Efficient Algorithms for Frequently Asked Questions

1. Introduction & Course Organisation

Prof. Dan Olteanu

DaST Data•(Systems+Theory)

University of Zurich[UZH]

February 21, 2022

https://lms.uzh.ch/url/RepositoryEntry/17185308706

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

Yet, they all admit **one** natural and elegant formalisation and algorithmic solution

Why Take This Course?

Many computational problems in Computer Science are deeply related

- They are widely taught and researched **separately**
- They are formalised **differently** and come with **specific** algorithmic solutions

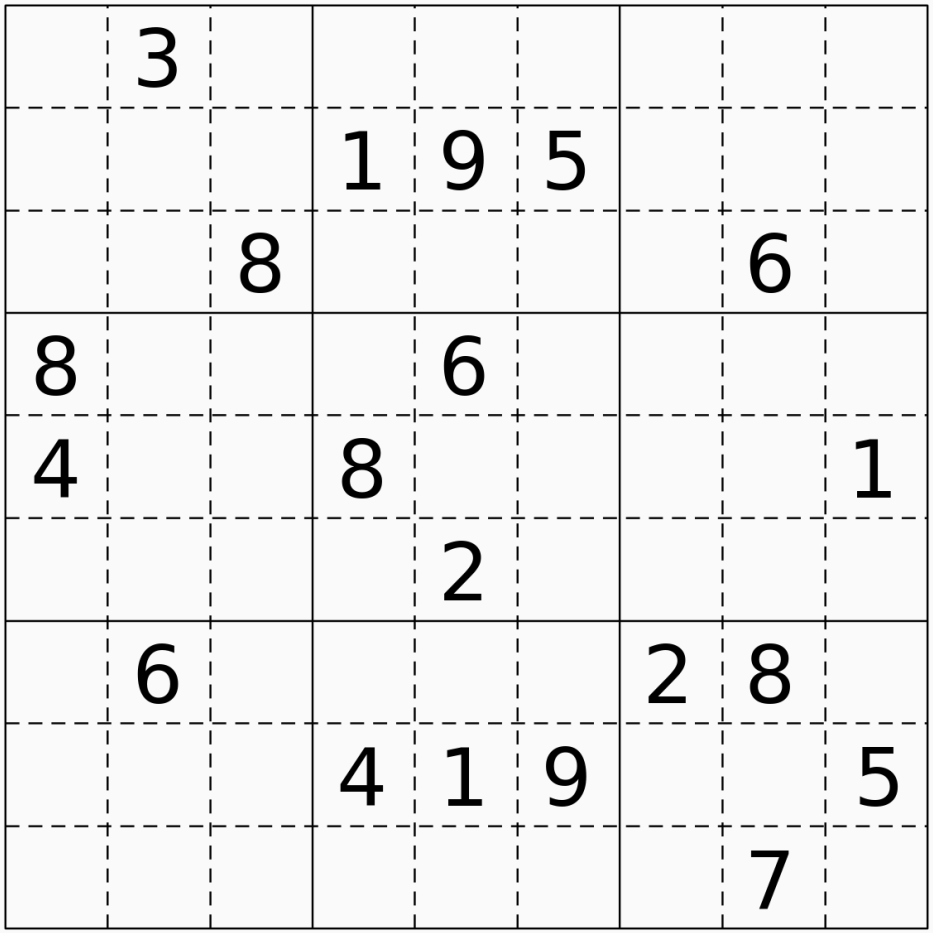
Yet, they all admit **one** natural and elegant formalisation and algorithmic solution

This course is:

- a journey from such problems to their unified formalisation and solution
- the first of its kind – to the best of our knowledge
- closely follows research done by many (ourselves included) across CS

A Very Small Sample of the Computational Problems under Consideration

Constraint Satisfaction Problems (CSPs)

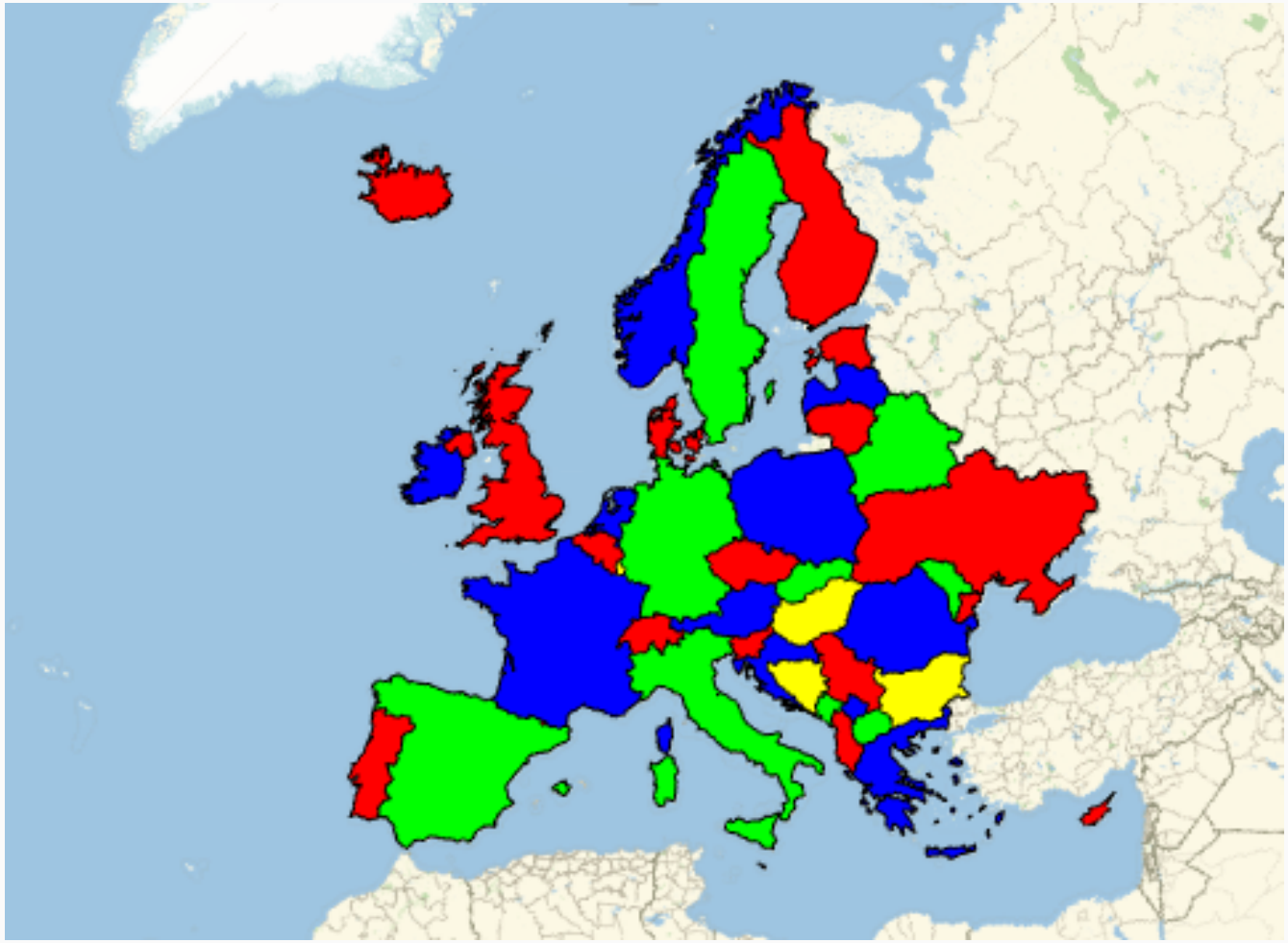


	3							
			1	9	5			
		8					6	
8				6				
4			8					1
				2				
	6					2	8	
			4	1	9			5
							7	

Sudoku: Find a completion of the grid with numbers 1..9 such that each number occurs once in a row, column, and 3x3 block

A Very Small Sample of the Computational Problems under Consideration

Satisfiability (SAT, #SAT)



Map colouring: Can we colour the map of Europe using 4 colours?

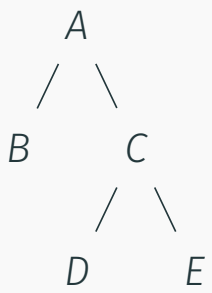
How many satisfying assignments of colours to European countries are there?

A Very Small Sample of the Computational Problems under Consideration

Query Evaluation in Relational Databases (QE)

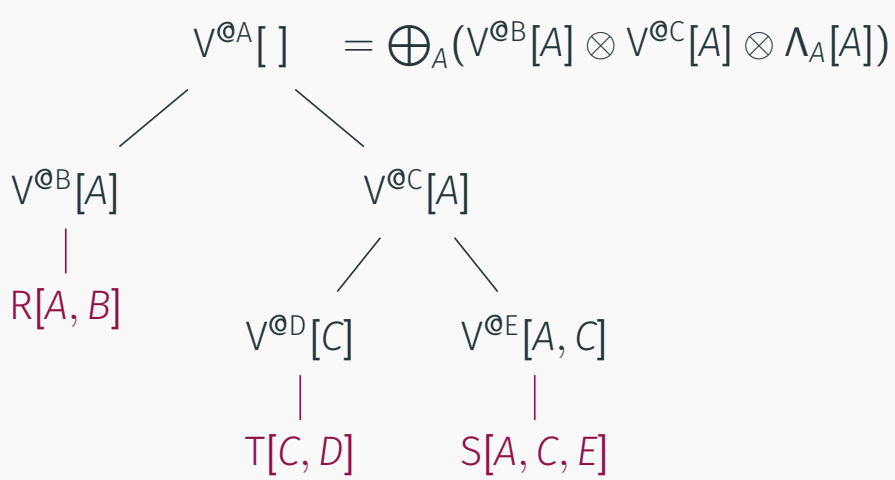
Variable order

$dep(A) = \emptyset$
$dep(B) = \{A\}$
$dep(C) = \{A\}$
$dep(D) = \{C\}$
$dep(E) = \{A, C\}$



$\Rightarrow$

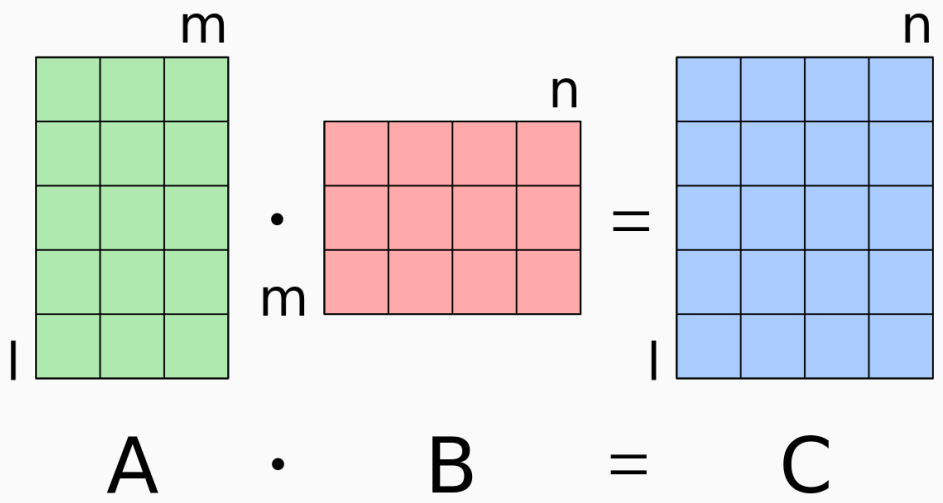
View tree



Given a database with relations R, S, and T, find the result of their natural join

A Very Small Sample of the Computational Problems under Consideration

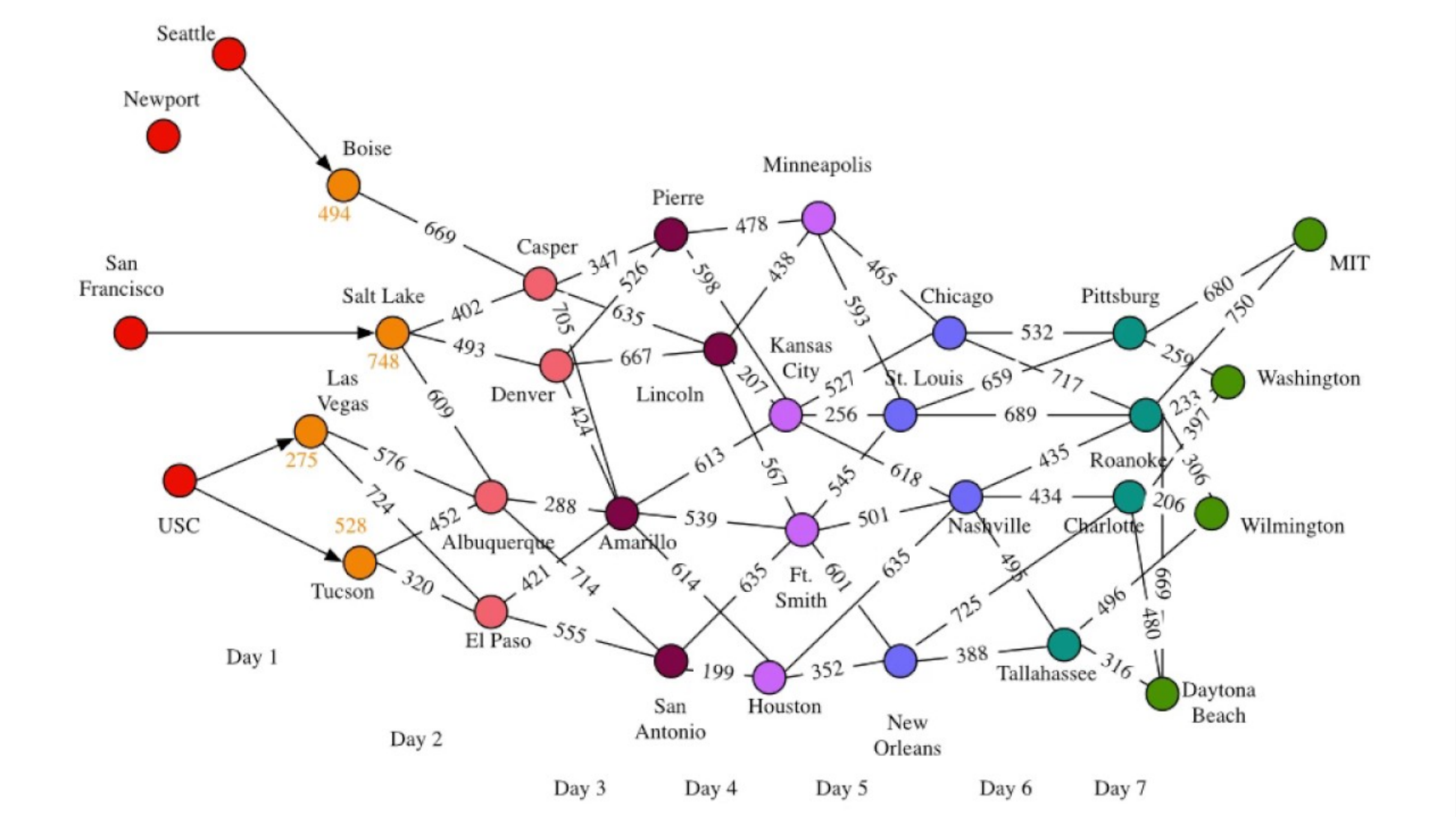
Matrix Chain Multiplication (MCM)



Given a chain of matrices with matching dimensions, compute the matrix representing their multiplication

A Very Small Sample of the Computational Problems under Consideration

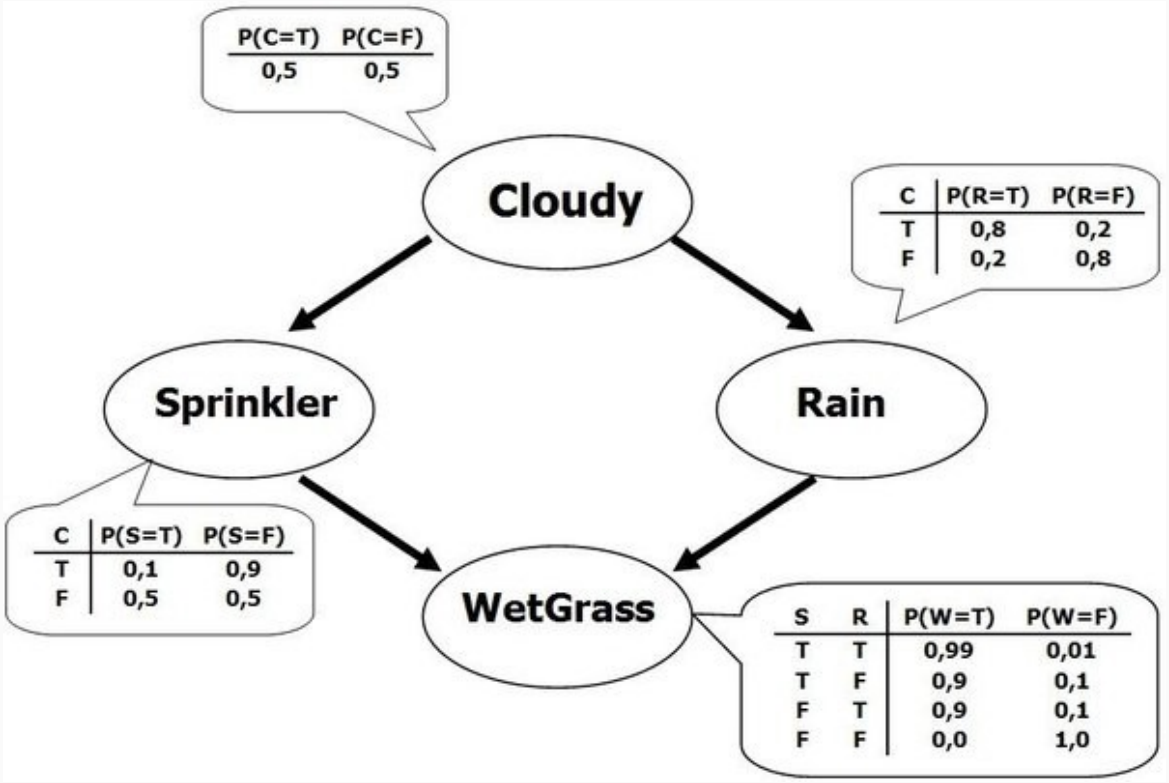
Viterbi Algorithm (VA)



Find the most likely sequence of hidden states that results in a sequence of observed events, e.g., in hidden Markov models.

A Very Small Sample of the Computational Problems under Consideration

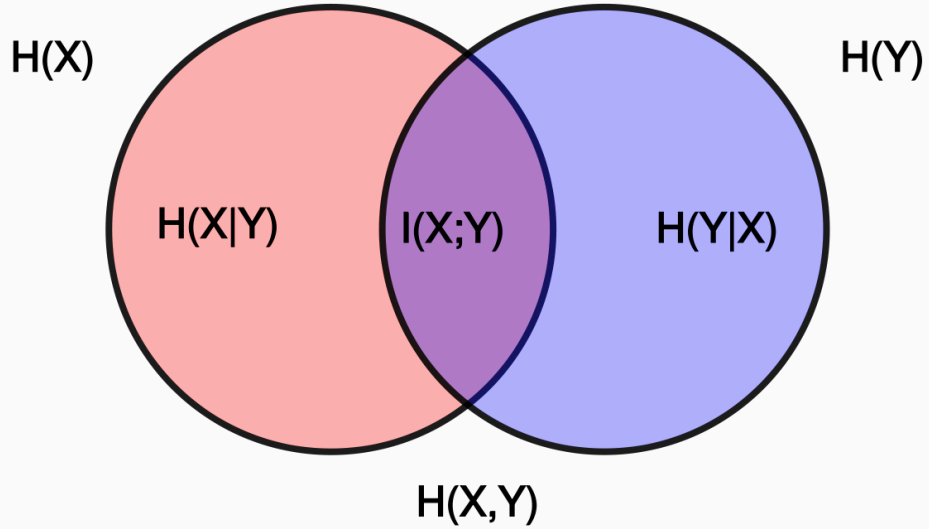
Inference in Bayesian Networks and Probabilistic Graphical Models



What is the probability that the grass is wet given that the sprinkler was off and it did not rain?

A Very Small Sample of the Computational Problems under Consideration

Cost Functions and Statistics for Machine Learning (ML)



$$\mathrm{K_{XX}} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_n - \mathrm{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_n - \mathrm{E}[X_n])] \end{bmatrix}$$

Compute the empirical entropy, the pairwise mutual information and covariance of the variables describing the data.

Did You Get the Gist?

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

2. Algorithm to efficiently compute any expression in the language

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

   - Our language in this course: Functional Aggregate Queries [Ngo et al 2016]
   - Several previous incarnations in AI & Information Theory, e.g.,

     Marginalise a Product Function [Aji & McEliece 2000]

2. Algorithm to efficiently compute any expression in the language

How to Uniformly Approach These Problems?

1. Unified language able to express a wide range of such problems

- Our language in this course: Functional Aggregate Queries [Ngo et al 2016]
- Several previous incarnations in AI & Information Theory, e.g.,

  Marginalise a Product Function [Aji & McEliece 2000]

2. Algorithm to efficiently compute any expression in the language

- Decompose the problem into smaller tasks based on its structure
  [Gottlob et al 1990, Olteanu and Zavodny 2015, Abo Khamis et al 2016]
- Arbitrary structure is reduced to tree structure worst-case optimally
  [Ngo et al 2012, Veldhuizen 2014]
- Best known algorithms used for tree structures [Yannakakis 1982]

Learning Outcomes

On completion of the course you should be able to

- Learn to formalise problems using various formalisms
- Analyse computational complexity of problems using a toolbox of techniques that exploit the algebraic and combinatorial structure of the problems
- Familiarise themselves with simple yet powerful algorithms
- Implement and benchmark such algorithms

Course Information

Website: https://lms.uzh.ch/url/RepositoryEntry/17185308706

Sessions:

- Mondays 16.15 - 18:00, with 15 min break
- Two Mondays are off: April 18; April 25
- 13 sessions in total: 10 lectures, 2 exercises, 1 practical
- Room: BIN-2.A.01

Team:

- Prof. Dan Olteanu
- PhD student Nadia Knorozova
- Last-minute help if needed: Dr. Ahmet Kara and PhD student Haozhe Zhang

Course Materials

Materials posted on OLAT:

1. Research monographs/surveys/papers on relevant topics
    - Cover a wealth of ideas across Computer Science
    - Cover theory, algorithmic development, and systems aspects
2. Slides made available before the lectures

Lecture slides from 2021:

`https://www.ifi.uzh.ch/en/dast/teaching/EA.html`

Lecture Topic 1: Commutative Semirings

- Definition, Benefits: Generalisation and Reduced Complexity
- Examples: Boolean, Sum-Product, Max-Product, Polynomial, Inner-Product
- Sample of Problems (and their Semirings)
  - Algebraic Path Problems via Semiring Fixpoint Equation:
    - Shortest Distance (Min-Sum)
    - Connectivity and Largest Capacity (Max-Min)
    - Maximum Reliability (Max-Product)
    - Language Accepted by Automaton (Union-Concatenate)
  - Satisfiability:
    - Map Colouring (Boolean)
  - Database Query Evaluation:
    - Factorised Joins (Union-Product)
    - Factorised Aggregate-Joins (Sum-Product)
  - Medical Diagnosis with Bayesian Networks:
    - Marginal Distribution (Sum-Product)
    - MAP (Max-Product)

Lecture Topic 2: Functional Aggregate Queries (FAQs)

- Hypergraphs, Definition and Examples
- Expressing Computational Problems in FAQ
- Examples:
  - Algebraic Path Problems
  - SATisfiability: (Count)Satisfiability, k-Colorability
  - Databases: Conjunctive Queries, Natural Joins
  - Linear Algebra: Matrix Chain Multiplication, Einsum Notation
  - Probabilistic Graphical Models: Marginal Distribution and MAP
  - Machine Learning: Cost Functions and Gradients
  - . . .
- Map of Expressing Problems in FAQ

Lecture Topic 3: Structural Decompositions of FAQs

FAQ compute time depends on the structure of its hypergraph

- (Hyper)tree decompositions
- Types of acyclicity, the GYO algorithm, free-connex property
- Join trees: Constructing join trees for acyclic hypergraphs
- Examples: triangles, cliques, grid

Lecture Topic 4: Width Measures for FAQs

Challenge of choosing a good hypertree decomposition

- (Fractional) Edge covers and independent sets in hypergraphs
- Integer programs and their linear program relaxations
- Hypertree width and fractional hypertree width

Lecture Topic 5: Solving Joins Optimally

- Instance output size versus worst-case output size
- Yannakakis algorithm for acyclic queries
- Worst-case optimal join algorithms: LeapFrog TrieJoin Join
- Suboptimality of state-of-the-art join algorithms for cyclic queries
- Efficient processing for queries with large output
    - Trade-off between preprocessing time and enumeration delay
    - Query classes: conjunctive, alpha acyclic, free-connex alpha acyclic

Lecture Topic 6: Worst-Case Optimal Size Bounds for Joins

- Upper Bound
  - Warm-up: Triangle join
  - Information theory: Random variables, (joint, conditional) entropy, Shannon inequalities, chain rule, Shearer's lemma
  - Connection of Shearer's lemma with fractional edge cover number
- Lower Bound
  - Warm-up: Triangle join
  - Dual linear program for fractional edge cover number
  - Construction of factors to meet the desired lower bound

Lecture Topic 7: Solving SAT

- SAT encoded as FAQ
- Clause representation
- The DP procedure: single-phase variables, tautological clauses, unit-clause contradictions, resolution, equi-satisfiability
- The DPLL procedure: single-phase variables, unit propagation, literal marginalisation
- DP and DPLL seen through glasses of FAQ solver
- Hardness and Tractability of Acyclic SAT

Lecture Topic 8: Solving Functional Aggregate Queries (FAQs)

- FAQ over one semiring
  - Rewriting FAQs into acyclic FAQs
  - Variable marginalisation orders
  - Indicator projections
  - Moving marginalisation past products
- FAQ over multiple semirings
  - Product aggregates, powering factors
  - Marginalisation orders for variables over different semirings
- The InsideOut algorithm
- The FAQ width

Lectures: Format

Lecturer: **Prof. Dan Olteanu**

- On-site lectures, you presence is expected
- Q&A: Live during lecture, asynchronous in OLAT forum
- Please post questions about the course on OLAT
  - Contact the course team by email for non-technical matters
  - Check whether your technical question has been already answered on OLAT

Classes

Tutor: **Nadia Knorozova**

Class Topics:

1. Express Problems as Functional Aggregate Queries; Decompositions
2. Solving Functional Aggregate Queries

Class Format:

- Exercise sheets made available two weeks before the class
- They are optional but ..
  - achieving at least 4/5 of a homework means 5 bonus points
- Model answers discussed in class, corrections released shortly before class
- Q&A: Live in class + at any time in OLAT forum

Practical Task

Demonstrator: **Nadia Knorozova**

Task:

1. Implement the worst-cast optimal join algorithm Leapfrog Triejoin (LFTJ)
2. Benchmark its performance on public and private tasks and data
3. Extend LFTJ with semiring computation

   Beyond joins: count aggregates, projections, and covariance matrix

Practical Setup

- Each student gets a GitHub repository
  - We set it up and control
  - Send your github username to the demonstrator by March 13
- Public data and expected output of your code available for testing
- Each student pushes their code to their repository
  - Make sure it compiles!
  - Make sure it works correctly on public data!
- Choice of programming language: wide range permitted
  - Check with the demonstrator before starting to write code
  - It needs to compile and run on our Linux test machine

Practical Task: Automated Tests and Leaderboard

We maintain a leaderboard with the runtime performance scores for private data

- Goal: Guide the effort and eventually improve the code
- Once code gets submitted, it is automatically compiled and tested
- Feedback on correctness and runtime performance pushed to the repositories
- If the correctness test is passed, then then runtime performance is added to the leaderboard

2021 Leaderboard:

`https://dastuzh.github.io/EA21-Leaderboard/index.html`

What Makes a Successful Practical Submission

- Pass the correctness tests for all given tasks on both public and private data
- The private data has the same structure as the public data, but different content. Not disclosed but used to test the correctness and runtime performance of submitted code.
- Submission deadline for the practical: EOD (Zurich time) on June 5, 2022
- Each student must work independently on their practical. It is not allowed to use existing code written by others (fellow students or otherwise).
- A successful practical submission before the deadline is a prerequisite for sitting the written exam and therefore passing the course.

Practical Task: Bonus points

- Bonus points awarded to students whose code performs above average in the leaderboard
- Allocation of bonus points to be decided after the submission deadline
- Max practical bonus points = 10

Written Exam and Final Grade

- Date: **Monday 20/06/2022, 16:15 - 18:00**
- Open book, online
- Formalise a problem in FAQ, analyse its computational complexity by decomposing it and propose an efficient strategy to evaluate it.
- Expect distinct questions per student but of similar difficulty
  - you need to know what you are doing
  - copying someone else's solution is futile and time consuming

Final Grade based on the following three components:

- Written Exam: 0-100 marks
- Homework: 0-10 marks
- Practical: 0-10 marks

Is this Course Right for You?

The topics addressed in this course are: mathematically rigorous, making use of computational complexity, algorithm design and analysis, formal proof & reasoning

Ideas across Computer Science areas at the frontier of research

Ideal prerequisites: BSc in Computer Science, in particular:

- **Foundations of Computing I + II, Informatik II**
- **Programming skills needed**

Join Us!

Student projects & theses:

`https://www.ifi.uzh.ch/en/dast/projects.html`

Research positions:

`https://www.ifi.uzh.ch/en/dast/jobs.html`