

The combined complexity of query answering in expressive DLs

Magdalena Ortiz

Institute for Information Systems, Vienna University of Technology

November 2008

KRDB Seminar, Free University of Bozen-Bolzano

Outline

- 1 Introduction and background
- 2 Two algorithms for CQ Answering in expressive DLs
 - 1 A knot-based one
 - For *ALCH*, an EXPTIME upper bound
 - For *SH*, a 2-EXPTIME upper bound

(AAAI 08/DL 08, joint work with Thomas Eiter and Mantas Šimkus)
 - 2 A domino-based one for Horn-*SHIQ*
 - An EXPTIME upper bound

(JELIA 08, joint work with Thomas Eiter, Georg Gottlob and Mantas Šimkus)

Informal discussion – what makes CQs hard?

Description Logics

Description Logics (DLs) are logics specifically tailored for Knowledge Representation.

- Most popular formalisms for writing ontologies.
- They underlie the **Ontology Web Languages (OWL)** proposed as Semantic Web standard.
- Typically, they consider a language comprising:
 - concepts**: classes, unary predicates.
 - roles**: relations between classes, binary predicates.

DL knowledge bases, an example

A DL knowledge base \mathcal{K} has two parts:

- The **terminological** knowledge is given by a set of axioms, called

TBox:

$$\begin{aligned} \textit{Man} &\sqsubseteq \textit{Human} \\ \textit{Parent} &\equiv \exists \textit{hasChild}.\textit{Human} \\ \textit{Uncle} &\equiv \exists \textit{hasSibling}.\textit{Parent} \\ \textit{hasSibling} &\equiv \textit{hasSibling}^{-} \end{aligned}$$

Men are human. A parent is someone that has a child. An uncle is someone that has a sibling who is a parent. Sibling is symmetric.

- The **assertional** knowledge is given by a set of ground facts, called

ABox:

$$\begin{array}{lll} \textit{Human}(\textit{Sam}) & \textit{Man}(\textit{John}) & \textit{Human}(\textit{Bob}) \\ \textit{hasChild}(\textit{Bob}, \textit{Sam}) & \textit{hasSibling}(\textit{Bob}, \textit{John}) & \end{array}$$

The **models** of \mathcal{K} are the FOL-interpretations that satisfy both components.

DL knowledge bases, an example

A DL knowledge base \mathcal{K} has two parts:

- The **terminological** knowledge is given by a set of axioms, called

TBox:

$$\begin{aligned} \textit{Man} &\sqsubseteq \textit{Human} \\ \textit{Parent} &\equiv \exists \textit{hasChild}.\textit{Human} \\ \textit{Uncle} &\equiv \exists \textit{hasSibling}.\textit{Parent} \\ \textit{hasSibling} &\equiv \textit{hasSibling}^{-} \end{aligned}$$

Men are human. A parent is someone that has a child. An uncle is someone that has a sibling who is a parent. Sibling is symmetric.

- The **assertional** knowledge is given by a set of ground facts, called

ABox:

$$\begin{array}{lll} \textit{Human}(\textit{Sam}) & \textit{Man}(\textit{John}) & \textit{Human}(\textit{Bob}) \\ \textit{hasChild}(\textit{Bob}, \textit{Sam}) & \textit{hasSibling}(\textit{Bob}, \textit{John}) & \end{array}$$

The **models** of \mathcal{K} are the FOL-interpretations that satisfy both components.

Expressive DLs

SHIQ is a Description Logic (underlying OWL-Lite) that:

- subsumes the basic DL *ALC*, and allows also for
- (*S*) transitive roles,
- (*H*) role hierarchies (inclusions),
- (*I*) inverse roles,
- (*Q*) qualified number restrictions.

We consider some fragments of *SHIQ*:

- *ALCH*, the ‘basic’ expressive DL *ALC* plus role hierarchies,
- *SH*, which adds transitive roles to *ALCH*, and
- Horn-*SHIQ*, the Horn (i.e., deterministic) fragment of *SHIQ*

Standard reasoning is EXPTIME-complete for all of them.

Query Answering in DLs

- DL ontologies are increasingly seen as mechanisms to describe and access **data repositories**.
E.g., in ontology-based data access, information integration, ...
- Research aiming at the use of **database query languages** to access DL ontologies.

We consider the popular **Conjunctive Queries** (CQs):

- Conjunction of atoms, variables are existentially quantified

hasUncle(x, z):-hasParent(x, y), hasSibling(y, z), Man(z)

- Equivalent to 'basic' SQL
- Popular in many other fields

Query Answering in DLs

- DL ontologies are increasingly seen as mechanisms to describe and access **data repositories**.
E.g., in ontology-based data access, information integration, ...
- Research aiming at the use of **database query languages** to access DL ontologies.

We consider the popular **Conjunctive Queries** (CQs):

- Conjunction of atoms, variables are existentially quantified

hasUncle(x, z):-hasParent(x, y), hasSibling(y, z), Man(z)

- Equivalent to 'basic' SQL
- Popular in many other fields

Conjunctive queries

The **CQ answering** problem over a KB \mathcal{K} is to decide whether there is a mapping for the query variables in every model of \mathcal{K} .

- More powerful data access than traditional DL reasoning,
individuals may be related in arbitrary ways
- and than querying standard DBs.
variables can be mapped to unnamed individuals

Many algorithms developed recently for CQ answering in DLs around *SHIQ*. We discuss two of them.

Conjunctive queries

The **CQ answering** problem over a KB \mathcal{K} is to decide whether there is a mapping for the query variables in every model of \mathcal{K} .

- More powerful data access than traditional DL reasoning,
individuals may be related in arbitrary ways
- and than querying standard DBs.
variables can be mapped to unnamed individuals

Many algorithms developed recently for CQ answering in DLs around *SHIQ*. We discuss two of them.

Conjunctive queries

The **CQ answering** problem over a KB \mathcal{K} is to decide whether there is a mapping for the query variables in every model of \mathcal{K} .

- More powerful data access than traditional DL reasoning,
individuals may be related in arbitrary ways
- and than querying standard DBs.
variables can be mapped to unnamed individuals

Many algorithms developed recently for CQ answering in DLs around *SHIQ*. We discuss two of them.

Normal Knowledge Bases and Canonical Models

An *ALCH* KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is in **normal form** if all the axioms in \mathcal{T} are of the forms:

$$\begin{array}{ll}
 (E) & A_0 \sqsubseteq \exists R.B_0, \\
 (U) & A_0 \sqsubseteq \forall R.B_0, \quad \text{or} \\
 (D) & A_0 \sqcap \dots \sqcap A_n \sqsubseteq B_0 \sqcup \dots \sqcup B_m.
 \end{array}$$

where A_i, B_j are concept names.

Canonical Models

- For query answering, we know that we only need to consider **canonical models**.
(minimal Herbrand models of the skolemized FO translation of the normalized KB)
- These models are **forest-shaped**:
 - TBoxes have **tree** shaped models,
 - but the models of ABoxes are arbitrary **graphs**.
(they can impose arbitrary relations, but only between the named individuals)

Each canonical model is composed of a **graph part** and a **set of trees**.

Knots

The trees can be represented using **knots**.

- Labeled trees of depth at most 1.
- Can be ‘instantiated’ with any domain element.
- They satisfy simple **local** conditions:
 - Propositional axioms satisfied at each node.
 - The root has the necessary successors.

They are the **pieces** that compose the **tree shaped models** of a TBox.



$$\alpha_0 = D \sqsubseteq A \sqcup B$$

$$\alpha_1 = B \sqsubseteq \exists P.A$$

$$\alpha_2 = B \sqsubseteq \exists P.C$$

$$\alpha_3 = A \sqsubseteq \exists Q.E$$

$$\alpha_4 = C \sqsubseteq \exists P.D$$

Knots and Canonical Models

To construct a model, we need:

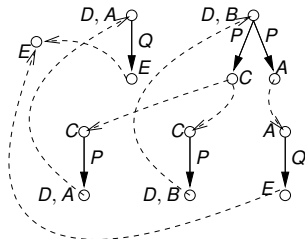
- A labeled graph that contains the ABox and where each node satisfies the propositional axioms (a **min-graph**).
- For each node, a knot that can be ‘plugged in’, i.e., whose root has the same label.

Knots and Canonical Models

$\mathcal{K}_1 = \langle \mathcal{A}_1, \mathcal{T}_1 \rangle$, $\mathcal{A}_1 = \{D(a)\}$, \mathcal{T}_1 contains:

$$\begin{array}{ll} \alpha_0 = D \sqsubseteq A \sqcup B & \alpha_3 = A \sqsubseteq \exists Q.E \\ \alpha_1 = B \sqsubseteq \exists P.A & \alpha_4 = C \sqsubseteq \exists P.D \\ \alpha_2 = B \sqsubseteq \exists P.C & \end{array}$$

We can build models for \mathcal{K}_1 from the min-graphs $\{D(a), B(a)\}$ and $\{D(a), A(a)\}$, and the knots:



Finite Representation of Models

Theorem

*There is a unique set $\mathbb{K}_{\mathcal{K}}$ of knots that, together with the min-graphs, captures all the canonical models of \mathcal{K} .
This set can be computed in single exponential time.*

This provides a (new) worst-case optimal decision procedure for KB satisfiability.

Query Answering Using Knots

We use this set of knots $\mathbb{K}_{\mathcal{K}}$ to answer a CQ q in the models of \mathcal{K} that L represents.

- We concentrate on the tree-shaped parts of the models.
- Each such part starts with a knot K from $\mathbb{K}_{\mathcal{K}}$.
- We consider subqueries ρ of q whose match can occur inside these trees.

We compute all the pairs (K, ρ) such that $K \models \rho$, i.e.,
there is a match for ρ in each tree that starts with K .

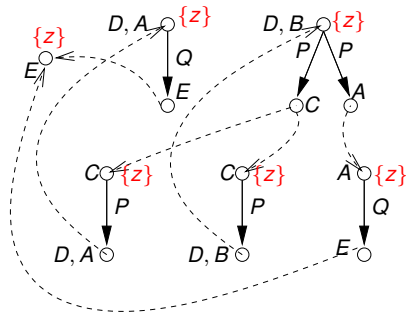
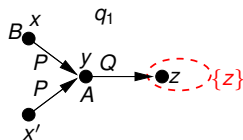
Deciding Subquery Entailment

We do this inductively, by considering the **depth** of the mappings in the trees.

- We start with mappings of depth 0.
- At each step, we compute mappings starting at K composed of:
 - mappings of smaller depth for the knots that can follow K in a tree construction, and
 - a partial embedding of q into K extending them.
- We continue until we reach a fix-point $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$.

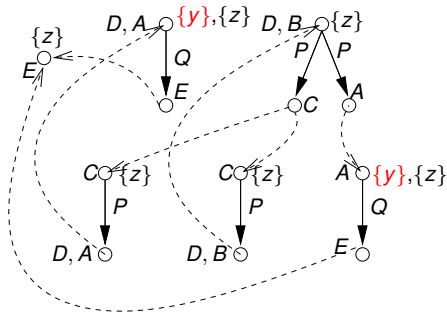
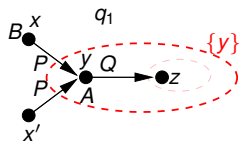
Computing the set $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$

$$K \models_{\mathbb{K}_{\mathcal{K}}}^0 \rho$$



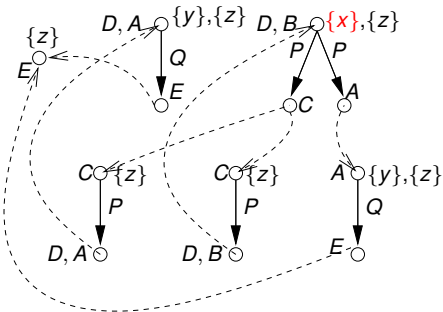
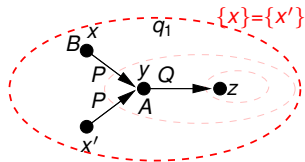
Computing the set $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$

$$K \models_{\mathbb{K}_{\mathcal{K}}}^1 \rho$$



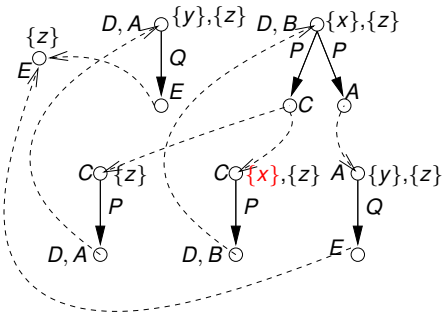
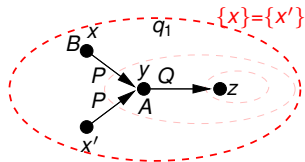
Computing the set $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$

$$K \models_{\mathbb{K}_{\mathcal{K}}}^2 \rho$$



Computing the set $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$

$$K \models_{\mathbb{K}_{\mathcal{K}}}^3 \rho$$



Evaluating the full query

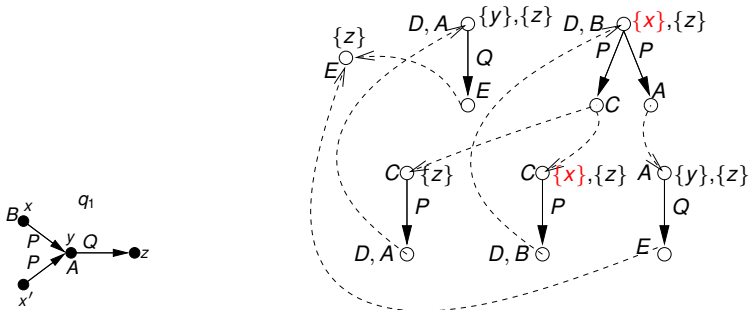
In a final step, we take the graph-part of the models into account:

- A min-graph extended with a knot that can start the tree construction for each constant is seen as a super-knot.
- Query matches can be found by computing (a generalization of) the inductive step above.

Deciding Query entailment

$\mathcal{A}_1 = \{a:D\}$, \mathcal{T}_1 contains:

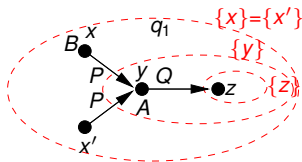
$\alpha_0 = D \sqsubseteq A \sqcup B$	$\alpha_3 = A \sqsubseteq \exists Q.E$
$\alpha_1 = B \sqsubseteq \exists P.A$	$\alpha_4 = C \sqsubseteq \exists P.D$
$\alpha_2 = B \sqsubseteq \exists P.C$	



There is a match in every model that starts with $\{D(a), B(a)\}$,
but no match in the ones starting with $\{D(a), A(a)\}$.

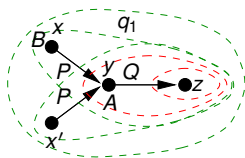
Complexity

- For *ALCH*, the algorithm has **EXPTIME** combined complexity
 - the set $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$ is polynomial in $|\mathbb{K}_{\mathcal{K}}|$ and single exponential in $|\mathcal{K}| + |q|$.
- This holds because only polynomially many subqueries must be considered:



Complexity

This would not hold, e.g., if P was transitive.



The algorithm works also for \mathcal{SH} , but it is double exponential in general.

Single exponential cases (i.e. with only polynomially many relevant subqueries) have been identified.

Other Features of our Algorithm

- It has CONP data complexity:
 - $\mathbb{K}_{\mathcal{K}}$ and $\Gamma(\mathbb{K}_{\mathcal{K}}, q)$ can be precomputed,
 - we can guess a min-graph and check in polynomial time whether it entails q .
- Provides a modular knowledge compilation technique.
- Everything can be easily encoded into a Datalog program.
- Answers non-Boolean queries.

The complexity of Query Answering

- For both *ALCH* and *SH*, the algorithm is worst-case optimal.
- For *ALCH* and *ALCHQ* [Lutz, DL'07] query answering is in **EXPTIME**. (Apparently, also for *S* and *SQ*.)
- CQs are **2EXPTIME-hard** already for:
 - ALCI* [Lutz, DL'07]
 - SH* [E.L.O.Š., 08]

How can we lower the complexity within the SHI fragment?

The complexity of Query Answering (ctd.)

- A possible answer: **disallow disjunction**.
- Horn-*SHIQ* is the Horn fragment of *SHIQ* [Hustadt et al., IJCAI'2005].
 - Consistency testing Horn-*SHIQ* is P-complete w.r.t. data complexity.
 - This is lower compared to CONP-completeness in *SHIQ*.
 - It's EXPTIME-complete in combined complexity, i.e. no easier than full *SHIQ*.
- We now discuss the complexity of **CQ answering**:
 - it is **EXPTIME-complete** w.r.t. combined complexity , and
 - **P-complete** w.r.t. data complexity.

In Horn-*SHIQ* CQ answering is **not harder than satisfiability** testing and **easier than in *SHIQ***.

The complexity of Query Answering (ctd.)

- A possible answer: **disallow disjunction**.
- Horn-*SHIQ* is the Horn fragment of *SHIQ* [Hustadt et al., IJCAI'2005].
 - Consistency testing Horn-*SHIQ* is P-complete w.r.t. data complexity.
 - This is lower compared to CONP-completeness in *SHIQ*.
 - It's EXPTIME-complete in combined complexity, i.e. no easier than full *SHIQ*.
- We now discuss the complexity of **CQ answering**:
 - it is **EXPTIME-complete** w.r.t. combined complexity , and
 - **P-complete** w.r.t. data complexity.

In Horn-*SHIQ* CQ answering is **not harder than satisfiability** testing and **easier than in *SHIQ***.

The Description Logic Horn-*SHIQ*

- We work on KBs in the following normal form:

$$\begin{array}{lll} A \sqcap B \sqsubseteq C & A \sqsubseteq \forall R.B & A \sqsubseteq \geq m S.B \\ \exists R.A \sqsubseteq B & A \sqsubseteq \exists R.B & A \sqsubseteq \leq 1 S.B \end{array}$$

- Role hierarchies and transitivity axioms are allowed.
- ABoxes are allowed.
- Only disjunctive axioms $A \sqcap B \sqsubseteq C \sqcup D$ are **not allowed**.

As usual, we only need to consider forest-shaped models.
We focus on the **tree part** for now.

Universal Models in Horn-*SHIQ*

- For DLs with disjunction, query answering inherently implies quantification over many models.
- Since Horn-*SHIQ* disallows disjunction, we can obtain *universal models*.
 - A model of \mathcal{K} is universal iff it can be homomorphically embedded into each model of \mathcal{K} .

Lemma

Existence of a query mapping in a universal model is equivalent to existence in all the models of a KB.

NOTE: There can be several universal models for a KB, but they are bisimilar.

Towards the Algorithm for CQs in Horn-*SHIQ*

- (A) We define **domino systems**.
 - Each domino system captures a possibly infinite tree-shaped interpretation.
- (B) We define a procedure for **CQs over domino systems**.
 - Allows to test if a query has a match in the tree represented by a domino system.
- (C) We give an EXPTIME tableau procedure for Horn-*SHIQ*.
 - Finite representation of a universal model.
- (D) The output of the tableau procedure is transformed into a domino system, which is then used to answer CQs over the initial KB.

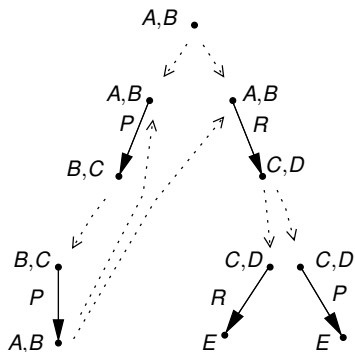
Finite representation using Domino Trees

Dominoes are similar to knots.

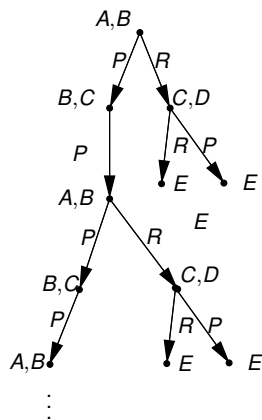
- A **domino** is a tuple $\langle c, r, c' \rangle$, where
 - c, c' are sets of concept names, and
 - r is a set of roles.
- A **domino system** is a tuple $\langle D, \triangleright, \mathcal{R} \rangle$ such that
 - D is a set of dominoes,
 - $\triangleright \subseteq D \times D$ is a **direct successor** relation with $c'_1 = c_2$ whenever $\langle c_1, r_1, c'_1 \rangle \triangleright \langle c_2, r_2, c'_2 \rangle$,
 - \mathcal{R} a set of role inclusions and transitivity axioms,
 - D contains one distinguished **initial tile** t_0 .
- Each domino system represents one (possibly infinite) tree-shaped interpretation.

Finite representation using Domino Trees: Example

Domino system \mathcal{D}



Domino tree $\mathcal{T}_{\mathcal{D}}$



Conjunctive Queries over Domino Trees

Technique: we **treeify** the query

Definition

A **query graph** q^G for a query q is a directed graph over variables of q with an edge from x to y iff $R(x, y) \in q$ for some R .

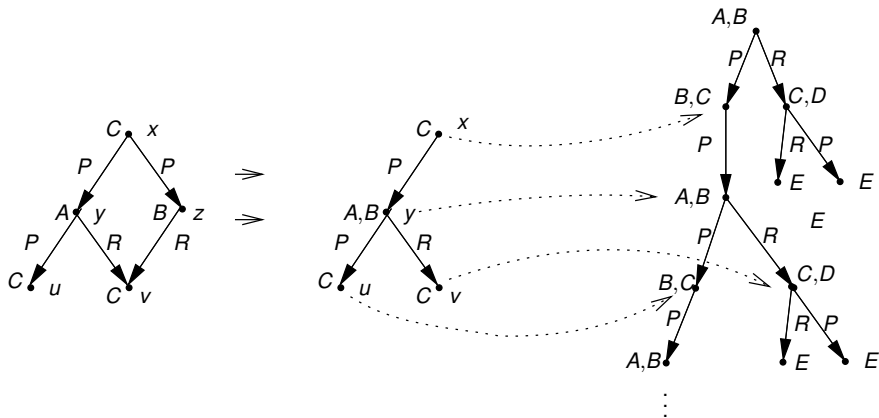
Definition

A query q is **tree-shaped** if its query graph q^G is a tree.

Lemma

For any q , we can obtain a set $T(q)$ of tree-shaped queries s.t. : $\mathcal{D} \models q$ iff $\mathcal{D} \models q'$ for some $q' \in T(q)$.

Queries over Domino Trees (Example)



Tree-Shaped queries over Domino Trees

- Existence of matches in the domino tree can be decided **without constructing it explicitly**.
- The procedure works on the underlying domino system.
- We search for a **suitable association** of each variable x of q with a domino t_x in \mathcal{D} .
- An association must witness a match:
 - the domino t_x associated with variable x must encode the concept names needed to satisfy each unary atom $A(x) \in tq$,
 - for each role atom $R(x, y) \in tq$, the domino t_x must 'reach' the domino t_y via an R -path.

Obtaining Domino Systems for Horn- \mathcal{SHIQ} KBs

- 1 We build a **tableau algorithm** for consistency testing in Horn- \mathcal{SHIQ} :
 - For a consistent KB \mathcal{K} , it returns a **complete and clash-free completion forest** that represents a model \mathcal{I} of \mathcal{K}
 - The represented model \mathcal{I} is **universal**, i.e., it suffices for query answering.
- 2 The completion forest is **decomposed into dominoes**, and the domino set is obtained.
 - The domino tree and the model \mathcal{I} correspond.
- 3 **A query over over a KB can then be answered by posing tree-shaped queries over the resulting domino tree.**

Computational Complexity

We analyze the combined complexity:

- The tableau algorithm runs in **exponential** time.
- The extracted domino system is of exponential size.
- There are exponentially many treeifications tq of the initial query q .
- For each treeification tq of q , there are exponentially many candidate domino-associations and each can be verified in exponential time.

Theorem

*CQ answering in Horn-*SHIQ* is EXPTIME-complete in **combined complexity**, i.e., in the size of the query and the knowledge base.*

Dealing with ABoxes: Data Complexity

- The method extends to arbitrary ABoxes.
- Models obtained by the tableau procedure are forest-shaped.
- They are encoded in domino systems using an artificial root:
 - ABox individuals are coded in the level 2 of domino trees.
- The possible links between individuals are taken into account by **additional query rewriting**.
- This does not alter the combined complexity, and allows to characterize the data complexity.

Theorem

*CQ answering in Horn-*SHIQ* is P-complete in **data complexity**, i.e., in the size of the ABox.*

Other Features of the Algorithm

- The algorithm and complexity results extend to **Unions of Conjunctive Queries** and **Positive Queries**.
- Domino systems enable **Knowledge Compilation**.
 - Once the domino system for a KB is obtained, it can be **used for answering all queries**.
- A suitable restriction on Horn-*SHIQ* lowers the combined complexity to PSPACE.
 - This involves removing inverse roles and existentials on the l.h.s. of axioms.

Conclusions

- A suitable representation of the models of a KB enables optimal algorithms for CQ answering.
- The combined complexity of CQ answering is, in general, provably harder than consistency testing:
 - To show non-entailment, exponentially many ‘treeifications’ of the query must be avoided at each node of each model.
- This blow-up can be avoided by:
 - Restricting the number of possible treeifications by disallowing suitable DL constructors.
 - Making the DL deterministic, so that only one model must be considered.