Querying the Semantic Web with Racer + nRQL

Volker Haarslev, Ralf Möller, and Michael Wessel

Presented By: Chris Hathaway

Overview

- nRQL (new Racer Query Language) – description logic query language to retrieve A-box individuals that satisfy conditions
- Much more expressive than previous concept-based retrieval languages
- Implemented in the Racer system; applicable to OWL semantic web repositories.

Racer

- Description logic system for very expressive DL $\mathcal{ALCQHI}_{R^+}(D^-)$ (aka $\mathcal{SHIQ}$)
- Can process OWL Lite, OWL DL knowledge bases with a highly optimized tableau calculus
- Support for processing RDF/RDFS/DAML/OWL
- Converts knowledge bases into DAML/OWL format with very few restrictions

nRQL

- new Racer Query Language (pronounced *nercle*) extends Racer's current functional API for querying a knowledge base
- Made up of Individuals, Variables, Object Names, Query Atoms, Ground Query Atoms
- Includes $\vee$, $\wedge$, $\neg$, and \ (negation as failure) operators

Definition 1: Individuals, Variables, Object Names

- Individuals – i, j, $betty$, $charles$
- Variables – x, y, …
- Object names – a, b, …

Definition 2: Query Atoms

- Basic building blocks of queries
  - Unary Concept: $C(a)$
  - Binary role: $R(a, b)$
  - Binary constraint: $P(f(a), g(b))$
  - Binary same-as: $same\_as(a, i)$
  - Unary has-known-successor: $has\_known\_successor(a, R)$
  - Negated (negation by failure): If A is nRQL atom, so is $\backslash(A)$
- Can be connected by normal logic operators

Definition 3: nRQL Queries

- Basic form:

$$ans(a_{i1}, \ldots, a_{im}) \leftarrow body(a_1, \ldots, a_n)$$

Projected variables

Query Atoms
(where clause)

Example: ans(child_of_betty) ← has_child(betty, child_of_betty)

Definition 4: Ground Query Atoms

- A query atom that does not contain any variables
- Can be resolved in knowledge base
- Positive Ground Query Atom – ground query atom that is not negated

Definition 5: Entailment of Positive Ground Query Atoms

- If A = C(i), then $\mathcal{I} \models$ A iff $i^{\mathcal{I}} \in R^{\mathcal{I}}$

…

- Unique Name Assumption:
  - If A = $same\_as(i, i)$, then $\mathcal{I} \models$ A
  - If A = $same\_as(i, j)$, then $\mathcal{I} \not\models$ A
- If A = $has\_known\_successor(i, R)$, then $\mathcal{I} \models$ A iff for some j $\in$ inds(A); $\mathcal{I} \models$ R(i, j)

Definition 6: Truth of Ground Query Atoms

- If A is positive (no '\'): $\mathcal{K} \models_{NF} A$ iff $\mathcal{K} \models A$
- Otherwise: $\mathcal{K} \models_{NF}$ \(A) iff $\mathcal{K} \not\models_{NF} A$
- Example: A-box {woman(betty)}
  - woman(betty) TRUE
  - mother(betty) FALSE
  - \(mother(betty)) TRUE
  - ¬(mother(betty)) FALSE

Example Knowledge Base

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), && woman(betty), && brother(charles), \\
&(\leq 1 has\_sibling)(charles), && has\_sister(eve, doris), && \\
&has\_child(alice, betty), && has\_child(alice, charles), && has\_child(betty, doris), \\
&has\_child(betty, eve), && has\_sibling(charles, betty), && has\_sister(doris, eve)
\end{aligned}
$$

Sample Queries

ans(x) ← grandmother(x)
{(alice)}

ans(x) ← (¬grandmother)(x)
{(charles)}

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

ans(x) ← \(grandmother(x))
{(doris)(eve)(charles)(betty)}

ans() ← grandmother(x)
TRUE

Sample Queries (con't)

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

ans(mother, child) ← has_child(mother,child)

{(betty, doris)(betty, eve)(alice, betty)(alice, charles)}

Sample Queries (con't)

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

$ans(x,y) \leftarrow mother(x) \wedge man(y) \wedge has\_child(x,y)$

{(alice, charles)}

Sample Queries (con't)

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

$$ans(x,y) \leftarrow man(x) \wedge man(y)$$

$$\varnothing$$

Sample Queries (con't)

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

$$ans(x,y) \leftarrow man(x) \wedge man(charles)$$

$$\varnothing$$

Sample Queries (con't)

Find A-box individuals who *do not* have a child.

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

ans(x) ← \(has_child(x,y))

retrieves duplicates, does too much work

Sample Queries (con't)

Find A-box individuals who *do not* have a child.

T-box:

$$
\begin{aligned}
has\_child &\sqsubseteq has\_descendant \\
inv\_has\_child &\doteq inv(has\_child) \\
has\_father &\sqsubseteq inv\_has\_child \\
has\_mother &\sqsubseteq inv\_has\_child \\
man &\sqsubseteq person \\
woman &\sqsubseteq person \\
brother &\sqsubseteq man \\
parent &\doteq person \sqcap (\exists has\_child.person) \\
mother &\doteq woman \sqcap parent \\
grandmother &\doteq mother \sqcap \\
&\quad \exists has\_child.\exists has\_child.person
\end{aligned}
$$

Role Declarations:

$$
\begin{aligned}
&transitive(has\_descendant) \\
&attribute(age, integer) \\
&feature(has\_father) \\
&feature(has\_mother)
\end{aligned}
$$

A-box:

$$
\begin{aligned}
&woman(alice), \quad woman(betty), \quad brother(charles), \\
&(\leq 1 has\_sibling)(charles), \quad has\_sister(eve, doris), \\
&has\_child(alice, betty), \quad has\_child(alice, charles), \quad has\_child(betty, doris), \\
&has\_child(betty, eve), \quad has\_sibling(charles, betty), \quad has\_sister(doris, eve)
\end{aligned}
$$

ans(x) ← \(has_known_successor(x,has_child))

{(charles)(doris)(eve)}

Conclusions

- nRQL is an improvement, but needs improvement
- Taking advantage of complete DL with a query language is an incremental task
- Benchmarking impressive, but more important is the coverage and functionality
- Should learn from present advances in querying relational databases to improve efficiency, robustness, etc.