A T-Box Generator for testing scalability of OWL mereotopological patterns

Martin Boeker[1], Janna Hastings[2], Daniel Schober[1], and Stefan Schulz[3]

martin.boeker@uniklinik-freiburg.de

[1] Institute of Medical Biometry and Medical Informatics,
University Medical Center Freiburg, Germany
[2] Chemoinformatics and Metabolism, EBI, Hinxton, UK
[3] Department of Medical Informatics, Medical University Graz, Austria

OWLED
San Francisco
June 6th 2011

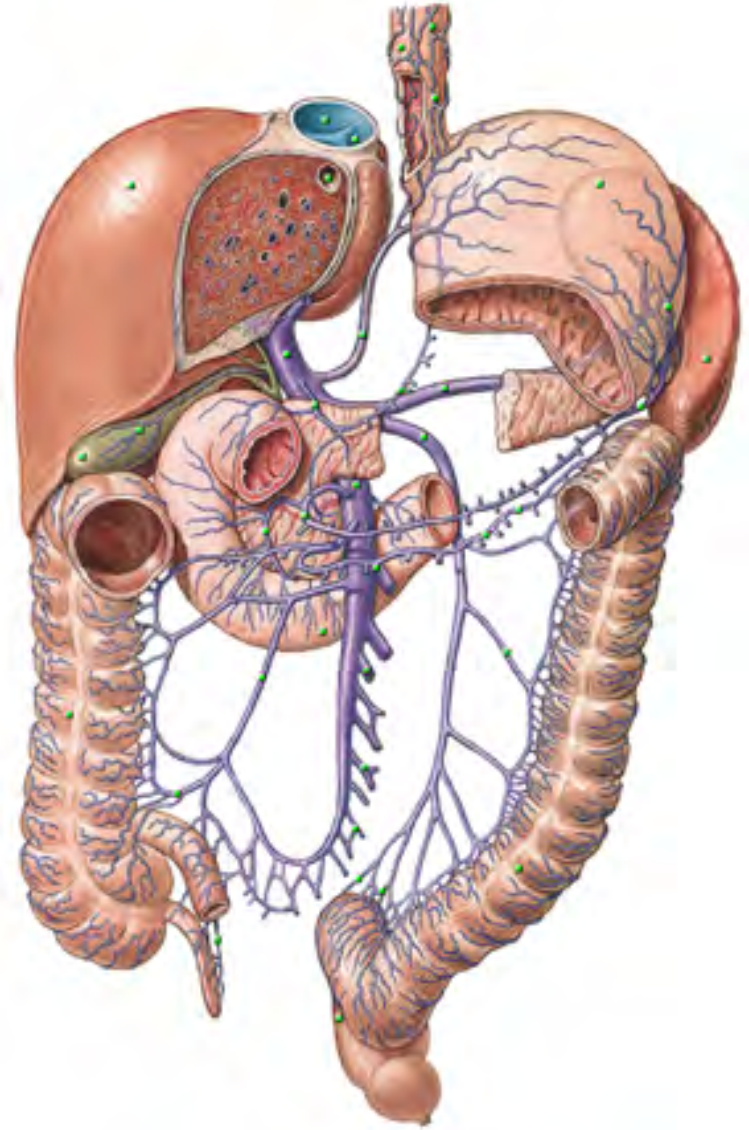
Background

@neurIST	Integrated Biomedical Informatics for the Management of Cerebral Aneurysms (European Community - 6th FP)
DebugIT	Detecting and Eliminating Bacteria UsinG Information Technology (European Community - 7th FP)
BioTop	A Top-Domain Ontology for the Life Sciences
GoodOD	Good Ontology Design (DFG grant JA 1904/2-1, SCHU 2515/1-1)

http://www.cistib.upf.edu/aneurist1/
http://www.debugit.eu/
http://www.imbi.uni-freiburg.de/ontology/biotop/

Taxonomy

- *A* subClassOf *B*
- *Stomach* subClassOf *CavitatedOrgan*

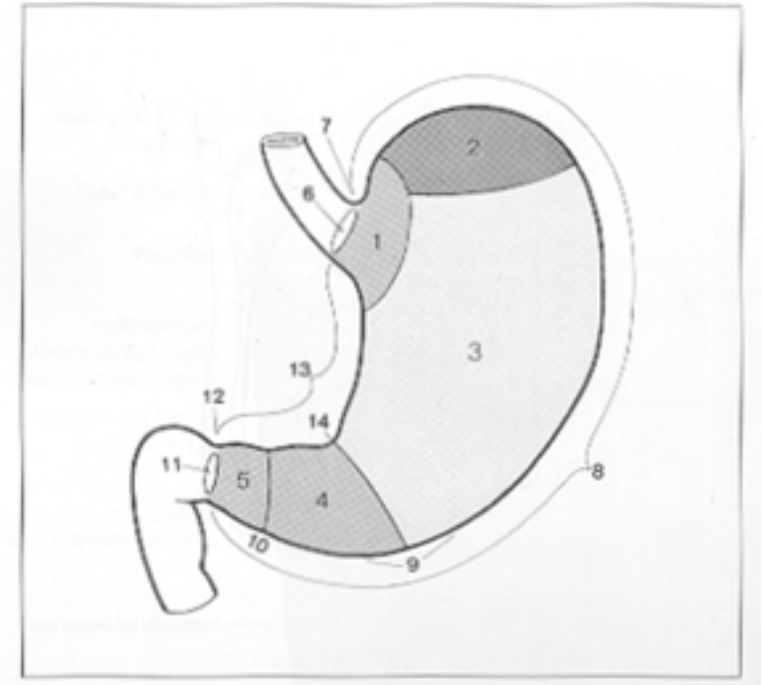


Sobotta, Atlas der Anatomie des Menschen

Mutual disjointness

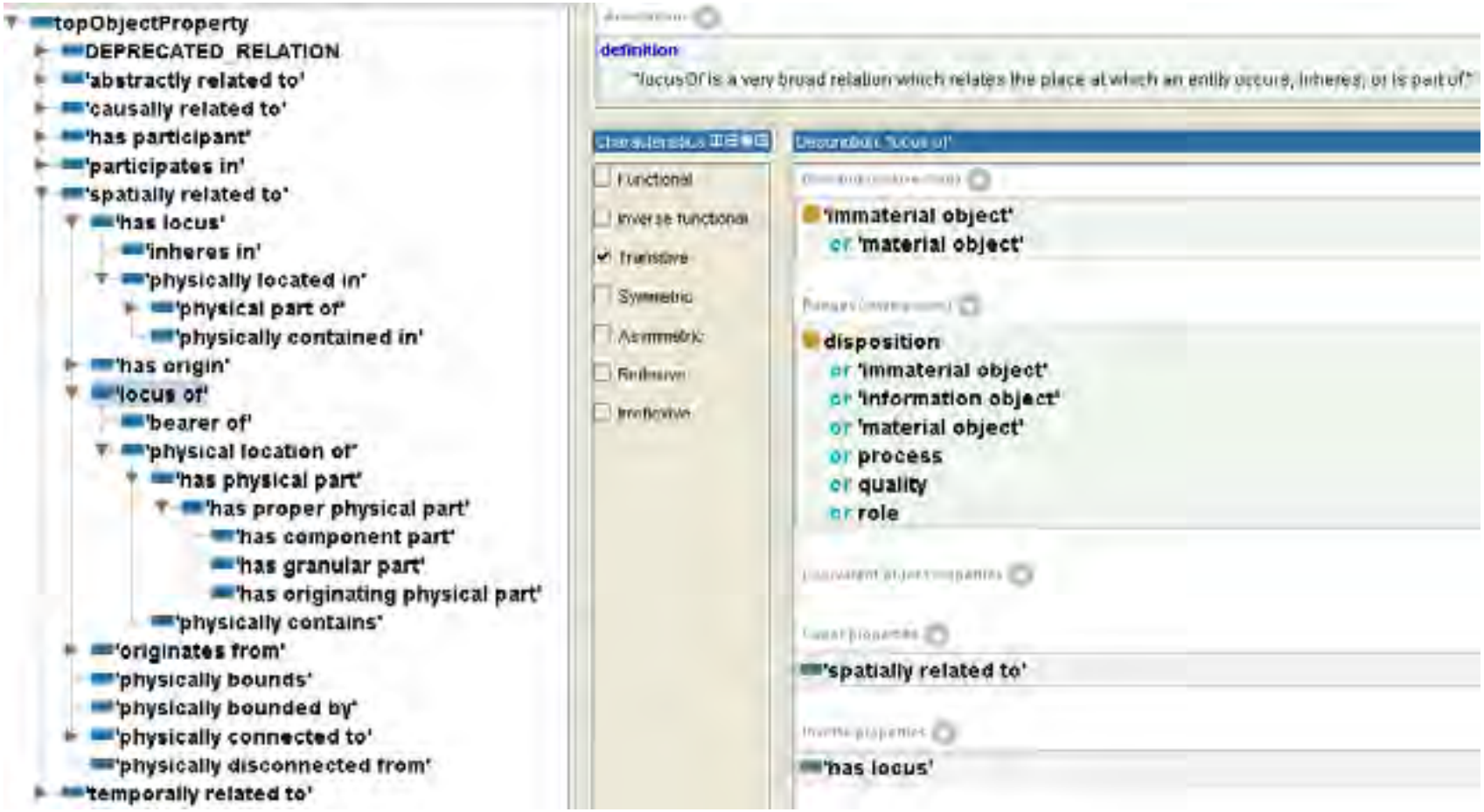
$$\texttt{DisjointClasses}\ (C_1, C_2, \ldots, C_n) =_{def} \{C_1\ \texttt{subclassOf not}\ C_2; \ldots; C_1\ \texttt{subclassOf not}\ C_n; C_2\ \texttt{subclassOf not}\ C_n; \ldots\}$$

- mutual disjointness: there are no entities both member of class C_1 *and* $C_2\,(\ldots C_n)$



Rauber, Anatomie des Menschen

locusOf-hasLocus implementation in BioTop

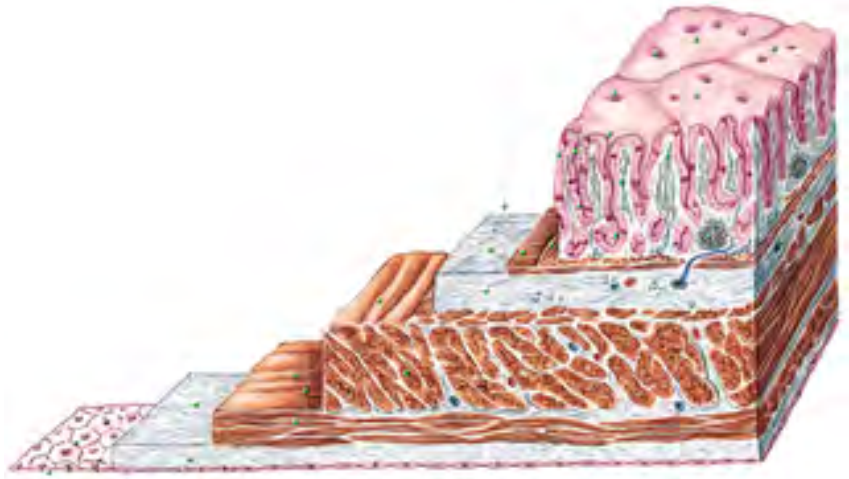


http://www.imbi.uni-freiburg.de/ontology/biotop/

Spatial disjointness

C_1 `subClassOf` *locusOf* `only (not (`*hasLocus* `some` C_2`))`
C_2 `subClassOf` *locusOf* `only (not (`*hasLocus* `some` C_1`))`

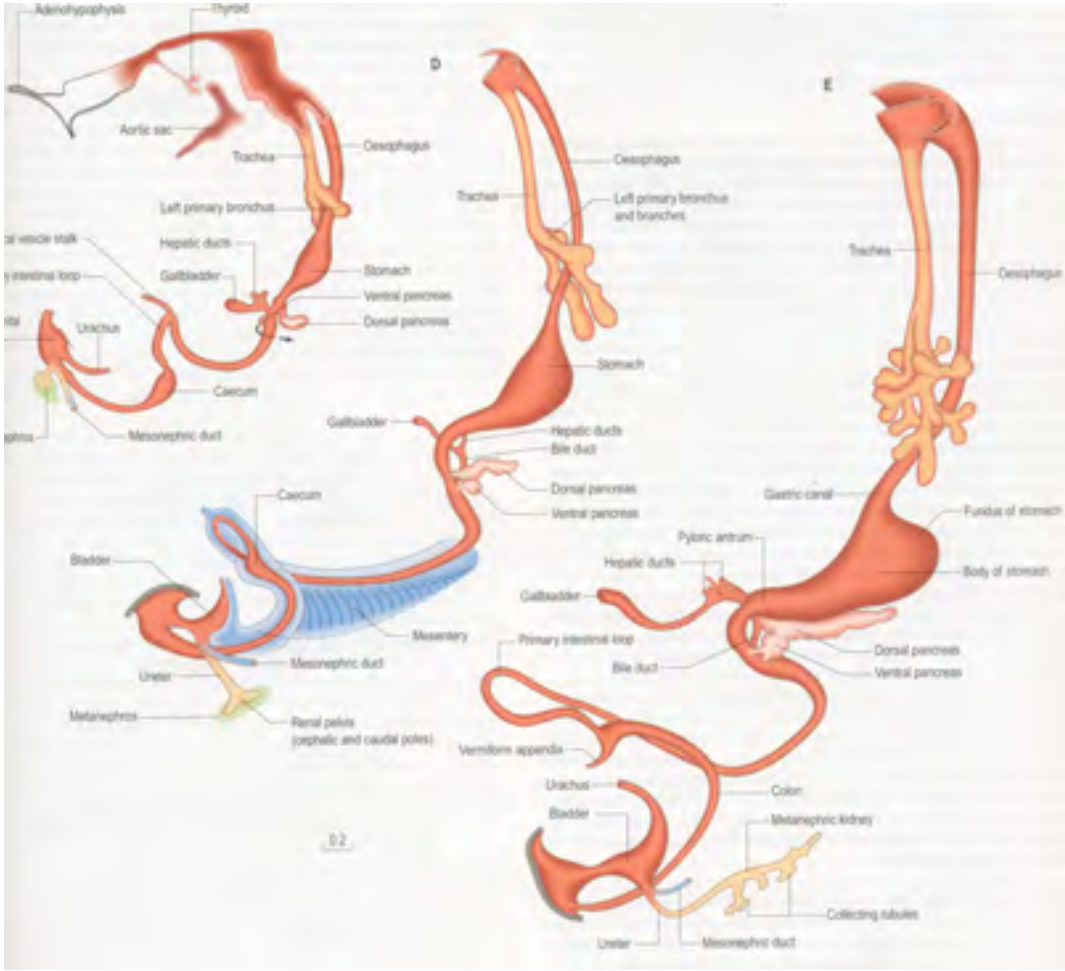
- spatial disjointness: “the location of an entity can only be where no spatial disjoint entity is located”.
  - why do we need that: mutual disjoint entities can be overlapping or in parthood relation



Sobotta, Atlas der Anatomie des Menschen

Partonomy

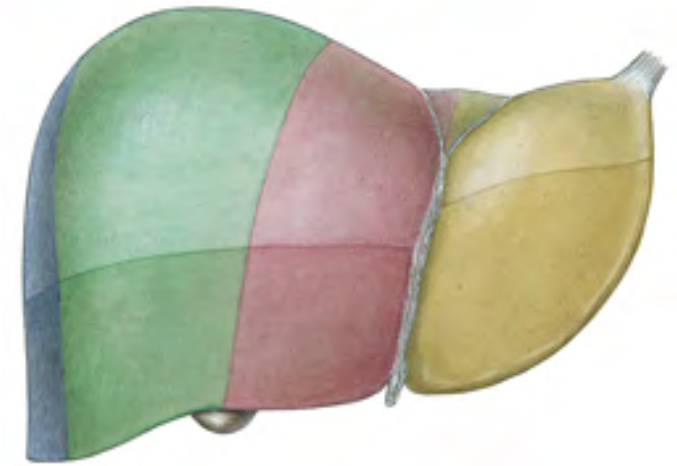
- Part implies whole
  - *Stomach* partOf SOME *GastrointestinalSystem*
- Whole implies part
  - *GastrointestinalSystem* hasPart SOME *Stomach*

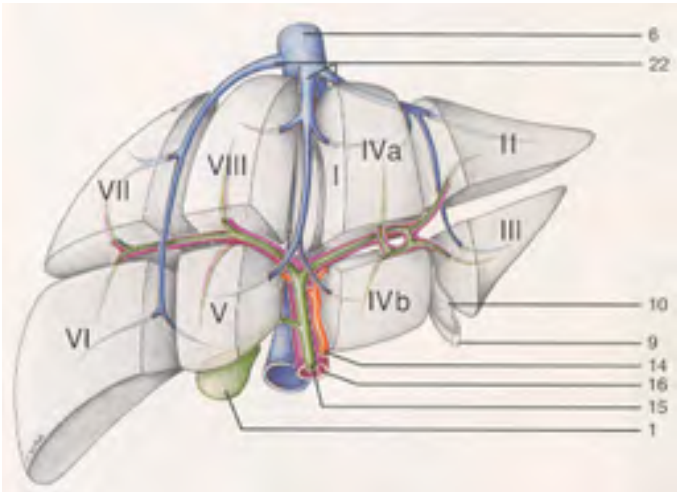


Gray's Anatomy, Editor: Susan Standring, 39th Edition

Exact cardinality

- Exact cardinalities in in biostructures: 1, 2, 5, n
  - *Hand* hasPart exactly 5 *Finger*
  - *Hand* hasPart exactly 1 *Thumb*
  - *Liver* hasPart exactly 8 *LiverSegment*





Sobotta, Atlas der Anatomie des Menschen
Rohen, Anatomie des Menschen. Fotografischer Atlas der systematischen und topografischen Anatomie.

Properties of the T-Box Generator

- Configuration of the
  - Number of hierarchical levels
  - Number of siblings on each level
  - Number of mutual disjoint classes on each level
  - Number of spatial disjoint classes on each level
  - Definition of partonomic restrictions as subclass or equivalent class expressions
  - Quantification of partonomic relation as existential or exact cardinal
  - Type of *locusOf-hasLocus*, *hasPart-partOf*: inverse and/ or transitive

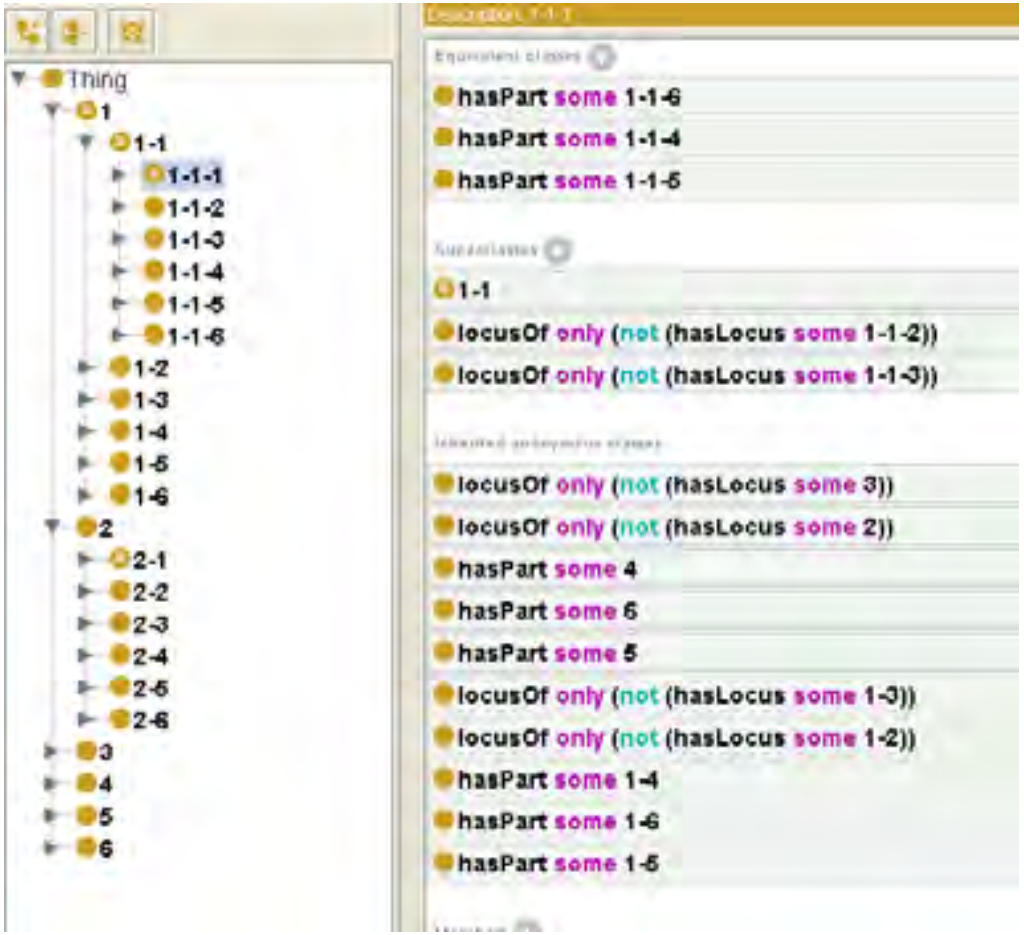
Implementation in Scala

- Scala version 2.8.2
  - object functional language for the JVM
- ca. 170 loc
- OWL API version 3.2.1 (Feb 4 2011)

```
98  /**
99   * process a List[OWLClass] depending on @cl (nested class level)
100  * call generateClassList for every parent class in List[OWLClass] recursively
101  */
102 def processClassList(cl: Int, classList: List[OWLClass]): Unit = {
103   // do some restrictions
104   setDisjoints(classList)
105
106   // do some hierarchy stuff
107   if (cl == 1) {
108     for (cls <- classList)
109       manager.addAxiom(ontology, factory.getOWLDeclarationAxiom(cls))
110   }
111   if (cl < depth) {
112     for (cls <- classList) {
113       val pref = cls.toStringID.split("#")(1) + "_"
114       val nextLevel = generateClassList(2, pref,
115                                         List(factory.getOWLClass(IRI.create
116                                           (ontologyIRI + "#" + pref + 1))))
117       for (nlcls <- nextLevel) {
118         manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(nlcls, cls))
119       }
120       //recurse!!!
121       processClassList(cl+1, nextLevel)
122     }
123   }
124 }
```

http://www.scala-lang.org/
http://owlapi.sourceforge.net
Download at: http://www.imbi.uni-freiburg.de/ontology/t-box-generator.zip

Example ontology: 6^5 classes, partonomy as equivalent classes, existential quantification



Estimating the performance of classification

- three groups of seven ontologies
  - each ca. 8000 classes: $6^5, 20^3, 90^2$ ($siblings^{hierarchical\ levels}$)
- "increasing complexity"
  - hierarchy only
  - add 50% siblings mutually disjoint and 50% siblings mutually spatially disjoint
  - add partonomical restrictions to mutually disjoints as subclass expressions with existential quantification
  - change restrictions to equivalent class expression with existential quantification
  - set properties of *locusOf-hasLocus*, *hasPart-partOf* relations to invers and transitive
  - change restrictions to exact cardinal quantification (1, no. of disjoints/ 2)

Experimental setting

- 1.6GHz Intel(R) Core(TM) i7 Q720, 4GB RAM, Windows 7 64-Bit, Java 1.6.25 64-Bit
- -Xmx3000m: effective memory allocation of 2796MB
- Three DL Reasoners
  - Fact++ (1.5.2), HermIT (1.3.3), Pellet (2.2.2)
- Three sequential measurements with Protegé (4.1 RC2 build 228)
  - unload Protege after each sequence

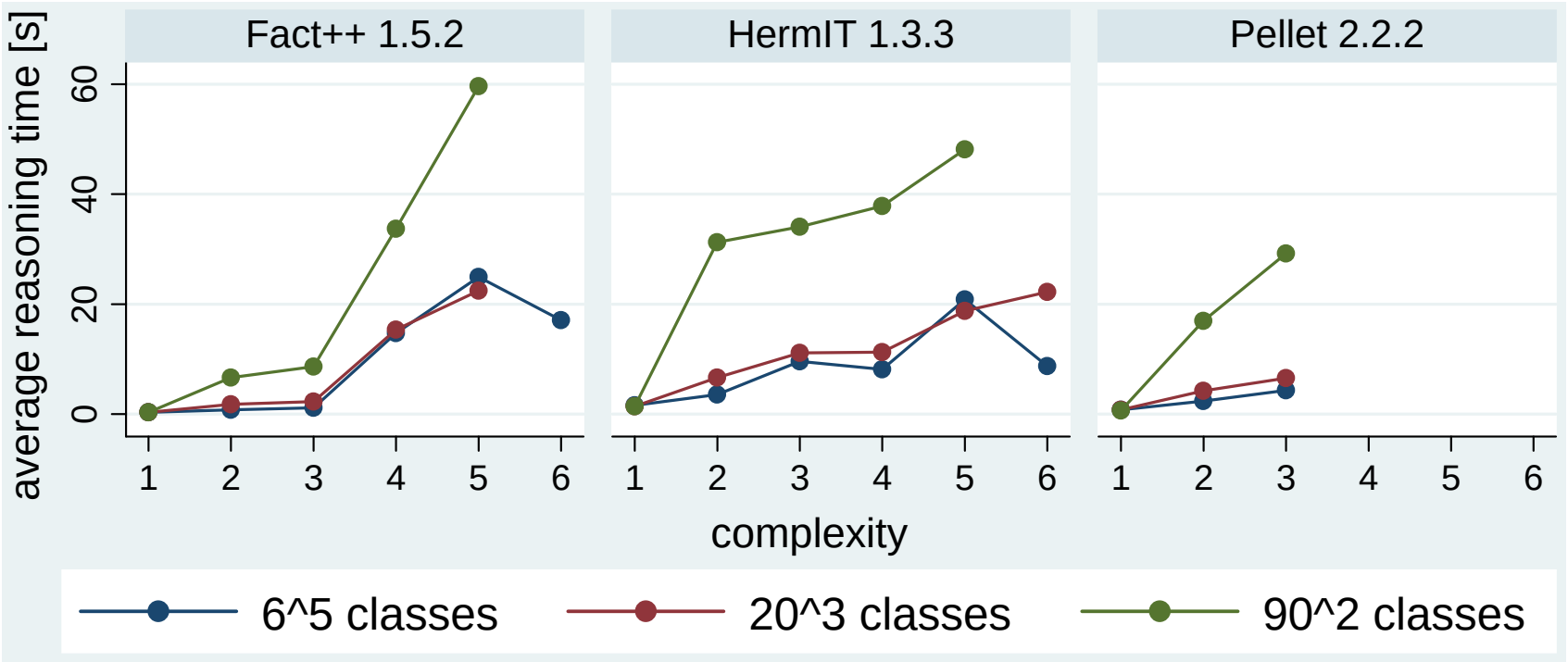
Results: Fact++ 1.5.2

disjoints	partonomy	subclasses or equivalent classes	object relations invers/ transitiv	quantification	complexity level	6^5=7776 [3; 2]	20^3=8000 [10; 5]	90^2=8100 [45; 22]
0	no				1	**0.3**	**0.3**	**0.3**
n	no				2	**0.7**	**1.6**	**6.7**
n	yes	sub	yes	existential	3	**1.1**	**2.0**	**8.4**
n	yes	eq	no	existential	4	**14.5**	**15.4**	**33.5**
n	yes	eq	yes	existential	5	**24.9**	**22.3**	**59.3**
n	yes	eq	yes	cardinality 1	6	**17.1**	**to**	**to**
n	yes	eq	yes	cardinality m	7	**to**	**to**	**oom**

Results: HermIT 1.3.3

disjoints	partonomy	subclasses or equivalent classes	object relations invers/ transitiv	quantification	complexity level	6^5=7776 [3; 2]	20^3=8000 [10; 5]	90^2=8100 [45; 22]
0	no				1	**0.7**	**0.5**	**0.8**
n	no				2	**1.8**	**3.8**	**26.1**
n	yes	sub	yes	existential	3	**7.6**	**7.6**	**30.4**
n	yes	eq	no	existential	4	**6.0**	**8.5**	**29.8**
n	yes	eq	yes	existential	5	**18.3**	**15.9**	**42.5**
n	yes	eq	yes	cardinality 1	6	**5.8**	**18.0**	**387.4**
n	yes	eq	yes	cardinality m	7	**276.4**	**oom**	**oom**

Results: comparison of average reasoning time



graphs produced with Stata 11.2

Limitations and further development

- introduce other object relations (pairwise inverse)
- relate to child classes of siblings
- introduce other patterns

Summary

- patterns of biostructure
  - taxonomy
  - mutuall disjointness
  - spatial disjointness
  - partonomy
  - exact cardinality
- T-Box generator
  - configurable to produce ontologies with patterns
  - estimation of reasoner performance
  - performance of Fact++ and HermIT equal on the tested ontologies