

# Logic and Constraint Satisfaction

## An Introduction

Phokion G. Kolaitis

IBM Almaden Research Center

American Institute of Mathematics

2008

# A Primer on Logic

- What is logic?

# A Primer on Logic

- What is logic?
- “Logic is logic. That’s all I say.”

*The Deacon’s Masterpiece*

Oliver Wendell Holmes, Sr., 1858

# Outline

- 1 Basic Notions
- 2 Conjunctive Queries & CSP
- 3 Datalog and CSP
- 4 Finite-variable Logics and CSP

# Vocabularies and Structures

## Definition

- *Vocabulary*  $\sigma$ : a set  $\sigma = \{R'_1, \dots, R'_m\}$  of relation symbols of specified arities.
- $\sigma$ -*structure*  $\mathbf{A} = (A, R_1, \dots, R_m)$ : a non-empty set  $A$  and relations on  $A$  such that  $\text{arity}(R_i) = \text{arity}(R'_i)$ ,  $1 \leq i \leq m$ .
- *Finite*  $\sigma$ -*structure*  $\mathbf{A}$ : universe  $A$  is finite

# Vocabularies and Structures

## Definition

- **Vocabulary**  $\sigma$ : a set  $\sigma = \{R'_1, \dots, R'_m\}$  of relation symbols of specified arities.
- $\sigma$ -**structure**  $\mathbf{A} = (A, R_1, \dots, R_m)$ : a non-empty set  $A$  and relations on  $A$  such that  $\text{arity}(R_i) = \text{arity}(R'_i)$ ,  $1 \leq i \leq m$ .
- **Finite  $\sigma$ -structure**  $\mathbf{A}$ : universe  $A$  is finite

## Example

- **Graph**:  $\mathbf{G} = (V, E)$ , where  $E$  is binary.
- **String**:  $\mathbf{S} = (\{1, 2, \dots, n\}, P)$ , where  $P$  is unary  
 $m \in P \iff$  the  $m$ -th bit of the string is 1.
  - String 10001 encoded as  $(\{1, 2, 3, 4, 5\}, \{1, 5\})$ .

# Vocabularies and Structures

## Example

Every 3-CNF formula can be viewed as a finite structure of the form  $\mathbf{A} = (A, R_0, R_1, R_2, R_3)$ , where each  $R_i$  is a 3-ary relation.

- 3-CNF formula  $\varphi$  with variables  $x_1, \dots, x_n$
- Structure  $\mathbf{A}^\varphi = (\{x_1, \dots, x_n\}, R_0^\varphi, R_1^\varphi, R_2^\varphi, R_3^\varphi)$ , where

$$R_0^\varphi = \{(x, y, z) : (x \vee y \vee z) \text{ is a clause of } \varphi\}$$

$$R_1^\varphi = \{(x, y, z) : (\neg x \vee y \vee z) \text{ is a clause of } \varphi\}$$

$$R_2^\varphi = \{(x, y, z) : (\neg x \vee \neg y \vee z) \text{ is a clause of } \varphi\}$$

$$R_3^\varphi = \{(x, y, z) : (\neg x \vee \neg y \vee \neg z) \text{ is a clause of } \varphi\}$$

# Queries

## Definition

- *Class  $\mathcal{C}$  of structures*: a collection of relational  $\sigma$ -structures closed under isomorphisms.
- *$k$ -ary Query  $Q$  on  $\mathcal{C}$* , where  $k \geq 1$ :  
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  $h : \mathbf{A} \rightarrow \mathbf{B}$  is an isomorphism, then  $Q(\mathbf{B}) = h(Q(\mathbf{A}))$ .
- *Boolean Query  $Q$  on  $\mathcal{C}$* :  
a mapping  $Q : \mathcal{C} \rightarrow \{0, 1\}$  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:** Binary query on graphs  $\mathbf{H} = (V, E)$

$$P2(\mathbf{H}) = \{(a, b) \in V^2 : \text{there is a path of length 2 from } a \text{ to } b\}.$$

- **CONNECTIVITY:** Boolean query on graphs  $\mathbf{H} = (V, E)$

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

- **$k$ -COLORABILITY**,  $k \geq 2$
- **3-SAT** (with formulas viewed as structures)

# Definability of Queries

## Definition

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  $\varphi(x_1, \dots, x_k)$  with  $x_1, \dots, x_k$  as free variables and such that for every  $\mathbf{A} \in \mathcal{C}$

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

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

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

# First-Order Logic

## Definition

First-Order Logic FO (on graphs):

- *first-order variables*:  $x, y, z, \dots$
- *atomic formulas*:  $E(x, y), x = y$
- *formulas*: atomic formulas, Boolean connectives, first-order quantifiers  $\exists x, \forall x, \exists y, \forall y, \dots$  that range over the nodes of the graph.

# First-Order Logic

## Definition

First-Order Logic FO (on graphs):

- *first-order variables*:  $x, y, z, \dots$
- *atomic formulas*:  $E(x, y), x = y$
- *formulas*: atomic formulas, Boolean connectives, first-order quantifiers  $\exists x, \forall x, \exists y, \forall y, \dots$  that range over the nodes of the graph.

## Example

On the class  $\mathcal{G}$  of finite graphs the query PATH OF LENGTH 2 is FO-definable

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

# Limitations of First-Order Logic

## Fact

- The queries TRANSITIVE CLOSURE, CONNECTIVITY,  $k$ -COLORABILITY,  $k \geq 2$ , are **not** FO-definable.
- On the class of all finite structures with 4 ternary relations, the query 3-SAT is **not** first-order definable.

**Note:** Results about non-definability in FO-logic can be proved using **Ehrenfeucht-Fraïssé games**.

# The Complexity of Logic

## Definition (Vardi, 1982)

- The *combined complexity* of  $L$  is the following decision problem:  
Given a finite structure  $\mathbf{B}$  and an  $L$ -sentence  $\psi$ , does  $\mathbf{B} \models \psi$ ?

# The Complexity of Logic

## Definition (Vardi, 1982)

- The *combined complexity* of  $L$  is the following decision problem:  
Given a finite structure  $\mathbf{B}$  and an  $L$ -sentence  $\psi$ , does  $\mathbf{B} \models \psi$ ?
- The *data complexity* of  $L$  is the family of the following decision problems  $P_\psi$ , one for each fixed  $L$ -sentence  $\psi$ :  
Given a finite structure  $\mathbf{B}$ , does  $\mathbf{B} \models \psi$ ?

# The Complexity of Logic

## Definition (Vardi, 1982)

- The *combined complexity* of  $L$  is the following decision problem:  
Given a finite structure  $\mathbf{B}$  and an  $L$ -sentence  $\psi$ , does  $\mathbf{B} \models \psi$ ?
- The *data complexity* of  $L$  is the family of the following decision problems  $P_\psi$ , one for each fixed  $L$ -sentence  $\psi$ :  
Given a finite structure  $\mathbf{B}$ , does  $\mathbf{B} \models \psi$ ?
- The *expression complexity* of  $L$  is the family of the following decision problems  $P_{\mathbf{B}}$ , one for each fixed finite structure  $\mathbf{B}$ :  
Given an  $L$ -sentence  $\psi$ , does  $\mathbf{B} \models \psi$ ?

# Some Basic Complexity Classes

## Definition

- L: problems solvable by a TM in logspace
- NL: problems solvable by a NTM in logspace
- P: problems solvable by a TM in polynomial time
- NP: problems solvable by a NTM in polynomial time
- PSPACE: problems solvable by a TM in polynomial space.

# Some Basic Complexity Classes

## Definition

- L: problems solvable by a TM in logspace
- NL: problems solvable by a NTM in logspace
- P: problems solvable by a TM in polynomial time
- NP: problems solvable by a NTM in polynomial time
- PSPACE: problems solvable by a TM in polynomial space.

## Fact

- $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSpace$ .
- $NL \neq PSPACE$
- **No** other proper containment is known at present.
- Each of them possesses natural **complete** problems.

# The Complexity of Logic

## Definition

Let  $L$  be a logic and  $C$  a complexity class.

- *The data complexity of  $L$  is in  $C$*  if for each  $L$ -sentence  $\psi$ , the problem  $P_\psi$  is in  $C$ .
- *The data complexity of  $L$  is  $C$ -complete* if it is in  $C$  and there is at least one  $L$ -sentence  $\psi$  such that  $P_\psi$  is  $C$ -complete.

# The Complexity of Logic

## Definition

Let  $L$  be a logic and  $C$  a complexity class.

- *The data complexity of  $L$  is in  $C$*  if for each  $L$ -sentence  $\psi$ , the problem  $P_\psi$  is in  $C$ .
- *The data complexity of  $L$  is  $C$ -complete* if it is in  $C$  and there is at least one  $L$ -sentence  $\psi$  such that  $P_\psi$  is  $C$ -complete.
- *The expression complexity of  $L$  is in  $C$*  if for each finite structure  $\mathbf{B}$ , the problem  $P_{\mathbf{B}}$  is in  $C$ .
- *The expression complexity of  $L$  is  $C$ -complete* if it is in  $C$  and there is at least one finite structure  $\mathbf{B}$  such that  $P_{\mathbf{B}}$  is  $C$ -complete.

# The Complexity of First-Order Logic

## Theorem

- The data complexity of FO is in L.
- Both the expression complexity of FO and the combined complexity of FO are PSPACE-complete.

## Proof.

- Fix a first-order sentence  $\psi$ . Given a finite structure **B**:  
Cycle through all possible instantiations of the quantifiers of  $\psi$  in **B**, keeping track of the number of them using a counter in binary.
- QBF is PSPACE-complete (Stockmeyer - 1976).



# Outline

- 1 Basic Notions
- 2 Conjunctive Queries & CSP**
- 3 Datalog and CSP
- 4 Finite-variable Logics and CSP

# Conjunctive Queries

## Definition

- A *primitive positive formula (pp-formula)* is a FO-formula in prenex normal form built from atomic formulas,  $\wedge$ , and  $\exists$  only, i.e., it is of the form:

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

where  $\psi$  is a conjunction of atomic formulas.

- A *conjunctive query* is a query definable by a pp-formula.

# Conjunctive Queries

## Definition

- A *primitive positive formula (pp-formula)* is a FO-formula in prenex normal form built from atomic formulas,  $\wedge$ , and  $\exists$  only, i.e., it is of the form:

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

where  $\psi$  is a conjunction of atomic formulas.

- A *conjunctive query* is a query definable by a pp-formula.

## Example

- PATH OF LENGTH 2 (Binary query)

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

Can also be written as a rule:

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

# Conjunctive Queries and Databases

## Fact

- Conjunctive queries are the most frequently asked queries in databases (a.k.a. SPJ queries)
- The main construct of SQL expresses conjunctive queries.

## Example

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

```
SELECT     $R_1.A, R_2.D$ 
FROM       $R_1, R_2$ 
WHERE      $R_1.B = R_2.B$  AND  $R_1.C = R_2.C$ 
```

This expresses the conjunctive query  $Q(x, w)$  definable by

$$\exists y \exists z (R_1(x, y, z) \wedge R_2(y, z, w))$$

# Conjunctive Query Evaluation

A fundamental problem about conjunctive queries

## Definition

### CONJUNCTIVE QUERY EVALUATION

- Given a CQ  $Q$  and a structure  $\mathbf{A}$ , find
$$Q(\mathbf{A}) = \{(a_1, \dots, a_k) : \mathbf{A} \models Q(a_1, \dots, a_k)\}$$
- For Boolean queries  $Q$ , this becomes:  
Given  $Q$  and  $\mathbf{A}$ , does  $\mathbf{A} \models Q$ ? (is  $Q(\mathbf{A}) = 1$ ?)
- Same problem as the *combined complexity of pp-formulas*.

## Example

- Given a graph  $\mathbf{H}$ , does it contain a triangle?

# Conjunctive Query Containment

Another fundamental problem about conjunctive queries

## Definition

### CONJUNCTIVE QUERY CONTAINMENT

- Given two  $k$ -ary CQs  $Q_1$  and  $Q_2$ , is it true that for every structure  $\mathbf{A}$ , we have that  $Q_1(\mathbf{A}) \subseteq Q_2(\mathbf{A})$ ?
- 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**  $Q_2$ ?)

## Example

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?

# Conjunctive Queries and Homomorphisms

- Chandra and Merlin (1977) showed that:

CONJUNCTIVE QUERY EVALUATION

and

CONJUNCTIVE QUERY CONTAINMENT

are the *same* problem.

# Conjunctive Queries and Homomorphisms

- Chandra and Merlin (1977) showed that:

CONJUNCTIVE QUERY EVALUATION

and

CONJUNCTIVE QUERY CONTAINMENT

are the *same* problem.

- The link is the HOMOMORPHISM PROBLEM:

Given **A** and **B**, does  $\mathbf{A} \rightarrow \mathbf{B}$ ?

# Canonical CQs and Canonical Structures

## Definition

- Given  $\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ , 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:  $\exists x_1 \cdots \exists x_n (\bigwedge_{i=1}^m \bigwedge_{\mathbf{t}} R_i^{\mathbf{A}}(\mathbf{t}))$
- If  $Q$  is a Boolean CQ, then  $\mathbf{A}^Q$  is the structure with the variables of  $Q$  as elements and the conjuncts of  $Q$  as “facts”.

## Example

- If  $\mathbf{A} = (\{a, b, c\}, \{(a, b), (b, c), (c, a)\})$ , then  $Q^{\mathbf{A}}$  is  

$$\exists x \exists y \exists z (E(x, y) \wedge E(y, z) \wedge E(z, x))$$
- If  $Q$  is  $\exists x \exists y (E(x, y) \wedge E(x, z))$ , then  

$$\mathbf{A}^Q = (\{x, y, z\}, \{(x, y), (x, z)\})$$

# Homomorphisms, CQE and CQC

## Theorem (Chandra and Merlin, 1977)

For all relational structures **A** and **B**, the following statements are equivalent:

- 1  $\mathbf{A} \rightarrow \mathbf{B}$
- 2  $\mathbf{B} \models Q^{\mathbf{A}}$
- 3  $Q^{\mathbf{B}} \subseteq Q^{\mathbf{A}}$ .

# Homomorphisms, CQE and CQC

Alternatively,

## Theorem (Chandra and Merlin, 1977)

For all conjunctive queries  $Q_1$  and  $Q_2$ , the following statements are equivalent:

- 1  $Q_1 \subseteq Q_2$
- 2  $\mathbf{A}^{Q_2} \rightarrow \mathbf{A}^{Q_1}$
- 3  $\mathbf{A}^{Q_1} \models Q_2$ .

# Illustration: 3-COLORABILITY

## Example

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

- 1  $\mathbf{H} \rightarrow \mathbf{K}_3$  (i.e.,  $\mathbf{H}$  is 3-colorable)
- 2  $\mathbf{K}_3 \models Q^{\mathbf{H}}$
- 3  $Q^{\mathbf{K}_3} \subseteq Q^{\mathbf{H}}$

(1)  $\implies$  (2): A hom.  $h : \mathbf{H} \rightarrow \mathbf{K}_3$  provides witnesses in  $\mathbf{K}_3$  for the  $\exists$  quantifiers in  $Q^{\mathbf{H}}$ .

(2)  $\implies$  (3): If  $\mathbf{K}_3 \models Q^{\mathbf{H}}$  and  $\mathbf{A} \models Q^{\mathbf{K}_3}$ , then there are witness functions  $h : \mathbf{H} \rightarrow \mathbf{K}_3$  and  $h^* : \mathbf{K}_3 \rightarrow \mathbf{A}$ . Then  $h^* \circ h : \mathbf{H} \rightarrow \mathbf{A}$  provides witnesses in  $\mathbf{A}$  for the  $\exists$  quantifiers in  $Q^{\mathbf{H}}$ .

(3)  $\implies$  (1): Since  $\mathbf{K}_3 \models Q^{\mathbf{K}_3}$ , we have  $\mathbf{K}_3 \models Q^{\mathbf{H}}$ . The witnesses to the  $\exists$  quantifiers give a homomorphism from  $\mathbf{H}$  to  $\mathbf{K}_3$ .

# CSP, Homomorphisms, CQE, and CQC

## Fact

- CONSTRAINT SATISFACTION
- THE HOMOMORPHISM PROBLEM
- CONJUNCTIVE QUERY EVALUATION
- CONJUNCTIVE QUERY CONTAINMENT

are the **same** problem.

# CSP, Homomorphisms, CQE, and CQC

## Fact

- The combined complexity of conjunctive queries (pp-formulas) coincides with the HOMOMORPHISM PROBLEM (UNIFORM CSP).
- The expression complexity of conjunctive queries (pp-formulas) coincides with the family of problems  $\text{CSP}(\mathbf{B})$ , where

$$\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \rightarrow \mathbf{B}\} = \{\mathbf{A} : \mathbf{B} \models Q^{\mathbf{A}}\}.$$

- Both the combined complexity and the expression complexity of conjunctive queries are NP-complete. (contrast with FO.)

# Tractability of CSP via Logic

## Fact

- The complexity of  $\text{CSP}(\mathbf{B})$  depends on  $\mathbf{B}$ :
  - $\text{CSP}(\mathbf{K}_3)$  is 3-COLORABILITY, hence is NP-complete.
  - $\text{CSP}(\mathbf{K}_2)$  is 2-COLORABILITY, hence is in P.

## Approach

- Use logic to identify tractable (polynomial-time solvable) cases of  $\text{CSP}(\mathbf{B})$ .
- Study when  $\text{CSP}(\mathbf{B})$  is definable in some logic  $L$  whose data complexity is in P.

# CSP and Unions of Conjunctive Queries

## Definition

For every structure  $\mathbf{B}$ , let

$$\neg\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \not\models \mathbf{B}\}.$$

## Fact

For every structure  $\mathbf{B}$ :

- $\neg\text{CSP}(\mathbf{B})$  is closed under homomorphisms.
- Moreover,

$$\neg\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \models \bigvee_{\mathbf{D} \not\models \mathbf{B}} Q^{\mathbf{D}}\},$$

i.e.,  $\neg\text{CSP}(\mathbf{B})$  is definable by an **infinite** union of conjunctive queries.

# CSP and Unions of Conjunctive Queries

## Definition

- $L_{\infty\omega}$  is the extension of FO with infinitary disjunctions and infinitary conjunctions.
- $\exists L_{\infty\omega}^+$  is the **existential positive** fragment of  $L_{\infty\omega}$ .

## Approach

- Thus, for every structure  $\mathbf{B}$ , we have that  $\neg\text{CSP}(\mathbf{B})$  is  $\exists L_{\infty\omega}^+$ -definable, since

$$\neg\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \models \bigvee_{\mathbf{D} \not\models \mathbf{B}} Q^{\mathbf{D}}\}.$$

- Study when  $\neg\text{CSP}(\mathbf{B})$  is definable in a tractable fragment of  $\exists L_{\infty\omega}^+$ .

# CSP and First-Order Logic

## Fact

Assume that  $\mathbf{B}$  is a structure such that  $\neg\text{CSP}(\mathbf{B})$  is definable by a **finite** union of conjunctive queries (i.e.,  $\neg\text{CSP}(\mathbf{B}) = \bigvee_{i=1}^m Q^{\mathbf{D}_i}$ ). Then  $\text{CSP}(\mathbf{B})$  is FO-definable; hence, it is in P.

# CSP and First-Order Logic

## Fact

Assume that  $\mathbf{B}$  is a structure such that  $\neg\text{CSP}(\mathbf{B})$  is definable by a **finite** union of conjunctive queries (i.e.,  $\neg\text{CSP}(\mathbf{B}) = \bigvee_{i=1}^m Q^{\mathbf{D}_i}$ ). Then  $\text{CSP}(\mathbf{B})$  is FO-definable; hence, it is in P.

## Theorem (Atserias, 2005)

For every structure  $\mathbf{B}$ , the following statements are equivalent.

- 1  $\text{CSP}(\mathbf{B})$  is FO-definable.
- 2  $\neg\text{CSP}(\mathbf{B})$  is definable by a **finite** union of conjunctive queries.

# CSP and First-Order Logic

## Example (Gallai-Hesse-Roy Theorem, circa 1965)

Let  $\mathbf{T}_k$  be the linear order with  $k$  elements and  $\mathbf{P}_{k+1}$  be the directed path with  $k + 1$  elements. Then, for every directed graph  $\mathbf{G}$ , we have that

$$\mathbf{G} \rightarrow \mathbf{T}_k \iff \mathbf{P}_{k+1} \not\rightarrow \mathbf{G}.$$

Consequently,

$$\neg \text{CSP}(\mathbf{T}_k) = \{\mathbf{G} : \mathbf{G} \models Q^{\mathbf{P}_{k+1}}\}.$$

# Beyond First-Order Logic

## Fact

- $\text{CSP}(\mathbf{K}_2)$  is 2-COLORABILITY.
- $\text{CSP}(\mathbf{K}_2)$  is in P, but it is **not** FO-definable.
- Hence,  $\neg\text{CSP}(\mathbf{K}_2)$  is definable by an infinite union of conjunctive queries, but it is **not** definable by any finite union of conjunctive queries.

## Question

Can the tractability of  $\text{CSP}(\mathbf{K}_2)$  be explained via definability in a logic other than FO?

# Outline

- 1 Basic Notions
- 2 Conjunctive Queries & CSP
- 3 Datalog and CSP**
- 4 Finite-variable Logics and CSP

# Tractability via Definability in Datalog

Fact (Feder and Vardi, 1993)

Definability of  $\neg\text{CSP}(\mathbf{B})$  in **Datalog** is a unifying explanation for many tractability results about  $\text{CSP}(\mathbf{B})$ , including  $\text{CSP}(\mathbf{K}_2)$ .

# Datalog

**Note:** Recall that every CQ can be written as a **rule**:

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

## Definition

- Datalog = Conjunctive Queries + Recursion  
Function, negation-free, and  $\neq$ -free logic programs
- A **Datalog program** is a finite set of rules given by conjunctive queries

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

**Intensional DB predicates (IDBs):** those that occur both in the *heads* and the *bodies* of rules (**recursive** predicates).

**Extensional DB predicates (EDBs):** all other predicates.

## Example (TRANSITIVE CLOSURE Query $TC$ )

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

A Datalog program for  $TC$  (**linear** Datalog)

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

Another Datalog program for  $TC$  (**non-linear** Datalog)

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

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

# Semantics of Datalog Programs

## Example

A Datalog program for  $TC$

$$\begin{array}{l|l} S(x,y) & :- E(x,y) \\ S(x,y) & :- E(x,z), S(z,y) \end{array}$$

Operational Semantics: "Bottom-up" Evaluation

$$\begin{array}{l|l} S^0 & = \emptyset \\ S^{m+1} & = \{(a,b) : \exists z (E(a,z) \wedge S^m(z,b))\} \end{array}$$

**Fact:** The following statements are true:

$$\begin{aligned} S^m &= \{(a,b) : \text{there is a path of length } \leq m \text{ from } a \text{ to } b\} \\ TC &= \bigcup_m S^m = S^{|V|}. \end{aligned}$$

# Datalog and 2-Colorability

## Example

- $\text{CSP}(\mathbf{K}_2) = \text{2-COLORABILITY}$ .
- Recall that a graph is 2-colorable if and only if it does not contain an odd cycle.
- Datalog program for NON-2-COLORABILITY:

|           |   |     |                             |
|-----------|---|-----|-----------------------------|
| $O(X, Y)$ | : | $-$ | $E(X, Y)$                   |
| $O(X, Y)$ | : | $-$ | $O(X, Z), E(Z, W), E(W, Y)$ |
| $Q$       | : | $-$ | $O(X, X)$                   |

# Data Complexity of Datalog

## Theorem

- Every Datalog query is definable by an “effective and uniform” union of conjunctive queries.
- Every Datalog query is in P.
- The data complexity of Datalog is P-complete.

## Proof.

- The “bottom-up” evaluation of Datalog programs converges in polynomially-many steps.
- Each iteration is definable by a finite union of conjunctive queries.
- HORN 3-UNSAT is P-complete and expressible in Datalog.



# Horn 3-SAT and Datalog

## Fact (HORN 3-UNSAT is expressible in Datalog)

- Horn 3-CNF formula  $\varphi$  viewed as a finite structure

$$\mathbf{A}^\varphi = (\{x_1, \dots, x_n\}, U, P, N), \text{ where}$$

- $U$  is the set of unit clauses  $x$
- $P$  is the set of clauses  $(\neg x \vee \neg y \vee z)$
- $N$  is the set of clauses  $(\neg x \vee \neg y \vee \neg z)$ .
- Datalog program for HORN 3-UNSAT: encodes the **unit propagation algorithm** for Horn Satisfiability.

$$\begin{array}{l|l} T(z) & :- U(z) \\ T(z) & :- P(x, y, z), T(x), T(y) \\ Q & :- N(x, y, z), T(x), T(y), T(z) \end{array}$$

Provably **non**-linearizable.

# Tractability via Definability in Datalog

## Fact (Feder and Vardi, 1993)

Definability of  $\neg\text{CSP}(\mathbf{B})$  in Datalog is a unifying explanation for many tractability results about  $\text{CSP}(\mathbf{B})$ .

## Theorem (Feder and Vardi, 1993)

- If  $\mathbf{B} = (B, R_1, \dots, R_k)$  is such that  $\text{Pol}(\{R_1, \dots, R_k\})$  contains a **near-unanimity** function, then  $\neg\text{CSP}(\mathbf{B})$  is Datalog-definable.

**Special Case:** 2-SAT

- If  $\mathbf{B} = (B, R_1, \dots, R_k)$  is such that  $\text{Pol}(\{R_1, \dots, R_k\})$  contains a **semi-lattice** function, then  $\neg\text{CSP}(\mathbf{B})$  is Datalog-definable.

**Special Case:** HORN  $k$ -SAT, DUAL HORN  $k$ -SAT,  $k \geq 2$ .

# Outline

- 1 Basic Notions
- 2 Conjunctive Queries & CSP
- 3 Datalog and CSP
- 4 Finite-variable Logics and CSP**

# Logics with Finitely Many Variables

An old, but fruitful, idea: the **number of variables** is a resource.

## Definition

- $\text{FO}^k$ : FO-formulas with at most  $k$  variables.
- $\text{L}^k$ :  $\text{FO}^k$ -formulas built from atomic formulas,  $\wedge$ , and  $\exists$  only.
- **Note**: Each  $\text{L}^k$ -formula defines a conjunctive query.

# Logics with Finitely Many Variables

An old, but fruitful, idea: the **number of variables** is a resource.

## Definition

- $\text{FO}^k$ : FO-formulas with at most  $k$  variables.
- $\text{L}^k$ :  $\text{FO}^k$ -formulas built from atomic formulas,  $\wedge$ , and  $\exists$  only.
- **Note**: Each  $\text{L}^k$ -formula defines a conjunctive query.

## Example

- $P^n(x, y)$ : there is a path of length  $n$  from  $x$  to  $y$ .

# Logics with Finitely Many Variables

An old, but fruitful, idea: the **number of variables** is a resource.

## Definition

- $\text{FO}^k$ : FO-formulas with at most  $k$  variables.
- $\text{L}^k$ :  $\text{FO}^k$ -formulas built from atomic formulas,  $\wedge$ , and  $\exists$  only.
- **Note**: Each  $\text{L}^k$ -formula defines a conjunctive query.

## Example

- $P^n(x, y)$ : there is a path of length  $n$  from  $x$  to  $y$ .
- $P^n(x, y)$  is  $\text{L}^3$ -definable.

$$\begin{aligned}P^1(x, y) &\equiv E(x, y) \\ P^{n+1}(x, y) &\equiv \exists z (E(x, z) \wedge \exists x ((x = z) \wedge P_n(x, y))).\end{aligned}$$

# $k$ -Datalog

## Definition

A  $k$ -Datalog program is a Datalog program in which each rule  $t_0 : - t_1, \dots, t_m$  has at most  $k$  distinct variables.

## Example

- NON 2-COLORABILITY revisited

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

- Therefore, NON 2-COLORABILITY is definable in 4-Datalog.
- **Exercise:** NON 2-COLORABILITY is definable in 3-Datalog.

# Datalog and Finite-Variable Logics

## Theorem (K ... and Vardi, 1990)

- Every  $k$ -Datalog definable query is also definable by a formula of the form  $\bigvee_{n \geq 1} \psi_n$ , where  $\psi_n$  is an  $L^k$ -formula.
- Consequently,  $k$ -Datalog  $\subseteq \exists L_{\infty\omega}^{k,+}$ .

## Note

In general,  $k$ -Datalog is a **proper** fragment of  $\exists L_{\infty\omega}^{k,+}$ .  
(The latter can express non-recursive queries using arbitrary infinite disjunctions.)

# Datalog, Finite-Variable Logics, and CSP

## Theorem (K ... and Vardi, 1998)

For every  $\mathbf{B}$  and every  $k \geq 1$ , the following are equivalent:

- 1  $\neg\text{CSP}(\mathbf{B})$  is definable in  $k$ -Datalog.
- 2  $\neg\text{CSP}(\mathbf{B})$  is definable by a formula of the form  $\bigvee_{n \geq 1} \psi_n$ , where each  $\psi_n$  is an  $L^k$ -formula.
- 3  $\neg\text{CSP}(\mathbf{B})$  is definable in  $\exists L_{\infty\omega}^{k,+}$ .

## Note

Recall that

$$\neg\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \models \bigvee_{\mathbf{D} \not\models \mathbf{B}} Q^{\mathbf{D}}\}$$

and each  $Q^{\mathbf{D}}$  is a conjunctive query.

# CSP and Logic

## Summary

For every structure  $\mathbf{B}$  and for every  $k \geq 1$ :

- $\neg\text{CSP}(\mathbf{B})$  is definable by an (infinite) union of conjunctive queries.
- $\neg\text{CSP}(\mathbf{B})$  is FO-definable if and only if it is definable by a finite union of conjunctive queries.
- $\neg\text{CSP}(\mathbf{B})$  is definable in  $k$ -Datalog if and only if it is definable by an (infinite) union of conjunctive queries each of which is  $L^k$ -definable.

# Existential $k$ -Pebble Games

**Spoiler** and **Duplicator** play on two structures **A** and **B**. Each player uses  $k$  pebbles, labeled  $1, \dots, k$ . In each move,

- **Spoiler** places a pebble on or removes a pebble from an element of **A**.
- **Duplicator** tries to duplicate the move on **B** using the pebble with the same label.

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

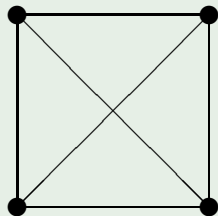
- **Spoiler** wins the  $(\exists, k)$ -pebble game if at some point the mapping  $a_i \mapsto b_i$ ,  $1 \leq i \leq l$ , is **not** a partial homomorphism.
- **Duplicator** wins the  $(\exists, k)$ -pebble game if the above never happens.

## Fact (Cliques of Different Size)

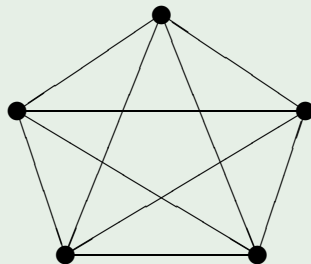
Let  $K_k$  be the  $k$ -clique. Then

- **Duplicator** wins the  $(\exists, k)$ -pebble game on  $K_k$  and  $K_{k+1}$ .
- **Spoiler** wins the  $(\exists, k)$ -pebble game on  $K_k$  and  $K_{k-1}$ .

## Example



$K_4$



$K_5$

# Winning Strategies in the $(\exists, k)$ -Pebble Game

## Definition

A *winning strategy* for the Duplicator in the  $(\exists, k)$ -pebble game is a non-empty set  $\mathcal{I}$  of partial homomorphisms from  $\mathbf{A}$  to  $\mathbf{B}$  such that

- If  $f \in \mathcal{I}$  and  $h \subseteq f$ , then  $h \in \mathcal{I}$   
( $\mathcal{I}$  is *closed under subfunctions*).
- If  $f \in \mathcal{I}$  and  $|f| < k$ , then for every  $a \in A$ , there is  $g \in \mathcal{I}$  so that  $f \subseteq g$  and  $a \in \text{dom}(g)$ .  
( $\mathcal{I}$  has the *forth property up to k*)

## Fact

If  $\mathbf{A} \rightarrow \mathbf{B}$ , then, for every  $k \geq 1$ , the Duplicator wins the  $(\exists, k)$ -pebble game on  $\mathbf{A}$  and  $\mathbf{B}$ .

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

## Theorem (K ... and Vardi)

- Let  $Q$  be a query definable in  $\exists L_{\infty\omega}^{k,+}$ . If  $\mathbf{A}$  satisfies  $Q$  and the **Duplicator** wins the  $(\exists, k)$ -pebble game on  $\mathbf{A}$  and  $\mathbf{B}$ , then also  $\mathbf{B}$  satisfies  $Q$ .
- There is a polynomial-time algorithm to decide whether, given two finite structures  $\mathbf{A}$  and  $\mathbf{B}$ , the **Spoiler** or the **Duplicator** wins the  $(\exists, k)$ -pebble game on  $\mathbf{A}$  and  $\mathbf{B}$ .
- For every fixed finite structure  $\mathbf{B}$ , there is a  $k$ -Datalog program that expresses the query: given a finite structure  $\mathbf{A}$ , does the **Spoiler** win the  $(\exists, k)$ -game on  $\mathbf{A}$  and  $\mathbf{B}$ ?

# $k$ -Datalog, $\exists L_{\infty\omega}^{k,+}$ , $(\exists, k)$ -pebble games, and CSP

## Theorem

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

- 1  $\neg\text{CSP}(\mathbf{B})$  is definable in  $k$ -Datalog.
- 2  $\neg\text{CSP}(\mathbf{B})$  is definable in  $\exists L_{\infty\omega}^{k,+}$ .
- 3  $\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \text{Duplicator wins the } (\exists, k)\text{-pebble game on } \mathbf{A} \text{ and } \mathbf{B}\}.$

## Note

Single *canonical* polynomial-time algorithm for all  $\text{CSP}(\mathbf{B})$ 's that are definable in  $k$ -Datalog:

Determine the winner in the  $(\exists, k)$ -pebble game.

# The Hierarchy Problem for Datalog-definable CSPs

## Problem

Prove or disprove:

For every  $k \geq 4$ , there is a directed graph  $\mathbf{G}_k$  such that

- $\neg\text{CSP}(\mathbf{G}_k)$  is definable in  $k$ -Datalog;
- $\neg\text{CSP}(\mathbf{G}_k)$  is **not** definable in  $(k - 1)$ -Datalog.

## Note

- NON 2-COLORABILITY is definable in 3-Datalog, but not in 2-Datalog.
- All  $\neg\text{CSP}(\mathbf{G})$ 's presently known to be definable in Datalog are actually definable in 3-Datalog.

# The Meta-problem for Datalog-definable CSPs

## Problem

Determine whether or not the following problems are decidable:

- Given a structure  $\mathbf{B}$ , is  $\neg\text{CSP}(\mathbf{B})$  definable in Datalog?
- Given a structure  $\mathbf{B}$ , is  $\neg\text{CSP}(\mathbf{B})$  definable in  $k$ -Datalog?  
(Here  $k$  is a fixed positive integer.)

## Theorem (Larose, Loten, and Tardif, 2006)

The following problem is NP-complete:

Given a structure  $\mathbf{B}$ , is  $\text{CSP}(\mathbf{B})$  definable in first-order logic?

# Epilogue

- “Logic is in the eye of the logician.”

*Outrageous Acts and Everyday Rebellions*

Gloria Steinem, 1986.