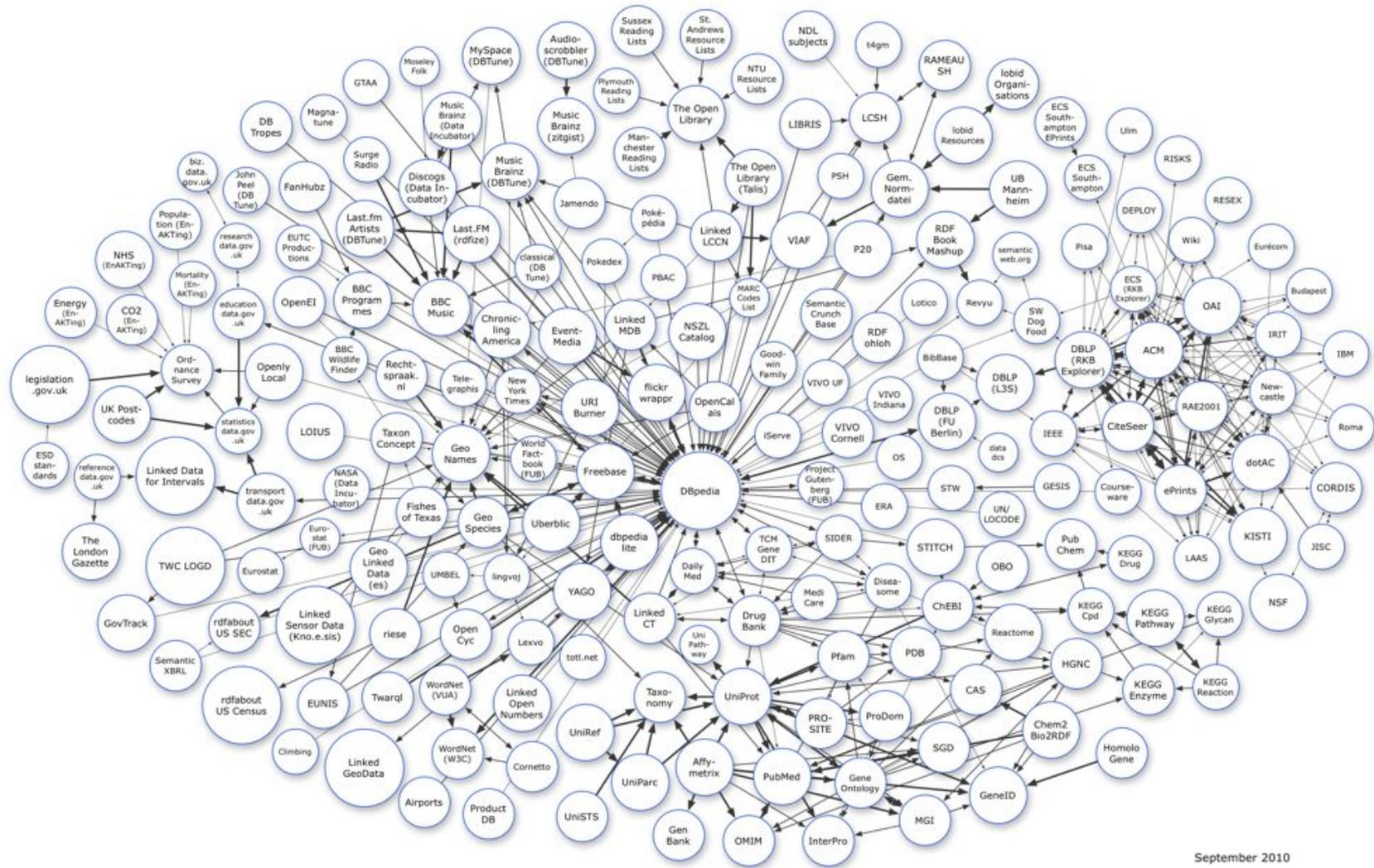


Semantic Web и продукционная модель знаний

Катериненко Р. С.

аспирантура ИТМО ВТ



RDF

- Hayes



RDF Semantics

W3C Recommendation 10 February 2004

This Version:

<http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>

Latest Version:

<http://www.w3.org/TR/rdf-mt/>

Previous Version:

<http://www.w3.org/TR/2003/PR-rdf-mt-20031215/>

Editor:

[Patrick Hayes](mailto:phayes@ihmc.us) (IHMC) <phayes@ihmc.us>

- Horst, H. J. *Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity.*
- etc

RDF

- Hayes



RDF Semantics

W3C Recommendation 10 February 2004

This Version:

<http://www.w3.org/TR/2004/REC-rdf-nt-20040210/>

Latest Version:

<http://www.w3.org/TR/rdf-nt/>

Previous Version:

<http://www.w3.org/TR/2003/PR-rdf-nt-20031215/>

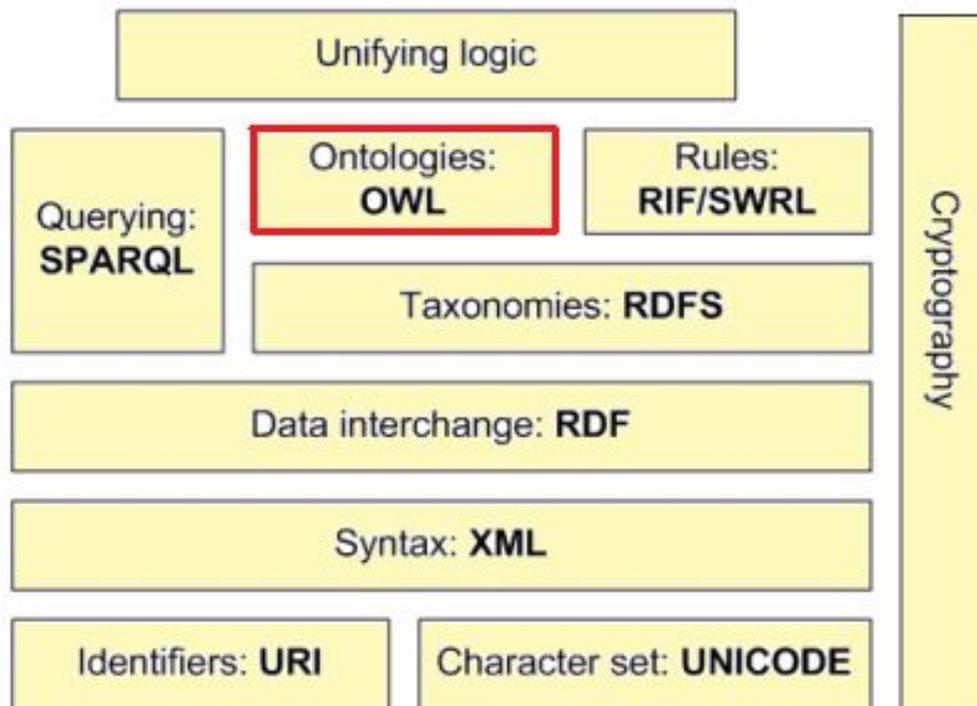
Editor:

[Patrick Hayes](mailto:phayes@ihmc.us) (IHMC) <phayes@ihmc.us>

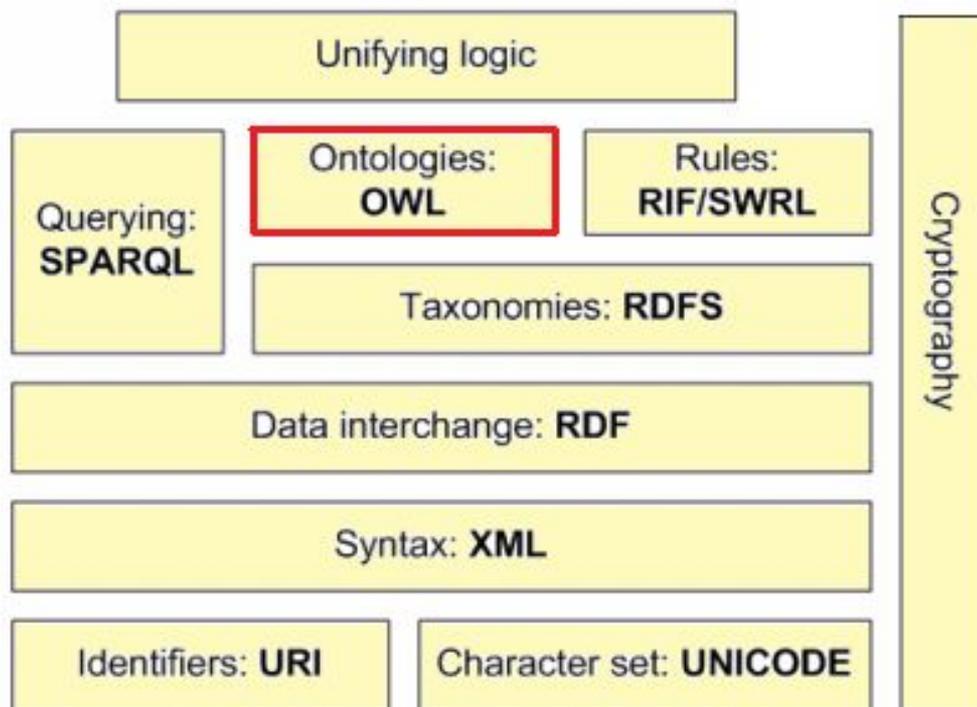
- Sesame
- Virtuoso

- Horst, H. J. *Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity.*
- etc

Логическая модель



Логическая модель



- Pallet
- Hawk
- Racer
- Fact

Логическое программирование

Логическое программирование – общее название для языков программирования, программы которых состоят из продукций.

- Prolog
- Datalog

Числа Фибоначчи:

$F(0,0)$

$F(1,1)$

$F(X+Y,N+2) :- F(X,N), F(Y,N+1), N>1$

? $F(X,100)$

Стратегии вычисления продукций

•Forward-chaining, (bottom-up)

Последовательное применение продукций к фактам, а затем и к комбинациям фактов и выведенных фактов

Достоинства: при ответе на запрос не требуется никакого логического вывода

Недостатки: генерация избыточного количества фактов
(попробуйте посчитать числа Фибоначчи)

•Backward-chaining, (top-down)

Сначала целью вывод объявляется запрос пользователя.

Ризонер пытается найти факты для текущей цели, если их нет,

то ищется продукции, которые могут быть использованы для текущей цели.

Найденная продукция объявляется текущей целью вывода и процесс повторяется.

Преимущества: нет порождения избыточных фактов

Недостатки: для сложных графов вычисления не эффективны,
некоторые цели могут проходиться много раз.

Что еще можно делать с продукциями?

Мы используем комбинированную стратегию. При этом применяются продукции к фактам, но мы ограничиваем множество всех продукций, только теми, которые могут быть действительно необходимы для ответа на запрос.

Существуют разные способы это сделать:

- Алгоритм magic set
- Стратификация
- Алгоритм Rete
- Переупорядочивание продукций

Мы используем зависимость продукций по предикатам

Граф зависимости продукций

Запрос:

?a(x)

Продукции:

R1: a(x) :- p(x), q(x)

R2: p(y) :- p1(y)

R3: p(k) :- p1(k)

R4: q(z) :- q1(z)

Генерация SQL

- Предикат $p(x)$ определяет множество :
Select * from F where F.s like "P"

- Join

$a(x) :- p(x), q(x)$

$p(x) - S1$

$q(x) - S2$

Select * from S1 join S2

- Union

$p(y) :- p1(y) //S3$

$p(k) :- p2(k) //S4$

Select * from S3 union S4

- Вложенный подзапрос:

$S1 = S3 \text{ union } S4 = \text{Select * from (Select * from S3 union S4)}$

Генерация продукций

Продукции => SQL

+ Совмещение SQL запросов с семантическими к реляционной СУБД

Генерация продукций

SPARQL => Продукции => SQL

+ Совмещение SQL запросов с семантическими к реляционной СУБД

+ Инфраструктура для семантических приложений

Генерация продукций

- + Быстрая инфраструктура для семантических приложений:
 - на 20% быстрее, чем Jess (Rete)

SPARQL => Продукции => SQL

- + Совмещение SQL запросов с семантическими к реляционной СУБД
- + Инфраструктура для семантических приложений



Спасибо!