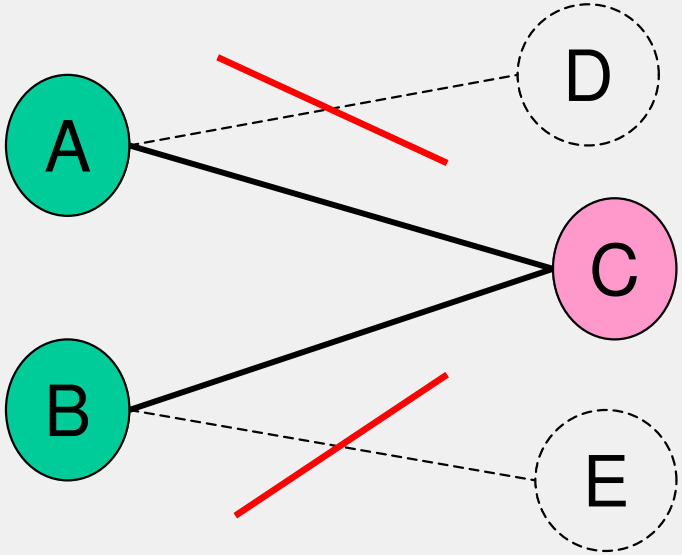
Discovering high-level changes between versions of structured data sources



Michael Hartung

Interdisciplinary Centre for Bioinformatics and Database Research Group Leipzig
University of Leipzig

Lab Seminar WS 09/10, December 2009

IZBI

UNIVERSITÄT LEIPZIG

Motivation

- **Evolving data sources in different domains**
  - new/revised knowledge, correction of design errors, …
- **Unstructured**
  - protein data sources
  - literature databases
- **Structured**
  - ontologies in life sciences
  - product catalogs, web directories in the web
- **Problem**
  - Providers only release new versions
  - Changes (diff) between old and new version are usually missing
  - But diff is required in dependent applications, e.g., for data migration or enhanced analysis/experiments

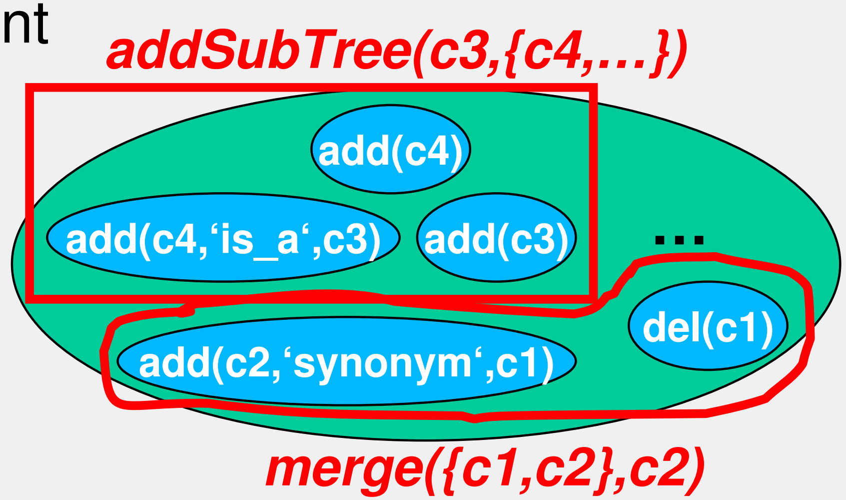
Crucial step: computation of changes/diff between an old and new data source version

Motivation

- **Diff in previous work on evolution**
  - *DILS 2008*: common model for ontologies, annotations and ontology mappings – *add*, *del*, *toObsolete*
  - *BMC Bioinformatics 2009*: inclusion of *merge* for concepts
  - *OntoContent 2009*: usage of information about *added/deleted* elements for efficient versioning of ontologies

- **Issues not addressed yet**
  - set of simple changes is not understandable for human users
  - applications need semantically richer diff between versions
  - inverse of changes as an important point

→ ***Need for a generic approach to detect invertible high-level changes between data source versions***

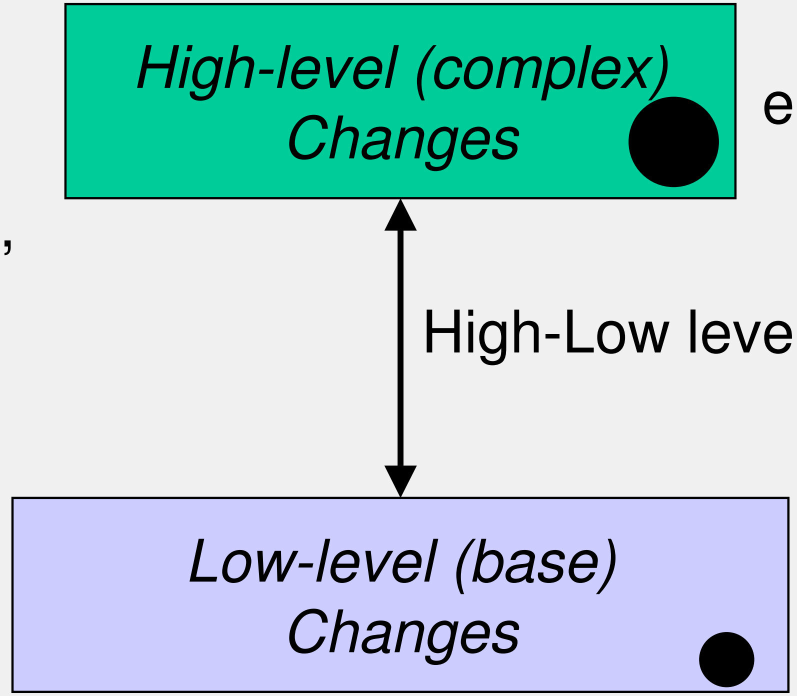


Agenda

- Motivation
- **Overview**
- **Approach**
  - Data model
  - Low- and high-level changes
  - Rules and Rule Engine
- **Evaluation results**
- **Summary and next steps**

How should the final result look like?

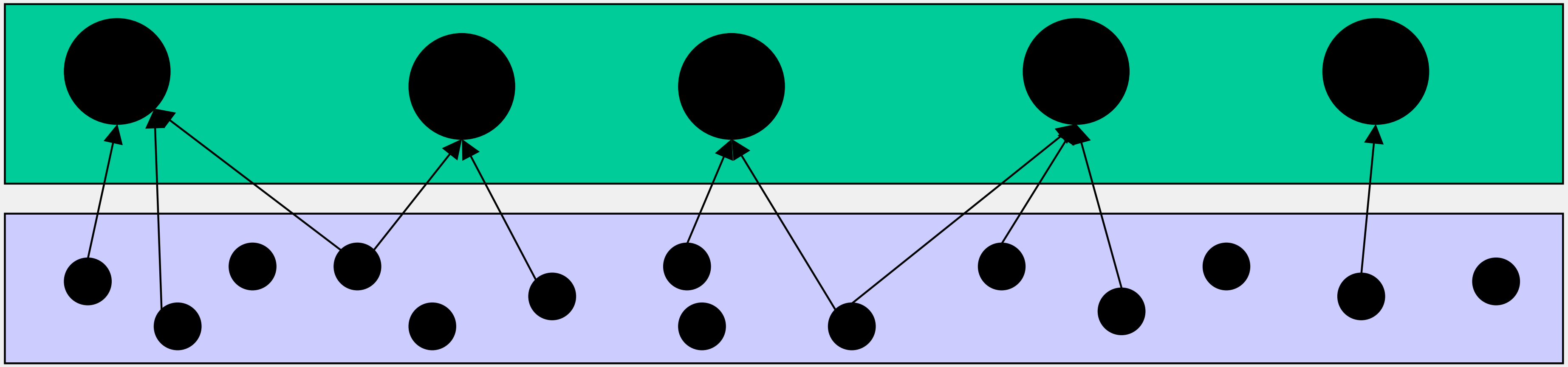
Human readable
For complex analysis
and investigations,
better understanding of evolution,
more intuitive



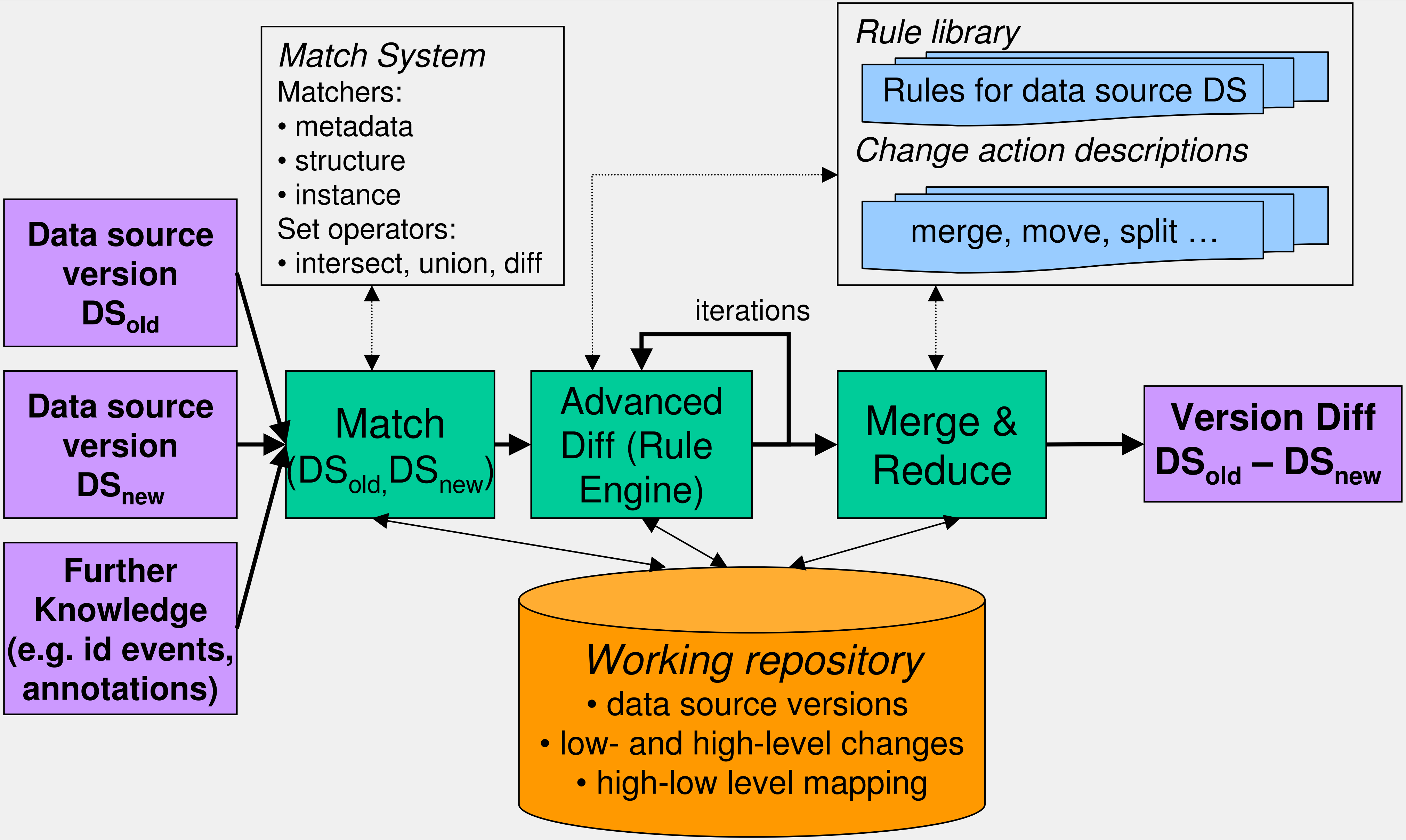
e.g., merge, …

e.g., add, …

Machine processable
For conversions (e.g.,
old to new version), base for
high-level changes



Schematic overview



Data Model

- data source *DS* in different versions $DS_1, \ldots, DS_n$ which can be compared, e.g., the diff between DS_6 and DS_7

$\boldsymbol{DS_i = (O, Att, Ass)_i}$

- ***O:*** set of objects *obj = (accession/key)* which are identified by an accession/key (given or created)
- ***Att:*** set of attributes *att = (obj, att_name, att_value)* which belong to the objects available in DS_i
- ***Ass:*** set of associations *ass = (obj_source, ass_type, obj_target)* which associate objects of DS_i through a specific *ass_type*

Example: Gene Ontology of June 2007 → $GO_{2007\text{-}07} = (O, Att, Ass)_{2007\text{-}07}$

- Objects: *GO:0005515, …*
- Attributes: *(GO:0005515, name, protein binding), …*
- Associations: *(GO:0005515, is_a, GO:0005488), …*

Low-level changes

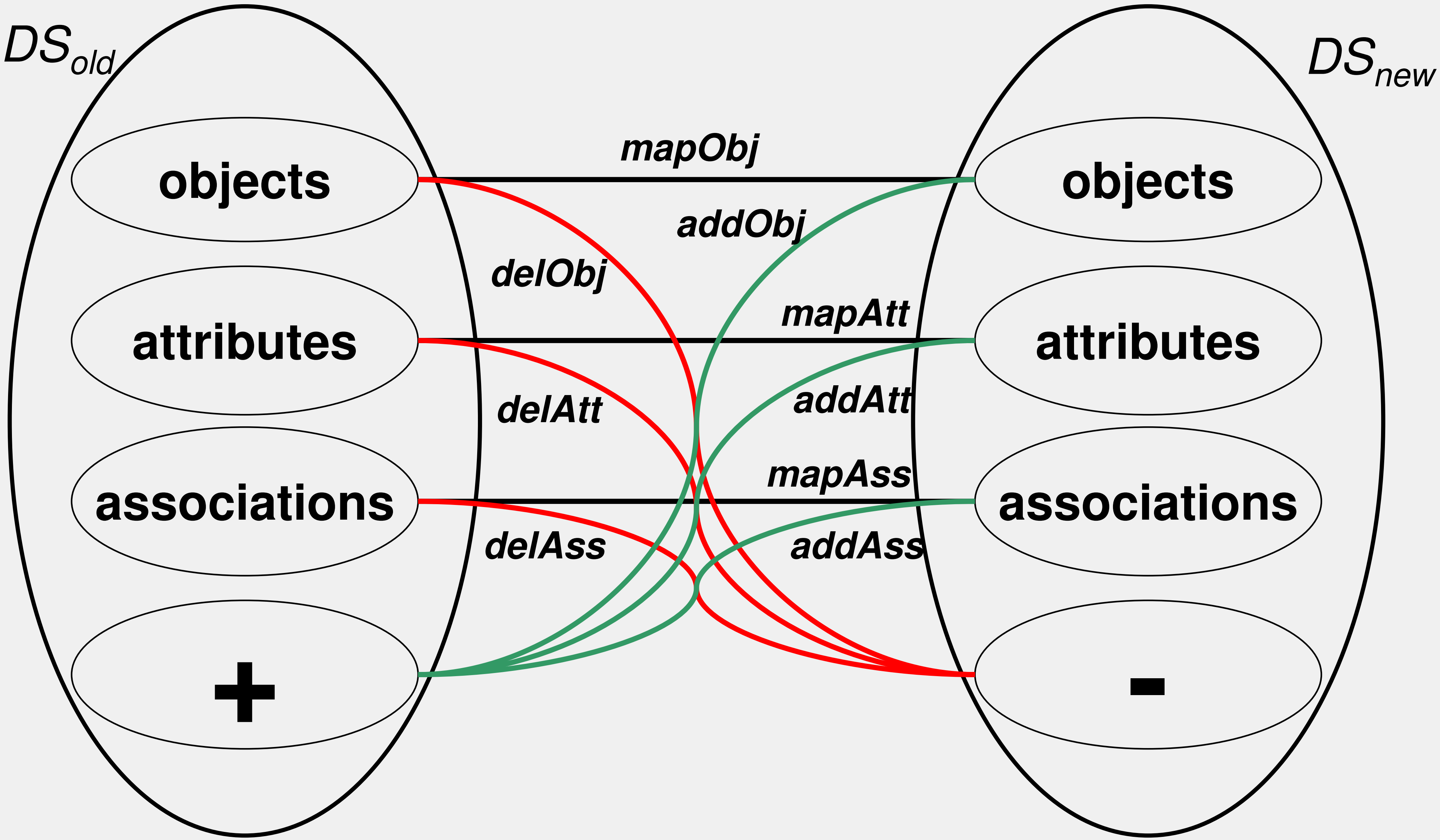
Low-level change: ***lc = name(<param>)***

- *Objects: mapObj(obj,obj)*, *addObj(obj)*, *delObj(obj)*
- *Associations: mapAss(ass,ass)*, *addAss(ass)*, *delAss(ass)*
- *Attributes: mapAtt(att,att)*, *addAtt(att)*, *delAtt(att)*
- Can be applied on data sources which support base operations such as *insert*, *delete* and *update* (e.g., relational database)
- Inverses are predefined:

Inverse:

mapObj(obj1,obj2)	*mapObj*(obj2,obj1)
addObj(obj)	*delObj*(obj)
mapAss(ass1,ass2)	*mapAss*(ass2,ass1)
addAss(ass)	*delAss*(ass)
mapAtt(att1,att2)	*mapAtt*(att2,att1)
addAtt(att)	*delAtt*(att)

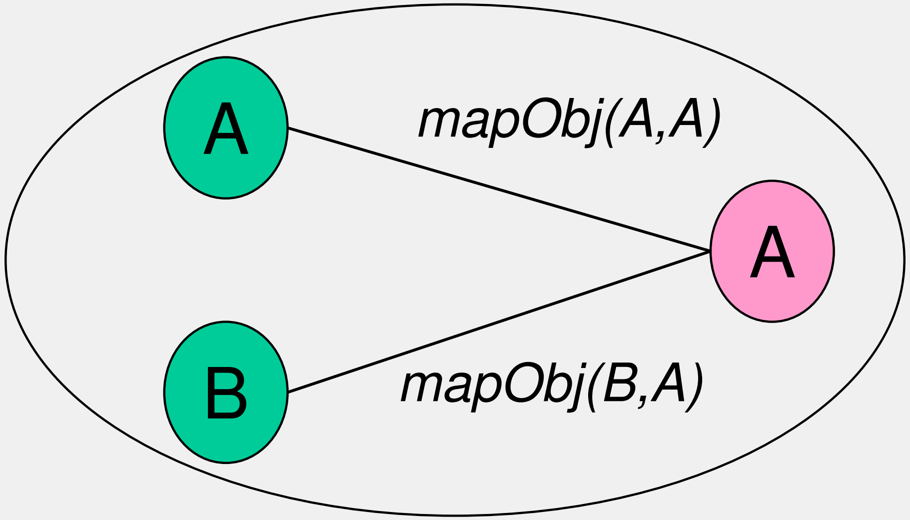
Low-level changes between two versions



High-level changes

High-level change: ***hc = name(<param>,<lc>)***

- Name and parameters, e.g., *merge*(*from*,*to*,<*lc*>)
- Parameter can be single- or multi-valued
- Set of associated low-level changes (*lc*) that lead to this high-level change

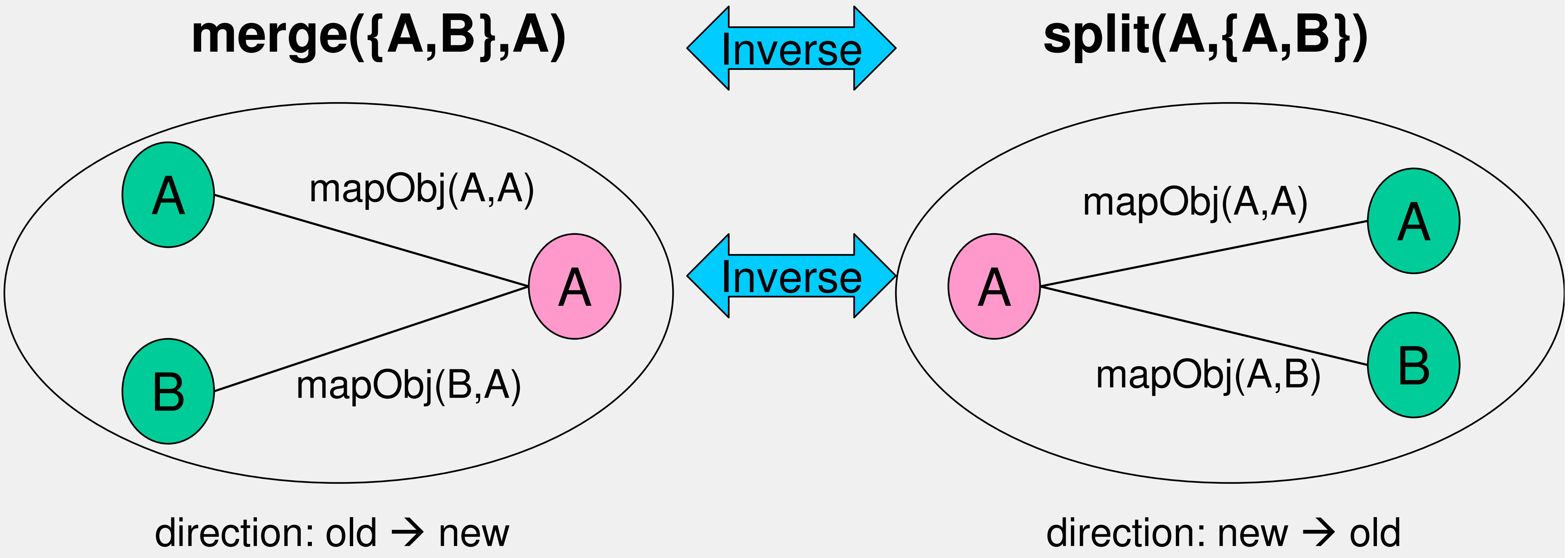


→ *merge({A,B},A,{mapObj(A,A), mapObj(B,A)})*

- Can usually not be performed directly on data sources → mapping to a set of corresponding low-level changes is required (high-low level mapping)

High-level changes - Inverse

- Inverse for high-level changes, e.g., **inverse**(*merge*(*from*,*to*)) = *split*(*to*,*from*)
- Inverse of a high-level change is realized by the inverse of its associated low-level changes



Identifying low-level changes

Match
(DS_{old}, DS_{new})

Input: data source *DS* of two versions (*old*, *new*) → DS_{old}, DS_{new}
Output: low-level changes (*map*, *add*, *del*) for all element types (*object*, *attribute*, *association*)

1. *Step:* Apply match to identify common elements
   Result: mapObj, *mapAtt*, *mapAss*

2. *Step:* Integration of further knowledge (e.g., id events)
   Result: extensions to *mapObj*, *mapAtt*, *mapAss*

3. *Step:* Computation of added and deleted elements based on DS_{new} and DS_{old} and the matched elements found so far
   Result: addObj, *delObj*, *addAtt*, *delAtt*, *addAss*, *delAss*

Identifying low-level changes (Script)

computeLowLevelChanges(DS_{old}, DS_{new})

- \$mapObj = *matchObj*(DS_{old}, DS_{new}) U {external ones}
- \$mapAtt = *matchAtt*(DS_{old}, DS_{new}) U {external ones}
- \$mapAss = *matchAss*(DS_{old}, DS_{new}) U {external ones}

Find same or modified elements

- \$addObj = DS_{new}.objects \ *range*(\$mapObj)
- \$delObj = DS_{old}.objects \ *domain*(\$mapObj)

- \$addAtt = DS_{new}.attributes \ *range*(\$mapAtt)
- \$delAtt = DS_{old}.attributes \ *domain*(\$mapAtt)

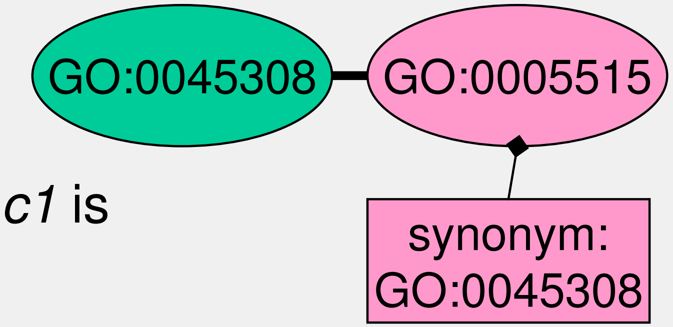
- \$addAss = DS_{new}.associations \ *range*(\$mapAss)
- \$delAss = DS_{old}.associations \ *domain*(\$mapAss)

Ensures that each element is at least covered by one low-level change → ***completeness***

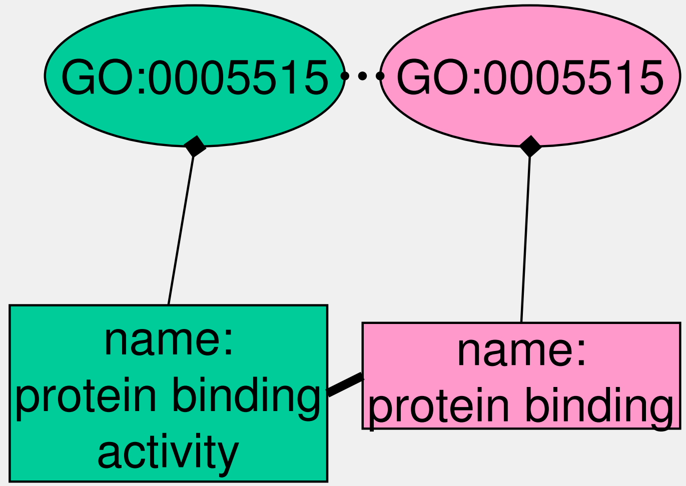
return [\$mapObj, \$mapAtt, \$mapAss,
\$addObj, \$delObj, \$addAtt,
\$delAtt, \$addAss, \$delAss]

Example match strategy for Gene Ontology

- **matchObj**
  - *Uses:* Concept accession number + synonyms
  - *Strategy:* two concepts (*c1*,*c2*) match if they have an equal accession number or the accession number of *c1* is available in a synonym of *c2*

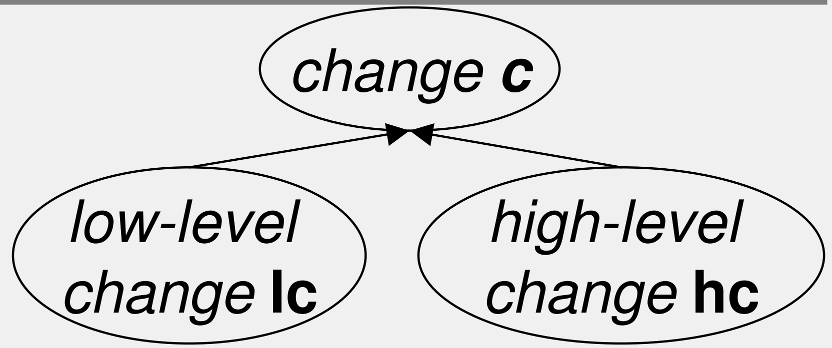


- **matchAtt**
  - *Uses:* Attribute name + corresponding accession number
  - *Strategy 1:* two attributes ((*c1*,*a1*,*v1*), (*c2*,*a2*,*v2*)) match if *c1*=*c2* and *a1*=*a2* (for single value attributes like name, obsolete, definition)
  - *Strategy 2:* two attributes ((*c1*,*a1*,*v1*), (*c2*,*a2*,*v2*)) match if *c1*=*c2* and *a1*=*a2* and *v1*=*v2* (for attributes with multiple values like synonym)



- **matchAss**
  - *Uses:* Accession numbers of associated concepts
  - *Strategy:* two relationships ((*c1*,*r1*,*c2*), (*c3*,*r2*,*c4*)) match if *c1*=*c3* and *r1*=*r2* and *c2*=*c4*

Rules for High-Level Changes



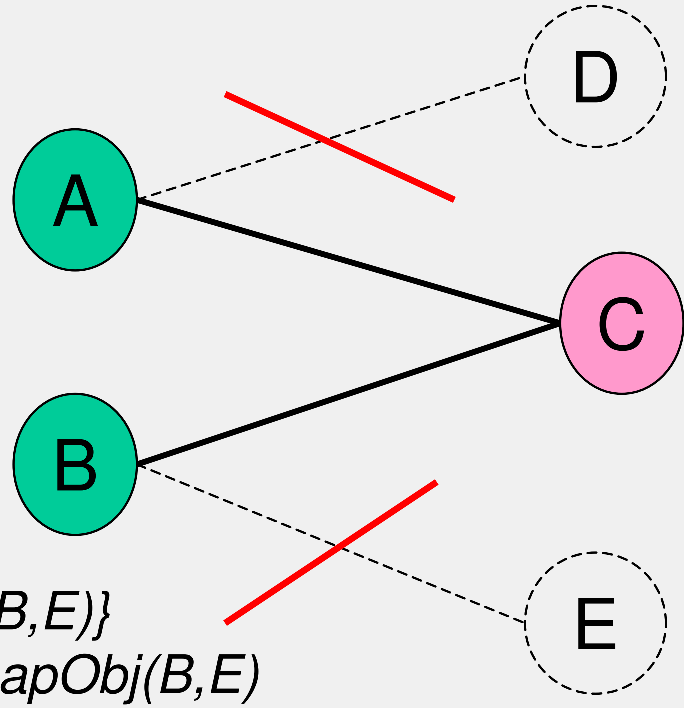
Rules R

- ***Idea:*** generate high-level changes based on existing changes + constraints
- ***Rule r*** **:** set of changes ***C*** and constraints ***CT*** → high-level change ***hc***
- ***Constraints:***
  - *existence constraint* → an input change in ***C*** must be present or not
  - *parameter constraint* → parameters of input changes are interrelated by specific operators, e.g., equality (=), inequality (!=), …
  - *reduce constraint* → defines which input changes are unnecessary after generating ***hc***
  - *build constraint* → defines how ***hc*** is built from the input changes, i.e., mapping between parameters of ***hc*** and the input changes of ***C***

Rules (cont.)

Rules (more formal):

- ***C* x *CT* → *hc***
- ***C*** = $\{c_1(p_1, p_2, \ldots), c_2(p_1, p_2, \ldots), \ldots, c_n(p_1, p_2, \ldots)\}$
- ***hc***$(p_1, p_2, \ldots)$
- ***CT*** = *[existence, parameter, reduce, build]*
    - *existence:* $c \in$ ***C*** → *{exist, not-exist}*
    - *parameter:* $c_i(p_j)$ <op> $c_k(p_l)$: <op> in {'=', '!=', …}
    - *reduce:* $c \in$ ***C*** → *{reduce, not-reduce}*
    - *build:* $p_j \in$ ***hc***: $hc(p_j) = c_k(p_l)$



Example: merge of objects

- ***C* x *CT* → *merge(from, to)***
- ***C*** = *{mapObj(A,C), mapObj(B,C), mapObj(A,D), mapObj(B,E)}*
- *existence: mapObj(A,C), mapObj(B,C), !mapObj(A,D), !mapObj(B,E)*
- *parameter: C!=D, C!=E, A!=B*
- *reduce: mapObj(A,C), mapObj(B,C)*
- *build: from=A, to=C → merge(A,C)*

Rule engine

Advanced Diff (Rule Engine)

Merge & Reduce

Input: set of low-level changes ***LC*** (result of match)
set of rules ***R***
Output: set of high-level changes ***HC***

C: set of all changes

applyRules(*LC,R*)

C = ***LC***
do {
for all rules ***r*** ϵ ***R*** *do*
- *applyRule*(***C,r***) → C_r [new changes generated by ***r***]
- ***C*** = ***C*** U C_r [save all newly generated changes]

} *while* (*hasChanged*(***C***))
compact(***C***) [merge of changes, deletion of unnecessary changes]
return ***C\LC*** [high-level changes are returned]

Applying a single rule

Input: set of changes ***C***
rule ***r***
Output: set of new changes $\boldsymbol{C_r}$ generated by ***r*** on ***C***

applyRule(*C,r*)

select(***C***,***r***) [select input change vectors $\hat{c}$ from ***C*** that satisfy existence and parameter constraint of ***r***] → $\boldsymbol{C_{satisfy}} = \{\boldsymbol{\hat{c}_1}, \boldsymbol{\hat{c}_2}, \ldots\}$; $\boldsymbol{\hat{c}_i} = (\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots)$
for all $\boldsymbol{\hat{c}_i} \in \boldsymbol{C_{satisfy}}$ *do*

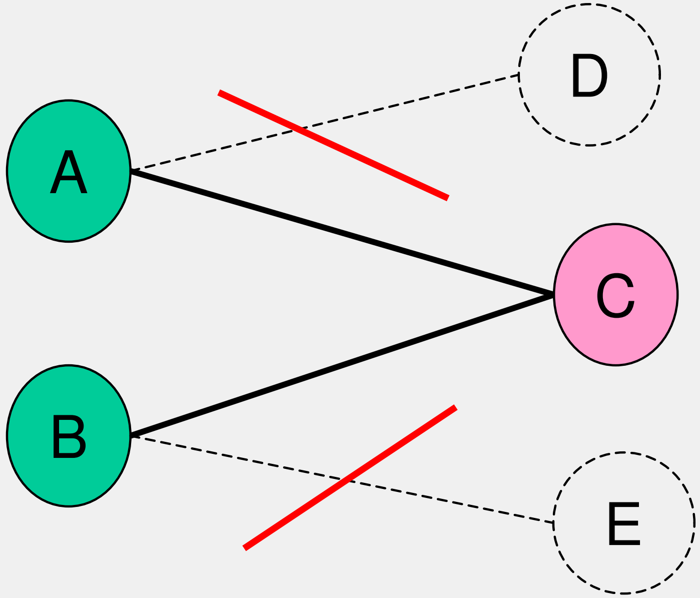
- build new change from changes in $\boldsymbol{\hat{c}_i}$ → $\boldsymbol{c_{result}}$ [build constraint of ***r***]
- assign low-level changes from changes in $\boldsymbol{\hat{c}_i}$ to $\boldsymbol{c_{result}}$
- mark unnecessary changes in $\boldsymbol{\hat{c}_i}$ [reduce constraint of ***r***]
- $\boldsymbol{C_r} = \boldsymbol{C_r} \cup \{\boldsymbol{c_{result}}\}$

return $\boldsymbol{C_r}$

Overview of possible high-level changes and their inverse

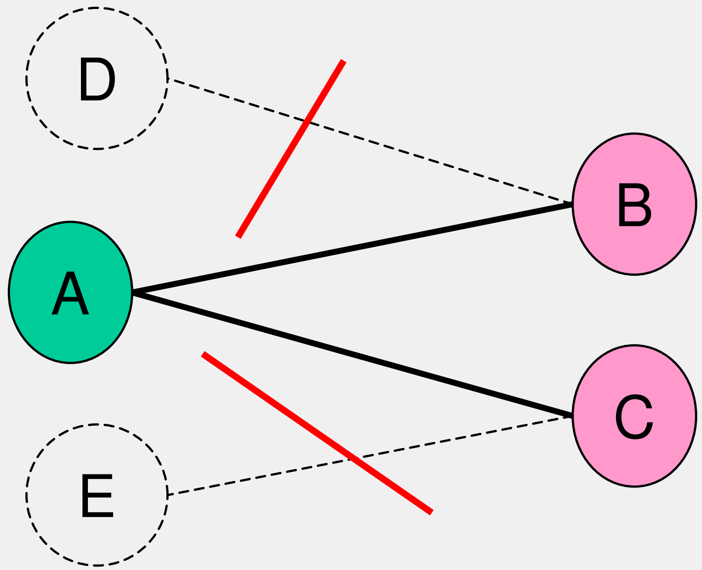
*High-level change **hc***	*Inverse of **hc***
merge(objs_from, obj_to)	***split***(obj_to, objs_from)
substitute(obj, obj_subs)	***substitute***(obj_subs, obj)
toObsolete(obj)	***revokeObsolete***(obj)
chgAttValue(obj, att_name, old_value, new_value)	***chgAttValue***(obj, att_name, new_value, old_value)
moveObj(obj, obj_from, obj_to)	***moveObj***(obj, obj_to, obj_from)
addLeafObj(obj)	***delLeafObj***(obj)
addInnerObj(obj)	***delInnerObj***(obj)
addSubTree(obj_root, objs)	***delSubTree***(obj_root, objs)
…	…

Common rules I



merge(A,C) and merge(B,C)
→$_{compact}$ merge({A,B},C)

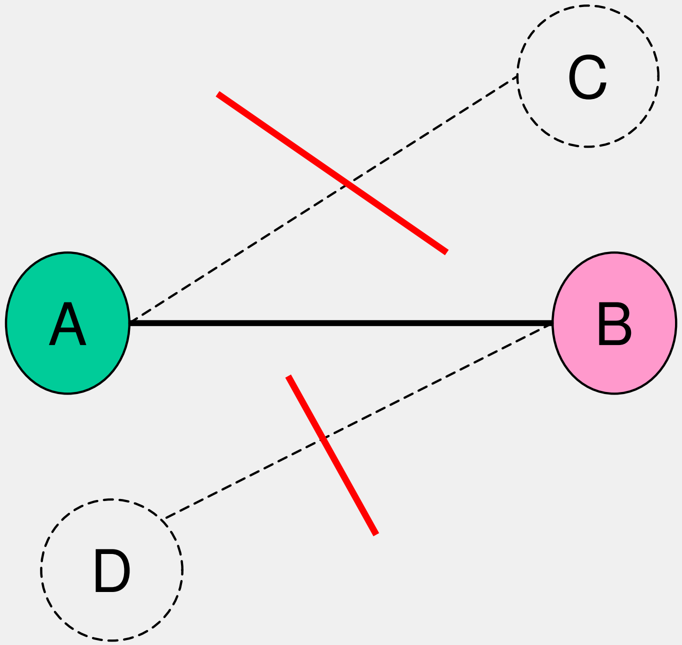
- mapObj(A,C), mapObj(B,C)
- !mapObj(A,D), !mapObj(B,E)
- C!=D, C!=E, A!=B



split(A,B) and split(A,C)
→$_{compact}$ split({A,{B,C}})

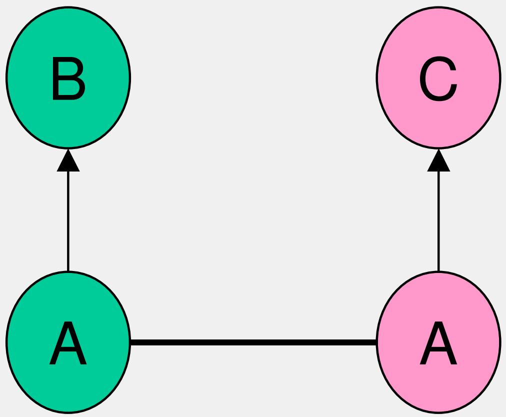
- mapObj(A,B), mapObj(A,C)
- !mapObj(D,B), !mapObj(E,C)
- A!=D, A!=E, B!=C

Common rules II



substitute(A,B)

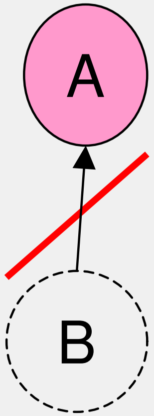
- mapObj(A,B)
- !mapObj(D,B), !mapObj(A,C)
- A!=D, B!=C, A!=B



moveObj(A,B,C)

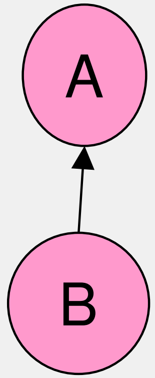
- mapObj(A,A)
- delAss(A,B), addAss(A,C)
- B!=C

Common rules III



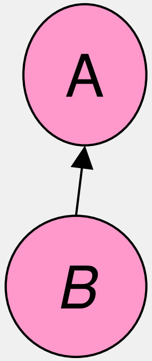
addLeafObj(A)

- addObj(A)
- !addAss(B,A)
- -



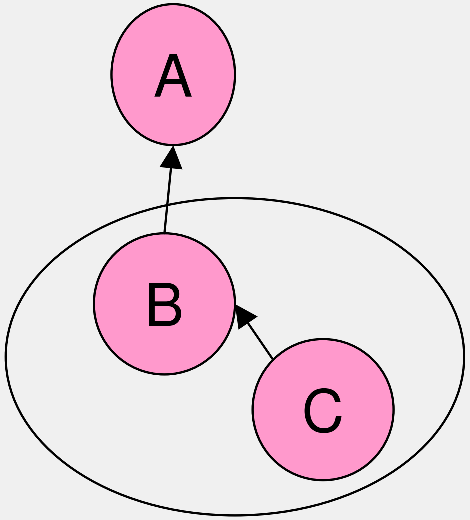
addInnerObj(A)

- addObj(A)
- addAss(B,A)
- -



addSubTree(A,B)

- addInnerObj(A), addLeafObj(B)
- addAss(B,A)
- A!=B



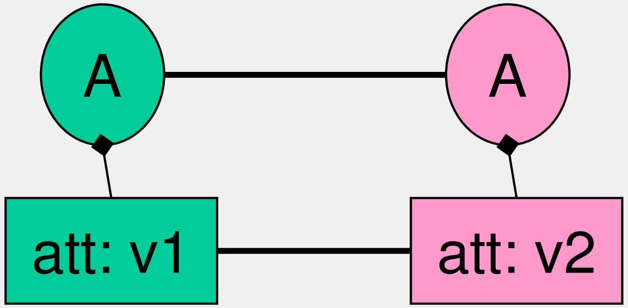
addSubTree(A,C)

- addInnerObj(A), addSubTree(B,C)
- addAss(B,A)
- A!=B

- *Similar rules for delLeafObj, delInnerObj, delSubTree*

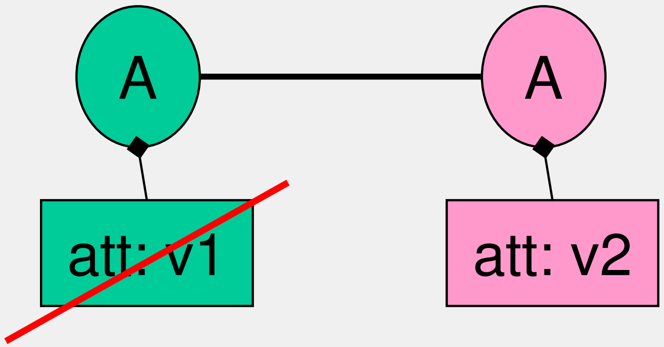
addSubTree(A,B)
addSubTree(A,C)
$\rightarrow_{\textbf{compact}}$ addSubTree(A,{B,C})

Common rules IV



chgAttValue(A,att,v1,v2)

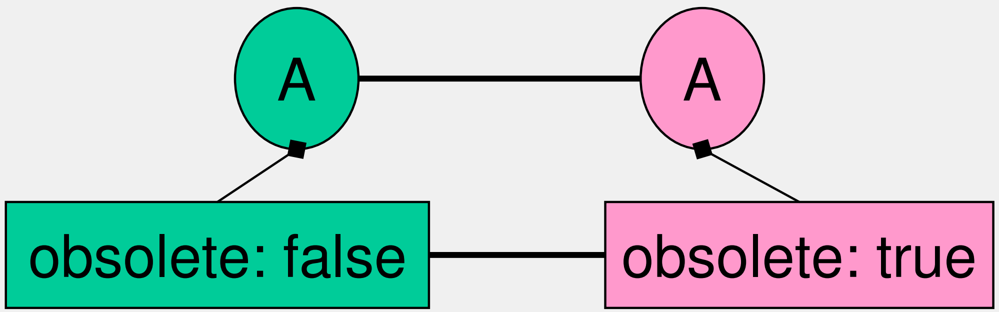
- mapAtt((A,att,v1),(A,att,v2))
- A=A, att=att, v1!=v2



chgAttValue(A,att,V1,V2)

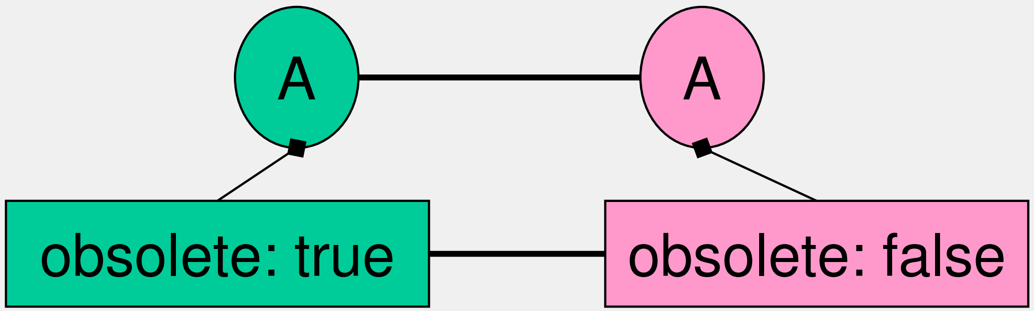
- addAtt((A,att,v2)), delAtt((A,att,v1))
- A=A, att=att, v1!=v2

Specific rules – Example Gene Ontology



toObsolete(A)

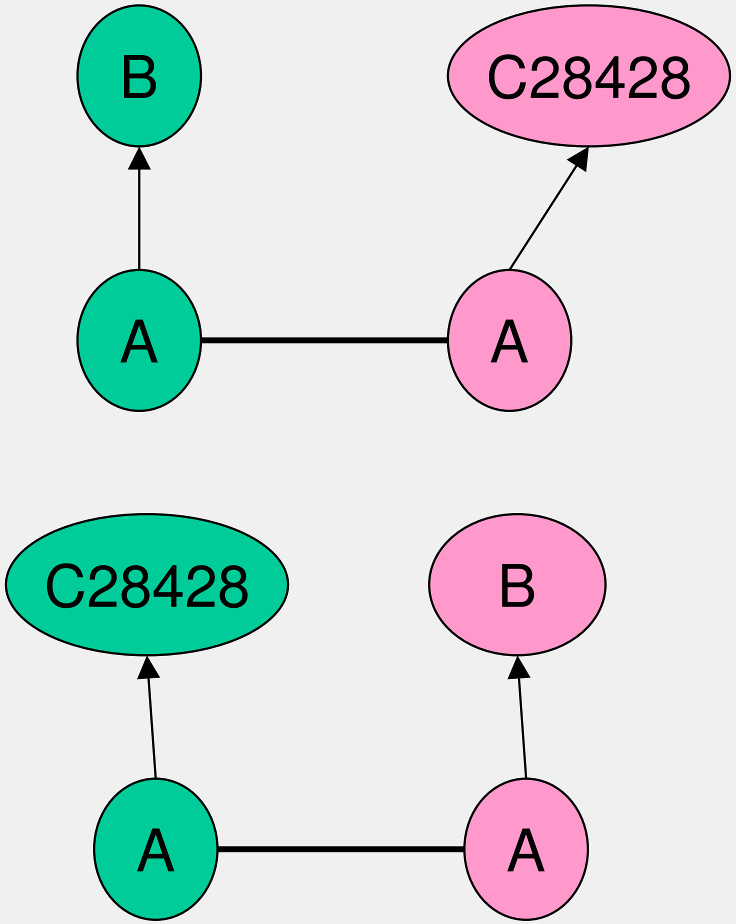
- chgAttVal(A,att,v1,v2)
- att = ‚obsolete'
- v1 = ‚false', v2 = ‚true'



revokeObsolete(A)

- chgAttVal(A,att,v1,v2)
- att = ‚obsolete'
- v1 = ‚true', v2 = ‚false'

Specific rules – Example NCI Thesaurus



toObsolete(A)

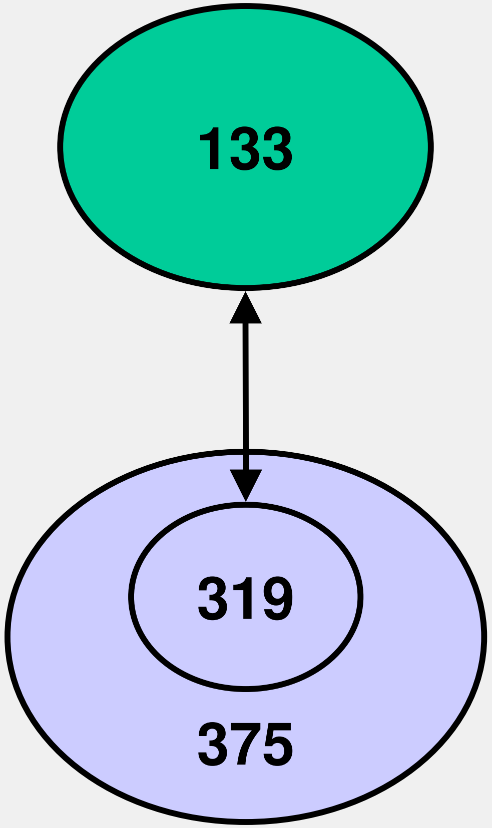
- moveObj(A,B,C)
- C = ‚C28428‘

revokeObsolete(A)

- moveObj(A,C,B)
- C = ‚C28428‘

Evaluation: GO-BP between 2007-05 and 2007-06

110,083 elements
→ 110,605 elements



High-level changes
133

High-Low Level Mapping
458 correspondences

Low-level changes
694 (596 add; 72 del; 26 map)

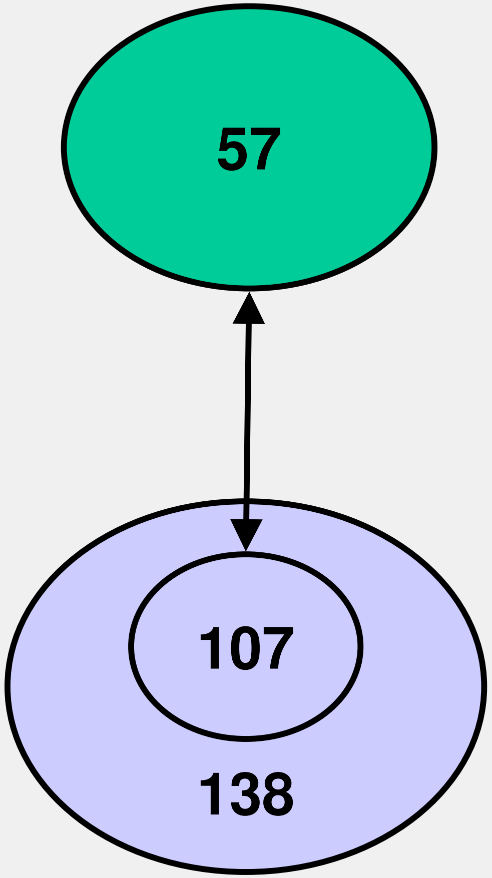
|domain(*High-Low Level Mapping*)| = **133 (100%)**
|range(*High-Low Level Mapping*)| = **319 (46%)** → **375** low level changes are standalone, i.e., they do not support any high-level change

Evaluation: GO-BP between 2007-05 and 2007-06

Selected high-level changes:

- **merge**({GO:0051225("spindle assembly"), GO:0051226("meiotic spindle assembly"), GO:0051227("mitotic spindle assembly")}, GO:0051225)
- **toObsolete**(GO:0006755("carbamoyl phosphate-ADP transphosphorylation"))
- **addSubTree**(GO:0033212("iron assimilation“), {GO:0033213, GO:0033214, GO:0033215}), … *(13 more)*
- **addInnerObj**(GO:0010431), **addInnerObj**(GO:0055048)
- **addLeafObj**(GO:0010424), … *(22 more)*
- **moveObj**(GO:0000281,'is_a', GO:0000910, GO:0033205), … *(26 more)*

Evaluation: GO-MF between 2007-05 and 2007-06



High-level changes
57

High-Low Level Mapping
127 correspondences

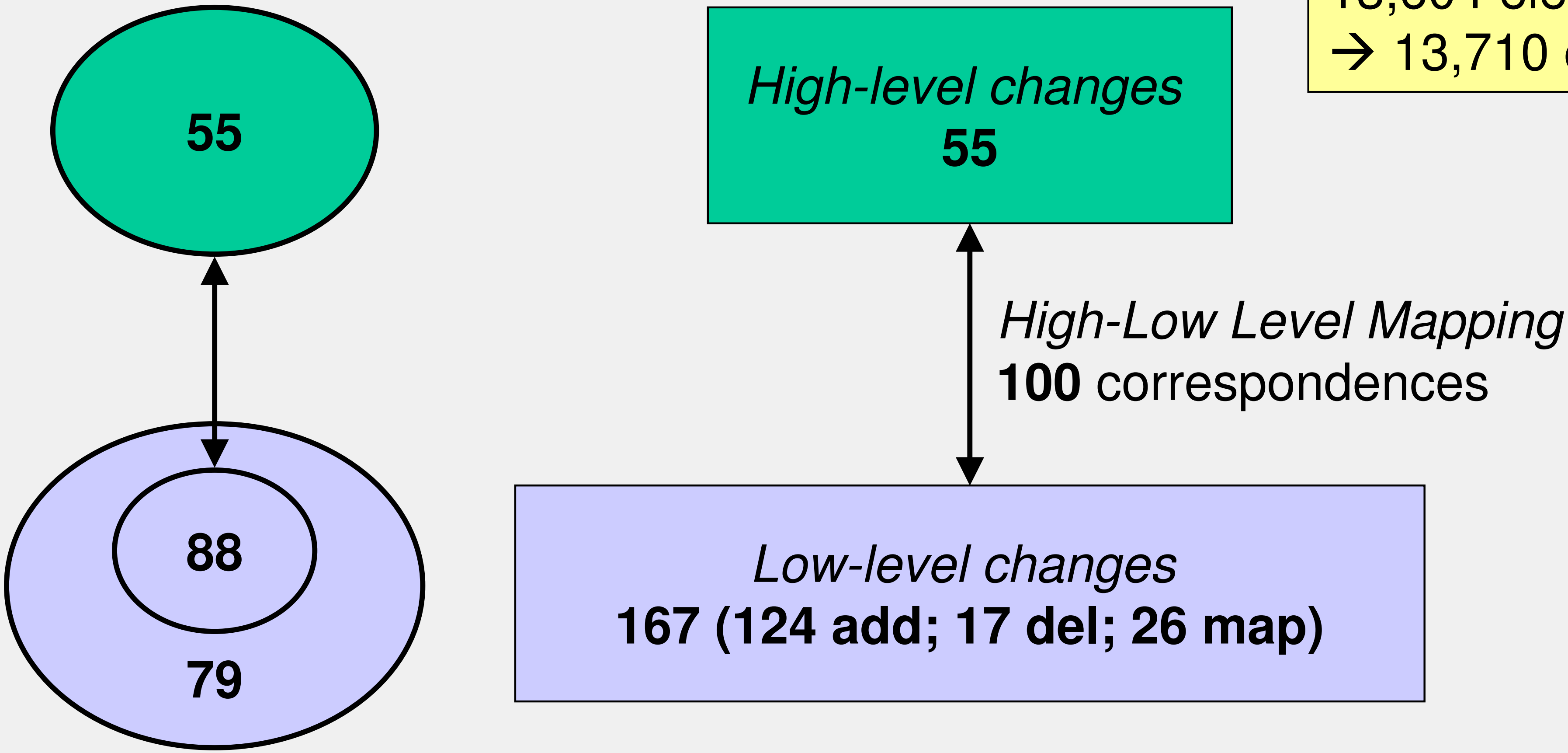
Low-level changes
245 (203 add; 27 del; 15 map)

46,702 elements
→ 46,875 elements

|domain(*High-Low Level Mapping*)| = **57 (100%)**
|range(*High-Low Level Mapping*)| = **107 (43%)** → **138** low level changes are standalone, i.e., they do not support any high-level change

Evaluation: GO-CC between 2007-05 and 2007-06

13,604 elements
→ 13,710 elements

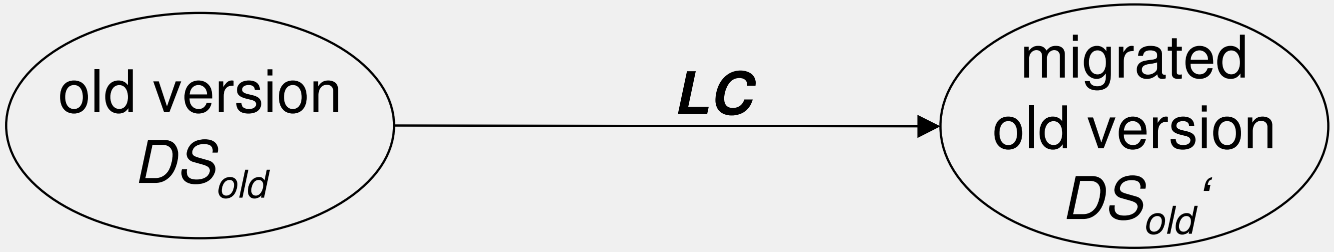


|domain(*High-Low Level Mapping*)| = **55 (100%)**
|range(*High-Low Level Mapping*)| = **88 (52%) → 79** low level changes are standalone, i.e., they do not support any high-level change

Completeness of low-level changes

Forward case

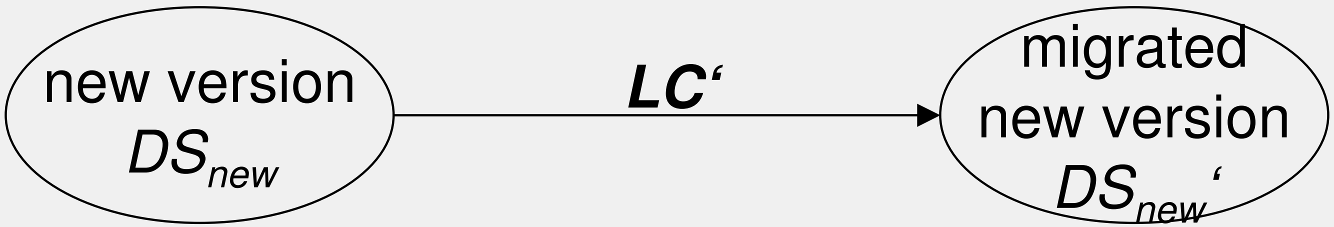
Low-level changes ***LC*** (changes to build DS_{new} based on DS_{old})



$\boldsymbol{DS_{old}‘ = DS_{new}}$ **?**

Inverse case

LC‘ = *inverse*(***LC***) → all changes in ***LC*** are reversed



$\boldsymbol{DS_{new}‘ = DS_{old}}$ **?**

Both together

$\boldsymbol{DS_{old}‘‘ = DS_{old}}$ **?**

Order of performing low-level changes

Performing order:

1. ***delAtt*** – *delete* all attributes not present in origin version
2. ***delAss*** – *delete* all associations not present in origin version
3. ***delObj*** – *delete* all objects not present in origin version
4. ***addObj*** – *insert* all new objects
5. ***mapObj*** – *update* all changed objects
6. ***addAss*** – *insert* all new associations
7. ***mapAss*** – *update* all changed associations
8. ***addAtt*** – *insert* all new attributes
9. ***mapAtt*** – *update* all changed attributes

Evaluation for GO sub ontologies between 2007-05 and 2007-06 using the computed low-level changes:

- reconstruction of versions worked for all three cases (forward, inverse, both) and all sub ontologies → complete low-level changes

Summary and Future Work

- **Approach for discovering high-level changes between structured data source versions**
  - Distinction between low-/high-level changes connected by a high-low level mapping
  - Usage of rules and generic rule engine to detect high-level changes based on low-level ones
  - Approach is customizable
    - Match for detection of low-level changes
    - Adaptable rules and changes for different data sources
  - First evaluation for GO sub ontologies between 2007-05 and 2007-06
- **Next steps**
  - Extended evaluation, e.g., other time periods (quarter, half year, year)
  - Evaluation in other domains, e.g., product catalogs or web directories (match as critical step)
  - Application: efficient matching; use in current/upcoming analysis, e.g., impact of ontology/annotation changes on algorithm results

Thank You for your attention !!

Questions ?