QUERY ANSWERING WITH DESCRIPTION LOGIC ONTOLOGIES

Meghyn Bienvenu (*CNRS & Université de Montpellier*)
Magdalena Ortiz (*Vienna University of Technology*)

QUERIES WITH NEGATION OR OTHER FORMS OF RECURSION

Conjunctive query with safe negation ($CQ^{\neg s}$):

- like a CQ, but can also have negated atoms
- safety condition: every variable occurs in some positive atom
- example: find menus whose main course is not spicy

$$\exists y\, \mathrm{Menu}(x) \wedge \mathrm{hasMain}(x, y) \wedge \neg \mathrm{Spicy}(y)$$

Conjunctive query with safe negation ($CQ^{\neg s}$):

- like a CQ, but **can also have negated atoms**
- **safety condition**: **every variable occurs in some positive atom**
- example: find menus whose main course is not spicy

$$\exists y\, \mathsf{Menu}(x) \wedge \mathsf{hasMain}(x, y) \wedge \neg \mathsf{Spicy}(y)$$

Conjunctive query with inequalities ($CQ^{\neq}$)

- like a CQ, but **can also have atoms** $t_1 \neq t_2$ (t_1, t_2 vars or individuals)
- example: find restaurant offering two menus having different dessert courses

$$\begin{aligned} \exists y_1 y_2 z_1 z_2 \quad & \mathsf{offers}(x, y_1) \wedge \mathsf{Menu}(y_1) \wedge \mathsf{hasDessert}(y_1, z_1) \wedge \\ & \mathsf{offers}(x, y_2) \wedge \mathsf{Menu}(y_2) \wedge \mathsf{hasDessert}(y_2, z_2) \wedge z_1 \neq z_2 \end{aligned}$$

- example: find menus with at least three courses

Conjunctive query with safe negation ($\mathsf{CQ}^{\neg s}$):

- like a CQ, but can also have negated atoms
- safety condition: every variable occurs in some positive atom
- example: find menus whose main course is not spicy

$$\exists y\, \mathsf{Menu}(x) \land \mathsf{hasMain}(x, y) \land \neg \mathsf{Spicy}(y)$$

Conjunctive query with inequalities ($\mathsf{CQ}^{\neq}$)

- like a CQ, but can also have atoms $t_1 \neq t_2$ (t_1, t_2 vars or individuals)
- example: find restaurant offering two menus having different dessert courses

$$\begin{aligned} \exists y_1 y_2 z_1 z_2 \quad & \mathsf{offers}(x, y_1) \land \mathsf{Menu}(y_1) \land \mathsf{hasDessert}(y_1, z_1) \land \\ & \mathsf{offers}(x, y_2) \land \mathsf{Menu}(y_2) \land \mathsf{hasDessert}(y_2, z_2) \land z_1 \neq z_2 \end{aligned}$$

- example: find menus with at least three courses

Note: can define $\mathsf{UCQ}^{\neg s}$s and $\mathsf{UCQ}^{\neq}$s in the obvious way

Adding negation leads to undecidability even in very restricted settings

Theorem The following problems are undecidable:

- $\mathrm{CQ}^{\neg s}$ answering in DL-Lite$_R$
- $\mathrm{UCQ}^{\neg s}$ answering in $\mathcal{EL}$
- $\mathrm{CQ}^{\neq}$ answering in DL-Lite$_R$
- $\mathrm{CQ}^{\neq}$ answering in $\mathcal{EL}$

Unbounded tiling problem: $(\mathbf{T}, H, V, T_0)$

- $\mathbf{T}$ is a set of **tile types**
- $V \subseteq \mathbf{T} \times \mathbf{T}$ and $H \subseteq \mathbf{T} \times \mathbf{T}$ are the **vertical** and **horizontal** compatibility relations
- $T_0 \in \mathbf{T}$ is the **initial tile type**

We want to **tile an $\mathbb{N} \times \mathbb{N}$ corridor**

- T_0 must be placed in bottom-left corner
- Neighboring tiles must respect V and H

An **unbounded tiling** simulates a (non-det.) **Turing Machine**

We only need the folowing TBox, to generate points of the grid:

$$\mathcal{T} = \{Point \sqsubseteq \exists h, Point \sqsubseteq \exists v, \exists h^- \sqsubseteq Point, \exists v^- \sqsubseteq Point\}$$

Check for errors in the grid or tiling, using UCQ q with disjuncts:

$$\exists x.Point(\mathsf{x}) \wedge \neg \mathsf{T}_1(\mathsf{x}) \cdots \wedge \neg \mathsf{T}_n(\mathsf{x})$$

$$\exists x.\mathsf{T}_i(\mathsf{x}) \wedge \mathsf{T}_j(\mathsf{x}) \qquad i \neq j, i, j \in \{1, \cdots, n\}$$

$$\exists x, y.\mathsf{T}_i(\mathsf{x}) \wedge \mathsf{h}(\mathsf{x}, \mathsf{y}) \wedge \bigwedge_{(\mathsf{T}_i, \mathsf{T}_j) \notin \mathsf{H}} \neg \mathsf{T}_j(\mathsf{y}) \qquad i \in \{1, \cdots, n\}$$

$$\exists x, y.\mathsf{T}_i(\mathsf{x}) \wedge \mathsf{v}(\mathsf{x}, \mathsf{y}) \wedge \bigwedge_{(\mathsf{T}_i, \mathsf{T}_j) \notin \mathsf{V}} \neg \mathsf{T}_j(\mathsf{y}) \qquad i \in \{1, \cdots, n\}$$

$$\exists x_1, x_2, y_1, y_2.\mathsf{h}(\mathsf{x}_1, \mathsf{x}_2) \wedge \mathsf{v}(\mathsf{x}_1, \mathsf{y}_1) \wedge \mathsf{h}(\mathsf{y}_1, \mathsf{y}_2) \wedge \neg \mathsf{v}(\mathsf{x}_2, \mathsf{y}_2)$$

$\mathcal{T}, \{A(c)\} \not\models q$ iff (T, H, V, t_0) has a solution

Adding negation leads to undecidability even in very restricted settings

Theorem The following problems are undecidable:

- $\mathrm{CQ}^{\neg s}$ answering in DL-Lite$_R$
- $\mathrm{UCQ}^{\neg s}$ answering in $\mathcal{EL}$
- $\mathrm{CQ}^{\neq}$ answering in DL-Lite$_R$
- $\mathrm{CQ}^{\neq}$ answering in $\mathcal{EL}$

Adding negation leads to undecidability even in very restricted settings

Theorem The following problems are undecidable:

- $\mathrm{CQ}^{\neg s}$ answering in DL-Lite$_R$
- $\mathrm{UCQ}^{\neg s}$ answering in $\mathcal{EL}$
- $\mathrm{CQ}^{\neq}$ answering in DL-Lite$_R$
- $\mathrm{CQ}^{\neq}$ answering in $\mathcal{EL}$

Possible solution: adopt alternative semantics (epistemic negation)

Significant interest in combining DLs with Datalog rules, already in the late 90s

Unfortunately, this almost always leads to undecidability:

Theorem Datalog query answering is undecidable in every DL that can express (directly or indirectly) $A \sqsubseteq \exists r.A$

In particular: undecidable in both DL-Lite and $\mathcal{EL}$

UNDECIDABILITY RESULTS FOR RECURSIVE DATALOG QUERIES

Significant interest in combining DLs with Datalog rules, already in the late 90s

Unfortunately, this almost always leads to undecidability:

Theorem Datalog query answering is undecidable in every DL that can express (directly or indirectly) $A \sqsubseteq \exists r.A$

In particular: undecidable in both DL-Lite and $\mathcal{EL}$

Possible solutions:

- use restricted classes of Datalog queries (e.g. path queries)
- DL-safe rules: can only apply rules to (named) individuals