Towards a Semantic Web
Unifying Logic

Matthias Knorr

CENTRIA, Universidade Nova de Lisboa

ICCL Summer School 2013
About the Lecturer

• PostDoc at CENTRIA, Universidade Nova de Lisboa (UNL)
  • One of the partners in the programs EMCL and EPCL
  • Strong history in Logic Programming
  • Two ECCAI fellows (Luís M. Pereira and José J. Alferes)

• Research focus: Non-monotonic Reasoning in the Semantic Web; currently in the project ERRO

• More info (including these slides) at http://centria.di.fct.unl.pt/~mknorr/
Recap on Ontologies and Rules

- Towards a Unifying Logic
- Building on the contents of “Ontologies and Rules”
- Focusing on Non-monotonic Extensions
Why Non-monotonic Extensions?

Example: matching patients to clinical trial criteria

- Open World Assumption (OWA) in general preferable on the Web
  - Without clinical test, no assumptions can be made on outcome
Why Non-monotonic Extensions?

Example: matching patients to clinical trial criteria

- Open World Assumption (OWA) in general preferable on the Web
  - Without clinical test, no assumptions can be made on outcome
- But with complete knowledge, Closed World Assumption (CWA) is better
  - Patient’s medication is fully known
Why Non-monotonic Extensions?

Example: matching patients to clinical trial criteria

• Open World Assumption (OWA) in general preferable on the Web
  • Without clinical test, no assumptions can be made on outcome
• But with complete knowledge, Closed World Assumption (CWA) is better
  • Patient’s medication is fully known
• Requirement for local closure of certain information
Why Non-monotonic Extensions?

\[
\begin{align*}
\text{Person} & \sqsubseteq \text{HeartLeft} \sqcup \text{HeartRight} \\
\text{HeartLeft} \sqcap \text{HeartRight} & \sqsubseteq \bot \\
\text{Person} & \sqsubseteq \exists \text{has.SpinalColumn} \\
\exists \text{has.SpinalColumn} & \sqsubseteq \text{Vertebrate} \\
\text{Person}(\text{Bob}) & 
\end{align*}
\]
Why Non-monotonic Extensions?

\[
\begin{align*}
\text{Person} & \sqsubseteq \text{HeartLeft} \sqcap \text{HeartRight} \\
\text{HeartLeft} \sqcap \text{HeartRight} & \sqsubseteq \bot \\
\text{Person} & \sqsubseteq \exists \text{has.SpinalColumn} \\
\exists \text{has.SpinalColumn} & \sqsubseteq \text{Vertebrate} \\
\text{Person}(\text{Bob}) & \\
\Rightarrow & \text{Vertebrate}(\text{Bob}), \text{Person} \sqsubseteq \text{Vertebrate}, \text{and} \exists x. \text{SpinalColumn}(x) \text{ derivable}
\end{align*}
\]
Why Non-monotonic Extensions?

\[ Person \sqsubseteq HeartLeft \uplus HeartRight \]
\[ HeartLeft \cap HeartRight \sqsubseteq \bot \]
\[ Person \sqsubseteq \exists has.SpinalColumn \]
\[ \exists has.SpinalColumn \sqsubseteq Vertebrate \]
\[ Person(Bob) \]

\[ HeartLeft(x) \leftarrow Vertebrate(x), \text{not} HeartRight(x) \]
\[ SSN\_OK(x) \leftarrow hasSSN(x, y) \]
\[ f \leftarrow Person(x), \text{not} SSN\_OK(x) \]
Why Non-monotonic Extensions?

\[
\begin{align*}
\text{Person} & \sqsubseteq \text{HeartLeft} \sqcap \text{HeartRight} \\
\text{HeartLeft} & \sqcap \text{HeartRight} \sqsubseteq \bot \\
\text{Person} & \sqsubseteq \exists \text{has.SpinalColumn} \\
\exists \text{has.SpinalColumn} & \sqsubseteq \text{Vertebrate} \\
\text{Person}(\text{Bob}) & \\
\text{HeartLeft}(x) & \leftarrow \text{Vertebrate}(x), \text{notHeartRight}(x) \\
\text{SSN}_{\text{OK}}(x) & \leftarrow \text{hasSSN}(x, y) \\
\mathbf{f} & \leftarrow \text{Person}(x), \text{notSSN}_{\text{OK}}(x) \\
\Rightarrow & \text{HeartLeft}(\text{Bob}); \text{hasSSN}(\text{Bob}, y) \text{ for ground } y \text{ required}
\end{align*}
\]

Model defaults and exceptions, and integrity constraints
Outline

1. Combining non-monotonic rules and DLs
2. Top-down querying in such combinations
3. Generalized unifying language
Combining non-monotonic rules and DLs
Why first focus on Rules?

- Direct non-monotonic extensions exist for DLs, based on, e.g., default logic, epistemic logic, and circumscription
- But non-trivial with few results in terms of implementations
- Rules easily extended to non-monotonic features
- Well-studied field Logic Programming including fast reasoners
- Leverage the knowledge and reasoners available
How to Combine DLs and Rules?

Non-trivial because of non-monotonicity

- Rules “on top” of ontologies
  - First define concepts then define rules “on top”
  - Rules deal with ontologies as external code
  - Separate semantics for rule and ontology predicates
How to Combine DLs and Rules?

Non-trivial because of non-monotonicity

• Rules “on top” of ontologies
  • First define concepts then define rules “on top”
  • Rules deal with ontologies as external code
  • Separate semantics for rule and ontology predicates

• Tight (full) integration
  • Allow for “defining” joint predicates both in the ontology and the rule layer and use them (almost) freely
How to Combine DLs and Rules?

Non-trivial because of non-monotonicity

- Rules “on top” of ontologies
  - First define concepts then define rules “on top”
  - Rules deal with ontologies as external code
  - Separate semantics for rule and ontology predicates
- Tight (full) integration
  - Allow for “defining” joint predicates both in the ontology and the rule layer and use them (almost) freely
- Modular combination (loose coupling)
  - Trades some expressiveness for an easier to implement interface integration
Hybrid MKNF Knowledge Bases

- Seamless, tight integration; reflexive, expressive, yet competitive w.r.t. computational complexity
- Introduced by Motik and Rosati in 2007 (extended in [Motik and Rosati, JACM10])
- Based on Logics of Minimal Knowledge and Negation as Failure by Lifschitz in 1991: first-order logic with equality and modal operators $K$ and $\text{not}$
Hybrid MKNF Knowledge Bases

• Consist of a DL knowledge base $\mathcal{O}$ and a finite set of rules, $\mathcal{P}$, of the form

$$KH_1 \lor \ldots \lor KH_l \leftarrow KA_1, \ldots, KA_n, \text{not} B_1, \ldots, \text{not} B_m$$

• $H_i$, $A_j$, and $B_k$ are generalized atoms - arbitrary first-order formulas - to capture, e.g., conjunctive queries

$$K\text{interestingCity}(x) \leftarrow K(\exists y : Has(x, y) \land \text{Recreational}(y)), \text{not RainyCity}(x)$$
Decidability(1)

Ensured by two technical restrictions:

1. **DL-safety of rules**: every variable appears in a positive non-DL-atom in the body (restriction of application of rules to known individuals when grounding)

   - Convention: capitalized predicates yield DL-atoms

   $$\text{K_{interestingCity}}(x) \leftarrow \text{K}(\exists y : \text{Has}(x, y) \land \text{Recreational}(y)),$$
   $$\text{not RainyCity}(x) \times$$
   $$\text{K_{interestingCity}}(x) \leftarrow \text{K}(\exists y : \text{Has}(x, y) \land \text{Recreational}(y)),$$
   $$\text{not RainyCity}(x), o(x)$$

   + $o(x)$ for all individuals $x$ appearing in the KB ✔
Decidability(2)

Ensured by two technical restrictions:

1. Decidability of DL reasoning in combination with
generalized atoms; related to the following:

\[ \text{K}_\text{interestingCity}(x) \leftarrow \text{K}(\exists y : \text{Has}(x, y) \land \text{Recreational}(y)), \not \text{RainyCity}(x), o(x) \]

corresponds to

\[ \exists \text{Has.Recreational} \sqsubseteq \text{RecreationalCity} \]
\[ \text{K}_\text{interestingCity}(x) \leftarrow \text{K}_\text{RecreationalCity}(x), \not \text{RainyCity}(x), o(x) \]
Reasoning – Example(1)

Collect all modal atoms $KA$ in ground hybrid MKNF KB $\mathcal{K}_G$ and add $KB$ for each $\text{not } B$ in $\mathcal{K}_G$ – obtain the set $KA(\mathcal{K}_G)$:

$$A \sqsubseteq B \quad B \sqsubseteq F \quad KA(a) \quad Kc(a) \leftarrow KB(a), \text{not}d(a)$$
Reasoning – Example(1)

Collect all modal atoms $KA$ in ground hybrid MKNF KB $\mathcal{K}_G$ and add $KB$ for each $\text{not } B$ in $\mathcal{K}_G$ – obtain the set $KA(\mathcal{K}_G)$:

\[
A \sqsubseteq B \quad B \sqsubseteq F \quad KA(a) \quad Kc(a) \leftarrow KB(a), \text{not d}(a)
\]

- $KA(\mathcal{K}_G) = \{KA(a), KB(a), Kc(a), Kd(a)\}$
Reasoning – Example(2)

Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

1. Check if \(\mathcal{K}_G\) is satisfiable w.r.t. \((P, N)\):

\[
\begin{align*}
\text{•} & \quad (P_1, N_1) = (\{K_A(a), K_B(a), K_c(a)\}, \{K_d(a)\}) & \checkmark \\
\text{•} & \quad (P_2, N_2) = (\{K_A(a), K_B(a), K_c(a), K_d(a)\}, \emptyset) & \checkmark \\
\text{•} & \quad (P_3, N_3) = (\{K_A(a), K_B(a)\}, \{K_c(a), K_d(a)\}) & \times
\end{align*}
\]
Reasoning – Example(2)

Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

1. Check if \(K_G\) is satisfiable w.r.t. \((P, N)\):

\[
A \sqsubseteq B \quad B \sqsubseteq F \quad KA(a) \quad Kc(a) \leftarrow KB(a), \text{not}d(a)
\]

- \((P_1, N_1) = (\{KA(a), KB(a), Kc(a)\}, \{Kd(a)\})\) ✓
Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

1. Check if \(\mathcal{K}_G\) is satisfiable w.r.t. \((P, N)\):

\[
A \sqsubseteq B \quad B \sqsubseteq F \quad KA(a) \quad Kc(a) \leftarrow KB(a), \text{not} d(a)
\]

- \((P_1, N_1) = (\{KA(a), KB(a), Kc(a)\}, \{Kd(a)\})\) ✓
- \((P_2, N_2) = (\{KA(a), KB(a), Kc(a), Kd(a)\}, \emptyset)\) ✓
Reasoning – Example(2)

Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

1. Check if \(\mathcal{K}_G\) is satisfiable w.r.t. \((P, N)\):

\[
A \sqsubseteq B \quad B \sqsubseteq F \quad \mathbf{K}A(a) \quad \mathbf{K}c(a) \leftarrow \mathbf{K}B(a), \neg \mathbf{d}(a)
\]

- \((P_1, N_1) = (\{\mathbf{K}A(a), \mathbf{K}B(a), \mathbf{K}c(a)\}, \{\mathbf{K}d(a)\})\) ✓
- \((P_2, N_2) = (\{\mathbf{K}A(a), \mathbf{K}B(a), \mathbf{K}c(a), \mathbf{K}d(a)\}, \emptyset)\) ✓
- \((P_3, N_3) = (\{\mathbf{K}A(a), \mathbf{K}B(a)\}, \{\mathbf{K}c(a), \mathbf{K}d(a)\})\) ✗
Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

2. Fixing \(N\) (used for the evaluation of \textsc{not} \)), check if \(P\) is minimal:
Reasoning – Example(2)

Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

2 Fixing \(N\) (used for the evaluation of \(\text{not}\)), check if \(P\) is minimal:

\[
\begin{align*}
A \subseteq B &\quad B \subseteq F & KA(a) &\quad Kc(a) \leftarrow KB(a), \text{not}d(a) \\
\end{align*}
\]

\[
\text{• } (P_1, N_1) = (\{KA(a), KB(a), Kc(a)\}, \{Kd(a)\}) \checkmark
\]
Guess a partition \((P, N)\) of the collected set (partition into true \((P)\) and false \((N)\) modal atoms) and

2. Fixing \(N\) (used for the evaluation of \(\text{not}\)), check if \(P\) is minimal:

\[
A \subseteq B \quad B \subseteq F \quad K_A(a) \quad K_c(a) \leftarrow K_B(a), \text{not}d(a)
\]

- \((P_1, N_1) = (\{K_A(a), K_B(a), K_c(a)\}, \{K_d(a)\})\) ✓
- \((P_2, N_2) = (\{K_A(a), K_B(a), K_c(a), K_d(a)\}, \emptyset)\) ×

Reasoning on \(\emptyset \cup P_1\): \(K_F(a)\) is also derivable
Properties of Hybrid MKNF

- Generalizes/captures (sometimes not entirely) quite a number of different approaches
- Faithful w.r.t. Stable Models for empty $\mathcal{O}$ and w.r.t. OWL for empty $\mathcal{P}$
- Data complexity of instance checking in MKNF:

<table>
<thead>
<tr>
<th>rules</th>
<th>$\mathcal{DL} = \emptyset$</th>
<th>$\mathcal{DL} \in \mathbb{P}$</th>
<th>$\mathcal{DL} \in \text{coNP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>definite</td>
<td>$\mathbb{P}$</td>
<td>$\mathbb{P}$</td>
<td>$\text{coNP}$</td>
</tr>
<tr>
<td>stratified</td>
<td>$\mathbb{P}$</td>
<td>$\mathbb{P}$</td>
<td>$\Delta^p_2$</td>
</tr>
<tr>
<td>normal</td>
<td>$\text{coNP}$</td>
<td>$\text{coNP}$</td>
<td>$\Pi^p_2$</td>
</tr>
<tr>
<td>disjunctive</td>
<td>$\Pi^p_2$</td>
<td>$\Pi^p_2$</td>
<td>$\Pi^p_2$</td>
</tr>
</tbody>
</table>
Problems of two-valued Hybrid MKNF

- Models have to be guessed and checked
- Unrestricted rules increase computational complexity
- Queries for particular information require computation of the entire model
- Limited robustness, e.g., w.r.t. merging of KBs
  \[(K_u \leftarrow \text{not} u)\]

[Knorr et al., AI11] provides alternative based on well-founded semantics for non-disjunctive logic programs
Stable Models vs. Well-Founded Model in LP

\[ p \leftarrow \text{not} q \quad q \leftarrow \text{not} p \quad a \leftarrow \text{not} b \quad b \leftarrow \]

has two stable models \{p, b\} and \{q, b\}, while the unique well-founded model assigns \(t\) to \(b\), \(f\) to \(a\), and \(u\) to both \(p\) and \(q\).
Stable Models vs. Well-Founded Model in LP

\[ p \leftarrow \text{not}q \quad q \leftarrow \text{not}p \quad a \leftarrow \text{not}b \quad b \leftarrow \]

has two stable models \( \{p, b\} \) and \( \{q, b\} \), while the unique well-founded model assigns \( t \) to \( b \), \( f \) to \( a \), and \( u \) to both \( p \) and \( q \).

For

\[ p \leftarrow \text{not}p \quad q \leftarrow \text{not}q \quad a \leftarrow \text{not}b \quad b \leftarrow \]

the well-founded model is the same, but there are no stable models!
Stable vs. Well-Founded Semantics in Logic Programming

Stable Models/Answer Sets
- More expressive language
- More derivable information
- Fast ASP solvers available

Well-founded Model
- Lower computational complexity
- always exists
- top-down derivations possible

Similar for combinations of rules and ontologies
Well-Founded MKNF semantics

- Restricted to non-disjunctive rules over simple atoms
- Partitions divided into true, undefined, and false atoms
- Computation of the unique model in a bottom-up fashion
Immediate Consequence Operator

**Definition**
We define on subsets $S$ of $\text{KA}(\mathcal{K}_G)$ for ground, **positive** $\mathcal{K}_G$:

$$R_{\mathcal{K}_G}(S) = \{KH \mid \mathcal{P}_G \text{ contains a rule of the form } KH \leftarrow KA_1, \ldots, KA_n$$
$$\text{such that, for all } i, 1 \leq i \leq n, KA_i \in S\}$$

$$D_{\mathcal{K}_G}(S) = \{K\xi \mid KA \in \text{KA}(\mathcal{K}_G) \text{ and } OB_{\mathcal{O},S} \models \xi\}$$

$$T_{\mathcal{K}_G}(S) = R_{\mathcal{K}_G}(S) \cup D_{\mathcal{K}_G}(S)$$

$OB_{\mathcal{O},S}$ - first-order representation of $\mathcal{O}$ and assumed set $S$
Example Iteration $T_{KG}$

$$R \sqsubseteq S$$

$$KR(a) \leftarrow$$
$$Kq(a) \leftarrow KS(a)$$

$$T_K \uparrow 0 = \emptyset$$
$$T_K \uparrow 1 = \{KR(a)\}$$
$$T_K \uparrow 2 = \{KR(a), KS(a)\}$$
$$T_K \uparrow 3 = \{KR(a), KS(a), Kq(a)\}$$
MKNF Transform

Definition
Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \text{KA}(\mathcal{K}_G)$. The MKNF transform $\mathcal{K}_G/S$ is defined as $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$, where $\mathcal{P}_G/S$ contains all rules

$$KH \leftarrow KA_1, \ldots, KA_n$$

for which there exists a rule

$$KH \leftarrow KA_1, \ldots, KA_n, \text{not} B_1, \ldots, \text{not} B_m$$

in $\mathcal{P}_G$ with $KB_j \notin S$ for all $1 \leq j \leq m$. 
Example Transformation

consider \( S = \{ KS(a), KR(a) \} \) (highlighted part is removed):

\[
R \sqsubseteq S
\]

\[
KR(a) \leftarrow \text{not} \ R(b)
\]

\[
KR(b) \leftarrow \text{not} \ R(a)
\]

\[
Kq(a) \leftarrow KS(a), \text{not} q(a)
\]

Compute least model of the resulting KB:

\[
\Gamma_{KG}(S) = T_{KG/S} \uparrow \omega.
\]
Coherence

Problem: classical negation from $\mathcal{O}$ does not imply default negation in rules

\[
R \sqsubseteq \neg S \\
K S(a) \leftarrow \text{not} t(a) \\
K t(a) \leftarrow \text{not} S(a) \\
K R(a) \leftarrow
\]

Operators so far yield undefined for $K S(a)$ and $K t(a)$
**MKNF-coherent transform**

**Definition**

Let $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ be a ground hybrid MKNF knowledge base and $S \subseteq \text{KA}(\mathcal{K}_G)$. The **MKNF-coherent transform** $\mathcal{K}_G//S$ is defined as $\mathcal{K}_G//S = (\mathcal{O}, \mathcal{P}_G//S)$, where $\mathcal{P}_G//S$ contains all rules

$$KH \leftarrow KA_1, \ldots, KA_n$$

for which there exists a rule

$$KH \leftarrow KA_1, \ldots, KA_n, \text{not}B_1, \ldots, \text{not}B_m$$

in $\mathcal{P}_G$ with $KB_j \notin S$ for all $1 \leq j \leq m$ and $\text{OB}_{\mathcal{O},S} \not\models \neg H$. 

Matthias Knorr - CENTRIA, UNL
Example Coherence

MKNF-coherent transform for $S = \{KR(a)\}$:

- $R \subseteq \neg S$
- $KS(a) \leftarrow \text{not} t(a)$
- $Kt(a) \leftarrow \text{not} S(a)$
- $KR(a) \leftarrow$

Compute least model of the resulting KB:

$$\Gamma'_{K_G}(S) = T_{K_G/\uparrow S} \uparrow \omega$$
Inconsistencies

\[ R \sqsubseteq \neg S \]

\[ KS(a) \leftarrow not\ t(a) \]

\[ Kt(a) \leftarrow not\ R(a) \]

\[ KR(a) \leftarrow \]

By \( \Gamma'_{KG} \), we simply derive that \( KS(a) \) is false.
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

Alternating Iteration

\[ P_0 = \emptyset \]
\[ P_{n+1} = \Gamma_{\mathcal{K}_G}(N_n) \]
\[ P_\omega = \bigcup P_i \]

\[ N_0 = KA(\mathcal{K}_G) \]
\[ N_{n+1} = \Gamma'_{\mathcal{K}_G}(P_n) \]
\[ N_\omega = \bigcap N_i \]

two fixpoints for (finite) \( \omega \):
- \( P_\omega \) – everything that is true
- \( N_\omega \) – everything that is not false
Properties of Well-Founded MKNF

- Sound w.r.t. two-valued MKNF semantics
- Faithful w.r.t. first-order semantics for empty $\mathcal{P}$ and w.r.t. the Well-Founded Semantics for empty $\mathcal{O}$
- Given complexity $\mathcal{C}$ for instance checking in $\mathcal{O}$ we obtain a data complexity $\mathcal{P}^\mathcal{C}$; for $\mathcal{C} = \mathcal{P}$, polynomial data complexity
- Inconsistencies can be detected by a simple additional computation
- Automated computation yields model (if it exists)
Summary

- Sophisticated reasoning possible with combinations of ontologies and non-monotonic rules
- MKNF-based framework links information based on derivable information
- Framework itself very general and flexible
- Algorithms for fast computation available

What if we do only want to know a small piece of information?
Top-down querying
Top-down-Procedure \( \text{SLG}(\mathcal{O}) \)

- \( \text{SLG}(\mathcal{O}) \) [Alferes et al., ACM TOCL13] combines a DL reasoner with the top-down query engine XSB Prolog
- Instead of computing the whole model, only compute the part relevant for the model
- XSB realizes well-founded semantics in Logic Programming utilizing tabling (re-using previous results, avoiding infinite loops)
- Special oracle resolves queries to the ontology
DL Oracle

- Builds on derivation relation for $O$
- Returns (possibly empty) set of atoms $L$ (potentially derivable from the set of rules $P$) that together with $O$ allow to derive the (ground) queried atom $S$
- If subsequent queries for atoms in $L$ succeed, then $S$ succeeds
MKNF KB Example

PortCity(Barcelona)  OnSea(Barcelona, Mediterranean)
PortCity(Hamburg)    NonSeaSideCity(Hamburg)
RainyCity(Manchester) Has(Manchester, AquaticsCenter)
Recreational(AquaticsCenter)

SeaSideCity ⊑∃Has.Beach
Beach ⊑ Recreational
∃Has.Recreational ⊑ RecreationalCity
K SeaSideCity(x) ← K PortCity(x), not NonSeaSideCity(x)
K interestingCity(x) ← K RecreationalCity(x), not RainyCity(x)
K hasOnSea(x) ← K OnSea(x, y)
K false ← K SeaSideCity(x), not hasOnSea(x)
K summerDestination(x) ← K interestingCity(x), K OnSea(x, y)
Barcelona

Interesting?
MKNF KB Example Query(1)

\[
\begin{align*}
\text{PortCity}(\text{Barcelona}) & \quad \text{OnSea}(\text{Barcelona}, \text{Mediterranean}) \\
\text{PortCity}(\text{Hamburg}) & \quad \text{NonSeaSideCity}(\text{Hamburg}) \\
\text{RainyCity}(\text{Manchester}) & \quad \text{Has}(\text{Manchester}, \text{AquaticsCenter}) \\
\text{Recreational}(\text{AquaticsCenter}) & \\
\text{SeaSideCity} & \sqsubseteq \exists \text{Has}.\text{Beach} \\
\text{Beach} & \sqsubseteq \text{Recreational} \\
\exists \text{Has}.\text{Recreational} & \sqsubseteq \text{RecreationalCity} \\
\end{align*}
\]

\[
\begin{align*}
\text{K}_{\text{SeaSideCity}}(x) & \leftarrow \text{K}_{\text{PortCity}}(x), \text{not} \text{NonSeaSideCity}(x) \\
\text{K}_{\text{interestingCity}}(x) & \leftarrow \text{K}_{\text{RecreationalCity}}(x), \text{not} \text{RainyCity}(x) \\
\text{K}_{\text{hasOnSea}}(x) & \leftarrow \text{K}_{\text{OnSea}}(x, y) \\
\text{K}_{\text{false}} & \leftarrow \text{K}_{\text{SeaSideCity}}(x), \text{not} \text{hasOnSea}(x) \\
\text{K}_{\text{summerDestination}}(x) & \leftarrow \text{K}_{\text{interestingCity}}(x), \text{K}_{\text{OnSea}}(x, y) \\
\end{align*}
\]

Query: \text{K}_{\text{interestingCity}}(\text{Barcelona})
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

MKNF KB Example Query(2)

\[ \text{PortCity}(\text{Barcelona}) \quad \text{OnSea}(\text{Barcelona}, \text{Mediterranean}) \]
\[ \text{PortCity}(\text{Hamburg}) \quad \text{NonSeaSideCity}(\text{Hamburg}) \]
\[ \text{RainyCity}(\text{Manchester}) \quad \text{Has}(\text{Manchester}, \text{AquaticsCenter}) \]
\[ \text{Recreational}(\text{AquaticsCenter}) \]

\[ \text{SeaSideCity} \sqsubseteq \exists \text{Has}.\text{Beach} \]
\[ \text{Beach} \sqsubseteq \text{Recreational} \]
\[ \exists \text{Has}.\text{Recreational} \sqsubseteq \text{RecreationalCity} \]

\[ \text{K}_{\text{SeaSideCity}}(x) \leftarrow \text{K}_{\text{PortCity}}(x), \neg \text{NonSeaSideCity}(x) \]
\[ \text{K}_{\text{interestingCity}}(x) \leftarrow \text{K}_{\text{RecreationalCity}}(x), \neg \text{RainyCity}(x) \]
\[ \text{K}_{\text{hasOnSea}}(x) \leftarrow \text{K}_{\text{OnSea}}(x, y) \]
\[ \text{K}_{\text{false}} \leftarrow \text{K}_{\text{SeaSideCity}}(x), \neg \text{hasOnSea}(x) \]
\[ \text{K}_{\text{summerDestination}}(x) \leftarrow \text{K}_{\text{interestingCity}}(x), \text{K}_{\text{OnSea}}(x, y) \]

Subquery: \text{K}_{\text{RecreationalCity}}(\text{Barcelona})
\text{Has}(\text{Barcelona}, x), \text{Recreational}(x) \text{ would fail}
MKNF KB Example Query(3)

PortCity(Barcelona)  OnSea(Barcelona, Mediterranean)
PortCity(Hamburg)    NonSeaSideCity(Hamburg)
RainyCity(Manchester) Has(Manchester, AquaticsCenter)
Recreational(AquaticsCenter)

SeaSideCity ⊑ ∃Has. Beach
Beach ⊑ Recreational
∃Has. Recreational ⊑ RecreationalCity
K SeaSideCity(x) ← K PortCity(x), not NonSeaSideCity(x)
K interestingCity(x) ← K RecreationalCity(x), not RainyCity(x)
K hasOnSea(x) ← K OnSea(x, y)
K false ← K SeaSideCity(x), not hasOnSea(x)
K summerDestination(x) ← K interestingCity(x), K OnSea(x, y)

Subquery: K RecreationalCity(Barcelona)
MKNF KB Example Query(4)

\[
\begin{align*}
\text{PortCity}(\text{Barcelona}) & \quad \text{OnSea}(\text{Barcelona, Mediterranean}) \\
\text{PortCity}(\text{Hamburg}) & \quad \text{NonSeaSideCity}(\text{Hamburg}) \\
\text{RainyCity}(\text{Manchester}) & \quad \text{Has}(\text{Manchester, AquaticsCenter}) \\
\text{Recreational}(\text{AquaticsCenter}) &
\end{align*}
\]

\[
\begin{align*}
\text{SeaSideCity} & \subseteq \exists \text{Has. Beach} \\
\text{Beach} & \sqsubseteq \text{Recreational} \\
\exists \text{Has. Recreational} & \sqsubseteq \text{RecreationalCity} \\
K \text{SeaSideCity}(x) & \leftarrow K \text{PortCity}(x), \text{not} \text{NonSeaSideCity}(x) \\
K \text{interestingCity}(x) & \leftarrow K \text{RecreationalCity}(x), \text{not} \text{RainyCity}(x) \\
K \text{hasOnSea}(x) & \leftarrow K \text{OnSea}(x, y) \\
K \text{false} & \leftarrow K \text{SeaSideCity}(x), \text{not} \text{hasOnSea}(x) \\
K \text{summerDestination}(x) & \leftarrow K \text{interestingCity}(x), K \text{OnSea}(x, y)
\end{align*}
\]

**Subquery:** \(K \text{SeaSideCity}(\text{Barcelona})\)
MKNF KB Example Query(5)

PortCity(Barcelona)  OnSea(Barcelona, Mediterranean)
PortCity(Hamburg)  NonSeaSideCity(Hamburg)
RainyCity(Manchester)  Has(Manchester, AquaticsCenter)
Recreational(AquaticsCenter)

SeaSideCity ⊑ ∃Has.Beach
Beach ⊑ Recreational
∃Has.Recreational ⊑ RecreationalCity

K SeaSideCity(x) ← K PortCity(x), not NonSeaSideCity(x)
K interestingCity(x) ← K RecreationalCity(x), not RainyCity(x)
K hasOnSea(x) ← K OnSea(x, y)
K false ← K SeaSideCity(x), not hasOnSea(x)
K summerDestination(x) ← K interestingCity(x), K OnSea(x, y)

Subquery: K PortCity(Barcelona) succeeds
MKNF KB Example Query(6)

\[\text{PortCity}(\text{Barcelona}) \quad \text{OnSea}(\text{Barcelona}, \text{Mediterranean})\]
\[\text{PortCity}(\text{Hamburg}) \quad \text{NonSeaSideCity}(\text{Hamburg})\]
\[\text{RainyCity}(\text{Manchester}) \quad \text{Has}(\text{Manchester}, \text{AquaticsCenter})\]
\[\text{Recreational}(\text{AquaticsCenter})\]

\[\text{SeaSideCity} \sqsubseteq \exists \text{Has}. \text{Beach}\]
\[\text{Beach} \sqsubseteq \text{Recreational}\]
\[\exists \text{Has}. \text{Recreational} \sqsubseteq \text{RecreationalCity}\]

\[\text{K SeaSideCity}(x) \leftarrow \text{K PortCity}(x), \text{not} \text{NonSeaSideCity}(x)\]

\[\text{K interestingCity}(x) \leftarrow \text{K RecreationalCity}(x), \text{not} \text{RainyCity}(x)\]

\[\text{K hasOnSea}(x) \leftarrow \text{K OnSea}(x, y)\]

\[\text{K false} \leftarrow \text{K SeaSideCity}(x), \text{not} \text{hasOnSea}(x)\]

\[\text{K summerDestination}(x) \leftarrow \text{K interestingCity}(x), \text{K OnSea}(x, y)\]

**Subquery:** \[\text{K NonSeaSideCity}(\text{Barcelona}) \text{ fails} \]
MKNF KB Example Query(7)

\[
\text{PortCity}(\text{Barcelona}) \quad \text{OnSea}(\text{Barcelona}, \text{Mediterranean}) \\
\text{PortCity}(\text{Hamburg}) \quad \text{NonSeaSideCity}(\text{Hamburg}) \\
\text{RainyCity}(\text{Manchester}) \quad \text{Has}(\text{Manchester}, \text{AquaticsCenter}) \\
\text{Recreational}(\text{AquaticsCenter}) \\
\text{SeaSideCity} \sqsubseteq \exists \text{Has}. \text{Beach} \\
\text{Beach} \sqsubseteq \text{Recreational} \\
\exists \text{Has}. \text{Recreational} \sqsubseteq \text{RecreationalCity} \\
K \text{SeaSideCity}(x) \leftarrow K \text{PortCity}(x), \textbf{not} K \text{NonSeaSideCity}(x) \\
K \text{interestingCity}(x) \leftarrow K \text{RecreationalCity}(x), \textbf{not} K \text{RainyCity}(x) \\
K \text{hasOnSea}(x) \leftarrow K \text{OnSea}(x, y) \\
K \text{false} \leftarrow K \text{SeaSideCity}(x), \textbf{not} K \text{hasOnSea}(x) \\
K \text{summerDestination}(x) \leftarrow K \text{interestingCity}(x), K \text{OnSea}(x, y) \\
\textbf{not} K \text{NonSeaSideCity}(\text{Barcelona}) \text{ and } K \text{SeaSideCity}(\text{Barcelona}) \\
succeed and subsequently K \text{RecreationalCity}(\text{Barcelona})
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

MKNF KB Example Query(8)

\[
\begin{align*}
\text{PortCity}(Barcelona) & \quad \text{OnSea}(Barcelona, Mediterranean) \\
\text{PortCity}(Hamburg) & \quad \text{NonSeaSideCity}(Hamburg) \\
\text{RainyCity}(Manchester) & \quad \text{Has}(Manchester, AquaticsCenter) \\
\text{Recreational}(AquaticsCenter) & \\
\text{SeaSideCity} & \sqsubseteq \exists \text{Has.Beach} \\
\text{Beach} & \sqsubseteq \text{Recreational} \\
\exists \text{Has.Recreational} & \sqsubseteq \text{RecreationalCity} \\
\text{KSeaSideCity}(x) & \leftarrow \text{KPortCity}(x), \text{not NonSeaSideCity}(x) \\
\text{KinterestingCity}(x) & \leftarrow \text{KRecreationalCity}(x), \text{not RainyCity}(x) \\
\text{KhasOnSea}(x) & \leftarrow \text{KOnSea}(x, y) \\
\text{Kfalse} & \leftarrow \text{KSeaSideCity}(x), \text{not hasOnSea}(x) \\
\text{KsummerDestination}(x) & \leftarrow \text{KinterestingCity}(x), \text{KOnSea}(x, y) \\
\text{Subquery: KRainyCity}(Barcelona) & \text{fails}
\end{align*}
\]
**MKNF KB Example Query (9)**

- `PortCity(Barcelona)`
- `PortCity(Hamburg)`
- `RainyCity(Manchester)`
- `Recreational(AquaticsCenter)`

**DL Rules:**

- `SeaSideCity ⊑ ∃Has.Beach`
- `Beach ⊑ Recreational`
- `∃Has.Recreational ⊑ RecreationalCity`

**MSO Rule:**

- `KSeaSideCity(x) ← KPortCity(x), not NonSeaSideCity(x)`
- `KinterestingCity(x) ← KRecreationalCity(x), not RainyCity(x)`
- `K hasOnSea(x) ← K OnSea(x, y)`
- `K false ← KSeaSideCity(x), not hasOnSea(x)`
- `K summerDestination(x) ← K interestingCity(x), K OnSea(x, y)`

**Main query:**

- `K InterestingCity(Barcelona) succeeds`
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

MKNF KB Example Query(10)

- PortCity(Barcelona)  OnSea(Barcelona, Mediterranean)
- PortCity(Hamburg)   NonSeaSideCity(Hamburg)
- RainyCity(Manchester) Has(Manchester, AquaticsCenter)
- Recreational(AquaticsCenter)

- SeaSideCity ⊑ ∃Has.Beach
- Beach ⊑ Recreational
- ∃Has.Recreational ⊑ RecreationalCity

KSeaSideCity(x) ← KPortCity(x), not NonSeaSideCity(x)
KinterestingCity(x) ← KRecreationalCity(x), not RainyCity(x)
KhasOnSea(x) ← KOnSea(x, y)
Kfalse ← KSeaSideCity(x), not hasOnSea(x)
KsummerDestination(x) ← KinterestingCity(x), KOnSea(x, y)

KinterestingCity(Barcelona) is stored
subsequently two look-ups yield summerDestination(Barcelona)
General Properties of $\text{SLG}(O)$

- Answers match the well-founded MKNF model if it exists.
- Otherwise, for consistent $O$, answers match a paraconsistent approximation.
- Data complexity $P^C$ maintained ($C$ complexity of instance checking in $O$) provided the number of answers returned from the oracle is polynomial.
- Three polynomial oracles, one for each OWL 2 profile.
Reasoner NoHR for OWL 2 EL

- first non-monotonic plug-in for the ontology editor Protégé
- presented in [Vadim et al., ISWC13]
- unlike OWL reasoners (such as ELK, Fact, Hermit, Racer) that focus on DL reasoning tasks, realizes top-down querying on ontologies and non-monotonic rules
- answers (safe) conjunctive queries, returning answer tuples for non-ground queries or true/undefined/false/inconsistent
- https://code.google.com/p/nohr-reasoner/
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

NoHR Protégé plug-in
Pre-processing Algorithm

- Compute implicit inferences using ELK reasoner
- Discard axioms/information not usable when querying, such as
  \[ \text{SeaSideCity} \sqsubseteq \exists \text{Has.Beach} \]
- Translate the remainder into rules and merge with non-monotonic rules (applying to both a special doubling in case DisjointWith axioms occur in \( \mathcal{O} \), to detect potential inconsistencies)
- Input the result to XSB Prolog
Test Translation ontologies

<table>
<thead>
<tr>
<th></th>
<th># Axioms</th>
<th>ELK</th>
<th>Translator</th>
<th>XSB</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChEBI</td>
<td>67184</td>
<td>3,92</td>
<td>7,41</td>
<td>1,75</td>
<td>9,16</td>
</tr>
<tr>
<td>EMAP</td>
<td>13730</td>
<td>2,94</td>
<td>5,14</td>
<td>0,00</td>
<td>5,14</td>
</tr>
<tr>
<td>Fly Anatomy</td>
<td>19211</td>
<td>2,70</td>
<td>4,99</td>
<td>0,83</td>
<td>5,82</td>
</tr>
<tr>
<td>FMA</td>
<td>126548</td>
<td>7,56</td>
<td>14,67</td>
<td>3,58</td>
<td>18,25</td>
</tr>
<tr>
<td>GALEN-OWL</td>
<td>36547</td>
<td>5,00</td>
<td>8,76</td>
<td>2,46</td>
<td>11,22</td>
</tr>
<tr>
<td>GALEN7</td>
<td>44461</td>
<td>5,88</td>
<td>9,67</td>
<td>4,56</td>
<td>14,23</td>
</tr>
<tr>
<td>GALEN8</td>
<td>73590</td>
<td>21,90</td>
<td>42,76</td>
<td>28,12</td>
<td>70,88</td>
</tr>
<tr>
<td>GO1</td>
<td>28897</td>
<td>3,74</td>
<td>6,06</td>
<td>1,09</td>
<td>7,15</td>
</tr>
<tr>
<td>GO2</td>
<td>73590</td>
<td>4,57</td>
<td>8,90</td>
<td>5,02</td>
<td>13,92</td>
</tr>
<tr>
<td>Molecule Role</td>
<td>9629</td>
<td>3,41</td>
<td>5,14</td>
<td>0,15</td>
<td>5,29</td>
</tr>
<tr>
<td>SNOMED CT</td>
<td>294480</td>
<td>20,61</td>
<td>43,32</td>
<td>24,69</td>
<td>68,01</td>
</tr>
</tbody>
</table>
Test Translation SNOMED CT with Rules

![Graph showing the time (s) vs. NM Rules + Facts (×1000) for different systems: ELK, Translator, and XSB. The graph illustrates the performance of each system as the number of rules and facts increases.]

- **ELK**
- **Translator**
- **XSB**
Query Time

- Queries of varying size, containing atoms of different depth in the ontology hierarchy with varying number of specialized classes, and differing connectivity between variables posed to real ontologies with added rules.
- Interactive response time usually considerably below a second; only higher if the number of answers (e.g., due to many subclasses of a queried atom) grows very high, or arbitrarily generated answers create too links, essentially computing the entire model in the worst case.
Summary

• Top-down querying over ontologies and non-monotonic rules (under well-founded MKNF) can be efficiently implemented
• NoHR for OWL 2 EL under active development
• current challenge: realize use-cases, develop KBs
• try it out at: https://code.google.com/p/nohr-reasoner/
Generalized unifying language
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

Non-monotonic DL Extension with MKNF

\(\text{ALCK}_{\text{NF}}\) [Donini et al., ACM TOCL02]

\(\text{ALC}\) with MKNF logic-style modal operators \(\mathbf{K}\) – minimal knowledge – and \(\mathbf{A}\) – autoepistemic assumption (corresponds to \(\neg\text{not}\))

\(\mathbf{K}\) can be used to derive new information, \(\mathbf{A}\) to verify if information is already known

Different expressiveness compared to Hybrid MKNF
Non-monotonic Features of $\mathcal{ALCK}_{NF}$

from [Donini et al., ACM TOCL02]

• Defaults:

$$KI \sqcap K(employee \sqcap \exists \text{belongsTo.programmingDept}) \sqcap \neg A\text{manager} \sqsubseteq K(\text{engineer} \sqcup \text{mathematican})$$

• Integrity Constraints:

$$K\text{employee} \sqsubseteq (A\text{male} \sqcup A\text{female})$$

$$K\text{employee} \sqsubseteq \exists A\text{SSN.Avalid}$$
Non-monotonic Features of $\text{ALCK}_{N,F}$

- Concept and Role Closure

\[ \neg \text{UScitizen}(Paula) \rightarrow \text{Manages}(Ann, Marc) \]
\[ \neg \text{UScitizen}(Carl) \rightarrow \text{UScitizen}(Marc) \]

Adding $\forall K \text{Manages}. \neg \text{UScitizen}(Ann)$ closes the role.

Adding $\exists K \text{Manages}. \neg \text{UScitizen}(Ann)$ closes the concept.
Can We find a joint formalism for both MKNF extensions?

- Contribute towards a unifying logic
- Reconcile OWL and Datalog together with CWA extensions (on both sides)
- Usage of one (DL-style) syntax in opposite to common hybrid languages
- Coverage of many different previous approaches
Combining non-monotonic rules and DLs
Top-down querying
Generalized unifying language

\[ SROIQV(B^s, \times)K_{NF} \]

- OWL 2 DL (\( SROIQ \)) with concept products (\( \times \) - [Krötzsch, SSW10]) and Boolean constructors over simple roles (\( B^s \) - [Rudolph et al., JELIA08])
- Nominal schemas (\( V \) - [Krötzsch et al., WWW11]) - variable nominals that can only bind to known individuals
- MKNF logic-style modal operators \( K \) – minimal knowledge – and \( A \) – autoepistemic assumption – (\( K_{NF} \) - from \( ALCK_{NF} \) [Donini et al., ACM TOCL02])
- \( SROIQVK \) for short
signature $\Sigma = \langle N_I, N_C, N_R, N_V \rangle$

**Definition**
The set of $SROIQV(B^s, \times)K_{\mathcal{NF}}$ concepts $C$ and (simple/non-simple) $SROIQV(B^s, \times)K_{\mathcal{NF}}$ roles $R$ ($R^s/R^n$) are defined by the following grammar.

$$
R^s ::= N^s_R \mid (N^s_R)^- \mid U \mid N_C \times N_C \mid \neg R^s \mid R^s \sqcap R^s \mid R^s \sqcup R^s \mid KR^s \mid AR^s
$$

$$
R^n ::= N^n_R \mid (N^n_R)^- \mid U \mid N_C \times N_C \mid KR^n \mid AR^n
$$

$$
R ::= R^s \mid R^n
$$

$$
C ::= \top \mid \bot \mid N_C \mid \{N_I\} \mid \{N_V\} \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists R.C \mid \forall R.C \mid \exists R^s.Self \mid \leq_k R^s.C \mid \geq_k R^s.C \mid KC \mid AC
$$
Semantics – Principal Notions

- Based on interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ plus variable assignments (for nominal variables) mapping each variable to the interpretation of one element in $\mathcal{N}_I$

- Variant of *Standard Name Assumption* applied: essentially $\mathcal{I}$ is a bijective function on $\mathcal{N}_I$ while still allowing that elements of $\mathcal{N}_I$ may be identified ($\rightarrow$ only one $\Delta$)

An *MKNF structure* is a triple $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ where $\mathcal{I}$ is an interpretation, $\mathcal{M}$ and $\mathcal{N}$ are sets of interpretations, and $\mathcal{I}$ and all interpretations in $\mathcal{M}$ and $\mathcal{N}$ are defined over $\Delta$. For any such $(\mathcal{I}, \mathcal{M}, \mathcal{N})$ and assignment $\mathcal{Z}$, the function $(\cdot)(\mathcal{I}, \mathcal{M}, \mathcal{N}, \mathcal{Z})$ is defined.
### Function \( (\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z} \) (parts of it)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(a^\mathcal{I} \in \Delta)</td>
</tr>
<tr>
<td>(x)</td>
<td>(\mathcal{Z}(x) \in \Delta)</td>
</tr>
<tr>
<td>(\neg C)</td>
<td>(\Delta \setminus C^{(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
<tr>
<td>({t})</td>
<td>({a \mid a \approx t^{(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z}}})</td>
</tr>
<tr>
<td>(KC)</td>
<td>(\bigcap_{\mathcal{J} \in \mathcal{M}} C^{(\mathcal{J}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
<tr>
<td>(AC)</td>
<td>(\bigcap_{\mathcal{J} \in \mathcal{N}} C^{(\mathcal{J}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
<tr>
<td>(KR)</td>
<td>(\bigcap_{\mathcal{J} \in \mathcal{M}} R^{(\mathcal{J}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
<tr>
<td>(AR)</td>
<td>(\bigcap_{\mathcal{J} \in \mathcal{N}} R^{(\mathcal{J}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
<tr>
<td>(C \subseteq D)</td>
<td>(C^{(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z}} \subseteq D^{(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z}})</td>
</tr>
</tbody>
</table>
(Monotonic) Semantics

Definition

$(\mathcal{I}, \mathcal{M}, \mathcal{N})$ satisfies axiom $\alpha$, written $(\mathcal{I}, \mathcal{M}, \mathcal{N}) \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{N}), \mathcal{Z} \models \alpha$ for all variable assignments $\mathcal{Z}$.

A (non-empty) set of interpretations $\mathcal{M}$ satisfies $\alpha$, written $\mathcal{M} \models \alpha$, if $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \alpha$ holds for all $\mathcal{I} \in \mathcal{M}$.

$\mathcal{M}$ satisfies a SROIQV($\mathcal{B}^s$, $\times$)$\mathcal{K}_{\mathcal{NF}}$ knowledge base $KB$, written $\mathcal{M} \models KB$, if $\mathcal{M} \models \alpha$ for all axioms $\alpha \in KB$. 

(Non-monotonic) Semantics

Definition
Given a \( SROIQV(B^s, \times)K_NF \) knowledge base \( KB \), a (non-empty) set of interpretations \( \mathcal{M} \) is an \textit{MKNF model} of \( KB \) if

1. \( \mathcal{M} \models KB \), and
2. for each \( \mathcal{M}' \) with \( \mathcal{M} \subseteq \mathcal{M}' \), \( (\mathcal{I}', \mathcal{M}', \mathcal{M}) \nvdash KB \) for some \( \mathcal{I}' \in \mathcal{M}' \).
Example

\( C \) Persons whose parents are married

\[
\text{HasParent}(\text{mary}, \text{john}) \quad (1)
\]
\[
(\exists \text{HasParent}. \exists \text{Married}. \{\text{john}\})(\text{mary}) \quad (2)
\]
\[
\exists \text{HasParent}. \{z\} \cap \exists \text{HasParent}. \exists \text{Married}. \{z\} \subseteq C \quad (3)
\]
Example

$C$ Persons whose parents are married

(1) $\text{HasParent}(\text{mary}, \text{john})$

(2) $(\exists \text{HasParent} . \exists \text{Married}. \{\text{john}\})(\text{mary})$

(3) $\exists \text{HasParent}. \{z\} \sqcap \exists \text{HasParent}. \exists \text{Married}. \{z\} \subseteq C$

We can substitute (3) by

(4) $K \exists \text{HasParent}. \{z\} \sqcap K \exists \text{HasParent}. \exists \text{Married}. \{z\} \subseteq KC$
Example

\( C \) Persons whose parents are married

\[
\begin{align*}
\text{HasParent}(\text{mary}, \text{john}) & \quad (1) \\
(\exists \text{HasParent} \land \exists \text{Married}.\{\text{john}\})(\text{mary}) & \quad (2) \\
\exists \text{HasParent}.\{\text{z}\} \sqcap \exists \text{HasParent} \land \exists \text{Married}.\{\text{z}\} & \subseteq C \quad (3)
\end{align*}
\]

We can also substitute (3) by

\[
K \exists \text{HasParent}.\{\text{z}\} \sqcap K \exists \text{HasParent} \land \exists \text{Married}.\{\text{z}\} \subseteq AC
\]

We now require that all elements of class \( C \) are explicitly mentioned
Example

$C$ Persons whose parents are married

1. $\exists \text{HasParent}(\text{mary, john})$  
2. $\left( \exists \text{HasParent.} \exists \text{Married.}\{\text{john}\} \right)(\text{mary})$  
3. $\exists \text{HasParent.}\{z\} \sqcap \exists \text{HasParent.}\exists \text{Married.}\{z\} \subseteq C$

We can also substitute (3) by

$\exists \text{HasParent.}\{z\} \sqcap \exists \text{HasParent.} \exists \neg \text{Married.}\{z\} \subseteq C$

Now $C$ are Persons that are known to be not married
Decidability

- First, reduce reasoning in $SROIQV(B^s, \times)K_{NF}$ to reasoning in $SROIQ(B^s)K_{NF}$ by grounding and by simulating concept products
- Then, follow approach for $ALCK_{NK}$:
  - each model of a knowledge base in $SROIQ(B^s)K_{NF}$ is cast into a $SROIQ(B^s)$ KB. Consequently, reasoning in $SROIQ(B^s)K_{NF}$ is reduced to a number of reasoning tasks in the non-modal $SROIQ(B^s)$
  - For simplicity, appearance of modal operators restricted to simple KBs as in $ALCK_{NF}$ (finitely many, finite representations of models)
(Monotonic) Coverage

- \textit{SROIQ} (a.k.a. OWL 2 DL);
- The tractable profiles OWL 2 EL, OWL 2 RL, OWL 2 QL;
- RIF-Core, i.e., \( n \)-ary Datalog, interpreted as DL-safe Rules (general case new result in [Knorr et al., ECAI12]);
- DL-safe SWRL [Motik et al., JWS05], \( \mathcal{AL} \)-log [Donini et al., JIIS98], and CARIN [Levy and Rousset, AI98].
(Non-monotonic) Coverage

- $\mathcal{ALCK}_{N,F}$ [Donini et al., ACM TOCL02]; includes notions of concept and role closure present in this formalism;
- Closed Reiter defaults covered through the coverage of $\mathcal{ALCK}_{N,F}$; includes coverage of DLs extended with default rules [Baader and Hollunder, JAR95];
- Hybrid MKNF [Motik and Rosati, JACM10];
- Answer Set Programming, i.e., disjunctive Datalog with classical negation and non-monotonic negation under the answer set semantics; follows from the coverage of Hybrid MKNF.
\( N \)-nary Datalog

\[ N_R = N_{P,2} \cup \{U\} \cup S, \text{ where } S \text{ is a special set of roles: If } P \in N_{P,>2} \text{ has arity } k, \text{ then } P_1, \ldots, P_k \in S \text{ are unique binary predicates associated with } P; \]

Translation: \( \text{dl}(P(t_1, \ldots, t_k)) := \exists U. (\exists P_1. \{t_1\} \cap \ldots \cap \exists P_k. \{t_k\}); \)

Family of interpretations of \( \mathcal{J} \) for interpretation \( \mathcal{I} \) of Datalog RB:

(a) To each \((d_1, \ldots, d_k) \in P^\mathcal{I}\), assign a unique element \(e\) in \(\Delta\) (i.e., we define a total, injective function from the set of tuples to \(\Delta\)).

(d) For each \(P \in N_{P,>2}\), if \((d_1, \ldots, d_k) \in P^\mathcal{I}\), then \((e, d_i) \in P_i^\mathcal{J}\), where \(e\) is the element assigned to \((d_1, \ldots, d_k)\) in point (a).
Hybrid MKNF

Seamless integration of DL ontology $\mathcal{O}$ and rules of the form

$$KH_1 \lor KH_l \leftarrow KA_1, \ldots, KA_n, \text{not} B_1, \ldots, \text{not} B_m$$

Based on the $n$-nary Datalog embedding, additionally:

$$\text{dl}(KH_1 \lor KH_l \leftarrow KA_1, \ldots, KA_n, \text{not} B_1, \ldots, \text{not} B_m) :=$$

$$K\text{dl}(A_1) \sqcap \cdots \sqcap K\text{dl}(A_n) \sqcap \neg A\text{dl}(B_1) \sqcap \cdots \sqcap \neg A\text{dl}(B_m) \sqsubseteq K\text{dl}(H_1) \sqcup \cdots \sqcup K\text{dl}(H_l)$$
Example

\[ KC(x) \leftarrow K\text{HasParent}(x, y), K\text{HasParent}(x, z), K(y \not\approx z), \text{notMarried}(y, z). \]

can be translated into

\[ K\exists U.(\{x\} \cap \exists \text{HasParent}.\{y\}) \cap K\exists U.(\{x\} \cap \exists \text{HasParent}.\{z\}) \cap \neg A\exists U.(\{y\} \cap \exists \text{Married}.\{z\}) \sqsubseteq K\exists U.(\{x\} \cap C) \]
Summary

• Very expressive and general language with an advanced semantics
• Large coverage makes it an interesting candidate for a unifying logic
• challenges
  • Improve reasoning algorithms
  • Find non-trivial fragments within the full language
The End

Thank you for your attention!
References (1)

• Description logics of minimal knowledge and negation as failure. Francesco M. Donini, Daniele Nardi, and Riccardo Rosati, ACM TOCL, 3(2), 2002, 227–252.


• Local closed world reasoning with description logics under the well-founded semantics. Matthias Knorr, José J. Alferes, and Pascal Hitzler, Artificial Intelligence, 175(9-10), 2011, 1528–1544.
References(2)

• A query tool for EL with non-monotonic rules. Vadim Ivanov, Matthias Knorr, and João Leite, ISWC 13, to appear.