Information Integration on the WEB with RDF, OWL and SPARQL

Review Material

Description Logics: dialect $\mathcal{SROIQ}(\mathbf{D})$

Grant Weddell

November 14, 2013

Syntax of DLs
Concepts and Roles in $\mathcal{ALC}$

Almost all DLs are a fragment of FOL with a signature S consisting of constant function symbols and predicate symbols that are unary or binary.

A *signature* S in a given DL is a (possibly countably infinite) selection of non-logical parameters:

- unary predicate symbols in S^P called *primitive concepts*,
- binary predicate symbols in S^P called *primitive roles* and
- constants in S^F called *individuals*.

A particular dialect allows more general concepts C and roles S to be expressed.

The *core* DL dialect is called $\mathcal{ALC}$, short for *attributive logic with compliment*. The concepts $\{C_i\}$ and roles $\{S_i\}$ induced by S for $\mathcal{ALC}$ are given by the following grammars, where A are primitive concepts and R are primitive roles.

$$\begin{array}{rcll} C & ::= & A & \\ & | & \exists S.C & (\textit{existential restriction}) \\ & | & C \sqcap C & (\textit{concept intersection}) \\ & | & \neg C & (\textit{concept complement}) \end{array} \qquad S ::= R$$

Syntax of DLs (cont'd)
Additional Concepts Constructors in $\mathcal{ALC}$

Symbols "$\exists$", "$\sqcap$", and "$\neg$" are called *concept constructors*.

Additional concept constructors, "$\bot$", "$\top$", "$\sqcup$" and "$\forall$", can also be used to formulate concepts in $\mathcal{ALC}$.

$$
\begin{array}{rcll}
C & ::= & \bot & (\textit{bottom}) \\
 & | & \top & (\textit{top}) \\
 & | & C \sqcup C & (\textit{concept disjunction}) \\
 & | & \forall R.C & (\textit{universal restriction})
\end{array}
$$

In particular: Such concepts occurring in a given knowledge base $\mathcal{K}$ can be replaced as follows, where A is an arbitrary primitive concept:

$$
\begin{array}{rcl}
\bot & \leadsto & A \sqcap \neg A \\
\top & \leadsto & \neg\bot \\
C_1 \sqcup C_2 & \leadsto & \neg(\neg C_1 \sqcap \neg C_2) \\
\forall R.C & \leadsto & \neg\exists R.\neg C
\end{array}
$$

Syntax of DLs (cont'd)
Knowledge Bases in $\mathcal{ALC}$

An ontology or *knowledge base* $\mathcal{K}$ in $\mathcal{ALC}$ consists of the following:

- a set of sentences or *constraints* $\mathcal{T}$ called a TBox (short for *terminology*), and
- a set of sentences or constraints $\mathcal{A}$ called an ABox (short for *assertion box*).

The constraints that can appear in $\mathcal{K}$ are given by the following grammar, where a and b are constants in S^{F}.

$$
\begin{array}{lcll}
\mathcal{C} & ::= & C \sqsubseteq C & (\textit{general concept inclusion}, \text{ or GCI}) \\
& | & C(a) & (\textit{concept assertion}) \\
& | & R(a, b) & (\textit{role assertion}) \\
& | & a = b & (\textit{individual equality})
\end{array}
$$

The TBox $\mathcal{T}$ of $\mathcal{K}$ consists of all GCIs.

The ABox $\mathcal{A}$ of $\mathcal{K}$ are the remaining constraints $\mathcal{K} \setminus \mathcal{T}$.

Syntax of DLs (cont'd)
Additional Constraints in $\mathcal{ALC}$

Additional TBox and ABox constraints can also be formulated in $\mathcal{ALC}$.

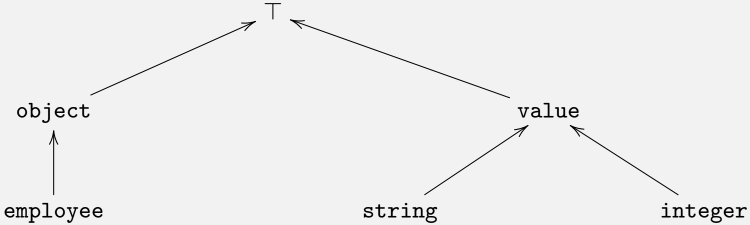
$$\begin{array}{lcll} \mathcal{C} & ::= & C \equiv C & (\textit{concept equivalence}) \\ & | & A \doteq C & (\textit{atomic concept definition}) \\ & | & \neg R(a,b) & (\textit{negated role assertion}) \\ & | & a \neq b & (\textit{individual inequality}) \end{array}$$

In particular: Subsets of a given knowledge base $\mathcal{K}$ consisting of these constraints can be replaced as follows, where A_1 and A_2 are fresh primitive concepts (i.e., do not appear in $\mathcal{K}$):

$$\begin{array}{rcl} \{C_1 \equiv C_2\} & \rightsquigarrow & \{C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1\} \\ \{A \doteq C\} & \rightsquigarrow & \{A \equiv C\} \\ \{\neg R(a,b)\} & \rightsquigarrow & \{A_1(a), A_2(b), A_1 \sqsubseteq \forall R.\neg A_2\} \\ \{a \neq b\} & \rightsquigarrow & \{A_1(a), A_2(b), A_1 \sqsubseteq \neg A_2\} \end{array}$$

ACME PAYROLL System
The Signature

(RECALL OPTION 3)



$$S^P \quad = \quad \{\texttt{employee}/1, \texttt{object}/1, \texttt{value}/1, \mathsf{integer}/1, \texttt{string}/1\}$$
$$\cup \; \{\texttt{emp-num}/2, \texttt{name}/2, \texttt{salary}/2\}$$

$$S^F \quad = \quad \{\texttt{mary}/0, \texttt{john}/0\}$$

ACME PAYROLL System (cont'd)
The Knowledge Base

The *data* and *metadata* correspond to an ABox and a TBox, respectively.

ABox (example PAYROLL data)

$\mathtt{employee(mary)}$	Mary is an employee.
$\mathtt{name(mary, ``Mary")}$	Mary's name is "Mary".
$\mathtt{salary(mary}, 72000)$	Mary's salary is 72000.
$\mathtt{employee(john)}$	John is an employee.
$\mathtt{emp\text{-}num(john}, 3412)$	John's employee number is 3412.
$\mathtt{mary} \neq \mathtt{john}$	Mary and John are different things.

TBox (PAYROLL metadata)

$$\begin{aligned} \mathtt{employee} \quad \equiv \quad & (\exists\, \mathtt{name} \,.\, \top) \\ & \sqcap (\exists\, \mathtt{emp\text{-}num} \,.\, \top) \\ & \sqcap (\exists\, \mathtt{salary} \,.\, \top) \end{aligned}$$

Employees are things that have a name, employee number and salary.

ACME PAYROLL System
The Knowledge Base (cont'd)

TBox (cont'd)

$$\begin{aligned}
\mathtt{employee} &\sqsubseteq \mathtt{object} \\
&\quad \sqcap (\forall\, \mathtt{name}\,.\,\mathtt{string}) \\
&\quad \sqcap (\forall\, \mathtt{emp\text{-}num}\,.\,\mathtt{integer}) \\
&\quad \sqcap (\forall\, \mathtt{salary}\,.\,\mathtt{integer}) \\
(\mathtt{object} \sqcap \mathtt{value}) &\sqsubseteq \bot \\
\mathtt{value} &\equiv (\mathtt{string} \sqcup \mathtt{integer}) \\
(\mathtt{string} \sqcap \mathtt{integer}) &\sqsubseteq \bot
\end{aligned}$$

What cannot be expressed:

- functional roles,
- keys and
- functional dependencies.

Semantics is based on the FOL notion of an interpretation due to Tarski.

An *interpretation* $\mathcal{I}$ of a $\mathcal{ALC}$ terminology $\mathcal{T}$ is a 2-tuple $(\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\triangle^{\mathcal{I}}$ is a non-empty (possibly infinite) domain of objects or things, and
- $\cdot^{\mathcal{I}}$ is an interpretation function for concepts, roles and individuals.

The interpretation function maps primitive concepts to subsets of $\triangle^{\mathcal{I}}$, primitive roles to subsets of $(\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}})$ and individuals to elements of $\triangle^{\mathcal{I}}$.

The function is extended to (arbitrary) $\mathcal{ALC}$ concepts as follows:

$$\begin{array}{rcl} (\exists R.C)^{\mathcal{I}} & = & \{e_1 \in \triangle^{\mathcal{I}} \mid \exists e_2 \in \triangle^{\mathcal{I}} \text{ s.t. } (e_1, e_2) \in R^{\mathcal{I}} \text{ and } e_2 \in C^{\mathcal{I}}\} \\ (C_1 \sqcap C_2)^{\mathcal{I}} & = & (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} & = & \triangle^{\mathcal{I}} \setminus C^{\mathcal{I}} \end{array}$$

An interpretation $\mathcal{I}$ is a *model* for a general concept inclusion (GCI) $C_1 \sqsubseteq C_2$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$, written $\mathcal{I} \models C_1 \sqsubseteq C_2$.

$\mathcal{I}$ is a model for a terminology $\mathcal{T}$ if it is a model for each GCI in $\mathcal{T}$, also written $\mathcal{I} \models \mathcal{T}$.

Semantics of DLs
Assertion Boxes via Nominals

Adding $\mathcal{O}$ to the name of a DL dialect indicates the inclusion of the *nominal concept constructor* "$\{\,\}$":

$$C \quad ::= \quad \{a\} \quad (\textit{nominal})$$

An interpretation function $\cdot^{\mathcal{I}}$ is extended to nominals as follows:

$$(\{a\})^{\mathcal{I}} = \{a^{\mathcal{I}}\}$$

An ABox constraint is then syntactic shorthand for a GCI in $\mathcal{ALCO}$:

$$\begin{array}{rcl} C(a) & \rightsquigarrow & \{a\} \sqsubseteq C \\ R(a,b) & \rightsquigarrow & \{a\} \sqsubseteq \exists R.\{b\} \\ \neg R(a,b) & \rightsquigarrow & \{a\} \sqsubseteq \neg\exists R.\{b\} \\ a = b & \rightsquigarrow & \{a\} \sqsubseteq \{b\} \\ a \neq b & \rightsquigarrow & (\{a\} \sqcap \{b\}) \sqsubseteq \bot \end{array}$$

Observation: An $\mathcal{ALC}$ knowledge base is an $\mathcal{ALCO}$ terminology in which nominal concepts have restricted use.

Subsumption Checking

Let $\mathcal{K}$ be an arbitrary knowledge base.

The *concept subsumption problem* for $\mathcal{K}$ is written

$$\mathcal{K} \models C_1 \sqsubseteq C_2$$

and is to determine if $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models C_1 \sqsubseteq C_2$ for all $\mathcal{I}$.

Special cases:

- *Role checking*: Given $R(a, b)$, to determine if $\mathcal{K} \models \{a\} \sqsubseteq \exists R.\{b\}$.
- *Instance checking*: Given $C(a)$, to determine if $\mathcal{K} \models \{a\} \sqsubseteq C$.
- *Knowledge base consistency*: To determine if $\mathcal{K} \not\models \{a\} \sqsubseteq \bot$, for some a.
- *Concept consistency*: Given C, to determine if $\mathcal{K} \not\models C \sqsubseteq \bot$.

Subsumption checking can be reduced to instance checking:

$$\mathcal{K} \models C_1 \sqsubseteq C_2 \quad \text{iff} \quad \mathcal{K} \cup \{(C_1 \sqcap \neg C_2)(a)\} \models \{a\} \sqsubseteq \bot,$$

for some a not occurring in $\mathcal{K}$.

Conjunctive Queries

A *conjunctive query* Q is a wff of the form

$$\exists\{x_i\}.\left(\bigwedge C_i(y_i)\right)\wedge\left(\bigwedge S_i(z_i,w_i)\right).$$

The $\{x_i\}$ are called the *non-distinguished* variables of Q.

The *distinguished* variables of Q are its free variables: $\mathsf{Fv}(Q)$.

A *substitution* θ is a mapping from a set of variables to individuals.

The *certain answers* of Q over a knowledge base $\mathcal{K}$ is a set of substitutions:

$$\{\theta \text{ over } \mathsf{Fv}(Q) \mid \mathcal{K}\models Q\theta\}.$$

Special cases:

- *Role assertion query*: Q has the form $S(x,y)$.
- *Instance query*: Q has the form $C(x)$.

Evaluating Conjunctive Queries

When a knowledge base is loaded, a reasoning engine usually starts by performing a *classification*.

A *classification* of a knowledge base $\mathcal{K}$ is a directed graph $G = (V, E)$, where:

- V is the set of atomic concepts and individuals, viewed as nominals, occuring in $\mathcal{K}$, and
- E consists of all edges $C \rightarrow D$ for which $\mathcal{K} \models C \sqsubseteq D$.

Performing a classification achieves a number of things:

- a check for knowledge base consistency,
- an efficient way of evaluating an instance query of the form $\top(x)$,
- an efficient way of evaluating role assertion queries of the form $S(x, y)$, and
- an efficient way of evaluating instance queries of the form $A(a)$.

Also: All reasoning engines will provide support for reasoning tasks that are instance checks, that is, determining if $\mathcal{K} \models C(a)$.

Evaluating Conjunctive Queries (Cont'd)

Class exercise: Consider the following algebra for evaluating conjunctive queries with no non-distinguished variables:

$$
\begin{array}{rcll}
E & ::= & \top(x) & \text{(all individuals in } \mathcal{K}\text{)} \\
 & | & R(x, y) & \text{(role assertion query)} \\
 & | & A(x) & \text{(instance query)} \\
 & | & \sigma_{C(x)}(E) & \text{(instance check)} \\
 & | & \pi_{\{x_1, \ldots, x_n\}}(E) & \text{(duplicate preserving projection)} \\
 & | & E_1 \overset{\rightarrow}{\bowtie} E_2 & \text{(nested loop join)}
\end{array}
$$

Explore how plans in this algebra can be used to evaluate SPARQL queries over an RDF source based on a hypothetical $\mathcal{ALC}$ entailment regime.

- How does one handle "second order" variables in basic graph patterns that bind to URIs denoting classes and properties?
- When such variables do not occur, is there always an initial plan that can easily be found?
- Can you think of rewriting rules that can improve efficiency? (Assume the rules themselves can require reasoning tasks.)

At-Least Restrictions

Adding $\mathcal{Q}$ to the name of a DL dialect indicates the inclusion of the *at-least concept constructor* "$\geqslant n$", a more general form of existential restriction:

$$C \quad ::= \quad \geqslant n\, S.C \quad (\textit{at-least restriction})$$

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ is extended to at-least restrictions as follows:

$$(\geqslant n\, S.C)^{\mathcal{I}} \;=\; \{e_1 \in \triangle^{\mathcal{I}} \mid n \leq \sharp\{e_2 \in \triangle^{\mathcal{I}} \text{ s.t. } (e_1, e_2) \in S^{\mathcal{I}} \text{ and } e_2 \in C^{\mathcal{I}}\}\}$$

There are two special cases:

- Adding $\mathcal{N}$ to a DL dialect indicates the inclusion of at-least restrictions of the form "$\geqslant n\, S.\top$".
- Adding $\mathcal{F}$ to a DL dialect indicates the inclusion of at-least restrictions of the form "$\neg(\geqslant 2\, S.\top)$" (denoting a set of objects for which S is a partial function).

Also:

- Allow concepts "$\leqslant n\, S.C$" as shorthand for "$\neg(\geqslant n+1\, S.C)$".
- Allow constraints "Func(S)" as shorthand for "$\top \sqsubseteq \leqslant 1\, S.\top$".

Inverse Roles

A number of dialects allow roles to be inverted:

$$S \quad ::= \quad S^{-} \quad (\textit{role inverse})$$

An interpretation function $\cdot^{\mathcal{I}}$ is extended to role inverse as follows:

$$(S^{-})^{\mathcal{I}} \;=\; \{(e2, e1) \in (\triangle \times \triangle) \mid (e_1, e_2) \in S^{\mathcal{I}}\}$$

Adding $\mathcal{I}$ to the name of a DL dialect indicates the inclusion of the *inverse role constructor* "$^{-}$".

Observation: $((S^{-})^{-})^{\mathcal{I}} = S^{\mathcal{I}}$, for any interpretation $\mathcal{I}$. Therefore assume S^{-} is short for a role of the form "R" or "R^{-}".

Role Boxes

Adding $\mathcal{H}$ to the name of a DL dialect indicates the ability to include *role inclusion constraints* in a knowledge base $\mathcal{K}$.

$$\mathcal{C} \quad ::= \quad S \sqsubseteq S \quad (\textit{role inclusion})$$

An interpretation $\mathcal{I}$ is a model for a role inclusion $S_1 \sqsubseteq S_2$ if $(S_1)^{\mathcal{I}} \subseteq (S_2)^{\mathcal{I}}$, written $\mathcal{I} \models S_1 \sqsubseteq S_2$.

The *role subsumption problem* for $\mathcal{K}$ is written $\mathcal{K} \models S_1 \sqsubseteq S_2$ and is to determine if $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models S_1 \sqsubseteq S_2$ for all $\mathcal{I}$.

The subset of $\mathcal{K}$ consisting of all role inclusions is called the *RBox* $\mathcal{R}$ of $\mathcal{K}$.

Role Composition and Transitivity

A number of dialects allow roles to be composed:

$$S \quad ::= \quad S \circ S \quad (\textit{role composition})$$

An interpretation function $\cdot^{\mathcal{I}}$ is extended to role composition as follows:

$$(S_1 \circ S_2)^{\mathcal{I}} \;=\; \{(e_1, e_3) \in (\triangle \times \triangle) \mid (e_1, e_2) \in S_1^{\mathcal{I}} \text{ and } (e_2, e_3) \in S_2^{\mathcal{I}}\}$$

A dialect that supports *transitive roles* allows an RBox $\mathcal{R}$ to have role inclusions of the form

$$C \quad ::= \quad R \circ R \sqsubseteq R \quad (\textit{transitive role})$$

Such constraints ensure that R is transitive in any interpretation of a knowledge base.

$\mathcal{ALC}$ with support for transitive roles is more simply referred to as dialect $\mathcal{S}$.

Note: There are now conditions on how roles may appear in concepts to ensure decidability of concept subsumption. (Return to this later.)

Role Composition (cont'd)

Adding $\mathcal{R}$ to the name of a DL dialect has a number of consequences:

- Implies adding $\mathcal{H}$.
- Allows a single use of composition on the left-hand-side of role inclusions:

$$\mathcal{C} \quad ::= \quad S \circ S \sqsubseteq S \quad (\textit{role composition})$$

- Allows roles to have additional properties:

$$\begin{array}{lcll} \mathcal{C} & ::= & \mathsf{Ref}(S) & (\textit{role reflexivity}) \\ & \mid & \mathsf{Asy}(S) & (\textit{role asymmetry}) \\ & \mid & \mathsf{Dis}(S_1, S_2) & (\textit{role disjointness}) \end{array}$$

- Admits a new concept constructor:

$$C \quad ::= \quad \exists S.\mathsf{Self} \quad (\textit{self restriction})$$

Class exercise: Define the semantics of the additional properties and of self restriction concepts.

Decidability with an RBox

For subsumption to be decidable, any dialect that admits role composition must restrict how roles may be used in a knowledge base $\mathcal{K}$.

The set NS of *non-simple basic roles* of knowledge base $\mathcal{K}$ is the smallest set satisfying the following conditions:

1. $S_1 \circ S_2 \sqsubseteq S_3 \in \mathcal{R}$ implies $\{S_1, S_2\} \subseteq \mathrm{NS}$;
2. $S \in \mathrm{NS}$ implies $S^- \in \mathrm{NS}$; and
3. $S_1 \sqsubseteq S_2 \in \mathcal{R}$ and $S_1 \in \mathrm{NS}$ implies $S_2 \in \mathrm{NS}$.

Subsumption checking is decidable for $\mathcal{K}$ if the following conditions are satisfied:

1. Any role occurring in an asymmetry constraint, in a role disjointness constraint or in an at-least restriction does not occur in NS.
2. The RBox of $\mathcal{K}$ is *regular*: there exists a total order $\prec$ over all roles such that, for any $S_1 \circ S_2 \sqsubseteq S_3 \in \mathcal{K}$, $S_1 \prec S_3$ and $S_2 \prec S_3$.

Concrete Domains

Adding (**D**) to the name of a DL dialect indicates the inclusion of (one or more) *concrete domains*, which has a number of consequences:

1. The addition of a (possibly infinite) set of *literal constants* $\triangle_{\mathrm{lit}}$ to the domain $\triangle$ of any interpretation $\mathcal{I}$ together with the addition of unary function symbols $\{f_i\}$ to S^{F} called *concrete features*. An interpretation function $\cdot^{\mathcal{I}}$ maps a concrete feature f to a function: $\triangle \rightarrow \triangle_{\mathrm{lit}}$.
2. The addition of a (possibly infinite) set of *interpreted* predicate symbols $\{P_i/n_i\}$ over respective $(\triangle_{\mathrm{lit}})^{n_i}$.
3. The addition of a new kind of concept:

$$C \quad ::= \quad P/n\,(f_1, \ldots, f_n) \quad (\textit{concrete domain})$$

An interpretation function $\cdot^{\mathcal{I}}$ is extended to concrete domain concepts as follows:

$$(P/n\,(f_1, \ldots, f_n))^{\mathcal{I}} \;=\; \{e \in \triangle \mid ((f_1)^{\mathcal{I}}(e), \ldots, (f_n)^{\mathcal{I}}(e)) \in P^{\mathcal{I}}\}$$

Concrete Domain $\mathbb{S}$

The concrete domain $\mathbb{S}$ of finite length strings is given as follows:

1. The literal constants $\triangle_{\mathsf{lit}}$ are finite strings $\{s_1, s_2, \ldots\}$.
2. Additional predicate symbols include: unary predicates over $\triangle_{\mathsf{lit}}$ corresponding to literals, $\{=s_1/1, =s_2/1, \ldots\}$, and Cmp/2, the strict lexicographic ordering over $\triangle_{\mathsf{lit}} \times \triangle_{\mathsf{lit}}$.

Examples:

- Mary's name is "Mary":

$$\{\mathtt{mary}\} \sqsubseteq \; = \text{"Mary"}(\mathtt{name})$$

- Mary's salary is less than one hundred thousand:

$$(\mathrm{Cmp}(\mathtt{salary}, \mathtt{oht}) \sqcap = \text{"100000"}(\mathtt{oht}))(\mathtt{mary})$$

Summary

A $\mathcal{SROIQ}(\mathbf{D})$ knowledge base is a $\mathcal{SROIQ}(\mathbf{D})$ terminology.

A $\mathcal{SROIQ}(\mathbf{D})$ terminology is a fragment of FOL with an initial signature S consisting of:

- constant function symbols $\{a_i\}$, called individuals,
- unary predicate symbols $\{A_i\}$, called primitive concepts,
- binary predicate symbols $\{R_i\}$, called primitive roles,
- unary function symbols $\{f_i\}$, called concrete features

and one or more initial theories called concrete domains.

The set of $\mathcal{SROIQ}(\mathbf{D})$ roles $\{S_i\}$ induced by S is given by the following grammar:

$$\begin{array}{rcll} S & ::= & R & \\ & \mid & S^{-} & (\textit{role inverse}) \end{array}$$

The set of $\mathcal{SROIQ}(\mathbf{D})$ concepts $\{C_i\}$ induced by S is given by the following grammar:

$$
\begin{array}{rcll}
C & ::= & A & \\
 & | & \exists S.C & (\textit{existential restriction}) \\
 & | & C \sqcap C & (\textit{concept intersection}) \\
 & | & \neg C & (\textit{concept complement}) \\
 & | & \top & (\textit{top}) \\
 & | & \bot & (\textit{bottom}) \\
 & | & C \sqcup C & (\textit{concept disjunction}) \\
 & | & \forall R.C & (\textit{universal restriction}) \\
 & | & \geqslant n\, S.C & (\textit{qualified at-least restriction}) \\
 & | & \leqslant n\, S.C & (\textit{qualified at-most restriction}) \\
 & | & \geqslant n\, S & (\textit{at-least restriction}) \\
 & | & \leqslant n\, S & (\textit{at-most restriction}) \\
 & | & \exists S.\mathrm{Self} & (\textit{self restriction}) \\
 & | & P(f_1, \ldots, f_n) & (\textit{concrete domain})
\end{array}
$$

Summary (cont'd)

The set of $\mathcal{SROIQ}(\mathbf{D})$ constraints $\{\mathcal{C}_i\}$ induced by S is given by the following grammar:

$$
\begin{array}{rcll}
\mathcal{C} & ::= & C \sqsubseteq C & (\textit{general concept inclusion}, \text{ or GCI}) \\
 & | & C(a) & (\textit{concept assertion}) \\
 & | & R(a, b) & (\textit{role assertion}) \\
 & | & a = b & (\textit{individual equality}) \\
 & | & C \equiv C & (\textit{concept equivalence}) \\
 & | & A \doteq C & (\textit{atomic concept definition}) \\
 & | & \neg R(a, b) & (\textit{negated role assertion}) \\
 & | & a \neq b & (\textit{individual inequality}) \\
 & | & \mathrm{Func}(S) & (\textit{role functionality}) \\
 & | & S \sqsubseteq S & (\textit{role inclusion}) \\
 & | & S \circ S \sqsubseteq S & (\textit{role composition}) \\
 & | & \mathrm{Ref}(S) & (\textit{role reflexivity}) \\
 & | & \mathrm{Asy}(S) & (\textit{role asymmetry}) \\
 & | & \mathrm{Dis}(S_1, S_2) & (\textit{role disjointness})
\end{array}
$$

Relation Algebra

The *relation algebra* is a fragment of FOL with roles and constraints given by the following respective grammars.

$$
\begin{array}{lcll}
S & ::= & R & \\
& | & S^{-} & (\textit{role inverse}) \\
& | & S \circ S & (\textit{role composition}) \\
& | & S \sqcap S & (\textit{role intersection}) \\
& | & \neg S & (\textit{role compliment}) \\
& | & \approx & (\textit{identity}) \\
& & & \\
\mathcal{C} & ::= & S \sqsubseteq S & (\textit{role inclusion})
\end{array}
$$

Observation: Rich enough to express Peano arithmetic and axiomatic set theories!

Longer term exercise: Explore if a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base can be translated to a theory in relation algebra.