Constraint Satisfaction

Complexity

and

Logic

Phokion G. Kolaitis

UC Santa Cruz

www.cs.ucsc.edu/~kolaitis/talks/nasslli.ps

Goals and Prerequisites

Goals:

- Show that Constraint Satisfaction Problems (CSP) form a broad class of algorithmic problems that arise naturally in several different areas.
- Present an overview of CSP with emphasis on the computational complexity aspects of CSP and on the connections with database theory and logic

Prerequisites:

- Familiarity with the basics of first-order logic.
- Familiarity with the basics of computational complexity (PTIME and NP-completeness).

Course Outline

Constraint Satisfaction Problems (CSP)

- Definition & Examples of CSP
- CSP & the Homomorphism Problem
- CSP & Database Theory
- Computational Complexity of CSP
  - Uniform & Non-Uniform CSP
  - Schaefer's Dichotomy Theorem for Boolean CSP
  - The Feder-Vardi Dichotomy Conjecture for general CSP
- The Pursuit of Tractable Cases of CSP
  - Datalog & Pebble Games
  - Consistency Properties & Bounded Treewidth

Algorithmic Problems

- k-SAT: ($k \geq 2$ is fixed)

  Given a k-CNF formula, is it satisfiable?

- k-COLORABILITY: ($k \geq 2$ is fixed)

  Given a graph $\mathbf{H} = (V, E)$, is it k-colorable?

- CLIQUE:

  Given a graph $\mathbf{H} = (V, E)$ and a positive integer k, does $\mathbf{H}$ contain a k-clique?

- HAMILTONIAN CYCLE:

  Given a graph $\mathbf{H} = (V, E)$, does it contain a Hamiltonian cycle?

Algorithmic Problems

Latin Squares: L. Euler – 1783

A *Latin square of order* n is an $n \times n$ matrix of n symbols such that each symbol occurs exactly once in each row and each column of the matrix.

A Latin Square of Order 4

2	3	1	4
3	4	2	1
4	1	3	2
1	2	4	3

LATIN SQUARE COMPLETION: given a $n \times n$ grid and a partial assignment of n symbols, can this partial assignment be completed to a Latin square of order n?

Latin Square Completion

- There is a completion to a Latin square

2			4
	4		1
4		3	
1			

- There is no completion to a Latin square

3			4
			1
		2	
	2		

Latin Square Completion

Is there a completion to a Latin square?

3			4	6		
	5			1	3	
		2	5			
	7	3				
				7		2
		4		3		
			2		5	3

Algorithmic Problems

Problems:

- k-SAT ($k \geq 2$)
- k-Colorability ($k \geq 2$)
- Clique
- Hamiltonian Cycle
- Latin Square Completion

Question: What do these have in common?

Constraint Satisfaction Problems

Problems:

- k-SAT ($k \geq 2$)
- k-Colorability ($k \geq 2$)
- Clique
- Hamiltonian Cycle
- Latin Square Completion

Question: What do these have in common?

Answer: Each of them is a

Constraint Satisfaction Problem.

Constraint Satisfaction Problem

U. Montanari – 1974

CSP – Informal Definition

Input: A set V of *variables*, a domain D of *values* for the variables, and a set C of *constraints*.

Question: Is there an *assignment* of values to variables such that all constraints in C are satisfied?

Example: k-SAT as a CSP

k-CNF φ: variables $x_1, \ldots, x_n$; clauses $c_1, \ldots, c_m$

- Variables: $x_1, \ldots, x_n$
- Values: $0, 1$
- Constraints: Given by the clauses of φ.

Constraint Satisfaction Problem

Input: (V, D, C)

- A finite set V of *variables*
- A finite *domain* D of values for the variables
- A set C of *constraints* (t, R) restricting the values that tuples of variables can take.
  - t: a tuple $t = (x_1, \ldots, x_m)$ of variables
  - R: a relation on D of arity $|t| = m$

Question: Does (V, D, C) have a solution?

Solution: A mapping $h : V \to D$ such that

$$h(t) = (h(x_1), \ldots, h(x_m)) \in R,$$

for every constraint $(t, R) \in C$.

An *assignment* of values to the variables such that all constraints are satisfied.

Example

- 3-SAT:

  Given a 3CNF-formula φ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, is φ satisfiable?

- 3-SAT as a CSP:

  – Variables $v_1, \ldots, v_n$

  – Domain $D = \{0, 1\}$

  – Constraints $((x, y, z), R_i)$, $i = 0, 1, 2, 3$

Clause	Relation
$(x \vee y \vee z)$	$R_0 = \{0, 1\}^3 - \{(0, 0, 0)\}$
$(\neg x \vee y \vee z)$	$R_1 = \{0, 1\}^3 - \{(1, 0, 0)\}$
$(\neg x \vee \neg y \vee z)$	$R_2 = \{0, 1\}^3 - \{(1, 1, 0)\}$
$(\neg x \vee \neg y \vee \neg z)$	$R_3 = \{0, 1\}^3 - \{(1, 1, 1)\}$

Example

- 3-COLORABILITY:

  Given a graph $\mathbf{H} = (V, E)$, is it 3-colorable?

- 3-COLORABILITY as a CSP:
  - The variables are the nodes in V
  - The domain is the set $D = \{R, G, B\}$ of three colors.
  - The constraints are the pairs $((u, v), Q)$, where $(u, v) \in E$ and Q is the relation

    $\{(R, G), (G, R), (R, B), (B, R), (R, G), (G, R)\}$.

More Examples

Homework Assignment #1:

Express the following problems as CSPs:

- 4-COLORABILITY
- CLIQUE
- HAMILTONIAN CYCLE
- LATIN SQUARE COMPLETION

Constraint Satisfaction Problems

Fundamental & ubiquitous problems in AI and CS:

- Boolean Satisfiability, Graph Colorability, ...
- Database Query Processing
- Planning and Scheduling
- Belief Maintenance
- Machine Vision
- $\cdots$
- Linguistics

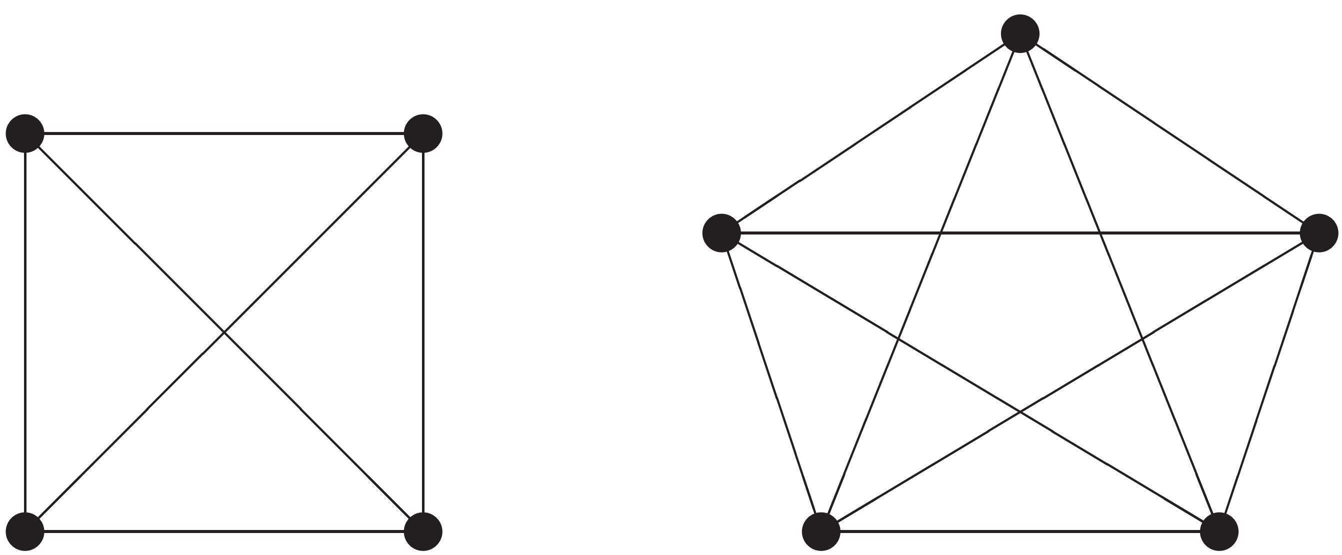
Homomorphisms

Definition: Consider two relational structures $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ and $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

$h : \mathbf{A} \to \mathbf{B}$ is a *homomorphism* if for every $i \leq m$ and every tuple $(a_1, \ldots, a_n) \in A^n$,

$$R_i^{\mathbf{A}}(a_1, \ldots, a_n) \implies R_i^{\mathbf{B}}(h(a_1), \ldots, h(a_n)).$$

Example: Clique $\mathbf{K}_4$ vs. Clique $\mathbf{K}_5$



- There is a homomorphism from $\mathbf{K}_4$ to $\mathbf{K}_5$;
- There is no homomorphism from $\mathbf{K}_5$ to $\mathbf{K}_4$.

The Homomorphism Problem

Definition: The HOMOMORPHISM PROBLEM

Given two relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism $h : \mathbf{A} \to \mathbf{B}$?

Fact: Feder & Vardi – 1993

CSP can be identified with the HOMOMORPHISM PROBLEM.

Example

3-SAT as a HOMOMORPHISM PROBLEM:

Given 3-CNF formula φ with variables $x_1, \ldots, x_n$:

- $\mathbf{A}^\varphi = (\{x_1, \ldots, x_n\}, R_0^\varphi, R_1^\varphi, R_2^\varphi, R_3^\varphi)$, where

$$\begin{aligned}
R_0^\varphi &= \{(x,y,z) : (x \vee y \vee z) \text{ is a clause of } \varphi\} \\
R_1^\varphi &= \{(x,y,z) : (\neg x \vee y \vee z) \text{ is a clause of } \varphi\} \\
R_2^\varphi &= \{(x,y,z) : (\neg x \vee \neg y \vee z) \text{ is a clause of } \varphi\} \\
R_3^\varphi &= \{(x,y,z) : (\neg x \vee \neg y \vee \neg z) \text{ is a clause of } \varphi\}
\end{aligned}$$

- $\mathbf{B} = (\{0,1\}, R_0, R_1, R_2, R_3)$, where

$$\begin{aligned}
R_0 &= \{0,1\}^3 - \{(0,0,0)\} \\
R_1 &= \{0,1\}^3 - \{(1,0,0)\} \\
R_2 &= \{0,1\}^3 - \{(1,1,0)\} \\
R_3 &= \{0,1\}^3 - \{(1,1,1)\}
\end{aligned}$$

Fact: φ is satisfiable iff there is hom. $h : \mathbf{A}^\varphi \to \mathbf{B}$.

Examples

3-Colorability as a Homomorphism Problem:

Fact: A graph $\mathbf{H} = (V, E)$ is 3-colorable

$$\Longleftrightarrow$$

there is a homomorphism $h : \mathbf{H} \rightarrow \mathbf{K}_3$, where $\mathbf{K}_3$ is the 3-clique, i.e., $\mathbf{K}_3 = (\{R, G, B\}, E_3)$, where

$$E_3 = \{(R, G), (G, R), (R, B), (B, R), (R, G), (G, R)\}.$$

k-Colorability as a Homomorphism Problem:

Fact: A graph $\mathbf{H} = (V, E)$ is k-colorable

$$\Longleftrightarrow$$

there is a homomorphism $h : \mathbf{H} \rightarrow \mathbf{K}_k$, where $\mathbf{K}_k$ is the k-clique.

More Examples

Homework Assignment #2:

Express the following problems as Homomorphism Problems:

- 2-Sat
- Clique
- Hamiltonian Cycle
- Latin Square Completion

From CSP to the Homomorphism Problem

Given a CSP-instance (V, D, C), let $R_1, \ldots, R_m$ be the distinct relations on D occurring in C.

Define the instance $(\mathbf{A}, \mathbf{B})$ of the Homomorphism Problem, where

$\mathbf{A} = (V, \{t : (t, R_1) \in C\}, \ldots, \{t : (t, R_m) \in C\})$

and

$\mathbf{B} = (D, R_1, \ldots, R_m)$.

Fact: The CSP instance (V, D, C) has a solution

$$\iff$$

there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

From the Homomorphism Problem to CSP

Given an instance $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ and $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ of the HOMOMORPHISM PROBLEM, define the instance (V, D, C) of CSP, where

- $V = A$ (i.e., the elements of A are variables);
- $D = B$ (i.e., the elements of B are values);
- $C = \{(t, R_i^{\mathbf{B}}) : t \in R_i^{\mathbf{A}}, 1 \leq i \leq m\}$.

Fact: There is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$

$$\iff$$

the CSP-instance (V, D, C) has a solution.

CSP vs. the Homomorphism Problem

Conclusion: Feder & Vardi – 1993

CSP = HOMOMORPHISM PROBLEM

Some Advantages:

- Clean, algebraic formulation of CSP.
- Makes it possible to apply methods from *universal algebra* to the study of CSP.
- Makes the connections between CSP and database theory transparent.

CSP vs. the Homomorphism Problem

- CSP = Homomorphism Problem

- "A *constraint network* consists of a finite set of variables, each associated with a set of discrete values, and a set of constraints. Each of the constraints is expressed as a relation, defined on some subset of variables, whose tuples are all the simultaneous value assignments to the members of this variable subset that, as far as this constraint is concerned, are legal[a]."

  Encyclopedia of Artificial Intelligence – 1992

[a]This does not mean that the actual representation of any constraint is necessarily in the form of its defining relation, but that the relation can, in principle, be generated using the constraints' specification without the need to consult other constraints in the network.

Course Outline

Constraint Satisfaction Problems

- Definition & Examples of CSP
- CSP & the Homomorphism Problem
- CSP & Database Theory
- Computational Complexity of CSP
  - Uniform & Non-Uniform CSP
  - Schaefer's Dichotomy Theorem for Boolean CSP
  - The Feder-Vardi Dichotomy Conjecture for general CSP
- The Pursuit of Tractable Cases of CSP
  - Datalog & Pebble Games
  - Consistency Properties & Bounded Treewidth

Relational Databases

Definition:

- *Relation Schema:* $R(A_1, \ldots, A_k)$

  R: relation name; $A_1, \ldots, A_k$: attribute names.

- *Relation conforming with R:*

  A set R' of k-tuples

- *Database Schema:* A set $\mathcal{S}$ of relation schemas

- *Database over* $\mathcal{S}$*:* A set S of relations conforming with the relation schemas in $\mathcal{S}$.

  Database = Relational Structure

Example:

- Relation Schemas:

  ENROLLS(Student, Course)

  TEACHES(Faculty, Course)

- Database Schema:

  {ENROLLS, TEACHES}.

Relational Databases

Example: (continued)

- Relation Schemas:

  ENROLLS(Student, Course)

  TEACHES(Faculty, Course)

- Relation ENROLLS

Student	Course
Adams	CS101
Bing	Math10
...	...

- Relation TEACHES

Faculty	Course
Euler	Math10
Turing	CS130
Codd	CS180
...	...

Relational Join

Notation: Given relations $R_1, \ldots, R_m$,

$$R_1 \bowtie R_2 \bowtie \cdots \bowtie R_m$$

is the *relational join* or, simply, the *join* of $R_1, \ldots, R_m$.

Definition: Given

$R_1(A, B, C)$, $R_2(B, C, D)$, $R_3(A, D, E)$,

$$\begin{aligned} R_1 \bowtie R_2 \bowtie R_3 \quad = \quad & \{(a, b, c, d, e) : \\ & R_1(a, b, c) \wedge R_2(b, c, d) \wedge R_3(a, d, e)\} \end{aligned}$$

Example: TAUGHT-BY(Student,Course,Faculty)

$$\text{TAUGHT-BY} \; = \; \text{ENROLLS} \bowtie \text{TEACHES}$$

Fact: Computing joins is the most frequent and most costly operation in database systems. (Why?)

Relational Joins

- Relation ENROLLS

Student	Course
Adams	CS101
Bing	Math10
...	...

- Relation TEACHES

Faculty	Course
Euler	Math10
Turing	CS130
Codd	CS180
...	...

- Relation TAUGHT-BY

Student	Course	Faculty
Bing	Math10	Euler
...	...	...

CSP and Joins

- Consider a CSP instance (V, D, C), where

$$\begin{aligned} V &= \{X_1, \ldots, X_n\} \\ C &= \{C_1, \ldots, C_m\} \\ C_i &= (t_i, R_i) \\ t_i &= (X_{j_1}, \ldots, X_{j_i}). \end{aligned}$$

- For each (t_i, R_i), view the occurrence of R_i as a relation over the relation schema $R_i(X_{j_1}, \ldots, X_{j_i})$.

Fact: Bibel, Gyssens, Jeavons, ...

$$R_1 \bowtie \cdots \bowtie R_m \;=\; \text{Solutions}((V, D, C)).$$

In particular,

(V, D, C) has a solution $\iff R_1 \bowtie \cdots \bowtie R_m \neq \emptyset$.

Conclusion: CSP is a database problem.

CSP and Databases

Fact: There are strong links between CSP and key problems in database systems:

- Relational Join Evaluation
- Conjunctive Query Evaluation (CQE)
- Conjunctive Query Containment (CQC)

Queries

Definitions:

- *Class $\mathcal{C}$ of structures:* a collection of relational structures closed under isomorphisms.
- *k-ary Query Q on $\mathcal{C}$:*

  a mapping Q with domain $\mathcal{C}$ and such that

  – $Q(\mathbf{A})$ is a k-ary relation on $\mathbf{A}$, for $\mathbf{A} \in \mathcal{C}$;

  – Q is *preserved under isomorphisms*, i.e., if $h : \mathbf{A} \to \mathbf{B}$ is an isomorphism, then

  $$Q(\mathbf{B}) \;=\; h(Q(\mathbf{A})).$$

- *Boolean Query Q on $\mathcal{C}$:*

  a mapping $Q : \mathcal{C} \to \{0,1\}$ preserved under isomorphisms. Thus, Q can be identified with the subclass $\mathcal{C}'$ of $\mathcal{C}$, where

  $$\mathcal{C}' \;=\; \{\mathbf{A} \in \mathcal{C} :\; Q(\mathbf{A}) = 1\}.$$

Examples of Queries

- PATH OF LENGTH 2 – $P2$:

  Binary query on graphs $\mathbf{H} = (V, E)$ such that

  $P2(\mathbf{H}) = \{(a, b) \in V^2 \colon (\exists c)(E(a, c) \wedge E(c, b))\}$.

- ARTICULATION POINT – AP:

  Unary query on graphs $\mathbf{H} = (V, E)$ such that

  $AP(\mathbf{H}) = \{a \in V \colon a \text{ is an articulation point of } \mathbf{H}\}$.

- CONNECTIVITY – CN:

  Boolean query on graphs $\mathbf{H} = (V, E)$ such that

$$CN(\mathbf{H}) = \begin{cases} 1 & \text{if } \mathbf{H} \text{ is connected} \\ 0 & \text{otherwise.} \end{cases}$$

- k-COLORABILITY, $k \geq 2$,

  HAMILTONIAN CYCLE, ...

Definability of Queries

Let L be a logic and $\mathcal{C}$ a class of structures

- A k-ary query Q on $\mathcal{C}$ is *L-definable* if there is an L-formula $\varphi(x_1, \ldots, x_k)$ with $x_1, \ldots, x_k$ as free variables and such that for every $\mathbf{A} \in \mathcal{C}$

  $$Q(\mathbf{A}) \;=\; \{(a_1, \ldots, a_k) \in A^k : \mathbf{A} \models \varphi(a_1, \ldots, a_k)\}.$$

- A Boolean query Q on $\mathcal{C}$ is *L-definable* if there is an L-sentence ψ such that for every $\mathbf{A} \in \mathcal{C}$

  $$Q(\mathbf{A}) = 1 \iff \mathbf{A} \models \psi.$$

Conjunctive Queries

Definition: A *conjunctive query* is a query definable by a FO-formula in prenex normal form built from atomic formulas, $\wedge$, and $\exists$ only.

$$(\exists z_1 \ldots \exists z_m)\psi(x_1, \ldots, x_k, z_1, \ldots, z_m),$$

where ψ is a conjunction of atomic formulas.

Also, written as a *rule:*

$$Q(x_1, \ldots, x_k) \ :- \ R(y_2, x_3, x_1), S(x_1, y_3), \ldots, S(y_7, x_2)$$

Examples:

- PATH OF LENGTH 2 (Binary query)

$$(\exists z)(E(x_1, z) \wedge E(z, x_2)$$

$$P2(x_1, x_2) \ :- \ E(x_1, z), E(z, x_2)$$

- CYCLE OF LENGTH 3 (Boolean query)

$$(\exists x_1 \exists x_2 \exists x_3)(E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1))$$

$$Q \ :- \ E(x_1, x_2), E(x_2, x_3), E(x_3, x_1)$$

Conjunctive Queries

- Every relational join *is* a conjunctive query

  $R_1(A, B, C), R_2(B, C, D)$,

  $R_1 \bowtie R_2(x, y, z, w) \quad :- \quad R_1(x, y, z), R_2(y, z, w)$

- Conjunctive Queries are the most frequently asked queries in databases (a.k.a. SPJ queries)

- The main construct of SQL expresses conjunctive queries

  `SELECT` $R_1.A, R_2.D$

  `FROM` R_1, R_2

  `WHERE` $R_1.B = R_2B$ `AND` $R_1.C = R_3.D$

Conjunctive Query Evaluation

A fundamental problem about conjunctive queries

Definition: *Conjunctive Query Evaluation* CQE

- Given a CQ Q and a structure $\mathbf{A}$, find

$$Q(\mathbf{A}) = \{(a_1, \ldots a_k) : \mathbf{A} \models Q(a_1, \ldots, a_k)\}$$

- For Boolean queries Q, this becomes:
  Given Q and $\mathbf{A}$, does $\mathbf{A} \models Q$? (is $Q(\mathbf{A}) = 1$?)

Examples:

- Given a graph H, find all pairs of nodes connected by a path of length 4.

- Given a graph H, does it contain a triangle?

Conjunctive Query Containment

A fundamental problem about conjunctive queries

Definition: *Conjunctive Query Containment* CQC

- Given two k-ary CQs Q_1 and Q_2, is it true that for every structure $\mathbf{A}$,

$$Q_1(\mathbf{A}) \subseteq Q_2(\mathbf{A})?$$

- For Boolean queries, this becomes:

  Given two Boolean queries Q_1 and Q_2, does $Q_1 \models Q_2$? (does Q_1 logically imply Q_2?)

Examples:

- Is it true that if two nodes of a graph $\mathbf{H}$ are connected by a path of length 4, then they are also connected by a path of length 3?

- It is true that if a graph $\mathbf{H}$ contains a $\mathbf{K}_4$, then it also contains a $\mathbf{K}_3$?

CQE, CQC, CSP

Theorem: Chandra & Merlin – 1977

CQE and CQC are the *same* problem.

Fact: K . . . & Vardi – 1998

CSP and CQC are the *same* problem

Question: What is the common link?

CQE, CQC, CSP

Theorem: Chandra & Merlin – 1977

CQE and CQC are the *same* problem.

Fact: K . . . & Vardi – 1998

CSP and CQC are the *same* problem

Question: What is the common link?

Answer: The Homomorphism Problem

Canonical CQs and Canonical Structures

Definition: *Canonical Conjunctive Query*

Given $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$, the *canonical CQ of* $\mathbf{A}$ is the Boolean CQ $Q^{\mathbf{A}}$ with the elements of A as variables and the "facts" of $\mathbf{A}$ as conjuncts:

$$Q^{\mathbf{A}} \ :- \ \bigwedge_{i=1}^{m} \bigwedge_{\mathbf{t}} R_i^{\mathbf{A}}(\mathbf{t})$$

Definition: *Canonical Structure*

Given a Boolean conjunctive query Q, let $\mathbf{A}^Q$ be the structure with the variables of Q as elements and the conjuncts of Q as "facts".

Example:

- $\mathbf{A} = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\}$

$$Q^{\mathbf{A}} \ :- \ E(x, y) \wedge E(y, z) \wedge E(z, x)$$

- $Q \ :- \ E(x, y) \wedge E(x, z)$

$$\mathbf{A}^Q = (\{a, b, c), \{(a, b), (a, c)\}$$

Homomorphisms, CQC and CQE

Theorem: Chandra & Merlin – 1977

For relational structures $\mathbf{A}$ and $\mathbf{B}$, TFAE

- There is a homomorphism $h : \mathbf{A} \rightarrow \mathbf{B}$
- $\mathbf{B} \models Q^{\mathbf{A}}$ (i.e., $Q^{\mathbf{A}}(\mathbf{B}) = 1$)
- $Q^{\mathbf{B}} \subseteq Q^{\mathbf{A}}$

Theorem: Chandra & Merlin – 1977

For conjunctive queries Q_1 and Q_2, TFAE

- $Q_1 \subseteq Q_2$
- There is a homomorphism $h : \mathbf{A}^{Q_2} \rightarrow \mathbf{A}^{Q_1}$
- $\mathbf{A}^{Q_1} \models Q_2$ (i.e., $Q_2(\mathbf{A}^{Q_1}) = 1$)

Homomorphisms, CQC, and CQE

Example: 3-COLORABILITY

For a graph $\mathbf{H}$, the following are equivalent:

1. $\mathbf{H}$ is 3-colorable
2. There is a homomorphism $h : \mathbf{H} \rightarrow \mathbf{K_3}$
3. $\mathbf{K}_3 \models Q^{\mathbf{H}}$
4. $Q^{\mathbf{K}_3} \subseteq Q^{\mathbf{H}}$

Proof:

(2) $\Longrightarrow$ (3): A homomorphism $h : \mathbf{H} \rightarrow \mathbf{K_3}$ provides witnesses in $\mathbf{K}_3$ for the existential quantifiers in $Q^{\mathbf{H}}$.

(3) $\Longrightarrow$ (4): If $\mathbf{K}_3 \models Q^{\mathbf{H}}$ and $\mathbf{A} \models Q^{\mathbf{K}_3}$, then there is a homomorphism $h : \mathbf{H} \rightarrow \mathbf{K_3}$ and a homomorphism and $h^* : \mathbf{K_3} \rightarrow \mathbf{A}$. ▮

The composition $h^* \circ h : \mathbf{H} \rightarrow \mathbf{A}$ provides witnesses in $\mathbf{A}$ for the existential quantifiers in $Q^{\mathbf{H}}$. ▮

CSP, CQE, and CQC

Theorem: Chandra & Merlin – 1977

For relational structures $\mathbf{A}$ and $\mathbf{B}$, TFAE

- There is a homomorphism $h : \mathbf{A} \to \mathbf{B}$
- $\mathbf{B} \models Q^{\mathbf{A}}$ (i.e., $Q^{\mathbf{A}}(\mathbf{B}) = 1$)
- $Q^{\mathbf{B}} \subseteq Q^{\mathbf{A}}$

Conclusion:

CSP, CQE, and CQC are the *same* problem.

Homework Assignment #3

- Complete the proof of the Chandra-Merlin Theorem

- Express the CLIQUE problem as a Conjunctive Query Evaluation problem.

- Express the CLIQUE problem as a Conjunctive Query Containment Problem.

- Express the LATIN SQUARE COMPLETION problem as a Conjunctive Query Evaluation Problem.

Course Outline

Constraint Satisfaction Problems

- Definition & Examples of CSP
- CSP & the Homomorphism Problem
- CSP & Database Theory
- Computational Complexity of CSP
  - Uniform & Non-Uniform CSP
  - Schaefer's Dichotomy Theorem for Boolean CSP
  - The Feder-Vardi Dichotomy Conjecture for general CSP
- The Pursuit of Tractable Cases of CSP
  - Datalog & Pebble Games
  - Consistency Properties & Bounded Treewidth

Goals and Prerequisites

Goals:

- Show that Constraint Satisfaction Problems (CSP) form a broad class of algorithmic problems that arise naturally in several different areas.

- Present an overview of CSP with emphasis on the computational complexity aspects of CSP and on the connections with database theory and logic

Prerequisites:

- Familiarity with the basics of first-order logic.

- Familiarity with the basics of computational complexity (PTIME and NP-completeness).

Computational Complexity

Computational Complexity:

- *The quantitative study of solvability*

  Y. Hartmanis – 1993

- Solvable decision problems are placed into classes according to the time/space resources required to solve them.

Some Major Complexity Classes:

P	polynomial time
NP	nondeterministic polynomial time
EXPTIME	exponential time

Complexity Classes

Fact: The following containments hold:

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$$

Theorem: Hartmanis & Stearns – 1965

$$\mathrm{P} \subset \mathrm{EXPTIME}$$

Open Problems:

- Is $\mathrm{P} \neq \mathrm{NP}$?
- Is $\mathrm{NP} \neq \mathrm{EXPTIME}$?

Note: It is generally believed that $\mathrm{P} \neq \mathrm{NP}$.

NP-Complete Problems

Definition: Let Q be a decision problem (a Boolean query). We say that Q is NP-*complete* if

1. $Q \in \mathrm{NP}$.
2. Q is NP-*hard*, i.e., for every $Q' \in \mathrm{NP}$, there a polynomial-time many-one reduction f of Q' to Q:

$$x \in Q' \Longleftrightarrow f(x) \in Q.$$

Theorem: S. Cook – 1971, R. Karp 1972, ...

The following problems are NP-complete:

- k-SAT, $k \geq 3$
- k-COLORABILITY, $k \geq 3$
- CLIQUE
- HAMILTONIAN CYCLE
- LATIN SQUARE COMPLETION

NP-Complete Problems

Fact: Let Q be an NP-complete problem.

The following are equivalent:

- $\mathrm{P} = \mathrm{NP}$
- $Q \in \mathrm{P}$.

Remark:

- NP-complete problems hold the *secret* of whether or not $\mathrm{P} = \mathrm{NP}$.
- Establishing that Q is NP-complete is viewed as evidence that $Q \notin \mathrm{P}$ and, hence, it is an *intractable* algorithmic problem.

Complexity of CSP

Definition: Uniform CSP

$$\{(\mathbf{A}, \mathbf{B}) : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

(Homomorphism Problem)

Theorem: Uniform CSP is NP-complete

Proof:

- Uniform CSP is in NP:

  Given $\mathbf{A}$, $\mathbf{B}$, "guess" $h : \mathbf{A} \to \mathbf{B}$ and verify it is a homomorphism.

- Uniform CSP is NP-hard:

  It contains NP-hard problems as special cases (e.g., 3-SAT) ∎

Complexity of CQC

Definition: CQC Problem

Given Boolean queries Q_1, Q_2, does $Q_1 \models Q_2$?

Theorem: CQC is NP-complete

Proof: Same as Uniform CSP

Remark:

- Recall that Boolean CQs are FO-sentences built from atomic formulas, $\wedge$, and $\exists$ only.

  So, for such sentences, logical implication is decidable.

- In contrast, for arbitrary FO-sentences, logical implication is undecidable.

Coping with NP-Completeness

Question: How to cope with NP-completeness?

Answer: Garey & Johnson – 1979

- Design *heuristic* algorithms that may work well in practice.

- Identify polynomial-time solvable cases of the problem, the so-called *islands of tractability*, by imposing restrictions on the inputs.

  Example 1:

  – 3-SAT is NP-complete.

  – HORN 3-SAT is in P.

  Example 2:

  – 3-COLORABILITY is NP-complete.

  – 3-COLORABILITY on graphs of *bounded treewidth* is in P.

Restricting Uniform CSP

Research Program: Identify all islands of tractability of UNIFORM CSP.

Definition: Let $\mathcal{A}$, $\mathcal{B}$ be two classes of structures.

$$\mathrm{CSP}(\mathcal{A},\mathcal{B}) = \{(\mathbf{A},\mathbf{B}) : \mathbf{A} \in \mathcal{A}, \mathbf{B} \in \mathcal{B} \text{ and}$$
$$\exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

Research Program: Identify all $\mathcal{A}$, $\mathcal{B}$ such that $\mathrm{CSP}(\mathcal{A},\mathcal{B})$ is in P.

Non-Uniform CSP

Definition: Fix a structure $\mathbf{A}$.

$\mathrm{CSP}(\{\mathbf{A}\}, All) = \{\mathbf{B} : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$

Fact: For every fixed $\mathbf{A}$,

$$\mathrm{CSP}(\{\mathbf{A}\}, All) \in \mathrm{P}$$

Proof: Exercise.

Definition: Fix a structure $\mathbf{B}$:

$\mathrm{CSP}(All, \{\mathbf{B}\}) = \{\mathbf{A} : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$

Fact: The complexity of $\mathrm{CSP}(All, \{\mathbf{B}\})$ depends on $\mathbf{B}$.

Proof:

- $\mathrm{CSP}(All, \{\mathbf{K}_3\})$ is NP-complete (3-Colorability)
- $\mathrm{CSP}(All, \{\mathbf{K}_2\})$ is in P. (2-Colorability) ∎

Complexity of Non-Uniform CSP

Notation: For simplicity, for every fixed structure $\mathbf{B}$, we put

$$\mathrm{CSP}(\mathbf{B}) = \mathrm{CSP}(\mathit{All}, \{\mathbf{B}\})$$

Research Program:

- Identify the islands of tractability of $\mathrm{CSP}(\mathbf{B})$.
- In other words, characterize the structures $\mathbf{B}$ for which $\mathrm{CSP}(\mathbf{B})$ is solvable in polynomial time.

Boolean Non-Uniform CSP

Definition: Boolean Non-Uniform CSP

CSP($\mathbf{B}$) with $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_1^{\mathbf{B}})$

Fact: Boolean Non-Uniform CSP problems are GENERALIZED SATISFIABILITY PROBLEMS, i.e., variants of Boolean satisfiability in CNF.

Examples: Each of the following problems is a CSP($\mathbf{B}$) problem for a suitable Boolean $\mathbf{B}$.

- 3-SAT: $\mathbf{B} = (\{0,1\}, R_0, R_1, R_2, R_3)$
- 2-SAT: $\mathbf{B} = (\{0,1\}, R'_0, R'_1, R'_2)$
- POSITIVE 1-IN-3-SAT:

  3CNF-formulas with clauses $(x \vee y \vee z)$;

  exactly one variable is true in each clause.

  $$\mathbf{B} = (\{0,1\}, R_{1/3}),$$

  where

  $$R_{1/3} = \{(1,0,0), (0,1,0), (0,0,1).$$

Complexity of Boolean Non-Uniform CSP

Schaefer's Dichotomy Theorem – 1978

- If $\mathbf{B}$ is Boolean structure, then CSP($\mathbf{B}$) is in P or it is NP-complete.
- Moreover, there is a polynomial-time algorithm to decide, given a Boolean structure $\mathbf{B}$, whether CSP($\mathbf{B}$) is in P or it is NP-complete. ∎

Question: Why is this a *dichotomy* theorem?

The Fine Structure of NP

Theorem: Ladner - 1975

If P $\neq$ NP, then there is a problem Q such that

- $Q \in \mathrm{NP} - \mathrm{P}$;
- Q is *not* NP-complete. ▮

NP-complete
NP – P not NP-complete
P

Remark: Consequently, it is conceivable that a family of NP-problems contains problems of such intermediate complexity problems.

This possibility cannot be ruled out *a priori*.

Dichotomy of Boolean Non-Uniform CSP

Schaefer's Dichotomy Theorem – 1978

- If $\mathbf{B}$ is Boolean structure, then CSP($\mathbf{B}$) is in P or it is NP-complete.

CSP($\mathbf{B}$)

NP-complete
NP – P, not NP-complete
P

- Moreover, there is a polynomial-time algorithm to decide, given a Boolean structure $\mathbf{B}$, whether CSP($\mathbf{B}$) is in P or it is NP-complete. $\blacksquare$

Question: What is the boundary in this dichotomy?

Dichotomy of Boolean Non-Uniform CSP

Proof Architecture:

$$\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$$

- Identify a number of cases for which CSP($\mathbf{B}$) is in P.

  Each such case will be described in terms of *conditions* on the relations $R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}}$ of $\mathbf{B}$.

- Show that CSP($\mathbf{B}$) is NP-complete in all other cases.

- Show that there is a polynomial-time algorithm for checking whether the conditions describing the tractable cases hold.

Tractable Cases of Boolean CSP($\mathbf{B}$)

Definition: $R \subseteq \{0,1\}^k$, for some $k \geq 1$.

- R is 0-*valid* if it contains the k-tuple $(0,\ldots,0)$.
- R is 1-*valid* if it contains the k-tuple $(1,\ldots,1)$.

Fact: $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$

- If every $R_i^{\mathbf{B}}$ is 0-valid, then CSP($\mathbf{B}$) is in P.
- If every $R_i^{\mathbf{B}}$ is 1-valid, then CSP($\mathbf{B}$) is in P.

Proof: Trivial

- The constant function $h(x) = 0$ is a homomorphism.
- The constant function $h(x) = 1$ is a homomorphism. ∎

Relations and Solutions

Notation: Let φ be a Boolean formula with k variables $x_1, \ldots, x_k$.

$\mathrm{Sol}(\varphi)$ is the set of all k-tuples $(a_1, \ldots, a_k)$ such that if s is a truth assignment with $s(x_i) = a_i$, $1 \leq i \leq k$, then $s(\varphi) = 1$.

Example: If φ is $(\neg x \vee y)$, then

$$\mathrm{Sol}(\varphi) = \{(0,0), (0,1), (1,1)\}.$$

Remark: For all remaining tractable cases, the condition on the relations of $\mathbf{B}$ will assert that each relation $R_i^{\mathbf{B}}$ is equal to $\mathrm{Sol}(\varphi)$, where φ belongs to a certain class of "well-behaved" Boolean formulas

(the class of formulas will change from tractable case to tractable case).

Tractable Cases of Boolean CSP($\mathbf{B}$)

Definition: $R \subseteq \{0,1\}^k$, for some $k \geq 1$.

R is *bijunctive* if there is a 2-CNF formula φ such that $R = \mathrm{Sol}(\varphi)$.

Theorem: $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$

If every $R_i^{\mathbf{B}}$ is bijunctive, then CSP($\mathbf{B}$) is in P.

Proof: 2-SAT is in P.

The *resolution* algorithm runs in polynomial time on 2-CNF formulas, because

- Every resolvent of two 2-clauses is a 2-clause:

$$\begin{array}{ccc} \{l_1, l_2\} & & \{\overline{l_1}, l_3\} \\ & \searrow \quad \swarrow & \\ & \{l_2, l_3\} & \end{array}$$

- Given n variables, there is a polynomial number of 2-clauses involving these variables. ∎

Tractable Cases of Boolean CSP(**B**)

Definition: $R \subseteq \{0,1\}^k$, $k \geq 1$.

- R is Horn if $R = \mathrm{Sol}(\varphi)$ for some *Horn formula* φ, i.e., a CNF-formula in which each conjunct has at most one positive literal.

$$x \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee w)$$

- R is *dual Horn* $R = \mathrm{Sol}(\varphi)$ for some *dual Horn formula* φ, i.e., a CNF-formula in which each conjunct has at most one negative literal.

$$\neg x \wedge (x \vee \neg z) \wedge (y \vee z \vee \neg w)$$

Theorem: $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$

- If every $R_i^{\mathbf{B}}$ is Horn, then CSP($\mathbf{B}$) is in P.
- If every $R_i^{\mathbf{B}}$ is dual Horn, then CSP($\mathbf{B}$) is in P.

Proof: *Unit Propagation* algorithm. ∎

Tractable Cases of Boolean CSP($\mathbf{B}$)

Definition: $R \subseteq \{0,1\}^k$, $k \geq 1$.

R is *affine* if it is the set of solutions of a system of linear equations over the 2-element field.

Example: $R = \{(0,1),(1,0)\}$ is affine, because

$$R = \mathrm{Sol}(x \oplus y = 1).$$

Theorem: $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$

If every $R_i^{\mathbf{B}}$ is affine, then CSP($\mathbf{B}$) is in P.

Proof: Use *Gaussian Elimination* algorithm. ∎

Homework Assignment #4

- Show that 2-SAT can be solved in polynomial time using *resolution*.
- Show that Horn SAT and Dual Horn SAT can be solved in polynomial time using suitable *unit propagation* algorithms.
- Show that Affine SAT can be solved in polynomial time using *Gaussian elimination*.
- Show that the formula $(x \vee y)$ is *not* logically equivalent to any Horn formula.
- Show that the formula $(x \vee y) \wedge (\neg x \vee \neg y)$ is not logically equivalent to any (dual) Horn formula.

Schaefer Structures

Definition: $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$

$\mathbf{B}$ is *Schaefer* if at least one of the following six conditions holds:

- Every relation $R_i^{\mathbf{B}}$ is 0-valid.
- Every relation $R_i^{\mathbf{B}}$ is 1-valid.
- Every relation $R_i^{\mathbf{B}}$ is bijunctive.
- Every relation $R_i^{\mathbf{B}}$ is Horn.
- Every relation $R_i^{\mathbf{B}}$ is dual Horn.
- Every relation $R_i^{\mathbf{B}}$ is affine.

Otherwise, we say that $\mathbf{B}$ is *non-Schaefer*.

Schaefer's Dichotomy Theorem

Theorem: Let $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

- If $\mathbf{B}$ is Schaefer, then CSP($\mathbf{B}$) is in P.

- If $\mathbf{B}$ is non-Schaefer, then CSP($\mathbf{B}$) is NP-complete.

Proof:

- The first part has already been established.

- The second part requires much more work.

  The key ingredient is a theorem about the expressive power of conjunctive queries on non-Schaefer structures.

Schaefer's Expressibility Theorem

Note: Recall that a *conjunctive query* is a FO-formula built from atomic formulas, $\wedge$, and $\exists$ only.

Theorem: Let $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

If $\mathbf{B}$ is non-Schaefer, then every Boolean relation $R \subset \{0,1\}^k$, $k \geq 1$, is definable on $\mathbf{B}$ by some conjunctive query over $\mathbf{B}$ with constants 0 and 1.

This means that for every $k \geq 1$ and for every $R \subset \{0,1\}^k$, there is a formula $\psi(x_1, \ldots, x_k, y, z)$ that is built from $R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}}$, $\wedge$, and $\exists$, and has the property that for every $(a_1, \ldots, a_k) \in \{0,1\}^k$

$$(a_1, \ldots, a_k) \in R \iff \mathbf{B} \models \psi(a_1, \ldots, a_k, 0, 1).$$

Remark: Intutitively, this result asserts that on non-Schaefer structures, conjunctive queries have *maximum* expressive power (they can define all Boolean relations).

Schaefer's Dichotomy Theorem

Theorem: Let $\mathbf{B} = (\{0,1\}, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

- If $\mathbf{B}$ is Schaefer, then CSP($\mathbf{B}$) is in P.
- If $\mathbf{B}$ is non-Schaefer, then CSP($\mathbf{B}$) is NP-complete.
- Moreover, there are simple polynomial-time criteria to test if $\mathbf{B}$ is Schaefer.

Remark: These criteria involve *closure properties* of the relations of $\mathbf{B}$.

Closure Properties

Theorem: Let $R \subseteq \{0,1\}^k$, $k \geq 1$.

- R is bijunctive iff it is closed under the 3-ary *majority* function $\mathrm{maj}(x, y, z)$.
- S is Horn iff it is closed under $g(x, y) = x \wedge y$.
- S is dual Horn iff it is is closed under $g'(x, y) = x \vee y$.
- S is affine iff it is closed under $h(x, y, z) = x \oplus y \oplus z$.

Closure Properties

Example: Let $R = \{(0,1), (1,0), (1,1)\}$

- R is *not* Horn, because

$$(0,1) \wedge (1,0) = (0,0) \notin R.$$

- R is *not* affine, because

$$(0,1) \oplus (1,0) \oplus (1,1) = (0,0) \notin R.$$

Example: Let $R = \{(0,0), (0,1), (1,0)\}$

- R is *not* dual Horn, because

$$(0,1) \vee (1,0) = (1,1) \notin R.$$

- R is *not* affine, because

$$(0,0) \oplus (0,1) \oplus (1,0) = (1,1) \notin R.$$

Positive 1-in-3 SAT

- 3-CNF formula with clauses $(x \vee y \vee z)$; exactly one variable/clause is true.
- Positive-1-In-3-SAT $=$ CSP$((\{0,1\}, R_{1/3}))$

$$R_{1/3} = \{(1,0,0), (0,1,0), (0,0,1).$$

Fact: Positive 1-In-3-SAT is NP-complete

Proof: Apply Schaefer's Dichotomy Theorem

- $R_{1/3}$ is not bijunctive, because

$$\text{maj}((1,0,0),(0,1,0),(0,0,1)) = (0,0,0) \notin R.$$

- $R_{1/3}$ is neither Horn nor dual Horn, since

$$(1,0,0) \wedge (0,1,0) = (0,0,0) \notin R_{1/3}.$$

$$(1,0,0) \vee (0,1,0) = (1,1,0) \notin R_{1/3}.$$

- $R_{1/3}$ is not affine, since

$$(1,0,0) \oplus (0,1,0) \oplus (0,0,1) = (1,1,1) \notin R_{1/3}.$$

Homework Assignment #5

- Given an example of a Boolean structure $\mathbf{B}$ such that CSP($\mathbf{B}$) is NP-complete, but every relation in $\mathbf{B}$ is Horn or dual Horn.

  Repeat for the other combinations of tractable cases in Schaefer's Dichotomy Theorem.

- Use Schaefer's Dichotomy Theorem to show that 2-COLORABILITY is in P.

- Use Schaefer's Dichotomy Theorem to show that the following are NP-complete:
  - NOT-ALL-EQUAL 3-SAT
  - MONOTONE 3-SAT

Complexity of CSP($\mathbf{B}$)

Conclusion: Schaefer's Dichotomy Theorem yields a complete classification of the complexity of CSP($\mathbf{B}$) for Boolean structures $\mathbf{B}$.

Question: What about arbitrary structures $\mathbf{B}$?

Complexity of CSP($\mathbf{B}$)

Conclusion: Schaefer's Dichotomy Theorem yields a complete classification of the complexity of CSP($\mathbf{B}$) for Boolean structures $\mathbf{B}$.

Question: What about arbitrary structures $\mathbf{B}$?

Dichotomy Conjecture: Feder & Vardi –1993

For every structure $\mathbf{B}$, one of the following holds:

- CSP($\mathbf{B}$) is in P.
- CSP($\mathbf{B}$) is NP-complete.

Towards the Dichotomy Conjecture

Theorem: Hell & Nešetril – 1990

Let $\mathbf{B}$ be an *undirected* graph.

- If $\mathbf{B}$ is 2-Colorable, then CSP($\mathbf{B}$) is in P.
- If $\mathbf{B}$ not 2-Colorable, then CSP($\mathbf{B}$) is NP-complete.

Theorem: A. Bulatov – 2002 (unpublished)

The Dichotomy Conjecture is true for CSP($\mathbf{B}$), where $B = \{0, 1, 2\}$.

Note: The Dichotomy Conjecture remains open for CSP($\mathbf{B}$) with $|B| \geq 4$.

Islands of Tractability of CSP(**B**)

Fact: An extensive pursuit of tractable cases of CSP(**B**) has taken place over the years.

First Stage: Numerous particular results.

Second Stage: Broad sufficient conditions for tractability of CSP(**B**).

Fact: Unifying explanations for many tractability results have been discovered:

- Expressibility in Datalog

  Feder & Vardi – 1993

- Closure Properties of Constraints

  Connections with *universal algebra*

  Feder & Vardi – 1993

  Jeavons & collaborators – 1995 to present

Course Outline

Constraint Satisfaction Problems (CSP)

- Definition & Examples of CSP
- CSP & the Homomorphism Problem
- CSP & Database Theory
- Computational Complexity of CSP
  - Uniform & Non-Uniform CSP
  - Schaefer's Dichotomy Theorem for Boolean CSP
  - The Feder-Vardi Dichotomy Conjecture for general CSP
- The Pursuit of Tractable Cases of CSP
  - Datalog & Pebble Games
  - Consistency Properties & Bounded Treewidth

Datalog

Note: Recall that CQs can be written as *rules*:

PATH OF LENGTH 2 - $P2$: $(\exists z)(E(x_1, z) \wedge E(z, x_2))$

$$P2(x_1, x_2) \ :- \ E(x_1, z), E(z, x_2)$$

Definition:

- Datalog $=$ Conjunctive Queries + Recursion
  Logic Programming without function symbols
- A Datalog program is a finite set of rules given by conjunctive queries

$$T(\overline{x}) \ :- \ S_1(\overline{y}_1), \ldots, S_r(\overline{y}_r).$$

  - Some relation symbols may occur both in the *heads* and the *bodies* of rules.
    These are the *recursive* relation symbols or *intensional database predicates* (IDBs).
  - The remaining relation symbols are the *extensional database predicates* (EDBs).

Datalog Examples

Definition: TRANSITIVE CLOSURE Query TC

Given graph $\mathbf{H} = (V, E)$,

$TC(\mathbf{H}) = \{(a, b) \in V^2 : \text{there is a path from } a \text{ to } b\}$.

Example 1: Datalog program for TC

$$
\left|
\begin{array}{lcl}
S(x, y) & :- & E(x, y) \\
S(x, y) & :- & E(x, z) \wedge S(z, y)
\end{array}
\right.
$$

Example 2: Another Datalog program for TC

$$
\left|
\begin{array}{lcl}
S(x, y) & :- & E(x, y) \\
S(x, y) & :- & S(x, z) \wedge S(z, y)
\end{array}
\right.
$$

Note:

- E is the EDB.
- S is the IDB; it defines TC.

Datalog Examples

Definition: S. Cook – 1974

Path Systems $\mathbf{S} = (F, A, R)$

Given a finite set of *formulas* F, a set of *axioms* $A \subseteq F$, and a *rule of inference* $R \subseteq F^3$, compute the *theorems* of this system.

Example: Datalog program for Path Systems:

$$\left| \begin{array}{lll} T(x) & :- & A(x) \\ T(x) & :- & T(y), T(z), R(x, y, z) \end{array} \right.$$

Note:

- A and R are the EDBs.
- T is the IDB; it defines the theorems of $\mathbf{S}$.

Datalog Properties

Fact:

- If a query Q is defined by a Datalog program, then Q is in P.

- If a query Q is defined by a Datalog program, then Q is *monotone*, i.e., the query is preserved if new tuples are added to the EDBs.

Proof:

- Datalog programs can be evaluated "bottom-up" in a polynomial number of iterations.

- Conjunctive queries are monotone, because they are negation-free and $\neq$-free. ∎

Evaluation of Datalog Programs

Example : Datalog program for TC

$$\left| \begin{array}{lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & E(x,z) \wedge S(z,y) \end{array} \right.$$

Bottom-up Evaluation

$$\left| \begin{array}{lcl} S^1 & = & \emptyset \\ S^{m+1} & = & \{(a,b)) : \exists z (E(a,z) \wedge S^m(z,b)\} \end{array} \right.$$

Fact:

$$\begin{array}{lcl} TC & = & \bigcup_m S^m \\ TC & = & S^{|V|}. \end{array}$$

Datalog and CSP

Fact: Let $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

- In general, CSP($\mathbf{B}$) is *not* monotone. (Why?)

- Hence, CSP($\mathbf{B}$) is *not* expressible in Datalog.

However,

- $\overline{\mathrm{CSP}(\mathbf{B})}$ is monotone, where

  $\overline{\mathrm{CSP}(\mathbf{B})} = \{\mathbf{A} : \nexists \text{ homomorphism } h : \mathbf{A} \rightarrow \mathbf{B}\}$.

- Hence, it is conceivable that $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in Datalog (and, thus, it is in P).

Datalog and CSP

Fact: Feder & Vardi – 1993

Expressibility of $\overline{\mathrm{CSP}(\mathbf{B})}$ in Datalog is a unifying explanation for many tractability results about $\mathrm{CSP}(\mathbf{B})$.

Example: 2-Colorability $=$ $\mathrm{CSP}(\mathbf{K}_2)$

Datalog program for Non 2-Colorability

$$\left|\begin{array}{lcl} O(X,Y) & :- & E(X,Y) \\ O(X,Y) & :- & O(X,Z), E(Z,W), E(W,Y) \\ Q & :- & O(X,X) \end{array}\right.$$

Example: Let $\mathbf{B}$ be Boolean structure.

If $\mathbf{B}$ is bijunctive, Horn, or dual Horn, then $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in Datalog.

Datalog and CSP

Question: Fix $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

When is $\overline{\mathrm{CSP}(\mathbf{B})}$ expressible in Datalog?

Datalog and CSP

Question: Fix $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

When is $\overline{\mathrm{CSP}(\mathbf{B})}$ expressible in Datalog?

Answer: K … & Vardi – 1998, 2000

Expressibility of $\overline{\mathrm{CSP}(\mathbf{B})}$ in Datalog can be characterized in terms of

- Pebble Games;
- Consistency Properties.

Existential k-Pebble Games

Spoiler and Duplicator play on two structures $\mathbf{A}$ and $\mathbf{B}$. Each player uses k pebbles. In each move,

- Spoiler places a pebble on or removes a pebble from an element of $\mathbf{A}$.

- Duplicator tries to duplicate the move on $\mathbf{B}$.

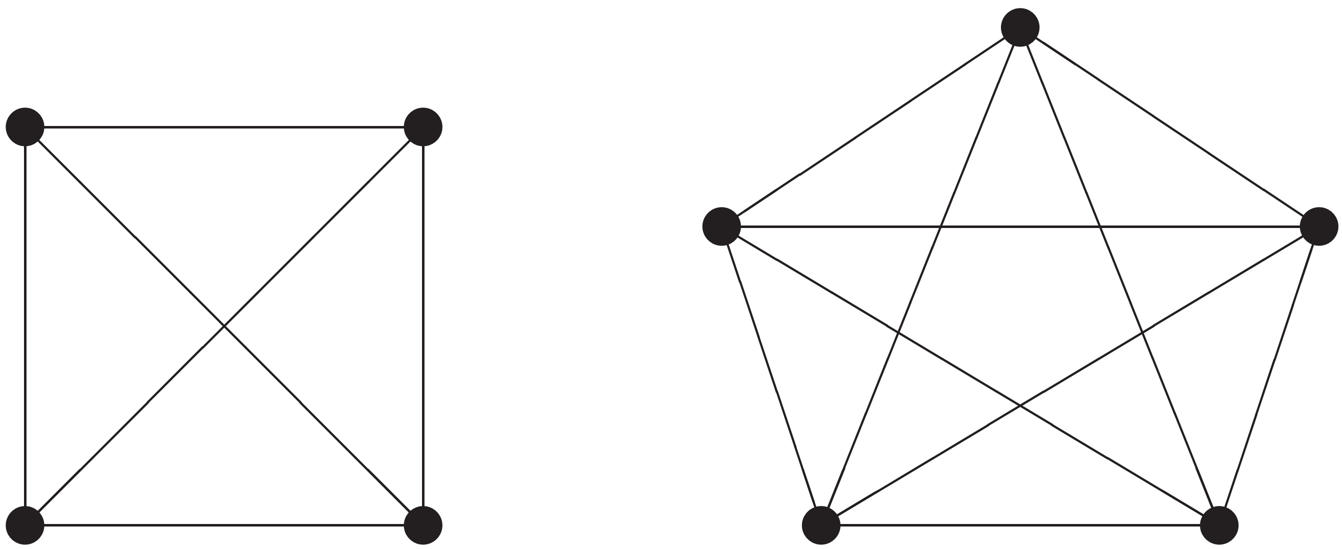
$$
\begin{array}{llllll}
\mathbf{A}: & a_1 & a_2 & \dots & a_l & \\
 & \downarrow & \downarrow & \cdots & \downarrow & \\
\mathbf{B}: & b_1 & b_2 & \dots & b_l & l \leq k
\end{array}
$$

- Spoiler *wins* the $(\exists, k)$-pebble game if at some point the mapping $a_i \mapsto b_i$, $1 \leq i \leq l$, is not a partial homomorphism.

- Duplicator *wins* the $(\exists, k)$-pebble game if the above never happens.

Example

Cliques of Different Size

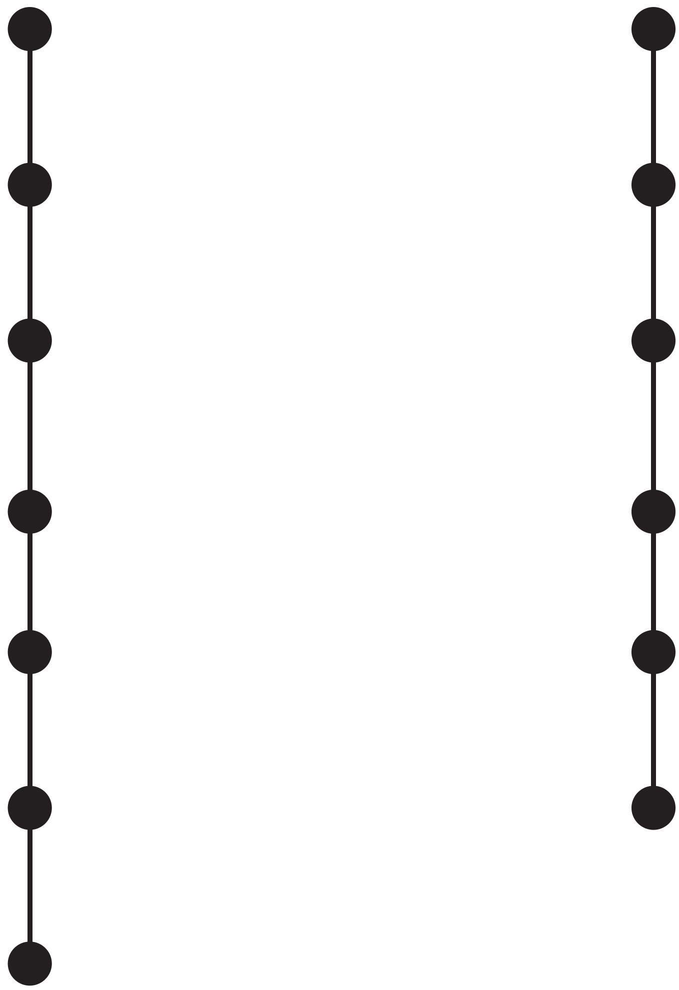


$\mathbf{K}_4$ $\mathbf{K}_5$

Fact: Let $\mathbf{K}_k$ be the k-clique

- Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$.
- Spoiler wins the $(\exists, k)$-pebble game on $\mathbf{K}_k$ and $\mathbf{K}_{k-1}$.

Linear Orders of Different Size



L_m $\qquad\qquad\qquad$ L_n

- Spoiler wins the $(\exists, 2)$-pebble game on L_m and L_n, where $m > n$.
- Duplicator wins the $(\exists, 2)$-pebble game on L_n and L_m, where $m > n$.

k-Datalog

Definition: A k-Datalog program is a Datalog program in which each rule

$$t_0 \ :- \ t_1, \ldots, t_m$$

has at most k distinct variables in the head t_0 and at most k distinct variables in the body $t_1, \ldots, t_m$.

Example: NON 2-COLORABILITY revisited

$$\left| \begin{array}{lcl} O(X,Y) & :- & E(X,Y) \\ O(X,Y) & :- & O(X,Z), E(Z,W), E(W,Y) \\ Q & :- & O(X,X) \end{array} \right.$$

Therefore,

NON 2-COLORABILITY is expressible in 4-Datalog.

k-Datalog and $(\exists, k)$-Pebble Games

Theorem: K ... & Vardi – 1995

- $(\exists, k)$-pebble games characterize expressibility in $\exists L^k_{\infty\omega}$ (an infinitary logic with k variables).
- On the class of all finite structures,

$$k\text{-Datalog} \subset \exists L^k_{\infty\omega}.$$

- There is a polynomial-time algorithm to decide whether, given two finite structures $\mathbf{A}$ and $\mathbf{B}$, the Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$.
- For each finite structure $\mathbf{B}$, there is a k-Datalog program $\pi_{\mathbf{B}}$ that expresses the query: given a finite σ-structure $\mathbf{A}$, does the Duplicator win the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$?

Datalog and CSP

Theorem: K … & Vardi – 1998

Let k be a positive integer and $\mathbf{B}$ a finite structure. Then the following are equivalent:

- $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog
- $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in $\exists \mathrm{L}^k_{\infty\omega}$
- $\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} :$ Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}\}$.

Note: It is always the case that

$\mathrm{CSP}(\mathbf{B}) \subseteq \{\mathbf{A} :$ Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}\}$.

Consistency Properties

Remark:

- Many heuristic algorithms for CSP involve *constraint propagation*, i.e., deriving new constraints that are implied from the given constraints.

  (recall *unit propagation* for Horn SAT.)

- This is formalized using the notion of *consistency properties*, such as *node/arc/path consistency*, and, generally, *strong k-consistency*.

- Given a CSP-instance it is important to determine whether or not *strong k-consistency can be established* for it.

  Intutively, this means whether additional constraints can be added to the given instance, so that the new instance

  - has the same space of solutions;
  - every partial solution on fewer than k-variables can be extended to a solution on k-variables.

Consistency Properties & Pebble Games

Fact: Let $k \geq 2$ and let $\mathbf{A}$, $\mathbf{B}$ be two structures.

Then the following are equivalent:

- Strong k-consistency can be established for the CSP-instance consisting of $\mathbf{A}$ and $\mathbf{B}$.

- The Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}$.

Reason:

Strong k-consistency properties correspond to *winning strategies* for the Duplicator in the $(\exists, k)$-pebble game. ∎

From Non-Uniform CSP to Uniform CSP

Note: So far, we have focused on tractable cases of *non-uniform* constraint satisfaction problems.

$$\mathrm{CSP}(\mathbf{B}) = \mathrm{CSP}(All, \{\mathbf{B}\})$$

Fact: Many applications in AI and in database systems involve *uniform* CSP($\mathcal{A}, \mathcal{B}$), where $\mathcal{A}$ and $\mathcal{B}$ are classes of structures.

For example, usually, conjunctive query containment and conjunctive query evaluation appear as uniform CSPs.

Question:

Can we obtain tractability results for uniform CSP from tractability results for non-uniform CSP?

From Non-Uniform CSP to Uniform CSP

Note: Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of structures.

It is possible that CSP$(\mathcal{A}, \{\mathbf{B}\})$ is in P, for every $\mathbf{B}$ in $\mathcal{B}$, but CSP$(\mathcal{A}, \mathcal{B})$ is NP-complete.

Example: $\mathcal{K}$ = all *cliques* and $\mathcal{G}$ = all graphs:

- CSP$(\mathcal{K}, \{\mathbf{H}\})$ is in P, for every graph $\mathbf{H}$;
- CSP$(\mathcal{K}, \mathcal{G})$ in NP-complete.
  (it is the CLIQUE problem) ∎

Conclusion: Tractability results for non-uniform CSP do not necessarily yield tractability results for uniform CSP.

k-Datalog and CSP

However, certain non-uniform tractable cases *do* uniformize!

Definition: For every positive integer k, let

$D_k \quad = \quad \{\mathbf{B} : \overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog$\}$.

Remark: Recall that if $\mathbf{B} \in \mathcal{D}_k$, then CSP($\mathbf{B}$) is in P.

Theorem: K ... & Vardi – 1998

CSP($All, \mathcal{D}_k$) is in P, for every positive integer k.

Proof: An easy consequence of the following:

- Characterization of when $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog (using $(\exists, k)$-pebble games).
- Polynomial-time algorithm for determining the winner in the $(\exists, k)$-pebble game. ∎

Datalog and CSP

Conclusions:

- Expressibility in Datalog provides a unifying explanation for many tractability results about non-uniform CSP.

- Expressibility of non-uniform CSPs in k-Datalog uniformizes, which implies that $\mathrm{CSP}(All, \mathcal{D}_k)$ is a large island of tractability for uniform CSP.

However,

Open Problem: Let k be a positive integer.

Is there an algorithm to test, given a structure $\mathbf{B}$, whether or not $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog?

Bounded Treewidth: Motivation

Observation: Many algorithmic problems that are “hard” on arbitrary graphs become “easy” on trees.

Question: Can the concept of *tree* be weakened while maintaining “good” computational properties?

Bounded Treewidth: Motivation

Observation: Many algorithmic problems that are "hard" on arbitrary graphs become "easy" on trees.

Question: Can the concept of *tree* be weakened while maintaining "good" computational properties?

Answer: Robertson & Seymour – 1985–1995

Yes, the concept of *bounded treewidth* achieves this.

Note: Part of their work on *graph minors*.

(Downey & Fellows: "Parametrized Complexity")

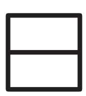
k-Trees, Partial k-Trees, Treewidth

Definition: Let k be a positive integer.

- A graph $\mathbf{H}$ is a *k-tree* if
  1. $\mathbf{H}$ is a $\mathbf{K}_{k+1}$-clique, or
  2. There are a k-tree $\mathbf{G}$, nodes $a_1, \ldots, a_k$ in $\mathbf{G}$ forming a $\mathbf{K}_k$-clique, and a node b in $\mathbf{H} - \mathbf{G}$ such that $\mathbf{H}$ is obtained from $\mathbf{G}$ by connecting b to each of the nodes $a_1, \ldots, a_k$ (thus, forming a $\mathbf{K}_{k+1}$-clique).
- A graph is a *partial k-tree* if it is a subgraph (not necessarily induced) of a k-tree.
- The *treewidth* of a graph $\mathbf{H}$, denoted by $\text{tw}(\mathbf{H})$, is the smallest k such that $\mathbf{H}$ is a partial k-tree.

Treewidth: Examples

Fact:

- If $\mathbf{T}$ is a tree, then $\mathrm{tw}(\mathbf{T}) = 1$.
- If $\mathbf{C}$ is a cycle, then $\mathrm{tw}(\mathbf{C}) = 2$.
- tw() $= 2$.
- tw() $= 2$.
- $\ldots$
- $\mathrm{tw}(\mathbf{K}_k) = k - 1$, for every $k \geq 2$.

Treewidth: Characterizations

Fact: The concept of treewidth can be characterized in two different ways:

- Using *tree decompositions* of graphs.
- Using *induced widths* of orderings of the nodes.

Note: All three descriptions of treewidth are useful, and are needed in different applications.

Treewidth: Complexity

Fact: The following problem is NP-complete:

Given graph $\mathbf{H}$ and integer $k \geq 1$, is $\mathrm{tw}(\mathbf{H}) \leq k$?

However,

Theorem: Bodlaender – 1993

For every integer $k \geq 1$, there is a linear-time algorithm such that, given a graph $\mathbf{H}$, it determines whether or not $\mathrm{tw}(\mathbf{H}) \leq k$.

Bounded Treewidth

Definition: For every integer $k \geq 1$,

$$\mathcal{T}(k) \;=\; \{\mathbf{H} : \mathrm{tw}(\mathbf{H}) < k\}.$$

Fact: Fix an integer $k \geq 2$.

Many problems that are NP-complete on arbitrary graphs become tractable on $\mathcal{T}(k)$.

3-Colorability, Monochromatic Triangle, ...

Treewidth & Monadic Second-Order Logic

Definition: Monadic Second-Order Logic

$$(\exists X_1 \forall X_2 \cdots \exists X_m)\psi,$$

where $X_1, \ldots, X_m$ are set variables and ψ is first-order.

Fact: For every $m \geq 1$, monadic second-order logic can express problems that are complete for the m-th level Σ_m^{P} of the polynomial hierarchy PH.

Theorem: Courcelle – 1990

Let Φ be a sentence of monadic second-order logic. For every integer $k \geq 2$, there is a polynomial-time algorithm such that, given a graph $\mathbf{H} \in \mathcal{T}(k)$, it determines whether or not $\mathbf{H} \models \Phi$.

CSP & Monadic Second-Order Logic

Fact: If $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ is a finite structure, then CSP($\mathbf{B}$) is definable by a sentence of existential monadic second-order logic with a universal first-order part, i.e., by a sentece of the form

$$(\exists X_1 \cdots \exists X_n)(\forall y_1 \cdots \forall y_s)\theta,$$

where θ is quantifier-free.

Proof: Use one X_i for each element of B. ∎

Example: 3-COLORABILITY

$$(\exists R \exists G \exists B)(\forall y_1 \forall y_2)\theta.$$

Corollary: (to Courcelle's Theorem)

If $k \geq 2$, then CSP($\mathcal{T}(k), \{\mathbf{B}\}$) is in P.

Uniform CSP and Bounded Treewidth

Note: The notion of *treewidth* can be defined for arbitrary finite structures $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$.

$$\mathrm{tw}(\mathbf{A}) \;=\; \mathrm{tw}(\mathbf{G}(\mathbf{A})),$$

where $\mathbf{G}(\mathbf{A}) = (A, E)$ is the *Gaifman* graph of $\mathbf{A}$, i.e.,

$E(a, a') \Longleftrightarrow a$, a' occur in a tuple of $R_i^{\mathbf{A}}$, for some i.

Theorem: Dechter & Pearl - 1989, Freuder - 1990

If $k \geq 2$, then $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ is in P.

Note: Thus, $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ is a large island of tractability for uniform CSP.

Moreover, Bodlaender's Theorem implies that it is an easily "accessible" island.

Logical Aspects of Bounded Treewidth

Fact: Fix an integer $k \geq 2$.

- The tractability of CSP$(\mathcal{T}(k), All)$ can be explained in terms of expressibility in k-Datalog.

- There are tight connections between having $\mathrm{tw}(\mathbf{A}) < k$ and the canonical query $Q^{\mathbf{A}}$ being definable in a fragment of first-order logic with k variables.

K . . . & Vardi – 2000

Dalmau, K . . ., Vardi – 2002

Finite-Variable Logics

Definition: Fix an integer $k \geq 2$.

- FO^k is the collection of all first-order formulas with k distinct variables.

- L^k is the collection of all FO^k-formulas built using atomic formulas, $\wedge$, and $\exists$ only.

Example: Let $\mathbf{C}_n$ be the n-element cycle, $n \geq 3$. The canonical conjunctive query $Q^{\mathbf{C}_n}$ is expressible in L^3.

For instance, $Q^{\mathbf{C}_4}$ is logically equivalent to

$$(\exists x \exists y \exists z)(E(x,y) \wedge E(y,z) \wedge (\exists y)(E(z,y) \wedge E(y,x))).$$

It is no accident that $\mathrm{tw}(\mathbf{C}_n) = 2$ ...

Bounded Treewidth & Finite-Variable Logics

Theorem: Fix an integer $k \geq 2$.

- If $\mathbf{A} \in \mathcal{T}(k)$, then the canonical conjunctive query $Q^{\mathbf{A}}$ is expressible in L^k.

- If $\mathbf{B}$ is a fixed structure, then $\overline{\mathrm{CSP}(\mathcal{T}(k), \{\mathbf{B}\})}$ is expressible in k-Datalog.

Proof:

- Use the construction of $\mathbf{A}$ as a partial k-tree.

- Combine the above with the characterization of expressibility of non-uniform CSP in k-Datalog in terms of the $(\exists, k)$-pebble game. ∎

Corollary: $\mathrm{CSP}(\mathcal{T}(k), \mathit{All})$ is in P.

Bounded Treewidth & Finite-Variable Logics

Definition: $\mathbf{A}$ and $\mathbf{B}$ are *homomorphically equivalent*, denoted $\mathbf{A} \sim_h \mathbf{B}$, if there are homomorphisms $h : \mathbf{A} \rightarrow \mathbf{B}$ and $h' : \mathbf{B} \rightarrow \mathbf{A}$.

Theorem: Fix a $k \geq 2$ and a finite structure $\mathbf{A}$.

Then the following are equivalent:

- $Q^{\mathbf{A}}$ is expressible in L^k.
- There is some $\mathbf{B} \in \mathcal{T}(k)$ such that $\mathbf{A} \sim_h \mathbf{B}$.
- $\text{core}(\mathbf{A}) \in \mathcal{T}(k)$.

Bounded Treewidth & Finite-Variable Logics

Definition: Fix an integer $k \geq 2$.

$\mathcal{H}(\mathcal{T}(k))$ is the class of all structures that are homomorphically equivalent to a structure in $\mathcal{T}(k)$.

Theorem: Fix an integer $k \geq 2$.

- CSP$(\mathcal{H}(\mathcal{T}(k)), \mathit{All})$ is in P.
- Determining membership in $\mathcal{H}(\mathcal{T}(k))$ is an NP-complete problem.

Conclusion:

- We have discovered a new island of tractability that is larger than $\mathcal{T}(k)$.
- Unfortunately, this island is "inaccessible".

Synopsis

Constraint Satisfaction is a meeting point of

- Computational Complexity
- Logic
- Database Theory
- Universal Algebra.

Solutions to Homework Assignment # 2

Exercise: Express the CLIQUE problem as a HOMOMORPHISM PROBLEM.

Given a graph $H = (V, E)$ and a positive integer $k \geq 2$, does H contain a k-clique?

Solution:

Fact: $\mathbf{H}$ contains a k-clique

$$\Longleftrightarrow$$

there is a homomorphism $h : \mathbf{K}_k \to \mathbf{H}$.

Solutions to Homework Assignment # 2

Exercise: Express the HAMILTONIAN CYCLE problem as a HOMOMORPHISM PROBLEM.

Given a graph $\mathbf{H} = (V, E)$, is there a cycle in which each node of $\mathbf{H}$ occurs exactly once?

Solution:

Given a graph $\mathbf{H} = (V, E)$ with $|V| = n$, let $\mathbf{C}_n = (\{1, \ldots, n\}, E_n)$ be a cycle of length n.

Fact:

$\mathbf{H}$ contains a Hamiltonian cycle

$$\Longleftrightarrow$$

there is a homomorphism

$h : (\{1, \ldots, n\}, E_n, \neq) \to (V, E, \neq)$.

Solutions to Homework Assignment #5

Exercise: Use Schaefer's Dichotomy Theorem to show that Monotone 3-SAT is NP-complete.

Solution: Monotone 3-Sat is $\mathrm{CSP}(\{0,1\}, R_0, R_3)$, where

$$R_0 = \{0,1\}^3 - \{(0,0,0\}) \quad \text{and} \quad R_1 = \{0,1\}^3 - \{(1,1,1\}).$$

- R_0 is not 0-valid, R_1 is not 1-valid.
- R_0 is neither bijunctive nor Horn, since

  $$\mathrm{maj}((1,0,0),(0,1,0),(0,0,1)) = (0,0,0) \notin R_0.$$

  $$(1,0,0) \wedge (0,1,0) = (0,0,0) \notin R_0.$$
- R_3 is neither dual Horn nor affince, since

  $$(1,1,0) \vee (1,0,1) = (1,1,1) \notin R_3$$

  $$(1,0,0) \oplus (0,1,0) \oplus (0,0,1) = (1,1,1) \notin R_3.$$

A Very Partial Bibliography

References

[Bod93] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 226–234, 1993.

[CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[DKV02] V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. of 8th International Conference on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, 2002 (to appear).

[Dec92] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, May 1992.

[Dec99] R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.

[DF99] R.G. Downey and M.R. Fellows. *Parametrized Complexity*. Springer-Verlag, 1999.

[DP89] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.

[Fre90] E.C Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proc. AAAI-90*, pages 4–9, 1990.

[FV98] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. on Computing*, 28:57–104, 1998. Preliminary version in *Proc. 25th ACM Symp. on Theory of Computing*, May 1993, pp. 612–622.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[HN90] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[HN92] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109:117–126, 1992.

[HNZ96] P. Hell, J. Nesetril, and X. Zhu. Complexity of Tree Homomorphism. *Discrete Appl. Math.*, 70:23–36, 1996.

[Hoo97] J.N. Hooker. Constraint satisfaction methods for generating valid cuts. In *Advances in computational and stochastic optimization, logic programming, and heuristic studies*, pages 1–30. Kluwer, 1997.

[Imm99] N. Immerman. *Descriptive Complexity.* Springer, 1999.

[Jea98] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1–2):185–204, 1998.

[KV95] Ph. G. Kolaitis and M. Y. Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, August 1995.

[KV00a] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, pages 302–332, 2000. Earlier version in: Proc. 17th ACM Symp. on Principles of Database Systems (PODS '98).

[KV00b] Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 175–181, 2000.

[PJ97] J. Pearson and P. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway University of London, 1997.

[Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 216–226, 1978.

[Ull89] J. D. Ullman. *Database and Knowledge-Base Systems, Volumes I and II.* Computer Science Press, 1989.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 137–146, 1982.