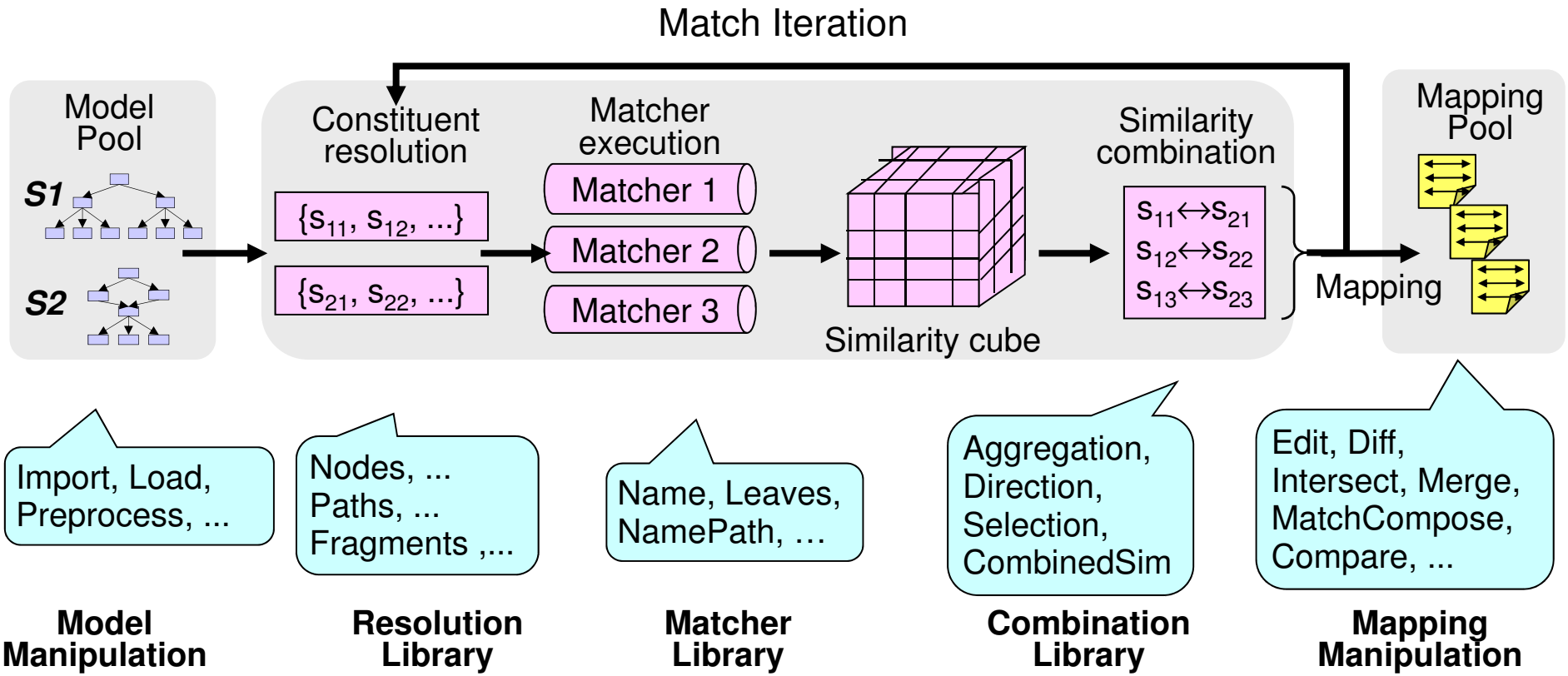


Do, Rahm: COMA - A System for Flexible Combination of Schema Matching Approaches. VLDB 2002

Aumüller, Do, Massmann, Rahm: Schema and Ontology Matching with COMA++. Sigmod 2005

Match-Verarbeitung in COMA++

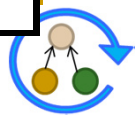
- Ausführung verschiedener Match Strategien
- Manipulation / Kombination erzeugter Mappings



COMA++: Matcher Bibliothek

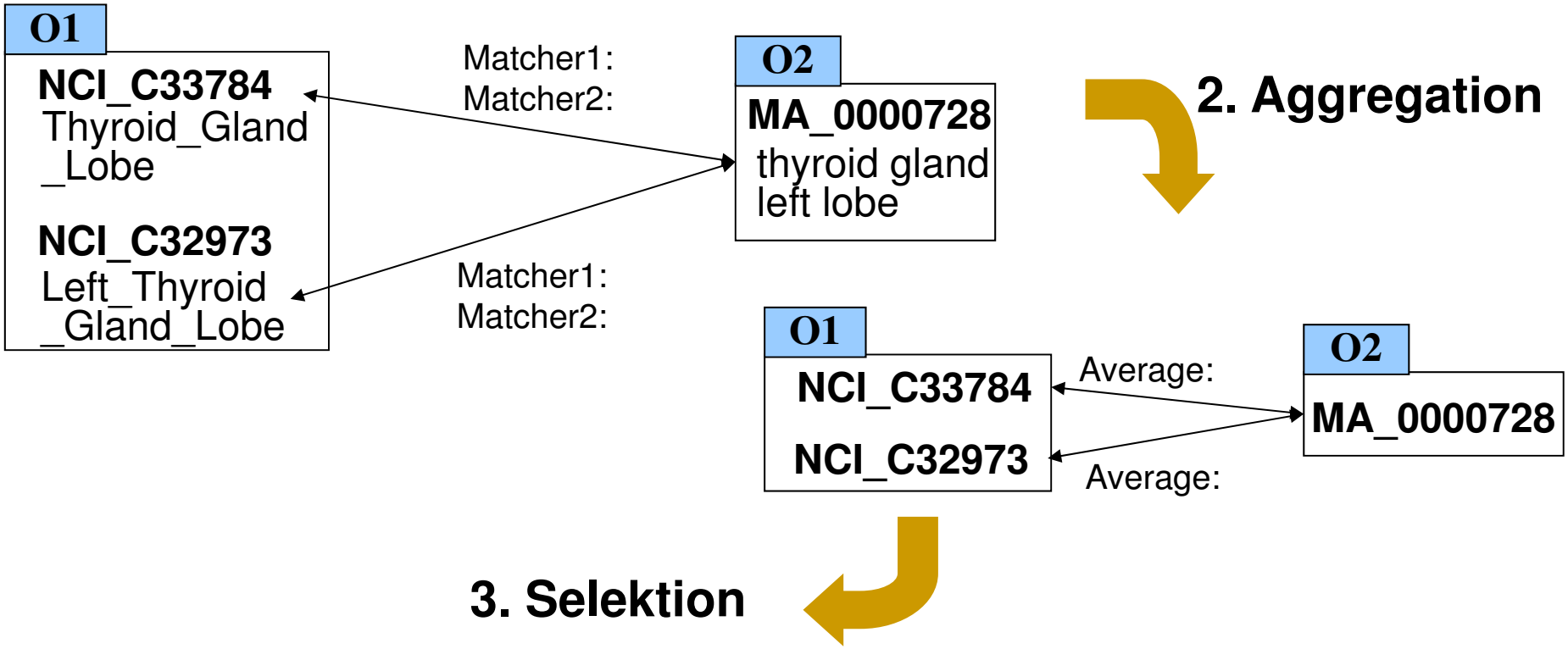
- Basis Matcher
  - String-Matcher: Synonym, Type, Trigram, Affix, Edit-Distance
  - Type-Matcher
  - Taxonomie-Matcher
  - Reuse-Matcher: Wiederverwendung von Mappings
- Hybrid-Matcher: feste Kombination anderer Matcher

Name	*Constituents*	*Matchers/Sim measures*	*Combination*
Name	Name tokens	Synonym/Taxonomy, Trigram	Avg, Both, Max1, Avg
NameType	Node	Name, Type	Wgt(0.7,0.3), Both, Max1, Avg
NameStat	Node	Name, Statistics	
Children	Children	NameType	Avg, Both, Max1, Avg
Leaves	Leaves	NameType	
Parents	Parents	Leaves	
Siblings	Siblings	Leaves	
NamePath	Ascendants	Name	



Kombination von Matchergebnissen

1. Matcher-Ausführung



Max1

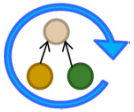
O2 concepts	*O1 concepts*	*Sim*

Threshold(0.5)

O2 concepts	*O1 concepts*	*Sim*

Selektionsstrategien

- Einfacher Schwellwert (Threshold) → *sim1(c1,c2)* > t
- Max1, MaxN
  - Nur die beste (die besten n) Korrespondenz(en) für jedes Konzept *c*
- MaxDelta
  - Beste Korrespondenz plus weitere innerhalb eines Deltas
- Regeln
  - *sim1(c1,c2)* > 0.6 ∧ *sim2(c1,c2)* > 0.7 → match(c1,c2)
- Globale Optimierung
  - Stable Marriage, Maximum Weighted Matching
- Kombination von Mappings
  - Vereinigung (Union), Durchschnitt (Intersect), Mehrheitsentscheid (Majority)

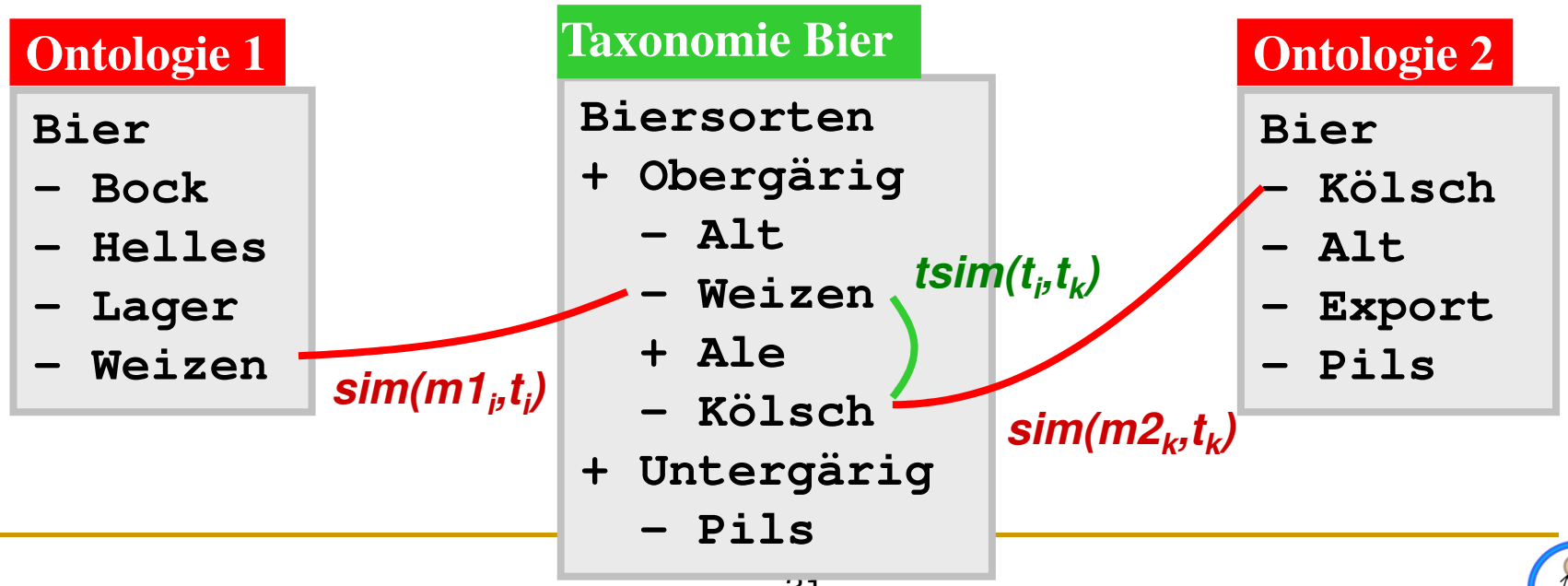


Taxonomie-Matcher

- Taxonomie (Ontologie) als Referenz zwischen zwei Ontologien (z.B. Standard-Terminologie)

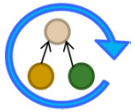
```
sim(Weizen,Kölsch) = 0.8
```

- Ähnlichkeit zweier Schema-Elemente → Kombination aus *sim(m,t)* und *tsim(t,t)*:
  - lexikalischen Ähnlichkeiten der Schema-Elemente *mit* der Taxonomie
  - semantische Ähnlichkeit durch Distanz der Begriffe *innerhalb* der Taxonomie (verschiedene Heuristiken denkbar)

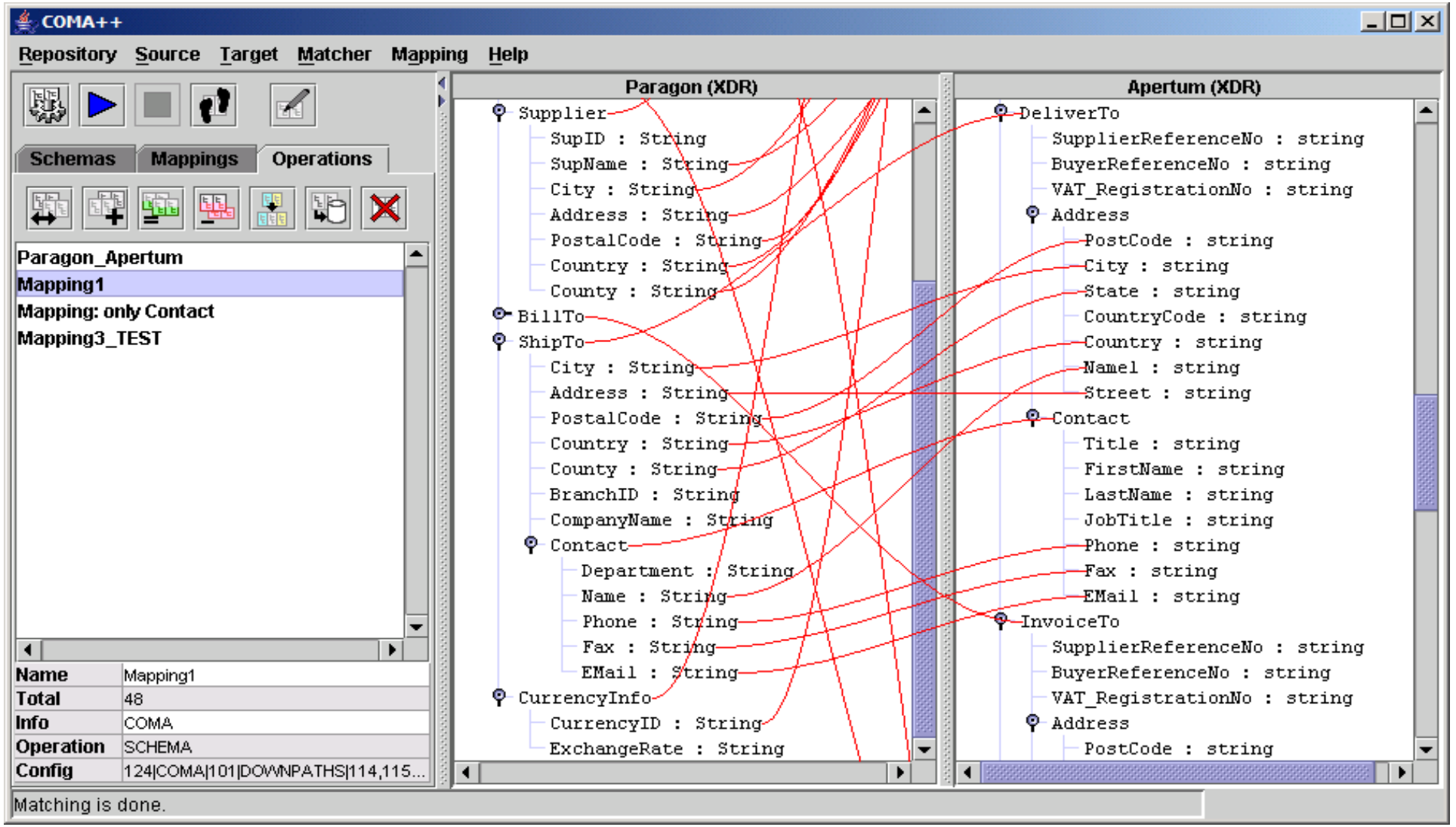


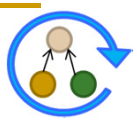
Wiederverwendung / Reuse

- Nutzung von Hilfsquellen
  - Nutzerspezifizierte Synonymtabellen, allgemeine/ domänenspezifische Vokabulare, Wörterbücher, gemeinsame Ontologien
- Nutzung bereits bestätigter Match-Ergebnisse für ähnliche Match-Probleme
  - Speichern von Ontologien und Mappings in Repository
  - Ausnutzung der Transitivität
    - Berechnung eines Mappings zwischen O1 und O2 falls bereits Mappings O1-O3 sowie O3-O2 existieren
    - Auch längere Mapping-Pfade möglich
    - Später genauer: Large-Scale Ontology Matching



Nutzerschnittstelle COMA++





Evaluierung von Matching-Verfahren

- Bewertung mittels Precision / Recall / F-Measure
  - Qualitätsmaße aus dem Information Retrieval

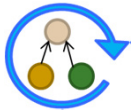
		Realität	
		Match	No Match
Verfahren	Match	true-positive	false-positive
	No Match	false-negative	true-negative

- Precision = TP / (TP + FP)
  - Wie viele der als Match identifizierten Korrespondenzen sind korrekt? (auch Spezifität)
- Recall = TP / (TP + FN)
  - Wie viele aller Matches hat die Methode überhaupt gefunden? (auch Sensitivität)
- F-Measure = 2*Precision*Recall / (Precision + Recall)
- Beispiel:
  - Verfahren: {(a1,b1),(a2,b3),(a3,b2)}
  - Realität: {(a1,b1),(a2,b2)}

Precision:

Recall:

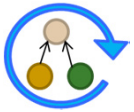
F-Measure:



Evaluierung von Matching-Verfahren

- Effizienz und Skalierbarkeit
  - Wie lange benötigt das Matching-Verfahren, um ein Mapping zu erzeugen? (einfache Zeitmessung)
  - Anzahl nötiger Vergleiche zur Erstellung eines Mappings
  - Welche Grundkomplexität besitzt das Verfahren?
  - Werden neuartige Hardwarearchitekturen ausgenutzt (z.B. Multi-Threading)?
  - Skalierbarkeit mit großen Ontologien
    - Können überhaupt große Ontologien verarbeitet werden?
    - Gibt es Obergrenzen (z.B. Speicherengpässe)?
- Ontology Alignment Evaluation Initiative (OAEI) *
  - Zahlreiche Benchmarks zum Testen von Verfahren
  - Jährlicher Wettbewerb mit Fokus auf Qualität sowie Effizienz

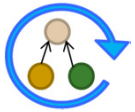
** OAEI: http://oaei.ontologymatching.org/*



Large-Scale Ontology Matching

Problem

- Trotz Automatisierung teilweise enormer Rechenaufwand für Erzeugung von Mappings erforderlich
- Grundkomplexität ist typischerweise quadratisch O(m·n)
  - Vergleiche alle Konzepte aus O1 mit allen aus O2 (kartesisches Produkt)
  - Vervielfachung durch Anwendung mehrerer Matcher in Workflows sowie durch Komplexität des Matchers selbst
  - *Beispiel:* NCI Thesaurus (90.000) – Foundational Model of Anatomy (70.000) → 90.000 · 70.000 = 6,3 Mrd. Vgl.
- OAEI Evaluierung
  - Teilweise mehrere Stunden bzw. Tage !!
- Für interaktive Systeme unzureichend

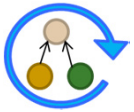


Large-Scale Ontology Matching

Generelle Lösungsansätze

- Suchraumreduzierung vor Matcher-Anwendung
  - Unnötige Vergleiche eliminieren („pruning“)
  - Nur ähnliche Teilbereiche (Partitionen) vergleichen
- Paralleles Matching
  - Durchführung der Matcher-Anwendung auf mehreren CPUs / Rechenknoten bzw. Cloud-Infrastrukturen
- Self-Tuning von Match-workflows
  - Erlernen von optimalen Konfigurationen anstatt manuelles Setting
- Wiederverwendung / Reuse
  - Bereits bekannte Mappings zur Neuberechnung heranziehen

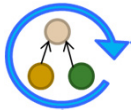
Rahm: Towards Large-Scale Schema and Ontology Matching, Springer, 2011.



Suchraumreduktion

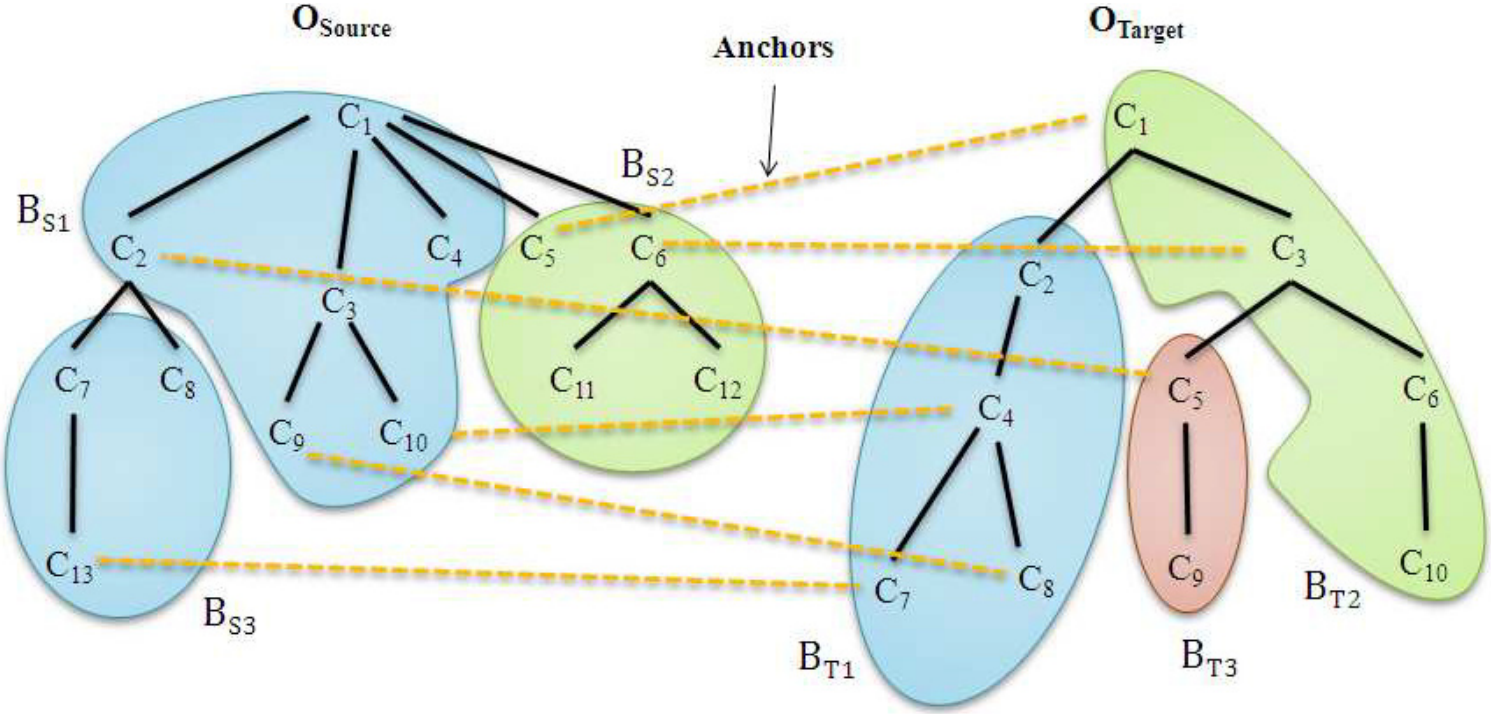
Zwei generelle Ansätze

- Vermeide kartesisches Produkt durch Eliminierung unnötiger Vergleiche im Vorfeld
  - Anwendung eines „weniger“ komplexen Matchers und Ausgrenzung aller Paare mit niedriger Ähnlichkeit von weiteren Berechnungen → gut bei sequentiellen Workflows
- Partition-based Matching (Blocking)
  - „Teile-Herrsche Prinzip“: Teile Ontologien in Partitionen auf und vergleiche nur Partitionen, welche sich stark ähneln
  - Beispiel für zwei Anatomie-Ontologien
    - „nervous system“ ≠ „body fluid or substance“
    - „cardiovascular system“ = „circulatory system“



Partition-based matching (Falcon-AO, Taxomap)

- Strukturelles Clustering der Ontologien für Partitionierung
- Ähnlichkeit von Partitionen über Anker (*anchors*)
  - Anker: sehr stark ähnelnde Konzepte



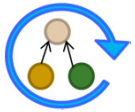
Hamdi, Safar, Reynaud, Zargayouna: Alignment-based partitioning of large-scale ontologies. Advances in Knowledge Discovery and Management, 2009.

Paralleles Ontology Matching

Grundprinzip

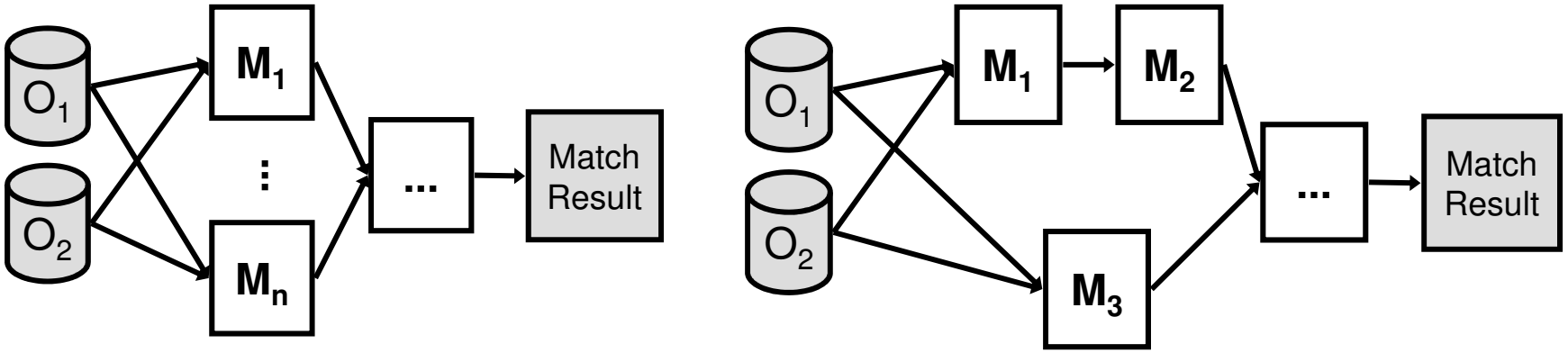
- Verteile Match-Anwendung auf mehreren CPUs / Rechenknoten
  - „Kill it with iron“-Technik
  - Neueste Hardware sehr gut Multi-Threading fähig (Dual / Quad Core CPUs) jedoch oft nicht ausgenutzt
- Zwei prinzipielle Möglichkeiten der Verteilung *
  - Verteilung auf Matcher-Ebene (*inter parallelization*)
    - Eine CPU führt jeweils einen Matcher aus
  - Parallelisierung einzelner Matcher (*intra parallelization*)
    - Eine CPU evaluiert lediglich Teil des kartesischen Produkts
  - Kombination aus beiden möglich (*inter-intra*)

** Groß, Hartung, Kirsten, Rahm: On matching large life science ontologies in parallel. Data Integration in the Life Sciences, 2010.*



Inter Matcher Parallelization

Parallele Ausführung unabhängig ausführbarer Matcher

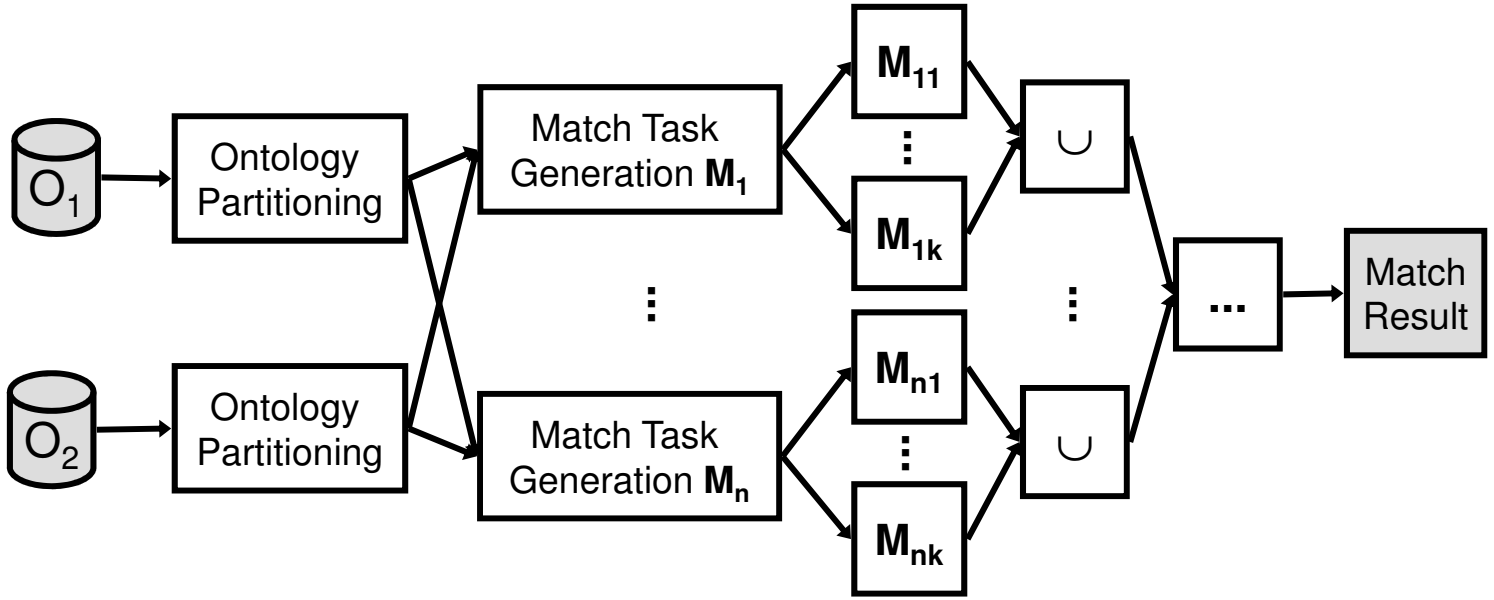


- Kann Laufzeit bis auf das n-fache reduzieren (n = |Matcher|)
- Anzahl parallel ausführbarer Matcher limitiert Parallelisierung
- Langsamster Matcher „bremst“ speed up
- Speicheranforderungen bleiben (keine Datenpartitionierung)

Intra Matcher Parallelization

Generierung von Match-Teilaufgaben zur Ausführung auf mehreren CPUs

- *Basis:* Partitionierung der Ontologien

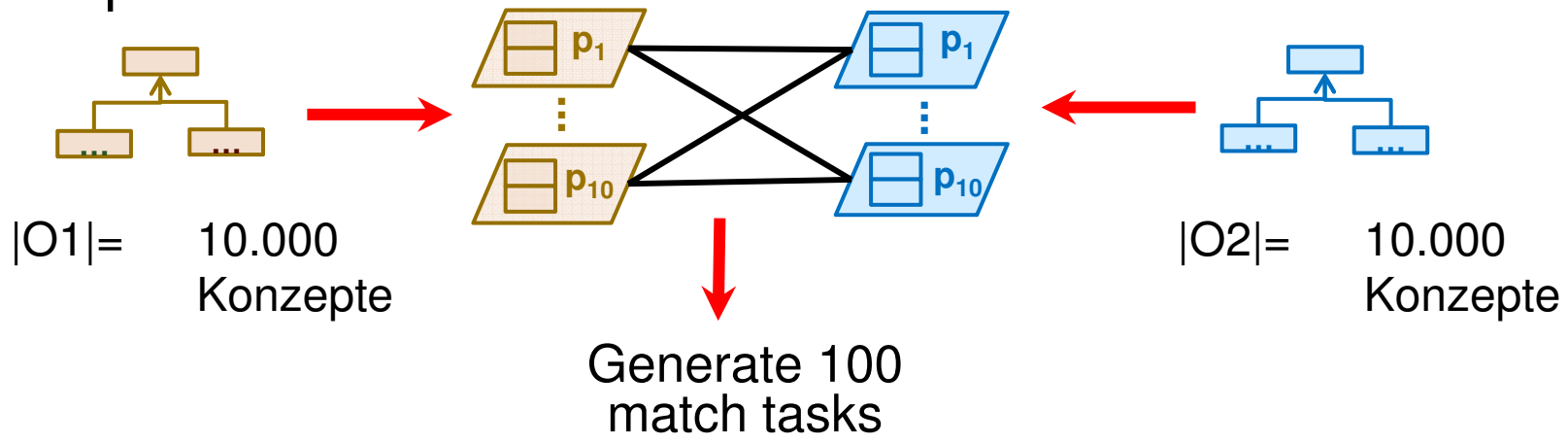


- Reduktion der Speicheranforderungen
- Auch für sequentielle Match-Workflows anwendbar

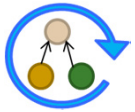
Partitionierung der Ontologien

Aufteilung der Ontologien in Partitionen gleicher Größe (Anzahl Konzepte)

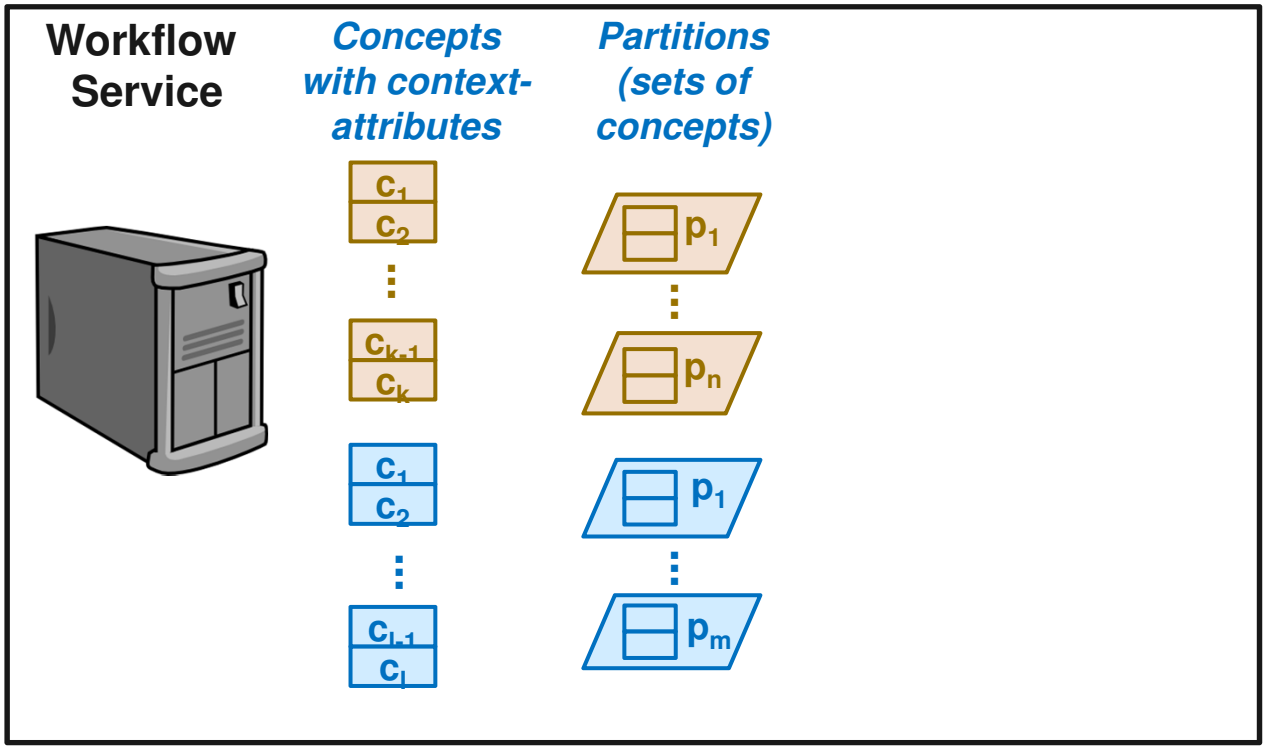
- Jeder Match-Task vergleicht eine Partition aus O1 mit einer Partition aus O2
- Beispiel:

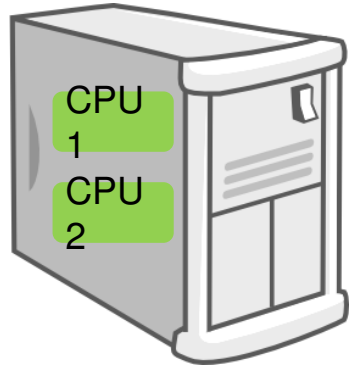


  - Mit großen Ontologien skalierbar (Lastbalancierung)
  - Alle Vergleiche werden ausgeführt (→ Match-Qualität)
  - Für verschiedene Matcher anwendbar

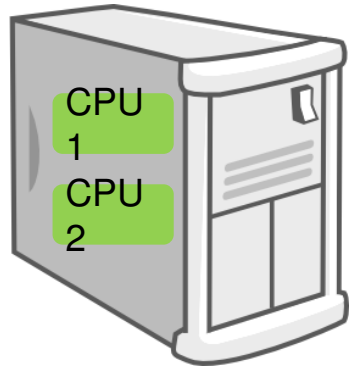


Infrastruktur und Ausführungsbeispiel

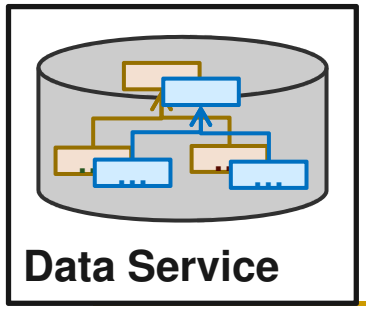




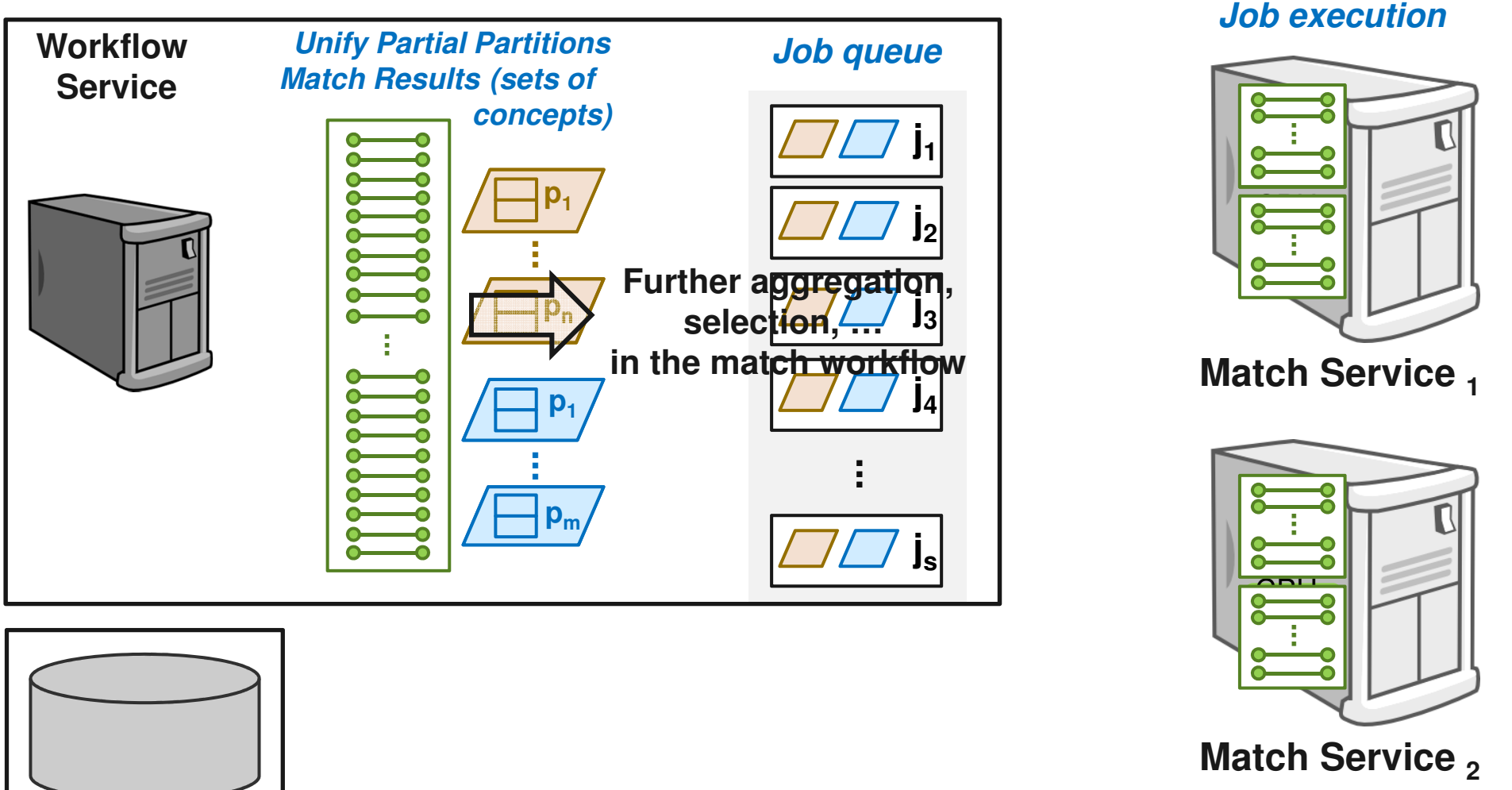
Match Service $_1$



Match Service $_2$



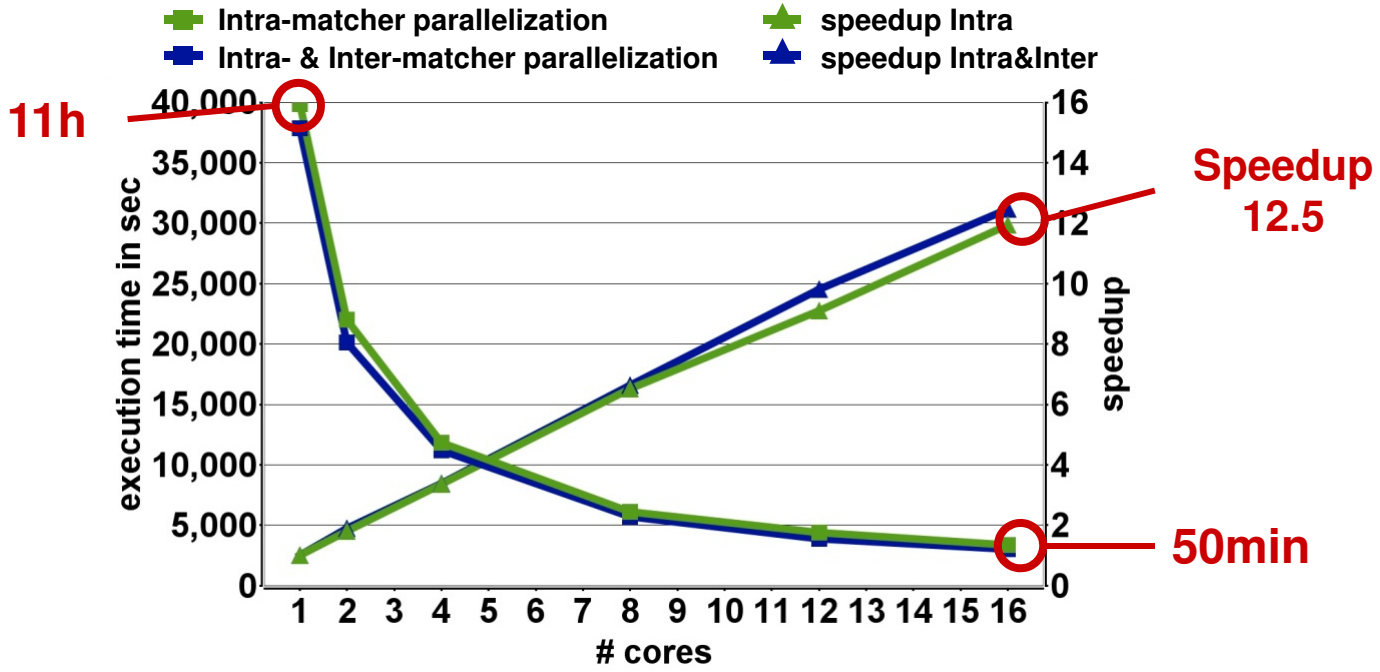
Infrastruktur und Ausführungsbeispiel



Evaluierung

Match-Problem: Molekulare Funktionen (~10.000) vs. Biologische Prozesse (~20.000) der Gene Ontology

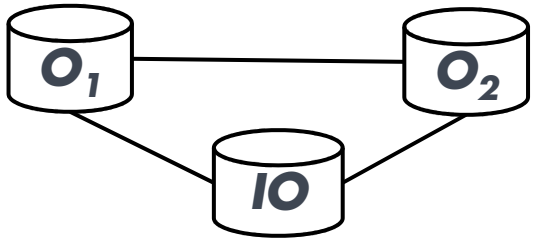
- 4 Rechenknoten mit jeweils 4 CPUs
- Kombination von 3 Matchern: Name/Synonym, NamePath, Children

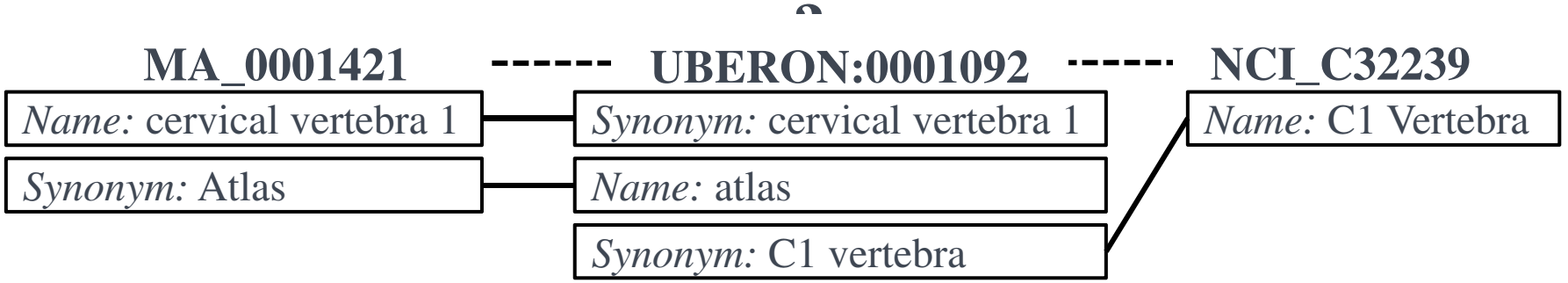


Komposition von Mappings

Indirektes (composition-based) Matching

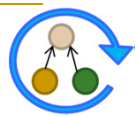
- Ausnutzung bereits existierender Mappings zu einer Mediator/Intermediate Ontologie (IO)
  - Hub (wichtige zentrale Terminologie)
  - Synonym-Verzeichnis



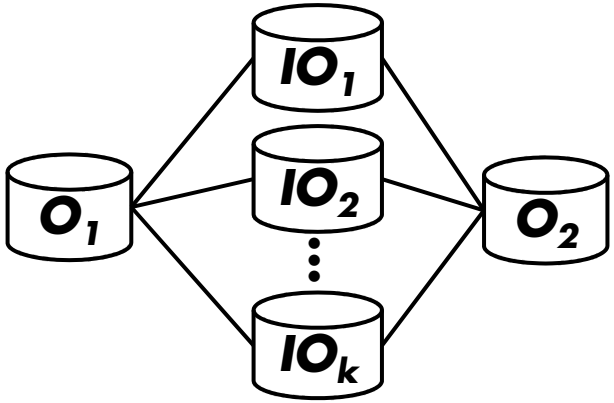


- Vorteile
  - Erkennung neuer Korrespondenzen durch Komposition
  - Einsparung von Rechenaufwand (kein kartesisches Produkt)

Groß, Hartung, Kirsten, Rahm: Mapping Composition for Matching Large Life Science Ontologies. Intl. Conference on Biomedical Ontology, 2011.

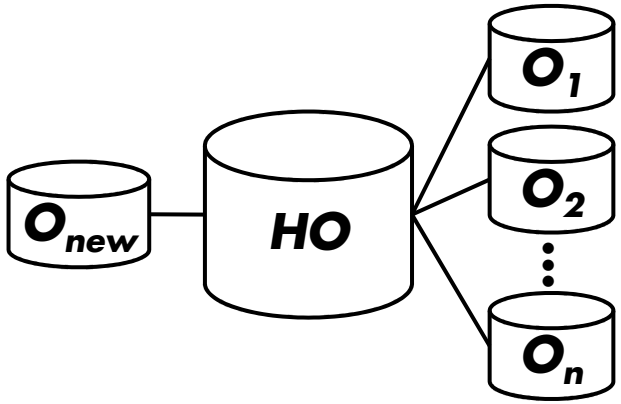


Möglichkeiten von Komposition



Komposition über mehrere Mediatorontologien

- IOs sollten starke Ähnlichkeit mit O1 und O2 aufweisen
- Gegenseitige Ergänzung ausnutzen



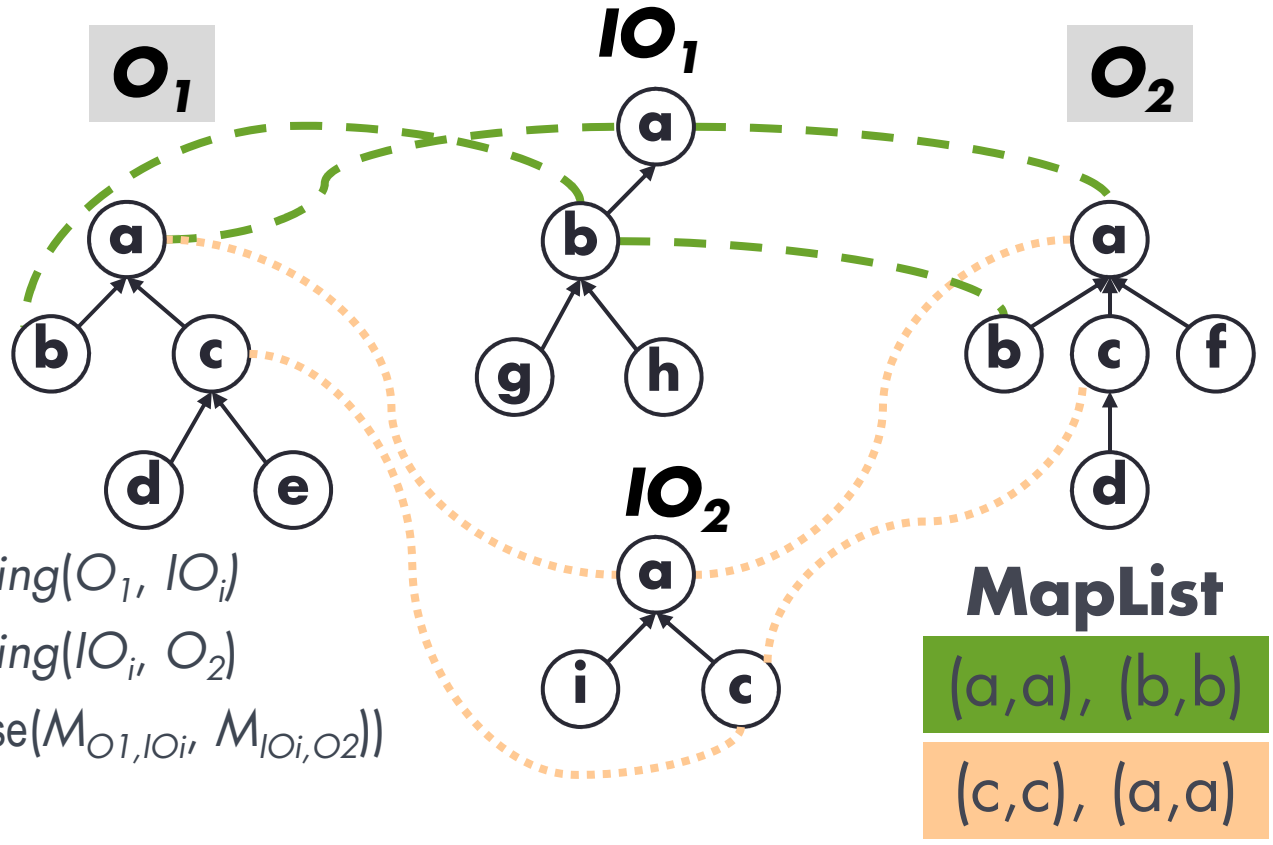
Zentraler Hub vorhanden

- Hub besitzt bereits zahlreiche Mappings zu anderen Ontologien
- Effizientes Matching einer neuen Ontologie zu allen anderen möglich

ComposeMatch

Eingabe: Ontologien O_1 und O_2, Liste von Mediatoren $IO_1 \ldots IO_k$, minimales Vorkommen *occ*

Ausgabe: Mittels Komposition erzeugtes Mapping $CM_{O1,O2}$



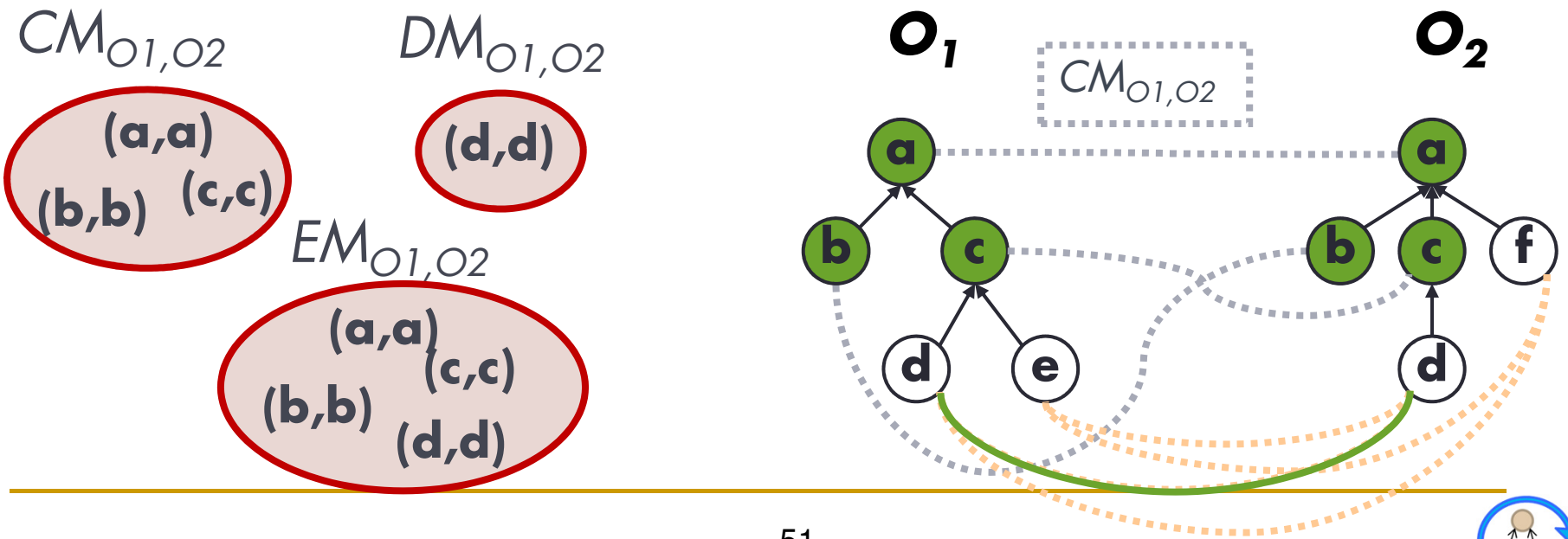
MapList ← empty
for each $IO_i \in IO$ **do**
  $M_{O1,IOi}$ ←*getMapping*(O_1, IO_i)
  $M_{IOi,O2}$ ←*getMapping*(IO_i, O_2)
  MapList.add(compose($M_{O1,IOi}$, $M_{IOi,O2}$))
end for
return merge(*MapList*, *occ*)

occ = 1: $CM_{O1,O2}$ = {(a,a),(b,b),(c,c)}
occ = 2: $CM_{O1,O2}$ = {(a,a)}

ExtendMatch

Eingabe: Ontologien O_1 and O_2, Mapping durch Komposition $CM_{O1,O2}$
Ausgabe: Komplette Mapping $EM_{O1,O2}$

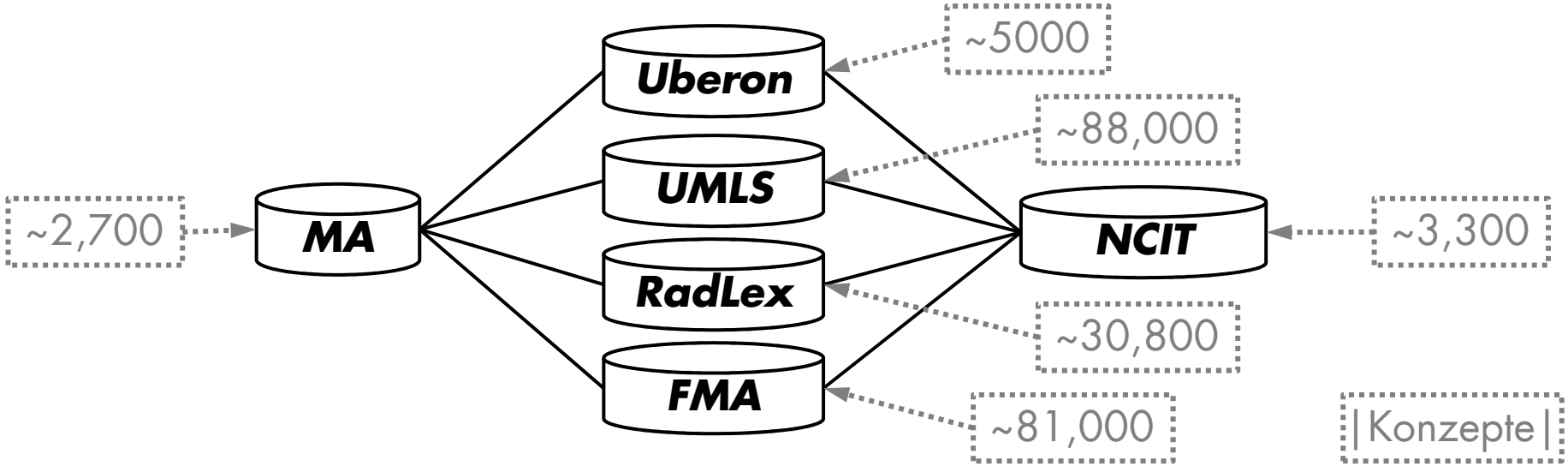
$\Delta O_1 \leftarrow \text{extract}(O_1, CM_{O1,O2})$
$\Delta O_2 \leftarrow \text{extract}(O_2, \text{inverse}(CM_{O1,O2}))$
$DM_{\Delta O1 \Delta O2} \leftarrow \text{match}(\Delta O_1, \Delta O_2)$ *//Herkömmlicher Match*
$EM_{O1,O2} \leftarrow \text{merge}(\{CM_{O1,O2}, DM_{\Delta O1 \Delta O2}\}, 1)$
return $EM_{O1,O2}$



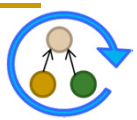
Evaluierung

Anatomie Match-Problem von OAEI

- NCI-Thesaurus (Anatomieteil) - Adult Mouse Anatomy

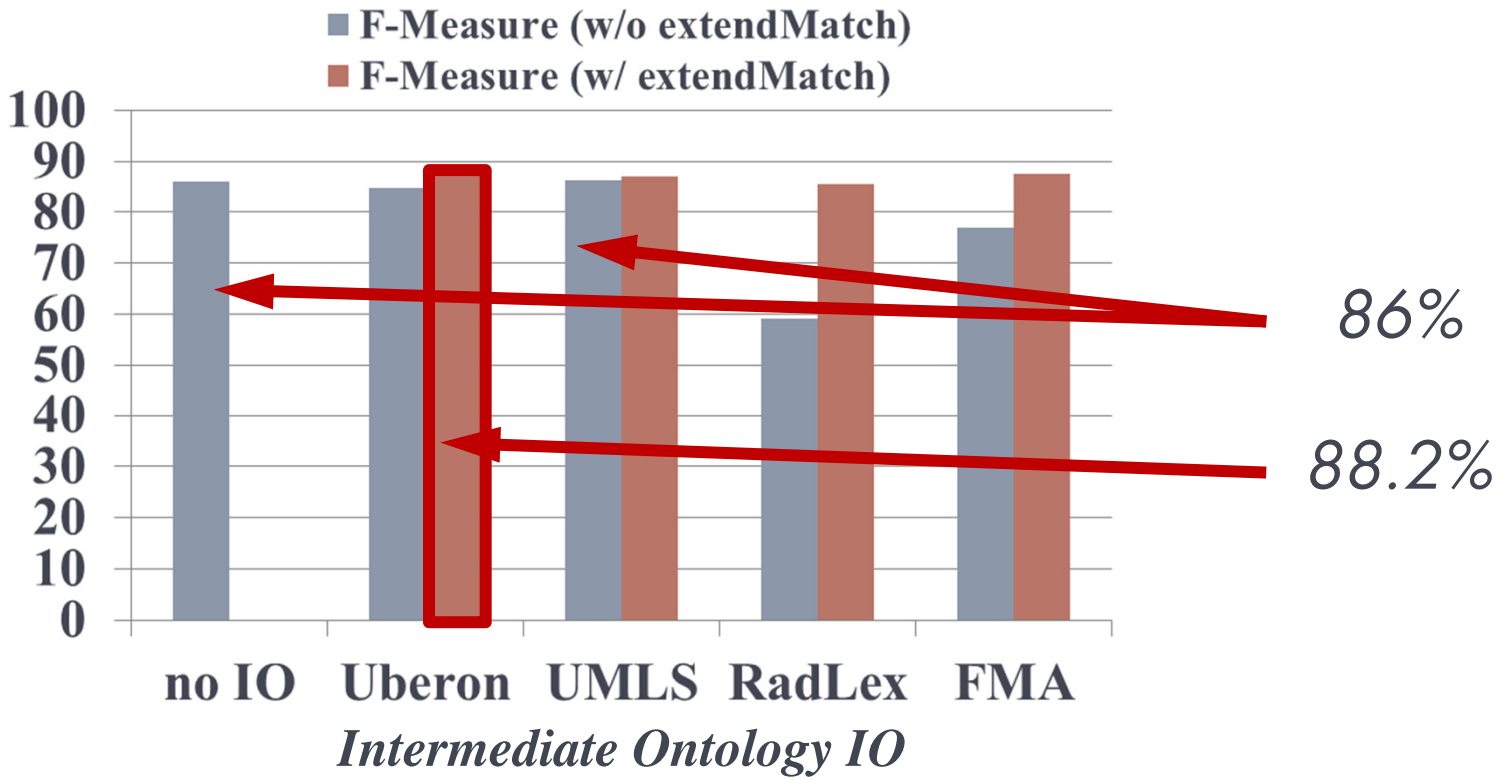


- Perfektes Mapping: ca. 1.500 Korrespondenzen
- Andere Mappings MA-IO, IO-NCIT vorberechnet



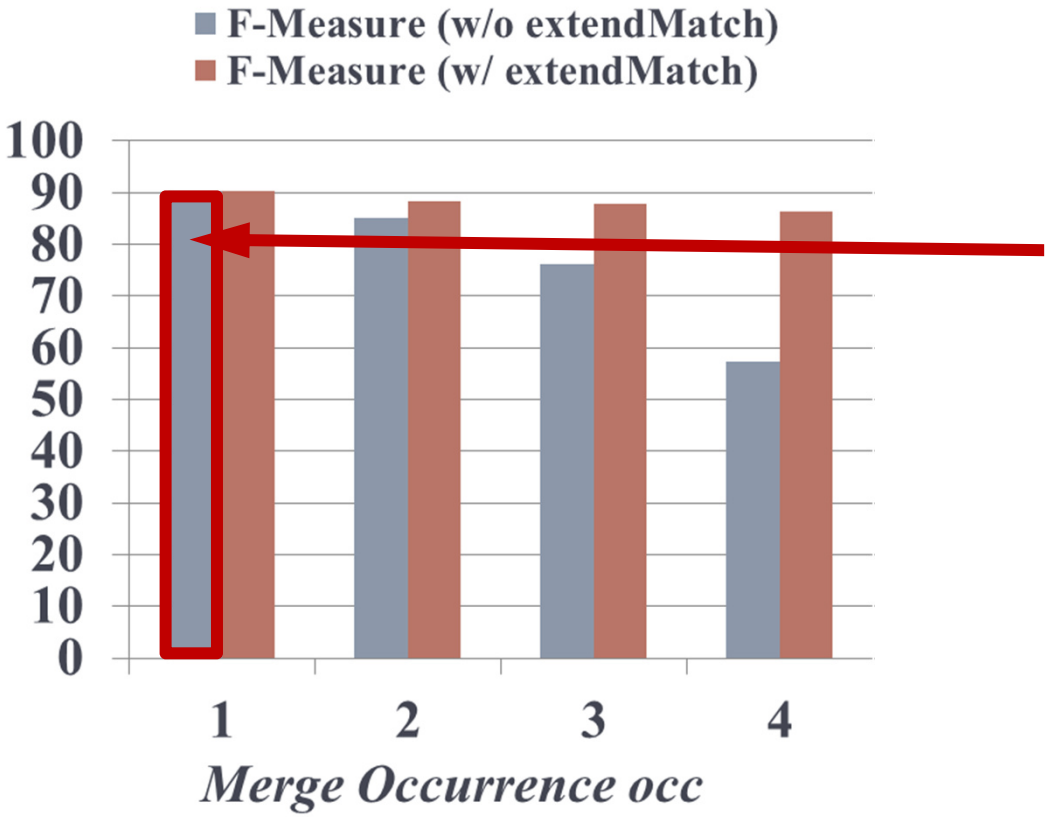
Komposition über einen Mediator

- Vergleich mit direktem Match-Ergebnis
- Zusätzliches Matching der nicht durch *composeMatch* abgedeckten Teile (*extendMatch*)



Komposition über alle Mediatoren

- Testen verschiedener Mehrheitsentscheidungen
  occ = 1, ..., 4



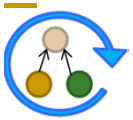
union(occ=1)

F-Measure	**90.2**
Precision	92.7
Recall	87.8

GOMMA @ OAEI 2012 *

- GOMMA System (Generic Ontology Matching and Mapping Management)
  - Spezialisierung auf das Matchen großer Ontologien
  - Parallelisierung auf mehreren CPU Cores
  - Anwendung von Mapping-Komposition und Blocking
- OAEI 2012
  - 21 Teilnehmer / 53 Tests (Qualität u. Laufzeit als Kriterien)
  - Tracks: Benchmark, Conference, Anatomy, LargeBio, Library, Multifarm
- Hauptergebnisse
  - Alle Tasks erfolgreich absolviert
  - Sieger bzgl. Qualität in Anatomy und Library
  - Stets unter Top-Systemen bzgl. Laufzeit

** Groß, Hartung, Kirsten, Rahm: GOMMA Results for OAEI 2012. Ontology Matching Workshop @ ISWC, 2012.*



Zusammenfassung

- **Ontology Matching**
  - Wichtiger Prozess zur (semi-)automatischen Erstellung von Mappings zwischen Ontologien
    - Anwendungen: Anfrageverarbeitung, Datenintegration, ...
  - Vers. Ansätze (element-, structure-, instance-based)
  - Kombination von Ansätzen zur Erhöhung der Match-Qualität
- **Large-Scale Ontology Matching**
  - Reduzierung der Komplexität des Match-Problems
  - Vers. Techniken
    - Pruning, Blocking
    - Parallel Matching
    - Reuse, Komposition