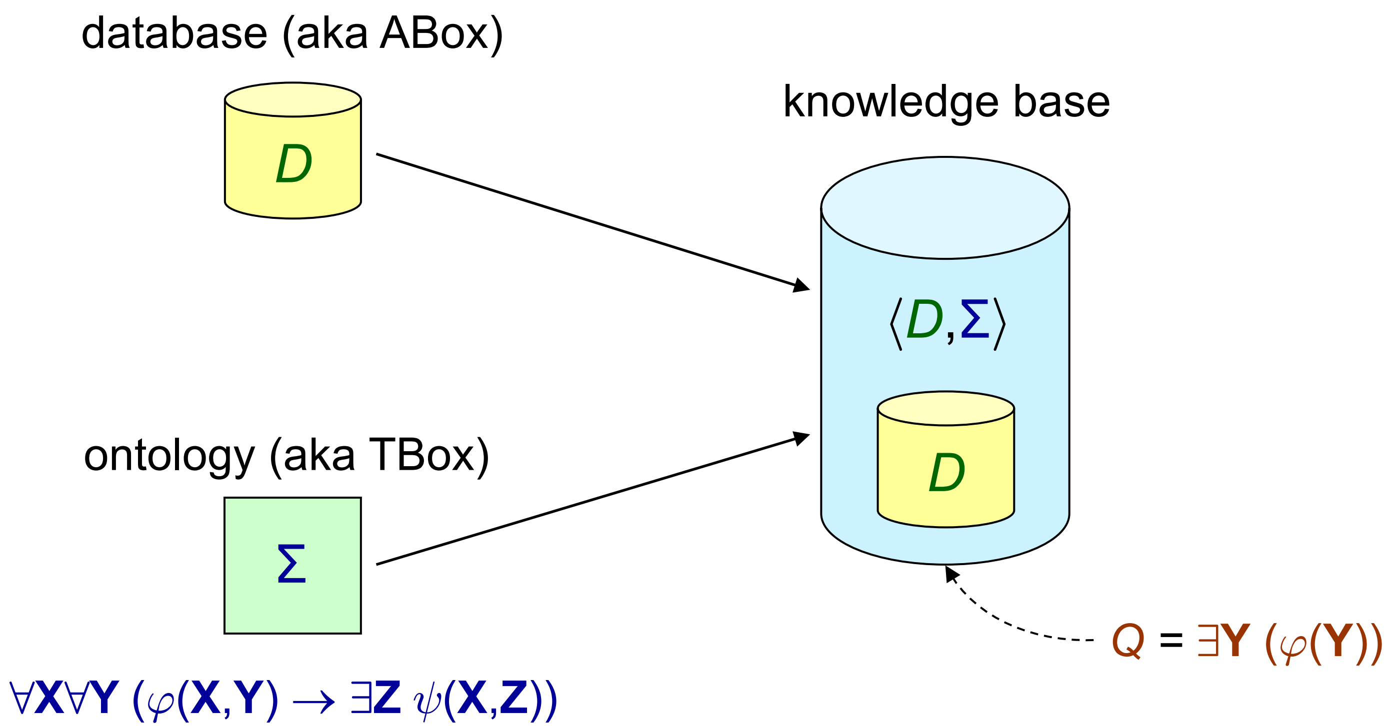
Sebastian Rudolph
International Center for Computational Logic
TU Dresden

Existential Rules – Lecture 6

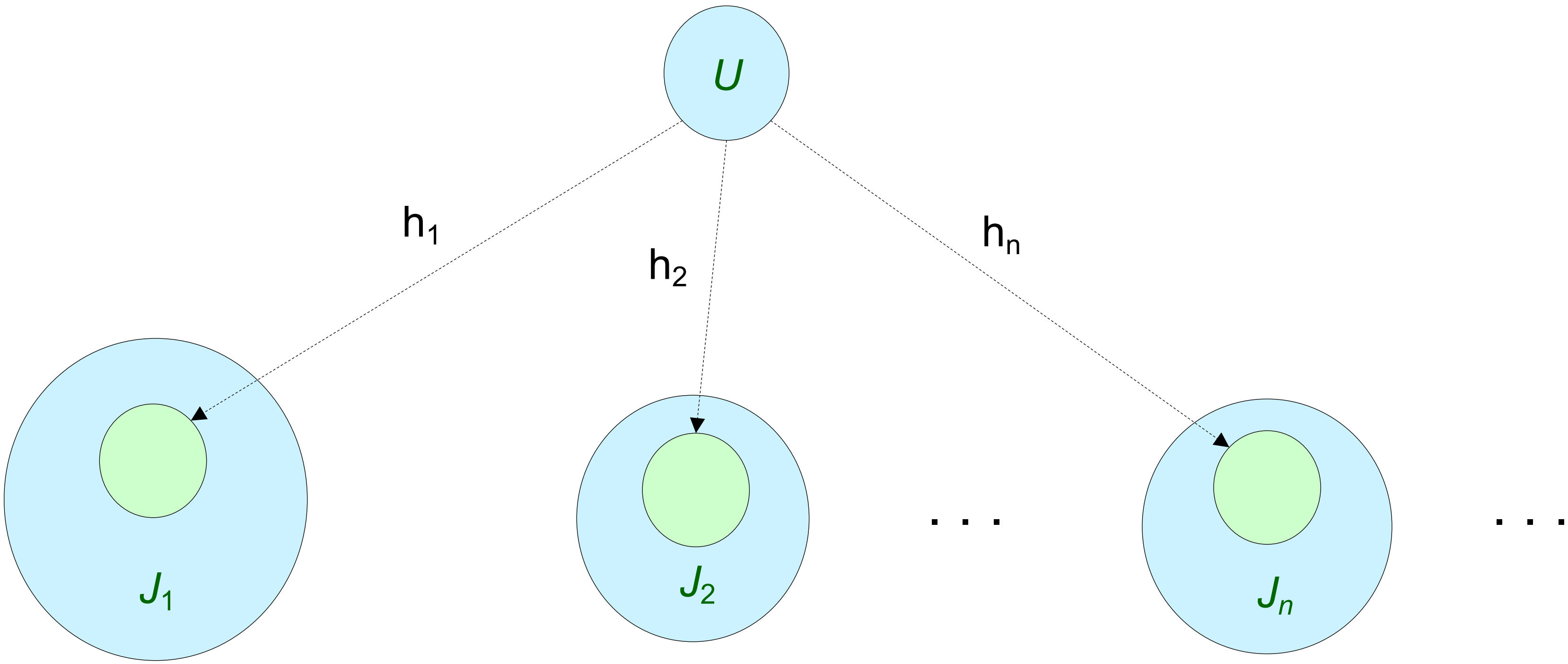
Adapted from slides by Andreas Pieris and Michaël Thomazo
Winter Term 2025/26

BCQ-Answering: Our Main Decision Problem



decide whether $D \wedge \Sigma \models Q$

Universal Models (a.k.a. Canonical Models)



An instance U is a universal model of $D \wedge \Sigma$ if the following holds:

1. U is a model of $D \wedge \Sigma$
2. $\forall J \in \text{models}(D \wedge \Sigma)$, there exists a homomorphism h_J such that $h_J(U) \subseteq J$

Query Answering via the Chase

Theorem: $D \wedge \Sigma \vDash Q$ iff $U \vDash Q$, where U is a universal model of $D \wedge \Sigma$

+

Theorem: chase(D, Σ) is a universal model of $D \wedge \Sigma$

=

Corollary: $D \wedge \Sigma \vDash Q$ iff chase(D,Σ) $\vDash Q$

Undecidability of BCQ-Answering

Theorem: BCQ-Answering is undecidable

Proof : By simulating a deterministic Turing machine with an empty tape

…syntactic restrictions are needed!!!

Termination of the Chase

- Drop the existential quantification
  - We obtain the class of full existential rules
  - Very close to Datalog

  ✓

- Drop the recursive definitions
  - We obtain the class of acyclic existential rules
  - A.k.a. non-recursive existential rules

  ✓

Sum Up

	Data Complexity	
FULL	PTIME-c	Naïve algorithm
		Reduction from Monotone Circuit Value problem
ACYCLIC	in LOGSPACE	Not covered here

	Combined Complexity	
FULL	EXPTIME-c	Naïve algorithm
		Simulation of a deterministic exponential time TM
ACYCLIC	NEXPTIME-c	Small witness property
		Reduction from Tiling problem

Recall our Example

D

person(Alice)

Σ

$\forall X\ (\mathit{Person}(X) \rightarrow \exists Y\ (\mathit{hasParent}(X,Y) \wedge \mathit{Person}(Y)))$

$\text{chase}(D,\Sigma) = D \cup \{\mathit{hasParent}(\text{Alice}, z_1), \mathit{Person}(z_1),$

$\mathit{hasParent}(z_1, z_2), \mathit{Person}(z_2),$

$\mathit{hasParent}(z_2, z_3), \mathit{Person}(z_3), \ldots$

The above rule can be written as the DL-Lite axiom

$\mathit{Person} \sqsubseteq \exists \mathit{hasParent}.\mathit{Person}$

Recall our Example

D

person(Alice)

Σ

$\forall X\ (\mathit{Person}(X) \rightarrow \exists Y\ (\mathit{hasParent}(X,Y) \wedge \mathit{Person}(Y)))$

$$\mathrm{chase}(D,\Sigma) = D \cup \{\mathit{hasParent}(\mathrm{Alice}, z_1), \mathit{Person}(z_1), \mathit{hasParent}(z_1, z_2), \mathit{Person}(z_2), \mathit{hasParent}(z_2, z_3), \mathit{Person}(z_3), \ldots$$

Existential quantification & recursive definitions

are key features for modelling ontologies

Challenge

We need classes of existential rules such that

- Existential quantification and recursive definition coexist

  $\Rightarrow$ the chase may be infinite

- BCQ-Answering is decidable, and tractable w.r.t. the data complexity

$$\Downarrow$$

Tame the infinite chase:

Deal with infinite structures without explicitly building them

Linear Existential Rules

- A linear existential rule is an existential rule of the form

$$\forall X \forall Y\, (P(X,Y) \to \exists Z\, \psi(X,Z))$$

  where $P(X,Y)$ is an atom

- We denote LINEAR the class of linear existential rules

- A local property - we can inspect one rule at a time
  - $\Rightarrow$ given Σ, we can decide in linear time whether $\Sigma \in$ LINEAR
  - $\Rightarrow \Sigma_1 \in$ LINEAR, $\Sigma_2 \in$ LINEAR $\Rightarrow (\Sigma_1 \cup \Sigma_2) \in$ LINEAR

- Strictly more expressive than DL-Lite

LINEAR vs. DL-Lite

Existential rules and DLs rely on first-order semantics - comparable formalisms

DL-Lite Axioms	Existential Rules
$A \sqsubseteq B$	$\forall X\ (A(X) \rightarrow B(X))$
$A \sqsubseteq \exists R$	$\forall X\ (A(X) \rightarrow \exists Y\ R(X,Y))$
$\exists R \sqsubseteq A$	$\forall X \forall Y\ (R(X,Y) \rightarrow A(X))$
$\exists R \sqsubseteq \exists P$	$\forall X \forall Y\ (R(X,Y) \rightarrow \exists Z\ P(X,Z))$
$A \sqsubseteq \exists R.B$	$\forall X\ (A(X) \rightarrow \exists Y\ (R(X,Y) \wedge B(Y)))$
$R \sqsubseteq P$	$\forall X \forall Y\ (R(X,Y) \rightarrow P(X,Y))$
$A \sqsubseteq \neg B$	$\forall X\ (A(X) \wedge B(X) \rightarrow \bot)$

Linear Existential Rules

- A linear existential rule is an existential rule of the form

$$\forall X \forall Y\ (P(X,Y) \rightarrow \exists Z\ \psi(X,Z))$$

  where $P(X,Y)$ is an atom (which is trivially a guard)

- We denote LINEAR the class of linear existential rules

- A local property - we can inspect one rule at a time
  - $\Rightarrow$ given Σ, we can decide in linear time whether $\Sigma \in$ LINEAR
  - $\Rightarrow$ $\Sigma_1 \in$ LINEAR, $\Sigma_2 \in$ LINEAR $\Rightarrow (\Sigma_1 \cup \Sigma_2) \in$ LINEAR

- Strictly more expressive than DL-Lite

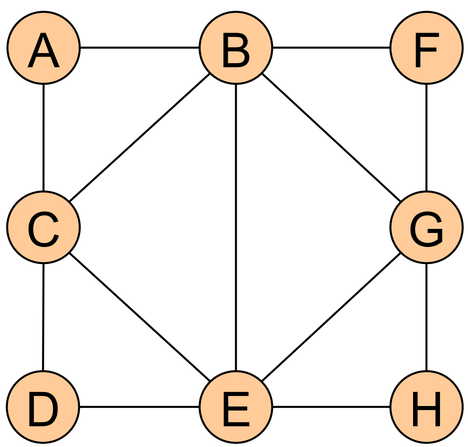
- Infinite chase - $\forall X\ (\mathit{Person}(X) \rightarrow \exists Y\ (\mathit{hasParent}(X,Y) \wedge \mathit{Person}(Y)))$

- But, BCQ-Answering is decidable - the chase has finite treewidth

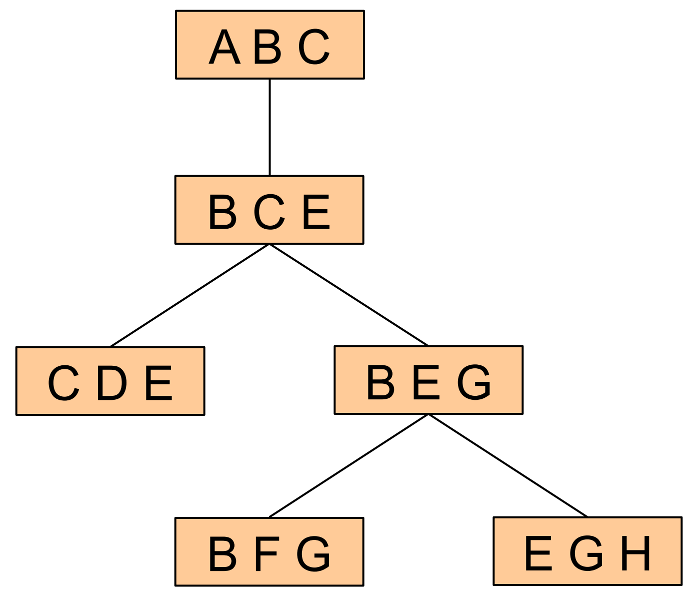
Treewidth of a Graph

Tree decomposition - a mapping of a graph into a tree

Graph G = (V,E)



Tree decomposition T = (V',E') of G

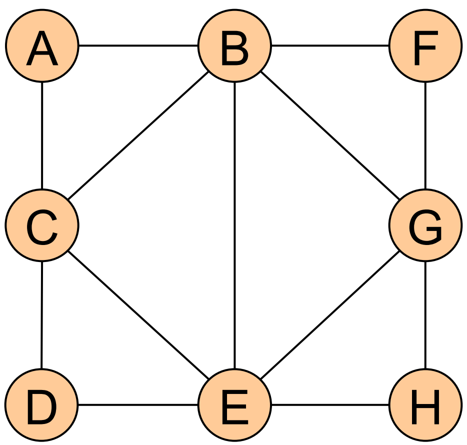


1. For each $v \in V$, there exists $u \in V'$ such that $v \in u$
2. For each $(v,w) \in E$, there exists $u \in V'$ such that $\{v,w\} \subseteq u$
3. For each $v \in V$, $\{u \mid v \in u\}$ induces a connected subtree

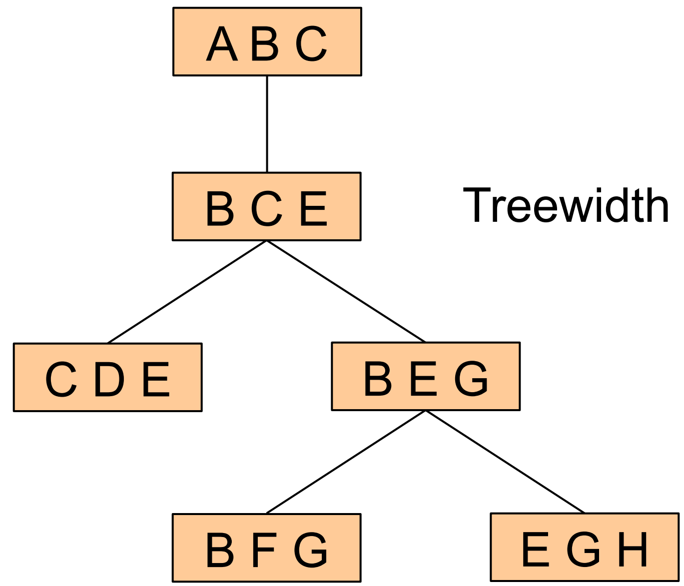
Treewidth of a Graph

Tree decomposition - a mapping of a graph into a tree

Graph G = (V,E)



Tree decomposition T = (V',E') of G



Treewidth = 2

- The width of T is defined as $\max_{u \in V'} \{|u|\} - 1$
- The treewidth of G, denoted tw(G), is the minimum width over all tree decompositions of G

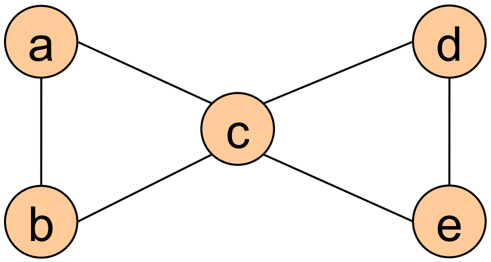
Treewidth of an Instance

- An instance J can be represented as a graph $\mathcal{G}_J$ - Gaifman graph

R(a,b,c)

S(c,d)

T(c,d,e)



- The treewidth of J, denoted tw(J), is defined as tw($\mathcal{G}_J$)

- Thus, we can talk about the treewidth of the chase

- Lemma: For a database D, and a set $\Sigma \in$ LINEAR, tw(chase(D,Σ)) is finite

Decidability of LINEAR

Theorem: BCQ-Answering under LINEAR is decidable

Proof: The ingredients of the proof are the following:

1. The chase under LINEAR has finite treewidth
2. The tree model property implies decidability of satisfiability - classical result

A fragment $\mathcal{L}$ of first-order logic enjoys the tree model property if: for every $\varphi \in \mathcal{L}$,

if φ is satisfiable, then there exists $J \in$ models(φ) such that tw(J) is finite

Decidability of LINEAR

Theorem: BCQ-Answering under LINEAR is decidable

Proof: The ingredients of the proof are the following:

1. The chase under LINEAR has finite treewidth
2. The tree model property implies decidability of satisfiability - classical result

- Consider a database D, a set $\Sigma \in$ LINEAR, and a BCQ Q
- Clearly, $D \wedge \Sigma \vDash Q$ iff $D \wedge \Sigma \wedge \neg Q \vDash \bot$
- Thus, it suffices to show that, if $D \wedge \Sigma \wedge \neg Q$ is satisfiable, then it has a model of finite treewidth
- By universality, $D \wedge \Sigma \wedge \neg Q$ is satisfiable implies chase(D,Σ) $\wedge \neg Q$ is satisfiable
- Therefore, $D \wedge \Sigma \wedge \neg Q$ is satisfiable implies chase(D,Σ) is a model of $D \wedge \Sigma \wedge \neg Q$
- The claim follows since tw(chase(D,Σ)) is finite

Decidability of LINEAR

Theorem: BCQ-Answering under LINEAR is decidable

Proof: The ingredients of the proof are the following:

1. The chase under LINEAR has finite treewidth
2. The tree model property implies decidability of satisfiability - classical result

…but, what about the complexity of the problem?

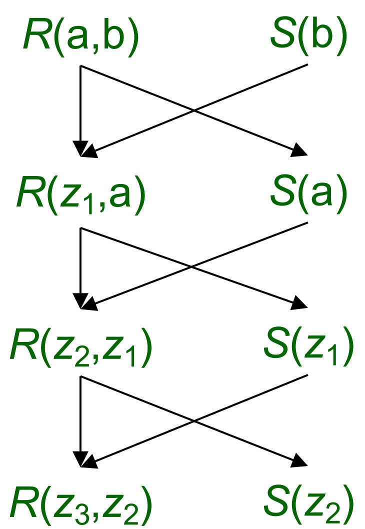
we need new techniques

Chase Graph

The chase can be naturally seen as a graph - chase graph

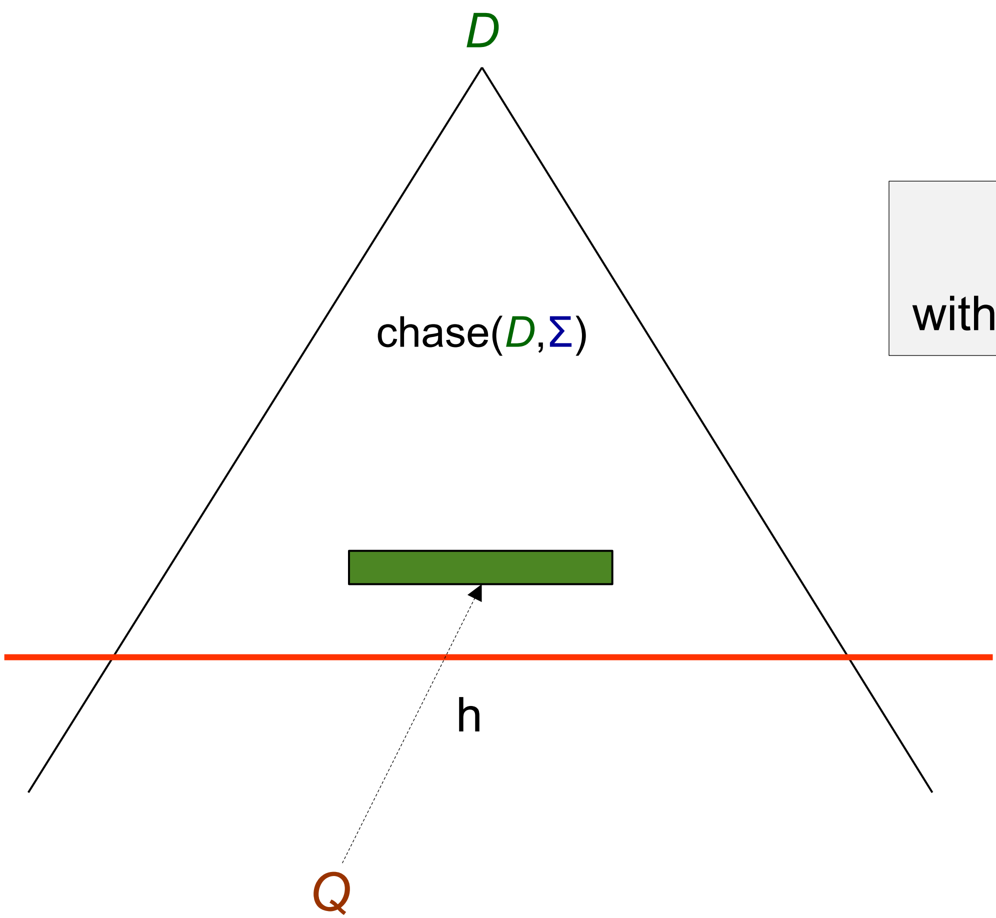
$D = \{R(a,b), S(b)\}$

$$\Sigma = \begin{cases} \forall X \forall Y\, (R(X,Y) \wedge S(Y) \rightarrow \exists Z\, R(Z,X)) \\ \forall X \forall Y\, (R(X,Y) \rightarrow S(X)) \end{cases}$$



For LINEAR, the chase graph is a forest

Bounded Derivation-Depth Property



For LINEAR, $k = |Q| \cdot m$

with $m = |\mathrm{sch}(\Sigma)| \cdot (2 \cdot \mathrm{maxarity})^{\mathrm{maxarity}}$

depth k

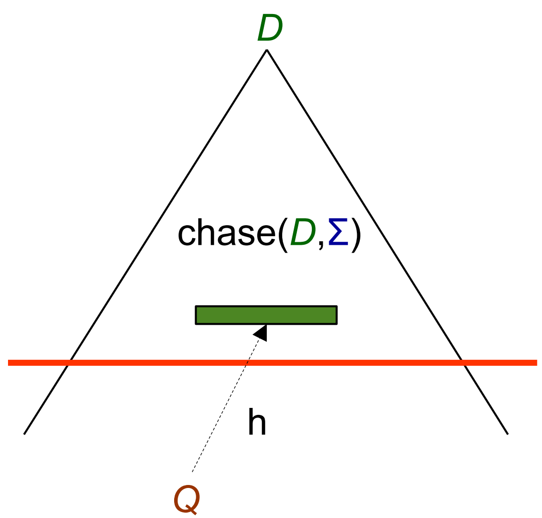
k does not depend on D

chase graph up to depth k

$$\mathrm{chase}(D,\Sigma) \models Q \quad \Rightarrow \quad \mathrm{chase}^k(D,\Sigma) \models Q$$

The Blocking Algorithm for **LINEAR**

- The blocking algorithm shows that BCQ-Answering under **LINEAR** is
    - in PTIME w.r.t. the data complexity
    - in 2EXPTIME w.r.t. the combined complexity



…we can do better than the blocking algorithm

$k = |Q| \cdot |\mathrm{sch}(\Sigma)| \cdot (2 \cdot \mathrm{maxarity})^{\mathrm{maxarity}}$

Data Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in LOGSPACE w.r.t. the data complexity

Proof: Not so easy! Different techniques must be applied. This will be the subject of the second part of our course.

Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in NEXPTIME w.r.t. the combined complexity

Proof: We first need to establish the so-called small witness property

Small Witness Property for LINEAR

Lemma: chase(D,Σ) $\models$ Q $\Rightarrow$ there exists a chase sequence

$$D\langle\sigma_1,h_1\rangle J_1\langle\sigma_2,h_2\rangle J_2\langle\sigma_3,h_3\rangle J_3 \ldots \langle\sigma_n,h_n\rangle J_n$$

of D w.r.t. Σ with

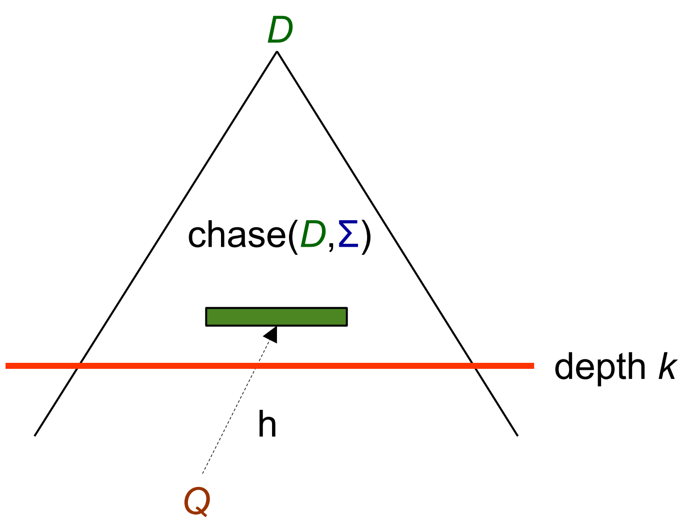
$$n = (|Q|)^2 \cdot |\mathrm{sch}(\Sigma)| \cdot (2 \cdot \mathrm{maxarity})^{\mathrm{maxarity}}$$

such that $J_n \models Q$

Proof:

- By hypothesis, there exists a homomorphism h such that h(Q) $\subseteq$ chase(D, Σ)
- By the bounded-derivation depth property

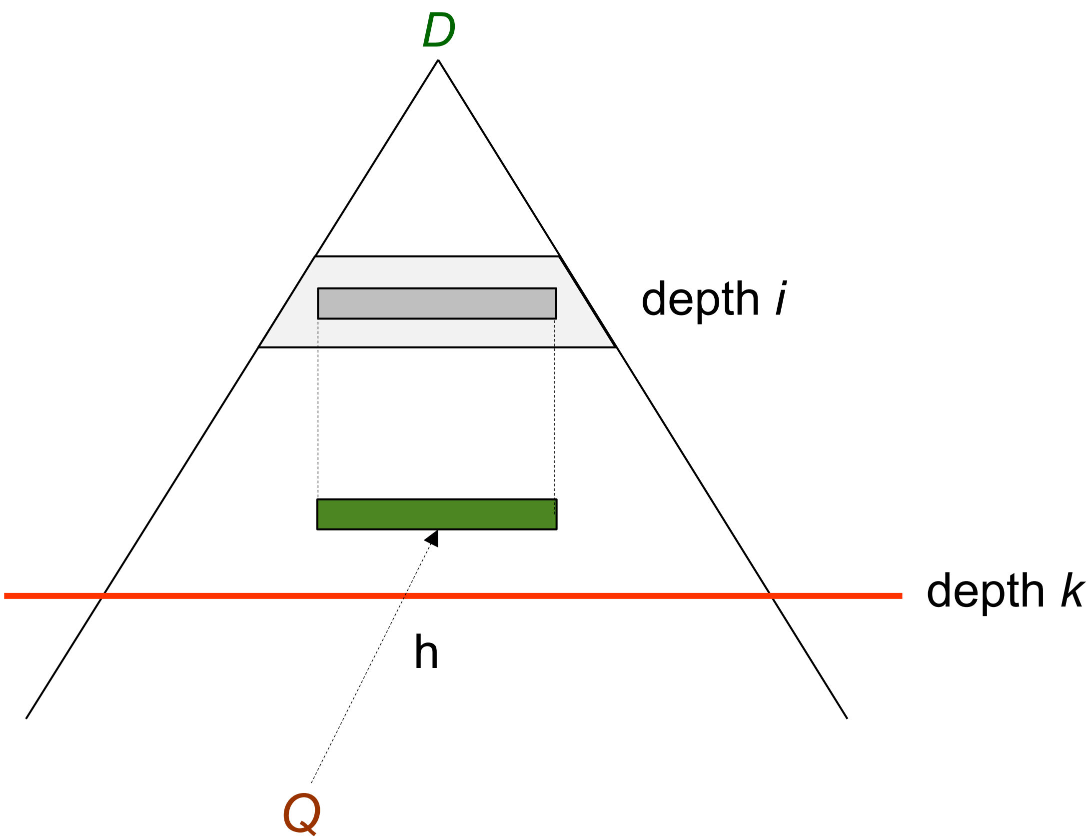
$$k = |Q| \cdot |\mathrm{sch}(\Sigma)| \cdot (2 \cdot \mathrm{maxarity})^{\mathrm{maxarity}}$$



Small Witness Property for LINEAR

Proof (cont.):

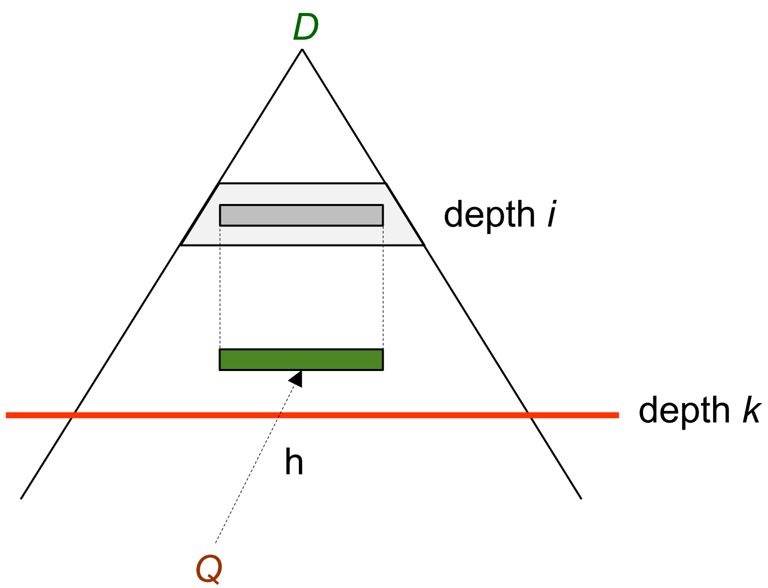
- Let us focus on depth *i* of the chase graph
- How many atoms do we need?
- No more than $|Q|$



Small Witness Property for LINEAR

Proof (cont.):

- Let us focus on depth *i* of the chase graph
- How many atoms do we need?
- No more than $|Q|$
- Thus, to entail the query we need at most



$$k \cdot |Q|$$

$$= |Q| \cdot |sch(\Sigma)| \cdot (2 \cdot maxarity)^{maxarity} \cdot |Q|$$

$$= (|Q|)^2 \cdot |sch(\Sigma)| \cdot (2 \cdot maxarity)^{maxarity}$$

Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in NEXPTIME w.r.t. the combined complexity

Proof: Consider a database D, a set $\Sigma \in$ LINEAR, and a BCQ Q

Having the small witness property in place, we can exploit the following algorithm:

1. Non-deterministically construct a chase sequence

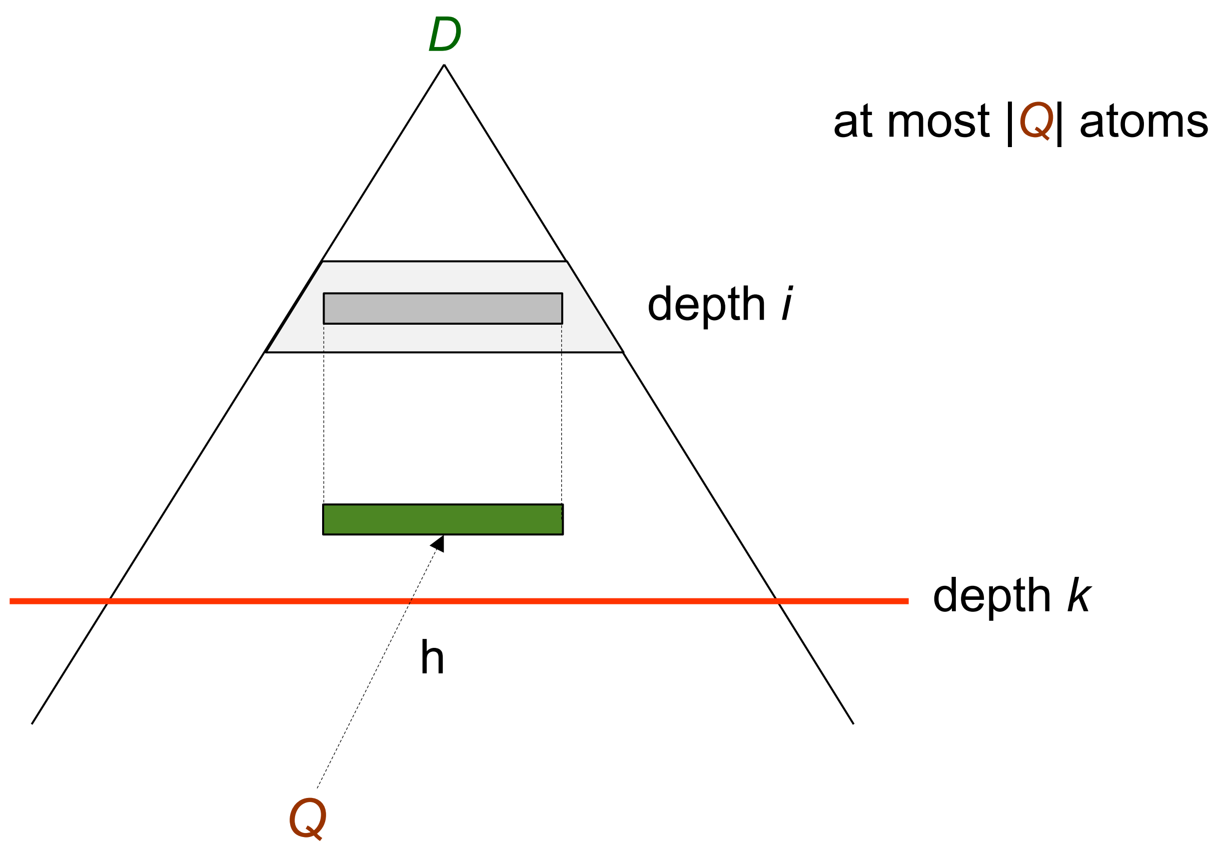
   $$D\langle\sigma_1,h_1\rangle J_1\langle\sigma_2,h_2\rangle J_2\langle\sigma_3,h_3\rangle J_3 \ \ldots\ \langle\sigma_n,h_n\rangle J_n$$

   of D w.r.t. Σ with $n = (|Q|)^2 \cdot |sch(\Sigma)| \cdot (2 \cdot \text{maxarity})^{\text{maxarity}}$

2. Check for the existence of a homomorphism h such that $h(Q) \subseteq J_n$

Can we do better? Any ideas?

Key Observation

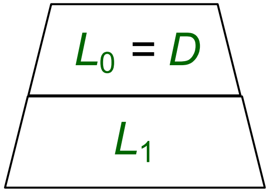


level-by-level construction

Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in PSPACE w.r.t. the combined complexity

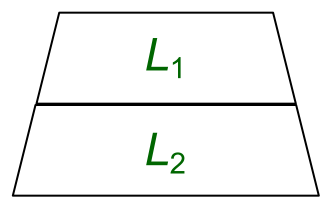
Proof:



Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in PSPACE w.r.t. the combined complexity

Proof:



Combined Complexity of LINEAR

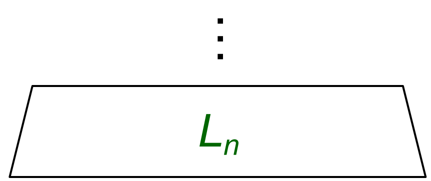
Theorem: BCQ-Answering under LINEAR is in PSPACE w.r.t. the combined complexity

Proof:

Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in PSPACE w.r.t. the combined complexity

Proof:



Combined Complexity of LINEAR

Theorem: BCQ-Answering under LINEAR is in PSPACE w.r.t. the combined complexity

Proof (cont.):

At each step we need to maintain

- $O(|Q|)$ atoms
- A counter $ctr \leq (|Q|)^2 \cdot |\mathrm{sch}(\Sigma)| \cdot (2 \cdot \mathrm{maxarity})^{\mathrm{maxarity}}$

- Thus, we need polynomial space

- The claim follows since NPSPACE = PSPACE

Combined Complexity of LINEAR

We cannot do better than the previous algorithm

Theorem: BCQ-Answering under LINEAR is PSPACE-hard w.r.t. the combined complexity

Proof : By simulating a deterministic polynomial space Turing machine

PSPACE-hardness of LINEAR

Our Goal: Encode the polynomial space computation of a DTM M on input string I using a database D, a set $\Sigma \in$ LINEAR, and a BCQ Q such that

$D \wedge \Sigma \vDash Q$ iff M accepts I using at most $n = (|I|)^k$ cells

PSPACE-hardness of LINEAR

- Assume that the tape alphabet is $\{0,1,\sqcup\}$
- Suppose that M halts on $I = \alpha_1 \ldots \alpha_m$ using $n = m^k$ cells, for $k > 0$

Initial configuration - the database D

$$\mathit{Config}(\mathrm{s}_{\mathrm{init}},\alpha_1,\ldots,\alpha_m,\underbrace{\sqcup,\ldots,\sqcup}_{n\,-\,m},1,\underbrace{0,\ldots,0}_{n\,-\,1})$$

PSPACE-hardness of LINEAR

- Assume that the tape alphabet is $\{0,1,\sqcup\}$
- Suppose that M halts on $I = \alpha_1 \ldots \alpha_m$ using $n = m^k$ cells, for $k > 0$

Transition rule - $\delta(s_1,\alpha) = (s_2,\beta,+1)$

for each $i \in \{1,\ldots,n\}$:

$$\forall X\ (\mathit{Config}(s_1,X_1,\ldots,X_{i-1},\alpha,X_{i+1},\ldots,X_n,\overbrace{0,\ldots,0}^{i-1},1,\overbrace{0,\ldots,0}^{n-i}) \rightarrow$$

$$\mathit{Config}(s_2,X_1,\ldots,X_{i-1},\beta,X_{i+1},\ldots,X_n,\underbrace{0,\ldots,0}_{i},1,\underbrace{0,\ldots,0}_{n-i-1}))$$

PSPACE-hardness of LINEAR

- Assume that the tape alphabet is $\{0,1,\sqcup\}$
- Suppose that *M* halts on $I = \alpha_1 \ldots \alpha_m$ using $n = m^k$ cells, for $k > 0$

$$D \wedge \Sigma \models \exists X\ \mathit{Config}(s_{acc}, X) \quad \text{iff} \quad M \text{ accepts } I$$

…but, the rules are not constant-free

we can eliminate the constants by applying a simple trick

PSPACE-hardness of LINEAR

Initial configuration - the database *D*

auxiliary constants for the states

and the tape alphabet

$$\mathit{Config}(s_{init},\alpha_1,\ldots,\alpha_m,\underbrace{\sqcup,\ldots,\sqcup}_{n-m},1,\underbrace{0,\ldots,0}_{n-1},s_1,\ldots s_\ell,0,1,\sqcup)$$

PSPACE-hardness of LINEAR

Transition rule - $\delta(s_1,0) = (s_2,\sqcup,+1)$

for each $i \in \{1,\ldots,n\}$:

$$Config(S_1,X_1,\ldots,X_{i-1},Z,X_{i+1},\ldots,X_n,\overbrace{Z,\ldots,Z}^{i-1},O,\overbrace{Z,\ldots,Z}^{n-i},S_1,\ldots S_\ell,Z,O,B) \rightarrow$$

$$Config(S_2,X_1,\ldots,X_{i-1},B,X_{i+1},\ldots,X_n,\underbrace{Z,\ldots,Z}_{i},O,\underbrace{Z,\ldots,Z}_{n-i-1},S_1,\ldots S_\ell,Z,O,B)$$

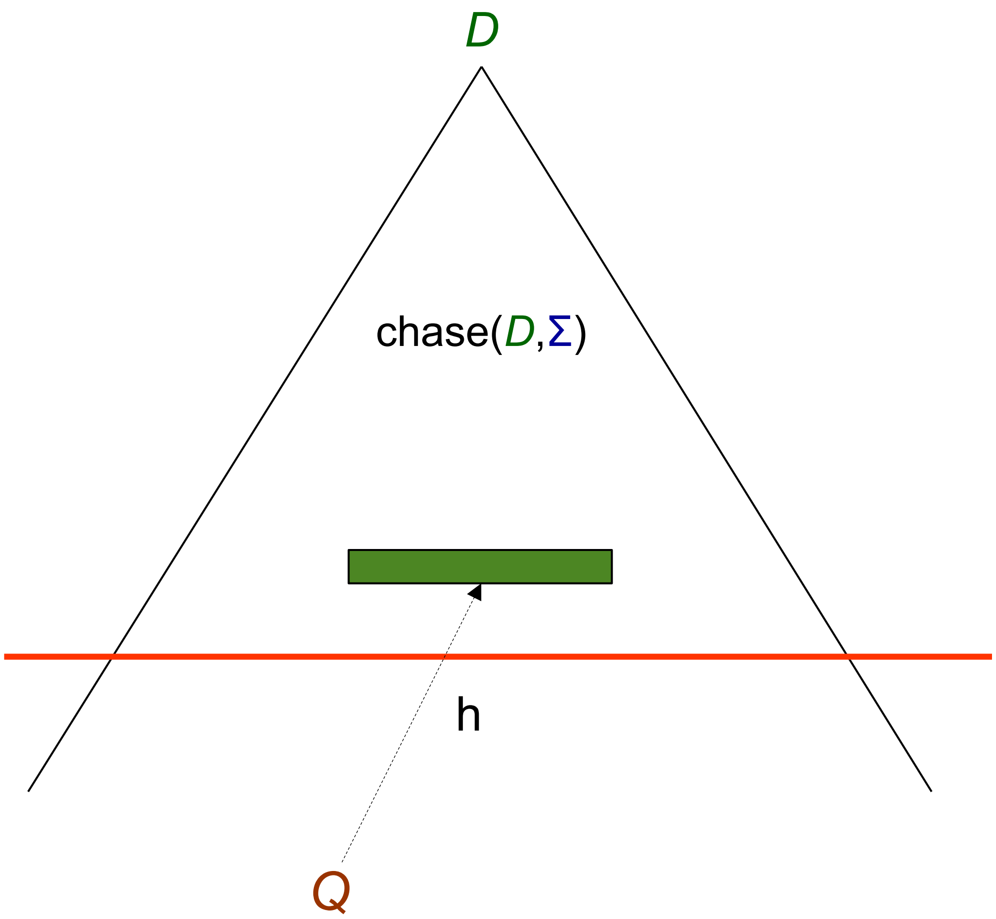
(∀-quantifiers are omitted)

Sum Up

	Data Complexity	
FULL	PTIME-c	Naïve algorithm
		Reduction from Monotone Circuit Value problem
ACYCLIC	in LOGSPACE	**Second part of our course**
LINEAR		

	Combined Complexity	
FULL	EXPTIME-c	Naïve algorithm
		Simulation of a deterministic exponential time TM
ACYCLIC	NEXPTIME-c	Small witness property
		Reduction from Tiling problem
LINEAR	PSPACE-c	Level-by-level non-deterministic algorithm
		Simulation of a deterministic polynomial space TM

Forward Chaining Techniques



Useful techniques for establishing optimal upper bounds

…but not practical - we need to store instances of very large size