Semantic Complexity of Controlled Languages

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

TiLPS - Tilburg - 14/4/2011

Outline

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?

Controlled Languages

- A controlled language 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]
- Controlled languages interfaces 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

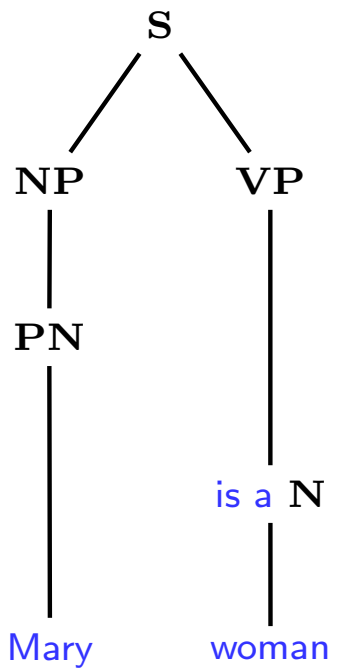
Controlled Languages

- A controlled language 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]
- Controlled languages interfaces 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 Languages - COP [Pratt & Third 2006]

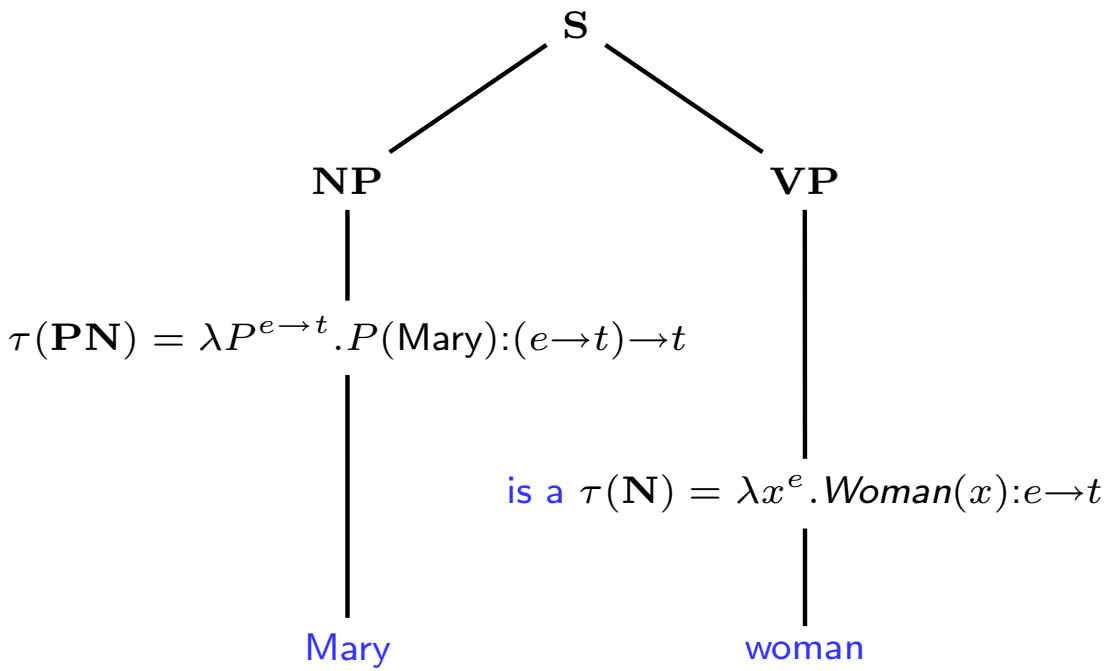
Syntax Rules	Semantics $(= \tau(\cdot))$
$\mathbf{S} \longrightarrow \mathbf{NP\ VP}$	$\tau(\mathbf{NP})(\tau(\mathbf{VP})) \rhd \tau(\mathbf{S})$
$\mathbf{VP} \longrightarrow$ is a $\mathbf{N}$	$\tau(\mathbf{VP}) = \tau(\mathbf{N})$
$\mathbf{VP} \longrightarrow$ is not a $\mathbf{N}$	$\tau(\mathbf{VP})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

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

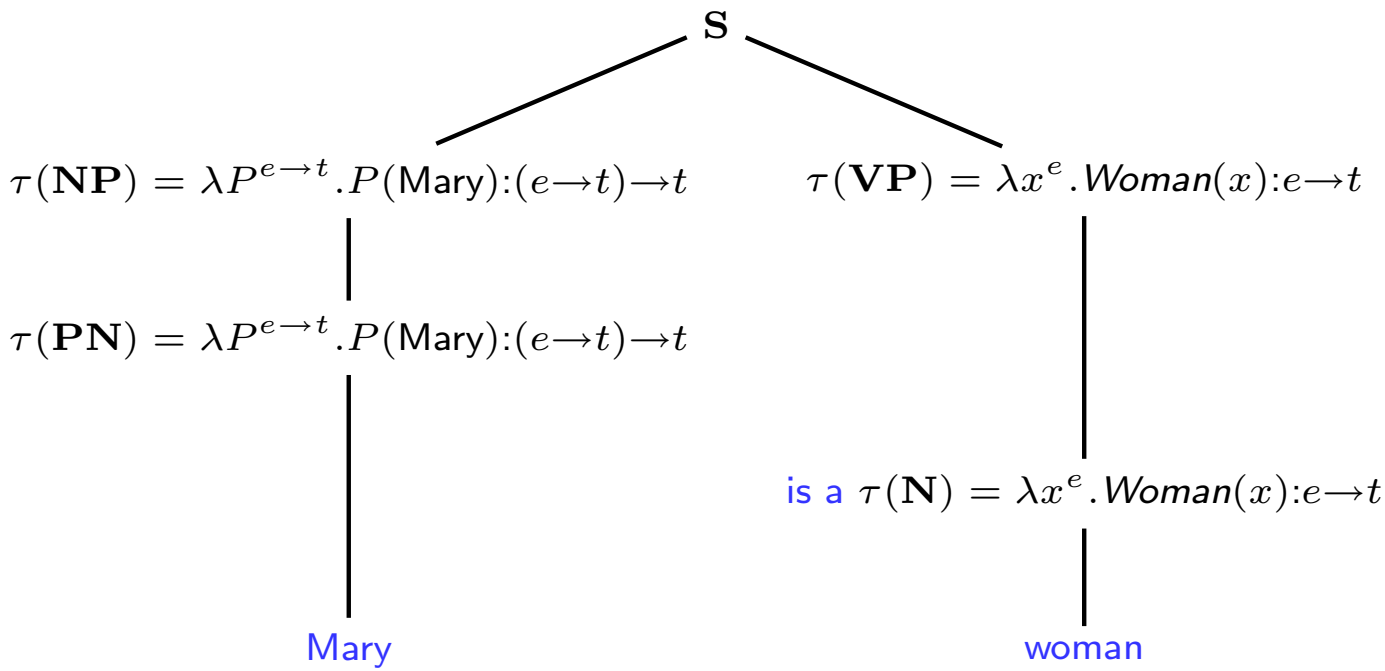


Parsing and interpreting "Mary is a woman"

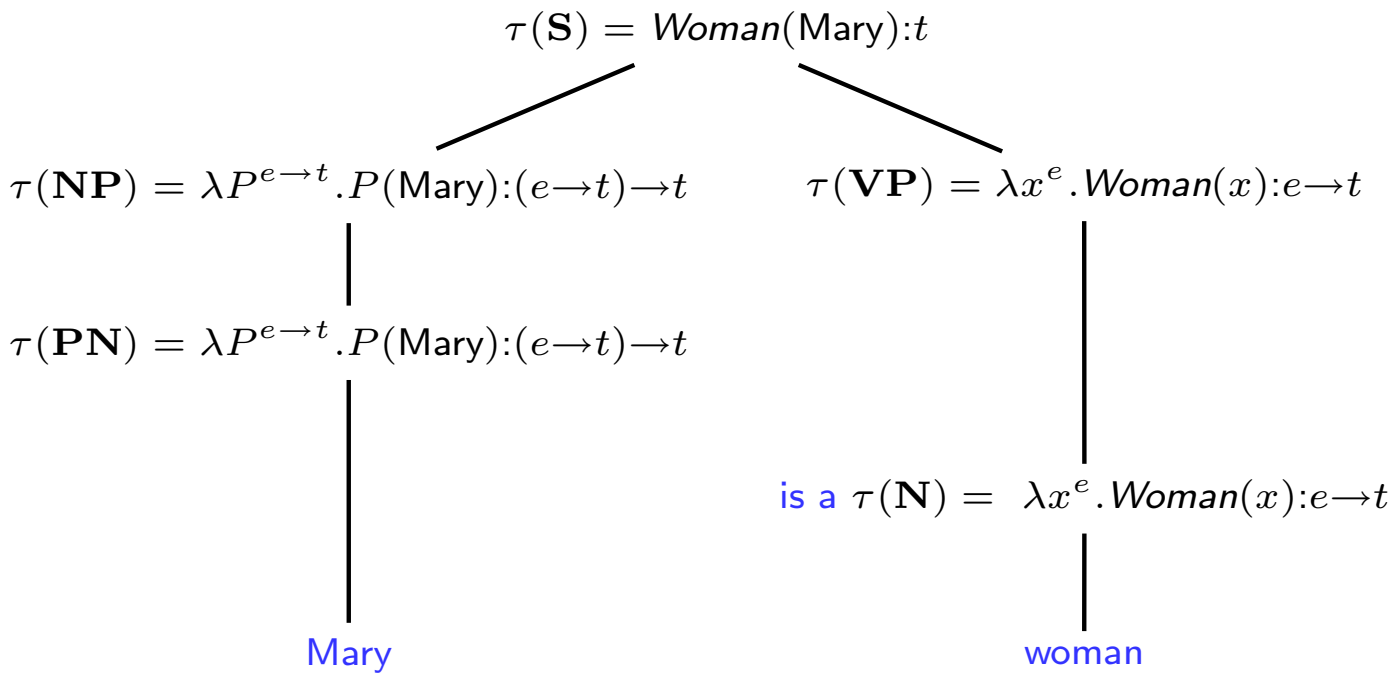
Controlled Languages - COP (cntd.)



Parsing and interpreting "Mary is a woman"

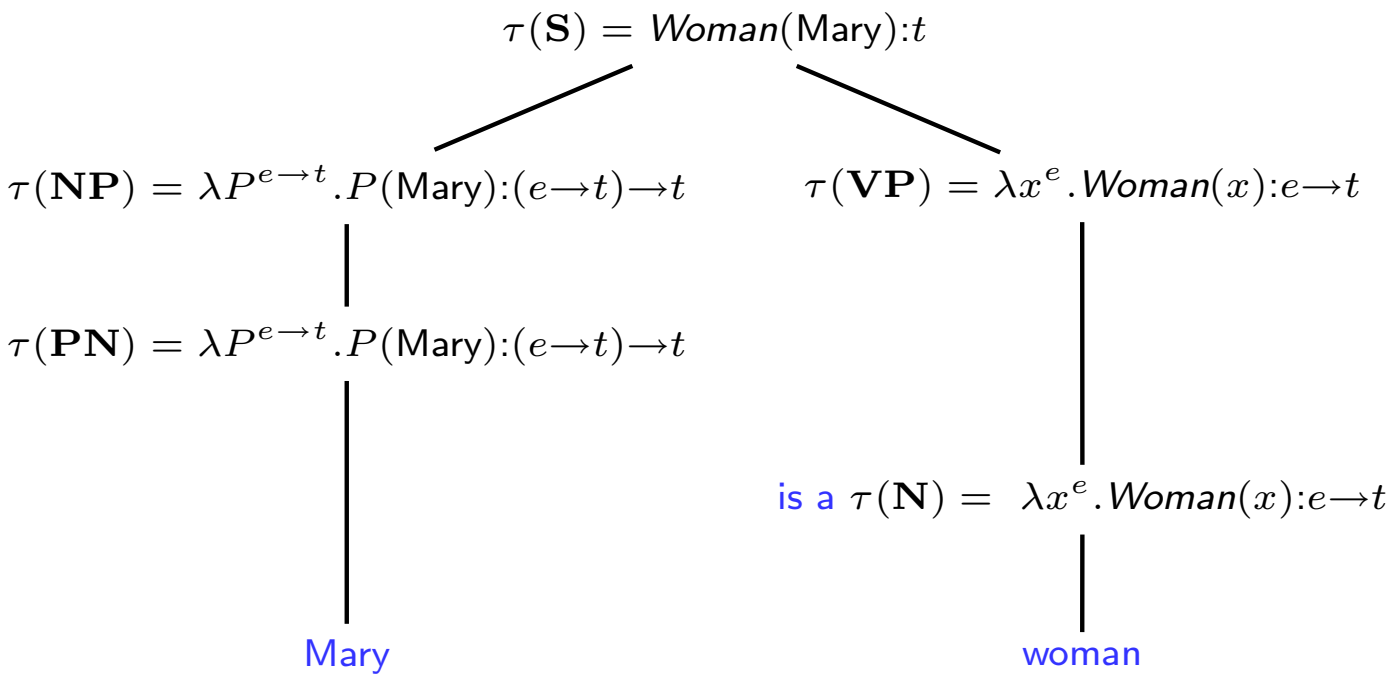


Parsing and interpreting "Mary is a woman"



Parsing and interpreting "Mary is a woman"

Controlled Languages - 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

- 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.

- 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

- 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

A Survery of Controlled Languages

CL (English)	Maps to	Goal
FOEs [Pratt & Third 2005]	FO fragments	Knowledge representation
ACE [Fuchs 2005]	FO	Knowledge representation
ACE-OWL [Kaaljurand 2007]	OWL-DL	Ontology authoring, querying
PENG [Schwitter 2003]	OWL-DL	Ontology authoring, querying
SOS [Schwitter2008]	OWL-DL	Ontology authoring, querying
CLCE [Sowa2004]	FOL	Knowledge representation
AECMA [Unwalla 2005]	?	User specifications
English Query (EQ) [Blum 1999]	SQL	DB querying/management
OWL-CNL [Schwitter 2006]	OWL-DL	Ontology authoring
Easy English [Bernth 1998]	?	User specifications
λ-SQL [Winter 2006]	SQL	Database querying
nRQL [Schwitter 2008]	FO queries	Ontology querying
Rabbit [Schwitter2008]	OWL	Ontology authoring
ACE-PQL [Bernstein 2005]	PQL	Ontology querying
QE-III [Clifford 1987]	IL	Database querying

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:
  1. if computation deterministic $\Rightarrow$ deterministic Turing machine

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:
  1. if computation deterministic $\Rightarrow$ deterministic Turing machine
  2. if computation non-deterministic $\Rightarrow$ non-deterministic Turing machine

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:
  1. if computation deterministic $\Rightarrow$ deterministic Turing machine
  2. if computation non-deterministic $\Rightarrow$ non-deterministic Turing machine
- We can model the time and space required by computations:

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:
  1. if computation deterministic $\Rightarrow$ deterministic Turing machine
  2. if computation non-deterministic $\Rightarrow$ non-deterministic Turing machine
- We can model the time and space required by computations:
  1. computation steps $\Rightarrow$ time $t(M)$ of M

(3-tape) Turing Machines

- Universal model of computation (Church-Turing thesis)
- A (3-tape) Turing machine M consists of:
  - an input (read-only) tape
  - an output (write-only) tape
  - a register (read/write) tape
  - a read/write head with internal states that at each computation step t
    - reads a character from the input and register tapes
    - (possibly) writes/deletes/overwrites a character in the register tape
    - (possibly) writes a character in the output tape
    - (possibly) changes state and
      - stays put
      - moves to the right of the tapes
      - moves to the left of the tapes
- Two kinds:
  1. if computation deterministic $\Rightarrow$ deterministic Turing machine
  2. if computation non-deterministic $\Rightarrow$ non-deterministic Turing machine
- We can model the time and space required by computations:
  1. computation steps $\Rightarrow$ time $t(M)$ of M
  2. size of register tape $\Rightarrow$ space $s(M)$ of M

Decision Problems

- Turing machines typically decide problems

Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

EX: Given a graph G, does G contain a Hamiltonian cycle?

Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

EX: Given a graph G, does G contain a Hamiltonian cycle?

- They are typically formalized as languages $L \subseteq \{0,1\}^*$

Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

EX: Given a graph G, does G contain a Hamiltonian cycle?

- They are typically formalized as languages $L \subseteq \{0,1\}^*$
- By deciding L we mean that a Turing machine M exists s.t., for all $w \in \{0,1\}^*$, M returns "yes" on w iff $w \in L$

Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

EX: Given a graph G, does G contain a Hamiltonian cycle?

- They are typically formalized as languages $L \subseteq \{0,1\}^*$
- By deciding L we mean that a Turing machine M exists s.t., for all $w \in \{0,1\}^*$, M returns "yes" on w iff $w \in L$
- The time $t(M)$ and space $s(M)$ that M spends on deciding L give way to the computational (time and space) complexity of L

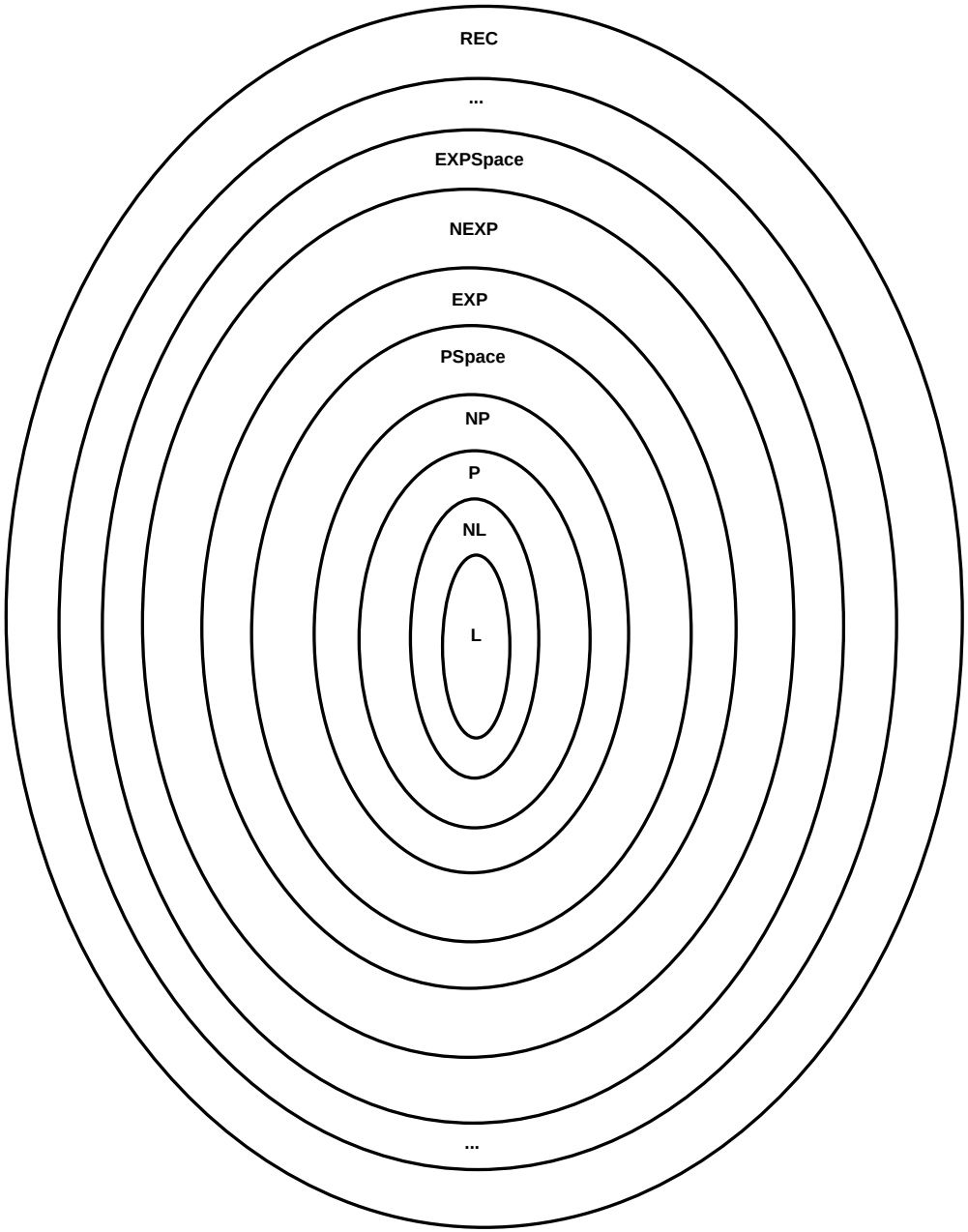
Decision Problems

- Turing machines typically decide problems
- A decision problem can be seen as a question asked over a collection of instances

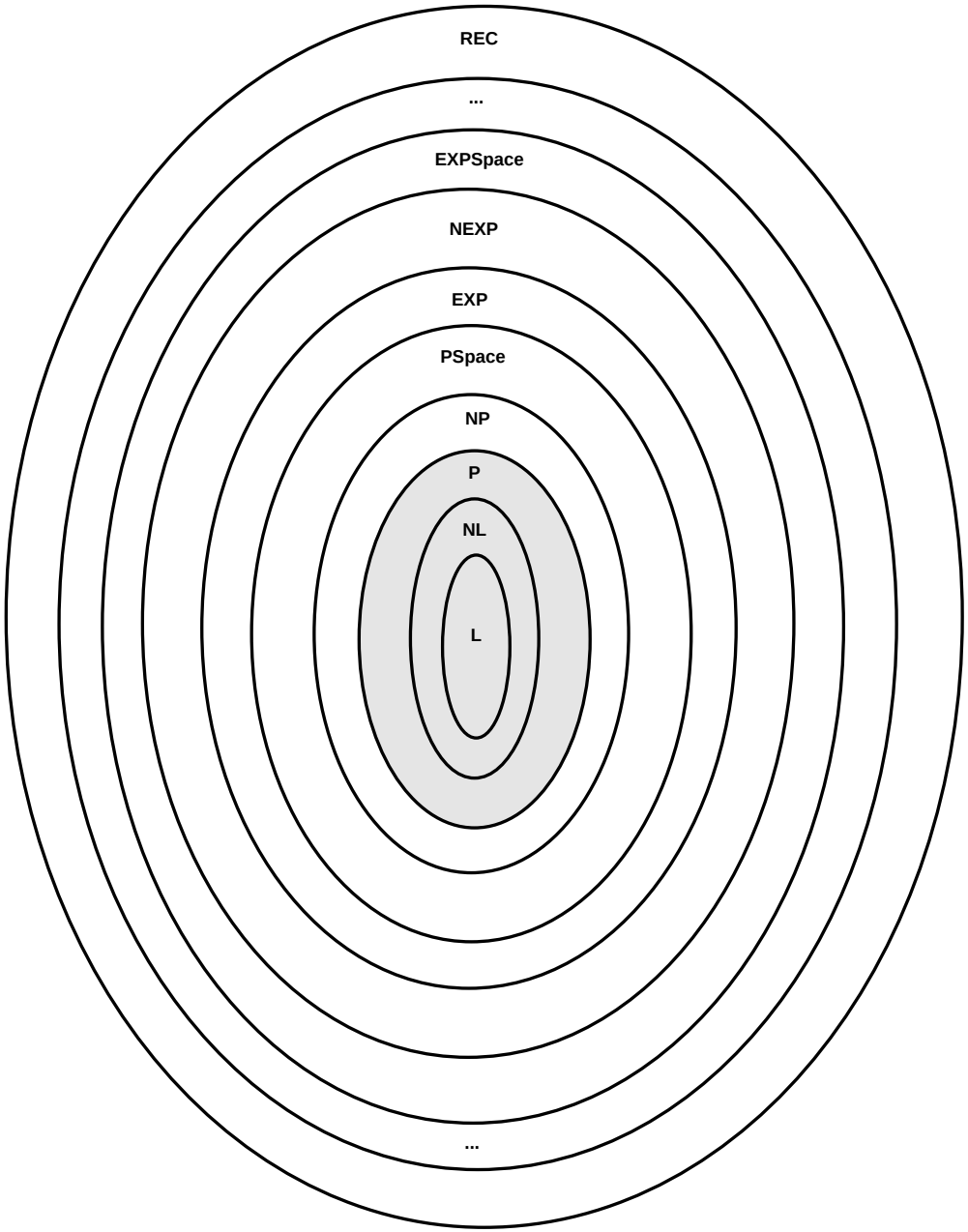
EX: Given a graph G, does G contain a Hamiltonian cycle?

- They are typically formalized as languages $L \subseteq \{0,1\}^*$
- By deciding L we mean that a Turing machine M exists s.t., for all $w \in \{0,1\}^*$, M returns "yes" on w iff $w \in L$
- The time $t(M)$ and space $s(M)$ that M spends on deciding L give way to the computational (time and space) complexity of L
- Problems can be classified into complexity classes on the grounds of their computational complexity

Complexity Classes

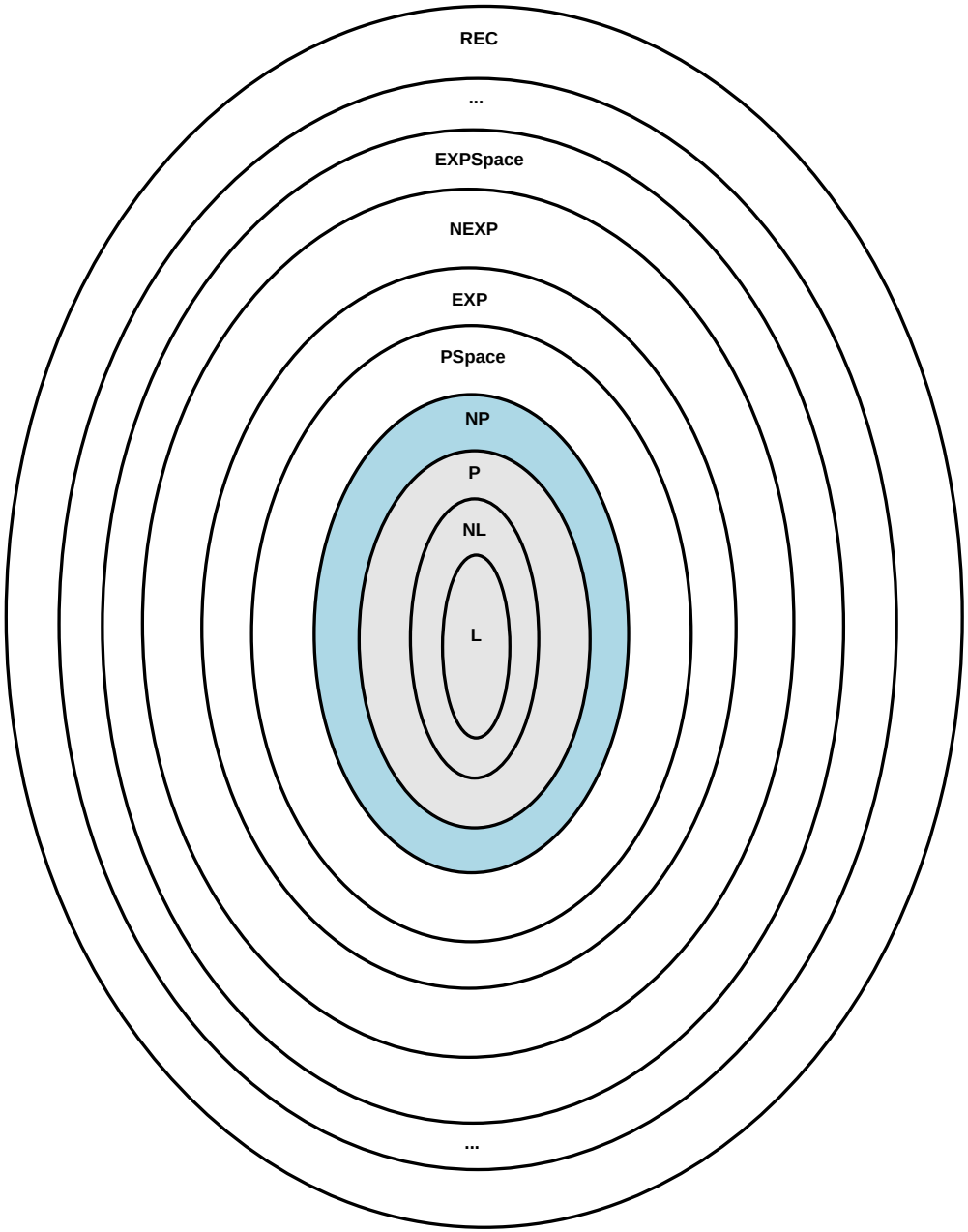


Complexity Classes



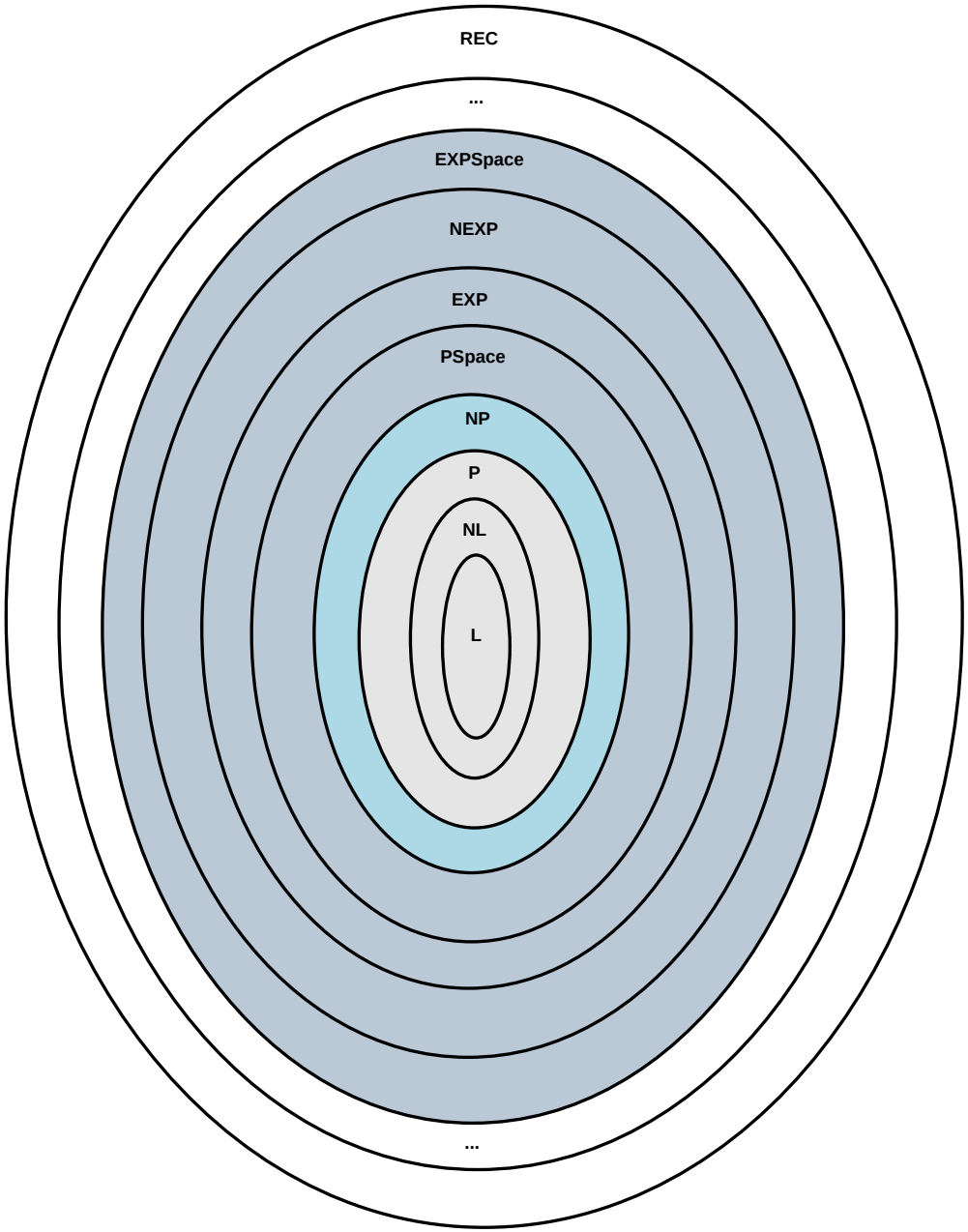
- Problems in P are said to be tractable

Complexity Classes



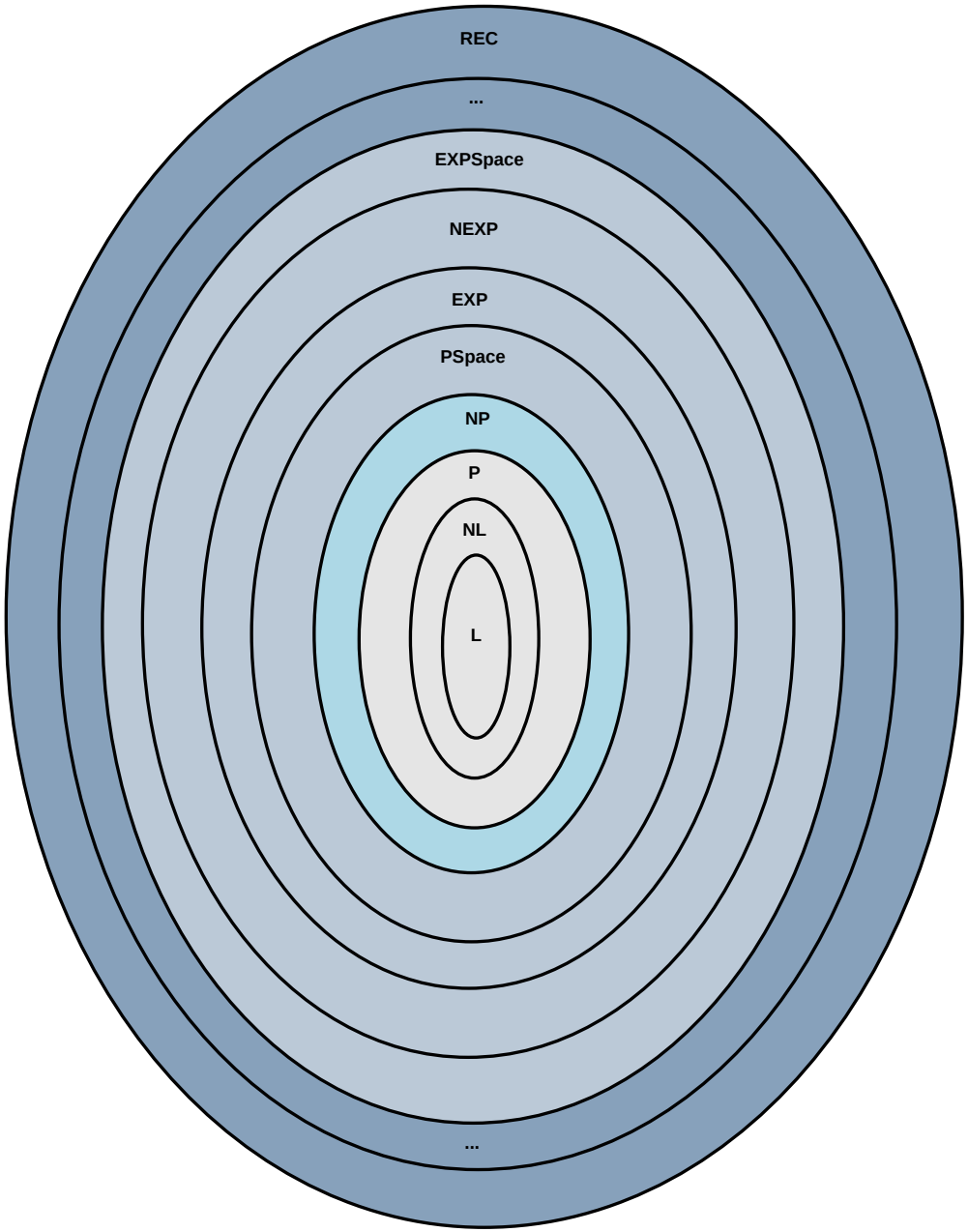
- Problems in P are said to be tractable
- Problems in NP or beyond are said to be intractable

Complexity Classes



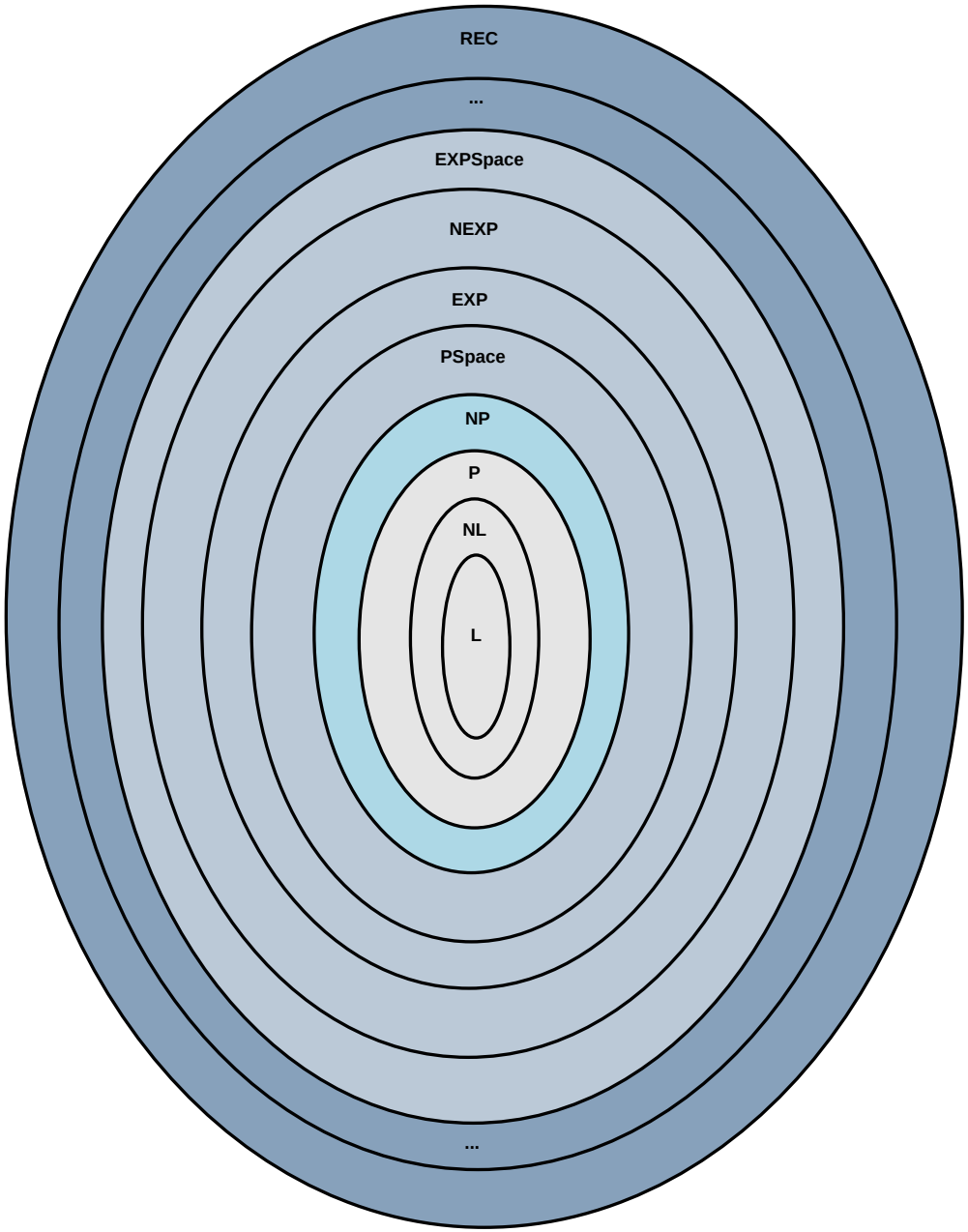
- Problems in P are said to be tractable
- Problems in NP or beyond are said to be intractable
- Beyond NP computation becomes harder and harder

Complexity Classes



- Problems in P are said to be tractable
- Problems in NP or beyond are said to be intractable
- Beyond NP computation becomes harder and harder
- Beyond Rec problems become undecidable

Complexity Classes



- Problems in P are said to be tractable
- Problems in NP or beyond are said to be intractable
- Beyond NP computation becomes harder and harder
- Beyond Rec problems become undecidable

REM: We conjecture that these inclusions are strict and that

$$\mathrm{NP} \neq \mathrm{P}$$

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges
- A problem L is said to be:

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C
  2. C-complete if it is C-hard and also in C

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C
  2. C-complete if it is C-hard and also in C
- C-hard problems can be used as "sieves" to separate classes

C-hard and C-complete Problems

- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C
  2. C-complete if it is C-hard and also in C
- C-hard problems can be used as "sieves" to separate classes
- Problems that are NP-hard are "intrinsecally" intractable

C-hard and C-complete Problems

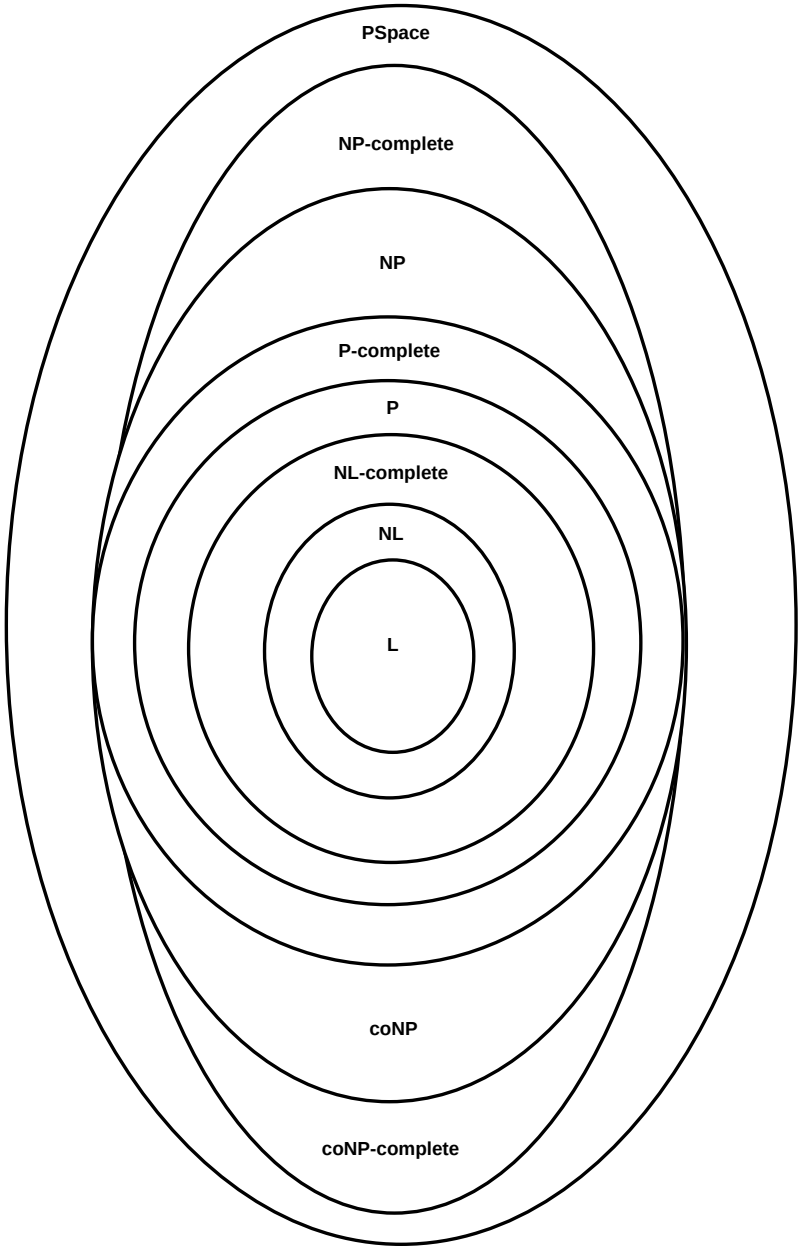
- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C
  2. C-complete if it is C-hard and also in C
- C-hard problems can be used as "sieves" to separate classes
- Problems that are NP-hard are "intrinsecally" intractable
- Similarly, we can define the complement coC of C as $\mathrm{coC} := \{\{0,1\}^* \setminus L \mid L \text{ in } \mathrm{C}\}$

C-hard and C-complete Problems

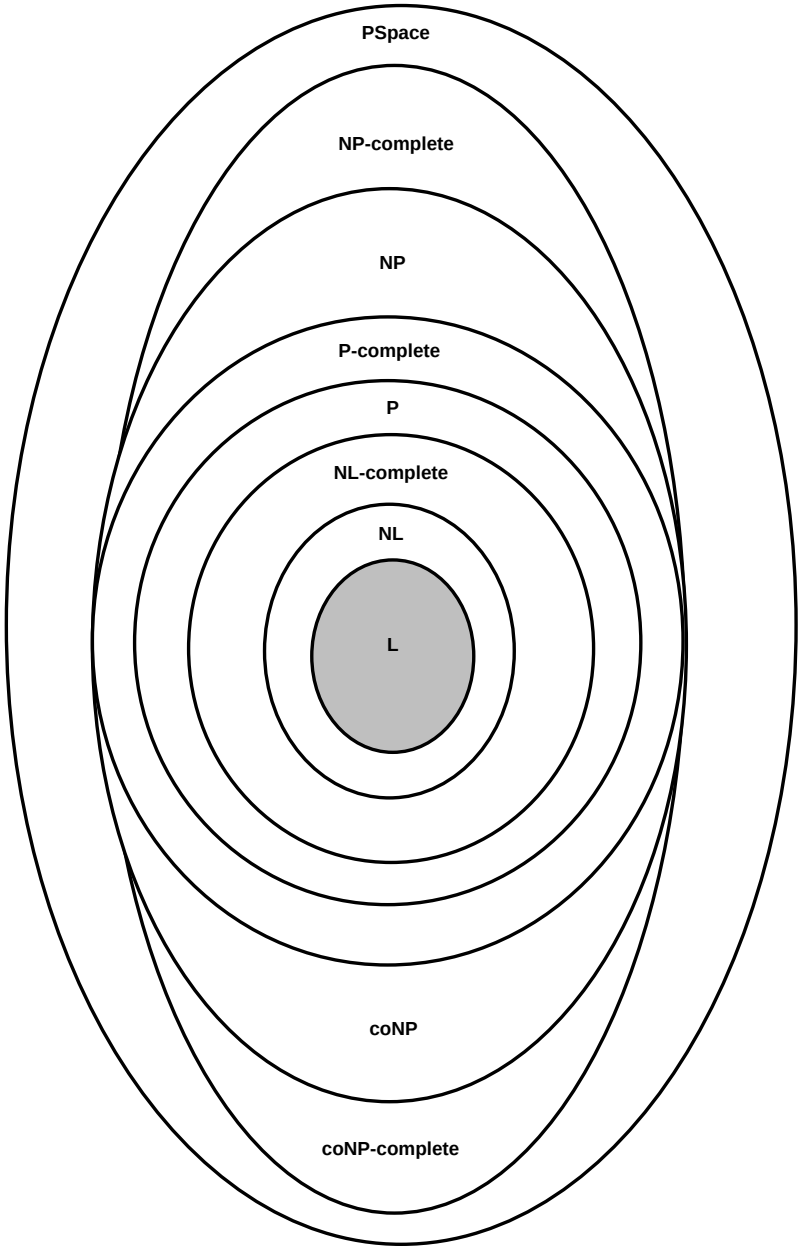
- If inclusions are strict, a refined classification emerges
- A problem L is said to be:
  1. C-hard if it is as hard as every problem L' in C
  2. C-complete if it is C-hard and also in C
- C-hard problems can be used as "sieves" to separate classes
- Problems that are NP-hard are "intrinsecally" intractable
- Similarly, we can define the complement coC of C as $\mathrm{coC} := \{\{0,1\}^* \setminus L \mid L \text{ in } \mathrm{C}\}$

NB: By "L as hard as L'" we understand that there exists a Turing machine M that reduces L to L' using logarithmic space!

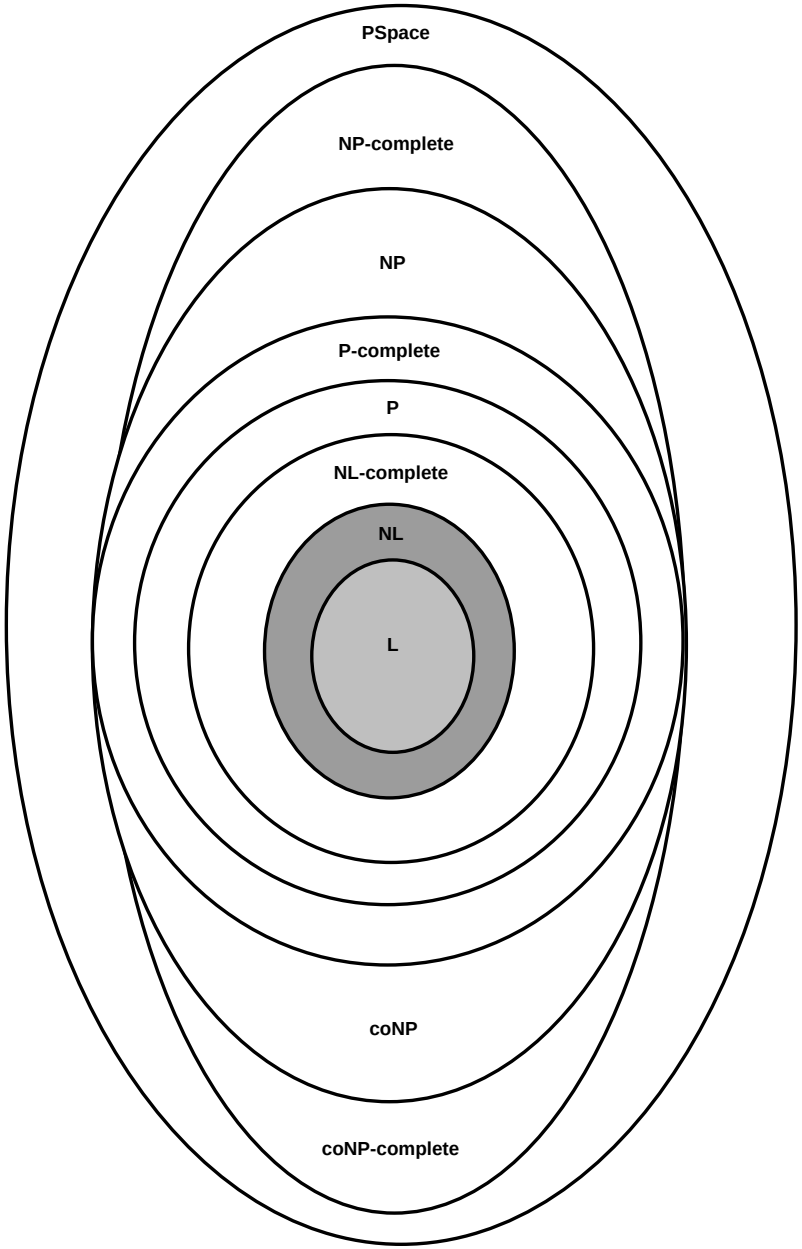
C-hard and C-complete Problems (cntd.)



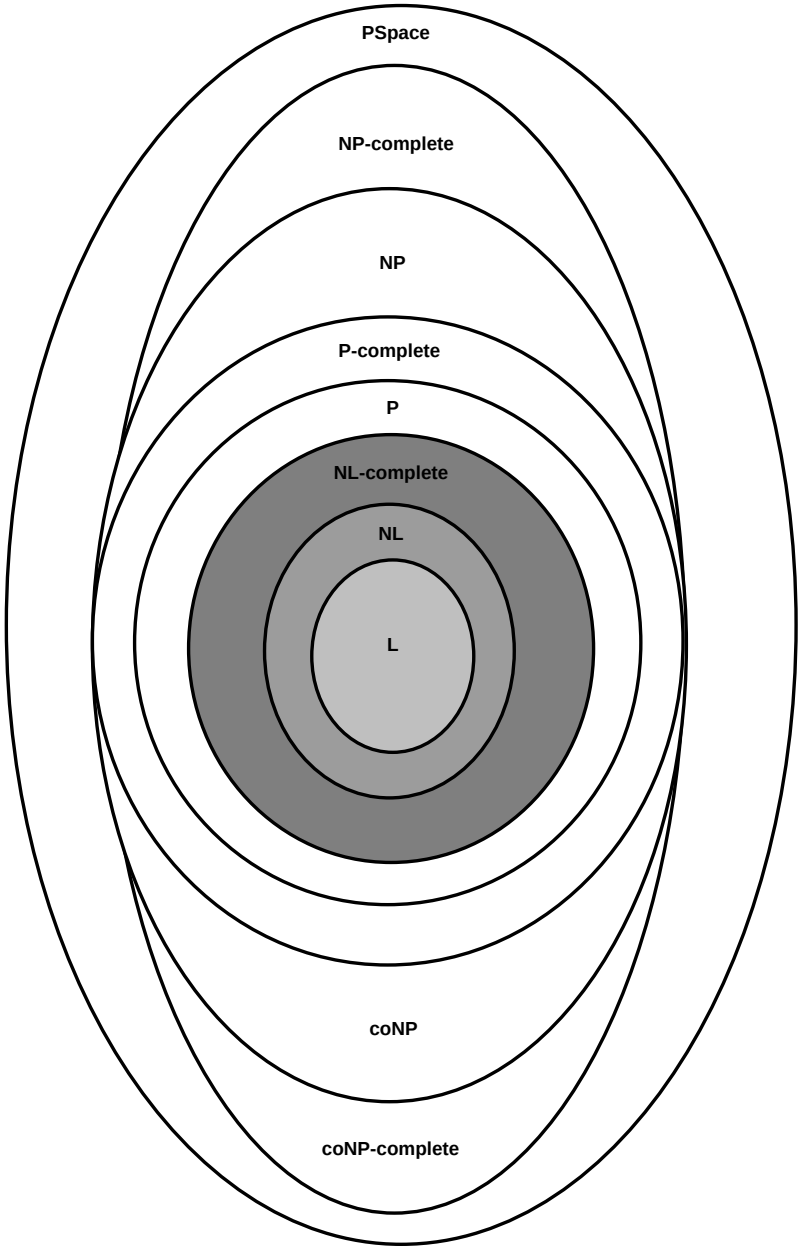
C-hard and C-complete Problems (cntd.)



C-hard and C-complete Problems (cntd.)

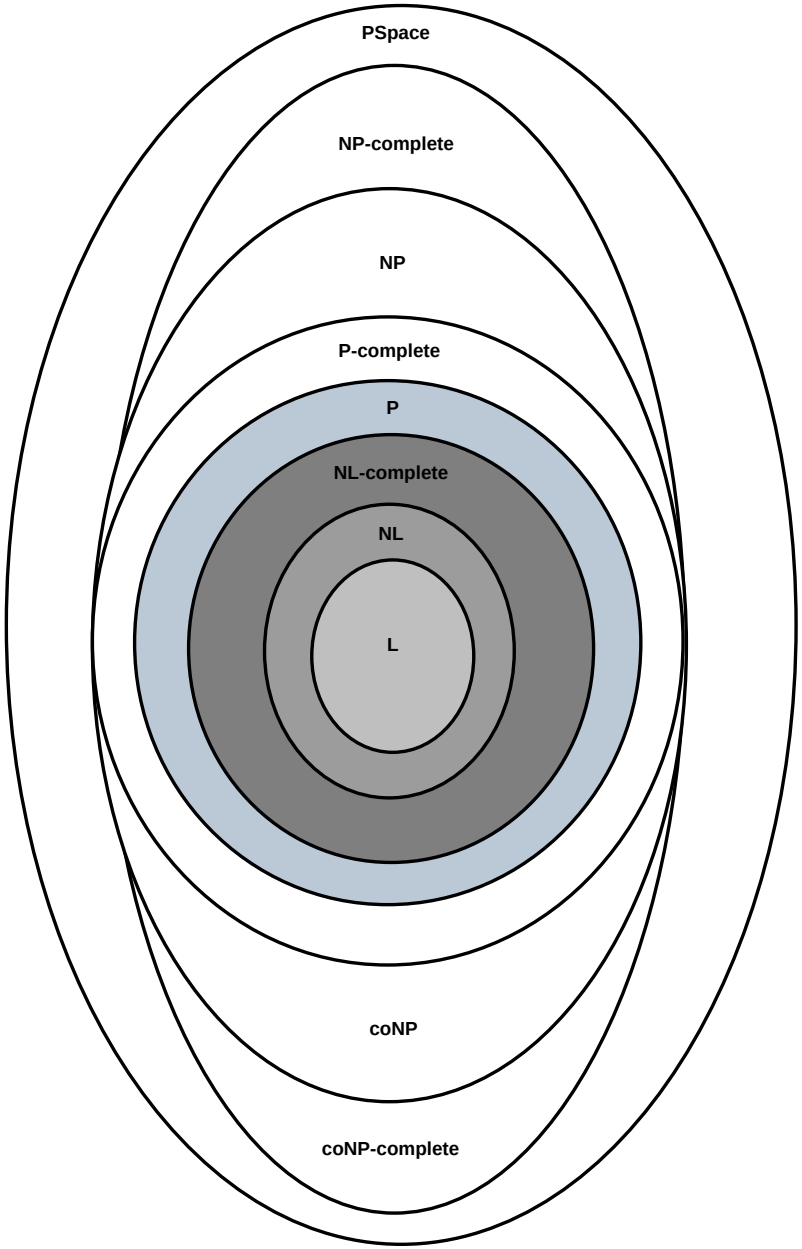


C-hard and C-complete Problems (cntd.)



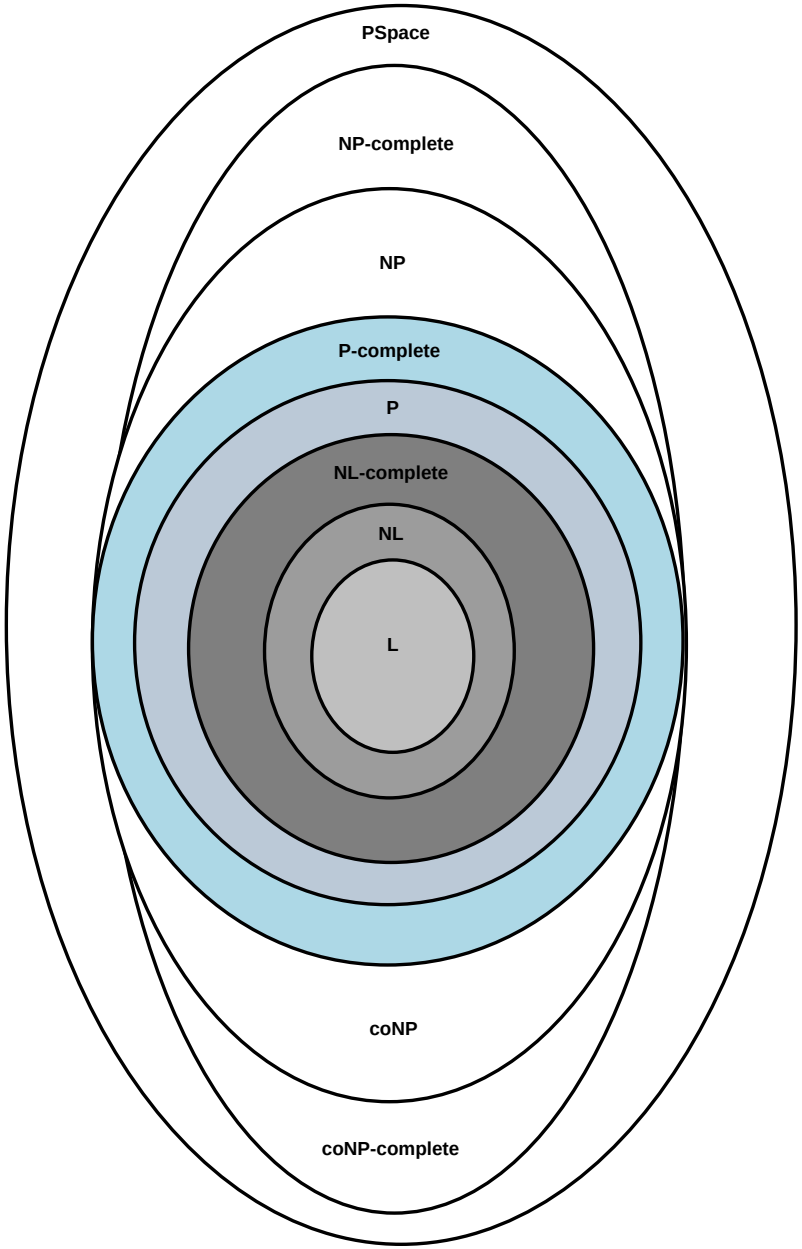
- NL-complete (or hard) problems are "intrinsecally" recursive

C-hard and C-complete Problems (cntd.)



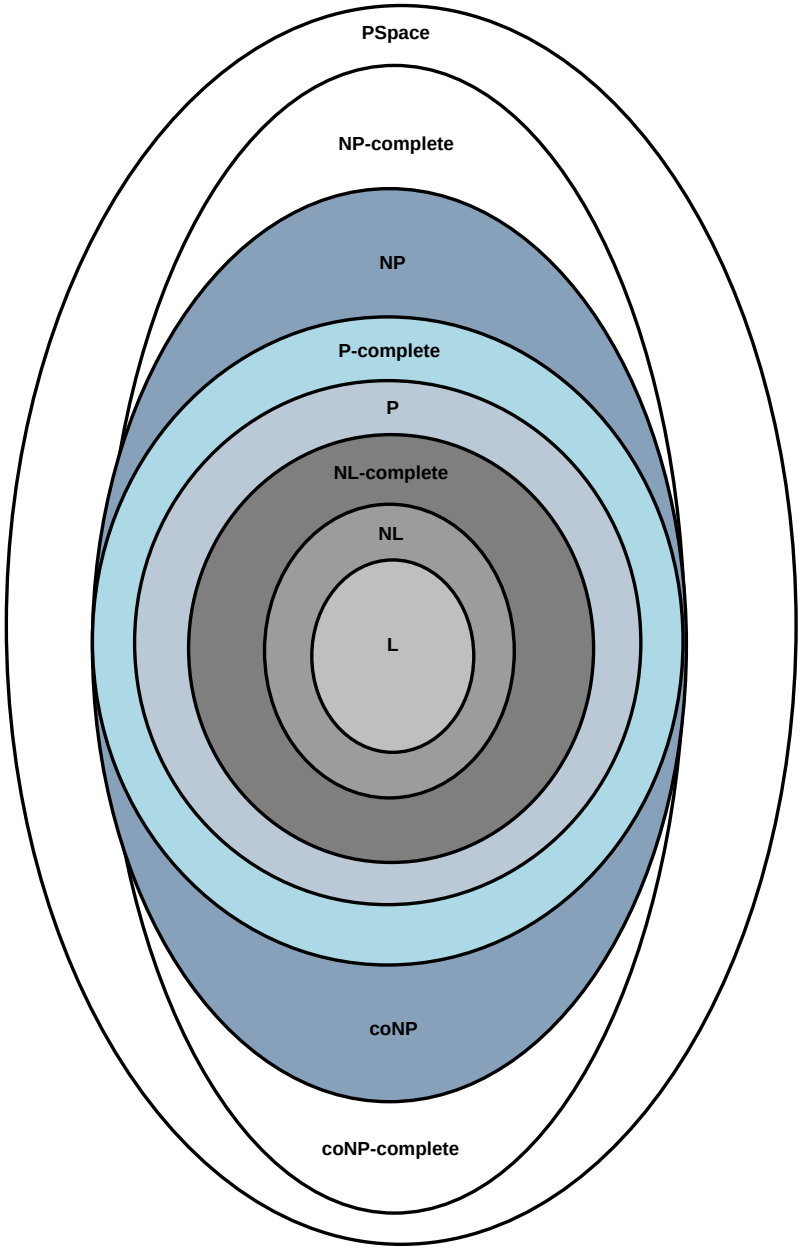
- NL-complete (or hard) problems are "intrinsecally" recursive

C-hard and C-complete Problems (cntd.)



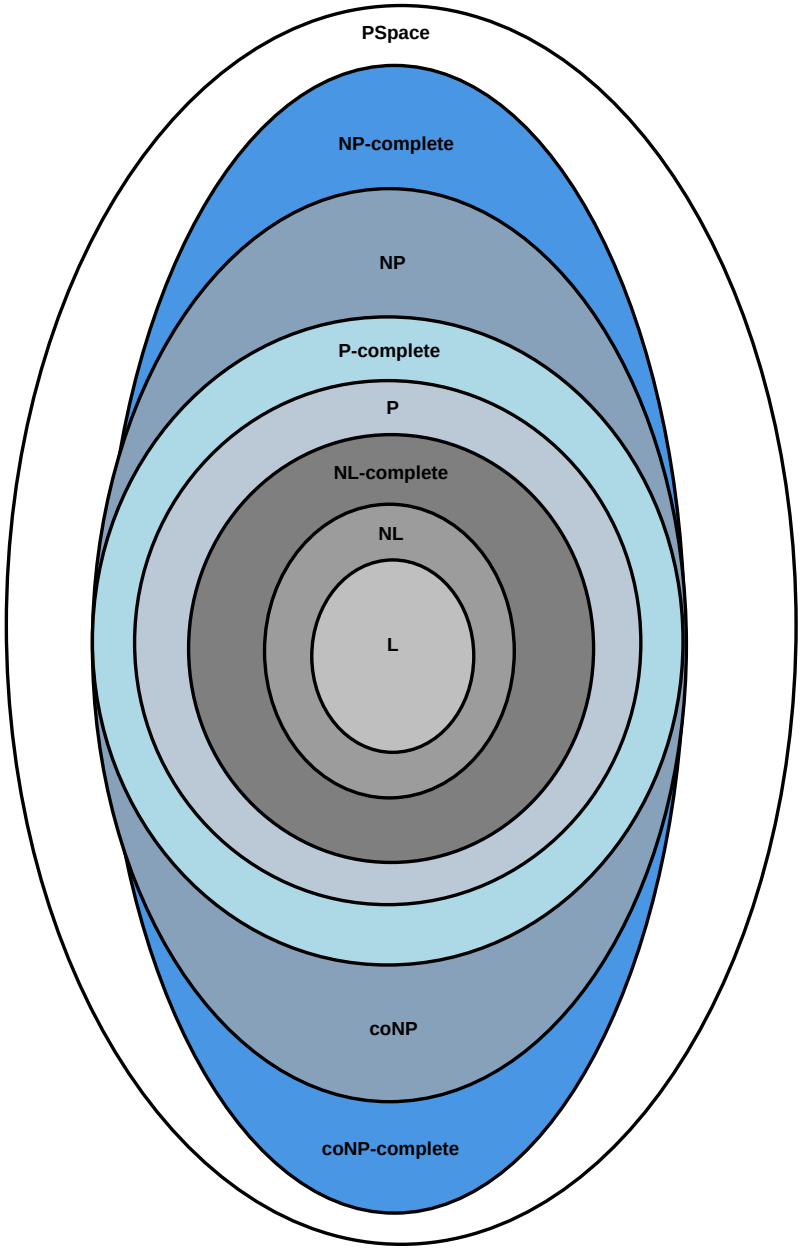
- NL-complete (or hard) problems are "intrinsecally" recursive
- P-complete (or hard) problems are "intrinsecally" non-parallelaizable

C-hard and C-complete Problems (cntd.)



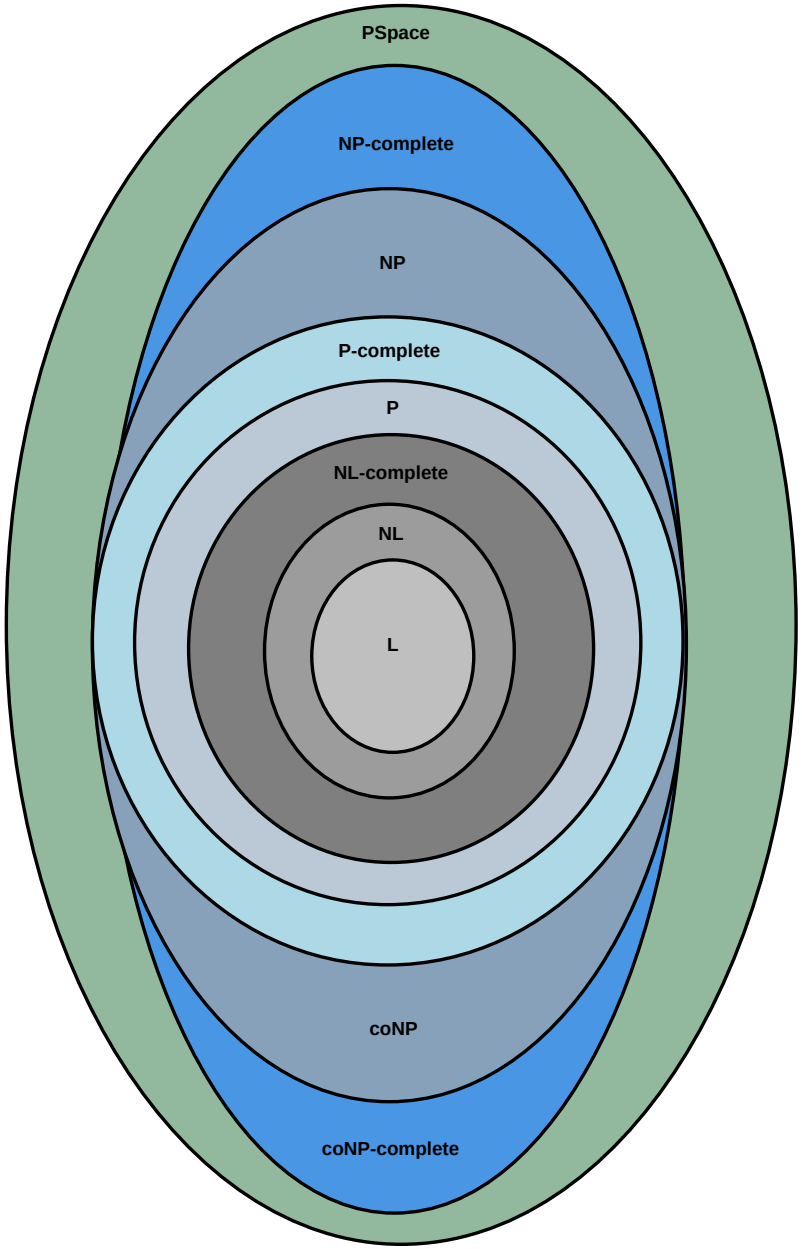
- NL-complete (or hard) problems are "intrinsecally" recursive
- P-complete (or hard) problems are "intrinsecally" non-parallelaizable

C-hard and C-complete Problems (cntd.)



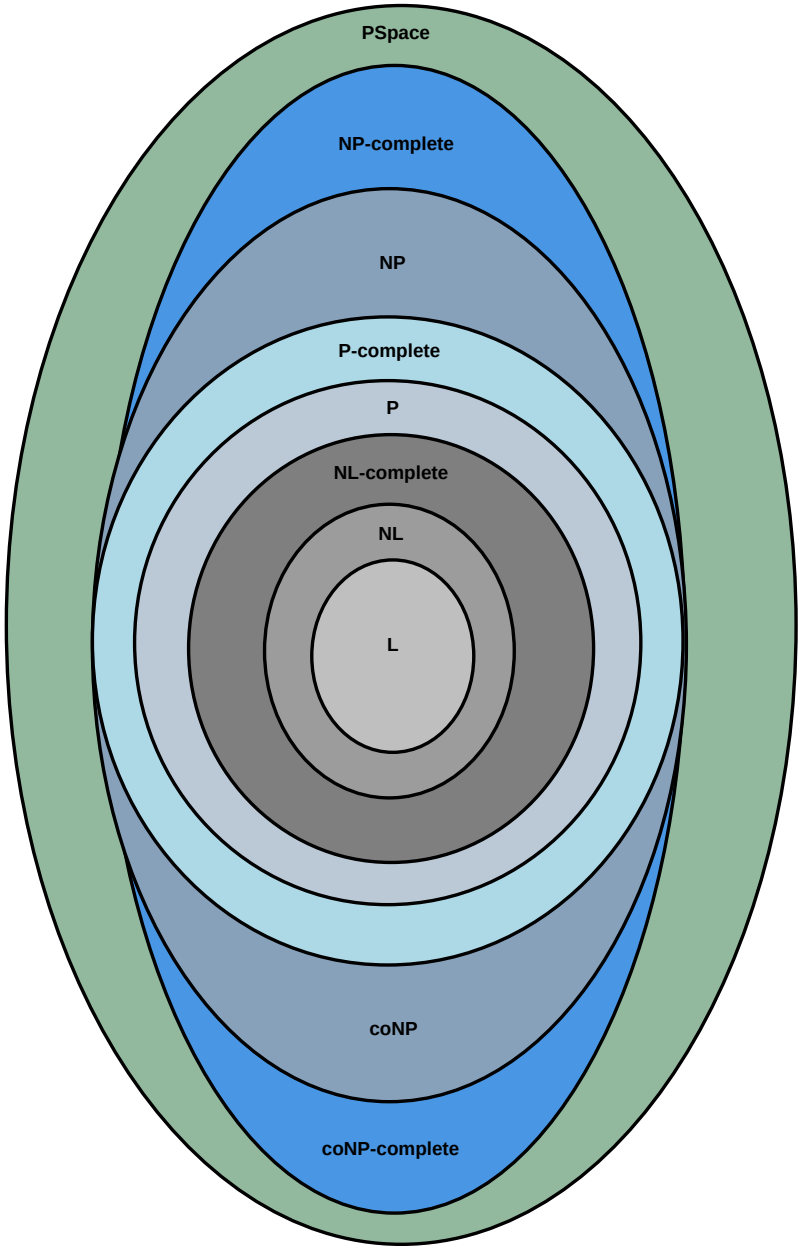
- NL-complete (or hard) problems are "intrinsecally" recursive
- P-complete (or hard) problems are "intrinsecally" non-parallelaizable
- NP-complete (or hard) are "intrinsecally" exponential, but can be easily optimized by heuristics

C-hard and C-complete Problems (cntd.)



- NL-complete (or hard) problems are "intrinsecally" recursive
- P-complete (or hard) problems are "intrinsecally" non-parallelaizable
- NP-complete (or hard) are "intrinsecally" exponential, but can be easily optimized by heuristics
- Beyond PSPACE, optimization by heuristics becomes harder and harder

C-hard and C-complete Problems (cntd.)



- NL-complete (or hard) problems are "intrinsecally" recursive
- P-complete (or hard) problems are "intrinsecally" non-parallelaizable
- NP-complete (or hard) are "intrinsecally" exponential, but can be easily optimized by heuristics
- Beyond PSPACE, optimization by heuristics becomes harder and harder

NB: Many more refinements are possible!

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?
  2. QA: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment and a question Q, does $\tau(S \cup F)$ logically entail $\tau(Q)$?

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?
  2. QA: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment and a question Q, does $\tau(S \cup F)$ logically entail $\tau(Q)$?
- A fine-grained analysis of semantic complexity is possible:

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?
  2. QA: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment and a question Q, does $\tau(S \cup F)$ logically entail $\tau(Q)$?
- A fine-grained analysis of semantic complexity is possible:
  - if measured only in F: data complexity

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?
  2. QA: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment and a question Q, does $\tau(S \cup F)$ logically entail $\tau(Q)$?
- A fine-grained analysis of semantic complexity is possible:
  - if measured only in F: data complexity
  - if measured in F, S and Q: combined complexity

Semantic Complexity

- We are interested in the semantic complexity of a controlled fragment
- Semantic complexity consists in the computational complexity of the following decision problems:
  1. SAT: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment, is $\tau(S \cup F)$ satisfiable?
  2. QA: Given a set of quantified sentences S and non quantified sentences F from a (controlled) fragment and a question Q, does $\tau(S \cup F)$ logically entail $\tau(Q)$?
- A fine-grained analysis of semantic complexity is possible:
  - if measured only in F: data complexity
  - if measured in F, S and Q: combined complexity

NB: This fine-grained analysis is possible if semantic complexity is decidable

Combined Complexity of the FOEs [Pratt & Third 2006]

COP	in NL
COP+TV	NL-complete
COP+TV+DTV	in P
COP+Rel	NP-complete
COP+TV+Rel	EXP-complete
COP+TV+Rel+RA	EXP-complete
COP+TV+DTV+Rel	NEXP-complete
COP+TV+Rel+GA	undecidable

Different function words yield different complexity!

Combined Complexity of the FOEs [Pratt & Third 2006]

COP	in NL
COP+TV	NL-complete
COP+TV+DTV	in P
COP+Rel	NP-complete
COP+TV+Rel	EXP-complete
COP+TV+Rel+RA	EXP-complete
COP+TV+DTV+Rel	NEXP-complete
COP+TV+Rel+GA	undecidable

Not "Boolean-closed" $\Rightarrow$ tractable!

Combined Complexity of the FOEs [Pratt & Third 2006]

COP	in NL
COP+TV	NL-complete
COP+TV+DTV	in P
COP+Rel	NP-complete
COP+TV+Rel	EXP-complete
COP+TV+Rel+RA	EXP-complete
COP+TV+DTV+Rel	NEXP-complete
COP+TV+Rel+GA	undecidable

"Boolean-closed" $\Rightarrow$ intractable!

Combined Complexity of the FOEs [Pratt & Third 2006]

COP	in NL
COP+TV	NL-complete
COP+TV+DTV	in P
COP+Rel	NP-complete
COP+TV+Rel	EXP-complete
COP+TV+Rel+RA	EXP-complete
COP+TV+DTV+Rel	NEXP-complete
COP+TV+Rel+GA	undecidable

"Boolean-closed" + restricted anaphora $\Rightarrow$ decidable!

Combined Complexity of the FOEs [Pratt & Third 2006]

COP	in NL
COP+TV	NL-complete
COP+TV+DTV	in P
COP+Rel	NP-complete
COP+TV+Rel	EXP-complete
COP+TV+Rel+RA	EXP-complete
COP+TV+DTV+Rel	NEXP-complete
COP+TV+Rel+GA	undecidable

"Boolean-closed" + full anaphora $\Rightarrow$ undecidable!

Data Complexity of the FOEs [Thorne & Calvanese 2010]

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Typical scenarios: interfaces to knowledge bases, ontologies
- It allows a substantial increase in expressivity:

	QA	SAT
COP	in L	in L
COP+TV	in P	in L
COP+TV+DTV	in coNP	in L
COP+Rel	coNP-complete	in L
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 [Thorne & Calvanese 2010]

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Typical scenarios: interfaces to knowledge bases, ontologies
- It allows a substantial increase in expressivity:

	QA	SAT
COP	in L	in L
COP+TV	in P	in L
COP+TV+DTV	in coNP	in L
COP+Rel	coNP-complete	in L
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 [Thorne & Calvanese 2010]

- Data complexity [Vardi 1982] measures whether reasoning scales to large data repositories
- Typical scenarios: interfaces to knowledge bases, ontologies
- But "Boolean-closedness" hits in quickly!

	QA	SAT
COP	in L	in L
COP+TV	in P	in L
COP+TV+DTV	in coNP	in L
COP+Rel	coNP-complete	in L
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) Scalable and Non-Scalable Constructs [Thorne 2010]

	Declarations	Questions
Constructs that scale (P 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

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