OWL API — programming interface to your ontologies

- Info on wiki
- Get an example from Dave (UiA)
  - read ✓
  - SWRL ✓
  - Unit ⊗

→ Dynamic features

→ "have rules for knowledge extension"

→ "need rules for dynamic..."

Which kind of rule engine ?

↳ external (easier to understand)

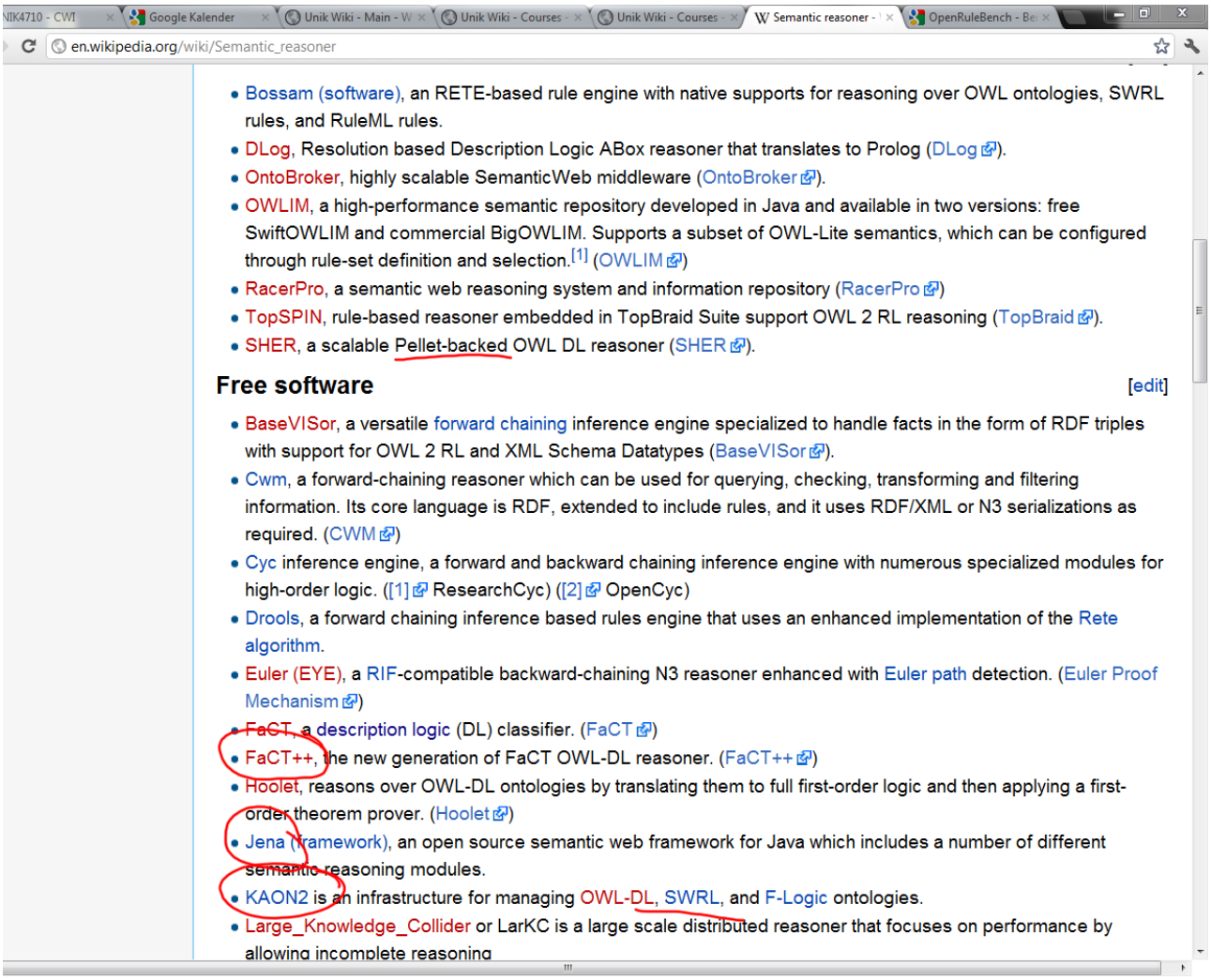
↳ internal

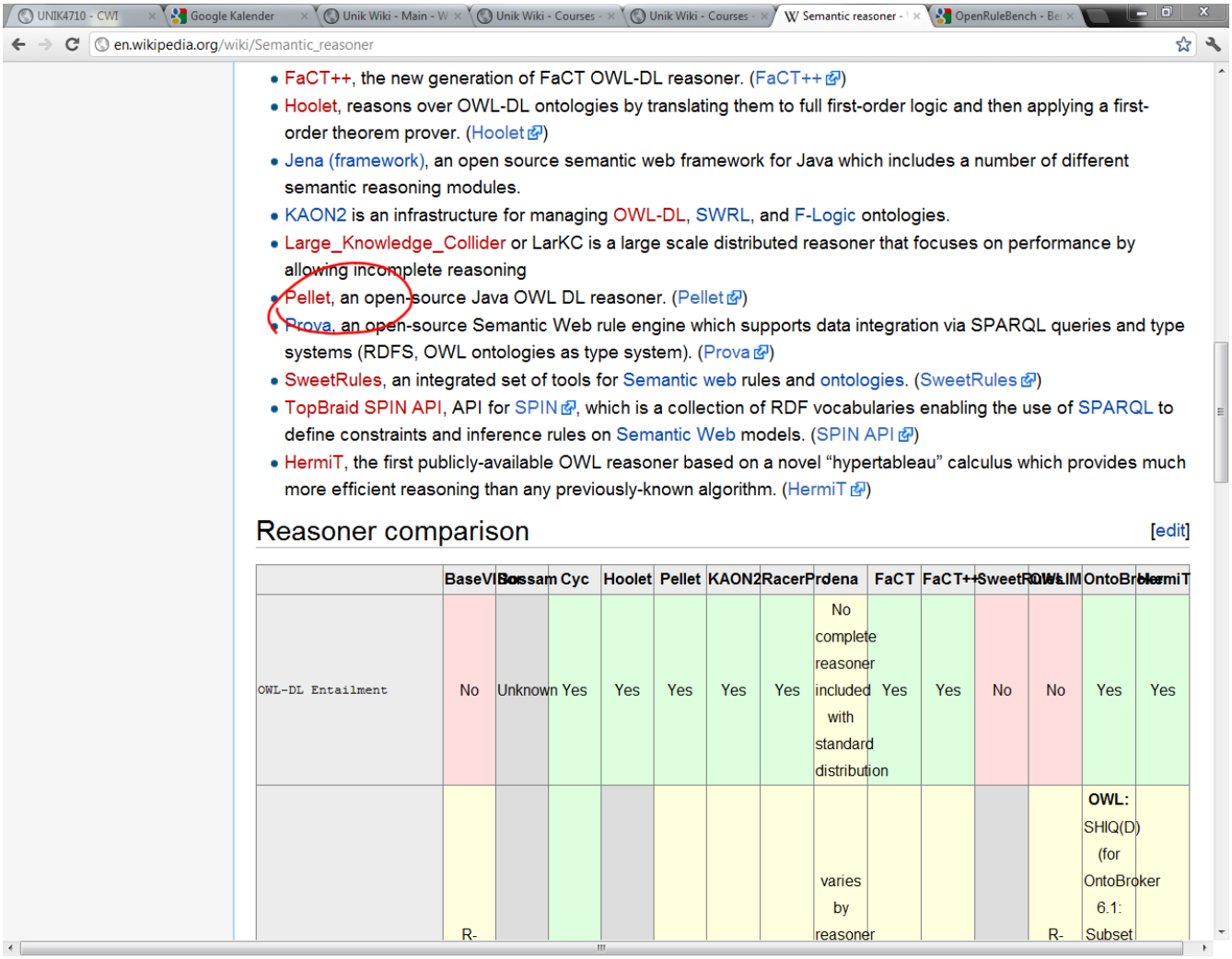
Which reasoner ?

↓

"Simplicity" - "review"

- Start with Simple SWRL
- Limits
- alternative approach

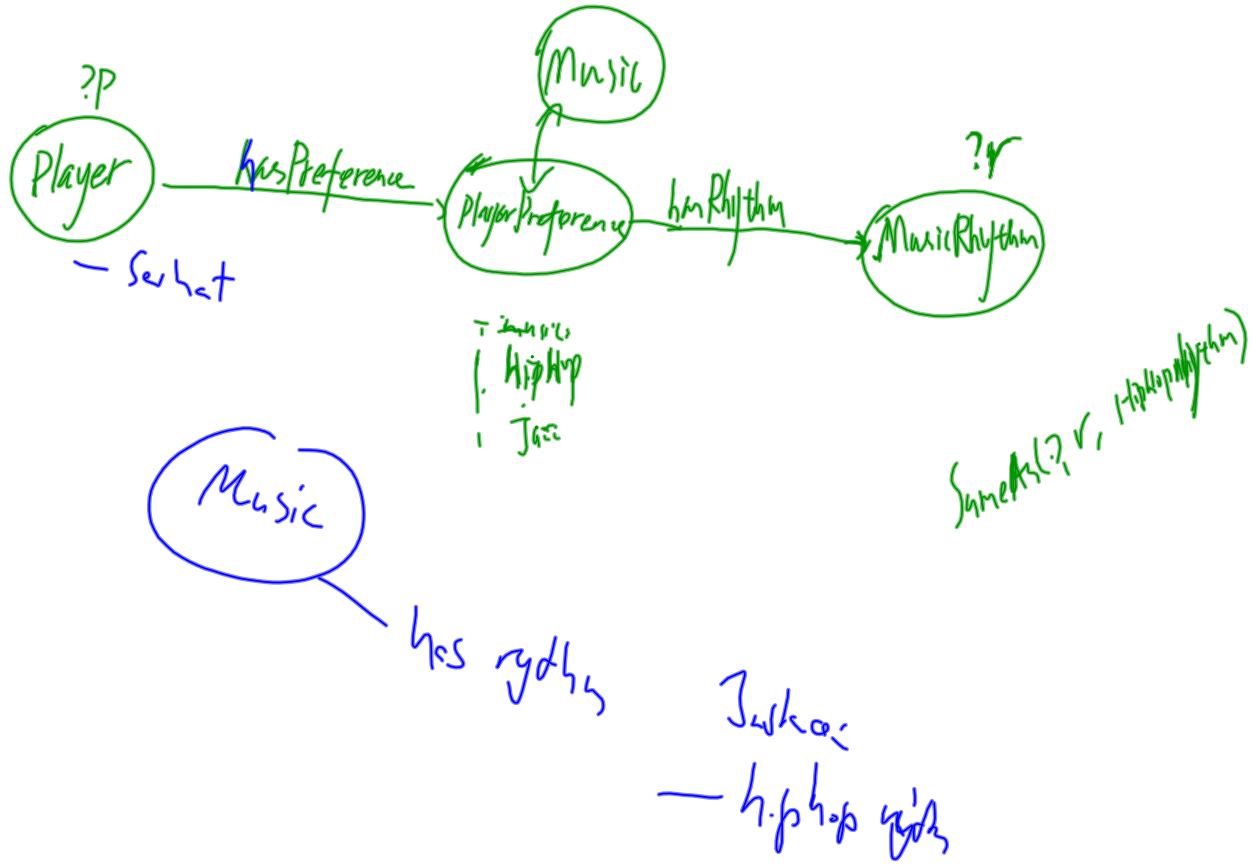

Google Kalender
Unik Wiki - Main - W
Unik Wiki - Courses
Unik Wiki - Courses
Semantic reasoner -
OpenRuleBench - Be
en.wikipedia.org/wiki/Semantic_reasoner
Bossam (software), an RETE-based rule engine with native supports for reasoning over OWL ontologies, SWRL rules, and RuleML rules.
DLog, Resolution based Description Logic ABox reasoner that translates to Prolog (DLog).
OntoBroker, highly scalable SemanticWeb middleware (OntoBroker).
OWLIM, a high-performance semantic repository developed in Java and available in two versions: free SwiftOWLIM and commercial BigOWLIM. Supports a subset of OWL-Lite semantics, which can be configured through rule-set definition and selection.[1] (OWLIM)
RacerPro, a semantic web reasoning system and information repository (RacerPro)
TopSPIN, rule-based reasoner embedded in TopBraid Suite support OWL 2 RL reasoning (TopBraid).
SHER, a scalable Pellet-backed OWL DL reasoner (SHER).
Free software
[edit]
BaseVISor, a versatile forward chaining inference engine specialized to handle facts in the form of RDF triples with support for OWL 2 RL and XML Schema Datatypes (BaseVISor).
Cwm, a forward-chaining reasoner which can be used for querying, checking, transforming and filtering information. Its core language is RDF, extended to include rules, and it uses RDF/XML or N3 serializations as required. (CWM)
Cyc inference engine, a forward and backward chaining inference engine with numerous specialized modules for high-order logic. ([1] ResearchCyc) ([2] OpenCyc)
Drools, a forward chaining inference based rules engine that uses an enhanced implementation of the Rete algorithm.
Euler (EYE), a RIF-compatible backward-chaining N3 reasoner enhanced with Euler path detection. (Euler Proof Mechanism)
FaCT, a description logic (DL) classifier. (FaCT)
FaCT++, the new generation of FaCT OWL-DL reasoner. (FaCT++)
Hoolet, reasons over OWL-DL ontologies by translating them to full first-order logic and then applying a first-order theorem prover. (Hoolet)
Jena (framework), an open source semantic web framework for Java which includes a number of different semantic reasoning modules.
KAON2 is an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies.
Large_Knowledge_Collider or LarKC is a large scale distributed reasoner that focuses on performance by allowing incomplete reasoning

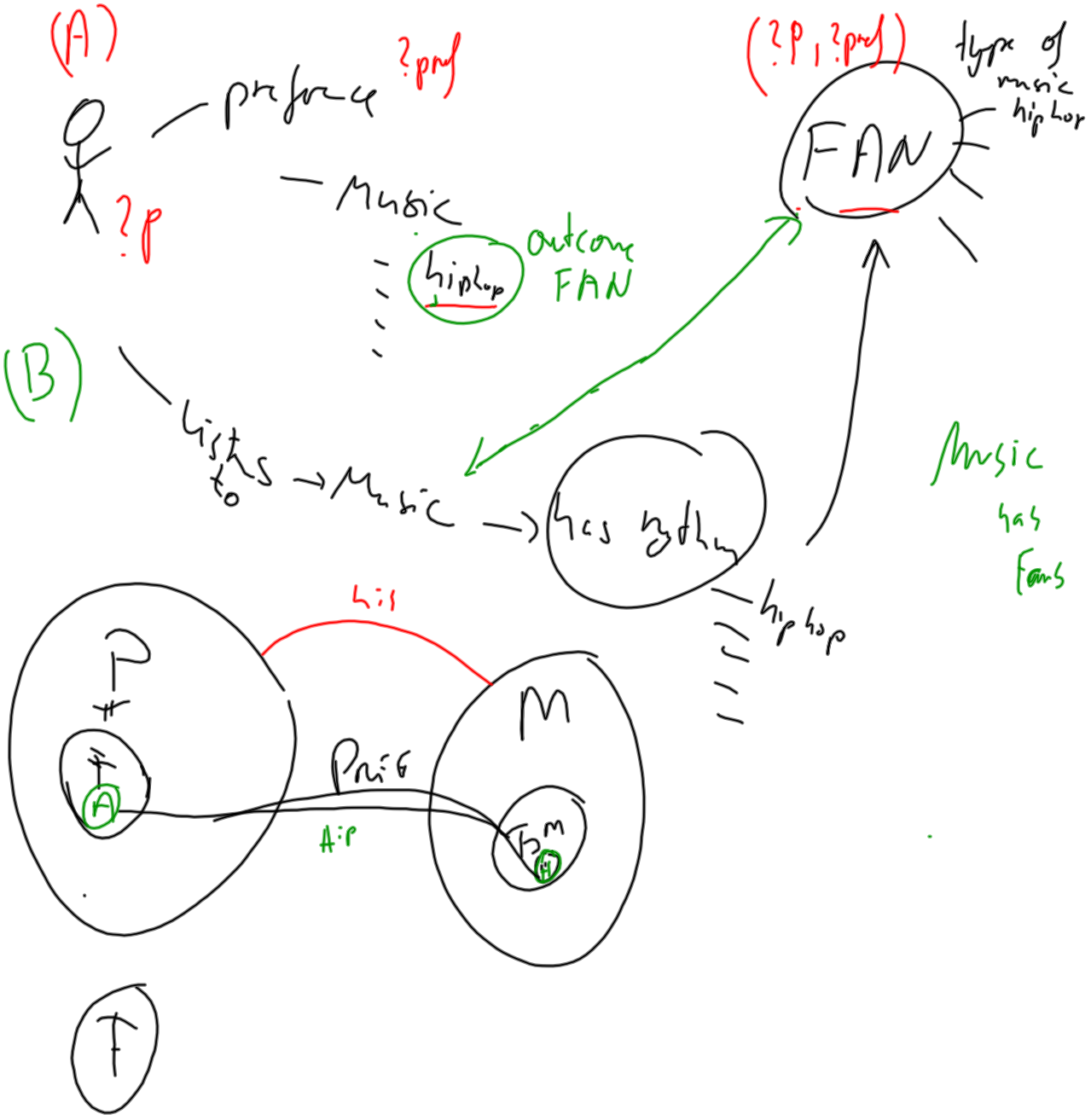


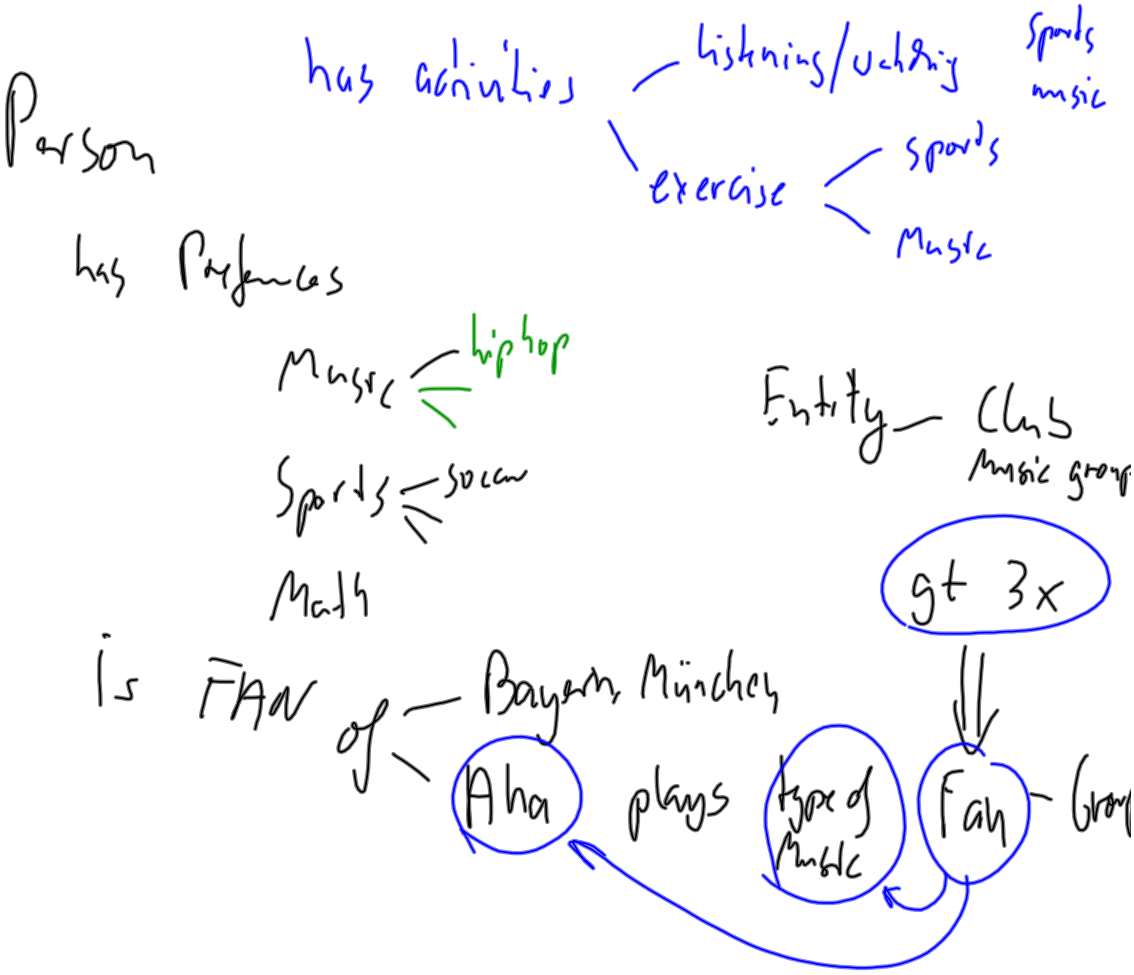
- FaCT++, the new generation of FaCT OWL-DL reasoner. (FaCT++)
- Hoolet, reasons over OWL-DL ontologies by translating them to full first-order logic and then applying a first-order theorem prover. (Hoolet)
- Jena (framework), an open source semantic web framework for Java which includes a number of different semantic reasoning modules.
- KAON2 is an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies.
- Large_Knowledge_Collider or LarKC is a large scale distributed reasoner that focuses on performance by allowing incomplete reasoning
- Pellet, an open-source Java OWL DL reasoner. (Pellet)
- Prova, an open-source Semantic Web rule engine which supports data integration via SPARQL queries and type systems (RDFS, OWL ontologies as type system). (Prova)
- SweetRules, an integrated set of tools for Semantic web rules and ontologies. (SweetRules)
- TopBraid SPIN API, API for SPIN, which is a collection of RDF vocabularies enabling the use of SPARQL to define constraints and inference rules on Semantic Web models. (SPIN API)
- HermiT, the first publicly-available OWL reasoner based on a novel "hypertableau" calculus which provides much more efficient reasoning than any previously-known algorithm. (HermiT)

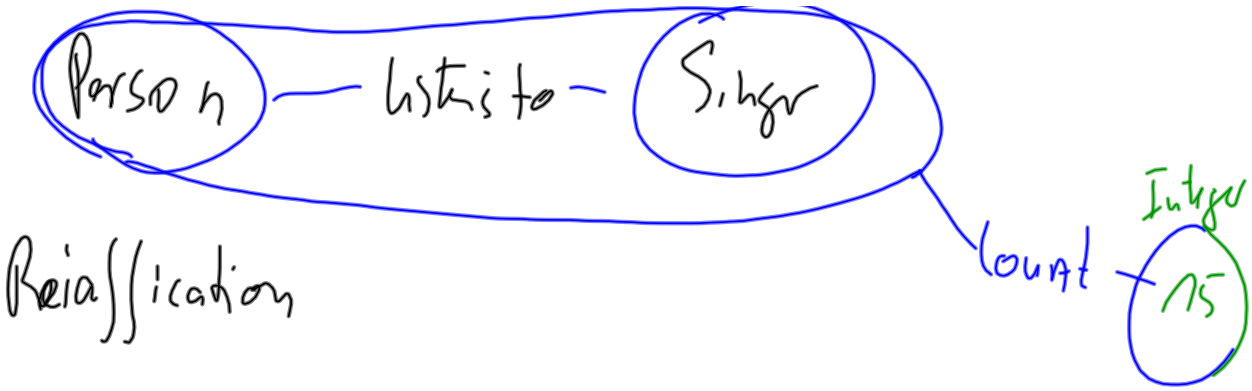
Reasoner comparison [edit]

	BaseVISor	Bossam	Cyc	Hoolet	Pellet	KAON2	RacerPro	dena	FaCT	FaCT++	SweetRules	OWLIM	OntoBroker	HermiT
OWL-DL Entailment	No	Unknown	Yes	Yes	Yes	Yes	Yes	No complete reasoner included with standard distribution	Yes	Yes	No	No	Yes	Yes
	R-							varies by reasoner				R-	**OWL:** SHIQ(D) (for OntoBroker 6.1: Subset	

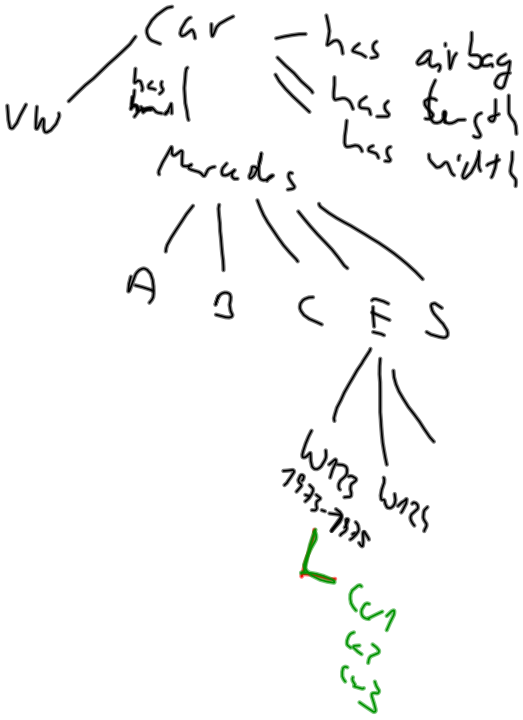

Player
Music
PlayerPreference
MusicRhythm


(A)
?prof
?p
prefers
Music
hiphop
outcome
FAN
(?p, ?prof)
type of
music
hip hop
FAN
(B)
listens to
Music
has
hip hop
Music
has
Fans
Likes
P
M
A
Price
Air
T


Person
has activities
listening/watching
sports music
exercise
sports
Music
has Preferences
Music
hip hop
Sports
soccer
Math
is FAN of
Bayern München
Entity
Club
Music group
gt 3x
Aha
plays
type of Music
Fan
Grou


Person
listens to
Singer
count
Integer
15
Reification

Class



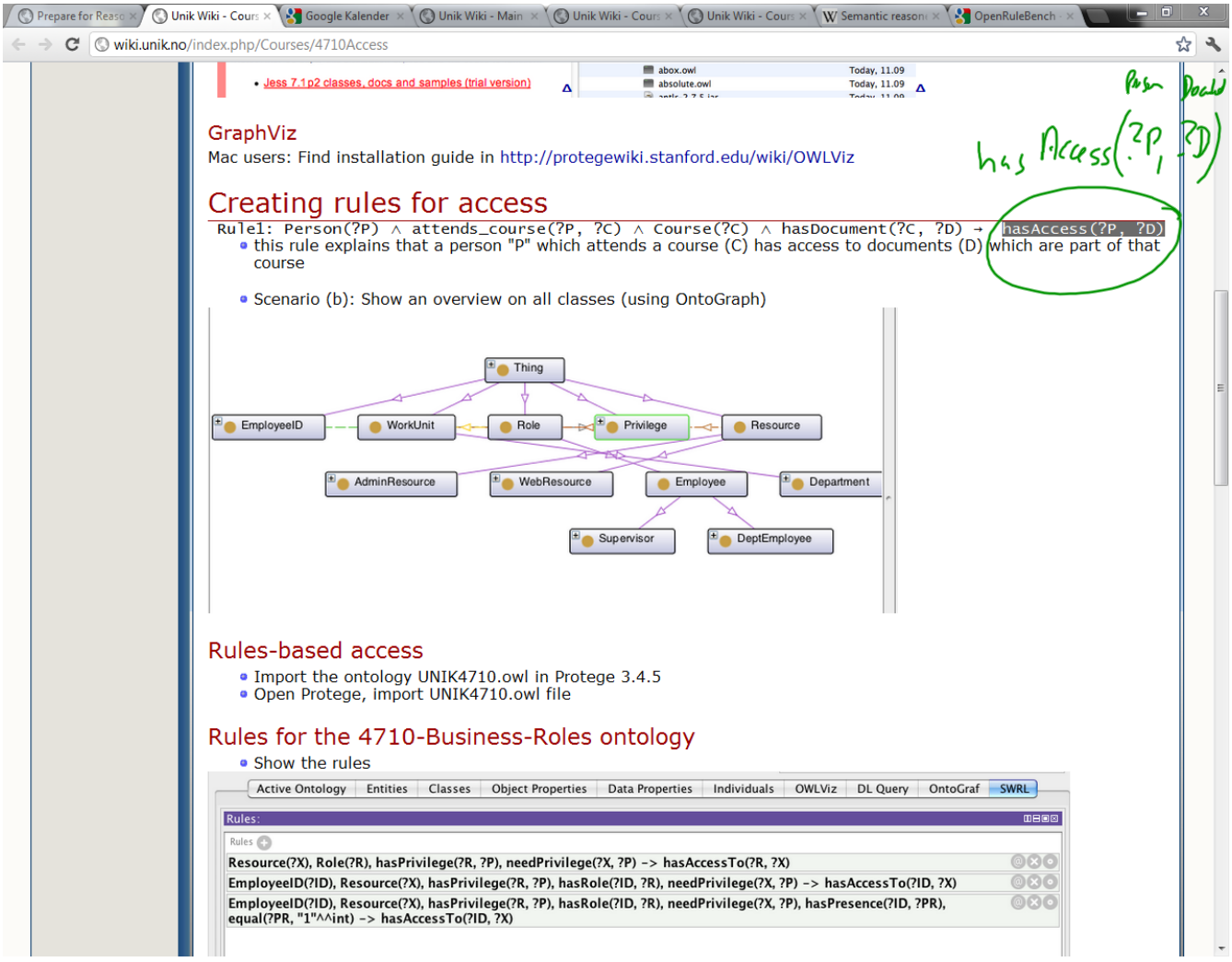
Mid Range

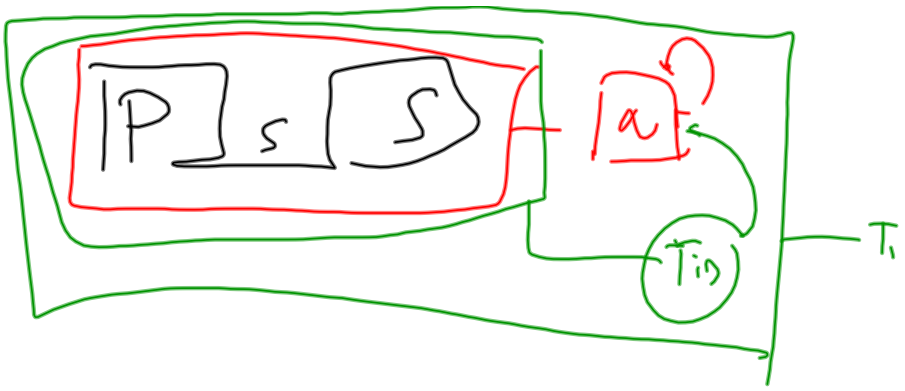
3,50m < length < 4,50

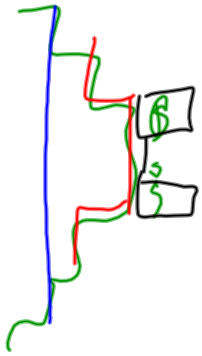
how do ? ⇒ ? classes — Golf, B-

? cars (= instances)

DF 37593


Prepare for Reaso
Unik Wiki - Cours
Google Kalender
Unik Wiki - Main
Unik Wiki - Cours
Unik Wiki - Cours
Semantic reasoni
OpenRuleBench
wiki.unik.no/index.php/Courses/4710Access
Jess 7.1p2 classes, docs and samples (trial version)
abox.owl Today, 11.09
absolute.owl Today, 11.09
GraphViz
Mac users: Find installation guide in http://protegewiki.stanford.edu/wiki/OWLViz
Creating rules for access
Rule1: Person(?P) ∧ attends_course(?P, ?C) ∧ Course(?C) ∧ hasDocument(?C, ?D) → hasAccess(?P, ?D)
this rule explains that a person "P" which attends a course (C) has access to documents (D) which are part of that course
Scenario (b): Show an overview on all classes (using OntoGraph)
Thing
EmployeeID
WorkUnit
Role
Privilege
Resource
AdminResource
WebResource
Employee
Department
Supervisor
DeptEmployee
Rules-based access
Import the ontology UNIK4710.owl in Protege 3.4.5
Open Protege, import UNIK4710.owl file
Rules for the 4710-Business-Roles ontology
Show the rules
Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OntoGraf
SWRL
Rules:
Rules
Resource(?X), Role(?R), hasPrivilege(?R, ?P), needPrivilege(?X, ?P) -> hasAccessTo(?R, ?X)
EmployeeID(?ID), Resource(?X), hasPrivilege(?R, ?P), hasRole(?ID, ?R), needPrivilege(?X, ?P) -> hasAccessTo(?ID, ?X)
EmployeeID(?ID), Resource(?X), hasPrivilege(?R, ?P), hasRole(?ID, ?R), needPrivilege(?X, ?P), hasPresence(?ID, ?PR), equal(?PR, "1"^^int) -> hasAccessTo(?ID, ?X)

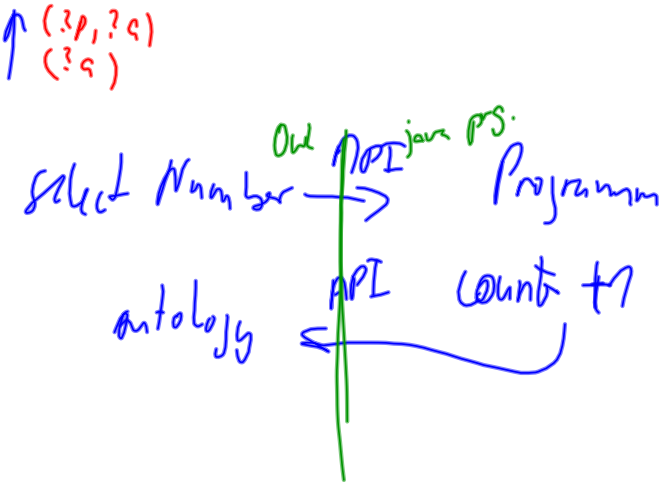

P
S
a
Tin
T1


S

protege owl api
.owl and java programme

If we assume that it is an owl class called Salary
with properties hasCurrency and hasAmount, you can write your rule as:

Person(?p) ^ hasMinSalary(?p, ?s) ^ hasAmount(?s, ?a) ^
swrlb:greaterThan(?a, 10000) -> query:select(?p)



List all adults with their age
Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) ->
> query:select(?p, ?age)
>

Examples from:
https://mailman.stanford.edu/pipermail/protege-owl/2007-July/002919.html