I.5.2  \( QL \to OWL \) and \( DL-Lite_R \to SROIQ(D) \)

\( QL \to OWL \) is the sublanguage inclusion obtained by the syntactic restriction according to the definition of \( QL \), see \( NR6 \).

Since by definition, \( DL-Lite_R \) is a syntactic restriction of \( SROIQ(D) \), \( DL-Lite_R \to SROIQ(D) \) is the corresponding sublogic inclusion.

I.5.3  \( RL \to OWL \) and \( RL \to SROIQ(D) \)

\( RL \to OWL \) is the sublanguage inclusion obtained by the syntactic restriction according to the definition of \( RL \), see \( NR6 \).

Since by definition, \( RL \) is a syntactic restriction of \( SROIQ(D) \), \( RL \to SROIQ(D) \) is the corresponding sublogic inclusion.

I.5.4  \( SimpleRDF \to RDF \)

\( SimpleRDF \to RDF \) is an obvious inclusion, except that \( SimpleRDF \) resources need to be renamed if they happen to have a predefined meaning in \( RDF \). The model translation needs to forget the fixed parts of \( RDF \) models. Since this part can always reconstructed in a unique way, the result is an isomorphic model translation.

I.5.5  \( RDF \to RDFS \)

This is entirely analogous to \( SimpleRDF \to RDF \).

I.5.6  \( SimpleRDF \to SROIQ(D) \)

A \( SimpleRDF \) signature is translated to \( SROIQ(D) \) by providing a class \( P \) and three roles \( sub, pred \) and \( obj \) (these reify the extension relation), and one individual per \( SimpleRDF \) resource. A \( SimpleRDF \) triple \((s, p, o)\) is translated to the \( SROIQ(D) \) sentence

\[
\top \sqsubseteq \exists u. (\exists sub. \{s\} \cap \exists pred. \{p\} \cap \exists obj. \{o\}).
\]

From an \( SROIQ(D) \) model \( I \), obtain a \( SimpleRDF \) model by inheriting the universe and the interpretation of individuals (then turned into resources). The interpretation \( P_I^D \) of \( P \) gives \( P_m \), and \( EXT_m \) is obtained by de-reifying, i.e.

\[
EXT_m(x) := \{(y, z) \mid \exists u. (u, x) \in pred^I, (u, y) \in sub^I, (u, z) \in obj^I\}.
\]

\( RDF \to SROIQ(D) \) is defined similarly. The theory of \( RDF \) built-ins is (after translation to \( SROIQ(D) \)) added to any signature translation. This ensures that the model translation can add the built-ins.

I.5.7  \( OWL \to FOL \)

I.5.7.1 Translation of signatures

\( \Phi((C, R, I)) = (F, P) \) with

- function symbols: \( F = \{a^{(1)} \mid a \in I\} \)
- predicate symbols \( P = \{A^{(1)} \mid A \in C\} \cup \{R^{(2)} \mid R \in R\} \)

I.5.7.2 Translation of sentences

Concepts are translated as follows:
— $\alpha_s(A) = A(x)$
— $\alpha_s(\top) = true$
— $\alpha_s(\bot) = false$
— $\alpha_s(\neg C) = \neg \alpha_s(C)$
— $\alpha_s(C \cap D) = \alpha_s(C) \land \alpha_s(D)$
— $\alpha_s(C \cup D) = \alpha_s(C) \lor \alpha_s(D)$
— $\alpha_s(\exists R.C) = \exists y.(R(x, y) \land \alpha_y(C))$
— $\alpha_s(\forall U.C) = \forall y.a_y(C)$
— $\alpha_s(\exists R.C) = \exists y.(R(x, y) \rightarrow \alpha_y(C))$
— $\alpha_s(\forall U.C) = \forall y.a_y(C)$
— $\alpha_s(\exists R.Sell) = R(x, x)$
— $\alpha_s(\leq n R.C) = \forall y_1, \ldots, y_{n+1}. \land_{i=1}^{n+1}(R(x, y_i) \land \alpha_{y_i}(C)) \rightarrow \lor_{1 \leq i \leq n+1} y_i = y_j$
— $\alpha_s(\geq n R.C) = \exists y_1, \ldots, y_n. \land_{i=1}^{n}(R(x, y_i) \land \alpha_{y_i}(C)) \land \land_{1 \leq i \leq n} y_i \neq y_j$
— $\alpha_s(\{a_1, \ldots, a_n\}) = (x = a_1 \lor \ldots \lor x = a_n)$

For inverse roles $R^\rightarrow$, $R^\rightarrow (x, y)$ has to be replaced by $R(y, x)$, e.g.

$$\alpha_s(\exists R^\rightarrow .C) = \exists y.(R(y, x) \land \alpha_y(C))$$

This rule also applies below.

Sentences are translated as follows:

— $\alpha_s(\subseteq D) = \forall x. (\alpha_s(C) \rightarrow \alpha_s(D))$
— $\alpha_s(a : C) = \alpha_s(C)[x \rightarrow a]$\footnote{[x \rightarrow a] means “in t, replace x by a”}
— $\alpha_s(R(a, b)) = R(a, b)$
— $\alpha_s(R \subseteq S) = \forall x, y. R(x, y) \rightarrow S(x, y)$
— $\alpha_s(R_1; \ldots; R_n \subseteq R) = \forall x, y. (\exists z_1, \ldots, z_{n-1}. R_1(x, z_1) \land R_2(z_1, z_2) \land \ldots \land R_n(z_{n-1}, y)) \rightarrow R(x, y)$
— $\alpha_s(\text{Dis}(R_1, R_2)) = \neg \exists x, y. R_1(x, y) \land R_2(x, y)$
— $\alpha_s(\text{Ref}(R)) = \forall x. R(x, x)$
— $\alpha_s(\text{Irr}(R)) = \forall x, \neg R(x, x)$
— $\alpha_s(\text{Asy}(R)) = \forall x, y. R(x, y) \rightarrow \neg R(y, x)$
— $\alpha_s(\text{Tra}(R)) = \forall x, y, z. R(x, y) \land R(y, z) \rightarrow R(x, z)$

I.5.7.3 Translation of models

— For $M' \in \text{Mod}^{\text{FOL}}(\Phi \Sigma)$ define $\beta_\alpha(M') = (\Delta, \tilde{I}, \tilde{\alpha}, \tilde{\beta}, \tilde{\gamma})$ with $\Delta = |M'|$ and

$$\begin{align*}
\Delta' & = M'_{\alpha}, a'_1 = M'_{\alpha}(a_1), a'_2 = M'_{\alpha}(a_2), R' = M_{\alpha}(R)
\end{align*}$$

Proposition 24 $C^\alpha = \{m \in M'_{\alpha} | m \models \alpha_s(C)\}$ and $C^\tilde{\alpha} = \{m \in \Delta | M' + \{x \mapsto m\} \models \alpha_s(C)\}$

Proof. By Induction over the structure of $C$.

$$\begin{align*}
\Delta' & = M'_{\alpha} \quad \text{and} \quad \Delta = |M'_{\alpha}| + \{x \mapsto m\} | \models \alpha_s(C) \quad \Delta'^{\tilde{\alpha}} = \{m \in \Delta | M' + \{x \mapsto m\} \models \alpha_s(C)\}
\end{align*}$$

\footnote{[x \rightarrow a] means “in t, replace x by a”}
The satisfaction condition holds as well; other cases are similar.

The satisfaction condition now follows easily.

I.5.8  **FOL → CL**

This comorphism maps classical first-order logic (FOL) to Common Logic.

A FOL signature is translated to CL.Fol by turning all constants into discourse names, and all other function symbols and all predicate symbols into non-discourse names. A FOL sentence is translated to CL.Fol by a straightforward recursion, the base being translations of predications:

\[ \alpha_\Sigma(P(t_1, \ldots, t_n)) = (P \alpha_\Sigma(t_1) \ldots \alpha_\Sigma(t_n)) \]

Within terms, function applications are translated similarly:

\[ \alpha_\Sigma(f(t_1, \ldots, t_n)) = (f \alpha_\Sigma(t_1) \ldots \alpha_\Sigma(t_n)) \]

A CL.Fol model is translated to a FOL model by using the universe of discourse as FOL universe. The interpretation of constants is directly given by the interpretation of the corresponding names in CL.Fol. The interpretation of a predicate symbol \( P \) is given by using \( \text{rel}^M(\text{int}^M(P)) \) and restricting to the arity of \( P \); similarly for function symbols (using \( \text{fun}^M \)). Both the satisfaction condition and model-expansiveness of the comorphism are straightforward.

I.5.9  **OWL → CL**

This comorphism is the composition of the comorphisms described in the previous two sections.

I.5.10  **UML class models → CL**

This translation has been described in annex \[F\]. Translation of signatures is detailed in section \[F.4.3\] and translation of sentences in section \[F.4.5\]. Models are translated identically.

I.5.11  **FOL → CASL**

This is an obvious sublogic.

I.5.12  **UML class model to OWL**

Let \( \Sigma = ((C, \leq_C), P, O, A, M) \) be a class/data type net representing a UML class model as described in annex \[F\]. This net can be translated to OWL2 using the approach described in \[76\]. The ontology is extended by translating parts of this net and its multiplicity constraints \( \text{Mult}(\Sigma) \):

- For each class \( c \in C \) with superclasses \( c_1, c_2, \ldots, c_n \in C \) (i.e. \( c \leq_C c_i \) for \( i = 1, \ldots, n \)):

\[
\text{Class: } c \\
\text{SubClassOf: } c_1 \\
\text{SubClassOf: } c_n
\]

- For each attribute declaration \( c.p : c' \) in \( P \)