Database Theory

Preamble — What is Database Theory?

Matthias Lanzinger / 2025

TU WIEN Informatics

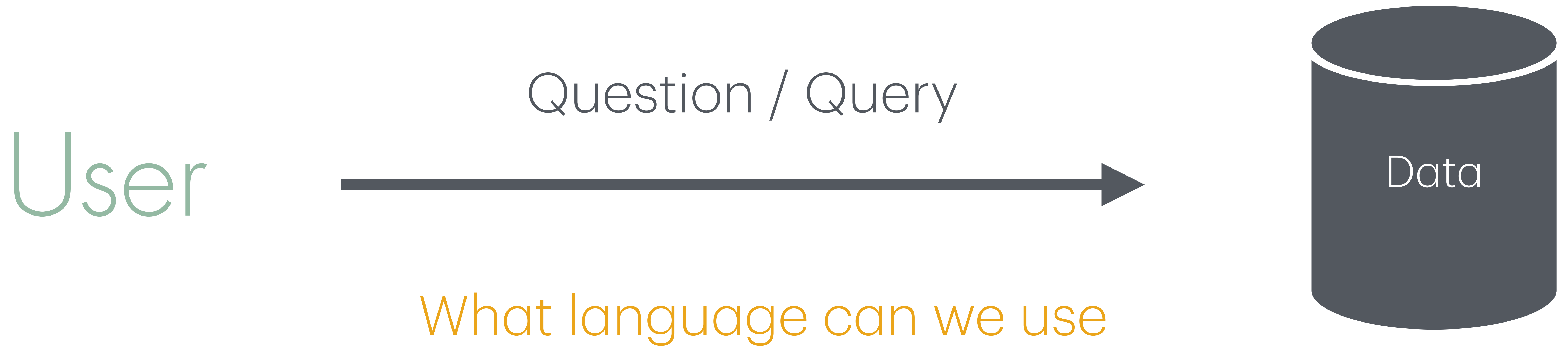
User Side
(very minor role in this course)

Interpretability of queries.
Natural language to formal query language.

What is the “shape” of our data?

Relational, Graph, Text, Semi-structured XML/JSON/etc), ...

Streaming, distributed, privacy preserving, ...



What language can we use to define the question?

Logic, Automata, SQL, GQL, Regular Path Queries, SPARQL, XPATH, GNNs, Document Spanners, ...

Queries

Query Languages

Complexity

How difficult is it to answer queries in the language?

Can we find smart ways to avoid the difficulty?

Expressivity

What kinds of question can be asked with the language?

What properties cannot be queried with a language?

Equivalence

When do two queries always give the same result over every database?

How can we use equivalence for faster execution of queries.

Complexity

- Dependence on only the data, only the query, both, tradeoffs?
- If intractable, can we identify special cases for which there are efficient algorithms?
- Which features of the language are the cause of higher complexity?

  → Influence back on language design.

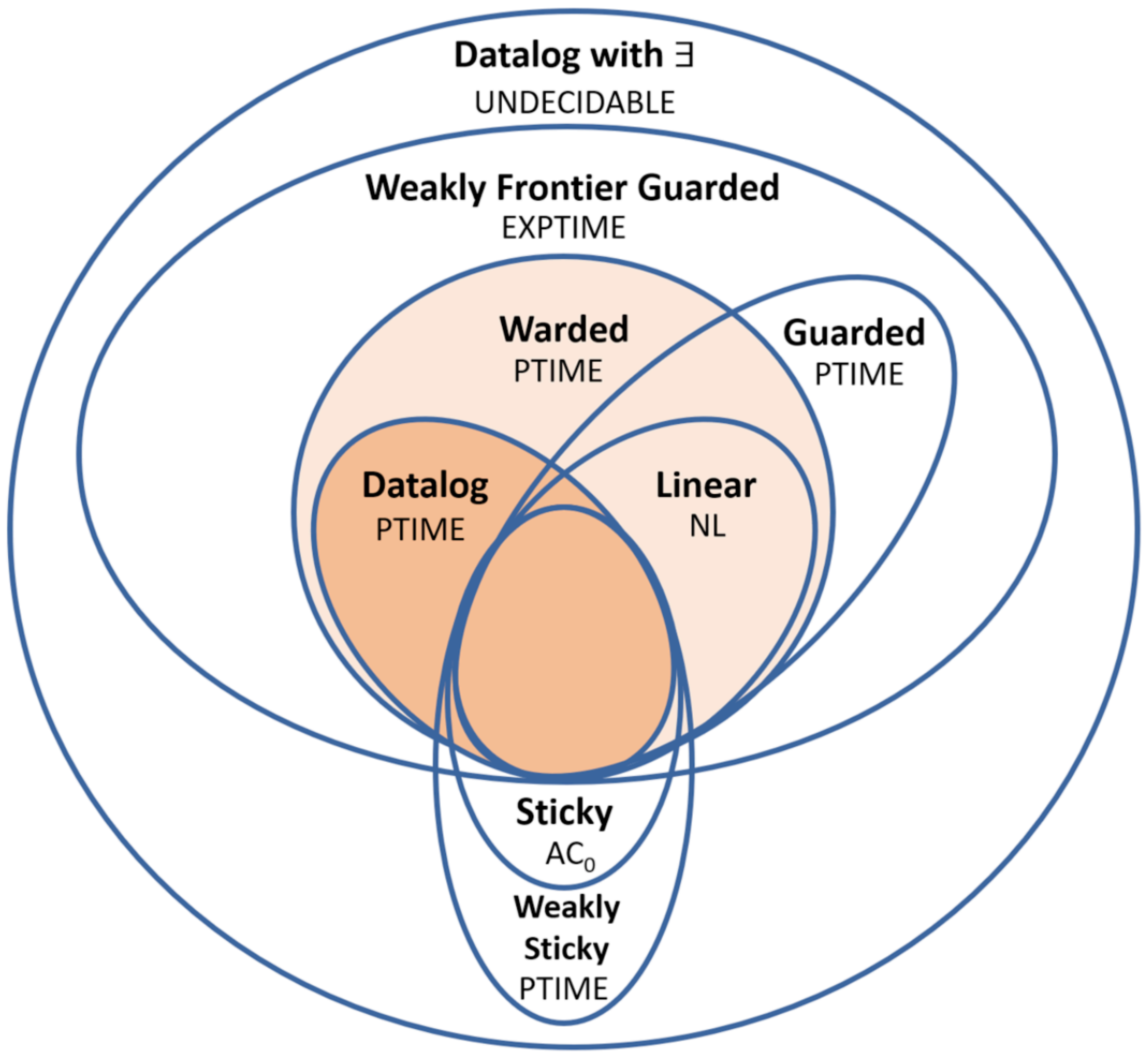


Figure 1: Syntactic containment of Datalog$^{\pm}$ languages. Annotations (non-bold) denote data complexity. All names that do not explicitly mention Datalog refer to the respective Datalog$^{\pm}$ languages. E.g., “Sticky” refers to “Sticky Datalog$^{\pm}$”.

Bellomarini, L., Sallinger, E., & Gottlob, G. (2018).
The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. PVLDB

Expressivity

- Given two query languages $\mathscr{L}_1$ and $\mathscr{L}_2$. For every query q in $\mathscr{L}_1$, can we write a query in $\mathscr{L}_2$ that always produces the same outputs as q?
- What kinds of concepts can a query language not ask about?
  For instance, with first-order logic we cannot ask about reachability in a graph.

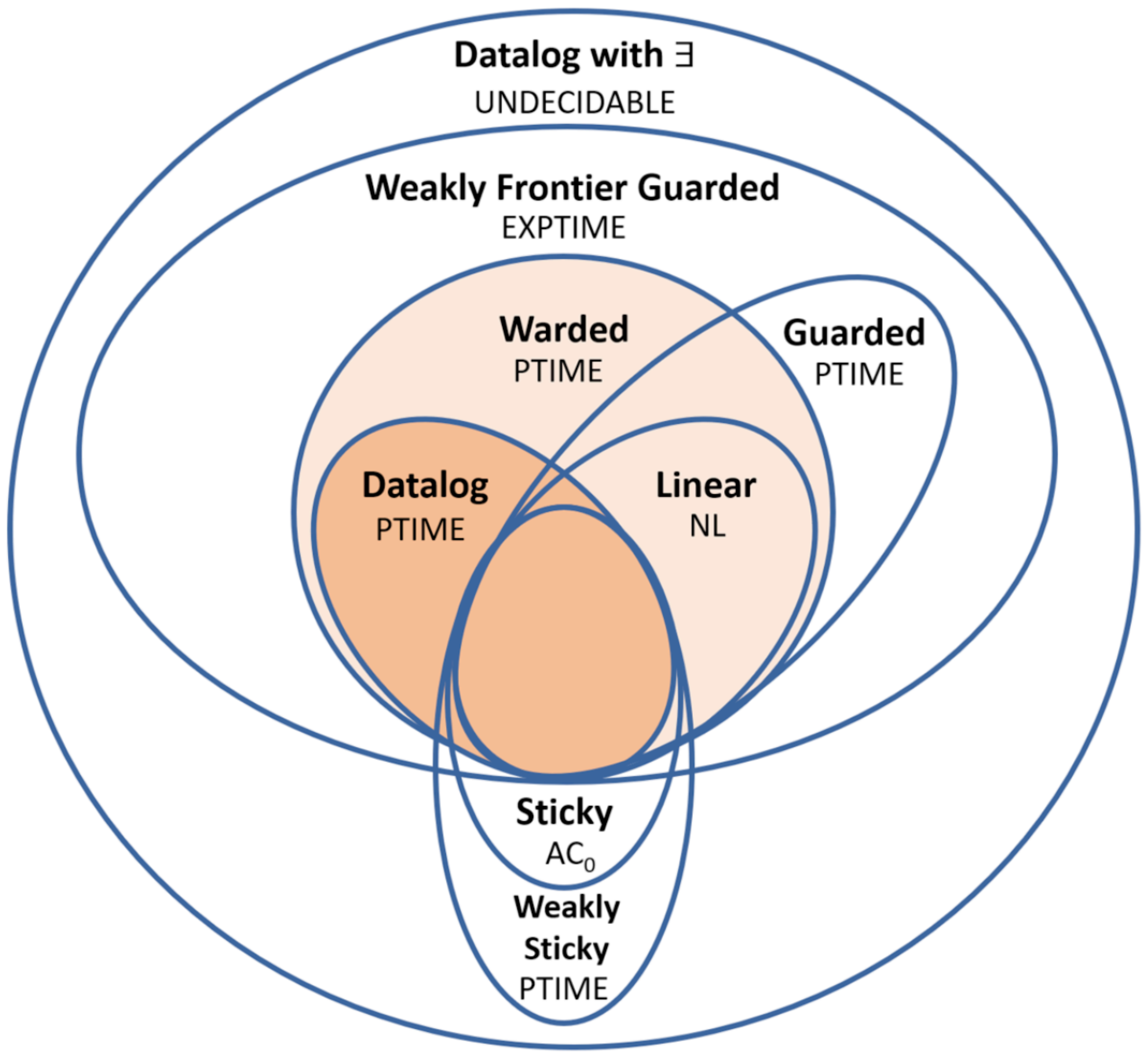


Figure 1: Syntactic containment of Datalog$^{\pm}$ languages. Annotations (non-bold) denote data complexity. All names that do not explicitly mention Datalog refer to the respective Datalog$^{\pm}$ languages. E.g., “Sticky” refers to “Sticky Datalog$^{\pm}$”.

Bellomarini, L., Sallinger, E., & Gottlob, G. (2018).
The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. PVLDB

Equivalence

- Say we have two queries q and q' in the same query language. Are the outputs of the two queries the same on every database?
- If there are many equivalent ways to write the same query, which way is the most efficient with respect to evaluation?

Data Models

Data Model Dimensions

"Shape"

What is the formal description of how the data is stored?

Common examples:

- Relational
- Graphs
- Semi-structured
- Documents

Availability

How can data be accessed?

In distributed settings, access to data that is not local induces additional cost.

Streaming data only gives a small window into the whole data at any given time.

Further Complexities

Various additional notions have been added on top of the other two.

Data can be stored in some way that inherently preserves privacy.

Data can be probabilistic.

Further Topics

Data Quality

Cleaning up data in a formally justified way.

Discovering new data / expanding on the data currently in the database

Incompleteness & Uncertainty

Parts of data is missing (e.g. NULLs)

Data might not be 100% certain, e.g., data from ML models or crowdsourcing.

Data Access

Managing data under concurrency / transactions.

Indexing structures.

Connections to Other Areas

Complexity Theory

Complexity of Query Evaluation

Logic Programming

More advanced query languages fall into logic programming.

Constraint Satisfaction

Constraint Satisfaction Problems are the same thing as Conjunctive Queries.

Logic / Model Theory

Expressiveness of languages.

Graph Theory

Structure of queries and data matters. Homomorphisms.

Automata

XML, Document Spanners, Stream processing, Graph Queries

This Lecture

The lecture will focus on the relational model.

Very tight connections to logic.

Most commonly studied and relevant for most query languages.

Common in practice (SQL, Logic, Graphs, Datalog, ...).

Time for a break.

Turn to your neighbour: briefly discuss the lecture
Stretch, grab water, reset

Database Theory

General Information

Matthias Lanzinger, 2025

Informatics

Classes

Language:
All parts of the class are held in English

Time:
Every week of term: Tuesday, 11:15 - 13:00
(see TISS in case of uncertainty)

Place:
All classes will be in person in room FAV EG C (Seminarraum Gödel).

Prerequisites

Level

This course is designed for students in master or doctoral programs.

Familiarity with mathematical notation and proof is assumed.

Courses

It is recommended to take Formal Methods in CS and a database course before this course.

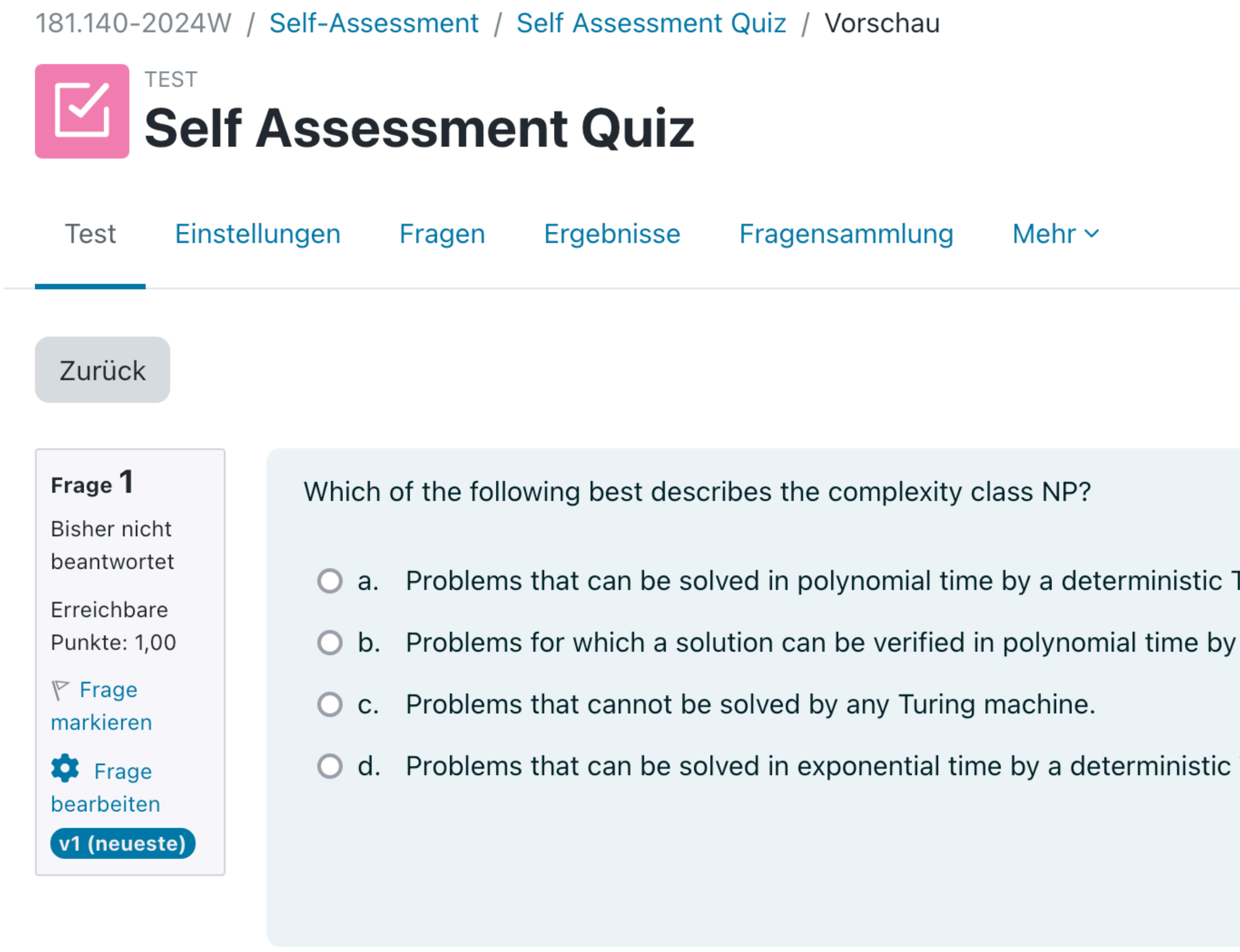
Complexity Theory can be helpful in parallel to this course.

Helpful Knowledge

- Database basics
- Formal logic
- Intro complexity theory
  In particular, reductions!

Self Assessment

- Quiz on TUWEL course for you to judge your own knowledge on course prerequisites.
- Not graded, entirely optional.
- If you are still unsure, talk to me.



181.140-2024W / Self-Assessment / Self Assessment Quiz / Vorschau

TEST

Self Assessment Quiz

Frage 1

Bisher nicht beantwortet

Erreichbare Punkte: 1,00

Which of the following best describes the complexity class NP?

- a. Problems that can be solved in polynomial time by a deterministic T
- b. Problems for which a solution can be verified in polynomial time by
- c. Problems that cannot be solved by any Turing machine.
- d. Problems that can be solved in exponential time by a deterministic

Communication

- During & after classes 🥇
- TUWEL
  Exercises / Extra material
- TISS (room and time information)
- Slides are on the course homepage
  https://dbai.tuwien.ac.at/staff/mlanzing/dbt/

Assessment

Exercise Portfolio

Towards the end of the term you should submit a portfolio of exercises.

You can pick yourself which exercises you want to do. Grading depends on how much you do.

This is sufficient to pass up to grade 2

Other Factors

Presence/participation in lectures is factored into grading of portfolios.

Optional: you can take an oral exam after the term to improve your grade.

There are three types of portfolio exercises

Software projects

Implement concepts, algorithms, and practical applications discussed in the lecture.

Medium sized projects that do not require a lot of code or complex software engineering. Focus on transferring the conceptual understanding to practice.

Example — Software Project

Important algorithms in the lecture

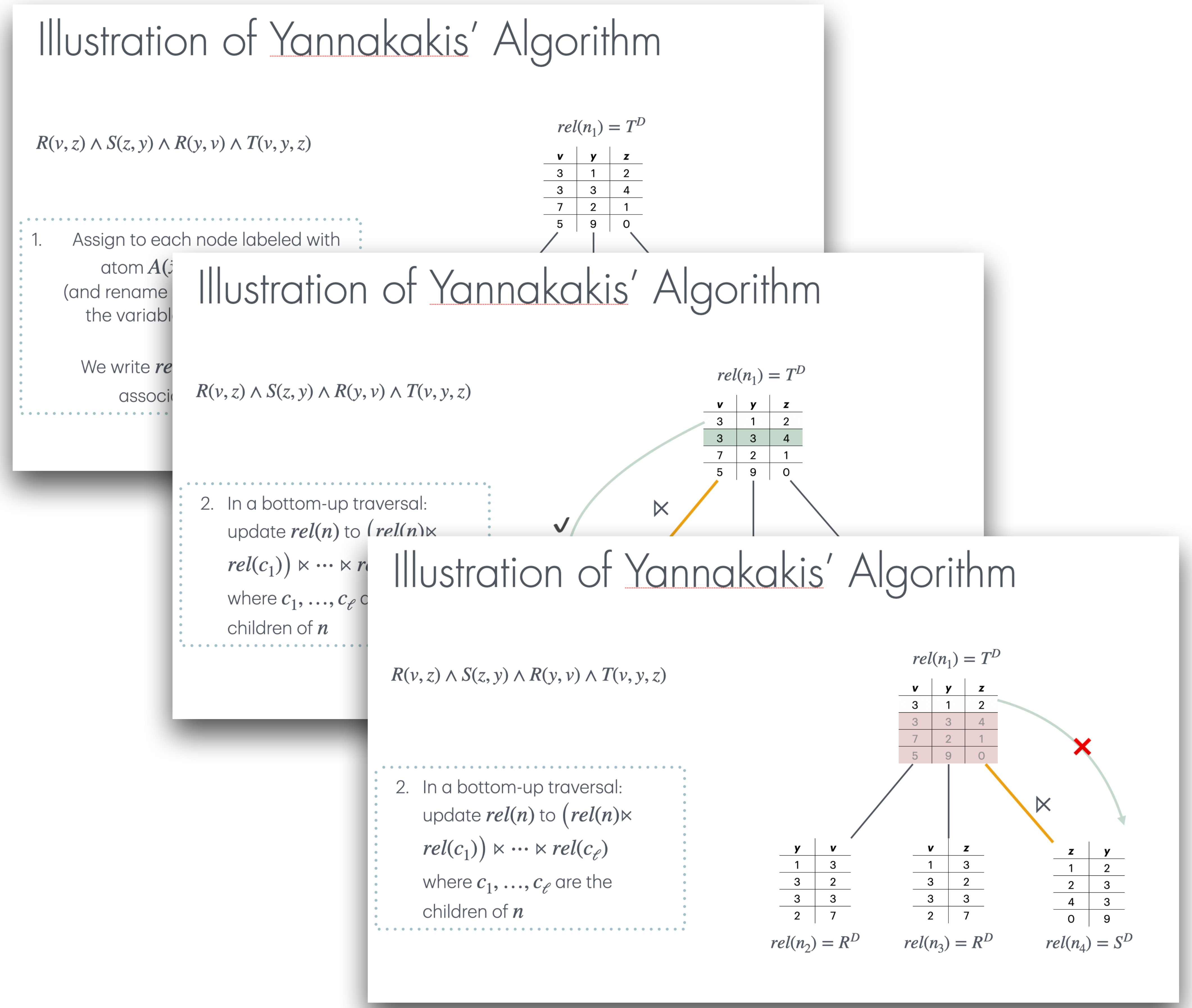


Illustration of Yannakakis' Algorithm
$R(v,z) \wedge S(z,y) \wedge R(y,v) \wedge T(v,y,z)$
$rel(n_1) = T^D$
1. Assign to each node labeled with atom $A($
(and rename
the variabl
We write re
associ
2. In a bottom-up traversal: update $rel(n)$ to $(rel(n) \ltimes$ $rel(c_1)) \ltimes \cdots \ltimes r$
where $c_1, \ldots, c_\ell$ c
children of n

Illustration of Yannakakis' Algorithm

$R(v,z) \wedge S(z,y) \wedge R(y,v) \wedge T(v,y,z)$

2. In a bottom-up traversal: update $rel(n)$ to $(rel(n) \ltimes rel(c_1)) \ltimes \cdots \ltimes rel(c_\ell)$ where $c_1, \ldots, c_\ell$ are the children of n

$rel(n_1) = T^D$

v	y	z
3	1	2
3	3	4
7	2	1
5	9	0

$rel(n_2) = R^D$

y	v
1	3
3	2
3	3
2	7

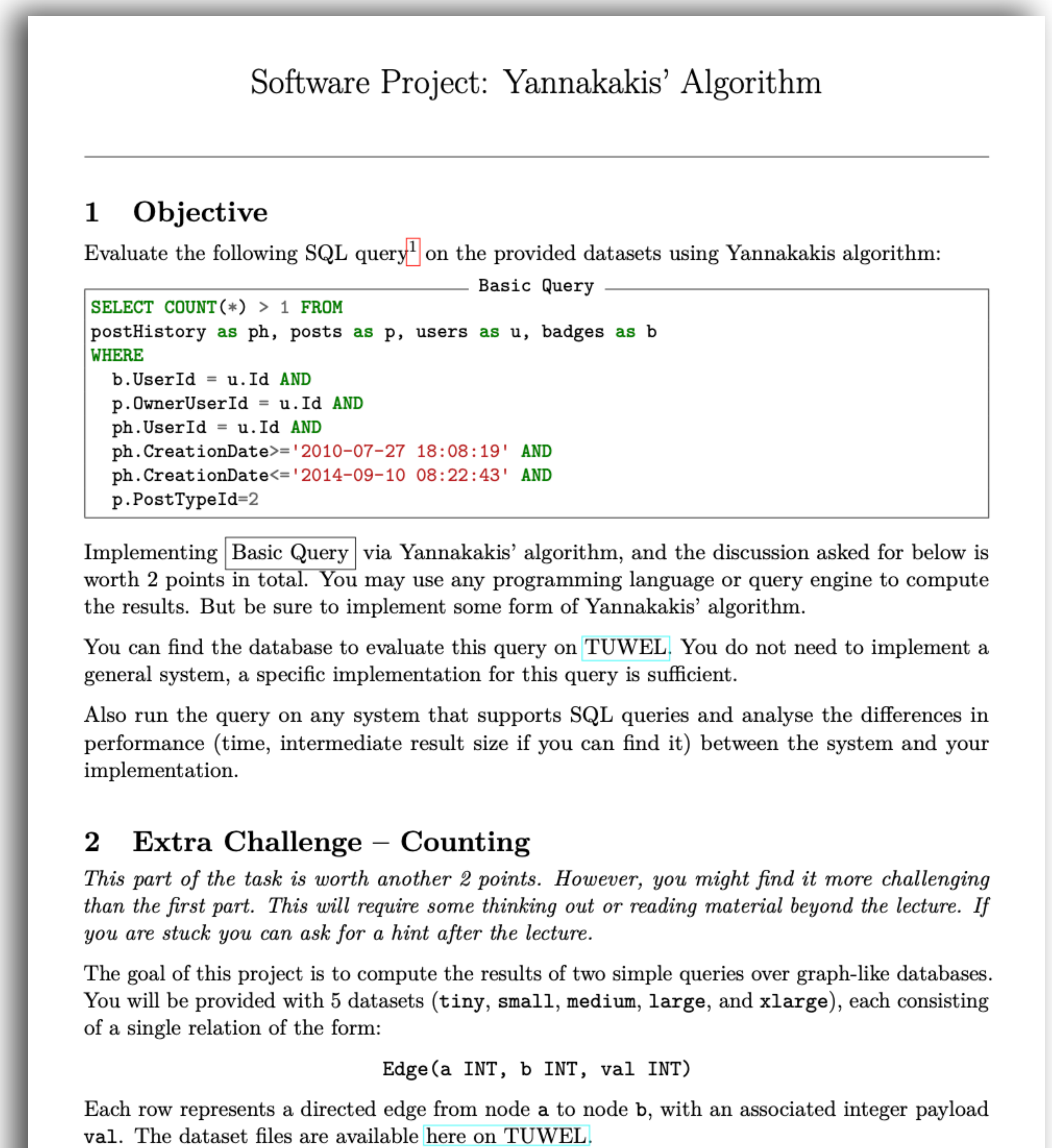
$rel(n_3) = R^D$

v	z
1	3
3	2
3	3
2	7

$rel(n_4) = S^D$

z	y
1	2
2	3
4	3
0	9

Software project to implement some small version of it.



Software Project: Yannakakis' Algorithm

1 Objective

Evaluate the following SQL query[1] on the provided datasets using Yannakakis algorithm:

Basic Query
```
SELECT COUNT(*) > 1 FROM
postHistory as ph, posts as p, users as u, badges as b
WHERE
  b.UserId = u.Id AND
  p.OwnerUserId = u.Id AND
  ph.UserId = u.Id AND
  ph.CreationDate>='2010-07-27 18:08:19' AND
  ph.CreationDate<='2014-09-10 08:22:43' AND
  p.PostTypeId=2
```

Implementing Basic Query via Yannakakis' algorithm, and the discussion asked for below is worth 2 points in total. You may use any programming language or query engine to compute the results. But be sure to implement some form of Yannakakis' algorithm.

You can find the database to evaluate this query on TUWEL. You do not need to implement a general system, a specific implementation for this query is sufficient.

Also run the query on any system that supports SQL queries and analyse the differences in performance (time, intermediate result size if you can find it) between the system and your implementation.

2 Extra Challenge – Counting

This part of the task is worth another 2 points. However, you might find it more challenging than the first part. This will require some thinking out or reading material beyond the lecture. If you are stuck you can ask for a hint after the lecture.

The goal of this project is to compute the results of two simple queries over graph-like databases. You will be provided with 5 datasets (`tiny`, `small`, `medium`, `large`, and `xlarge`), each consisting of a single relation of the form:

```
Edge(a INT, b INT, val INT)
```

Each row represents a directed edge from node `a` to node `b`, with an associated integer payload `val`. The dataset files are available here on TUWEL.

There are three types of portfolio exercises

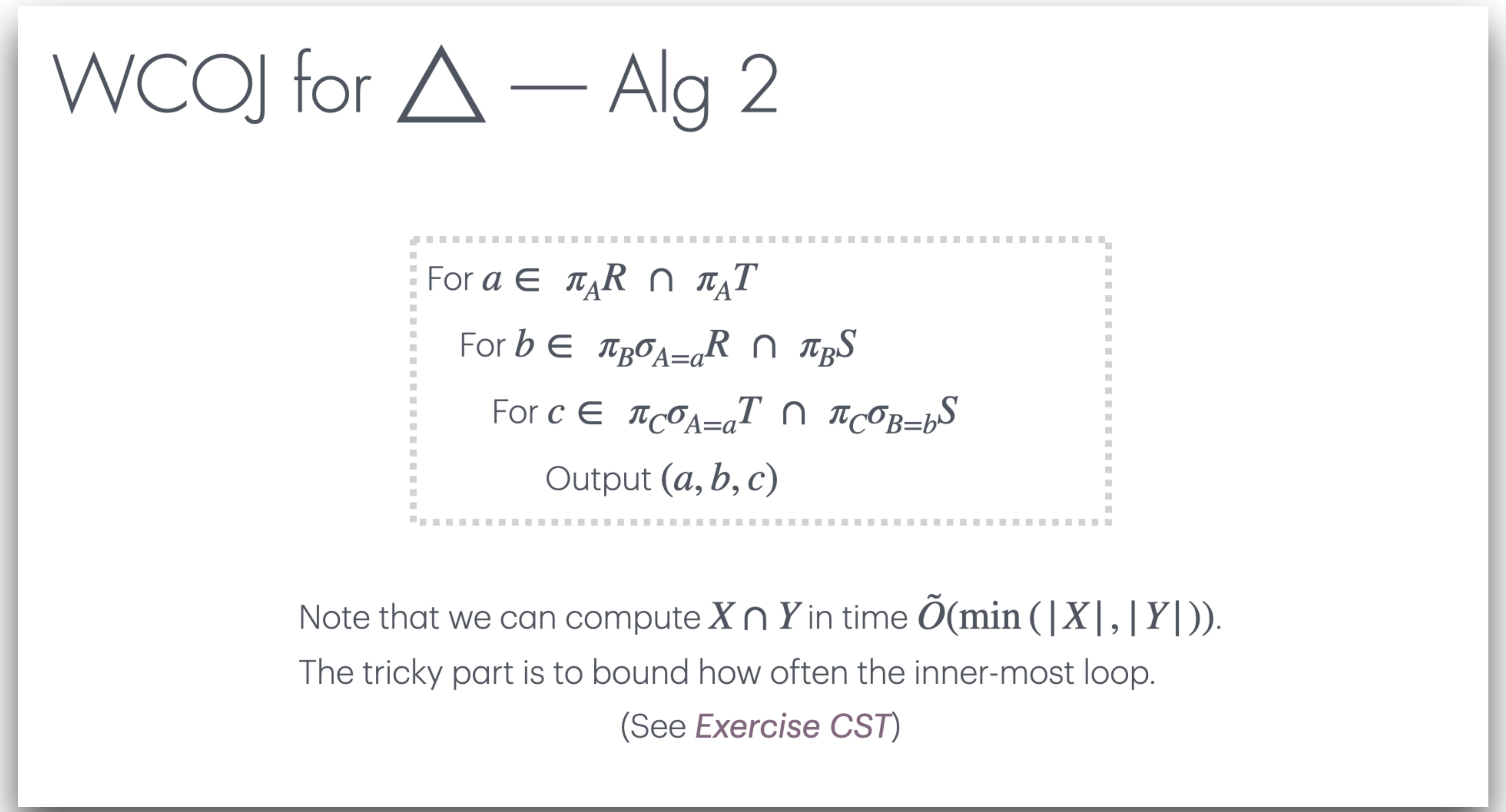
Theory Tasks

Explore theoretical concepts that were only briefly covered during the lecture due to time constraints.

Reporting more light-weight. Answer some simple questions to demonstrate that you engaged with the details of the topic at hand.

Example — Theory Tasks

Interesting details that are too much for the lecture.

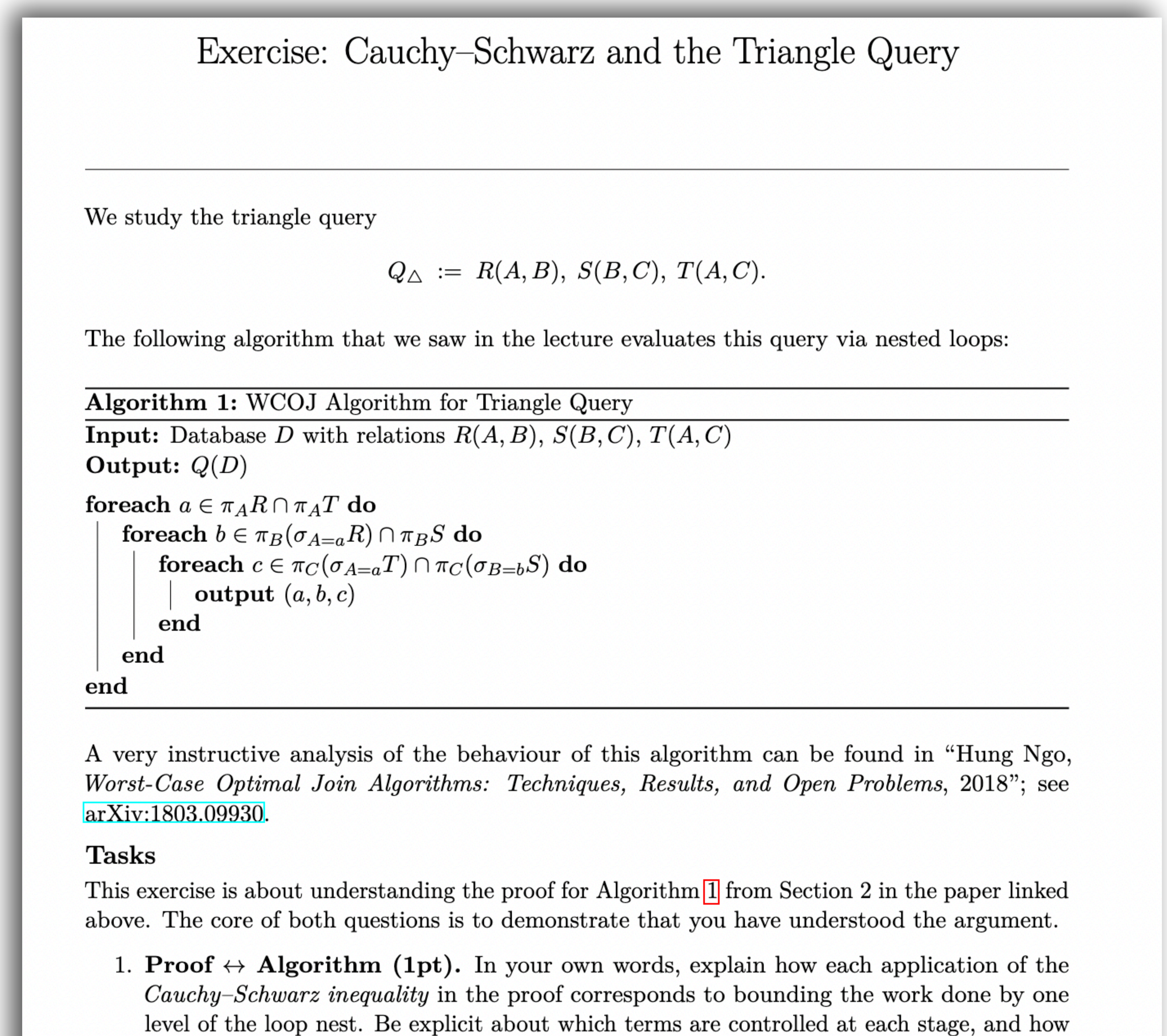


WCOJ for △ — Alg 2

```
For $a \in \pi_A R \cap \pi_A T$
  For $b \in \pi_B \sigma_{A=a} R \cap \pi_B S$
    For $c \in \pi_C \sigma_{A=a} T \cap \pi_C \sigma_{B=b} S$
      Output $(a, b, c)$
```

Note that we can compute $X \cap Y$ in time $\tilde{O}(\min(|X|, |Y|))$.
The tricky part is to bound how often the inner-most loop.
(See Exercise CST)

Theory task for diving into the details.



Exercise: Cauchy–Schwarz and the Triangle Query

We study the triangle query

$$Q_\triangle := R(A,B),\ S(B,C),\ T(A,C).$$

The following algorithm that we saw in the lecture evaluates this query via nested loops:

Algorithm 1: WCOJ Algorithm for Triangle Query
Input: Database D with relations $R(A,B)$, $S(B,C)$, $T(A,C)$
Output: $Q(D)$

```
foreach $a \in \pi_A R \cap \pi_A T$ do
    foreach $b \in \pi_B(\sigma_{A=a} R) \cap \pi_B S$ do
        foreach $c \in \pi_C(\sigma_{A=a} T) \cap \pi_C(\sigma_{B=b} S)$ do
            output $(a, b, c)$
        end
    end
end
```

A very instructive analysis of the behaviour of this algorithm can be found in "Hung Ngo, *Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems*, 2018"; see arXiv:1803.09930.

Tasks

This exercise is about understanding the proof for Algorithm 1 from Section 2 in the paper linked above. The core of both questions is to demonstrate that you have understood the argument.

1. **Proof ↔ Algorithm (1pt).** In your own words, explain how each application of the *Cauchy–Schwarz inequality* in the proof corresponds to bounding the work done by one level of the loop nest. Be explicit about which terms are controlled at each stage, and how

There are three types of portfolio exercises

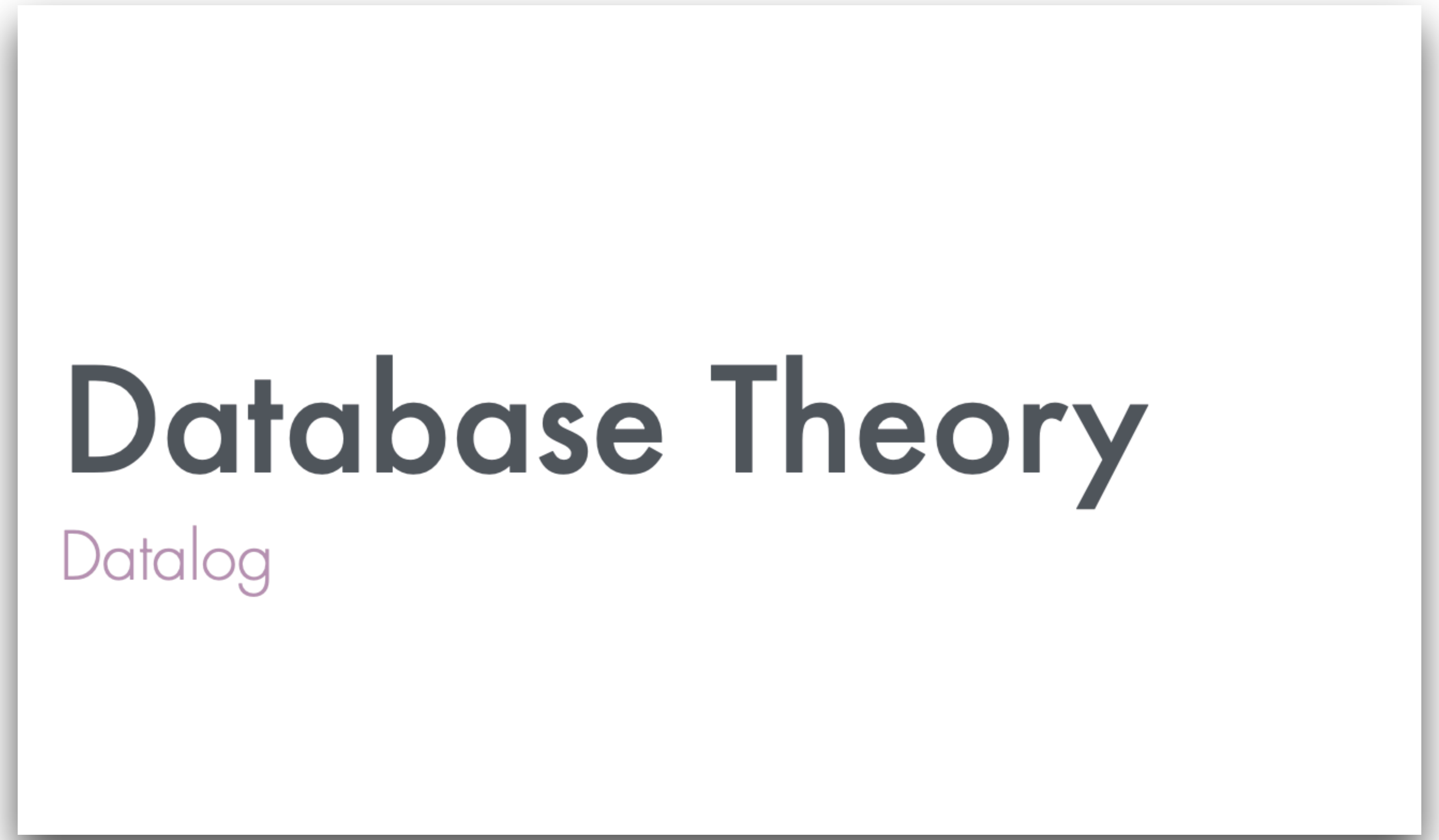
Literature Research

Expand on topics beyond the lecture by diving into the academic literature.

One or two starting points for your exploration is provided. You should get a high-level overview of the topic and what people are working on.

Example — Literature Research

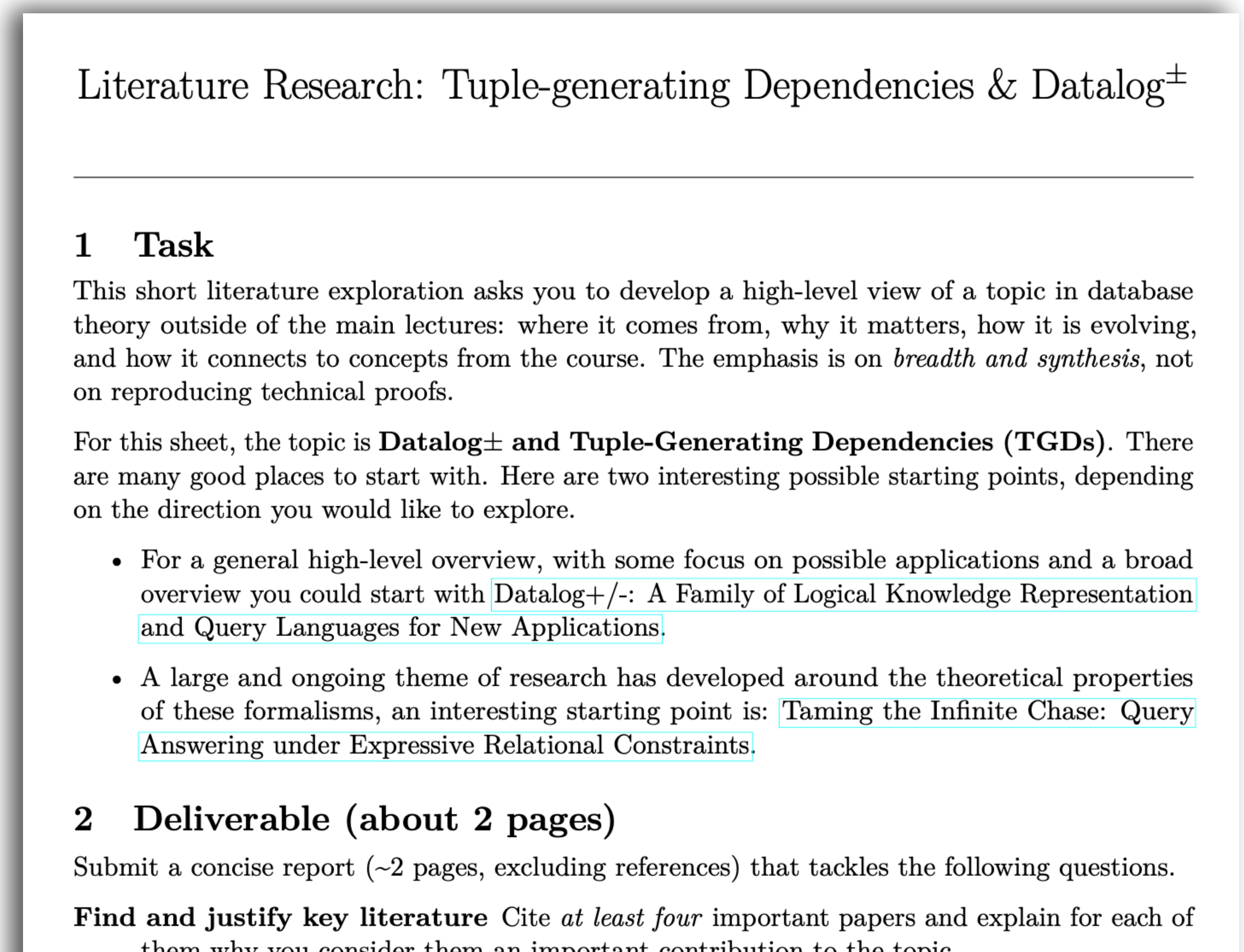
Basics of a topic in the lecture



Database Theory

Datalog

Explore modern or advanced developments in your own literature research.



Literature Research: Tuple-generating Dependencies & Datalog$^{\pm}$

1 Task

This short literature exploration asks you to develop a high-level view of a topic in database theory outside of the main lectures: where it comes from, why it matters, how it is evolving, and how it connects to concepts from the course. The emphasis is on *breadth and synthesis*, not on reproducing technical proofs.

For this sheet, the topic is **Datalog± and Tuple-Generating Dependencies (TGDs)**. There are many good places to start with. Here are two interesting possible starting points, depending on the direction you would like to explore.

- For a general high-level overview, with some focus on possible applications and a broad overview you could start with Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications.
- A large and ongoing theme of research has developed around the theoretical properties of these formalisms, an interesting starting point is: Taming the Infinite Chase: Query Answering under Expressive Relational Constraints.

2 Deliverable (about 2 pages)

Submit a concise report (~2 pages, excluding references) that tackles the following questions.

Find and justify key literature Cite *at least four* important papers and explain for each of them why you consider them an important contribution to the topic.

Portfolio Info

Exercises are available from the beginning of the term.
(since this system is new I might add some during the term)

Individual work — no teams
Answers that are based on work by your colleagues/LLMs will be graded 0. That does not mean you should not discuss with each other.

Portfolio Grading

Each submitted task gives points. Usually there are 1 - 3 points per problem sheet, depending on how much of the problem you choose to work on.

Your overall grade for the portfolio is determined as follows:

[0, 13)	[14, 17)	[17, 20)	20+
Fail	Grade 4	Grade 3	Grade 2

Portfolio Grading

It is possible that you do not get awarded points for tasks that you have submitted!

Most likely reasons:

LLM generated answers

Very wrong answers (if seen as not a serious attempt)

Plagiarism

Portfolio Submission

At the end of term, submit a zip file containing the following:

- A single pdf report with the answers for of each submitted tasks.
  The problem sheets will specify what is expected.
  A TeX template is available on TUWEL.
- A folder for each software project containing the code
  (in an executable/reproducible way)

Portfolio Submission

On top of the final submission, there will be two mandatory intermediate submissions. You need to have at least something by then.

This way you can get feedback earlier and you get an idea of where you stand in terms of points. Avoids leaving everything to last week.

Can contain work-in-progress notes for feedback and notes.

Intermediate submissions on November 1st and December 14th.

Literature

The following two books are recommended if you would like to supplement the contents of this lecture:

Abiteboul, Vianu, Hull — Foundations of Databases

Arenas, Barceló, Libkin, Martens, Pieris — Database Theory
(Still in progress but fully usable already for the parts concerning this lecture)
https://github.com/pdm-book/community

More papers related to lecture topics and exercises are available in TUWEL.