Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

Combining Ontologies with Rules

Knut Hinkelmann

Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

Copyright Note

The slides in this lecture are based on the following sources

- Vassilis Papataxiarhis: Combining Ontologies with RUles (Two Different Worlds?)
- Martin O'Connor: Efficiently Querying Relational Databases using OWL and SWRL

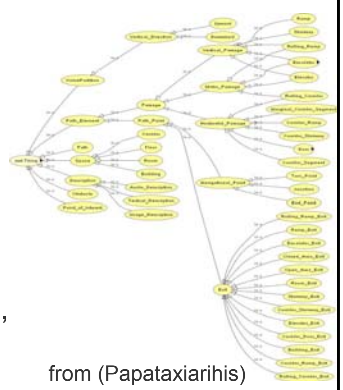
What is an Ontology?

- Ontologies are used not only to represent a domain of interest, but also DEFINE concepts, describe relations among them and insert individuals.
- An ontology is not just a taxonomy
- Ontology=

    Όν (categories of being) +

    λόγος (treatise)

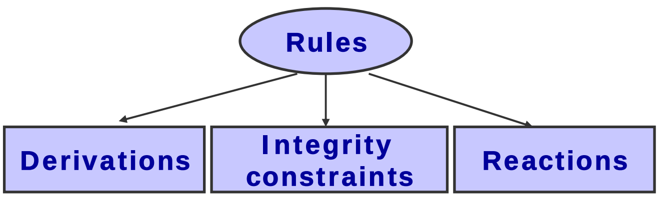
  (i.e. the philosophy of being, Metaphysics,



from (Papataxiarihis)

Rules

- Rules are mainly based on subsets of First Order Logic (FOL) + possible extensions.
- Rule Formalisms (in Semantic Web):
    - Semantic Web Rule Language (SWRL)
    - Answer Set Programming (ASP) (Datalog$^{\vee\neg}$)



from (Papataxiarihis)

Rule-based Systems are common in many domains

- Engineering: Diagnosis rules
- Commerce: Business rules
- Law: Legal reasoning
- Medicine: Eligibility, Compliance
- Internet: Access authentication

from (O'Connor)

Rule Markup (RuleML) Initiative

- Effort to standardize inference rules.
- RuleML is a markup language for publishing and sharing rule bases on the World Wide Web.
- Focus is on rule interoperation between industry standards.
- RuleML builds a hierarchy of rule sublanguages upon XML, RDF, and OWL, e.g., SWRL

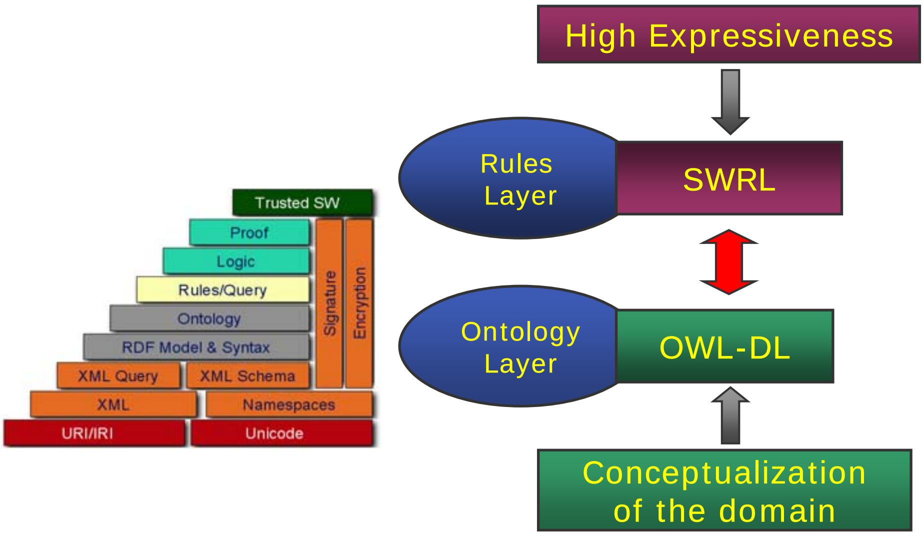
from (O'Connor)

The need for Ontologies and Rules

- Ontologies are based on Description Logics (and thus in classical logic).
  - An ontology model is easy to understand.
  - Reasoning is based on classification.
  - For the sake of decidability, expressiveness of ontology languages is restricted
- Rules are based on logic programming.
  - Expressiveness of rules has not the same limitations as description logics
  - Efficient reasoning support already exists.
  - Rules are well-known in practice.

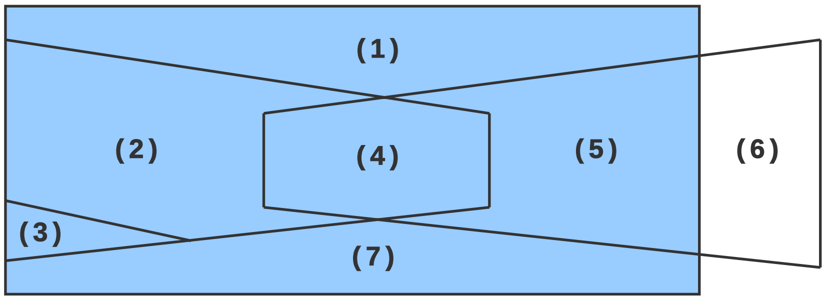
from (Papataxiarihis)

Usual combination



from (Papataxiarihis)

LP and Classical logic Overlap



FOL: (All except (6)),	(2)+(3)+(4): DLs
(4): Description Logic Programs (DLP),	(3): Classical Negation
(4)+(5): Horn Logic Programs,	(4)+(5)+(6): LP
(6): Non-monotonic features (like NAF, etc.)	(7): $\wedge$head and, $\vee$body

from (Papataxiarihis)

Basic Difficulties

Classical Logic vs. Logic Programming

- Monotonic vs. Non-monotonic Features
  - Open-world vs. Closed-world assumption
  - Negation-as-failure vs. classical negation
- Non-ground entailment
- Equality
- Decidability

from (Papataxiarihis)

Open-world vs. Closed-world assumption

- Logic Programming – CWA
  - If $KB \not\models a$, then $KB = KB \cup \neg a$
- Classical Logic – OWA
  - It keeps the world open.
  - KB:
  - $Man \sqsubseteq Person$, $Woman \sqsubseteq Person$
  - $Bob \in Man$, $Mary \in Woman$
  - Query: "find all individuals that are not women"

from (Papataxiarihis)

NAF vs. Classical negation

- Example:

KB_{LP}: *likesFootball(x)* ← *liverpoolSupporter(x)*

didNotCelebrateLVPEuroCup(x) ← *not liverpoolSupporter(x)*

likesFootball(gerrard).

KB_{CL}: $\forall x$ *liverpoolSupporter(x)* ⊃ *likesFootball(x)*

$\forall x$ ¬*liverpoolSupporter(x)* ⊃ *didNotCelebrateLVPEuroCup(x)*

likesFootball(gerrard).

KB_{LP} |= *didNotCelebrateLVPEuroCup(gerrard)*!

from (Papataxiarihis)

Non-ground entailment

- The LP-semantics is defined in terms of minimal Herbrand model, i.e. sets of ground facts.
- Example:

  likesFootball(x) ← liverpoolSupporter(x)

  liverpoolSupporter(x) ← liverpoolPlayer(x)

  liverpoolPlayer(gerrard).

- Both LP and classical logic yields the facts *liverpoolSupporter(gerrard), likesFootball(gerrard).*

Only the classical logic would allow further non-factual inferences, s.a.
liverpoolPlayer(x) ⊃ likesFootball(x)

from (Papataxiarihis)

Equality

- LP: Unique Name Assumption (UNA)
- Classical logic: different names may represent the same atom
- Example:

  differentPlayers(x,y) ← player(x), player(y), x≠y

  player(gerrard_of_liverpool).

  player(gerrard_of_england).

- In LP, we could conclude:

 ≠ 

differentPlayers(gerrard_of_liverpool, gerrard_of_england)

from (Papataxiarihis)

Decidability

- The largest obstacle!
  - Tradeoff between expressiveness and decidability.
- Facing decidability issues from 2 different angles
  - In LP: Finiteness of the domain
  - In classical logic (and thus in DL): Combination of constructs
- Problem:
  Combination of “simple” DLs and Horn Logic are undecidable. (Levy & Rousset, 1998)

from (Papataxiarihis)

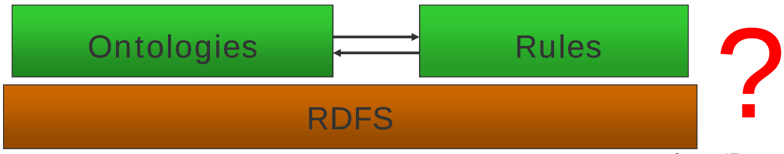
Rules + Ontologies

- Still a challenging task!
- A number of different approaches exists: SWRL, DLP (Grosof), dl-programs (Eiter), DL-safe rules, Conceptual Logic Programs (CLP), AL-Log, DL+log.
- 2 Main Strategies:
  - Strict Semantic Separation (Hybrid Approaches)
  - Tight Semantic Integration (Homogeneous Approaches)

from (Papataxiarihis)

Hybrid Approach

- Integration with strict semantic separation between the two layers.
- Ontology is used as a conceptualization of the domain.
- Rules cannot define classes and properties of the ontology, but some application-specific relations.
- Communication via a “safe interface”.
- Example: Answer Set Programming (ASP)



from (Papataxiarihis)

Answer Set Programming (ASP)

- Main Idea: models are solutions
- Generic Formula:

  $a_1 \lor \ldots \lor a_n \leftarrow b_1 \land \ldots \land b_k \land \text{not } b_{k+1} \land \ldots \land \text{not } b_m$,

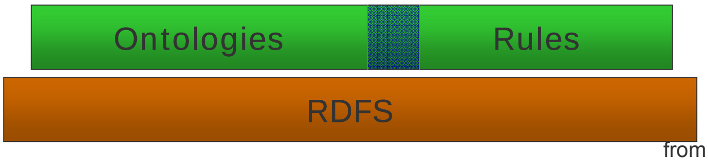
  where not: either NAF or strong negation

- Supports negation (NAF and strong) as well as disjunction
- Decidable

from (Papataxiarihis)

Homogeneous Approach

- Interaction with tight semantic integration.
- Both ontologies and rules are embedding in a common logical language.
- No distinction between rule predicates and ontology predicates.
- Rules may be used for defining classes and properties of the ontology.
- Example: SWRL, DLP



from (Papataxiarihis)

What is SWRL?

- SWRL is an acronym for Semantic Web Rule Language.
- SWRL is intended to be the rule language of the Semantic Web.
- SWRL includes a high-level abstract syntax for Horn-like rules.
- All rules are expressed in terms of OWL concepts (classes, properties, individuals).

from (O‘Connor)

SWRL

- Extend OWL axioms to include Horn-like clauses.
- Maximum compatibility with OWL
- Built on top of OWL (same semantics)
- Generic Formula:

  $a_1 \wedge \ldots \wedge a_n \leftarrow b_1 \wedge \ldots \wedge b_k$

- Limitations
  - Negation, Disjunction
  - Undecidable

from (Papataxiarihis)

Example SWRL Rule: Has uncle

SWRL Rule: Has uncle

hasParent(?x, ?y) ^ hasBrother(?y, ?z) → hasUncle(?x, ?z)

SWRL Rule with Named Individuals: Has brother

Person(Fred) ^ hasSibling(Fred, ?s) ^ Man(?s)
→ hasBrother(Fred, ?s)

SWRL Rule with Literals and Built-ins: is adult?

Person(?p) ^ hasAge(?p,?age) ^ swrlb:greaterThan(?age,17)
→ Adult(?p)

from (O'Connor)

SWRL Characteristics

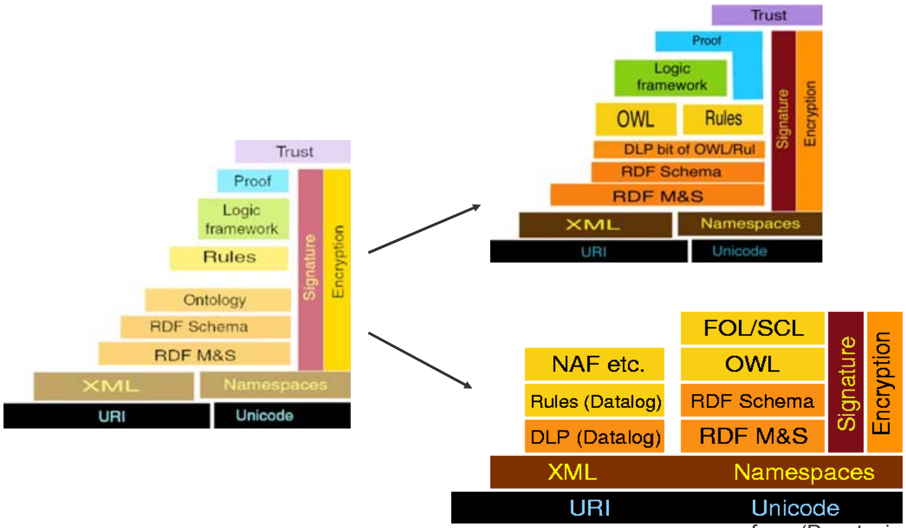
- W3C Submission in 2004: http://www.w3.org/Submission/SWRL/
- Based on OWL-DL
- Has a formal semantics
- Rules saved as part of ontology
- Increasing tool support: Bossam, R2ML, Hoolet, Pellet, KAON2, RacerPro, SWRLTab
- Can work with reasoners

from (O'Connor)

Übung

- Sie sollen für ein Unternehmen ein semantisches Informationssystem entwickeln, das auf einer Datenbank aufbaut. Konzepte werden in einer Ontologie definiert. Das System enthält Informationen über Kunden und Einkäufe.
- Repräsentieren Sie die folgenden Informationen als Ontologien mit Regeln nach dem Hybridansatz.
- *Ein Unternehmen hat Kunden, wobei Kunden juristische oder natürliche Personen sein können. Kunden mit einer goldenen Kundenkarte sind Goldkunden. Ein Kunde, der für mehr als 10'000 Euro pro Jahr einkauft bekommt die goldene Kundenkarte. Ein Goldkunde bekommt bei einem Einkauf über 200 Euro 10 % Rabatt. Kunden, die als kreditwürdig eingestuft sind, dürfen mit Kreditkarte zahlen. Goldkunden sind kreditwürdig. Der Gesamtbetrag einer Bestellung errechnet sich aus der Summe der Einzelposten abzüglich des Rabatts.*
- Was sind Inhalte der Ontologie, was Prädikate, die durch Regeln definiert werden? Warum?

Two Semantic Webs?



from (Papataxiarihis)

Tools

- Ontology Editors
  - Protégé, Swoop, TopBraid Composer
- Rule Editors
  - Protégé (SWRL-Tab)
- Ontology Reasoners
  - RacerPro, Bossam, Pellet, Fact++
- RuleEngines
  - Bossam, Jess, Jena Framework (only JRules)
  - ASP solvers: DLV, Smodels, nomore++

from (Papataxiarihis)

SWRLTab

- A Protégé-OWL development environment for working with SWRL rules
- Supports editing and execution of rules
- Extension mechanisms to work with third-party rule engines
- Mechanisms for users to define built-in method libraries
- Supports querying of ontologies

from (O'Connor)

SWRLTab

WikiHomePage | RecentChanges | Page Index
Login (create account)

Your Visited Pages
SWRLTab
View Backlinks
Search

The SWRLTab is an extension to Protege-OWL that supports editing and execution of SWRL rules. It provides a graphical editor to create and modify SWRL rules in an OWL knowledge base. It also provides extension mechanisms to support the execution of SWRL rules with a variety of rule engines. At present, the Jess rule engine is supported. (6GX)

The SWRLTab has four main software components: (6GY)

- **SWRL Editor:** The editor supports editing and saving of SWRL rules in an OWL knowledge base. See the SWRL Editor FAQ for more details. An introduction to the SWRL language can be found here. (6ZI)
- **SWRL Factory:** The factory provides high-level Java APIs that support the creation and modification of SWRL rules in an OWL knowledge base. This API can be used by developers who wish to work with SWRL rules in their applications. See the SWRL Factory FAQ for more details. (6H0)
- **SWRL Bridge:** The bridge provides the infrastructure necessary to incorporate rule engines into Protege-OWL to execute SWRL rules. See the SWRL Rule Engine Bridge FAQ for more details. A bridge for the Jess rule engine is provided in the Protege-OWL distribution. A user interface called the SWRLJessTab is also provided to interact with this bridge. The hope is that bridges for other rule engines will be developed by the Protege-OWL community and than an array of inference mechanism will become available for executing SWRL rules. (6H1)
- **SWRL Built-in Bridge:** SWRL built-ins are predicates that accept one or more arguments. These predicates can be used in SWRL rules to support the definition of arbitrary user-defined built-ins, which can then be used in rules. The SWRLTab has a subcomponent called the built-in bridge that provides a mechanism to define Java implementations of SWRL built-ins. These implementations can then be dynamically loaded by the bridge and invoked from a rule engine. (6H2)

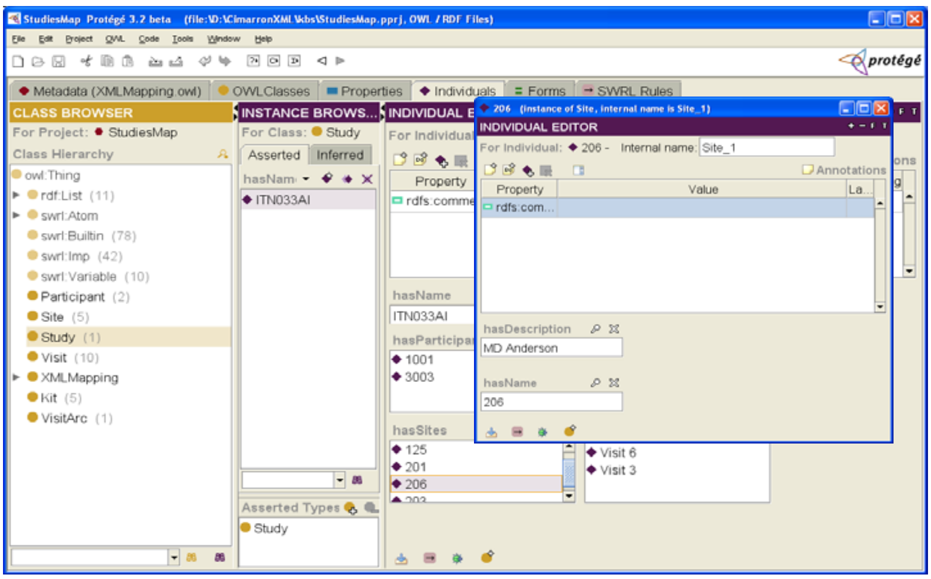
SWRLTab: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

from (O'Connor)

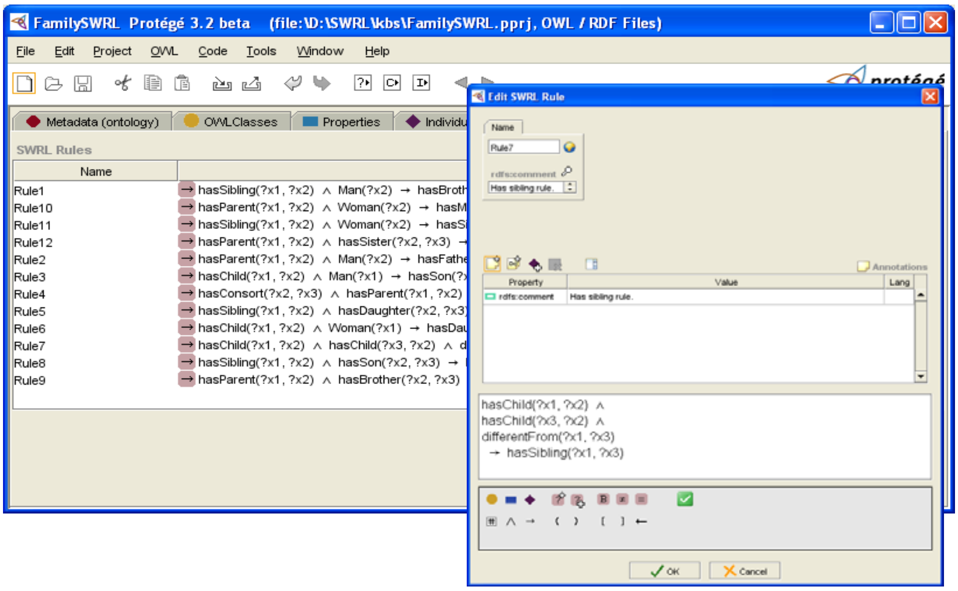
The SWRL Editor

- The SWRL Editor is an extension to Protégé-OWL that permits the interactive editing of SWRL rules.
- The editor can be used to create SWRL rules, edit existing SWRL rules, and read and write SWRL rules.
- It is accessible as a tab within Protégé-OWL.

from (O'Connor)



from (O'Connor)



from (O'Connor)

SWRL Java API

- The SWRL API provides a mechanism to create and manipulate SWRL rules in an OWL knowledge base.
- This API is used by the SWRL Editor. However, it is accessible to all OWL Plugin developers.
- Third party software can use this API to work directly with SWRL rules and integrate rules into their applications
- Fully documented in SWRLTab Wiki.

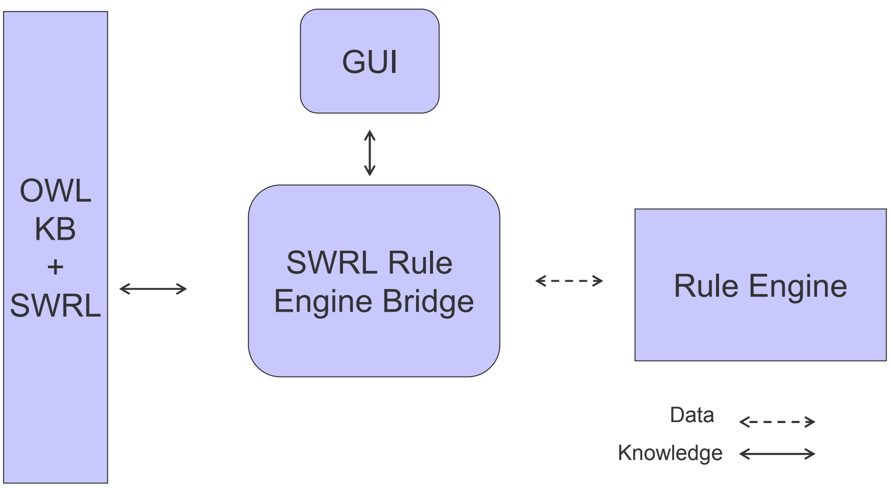
from (O'Connor)

Executing SWRL Rules

- SWRL is a language specification
- Well-defined semantics
- Developers must implement engine
- Or map to existing rule engines
- Hence, a bridge…

from (O'Connor)

SWRL Rule Engine Bridge



from (O'Connor)

SWRL Rule Engine Bridge

- Given an OWL knowledge base it will extract SWRL rules and relevant OWL knowledge.
- Also provides an API to assert inferred knowledge.
- Knowledge (and rules) are described in non Protégé-OWL API-specific way.
- These can then be mapped to a rule-engine specific rule and knowledge format.
- This mapping is developer's responsibility.

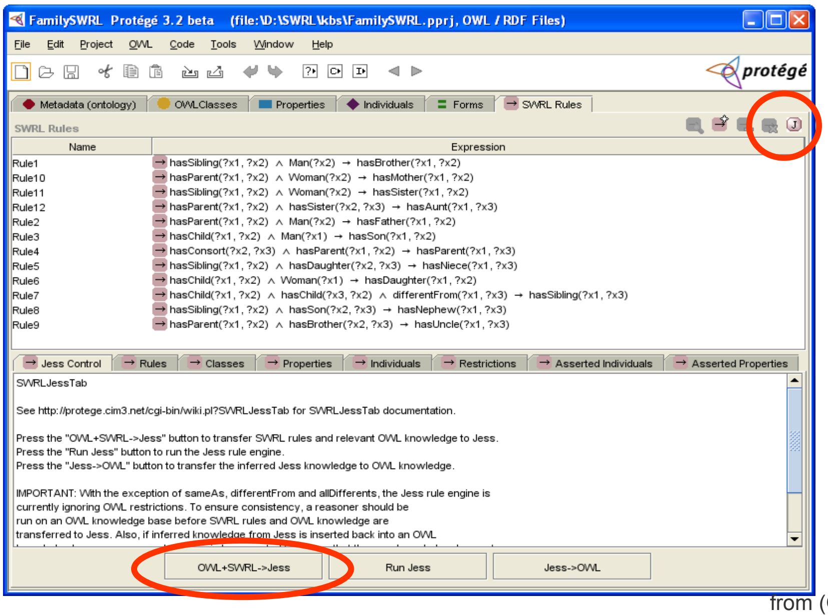
from (O'Connor)

Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft

SWRL Bridge is used to Integrate Jess Rule Engine with Protégé-OWL

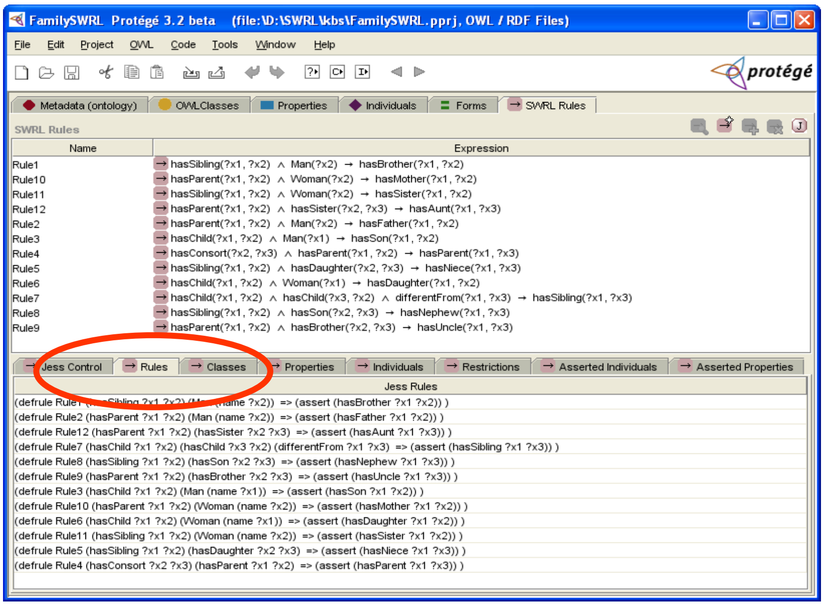
- Jess is a Java-based rule engine.
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users
- Has been used in Protégé-based tools, e.g., JessTab.

from (O'Connor)

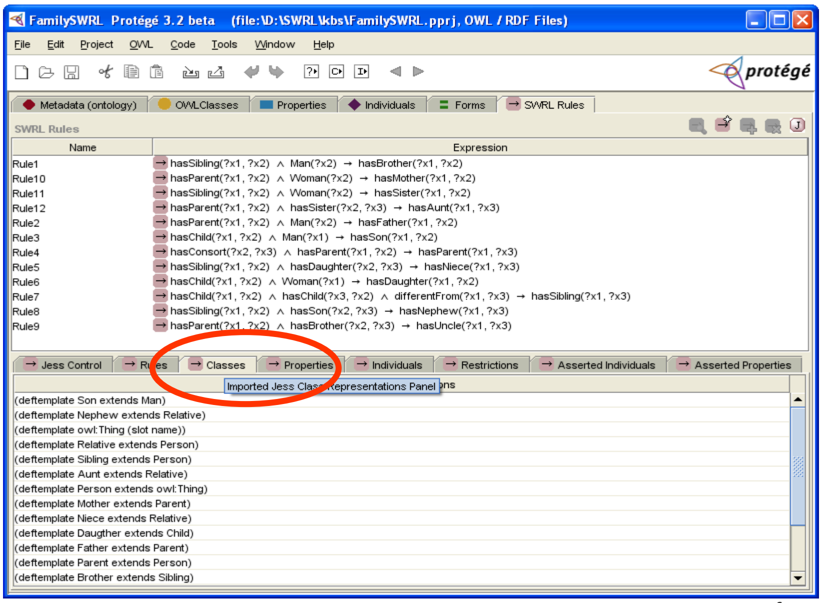


from (O'Connor)

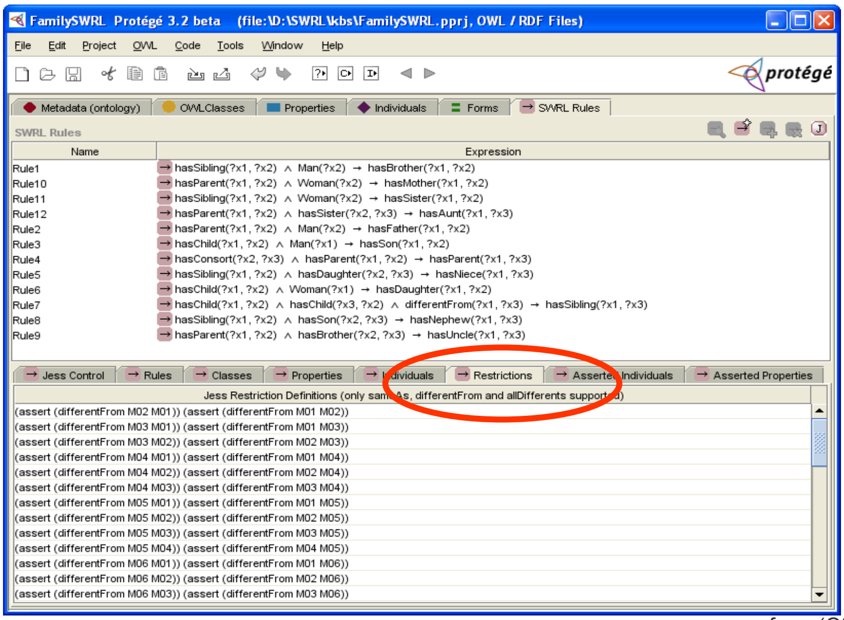
Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft



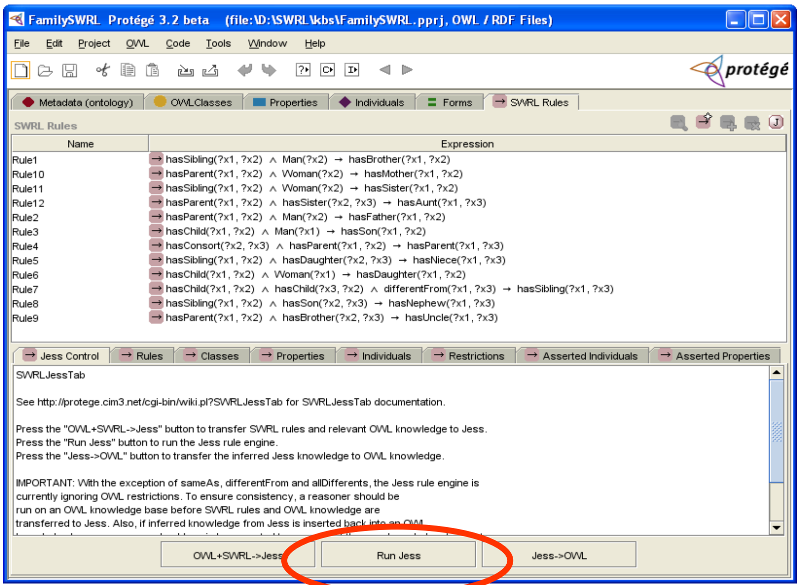
from (O'Connor)



from (O'Connor)

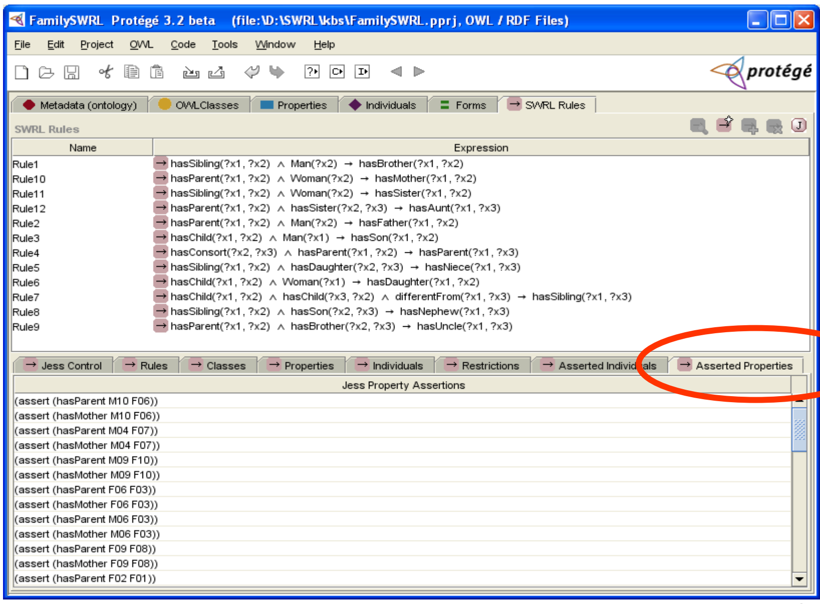


from (O'Connor)



from (O'Connor)

Fachhochschule Nordwestschweiz
Hochschule für Wirtschaft



from (O'Connor)

Outstanding Issues

- SWRL Bridge does not know about all OWL constraints:
  - Contradictions with rules possible!
  - Consistency must be assured by the user incrementally running a reasoner.
  - Hard problem to solve in general.
- Integrated reasoner and rule engine would be ideal.
- Possible solution with KAON2.

from (O'Connor)

SWRL Built-in Bridge

- SWRL provides mechanisms to add user-defined predicates, e.g.,
  - hasDOB(?x, ?y) ^ temporal:before(?y, '1997')…
  - hasDOB(?x, ?y) ^ temporal:equals(?y, '2000')…
- These built-ins could be implemented by each rule engine.
- Core SWRL built-ins defined by:
  - http://www.w3.org/2003/11/swrlb
- Provides commonly needed built-ins, e.g., add, subtract, string manipulation, etc.
- Normally aliased as 'swrlb'.

from (O'Connor)

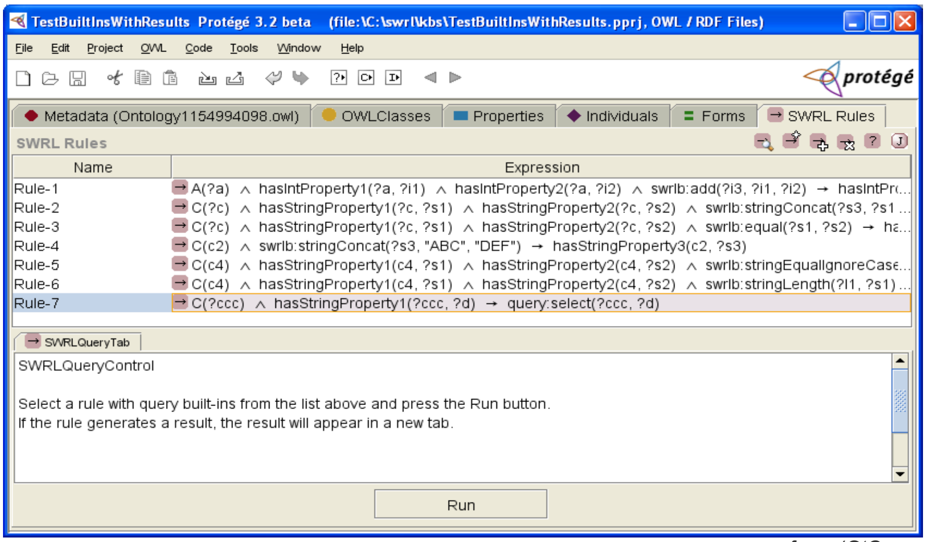
SWRL and Querying

- SWRL is a rule language, not a query language
- However, a rule antecedent can be viewed as a pattern matching specification, i.e., a query
- With built-ins, language compliant query extensions are possible.
- Return all adults in ontology:

  Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17)
  → swrlq:select(?p) ^ swrlq:orderBy(?age)

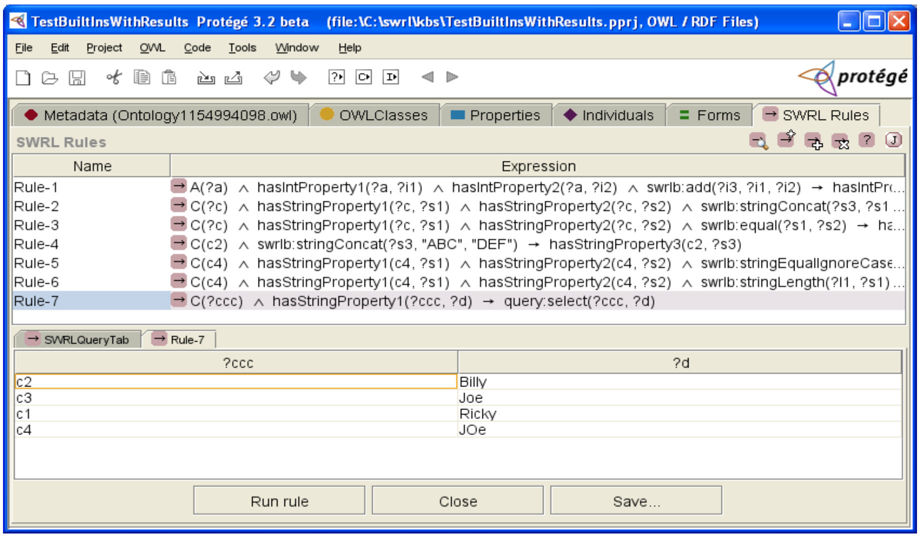
from (O'Connor)

SWRLQueryTab



from (O'Connor)

SWRLQueryTab: Displaying Results



from (O'Connor)

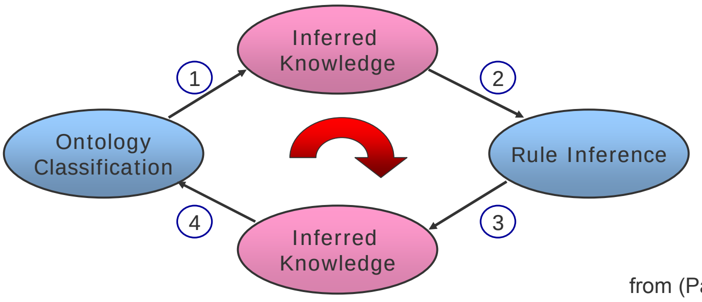
Querying: Semantic Issues

- Syntactic SWRL conformance is easy
- However, SWRL is based on OWL-DL so assumes open world semantics
- Querying closes the world, e.g., how many adults in ontology?
- Should not make inferences based on query results – nonmonotonicity!

from (O'Connor)

Limitations (1/2)

- The rule inference support is not integrated with an OWL classifier.
  - So, new assertions by rules may violate existing restrictions in ontology. New inferred knowledge from classification may in turn produce knowledge useful for rules.



from (Papataxiarihis)

Limitations (2/2)

- Existing solution:
  - Solve these possible conflicts manually.
- Ideal solution:
  - Have a single module for both ontology classification and rule inference.
- What if we want to combine non-monotonic features with classical logic?
  - Partial Solutions:
    - ASP
    - Externally (through the use of appropriate rule engines)

from (Papataxiarihis)

Some References

- Reasoning with Rules and Ontologies. Thomas Eiter, Giovambattista Ianni, Axel Polleres, Roman Schindlauer, Hans Tompits, 2006.
- Description Logic Programs: Combining Logic Programs with Description Logics. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, 2003.
- Combining Rules and Ontologies: A survey. G. Antoniou, C. V. Damnasio, B. Grosof, I. Horrocks, M. Kifer, J. Maluszynski, and P. F. Patel-Schneider, 2005.
- Semantic Web Architecture: Stack or Two Towers?. Horrocks, I., Parsia, B., Schneider, P., Hendler, J., 2005.
- Can OWL and Logic Programming Live Together Happily Ever After?. Motik, B., Horrocks, I., Rosati, R., Sattler, U., 2006.