Mapping Data to Ontologies With Exceptions Using Answer Set Programming

Daniel P. Lupp and Evgenij Thorstensen

11th July 2016
databases are connected to an ontology using mappings of the form $\varphi \rightsquigarrow \psi$, where $\varphi$ is a database query and $\psi$ is an ontology query.
Ontology-Based Data Access

- Databases are connected to an ontology using mappings of the form $\varphi \leadsto \psi$, where $\varphi$ is a database query and $\psi$ is an ontology query.
- In order to query a database, users can phrase queries in the ontology language.
- These queries are then translated to database queries using the mappings.

![Diagram: User → Ontology → Mappings → DB → Answers]
databases are connected to an ontology using mappings of the form $\varphi \leadsto \psi$, where $\varphi$ is a database query and $\psi$ is an ontology query.

In order to query a database, users can phrase queries in the ontology language.

These queries are then translated to database queries using the mappings.

Enables the use of a domain model that closely resembles end-users’ understanding of a domain as opposed to complex and convoluted database schemas.
Formally, for $\varphi \rightsquigarrow \psi$:
Given a database instance $D$ and a set of mappings $M$, $I$ is a model of $(D, M)$ if $I \models \psi(t)$ for every query answer $t$ of $\varphi$ over $D$.

Mapping rewriting of an ontology query $\psi$ w.r.t. $M$: $\bigvee \varphi_i$ for every $i$ with $\varphi_i \rightsquigarrow \psi \in M$. 

Example:
- $\text{JOBS_DB}(x, "Accountant") \rightsquigarrow \text{Empl}(x)$
- $\text{JOBS_DB}(x, "IT") \rightsquigarrow \text{Empl}(x)$

Then $\text{Empl}(x)$ would be rewritten to $\text{JOBS_DB}(x, "IT") \lor \text{JOBS_DB}(x, "Accountant")$. 

---

Ontology-Based Data Access

User $\rightarrow$ Ontology $\rightarrow$ Mappings $\rightarrow$ DB $\rightarrow$ Answers
Formally, for $\varphi \rightsquigarrow \psi$:

Given a database instance $\mathcal{D}$ and a set of mappings $\mathcal{M}$, $\mathcal{I}$ is a model of $(\mathcal{D}, \mathcal{M})$ if $\mathcal{I} \vDash \psi(t)$ for every query answer $t$ of $\varphi$ over $\mathcal{D}$.

Mapping rewriting of an ontology query $\psi$ w.r.t. $\mathcal{M}$: $\bigvee \varphi_i$ for every $i$ with $\varphi_i \rightsquigarrow \psi \in \mathcal{M}$.

Example:

$$\text{JOBS\_DB}(x, "Accountant") \rightsquigarrow \text{Empl}(x)$$

$$\text{JOBS\_DB}(x, "IT") \rightsquigarrow \text{Empl}(x)$$

then $\text{Empl}(x)$ would be rewritten to $\text{JOBS\_DB}(x, "IT") \lor \text{JOBS\_DB}(x, "Accountant")$. 
databases typically use closed-world (CW) reasoning: if data cannot be explicitly found in the database, it is assumed to be false.

| Person | Name |
|--------|------|
| Alice  |      |
| Bob    |      |
| Carla  |      |

John is not a person, i.e., $\neg Person(John)$ is true.
Limitations - OBDA

- Databases typically use closed-world (CW) reasoning: if data cannot be explicitly found in the database, it is assumed to be false.

| Person | Name |
|--------|------|
|        | Alice|
|        | Bob  |
|        | Carla|

→ John is not a person, i.e., \( \neg \text{Person}(John) \) is true.

- Ontologies employ open-world (OW) reasoning, where, in the above example, \( \neg \text{Person}(John) \) could be either true or false.
Limitations - OBDA

- databases typically use *closed-world (CW) reasoning*: if data cannot be explicitly found in the database, it is assumed to be false.

| Person | Name   |
|--------|--------|
|        | Alice  |
|        | Bob    |
|        | Carla  |

John is not a person, i.e., \(\neg \text{Person}(John)\) is true.

- ontologies employ *open-world (OW) reasoning*, where, in the above example, \(\neg \text{Person}(John)\) could be either true or false

- mapping assertions \(\varphi \rightsquigarrow \psi\) are interpreted as first-order implications, and thus inherently open-world!
- no support for ontology constraints over the data, e.g., “every instance of Person must be in the table JOBS_DB”.

Limitations - OBDA

- No support for ontology constraints over the data, e.g., “every instance of Person must be in the table JOBS_DB”.
- Poor handling of exceptions in mappings: must be named explicitly in each mapping, since these are first-order. → difficult to maintain and prone to error!
Limitations - OBDA

- no support for ontology constraints over the data, e.g., “every instance of Person must be in the table JOBS_DB”.
- poor handling of exceptions in mappings: must be named explicitly in each mapping, since these are first-order. → difficult to maintain and prone to error!
  
  → nonmonotonicity
Nonmonotonic Extensions

- adding nonmonotonicity to OBDA and description logic ontologies is an ongoing research topic, e.g., DL-programs \([EIL^+08]\), hybrid-MKNF knowledge bases \([DNR02, MR10]\), closed predicates \([LSW13]\).
- However, the focus is on adding these capabilities to the ontologies
adding nonmonotonicity to OBDA and description logic ontologies is an ongoing research topic, e.g., DL-programs \([EIL^+08]\), hybrid-MKNF knowledge bases \([DNR02, MR10]\), closed predicates \([LSW13]\).

However, the focus is on adding these capabilities to the ontologies.

propose extending mappings instead, as they are the tool used to connect closed-world and open-world

→ *mapping programs*, an extension of \(\exists\)-ASP \([GGLS15]\)
An extension of classical ASP that supports

- existential quantification in the heads and negative bodies of rules,
- conjunctive queries in the heads and negative bodies of rules

An \( \exists \)-rule is of the form

\[
H_1, \ldots, H_n \leftarrow B_1, \ldots, B_m, \\
\text{not} \ (C^1_1, \ldots, C^1_{u_1}), \ldots, \text{not} \ (C^s_1, \ldots, C^s_{u_s}).
\]

where the \( H_i, B_j, C^l_k \) are atoms.

- all variables not occurring in the positive body are interpreted existentially.
due to the presence of existentials, only variables that are not existentials in negative bodies are grounded (*partial grounding*)

- the existential variables in the rule heads are *Skolemized*

- then the reduct and $\exists$-answer sets are defined analogously to their classical ASP counterparts
due to the presence of existentials, only variables that are not existentials in negative bodies are grounded (\textit{partial grounding})

- the existential variables in the rule heads are \textit{Skolemized}
- then the reduct and $\exists$-answer sets are defined analogously to their classical ASP counterparts

\textbf{Theorem ([GGLS15])}

\textit{For a given $\exists$-ASP program there exists an equivalent (w.r.t. answer sets) classical ASP program.}

$\rightarrow$ reasoning in $\exists$-ASP can be reduced to reasoning in ASP
Extends $\exists$-ASP to allow for ontology queries in rule bodies

- A mapping rule is of the form

$$m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k),$$
$$J_1^+(y_1'), \ldots, J_l^+(y_l'), Q^S(x).$$

where $Q^S$ is a first-order formula over the database and $H^T$, $J_i^-$, $J_j^+$ are first-order formulas over the ontology. The variables in $z$ are existential variables, and $y_i, y_j' \subseteq x$. 
Mapping Programs

Extends $\exists$-ASP to allow for ontology queries in rule bodies

- A mapping rule is of the form

$$m : H^T(x, z) \leftarrow \textbf{not} J_1^-(y_1), \ldots, \textbf{not} J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x).$$

where $Q^S$ is a first-order formula over the database and $H^T$, $J_i^-$, $J_j^+$ are first-order formulas over the ontology. The variables in $z$ are existential variables, and $y_i, y'_j \subseteq x$.

- Intuitively, this can be read as follows:

$Q^S$ is mapped to $H^T$ if all $J_j^+$ are certain answers and all $J_i^-$ are not certain answers w.r.t. the mapping and ontology.

$J^+$ and $J^-$ are called the positive and negative justifications, respectively.
Example:
Let $\mathcal{D}$ consist of just one table, $\text{JOBS}_{\text{DB}}(<\text{NAME}>,<\text{JOB}>)$, and
$\Sigma_T = \{\text{Empl}, \text{hasSup}, \text{depHeadOf}\}$ be the signature of $T$. The mapping rule

$$m_1 : \exists Z. \text{hasSup}(X, Z) \leftarrow \textbf{not} \exists Y. \text{depHeadOf}(X, Y),$$

$$\text{Empl}(X), \text{Jobs}_{\text{DB}}(X, P).$$

describes the default rule “employees, of whom we do not know that they are the head of a department, have a supervisor.”
Mapping Programs — Skolemization

For a mapping rule

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x). \]

define the *Skolem mapping rule* \( sk(m) \) by replacing each existential variable \( z \) in \( H^T(x, z) \) with a Skolem function symbol \( sk_z(s) \), where \( s \) is an ordered sequence of the universal variables \( x \).
For a mapping rule

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x). \]

define the \textit{Skolem mapping rule} \( sk(m) \) by replacing each existential variable \( z \) in \( H^T(x, z) \) with a Skolem function symbol \( sk_z(s) \), where \( s \) is an ordered sequence of the universal variables \( x \).

**Definition (Skolem program [GGLS15])**

For a mapping program \( M \), the set \( sk(M) = \{ sk(m) \mid m \in M \} \) is called the \textit{Skolem program} of \( M \).
Mapping Programs — Skolemization

For a mapping rule

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y_1'), \ldots, J_i^+(y_i'), Q^S(x). \]

define the Skolem mapping rule \( sk(m) \) by replacing each existential variable \( z \) in \( H^T(x, z) \) with a Skolem function symbol \( sk_z(s) \), where \( s \) is an ordered sequence of the universal variables \( x \).

Definition (Skolem program \([GGLS15]\))

For a mapping program \( M \), the set \( sk(M) = \{ sk(m) \mid m \in M \} \) is called the Skolem program of \( M \).

Example:

\[ sk(m_1) : \text{hasSup}(X, sk_z(X)) \leftarrow \text{not } \exists Y. \text{depHeadOf}(X, Y), \text{Empl}(X), \text{Jobs_DB}(X, P). \]
Definition (Partial ground programs, analogous to [GGLS15])

The partial grounding $PG(m)$ of a mapping rule $m$ is the set of all partial ground instances of $m$ over constants in $\Sigma_D$ for those variables that are not existential variables in the $J^-_i$. The partial ground program of a mapping program $\mathcal{M}$ is the set $PG(\mathcal{M}) = \bigcup_{m \in \mathcal{M}} PG(m)$.

Example:

$$
sk(m_1) : \text{hasSup}(X, sk_z(X)) \leftarrow \textbf{not} \ \exists Y. \text{depHeadOf}(X, Y),
$$
$$Empl(X), Jobs_{DB}(X, P).
$$
Definition (Partial ground programs, analogous to [GGLS15])

The partial grounding $PG(m)$ of a mapping rule $m$ is the set of all partial ground instances of $m$ over constants in $\Sigma_D$ for those variables that are not existential variables in the $J_i^-$. The partial ground program of a mapping program $\mathcal{M}$ is the set $PG(\mathcal{M}) = \bigcup_{m \in \mathcal{M}} PG(m)$.

Example:

\[
\text{sk}(m_1) : \text{hasSup}(X, \text{sk}_z(X)) \leftarrow \textbf{not } \exists Y . \text{depHeadOf}(X, Y), \\
\text{Empl}(X), \text{Jobs_DB}(X, P).
\]
Example:

\[ sk(m_1) : hasSup(X, sk_z(X)) \leftarrow \neg \exists Y. \text{depHeadOf}(X, Y), \]
\[ \text{Empl}(X), \text{Jobs_DB}(X, P). \]

Assume the constants occurring in the database are \{a, b\}, then \( PG(sk(m_1)) \) consists of the four mapping rules

\[ hasSup(a, sk_z(a)) \leftarrow \neg \exists Y. \text{depHeadOf}(a, Y), \text{Empl}(a), \text{Jobs_DB}(a, a). \]
\[ hasSup(a, sk_z(a)) \leftarrow \neg \exists Y. \text{depHeadOf}(a, Y), \text{Empl}(a), \text{Jobs_DB}(a, b). \]
\[ hasSup(b, sk_z(b)) \leftarrow \neg \exists Y. \text{depHeadOf}(b, Y), \text{Empl}(b), \text{Jobs_DB}(b, a). \]
\[ hasSup(b, sk_z(b)) \leftarrow \neg \exists Y. \text{depHeadOf}(b, Y), \text{Empl}(b), \text{Jobs_DB}(b, b). \]
Since mapping rules include ontology predicates, we must take the ontology $\mathcal{T}$ into account when constructing the reduct:

**Definition ($\mathcal{T}$-reduct)**

Given an ontology $\mathcal{T}$, the $\mathcal{T}$-reduct $\text{PG}(\mathcal{M})^\mathcal{A}$ of a partial ground mapping program $\text{PG}(\mathcal{M})$ w.r.t. an interpretation $\mathcal{A}$ is the program obtained from $\text{PG}(\mathcal{M})$ after applying the following:

1. Remove all mapping rules $m$ where there exists some $i \leq k$ such that $\mathcal{T} \cup \mathcal{A} \models J_i^-$.
2. Remove all negative justifications from the remaining rules.
Since mapping rules include ontology predicates, we must take the ontology $\mathcal{T}$ into account when constructing the reduct:

**Definition ($\mathcal{T}$-reduct)**

Given an ontology $\mathcal{T}$, the $\mathcal{T}$-reduct $PG(M)^A$ of a partial ground mapping program $PG(M)$ w.r.t. an interpretation $A$ is the program obtained from $PG(M)$ after applying the following:

1. Remove all mapping rules $m$ where there exists some $i \leq k$ such that $\mathcal{T} \cup A \models J_i^-$.
2. Remove all negative justifications from the remaining rules.

$\rightarrow$ the $\mathcal{T}$-reduct is a positive mapping program
A mapping interpretation $\mathcal{A}$ is a consistent subset of $HB_{sk(\mathcal{M})}$ (Herbrand base over $sk(\mathcal{M})$).

$\mathcal{A}$ satisfies the body of a positive Skolemized mapping rule

$$sk(m) : H^T(x, sk_z(x)) \leftarrow J^+_1(y'_1), \ldots, J^+_l(y'_l), Q^S(x).$$

if the following holds: for every query answer $t$ of $Q^S$ over $\mathcal{D}$, every interpretation $I$ with $I \models \mathcal{T} \cup \mathcal{A}$ satisfies $J^+_j[t]$ for all $j \leq l$.

$\mathcal{A} \models sk(m)$ if $\mathcal{A}$ satisfies the head or does not satisfy the body of $sk(m)$.

---

Mapping Data to Ontologies With Exception

Lecture :: 11th July
A *mapping interpretation* $\mathcal{A}$ is a consistent subset of $HB_{sk(M)}$ (Herbrand base over $sk(M)$).

$\mathcal{A}$ satisfies the body of a positive Skolemized mapping rule

$$sk(m) : H^T(x, sk_z(x)) \leftarrow J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x).$$

if the following holds: for every query answer $t$ of $Q^S$ over $D$, every interpretation $I$ with $I \models T \cup \mathcal{A}$ satisfies $J^+_j[t]$ for all $j \leq l$.

$\mathcal{A} \models sk(m)$ if $\mathcal{A}$ satisfies the head or does not satisfy the body of $sk(m)$.

**Definition ($\mathcal{T}$-Answer Set)**

A mapping interpretation $\mathcal{A} \subseteq HB_{sk(M)}$ is a *$\mathcal{T}$-answer set* of $M$ if it is a $\subseteq$-minimal model of the reduct $PG(sk(M))^\mathcal{A}$.
Example:
Let $T = \{ \text{Boss} \sqsubseteq \exists \text{depHeadOf} \}$ and $M$ consist of

$m_1 : \exists Z. \text{hasSup}(X, Z) \leftarrow \text{not} \exists Y. \text{depHeadOf}(X, Y), \text{Empl}(X), \text{Jobs_DB}(X, P)$.

$m_2 : \text{Boss}(X) \leftarrow \text{Jobs_DB}(X, b)$.

$m_3 : \text{Empl}(X) \leftarrow \text{Jobs_DB}(X, P)$.
Example:
Let $T = \{\text{Boss} \sqsubseteq \exists \text{depHeadOf}\}$ and $\mathcal{M}$ consist of

$m_1 : \exists Z. \text{hasSup}(X, Z) \leftarrow \text{not} \exists Y. \text{depHeadOf}(X, Y), \text{Empl}(X), \text{Jobs_DB}(X, P)$.

$m_2 : \text{Boss}(X) \leftarrow \text{Jobs_DB}(X, b)$.

$m_3 : \text{Empl}(X) \leftarrow \text{Jobs_DB}(X, P)$.

Then for $\mathcal{A} = \{\text{Jobs_DB}(a, b), \text{Empl}(a), \text{Boss}(a)\}$, the rules

$\text{hasSup}(a, \text{sk}_2(a)) \leftarrow \text{not} \exists Y. \text{depHeadOf}(a, Y), \text{Empl}(a), \text{Jobs_DB}(a, v)$.

for $v \in \{a, b\}$ are removed from $PG(\text{sk}(\mathcal{M}))^\mathcal{A}$, since $T \cup \mathcal{A} \not\vdash \exists Y. \text{depHeadOf}(a, Y)$.
Example:
Let $T = \{\text{Boss} \sqsubseteq \exists \text{depHeadOf}\}$ and $M$ consist of

$m_1 : \exists Z. \text{hasSup}(X, Z) \leftarrow \text{not} \exists Y. \text{depHeadOf}(X, Y), \text{Emp}(X), \text{Jobs_D}\text{b}(X, P)$.

$m_2 : \text{Boss}(X) \leftarrow \text{Jobs_D}\text{b}(X, b)$.

$m_3 : \text{Emp}(X) \leftarrow \text{Jobs_D}\text{b}(X, P)$.

Then for $A = \{\text{Jobs_D}\text{b}(a, b), \text{Emp}(a), \text{Boss}(a)\}$, the rules

$\text{hasSup}(a, \text{sk}_z(a)) \leftarrow \text{not} \exists Y. \text{depHeadOf}(a, Y), \text{Emp}(a), \text{Jobs_D}\text{b}(a, v)$.

for $v \in \{a, b\}$ are removed from $\text{PG}((\text{sk}(M))^A)$, since $T \cup A \models \exists Y. \text{depHeadOf}(a, Y)$.

Then the $T$-reduct consists of all groundings of:

$\text{hasSup}(b, \text{sk}_z(b)) \leftarrow \text{Emp}(b), \text{Jobs_D}\text{b}(b, Y)$.

$\text{Boss}(X) \leftarrow \text{Jobs_D}\text{b}(X, b)$.

$\text{Emp}(X) \leftarrow \text{Jobs_D}\text{b}(X, P)$.
Example:
Let $T = \{ Boss \sqsubseteq \exists \text{depHeadOf} \}$ and $M$ consist of

$m_1 : \exists Z. \text{hasSup}(X, Z) \leftarrow \textbf{not} \exists Y. \text{depHeadOf}(X, Y), \text{Empl}(X), \text{Jobs_DB}(X, P)$.
$m_2 : \text{Boss}(X) \leftarrow \text{Jobs_DB}(X, b)$.
$m_3 : \text{Empl}(X) \leftarrow \text{Jobs_DB}(X, P)$.

Then for $A = \{ \text{Jobs_DB}(a, b), \text{Empl}(a), \text{Boss}(a) \}$, the rules $T$-answer set!

$\text{hasSup}(a, sk_z(a)) \leftarrow \textbf{not} \exists Y. \text{depHeadOf}(a, Y), \text{Empl}(a), \text{Jobs_DB}(a, v)$.

for $v \in \{a, b\}$ are removed from $PG(sk(M))^A$, since $T \cup A \models \exists Y. \text{depHeadOf}(a, Y)$.

Then the $T$-reduct consists of all groundings of:

$\text{hasSup}(b, sk_z(b)) \leftarrow \text{Empl}(b), \text{Jobs_DB}(b, Y)$.
$\text{Boss}(X) \leftarrow \text{Jobs_DB}(X, b)$.
$\text{Empl}(X) \leftarrow \text{Jobs_DB}(X, P)$.
Mapping Programs in OBDA

Putting mapping programs into an OBDA context:

**Definition (Generalized OBDA)**

A *generalized OBDA specification* is a tuple \((D, M, T)\) consisting of a database \(D\), a mapping program \(M\), and an ontology \(T\).

**Definition (Generalized OBDA semantics)**

A tuple \((I, A)\) consisting of a first-order model \(I\) and a mapping interpretation \(A\) is a model of \((D, M, T)\) if it satisfies the following:

1. \(I \models T \cup A\),
2. \(A\) is a \(T\)-answer set of \(M\).
Mapping Programs

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y_1'), \ldots, J_l^+(y_l'), Q^S(x). \]

Noteworthy:

- Due to \( y, y' \subseteq x \), \( Q^S(x) \) acts as a guard. Mapping rules are only applicable to tuples of constants from the database, not existential witnesses generated by mapping heads!
Mapping Programs

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x). \]

Noteworthy:

- Due to \( y, y' \subseteq x \), \( Q^S(x) \) acts as a guard. Mapping rules are only applicable to tuples of constants from the database, not existential witnesses generated by mapping heads!
- The partial grounding is always finite (for finite databases)
Mapping Programs

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y_1'), \ldots, J_l^+(y_l'), Q^S(x). \]

Noteworthy:

- Due to \( y, y' \subseteq x \), \( Q^S(x) \) acts as a guard. Mapping rules are only applicable to tuples of constants from the database, not existential witnesses generated by mapping heads!
- the partial grounding is always finite (for finite databases)
- Can express ontology constraints on the database: Let \( \varphi \) be the query to retrieve all tuples that are not in \( \text{JOBS\_DB} \). Then

\[ \bot \leftarrow \text{Person}(X), \varphi(X). \]

expresses “All instances of \( \text{Person} \) must be contained in the table \( \text{JOBS\_DB} \).”
Complexity

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x). \]

In general, mapping programs are extremely expressive: arbitrary first-order formulas \( H^T, J^-, J^+ \).
Complexity

\[ m : H^T(x, z) \leftarrow \text{not } J_1^-(y_1), \ldots, \text{not } J_k^-(y_k), J_1^+(y'_1), \ldots, J_l^+(y'_l), Q^S(x). \]

In general, mapping programs are extremely expressive: arbitrary first-order formulas \( H^T, J^-, J^+. \)

**Theorem**

*The problem of checking \( M \models A \) for a given mapping program \( M \) and a ground atom \( A \) is undecidable.*
Let $(\mathcal{T}, \mathcal{L})$ consist of an ontology $\mathcal{T}$ and a set $\mathcal{L}$ of formulas such that $\mathcal{T}$-entailment of any $\varphi \in \mathcal{L}$ is decided by an oracle $O(\mathcal{T}, \mathcal{L})$. 

- Restrict $H^{T, J−, J+}$ to formulas from $\mathcal{L}$. Then for a partially ground Skolem program $M$, a guess-and-check algorithm can be used for $\mathcal{T}$-answer set construction: guess $A$, construct the $\mathcal{T}$-reduct $M_A$, check satisfiability and minimality of $A$.

- Generalization of the classical ASP guess-and-check: for $\mathcal{T} = \emptyset$ and $\mathcal{L} =$ {ground atoms}, mapping programs are precisely ASP programs. 

$\rightarrow$ at least NP-hard (data complexity)
Let $(\mathcal{T}, \mathcal{L})$ consist of an ontology $\mathcal{T}$ and a set $\mathcal{L}$ of formulas such that $\mathcal{T}$-entailment of any $\varphi \in \mathcal{L}$ is decided by an oracle $\mathcal{O}_{(\mathcal{T}, \mathcal{L})}$.

Restrict $H^\mathcal{T}$, $J^-$, $J^+$ to formulas from $\mathcal{L}$. 
Complexity

- Let \((\mathcal{T}, \mathcal{L})\) consist of an ontology \(\mathcal{T}\) and a set \(\mathcal{L}\) of formulas such that \(\mathcal{T}\)-entailment of any \(\varphi \in \mathcal{L}\) is decided by an oracle \(\mathcal{O}_{(\mathcal{T}, \mathcal{L})}\).
- Restrict \(H^\mathcal{T}, J^-, J^+\) to formulas from \(\mathcal{L}\).
- Then for a partially ground Skolem program \(\mathcal{M}\), a guess-and-check algorithm can be used for \(\mathcal{T}\)-answer set construction: guess \(\mathcal{A}\), construct the \(\mathcal{T}\)-reduct \(\mathcal{M}^\mathcal{A}\), check satisfiability and minimality of \(\mathcal{A}\).
Complexity

- Let \((T, L)\) consist of an ontology \(T\) and a set \(L\) of formulas such that \(T\)-entailment of any \(\varphi \in L\) is decided by an oracle \(O_{(T, L)}\).

- Restrict \(H^T, J^-, J^+\) to formulas from \(L\).

- Then for a partially ground Skolem program \(M\), a guess-and-check algorithm can be used for \(T\)-answer set construction: guess \(A\), construct the \(T\)-reduct \(M^A\), check satisfiability and minimality of \(A\).

- Generalization of the classical ASP guess-and-check: for \(T = \emptyset\) and \(L = \{\text{ground atoms}\}\), mapping programs are precisely ASP programs.
Let \((\mathcal{T}, \mathcal{L})\) consist of an ontology \(\mathcal{T}\) and a set \(\mathcal{L}\) of formulas such that \(\mathcal{T}\)-entailment of any \(\varphi \in \mathcal{L}\) is decided by an oracle \(\mathcal{O}_{(\mathcal{T}, \mathcal{L})}\).

Restrict \(H^T, J^-, J^+\) to formulas from \(\mathcal{L}\).

Then for a partially ground Skolem program \(\mathcal{M}\), a guess-and-check algorithm can be used for \(\mathcal{T}\)-answer set construction: guess \(\mathcal{A}\), construct the \(\mathcal{T}\)-reduct \(\mathcal{M}^\mathcal{A}\), check satisfiability and minimality of \(\mathcal{A}\).

A generalization of the classical ASP guess-and-check: for \(\mathcal{T} = \emptyset\) and \(\mathcal{L} = \{\text{ground atoms}\}\), mapping programs are precisely ASP programs. \(\rightarrow\) at least NP-hard (data complexity)
Complexity

More generally:

**Theorem**

Let \((\mathcal{T}, \mathcal{L})\) be a pair consisting of an first-order ontology \(\mathcal{T}\) and a set of formulas \(\mathcal{L}\) over the language of \(\mathcal{T}\) such that \(\mathcal{T}\)-entailment is \(|\mathcal{O}(\mathcal{T}, \mathcal{L})|\)-hard for an oracle \(\mathcal{O}(\mathcal{T}, \mathcal{L})\). Then for a partially ground Skolemized mapping program \(\mathcal{M}\) where \(H^\mathcal{T}, J^-, J^+\) are from \(\mathcal{L}\), \(\mathcal{T}\)-answer set existence is \(\mathbb{NP}^{\mathcal{O}(\mathcal{T}, \mathcal{L})}\)-complete.

Proof idea:

Universal reduction argument: An \(\mathbb{NP}^{\mathcal{O}(\mathcal{T}, \mathcal{L})}\) Turing machine can be encoded as a mapping program in the same manner an \(\mathbb{NP}\) TM can be encoded in ASP, but allowing for oracle calls in rule bodies.
Complexity

More generally:

**Theorem**

Let \((T, \mathcal{L})\) be a pair consisting of an first-order ontology \(T\) and a set of formulas \(\mathcal{L}\) over the language of \(T\) such that \(T\)-entailment is \(|\mathcal{O}(T, \mathcal{L})|\)-hard for an oracle \(\mathcal{O}(T, \mathcal{L})\). Then for a partially ground Skolemized mapping program \(\mathcal{M}\) where \(H^T, J^-, J^+\) are from \(\mathcal{L}\), \(T\)-answer set existence is \(NP^{\mathcal{O}(T, \mathcal{L})}\)-complete.

**Proof idea:**

Universal reduction argument: An \(NP^{\mathcal{O}(T, \mathcal{L})}\) Turing machine can be encoded as a mapping program in the same manner an NP TM can be encoded in ASP, but allowing for oracle calls in rule bodies.
The $T$-rewriting of a query $\varphi$ is a query $\overline{\varphi}$ such that
\[(D, M, T) \models \varphi \text{ iff } (D, M, \emptyset) \models \overline{\varphi}.\]

A query $\varphi$ is UCQ-rewritable if its $T$-rewriting is a union of conjunctive queries.
The $T$-rewriting of a query $\varphi$ is a query $\overline{\varphi}$ such that

$$(D, M, T) \models \varphi \text{ iff } (D, M, \emptyset) \models \overline{\varphi}. $$

A query $\varphi$ is **UCQ-rewritable** if its $T$-rewriting is a union of conjunctive queries.

Let $M$ contain only rules where $J^+$, $J^-$ are UCQ-rewritable w.r.t. $T$ and $H^T$ are conjunctive queries.
The $\mathcal{T}$-rewriting of a query $\varphi$ is a query $\overline{\varphi}$ such that

$$(D, M, T) \models \varphi \text{ iff } (D, M, \emptyset) \models \overline{\varphi}.$$  

A query $\varphi$ is UCQ-rewritable if its $\mathcal{T}$-rewriting is a union of conjunctive queries.

Let $M$ contain only rules where $J^+, J^-$ are UCQ-rewritable w.r.t. $\mathcal{T}$ and $H^\mathcal{T}$ are conjunctive queries.

Define $\overline{M}$ as the program obtained by replacing $J^+, J^-$ with their $\mathcal{T}$-rewritings $\overline{J^+}$ and $\overline{J^-}$.

$\overline{M}$ is equivalent to a $\exists$-program! (standard logic programming transformations)
Theorem

For a generalized OBDA specification \((\mathcal{D}, \mathcal{M}, \mathcal{T})\), where \(J^+, J^-\) in \(\mathcal{M}\) are UCQ-rewritable with respect to \(\mathcal{T}\), there exists an \(\exists\)-ASP program \(\mathcal{M}'\) such that for a query \(q\) over \(\mathcal{T}\)

\[(\mathcal{D}, \mathcal{M}, \mathcal{T}) \models q[t] \iff \mathcal{M}' \models \overline{q}[t],\]

i.e., query answering over \((\mathcal{D}, \mathcal{M}, \mathcal{T})\) reduces to cautious reasoning over \(\mathcal{M}'\).

Proof idea: Straightforward calculation.
UCQ-Rewritability — Reduction to ASP

**Theorem**

For a generalized OBDA specification $(\mathcal{D}, \mathcal{M}, T)$, where $J^+, J^-$ in $\mathcal{M}$ are UCQ-rewritable with respect to $T$, there exists an $\exists$-ASP program $\mathcal{M}'$ such that for a query $q$ over $T$

$$(\mathcal{D}, \mathcal{M}, T) \models q[t] \iff \mathcal{M}' \models \overline{q}[t],$$

i.e., query answering over $(\mathcal{D}, \mathcal{M}, T)$ reduces to cautious reasoning over $\mathcal{M}'$.

**Proof idea:** Straightforward calculation.

[GGLS15] ⇒ can be further reduced to ASP.
Summary and Future Work

Summary:
- mapping programs as new mapping framework based on an extension of $\exists$-ASP.
- supports default exception handling and ontology constraints
- Algorithm for general, decidable case
- Reasoning reduces to ASP if the mapping program is UCQ-rewritable.

Future Work:
- analyze mapping programs from a parameterized complexity perspective
- determine which fragments of mapping programs admit a query rewriting process (currently not possible)
- Analyze when mapping programs can be rewritten to a classical OBDA mapping
- proof-of-concept implementation of an OBDA system using mapping programs
Francesco M. Donini, Daniele Nardi, and Riccardo Rosati.
Description logics of minimal knowledge and negation as failure.
*ACM Trans. Comput. Logic*, 3(2):177–225, April 2002.

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.
Combining answer set programming with description logics for the semantic web.
*Artificial Intelligence*, 172(12–13):1495 – 1539, 2008.

Fabien Garreau, Laurent Garcia, Claire Lefèvre, and Igor Stéphan.
∃-ASP.
In *Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, Buenos Aires, Argentina, July 25-27, 2015., 2015.

Carsten Lutz, İnanç Seylan, and Frank Wolter.
Ontology-based data access with closed predicates is inherently intractable (sometimes).
In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1024–1030. AAAI Press, 2013.

Boris Motik and Riccardo Rosati.
Reconciling description logics and rules.
*J. ACM*, 57(5):30:1–30:62, June 2010.