SPARQL Extensibility

- Arbitrary functions in FILTERs
  - Identified by URI
  - Can extend operators as well
- New semantics for Basic Graph Patterns
  - BGPs *extract* mappings from data sets
    - The algebra is independent of the extraction
      - We hope!
  - One document/graph has many semantics
    - Simple, RDF, RDFS, OWL....
  - Queries (should be) sensitive to the semantics
    - See prior unit for some examples (RDF, etc.)

Trickiness

- Controlling answers
  - Too many:
    - Simple entailment can yield infinite answers
  - Too few:
    - Finding all proofs of an answer difficult
- New sorts of issue
  - E.g., inconsistent data or equality
- Performance
  - Bare consistency of OWL DL is NEXPTIME
  - Query languages very expressive!
  - Performance model unclear to users

Standardizing these things is hard

- Not a lot of experience
  - Conjunctive query for DLs is (fairly) new
    - Theoretically and implementationally
  - Concept language expressive
    - Can express many common queries
  - Lots of decisions
- Database experience not always reliable
- LP experience not always reliable

Inconsistency

- Some logics can express **inconsistent** data
  - RDF (with certain datatypes), RDFS, and OWL
  - Inconsistencies entail **everything**
    - So, every mapping is a "**correct**" answer!
  - Inconsistencies often signal error
    - But may indicate mere disagreement!
- What should a query engine return?
  - Nothing
  - No answers, but explanations
  - Implementation dependent "best" answers
  - Answers from a weaker logic

"Strange" queries

- In RDF(S)
  - Thin distinction between schema and data
    - Schema language very inexpressive
  - So easy to treat the schema and data uniformly
- In OWL-DL
  - Strong distinction between schema and data
    - TBox vs. Abox
    - Concept language very expressive
  - OWL Full tries to do the RDF thing
    - High cost: Undecidability, no implementations, hard to understand semantics

Types of Query Variables

- 2 key axis of a variable with 4 combinations

  A) Distinguished

  B) "Semi-"distinguished

  C) "Projected away"

  D) Non-distinguished

		In head of query
Binds to names only	A. Yes/Yes	A. Yes/No
	B. No/Yes	A. No/No

- In a databases, only A and C are possible
  - C and D collapse (no non-named entities)
- In DLs, A and D are standard
  - D make query answering harder!
- In SPARQL/RDF, all variables are B

Considerations

- A and C are easiest
  - Bind only to names
  - Reduces SPARQL over RDF to relational (mostly)
  - Tables don't contain variables
  - Lean vs. non-lean source graphs don't matter
  - Miss answers!
- Non-distinguished
  - Basic answering is much more difficult (esp. in DL)
  - But tables and algebra remain the same
  - Miss some and co-reference of answers

Semi-distinguishedness

- Semi-distinguished
  - Basic answering is much more difficult
    - Less so in RDF
    - Not clearly defined yet
      - A binding is supposed to be true in all models
      - How exactly to identify generated individuals across models?
  - New issues for the algebra and final results
    - Even in RDF!

(Non-)distinguished Example

Data:

```
:bob :loves _:x.
_:x :loves :bob.
_:y :loves :sally
:sally :loves :sheevah.
```

In (pseudo-)OWL:

```
:bob a [onProperty :loves
        someValuesFrom
            [onProperty :loves;
             hasValue :bob].
:sally a [onProperty inv(:loves);
          someValuesFrom Thing]
```

SELECT ?lover

WHERE { ?lover :loves ?beloved}

?lover
:sally

WHERE { ?lover :loves _:beloved}

?lover
:bob
:sally

Semi-distinguished Example

- Data (co-reference in data (could) show up in results):

```
:bob :loves _:x.
_:x :loves :bob.
_:y :loves :sally
:sally :loves :sheevah.
```

SELECT ?lover ?beloved

WHERE { ?lover :loves ?beloved}

?lover	?beloved
:bob	_:G1
_:G1	:bob
_:G2	:sally
:sally	:sheevah

Oedipus Example

```
SELECT ?x FROM
<http://www.mindswap.org/ontologies/oedipus>
```

```
WHERE { ?x ns:hasChild
        [ rdf:type ns:Patricide;
          ns:hasChild ?y].
          ?y rdf:type ns:NotPatricide.}
```

NO RESULTS

```
WHERE { ?x ns:hasChild
        [ rdf:type ns:Patricide;
          ns:hasChild
            [ rdf:type ns:NotPatricide]]}
```

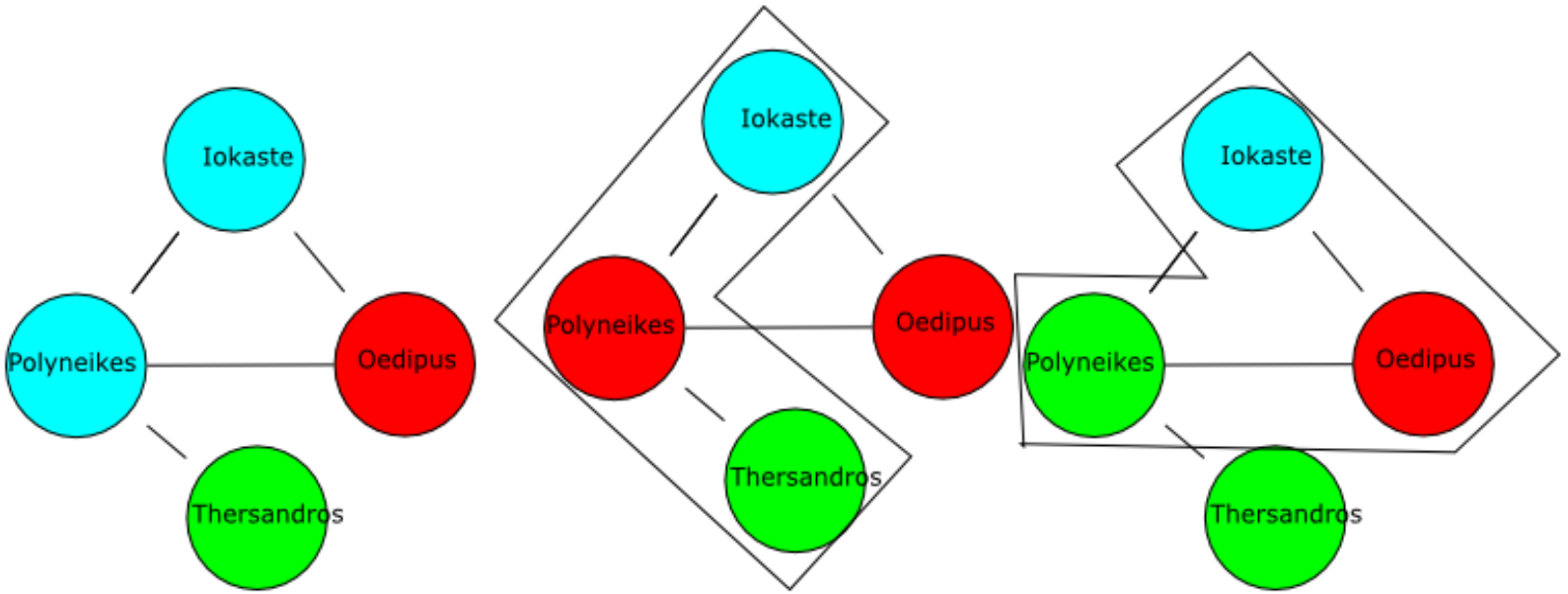
x
:IOKASTE

(This relies on a set of OWL DL axioms in the background.)
(Example from the Description Logic Handbook, chapter 2.)

The data

```
:Patricide     a owl:Class .
:NotPatricide   a owl:Class;
     owl:complementOf :Patricide .
:hasChild      a owl:ObjectProperty .

:IOKASTE
     ns:hasChild :OEDIPUS;
     ns:hasChild :POLYNEIKES .

:OEDIPUS      a ns:Patricide
     ns:hasChild :POLYNEIKES .

:POLYNEIKES
     ns:hasChild :THERSANDROS .

:THERSANDROS      a ns:NotPatricide .
```

Reasoning with Oedipus



Reasoning by cases

- Iokaste hasChild Polyneikes, Oedipus
- Oedipus is a Parricide and Thersandros is not
- Polyneikes is either a Parricide or not a Parricide

Thoughts and Implications

- If ?y in the Oedipus query were semi-distinguished:
  - What should its value be?
  - Polyneikes or Thersandros?
  - _:x instance of ({Polyneikes} or {Thersandros})?
- In the simple/RDF case:
  - Entailment can only introduce redundant BNodes
  - The data could be non-lean
- Suggests new ways to treat Bnodes
  - By the algebra

Counting

- What is a redundant answer?
  - BNodes in answers can be tricky
    - Several *notions* of redundancy
    - Distinguish between redundancy due to algebra and stated redundancy and inferred redundancy
  - Equality can be tricky
    - If two answers differ only by the value of one binding, and those values are inferred to be sameAs, how many answers?
    - No UNA in RDF-OWL
- We could count **answers** (instead of entities)
  - But then answers proliferate, often pointlessly

Query Variable Position

- "Conjunctive ABox queries"
  - Standard in DL systems: KAON2, Racer, Pellet
  - No variables in property or class positions
- "Higher order" queries
  - `?x rdf:type ?C. ?C rdfs:subClassOf ?D.`
  - Careful restrictions make this feasible
- "Syntax reflective" queries
  - `?s ?p ?o`, where `?p` can bind to e.g., intersectionOf
  - `?x a [a Restriction; someValuesFrom ?C]`
    - Only bind to asserted axioms? (essentially SPARQL/RDF)
- Variants of latter two coming (see OWLED)

More General OWL Queries

- What should `?x rdf:type ?C` return?
  - Graph: `:bob rdf:type :Person`
  - Some possible mappings for ?C:
    - :Person
    - owl:Thing
    - unionOf (:Person, not :Person)
    - Bnode problem writ large!
- Some hope (but very sketchy)
  - Concept matching and unification
  - Traditionally defined for logics without disjunction

For more thoughts

- See my presentation “Sparqling Queries”
  - http://www.cs.man.ac.uk/~bparsia/2006/row-tutorial/
- Also, at OWLED 2007 SPARQL/OWL?
  - Just enough to cover and encourage current implementations
  - Syntax for Distinguished and Non-distinguished
    - No Semi-distinguished except perhaps for explicit bnodes
  - Mixed Tbox/ABox queries
    - I.e., liberalize variable position
    - But also perhaps define a profile for strict Abox query