Querying the Fragments of English

Camilo Thorne[1]

[1] KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano
cthorne@inf.unibz.it
http://www.inf.unibz.it/∼cathorne

WoLLIC 2011 - 19/5/2011

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

- Such a system would require, possibly, a deep semantic analysis of English

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

- Such a system would require, possibly, a deep semantic analysis of English

- Semantic processing would translate it into a set of (internal) semantic representations and then reason over them

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

- Such a system would require, possibly, a deep semantic analysis of English
- Semantic processing would translate it into a set of (internal) semantic representations and then reason over them
- But, natural language is ambigous:
  - ambiguity blows up processing (exponentially many semantic representations)
  - in the worst case, it is not computationally solvable

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

- Such a system would require, possibly, a deep semantic analysis of English
- Semantic processing would translate it into a set of (internal) semantic representations and then reason over them
- But, natural language is ambigous:
  - ambiguity blows up processing (exponentially many semantic representations)
  - in the worst case, it is not computationally solvable

QS: What if we remove ambiguity?

Motivation

- Suppose we want a computer system which, given as input the English sentences

  Every Italian loves pasta and football
  Silvio is Italian

  returns as output the sentence

  Silvio loves pasta

- Such a system would require, possibly, a deep semantic analysis of English
- Semantic processing would translate it into a set of (internal) semantic representations and then reason over them
- But, natural language is ambigous:
  - ambiguity blows up processing (exponentially many semantic representations)
  - in the worst case, it is not computationally solvable

$\Rightarrow$ Use controlled fragments!

Controlled Fragments

- A controlled fragment is an ambiguity-free subset of a natural language (e.g., English) that
  - polynomially translates into logic formulas φ called meaning representations
  - the translation can be modelled by formal semantics compositional translations $\tau(\cdot)$
- We can use the standard machinery of Montague semantics and type theory [Montague 1970]
- Front-ends based on these languages have been proposed to address the ambiguity problem, by trading off expressiveness [Sowa 2004, Fuchs et al. 2006]
- They are defined by constraining the syntax, semantics and vocabulary of natural languages

Controlled Fragments

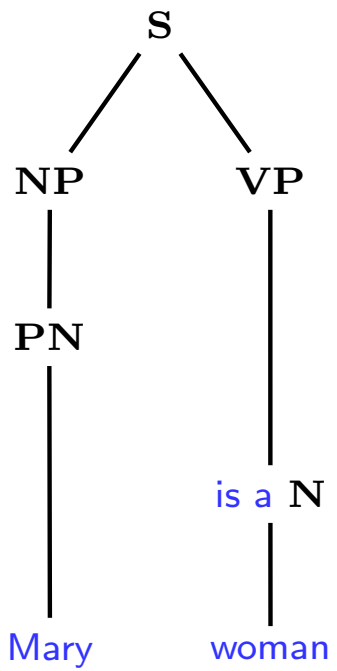
- A controlled fragment is an ambiguity-free subset of a natural language (e.g., English) that
  - polynomially translates into logic formulas φ called meaning representations
  - the translation can be modelled by formal semantics compositional translations $\tau(\cdot)$
- We can use the standard machinery of Montague semantics and type theory [Montague 1970]
- Front-ends based on these languages have been proposed to address the ambiguity problem, by trading off expressiveness [Sowa 2004, Fuchs et al. 2006]
- They are defined by constraining the syntax, semantics and vocabulary of natural languages
- Being ambiguity-free entails efficient translation, but less is known about how costly it is to logically reason with their meaning representations

Controlled Fragments - COP [Pratt & Third 2006]

Syntax Rules	Semantics $(= \tau(\cdot))$
$\mathbf{S} \longrightarrow \mathbf{NP\ VP}$	$\tau(\mathbf{NP})(\tau(\mathbf{VP})) \rhd \tau(\mathbf{S})$
$\mathbf{VP} \longrightarrow$ is a $\mathbf{N}$	$\tau(\mathbf{VP}) = \tau(\mathbf{N})$
$\mathbf{VP} \longrightarrow$ is not a $\mathbf{N}$	$\tau(\mathbf{VP}) = \neg\tau(\mathbf{N})$
$\mathbf{NP} \longrightarrow \mathbf{PN}$	$\tau(\mathbf{NP}) = \tau(\mathbf{PN})$
$\mathbf{NP} \longrightarrow \mathbf{Det\ N}$	$\tau(\mathbf{Det})(\tau(\mathbf{N})) \rhd \tau(\mathbf{NP})$

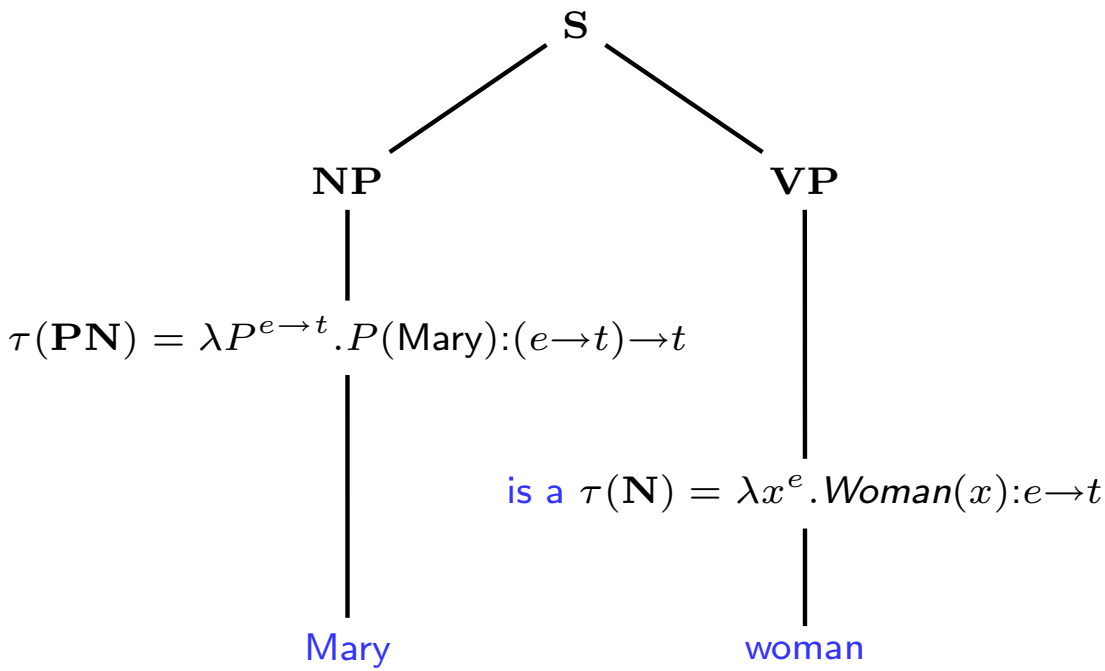
Lexicon	Semantics $(= \tau(\cdot))$
$\mathbf{N} \longrightarrow$ woman	$\tau(\mathbf{N}) = \lambda x^{e}.\mathit{Woman}(x){:}e{\to}t$
$\mathbf{N} \longrightarrow$ man	$\tau(\mathbf{N}) = \lambda x^{e}.\mathit{Man}(x){:}e{\to}t$
$\mathbf{PN} \longrightarrow$ Mary	$\tau(\mathbf{PN}) = \lambda P^{e\to t}.P(\mathsf{Mary}){:}(e{\to}t){\to}t$
$\mathbf{Det} \longrightarrow$ every	$\tau(\mathbf{Det}) = \lambda P^{e\to t}.\lambda Q^{e\to t}.\forall x^{e}(P(x) \Rightarrow Q(x)){:}(e{\to}t){\to}((e{\to}t){\to}t)$
$\mathbf{Det} \longrightarrow$ some	$\tau(\mathbf{Det}) = \lambda P^{e\to t}.\lambda Q^{e\to t}.\exists x^{e}(P(x) \wedge Q(x)){:}(e{\to}t){\to}((e{\to}t){\to}t)$
$\mathbf{Det} \longrightarrow$ no	$\tau(\mathbf{Det}) = \lambda P^{e\to t}.\lambda Q^{e\to t}.\forall x^{e}(P(x) \Rightarrow \neg Q(x)){:}(e{\to}t){\to}((e{\to}t){\to}t)$

Controlled Fragments - COP (cntd.)



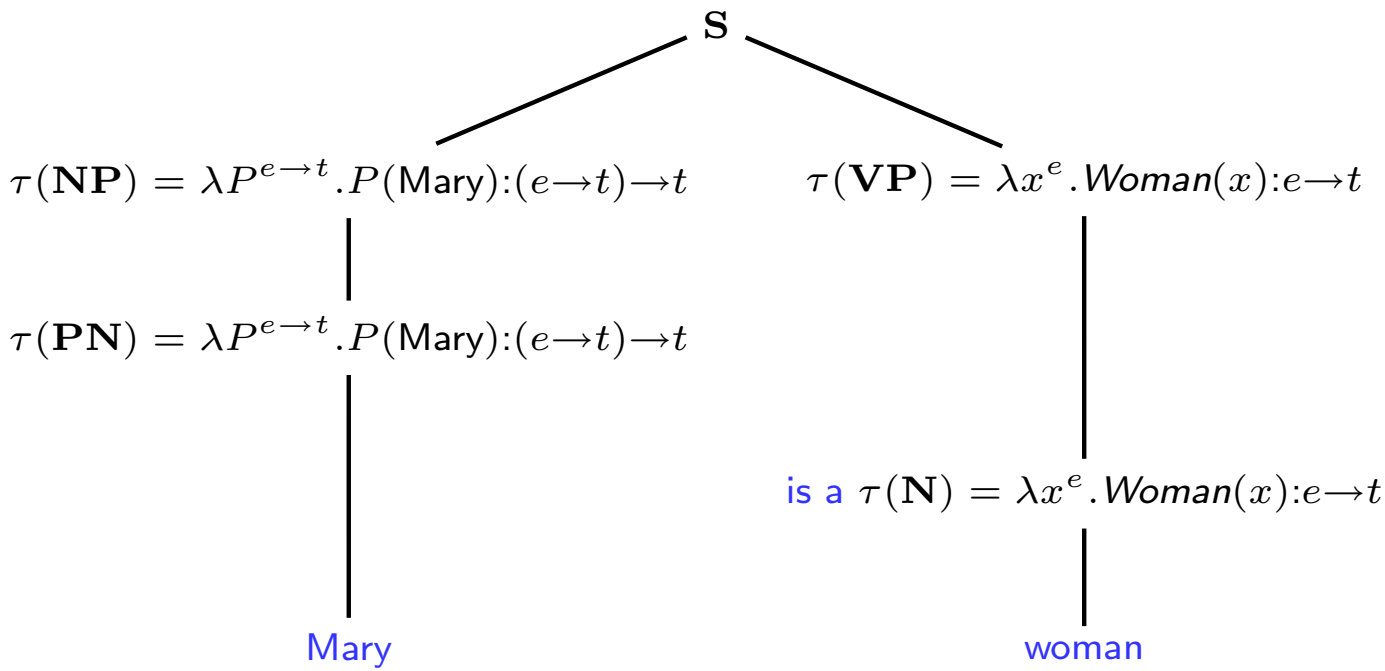
Parsing and interpreting "Mary is a woman"

Controlled Fragments - COP (cntd.)



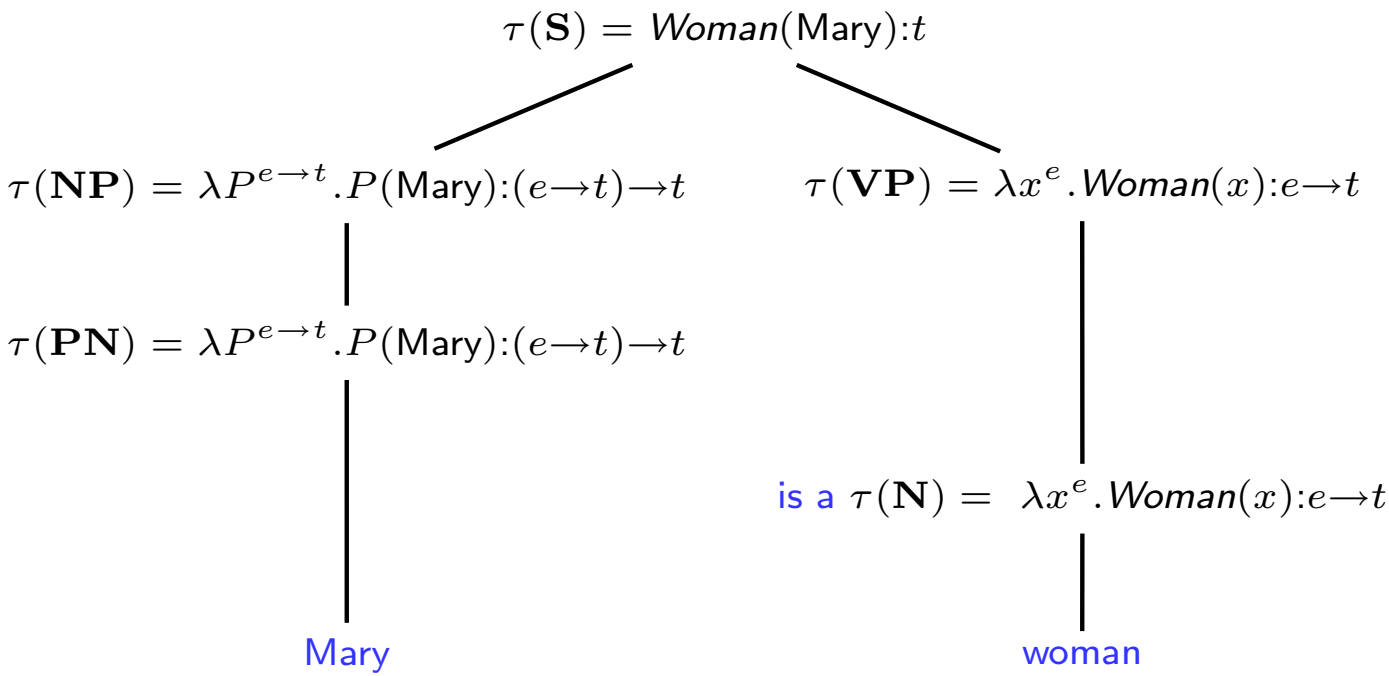
Parsing and interpreting "Mary is a woman"

Controlled Fragments - COP (cntd.)



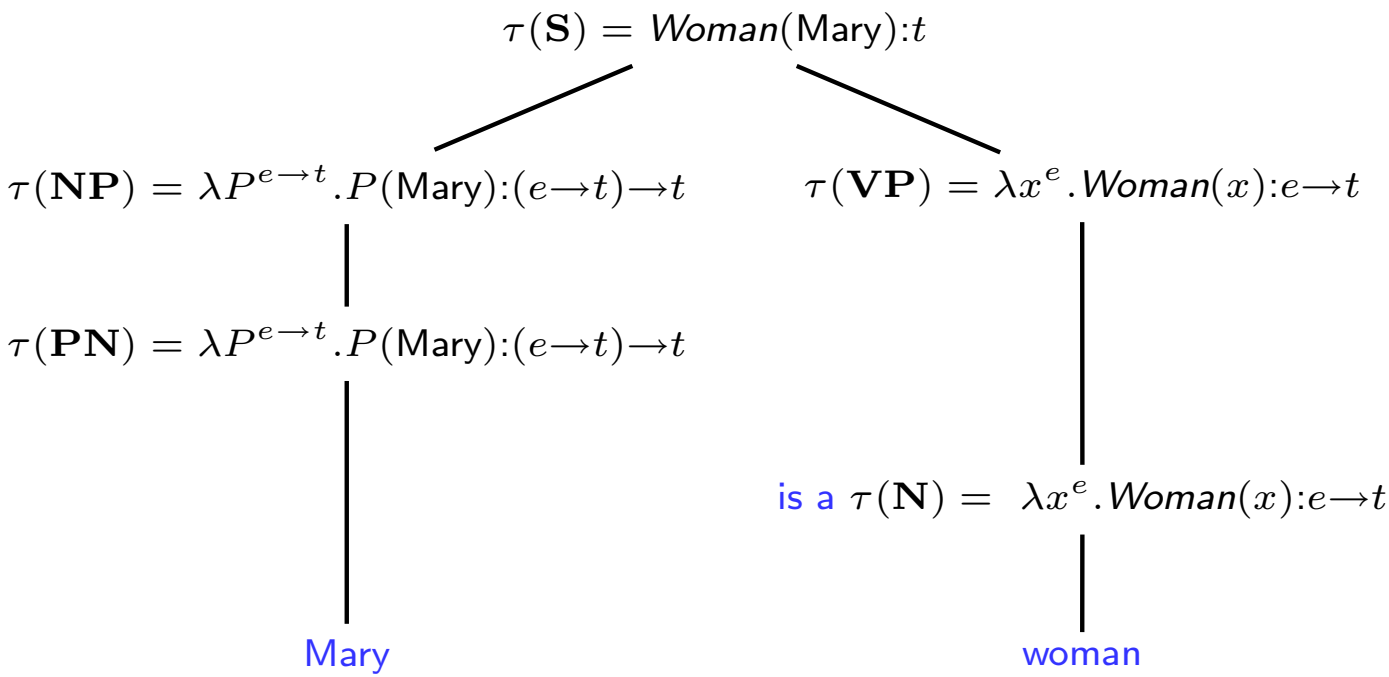
Parsing and interpreting "Mary is a woman"

Controlled Fragments - COP (cntd.)



Parsing and interpreting "Mary is a woman"

Controlled Fragments - COP (cntd.)



Parsing and interpreting "Mary is a woman"

NB: Computing $\tau(\cdot)$ is "easy" for context-free fragments of, e.g., English:

- parsing: polynomial time in the size of the input utterance
- interpreting: linear in the size of the parse tree

The Fragments of English (FOEs) [Pratt & Third 2006]

COP	Copula, common and proper nouns, negation, universal, existential quantifiers
COP+Rel	COP plus relative pronouns
COP+TV	COP plus transitive verbs
COP+TV+DTV	COP+TV plus ditransitive verbs
COP+Rel+TV	COP+Rel plus transitive verbs
COP+Rel+TV+DTV	COP+Rel+TV plus ditransitive verbs
COP+Rel+TV+RA	COP+Rel+TV plus anaphoric pronouns (e.g., he, him, it, herself) of bounded scope
COP+Rel+TV+GA	COP+Rel+TV plus unbounded anaphoric pronouns
COP+Rel+TV+DTV+RA	COP+Rel+TV+DTV plus bounded anaphoric pronouns

Semantic Complexity [Pratt 2003]

- Modulo $\tau(\cdot)$ controlled fragments express (translate exactly into) a fragment of FO

EX: COP expresses the FO fragment containing the following finite set of sentence forms:

$\mathit{Woman}(\mathrm{Mary})$	Mary is a woman.
$\neg \mathit{Man}(\mathrm{Mary})$	Mary is not a man.
$\forall x(\mathit{Man}(x) \Rightarrow \mathit{Person}(x))$	Every man is a person.
$\forall x(\mathit{Woman}(x) \Rightarrow \neg \mathit{Man}(x))$	No woman is a man.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\exists x(\mathit{Person}(x) \wedge \mathit{Woman}(x))$	Some person is a woman
$\exists x(\mathit{Person}(x) \wedge \neg \mathit{Woman}(x))$	Some person is not a woman.

Semantic Complexity [Pratt 2003]

- Modulo $\tau(\cdot)$ controlled fragments express (translate exactly into) a fragment of FO

EX: COP expresses the FO fragment containing the following finite set of sentence forms:

$\mathit{Woman}(\mathrm{Mary})$	Mary is a woman.
$\neg \mathit{Man}(\mathrm{Mary})$	Mary is not a man.
$\forall x(\mathit{Man}(x) \Rightarrow \mathit{Person}(x))$	Every man is a person.
$\forall x(\mathit{Woman}(x) \Rightarrow \neg \mathit{Man}(x))$	No woman is a man.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\exists x(\mathit{Person}(x) \wedge \mathit{Woman}(x))$	Some person is a woman
$\exists x(\mathit{Person}(x) \wedge \neg \mathit{Woman}(x))$	Some person is not a woman.

- By studying such FO fragments we can:
  - exploit computational logic for semantic processing $\Rightarrow$ computational semantics

Semantic Complexity [Pratt 2003]

- Modulo $\tau(\cdot)$ controlled fragments express (translate exactly into) a fragment of Fo

EX: COP expresses the Fo fragment containing the following finite set of sentence forms:

$\mathit{Woman}(\mathrm{Mary})$	Mary is a woman.
$\neg \mathit{Man}(\mathrm{Mary})$	Mary is not a man.
$\forall x(\mathit{Man}(x) \Rightarrow \mathit{Person}(x))$	Every man is a person.
$\forall x(\mathit{Woman}(x) \Rightarrow \neg \mathit{Man}(x))$	No woman is a man.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.
$\exists x(\mathit{Person}(x) \wedge \mathit{Woman}(x))$	Some person is a woman
$\exists x(\mathit{Person}(x) \wedge \neg \mathit{Woman}(x))$	Some person is not a woman.

- By studying such Fo fragments we can:
  - exploit computational logic for semantic processing $\Rightarrow$ computational semantics
  - understand the complexity of semantic processing $\Rightarrow$ semantic complexity

Queries and Data Complexity

- Semantic complexity can be studied by analyzing reasoning problems
- One reasoning problem of interest is answering questions (information requests)

Queries and Data Complexity

- Semantic complexity can be studied by analyzing reasoning problems
- One reasoning problem of interest is answering questions (information requests)

- We consider tree-shaped questions Q and queries $\tau(Q)$ of the form

$$\phi(x) ::= A(x) \mid R(x,c) \mid \exists y R(x,y) \mid \exists y(R(x,y) \wedge \phi'(y)) \mid \phi'(x) \wedge \phi''(x)$$

  sets of controlled sentences Σ and databases $\mathcal{D}$ and check if

$$\tau(\Sigma) \cup \mathcal{D} \models \tau(Q)[c/x]$$

Queries and Data Complexity

- Semantic complexity can be studied by analyzing reasoning problems
- One reasoning problem of interest is answering questions (information requests)

- We consider tree-shaped questions Q and queries $\tau(Q)$ of the form

$$\phi(x) ::= A(x) \mid R(x,c) \mid \exists y R(x,y) \mid \exists y(R(x,y) \wedge \phi'(y)) \mid \phi'(x) \wedge \phi''(x)$$

  sets of controlled sentences Σ and databases $\mathcal{D}$ and check if

$$\tau(\Sigma) \cup \mathcal{D} \models \tau(Q)[c/x]$$

- We call (by abuse) this problem the (linguistic) knowledge base query answering problem (KBQA)

Queries and Data Complexity

- Semantic complexity can be studied by analyzing reasoning problems
- One reasoning problem of interest is answering questions (information requests)

- We consider tree-shaped questions Q and queries $\tau(Q)$ of the form

$$\phi(x) ::= A(x) \mid R(x,c) \mid \exists y R(x,y) \mid \exists y(R(x,y) \wedge \phi'(y)) \mid \phi'(x) \wedge \phi''(x)$$

  sets of controlled sentences Σ and databases $\mathcal{D}$ and check if

$$\tau(\Sigma) \cup \mathcal{D} \models \tau(Q)[c/x]$$

- We call (by abuse) this problem the (linguistic) knowledge base query answering problem (KBQA)

- Following [Vardi 1982] we are interested in the data complexity of KBQA: its computational complexity measured solely in $\#(\mathcal{D})$

Queries and Data Complexity

- Semantic complexity can be studied by analyzing reasoning problems
- One reasoning problem of interest is answering questions (information requests)

- We consider tree-shaped questions Q and queries $\tau(Q)$ of the form

$$\phi(x) ::= A(x) \mid R(x,c) \mid \exists y R(x,y) \mid \exists y(R(x,y) \wedge \phi'(y)) \mid \phi'(x) \wedge \phi''(x)$$

  sets of controlled sentences Σ and databases $\mathcal{D}$ and check if

$$\tau(\Sigma) \cup \mathcal{D} \models \tau(Q)[c/x]$$

- We call (by abuse) this problem the (linguistic) knowledge base query answering problem (KBQA)

- Following [Vardi 1982] we are interested in the data complexity of KBQA: its computational complexity measured solely in $\#(\mathcal{D})$

$\Rightarrow$ Use resolution decision procedures to derive upper bounds and reductions to derive lower bounds!

Resolution Saturations

- Define derivability function $\rho(\cdot)$ with rules

$$\text{res}\ \frac{\Gamma, C \vee \overline{L} \qquad \Gamma, C \vee L'}{(C \vee C')\sigma} \qquad \text{fact}\ \frac{\Gamma, C \vee L \vee L'}{(C \vee L)\sigma}$$

- The resolution calculus is a function $R(\cdot)$ s.t.

$$R(\Gamma) := \Gamma \cup \rho(\Gamma)$$

- The saturation $R^{\infty}(\Gamma)$ of Γ is defined as
  - $R^0(\Gamma) := \Gamma$
  - $R^{i+1}(\Gamma) := R(R^i(\Gamma))$, for $i > 0$
  - $R^{\infty}(\Gamma) := \bigcup\{R^i(\Gamma) \mid i \geq 0\}$

Resolution Saturations

- Define derivability function $\rho(\cdot)$ with rules

$$\text{res}\ \frac{\Gamma, C \vee \overline{L} \qquad \Gamma, C \vee L'}{(C \vee C')\sigma} \qquad \text{fact}\ \frac{\Gamma, C \vee L \vee L'}{(C \vee L)\sigma}$$

- The resolution calculus is a function $R(\cdot)$ s.t.

$$R(\Gamma) := \Gamma \cup \rho(\Gamma)$$

- The saturation $R^{\infty}(\Gamma)$ of Γ is defined as
  - $R^0(\Gamma) := \Gamma$
  - $R^{i+1}(\Gamma) := R(R^i(\Gamma))$, for $i > 0$
  - $R^{\infty}(\Gamma) := \bigcup\{R^i(\Gamma) \mid i \geq 0\}$

- Sound and complete for Fo (un)satisfiability, modulo clausification and skolemization

Resolution Saturations

- Define derivability function $\rho(\cdot)$ with rules

$$\text{res}\ \frac{\Gamma, C \vee \overline{L} \qquad \Gamma, C \vee L'}{(C \vee C')\sigma} \qquad \text{fact}\ \frac{\Gamma, C \vee L \vee L'}{(C \vee L)\sigma}$$

- The resolution calculus is a function $R(\cdot)$ s.t.

$$R(\Gamma) := \Gamma \cup \rho(\Gamma)$$

- The saturation $R^{\infty}(\Gamma)$ of Γ is defined as
  - $R^0(\Gamma) := \Gamma$
  - $R^{i+1}(\Gamma) := R(R^i(\Gamma))$, for $i > 0$
  - $R^{\infty}(\Gamma) := \bigcup\{R^i(\Gamma) \mid i \geq 0\}$

- Can be refined to decide fragments of FO!

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:
  1. the (functional) depth of terms/litterals is bounded by an $d \in \mathbb{N}$:

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:
  1. the (functional) depth of terms/litterals is bounded by an $d \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with an acceptable order like $\prec_d$

$$L \prec_d L \Leftrightarrow_{df} \begin{cases} d(L) < d(L'), \mathit{Var}(L) \subseteq \mathit{Var}(L'), \text{ and} \\ d(x, L) < d(x, L), \text{ for all } x \in \mathit{Var}(L) \end{cases}$$

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:
  1. the (functional) depth of terms/litterals is bounded by an $d \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with an acceptable order like $\prec_d$

$$L \prec_d L \Leftrightarrow_{df} \begin{cases} d(L) < d(L'), \mathit{Var}(L) \subseteq \mathit{Var}(L'), \text{ and} \\ d(x, L) < d(x, L), \text{ for all } x \in \mathit{Var}(L) \end{cases}$$

  2. the length of clauses is bounded by $l \in \mathbb{N}$:

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:
  1. the (functional) depth of terms/litterals is bounded by an $d \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with an acceptable order like $\prec_d$

$$L \prec_d L \Leftrightarrow_{df} \begin{cases} d(L) < d(L'), \mathit{Var}(L) \subseteq \mathit{Var}(L'), \text{ and} \\ d(x, L) < d(x, L), \text{ for all } x \in \mathit{Var}(L) \end{cases}$$

  2. the length of clauses is bounded by $l \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with the splitting rule

$$\text{split } \frac{\Gamma, C \vee L \vee L' \qquad \begin{matrix} \Gamma, C \vee L \\ \vdots \\ C'\sigma \end{matrix} \qquad \begin{matrix} \Gamma, C \vee L' \\ \vdots \\ C'\sigma \end{matrix}}{C'\sigma} \; (\mathit{Var}(L) \cap \mathit{Var}(L') = \emptyset)$$

Resolution Decision Procedures [Joyner 1976]

- Saturations finitely converge when:
  1. the (functional) depth of terms/litterals is bounded by an $d \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with an acceptable order like $\prec_d$

$$L \prec_d L \Leftrightarrow_{df} \begin{cases} d(L) < d(L'), \mathit{Var}(L) \subseteq \mathit{Var}(L'), \text{ and} \\ d(x, L) < d(x, L), \text{ for all } x \in \mathit{Var}(L) \end{cases}$$

  2. the length of clauses is bounded by $l \in \mathbb{N}$:

  $\Rightarrow$ can be achieved with the splitting rule

$$\text{split } \frac{\Gamma, C \vee L \vee L' \qquad C'\sigma \qquad C'\sigma}{C'\sigma} \; \begin{array}{c} \Gamma, C \vee L \\ \vdots \\ \end{array} \; \begin{array}{c} \Gamma, C \vee L' \\ \vdots \\ \end{array} \; (\mathit{Var}(L) \cap \mathit{Var}(L') = \emptyset)$$

- Both refinements are sound and complete!

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

EX:

$P(x) \vee Q(y) \vee R(x, y)$	yes
$P(x) \vee Q(x) \vee N(f(x))$	yes
$P(c) \vee T(a, c, d, e)$	yes
$S(x, y) \vee Q(z)$	no
$S(x, y) \vee R(z, f(c))$	no

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

EX:

$P(x) \vee Q(y) \vee R(x, y)$	yes
$P(x) \vee Q(x) \vee N(f(x))$	yes
$P(c) \vee T(a, c, d, e)$	yes
$S(x, y) \vee Q(z)$	no
$S(x, y) \vee R(z, f(c))$	no

- Refinements decide $\mathcal{S}^+$ when combined with monadization [Joyner 1976]
  - Resolution may result in clause that lie outside the $\mathcal{S}^+$ class

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

EX:

$P(x) \vee Q(y) \vee R(x, y)$	yes
$P(x) \vee Q(x) \vee N(f(x))$	yes
$P(c) \vee T(a, c, d, e)$	yes
$S(x, y) \vee Q(z)$	no
$S(x, y) \vee R(z, f(c))$	no

- Refinements decide $\mathcal{S}^+$ when combined with monadization [Joyner 1976]
  - Resolution may result in clause that lie outside the $\mathcal{S}^+$ class
  - Monadization defines equisatisfiable variants of such clauses that are in $\mathcal{S}^+$

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

EX:

$P(x) \vee Q(y) \vee R(x, y)$	yes
$P(x) \vee Q(x) \vee N(f(x))$	yes
$P(c) \vee T(a, c, d, e)$	yes
$S(x, y) \vee Q(z)$	no
$S(x, y) \vee R(z, f(c))$	no

- Refinements decide $\mathcal{S}^+$ when combined with monadization [Joyner 1976]
  - Resolution may result in clause that lie outside the $\mathcal{S}^+$ class
  - Monadization defines equisatisfiable variants of such clauses that are in $\mathcal{S}^+$

EX: we compute sets of substitutions by the terms in the clause

$$S(x, y) \wedge R(z, f(c)) \quad \rightsquigarrow \quad \begin{array}{c} S(x, x) \vee R(z, f(c)), S(x, z) \vee R(z, f(c)) \\ S(x, c) \vee R(z, f(c)), S(x, f(c)) \vee R(z, f(c)) \end{array}$$

The $\mathcal{S}^+$ and Monadization [Leitsch 2003]

- The $\mathcal{S}^+$ class of clauses is the class where
  - every litteral (positive or negative) contains at most one variable
  - or else, it contains all the variables of the clause

EX:

$P(x) \vee Q(y) \vee R(x, y)$	yes
$P(x) \vee Q(x) \vee N(f(x))$	yes
$P(c) \vee T(a, c, d, e)$	yes
$S(x, y) \vee Q(z)$	no
$S(x, y) \vee R(z, f(c))$	no

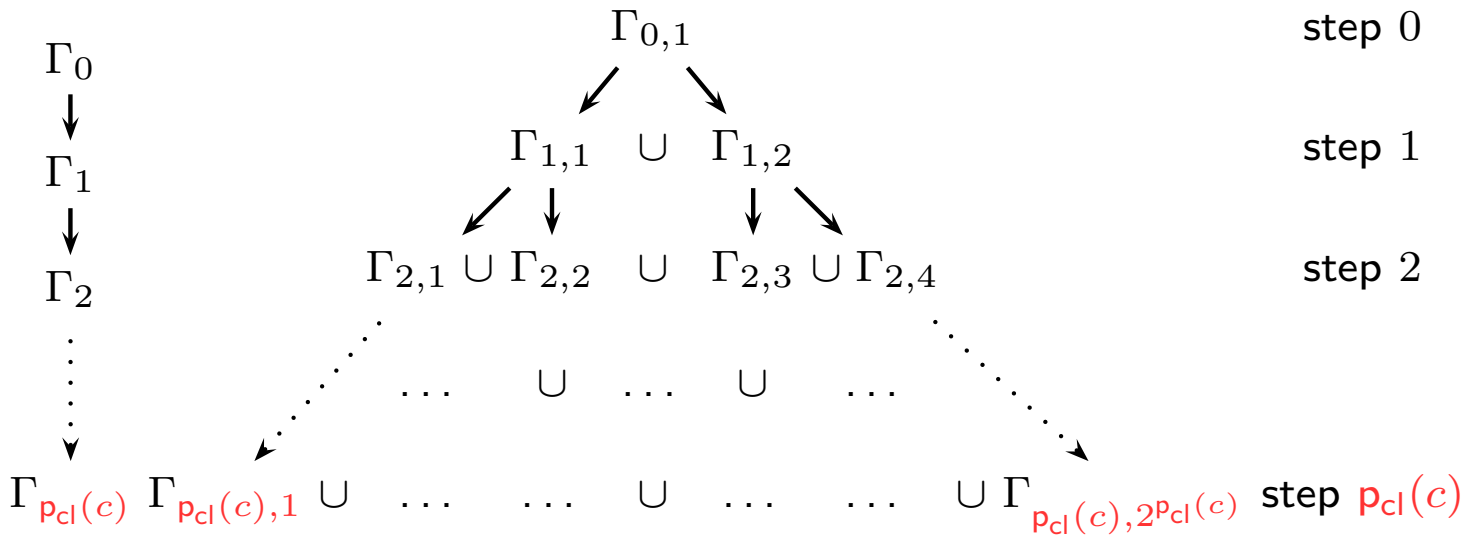
- Refinements decide $\mathcal{S}^+$ when combined with monadization [Joyner 1976]
  - Resolution may result in clause that lie outside the $\mathcal{S}^+$ class
  - Monadization defines equisatisfiable variants of such clauses that are in $\mathcal{S}^+$

EX: we compute sets of substitutions by the terms in the clause

$$S(x, y) \wedge R(z, f(c)) \quad \rightsquigarrow \quad \begin{array}{c} S(x, x) \vee R(z, f(c)), S(x, z) \vee R(z, f(c)) \\ S(x, c) \vee R(z, f(c)), S(x, f(c)) \vee R(z, f(c)) \end{array}$$

- Many of the FOEs are contained or can be reduced to the $\mathcal{S}^+$ class!

Data Complexity of Saturations [Thorne 2010]



- If the signature of a set Γ of clauses is finite, and a a bound depth d and a length bound l exist, $R^{\infty}(\Gamma)$ finitely converges
- If Γ has c constants, convergence is reached after $\mathsf{p_{cl}}(c)$ (without splitting) or $2^{\mathsf{p_{cl}}(c)}$ (with splitting) steps

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

It can be proven that no splittings occur

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

It can be proven that no splittings occur

Therefore $(\tau(\Sigma)^{cl} \cup \mathcal{D})^{\infty}$ is finite and can be computed in time polynomial in $\#(\mathcal{D})$

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

Furthermore, TSQs ϕ also yield HORN clauses ϕ^{cl}

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

Grounding $(\tau(\Sigma)^{cl} \cup \mathcal{D})^{\infty} \cup \{\phi^{cl}\}$ yields polynomially many propositional Horn clauses in $\#(\mathcal{D})$

Query evaluation: PTIME for COP+TV+DTV

NB: SAT for propositional (or ground) clauses is in PTIME (actually, PTIME-complete)

COP+TV+DTV meaning representations $\tau(\Sigma)$ yield Horn clauses $\tau(\Sigma)^{cl}$

$$\neg P(x) \vee \pm Q(x), \quad \neg P(x) \vee \pm L(x), \quad \neg P(x) \vee Q(f(x)), \quad \neg P(x) \vee \neg Q(y) \vee \pm L(x,y), \\ \pm L, \quad \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z), \quad \neg P(x) \vee \neg Q(y) \vee N(g(x,y)),$$

These clauses are contained in $\mathcal{S}^+ \Rightarrow$ resolution decides COP+TV+DTV

Grounding $(\tau(\Sigma)^{cl} \cup \mathcal{D})^{\infty} \cup \{\phi^{cl}\}$ yields polynomially many propositional Horn clauses in $\#(\mathcal{D})$

Finally, the satisfiablity of $GR((\tau(\Sigma)^{cl} \cup \mathcal{D})^{\infty} \cup \{\phi^{cl}\})$ can be checked in time polynomial in $\#(\mathcal{D})$

Data Complexity of the FOEs

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Resolution can be used to derive lower and upper complexity bounds for SAT [Thorne 2010]
- We get as complete picture:

	KBQA	SAT
COP	in LogSpace	in LogSpace
COP+TV	in PTime	in LogSpace
COP+TV+DTV	in PTime	in LogSpace
COP+Rel	coNP-complete	in LogSpace
COP+Rel+TV	coNP-complete	NP-complete
COP+Rel+DTV	coNP-complete	NP-complete
COP+Rel+DTV+TV	coNP-complete	NP-complete

NB: Questions express $\mathrm{FO}^{\exists}_{+}$ formulas!

Data Complexity of the FOEs

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Resolution can be used to derive lower and upper complexity bounds for SAT [Thorne 2010]
- We get as complete picture:

	KBQA	SAT
COP	in LogSpace	in LogSpace
COP+TV	in PTime	in LogSpace
COP+TV+DTV	in PTime	in LogSpace
COP+Rel	coNP-complete	in LogSpace
COP+Rel+TV	coNP-complete	NP-complete
COP+Rel+DTV	coNP-complete	NP-complete
COP+Rel+DTV+TV	coNP-complete	NP-complete

NB: Questions express $\mathrm{FO}^{\exists}_{+}$ formulas!

Data Complexity of the FOEs

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Resolution can be used to derive lower and upper complexity bounds for SAT [Thorne 2010]
- "Boolean-closedness":

	KBQA	SAT
COP	in LogSpace	in LogSpace
COP+TV	in PTime	in LogSpace
COP+TV+DTV	in PTime	in LogSpace
COP+Rel	coNP-complete	in LogSpace
COP+Rel+TV	coNP-complete	NP-complete
COP+Rel+DTV	coNP-complete	NP-complete
COP+Rel+DTV+TV	coNP-complete	NP-complete

NB: Questions express $\mathrm{FO}^{\exists}_{+}$ formulas!

Conclusions and Further Work

- We have given an overview of controlled languages and semantic complexity
- We have argued that semantic complexity is an important issue in computational semantics
- While eliminating ambiguity makes translation efficient, reasoning becomes intractable already for very simple fragments
- We have also pinpoited maximal and minimal combinations of English constructs that give rise, resp., to tractable and intractable data complexity

Conclusions and Further Work

- We have given an overview of controlled languages and semantic complexity
- We have argued that semantic complexity is an important issue in computational semantics
- While eliminating ambiguity makes translation efficient, reasoning becomes intractable already for very simple fragments
- We have also pinpoited maximal and minimal combinations of English constructs that give rise, resp., to tractable and intractable data complexity
- As further work we would like to see how their theoretical (worst-case) complexity
  - correlates with the observed behavior (processing time) of systems
  - by possibly using off-the-shelf automated reasoners that decide the FOEs and other controlled fragments
  - and see how meaningful a measure of complexity it is, for natural language fragments

(Data) Scalable and Non-Scalable Constructs [Thorne 2010]

	Declarations	Questions
Constructs that scale (PTIME or less)	– Negation in predicate **VP**s, relatives in predicate **VP**s, conjunction in predicate **VP**s – Relatives and conjunction in subject **NP**s and predicate **VP**s, but no negation	– Existential quantifiers, conjunction, relatives, disjunctions
Constructs that do not (coNP-hard)	– Negation in subject **NP**s – Relatives and negation in subject **NP**s and predicate **VP**s	– Full negation – Comparisons – Universal restrictions
Undecidable Constructs	– **TV**s, relatives, negation, existential and universal quantifiers, restricted anaphoric pronouns and indeterminate pronouns in subject **NP**s and predicate **VP**s, copula	– **TV**s, existential indeterminate pronouns, relatives and restricted anaphoric pronouns

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Encode Φ into DB $\mathcal{F}_\Phi$

$$\vdots$$
$$c_i \text{ Ps } l_{i,1}, c_i \text{ Ps } l_{i,2}, c_i \text{ Ns } n_{i,1}, c_i \text{ Ns } n_{i,2}$$
$$\vdots$$
$$\text{true is an } A_t$$

Consider Σ_Φ

no A_t is an A_f everything that is not an A_t is an A_f

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Encode Φ into $\mathcal{D}_\Phi$

$$\vdots$$
$$P(c_i, l_{i,1}), P(c_i, l_{i,2}), N(c_i, n_{i,1}), N(c_i, n_{i,2})$$
$$\vdots$$
$$A_t(\mathsf{true})$$

Consider $\tau(\Sigma)_\Phi$

$$\forall x(A_f(x) \leftrightarrow \neg A_t(x))$$

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Ask TSQ Q_Φ

does somebody P_1s something that *Val*s some A_f and
P_2s something that *Val*s some A_f
N_1s something that *Val*s some A_t
N_2s something that *Val*s some A_t?

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Ask query ϕ_Φ

$$\begin{array}{c} \exists c \exists l_1 \exists l_2 \exists l_3 \exists l_4 (\\ P_1(c, l_1) \wedge \exists v_1 (\mathit{Val}(l_1, v_1) \wedge A_f(v_1)) \wedge \\ P_2(c, l_2) \wedge \exists v_2 (\mathit{Val}(l_2, z_2) \wedge A_f(v_2)) \wedge \\ N_1(c, l_3) \wedge \exists v_2 (\mathit{Val}(l_3, v_3) \wedge A_t(v_3)) \wedge \\ N_2(c, l_4) \wedge \exists v_3 (\mathit{Val}(l_4, v_4) \wedge A_t(v_4))) \end{array}$$

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

We claim

$$\mathcal{O}_\Phi \cup \mathcal{D}_\Phi \not\models \phi_\Phi \text{ iff } \Phi \text{ has no model}$$

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

"$\Rightarrow$" given a $\mathcal{I}$ s.t.

$$\mathcal{I} \models \tau(\Sigma)_\Phi, \quad \mathcal{I} \models \mathcal{D}_\Phi \quad \text{and} \quad \mathcal{I} \not\models \phi_\Phi$$

Idea:

$$\begin{array}{lll} \mathcal{I}, \gamma \not\models \exists v(\mathit{Val}(l, v) \wedge A_f(v)) & \Leftrightarrow & \mathcal{I}, \gamma \models \neg\exists v(\mathit{Val}(l, v) \wedge A_f(v)) \\ & \Leftrightarrow_{\tau(\Sigma)_\Phi} & \mathcal{I}, \gamma \models \neg\exists v(\mathit{Val}(l, v) \wedge \neg A_t(v)) \\ & \Leftrightarrow & \mathcal{I}, \gamma \models \forall v(\mathit{Val}(l, v) \rightarrow A_t(v)) \end{array}$$

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

"$\Rightarrow$" given a $\mathcal{I}$ s.t.

$$\mathcal{I} \models \tau(\Sigma)_\Phi, \quad \mathcal{I} \models \mathcal{D}_\Phi \quad \text{and} \quad \mathcal{I} \not\models \phi_\Phi$$

Idea:

$$\begin{array}{lll} \mathcal{I}, \gamma \not\models \exists v(\mathit{Val}(l, v) \wedge A_f(v)) & \Leftrightarrow & \mathcal{I}, \gamma \models \neg\exists v(\mathit{Val}(l, v) \wedge A_f(v)) \\ & \Leftrightarrow_{\tau(\Sigma)_\Phi} & \mathcal{I}, \gamma \models \neg\exists v(\mathit{Val}(l, v) \wedge \neg A_t(v)) \\ & \Leftrightarrow & \mathcal{I}, \gamma \models \forall v(\mathit{Val}(l, v) \rightarrow A_t(v)) \end{array}$$

we partition the domain!

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Define truth assignment $\delta(\cdot)$ by putting

$$\delta(l) = \mathsf{true} \Leftrightarrow (l, \mathsf{true}) \in \mathit{Val}^{\mathcal{I}}$$

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

"$\Leftarrow$": Symmetrical argument

Query evaluation: coNP-hard for COP+Rel

NB: A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses is NP-complete [Scharf, 1994]

Therefore:

$$\tau(\Sigma)_\Phi \cup \mathcal{D}_\Phi \not\models \phi_\Phi \text{ iff } \Phi \text{ has a model}$$

$\Rightarrow$ full negation blows up data complexity!