

Finite Model Theory

Qing Wang

qing.wang@anu.edu.au

Logic and Computation
Research School of Computer Science

Complexity Theory

- Classify computational problems into different classes according to their inherent difficulty, and studying relationship among those classes.
- **Complexity class**: a set of languages determined by an abstract machine M using $O(f(n))$ of resource R , where n is the size of the input.
 - **An abstract machine** – a deterministic Turing machine, etc.
 - **Resource** – time, space, communications, processors, etc.
 - **Resource bound** – a function to bound the amount of resource.

$$\text{Ptime} \subseteq \begin{cases} \text{NP} \\ \text{coNP} \end{cases} \subseteq \text{Pspace}$$

Overview of Topics

• Part 3: – Logic and Complexity

- 1 Basics in complexity theory
- 2 Descriptive complexity
- 3 Some logics
 - Second-order logic ($\forall\text{SO}$, $\exists\text{SO}$)
 - Fixed Point Logics (LFP, IFP, PFP)
- 4 Logical characterizations
 - NP and coNP
 - Ptime + >
 - Pspace + >
- 5 Open problem: a logic for Ptime?

Time Complexity

- Let M be a Turing machine (TM) that halts on all inputs.
- The **time complexity** of M is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the **maximum number of steps** that M uses on an input of length n .
 - A language L is in $\text{DTIME}(f(n))$ if L is accepted by a deterministic M and the running time of M is $O(f(n))$.
 - A language L is in $\text{NTIME}(f(n))$ if L is accepted by a nondeterministic M and the running time of M is $O(f(n))$.

Complexity Class – P

- **Ptime (P)** is the class of languages decidable by a **deterministic TM** in **polynomial time**:

$$P = \bigcup_{k \in \mathbb{N}} DTIME(n^k).$$

- P plays a central role in complexity theory:
 - 1 mathematically robust class – invariant for models of computation;
 - 2 practically important class – realistically solvable on a computer.

Space Complexity

- Let M be a deterministic TM that halts on all inputs.
 - The **space complexity** of M is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is **the maximum number of tape cells** that M uses on inputs of length n .
 - A language L is in $DSPACE(f(n))$ if L is decided by M and the running space of M is $O(f(n))$.

Complexity Class – NP

- **NP** is the class of languages accepted by a **nondeterministic TM** in **polynomial time**:

$$NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k).$$

- **coNP** is the class of languages whose **complements** are in NP (i.e., “no”-instances can be accepted in polynomial time by a non-deterministic TM).

$$P \subseteq NP \cap \text{coNP}$$

It is not known:

- whether the containment is proper, and
- whether NP equals coNP.

Complexity Class – Pspace

- **Pspace** is the class of languages decidable in polynomial-space on a deterministic TM:

$$Pspace = \bigcup_{k \in \mathbb{N}} DSPACE(n^k).$$

- By Savitch's theorem, the nondeterministic case collapses to the deterministic one, in the case of space complexity.

P versus NP

$$P \stackrel{?}{=} NP$$

- P = the class of languages for which membership can be **decided** quickly.
- NP = the class of languages for which membership can be **verified** quickly.

Logic and Complexity

- Can computational complexity theory be helpful to the logician? Can logic be used as a tool to obtain pure complexity-theoretic results?
- **Logical definability**: logical resources needed to express queries \rightsquigarrow lead to different logics.
 - 1 number of variables
 - 2 type of quantifiers
 - 3 restriction on negation
 - 4 ...
- **Computational complexity**: computational resources needed to compute queries \rightsquigarrow lead to different complexity classes.
 - 1 time
 - 2 space
 - 3 ...

Descriptive Complexity

- Studies the connections between **computational complexity** and **logical definability**.
- By saying that **a logic L captures a complexity class C**, it means that
 - 1 Every query that can be evaluated in C over finite structures is definable in L
 - 2 Every query definable in L can be evaluated in C.
- Provides **machine-independent characterization** of complexity classes, without using any notion of machine, computation, or time.
- Allows us to convert a problem in complexity theory into an equivalent problem in logic, and vice versa.

Descriptive Complexity – Main Results

- Many computational complexity classes can be characterized in terms of logical definability on classes of finite structures.
 - 1 Fagin, 1974
NP \equiv \exists SO on all finite structures
 - 2 Immerman and Vardi, 1982
Ptime \equiv LFP on all ordered finite structures
 - 3 Abiteboul and Vianu, 1982
Pspace \equiv PFP on all ordered finite structures
 - 4 ...
- Generally, there are two cases so far:
 - A complexity class (NP or higher) is characterized by a logic over all finite structures.
 - A complexity class (P or lower) is characterized by a logic over all ordered finite structures.

Second-Order Logic

- **Second-order logic** (SO) extends FO logic by allowing second-order quantifiers:

$$\exists X\varphi \text{ and } \forall X\varphi,$$

where X is a second-order variable, ranging over relations on the universe.

Let \mathfrak{A} be a structure.

- \mathfrak{A} satisfies $\exists X\varphi$ if, for *some* n -ary relation R on the universe of \mathfrak{A} , $\mathfrak{A} \models \varphi(X/R)$.
- \mathfrak{A} satisfies $\forall X\varphi$ if, for *all* n -ary relations R on the universe of \mathfrak{A} , $\mathfrak{A} \models \varphi(X/R)$.

Let $\varphi(X_1, \dots, X_k)$ be a FO formula.

- **Existential second-order logic** (\exists SO) has the form:

$$\exists X_1 \dots \exists X_n \varphi(X_1, \dots, X_n);$$
- **Universal second-order logic** (\forall SO) has the form:

$$\forall X_1 \dots \forall X_n \varphi(X_1, \dots, X_n),$$

Second-Order Logic - Examples

- **Colourability**: Color the vertices of a graph such that no two vertices sharing the same edge have the same color.

- A graph (V, E) is **2-colourable** iff the following formula is true.

$$\exists R \forall x \forall y (E(x, y) \Rightarrow (R(x) \Leftrightarrow \neg R(y)))$$

- A graph (V, E) is **3-colourable** iff the following formula is true.

$$\begin{aligned} \exists R \exists B \exists G \quad & \forall x (R(x) \vee B(x) \vee G(x)) \wedge \\ & \forall x (\neg(R(x) \wedge B(x)) \wedge \neg(B(x) \wedge G(x)) \wedge \neg(R(x) \wedge G(x))) \wedge \\ & \forall x \forall y (E(x, y) \Rightarrow (\neg(R(x) \wedge R(y)) \wedge \\ & \quad \neg(B(x) \wedge B(y)) \wedge \\ & \quad \neg(G(x) \wedge G(y)))) \end{aligned}$$

Second-Order Logic - Examples

- **Hamiltonian graph**: The following formula can test if a graph contains a Hamiltonian cycle.

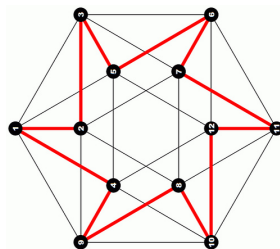
$$\begin{aligned} \exists L \exists S \text{ linear order}(L) \wedge \\ S \text{ is the successor relation of } L \wedge \\ \forall x \exists y (L(x, y) \vee L(y, x)) \wedge \\ \forall x \forall y (S(x, y) \Rightarrow E(x, y)) \end{aligned}$$

L is a linear ordering relation.

$$\begin{aligned} (\forall x \neg L(x, x)) \wedge \\ (\forall x \forall y \forall z (L(x, y) \wedge L(y, z) \Rightarrow L(x, z))) \wedge \\ \forall x \forall y ((x \neq y) \Rightarrow (L(x, y) \vee L(y, x))) \end{aligned}$$

S is a circular successor relation:

$$\begin{aligned} \forall x \forall y S(x, y) \Leftrightarrow \\ ((L(x, y) \wedge \neg \exists z (L(x, z) \wedge L(z, y))) \wedge \\ (\neg \exists z L(x, z) \wedge \neg \exists z L(z, y))) \end{aligned}$$



Second-Order Logic - Examples

- **Evenness**: The following formula can test if the size of a domain is even.

$$\begin{aligned} \exists A \exists B \quad & \forall x \exists y A(x, y) \wedge \\ & \forall x \forall y \forall z (A(x, y) \wedge A(x, z) \Rightarrow y = z) \wedge \\ & \forall x \forall y \forall z (A(x, z) \wedge A(y, z) \Rightarrow x = y) \wedge \\ & \forall x \forall y B(x) \wedge A(x, y) \Rightarrow \neg B(y) \wedge \\ & \forall x \forall y \neg B(x) \wedge A(x, y) \Rightarrow B(y) \end{aligned}$$

Characterizing NP

- **Theorem** (Fagin 1974)

Let Q be a query over the class of **finite structures**. Then the following are equivalent:

- Q is in NP.
- Q is definable by \exists SO.

$$\exists\text{SO} \equiv \text{NP}$$

- It reflects a normal form for NP algorithms:

- 1 “**Guessing**” phase
non-deterministic guess of a polynomially size bounded “certificate”;
- 2 “**Verifying**” phase
deterministic polynomial time verification of this certificate.

Characterizing NP

- **Lemma**

Every \exists SO definable query can be evaluated in NP.

- **Proof:**

- Let Q be $\exists X_1 \dots \exists X_n \varphi(X_1, \dots, X_n)$, where φ is a FO formula.
- Given \mathfrak{A} , the non-deterministic machine first

Step 1 : guesses R_1, \dots, R_n for X_1, \dots, X_n , and

Step 2 : checks if $\varphi(R_1, \dots, R_n)$ holds.

- Step 2 can be done in polynomial time in the length of the encoding string of \mathfrak{A} plus the size of R_1, \dots, R_n .
- Hence, the computation is polynomially time bounded.

Characterizing NP

- **Lemma**

Every query over finite structures in NP can be expressed in \exists SO.

- **Proof:**

- Suppose $M = (S, \Sigma, \Delta, \delta, q_0, S_a, S_r)$ is a nondeterministic polynomial time TM with a one-way infinite tape, where $\Sigma = \{0, 1\}$, $S = \{q_0, \dots, q_{m-1}\}$ and $\Delta = \Sigma \cup \{-\}$.
- The sentence describing acceptance by M on encodings of structures has the form

$$\exists L \exists T_0 \exists T_1 \exists T_2 \exists H_{q_0} \dots \exists H_{q_{m-1}} \varphi,$$

where

- L is linear order on the universe;
- T_0, T_1, T_2 are tape predicates;
- $\{H_q \mid q \in S\}$ are head predicates;
- φ is a FO formula stating that when M starts on the encoding of \mathfrak{A} , the predicates T_i 's and H_q 's correspond to its computation, and eventually M reaches an accepting state.

Characterizing NP

- The class coNP consists of the problem whose complements are in NP, while the negation of an \exists SO sentence is an \forall SO sentence.

- **Corollary**

\forall SO captures coNP.

- To show that $\text{NP} \neq \text{coNP}$, it would suffice to exhibit a query definable in \forall SO but not definable in \exists SO. Hence, we have:

$$\begin{aligned} \forall\text{SO} \neq \exists\text{SO} &\Rightarrow \text{NP} \neq \text{coNP} \\ &\Rightarrow \text{Ptime} \neq \text{NP} \end{aligned}$$

- Note that, the separation of \exists SO and \forall SO is specific to finite structures. Over some infinite structures (e.g., $(\mathbb{N}, +, \cdot)$), the logics \exists SO and \forall SO are known to be different.

Fixed Point Logics

- Extending FO with fixed point operators to formalize inductive definitions.
 - FP: define new relations inductively;
 - SO: quantify over relations arbitrarily.
- Transitive closure:** Given a binary relation E , the **transitive closure** of E is the smallest relation R satisfying:
 - $E \subseteq R$, and
 - if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$.
- We can have a sequence of relations in the computation:

$$R^0, R^1, \dots, R^n,$$
 where R^n is a fixed point of an operator that sends each R^i to R^{i+1} .
- Transitive closure is not expressible in FO.

Fixed Point Logics

- Let D be a finite set, and $\mathcal{P}(D)$ be its powerset. An **operator** on D is a mapping $F : \mathcal{P}(D) \mapsto \mathcal{P}(D)$.

Some properties of operators:

- F is **monotone** if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$;
- F is **inflationary** if $X \subseteq F(X)$ for all $X \in \mathcal{P}(D)$, i.e., the sequence $(F^n(X))_{n \in \mathbb{N}}$ is increasing in the sense that $X \subseteq F(X) \subseteq F(F(X)) \subseteq \dots$

$$(F \text{ monotone}) \Rightarrow (F \text{ inflationary})$$

- A set $X \subseteq D$ is a **fixed point** of F if $F(X) = X$.
- A set $X \subseteq D$ is a **least fixed point** of F , denoted as **lfp**(F), if it is a fixed point and $X \subseteq Y$ for every other fixed point Y of F .

Fixed Point Logics

- Every monotone operator has a least fixed point, which is the union of relations in an increasing sequence X^0, X^1, \dots, X^{i+1} , i.e.,

$$\mathbf{lfp}(F) = \bigcup_{i=0}^{\infty} X^i,$$

- However, not all the operators are monotone.

1 Inflationary fixed point

$$\mathbf{ifp}(F) = \bigcup_{i=0}^n X^i \cup F(X^i)$$

2 Partial fixed point

$$\mathbf{pfp}(F) = \begin{cases} X^n & \text{if } X^n = X^{n+1} \\ \emptyset & \text{if } X^n \neq X^{n+1} \text{ for all } n \leq 2^{|D|}. \end{cases}$$

- If F is monotone, then **lfp**(F) = **ifp**(F) = **pfp**(F).

Fixed Point Logics - IFP and PFP

- We now add formulas for computing fixed points into FO.
- Suppose $\varphi(X, \vec{x})$ be a formula of $\sigma \cup \{X\}$, i.e., contains a SO variable X of arity k , and a tuple \vec{x} of k free FO variables. For each structure \mathfrak{A} , φ defines an operator

$$F_\varphi : \mathcal{P}(A^k) \rightarrow \mathcal{P}(A^k)$$

such that

$$F_\varphi(X) = \{\vec{a} \mid \mathfrak{A} \models \varphi(X/R, \vec{a})\}.$$

- Syntax:** If $\varphi(X, \vec{x})$ is a formula, then $[\mathbf{fp}_{X, \vec{x}} \varphi(X, \vec{x})](\vec{t})$ is also a formula, whose free variables are those of \vec{t} .

Semantics:

- For IFP: $\mathfrak{A} \models [\mathbf{fp}_{X, \vec{x}} \varphi(X, \vec{x})](\vec{a})$ iff $\vec{a} \in \mathbf{ifp}(F_\varphi)$.
- For PFP: $\mathfrak{A} \models [\mathbf{pfp}_{X, \vec{x}} \varphi(X, \vec{x})](\vec{a})$ iff $\vec{a} \in \mathbf{pfp}(F_\varphi)$.

- Can we define an extension of FO with the least fixed point in exactly the same way?

Fixed Point Logics - LFP

- Unfortunately, testing if F_φ is monotone is undecidable for FO formulas φ .
- To ensure that lfps are only taken for monotone operators, we impose syntactic restrictions.
 - $\varphi(X, \bar{x})$ is **positive** in X if every occurrence of X is inside of an even number of negations.

Example: Are the following formulas positive in R or S ?

 - $\exists x \neg R(x) \vee \neg \forall y \forall z \neg (R(y) \wedge \neg R(z))$
 - $R(x, y) \wedge \neg \exists y \forall z (\neg R(x, y) \wedge S(x, y))$
 - If $\varphi(X, \bar{x})$ is positive in X , then F_φ is monotone (results from classical FO logic).
- Syntax:** the same as IFP and PFP.
- Semantics:** $\mathfrak{A} \models [\text{lfp}_{X, \bar{x}} \varphi(X, \bar{x})](\vec{a})$ iff $\vec{a} \in \text{lfp}(F_\varphi)$.

Fixed Point Logics - Examples

- Consider a graph whose edge relation is E .
 - Suppose that $\varphi(R, x, y)$ is defined by:

$$E(x, y) \vee \exists z (E(x, z) \wedge R(z, y))$$
 - Transitive closure:**

$$[\text{lfp}_{R, x, y} \varphi(R, x, y)](u, v)$$
 - Graph connectivity:**

$$\forall u \forall v [\text{lfp}_{R, x, y} \varphi(R, x, y)](u, v)$$
 - Suppose that $\psi(R, x)$ is defined by:

$$\forall y (E(y, x) \Rightarrow R(y))$$
 - Acyclicity:**

$$\forall u [\text{lfp}_{R, x} \psi(R, x)](u)$$

Fixed Point Logics - Examples

- Consider a graph whose edge relation is E , and $\varphi_1(R, x, y)$ defined by:

$$E(x, y) \vee \exists z (E(x, z) \wedge R(z, y))$$

Are the least, inflationary and partial fixed points of φ_1 the same?
- They all compute the **transitive closure** of the graph. In each case, $R^n = \{(u, v) \mid \text{there is a path between } u \text{ and } v \text{ of length less than } n\}$.

$$[\text{lfp}_{R, x, y} \varphi_1(R, x, y)](u, v) =$$

$$[\text{ifp}_{R, x, y} \varphi_1(R, x, y)](u, v) =$$

$$[\text{pfp}_{R, x, y} \varphi_1(R, x, y)](u, v)$$

Fixed Point Logics - Examples

- Consider a graph whose edge relation is E , and $\varphi_2(R, x, y)$ defined by:

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

Are the least, inflationary and partial fixed points of φ_2 the same?
- LFP and PFP return an empty set, i.e., $R^n = \emptyset$.

$$[\text{lfp}_{R, x, y} \varphi_2(R, x, y)](u, v) =$$

$$[\text{pfp}_{R, x, y} \varphi_2(R, x, y)](u, v)$$
- IFP still computes the **transitive closure** of the graph, i.e., $R^n = \{(u, v) \mid \text{there is a path between } u \text{ and } v \text{ of length less than } n\}$.

- **Theorem** (Gurevich and Shelah, 1986)
For every formula in IFP there is an LFP formula equivalent to it on all finite structures, i.e.,

$$\text{LFP} \equiv \text{IFP over all finite structures}$$

- **Theorem** (Kreutzer, 2002)
For every formula in IFP there is an equivalent formula in LFP, i.e.,

$$\text{LFP} \equiv \text{IFP over all structures}$$

- These results give us freedom to formalize inductive definitions in IFP (rather than in LFP) – don't need to check whether or not a formula is positive!

Characterizing Ptime

- **Theorem** (Immerman and Vardi, 1982)
Let Q be a query over the class of **ordered finite structures**. Then the following are equivalent:

- Q is in Ptime.
- Q is definable by LFP.

$$\text{LFP} + < \equiv \text{IFP} + < \equiv \text{Ptime}$$

- The **restriction to ordered structures** is:
 - 1 **necessary** for enabling sufficient coding machinery; (algorithms/Turing machines always work with an ordered input representation)
 - 2 **unsatisfactory** for guarantee semantic independence. (the answer of a query needs to be independent of the ordering in an input representation, but there are $n!$ many orderings to consider!)

- Expressiveness of fixed point logics over all structures:

$$\text{FO} \subsetneq \text{LFP} \equiv \text{IFP} \subseteq \text{PFP}$$

- Adding an order can increase their expressiveness, i.e.,

$$\begin{aligned} \text{LFP} &\subsetneq \text{LFP} + < \\ \text{IFP} &\subsetneq \text{IFP} + < \\ \text{PFP} &\subsetneq \text{PFP} + < \end{aligned}$$

- The query that separates these logics on ordered and unordered structures is **evenness**.
 - **evenness** is not expressible in any of LFP, IFP and PFP.
 - **evenness** is expressible in $\text{LFP} + <$, $\text{IFP} + <$ and $\text{PFP} + <$.

Characterizing Ptime

- **Lemma**
Every LFP definable query can be evaluated in Ptime over ordered finite structures.

- **Proof:**
 - By induction on formulas
 - Formulas with only Boolean connectives and quantifiers can be handled as for FO.
 - For formulas of the form $[\text{ifp}_{X, \bar{x}\varphi}](\vec{t})$, $F_\varphi : \mathcal{P}(A^k) \rightarrow \mathcal{P}(A^k)$ is a Ptime-computable monotone operator (i.e., to compute X^{i+1} from X^i , it goes through all $\vec{a} \in A^k$ and evaluates φ).
 - Since the fixed point computation stops after at most $|A^k|$ iteration, and in each iteration $[\text{ifp}_{X, \bar{x}\varphi}](\vec{t})$ can be computed in polynomial time, every LFP formula can be evaluated in Ptime.

Characterizing Ptime

- **Lemma**

Every Ptime query of ordered finite structures can be expressed in LFP.

- **Proof:**

- Suppose that $M = (S, \Sigma, \Delta, \delta, q_0, S_a, S_r)$ is a deterministic polynomial time TM with a one-way infinite tape, where $\Sigma = \{0, 1\}$ and $\Delta = \Sigma \cup \{-\}$. The linear order that is defined existentially in the proof of the only if direction of

Characterizing Pspace

- **Theorem** (Abiteboul and Vianu, 1982)

Let Q be a query over the class of **ordered finite structures**. Then the following are equivalent:

- Q is in Pspace.
- Q is definable by PFP.

PFP + \equiv Pspace

- **Lemma**

every Pspace query of ordered finite structures can be expressed in PFP

(The proof can be done by modifying the proof of the Immerman-Vardi theorem to simulate the accepting condition of a Turing machine using a PFP formula.)

Characterizing Pspace

- **Lemma**

Every PFP definable query can be evaluated in Pspace over ordered finite structures.

- **Proof:**

- We need to show that the evaluation of F_φ for a partial fixed point $[\mathbf{pfp}_{X, \bar{x}\varphi}](\vec{t})$ with k -ary X is in Pspace.
- By the definition, one only need to check the following:

$$\mathbf{pfp}(F) = \begin{cases} X^n & \text{if } X^n = X^{n+1} \\ \emptyset & \text{if } X^n \neq X^{n+1} \text{ for all } n \leq 2^{|D|}. \end{cases}$$

For that, it suffices to enumerate all the subsets of A^k , one by one (which can be done in Pspace).

- Since only $2^{|D|^k}$ steps need to be made, the entire computation is in Pspace.

Ptime vs Pspace

- Whether $Ptime \subsetneq Pspace$ is one of the open problems in computational complexity theory.

- Based on their logical definability, we have:

Ptime \equiv Pspace \Leftrightarrow LFP \equiv PFP over all ordered finite structures

- **Theorem** (Abiteboul and Vianu, 1991)

The following are equivalent:

- LFP and PFP have the expressive power over all finite structures.
- Pspace collapses to Ptime.

Ptime \equiv Pspace \Leftrightarrow LFP \equiv PFP over all finite structures

That is, the restriction to ordered structures can indeed be removed.

The Role of Order

- No results show that a logic can characterize a complexity class below NP on the class of all finite structures.
- Order provides a specific encoding, but algorithms in a complexity class should yield the same result on all different encodings of the input.

Open Problem – A Logic for P?

- **Open problem** (Chandra and Harel, 1982)
Is there a logic that captures Ptime on all and not just ordered structures?
- Recall that, **LFP + \leq \equiv Ptime**.
- *Can we drop the ordering \leq but still capture Ptime?*
 - If dropping “ordered” from Immerman and Vardi’s Theorem, then the theorem would be false.
 - **evenness** is not expressible in LFP, IFP or PFP.

Open Problem – A Logic for P?

- **Conjecture** (Gurevich, 1988)
There is no logic that captures Ptime on the class of all finite structures.
- *Can we enrich LFP so that LFP can express counting properties? Can we restrict structures so that the ordering is not necessary?*
 - Study various extensions of FO, such as generalizing qualifiers and adding fixed-point operators
 - Study various restrictions on finite structures, such as the class of all graphs with certain properties.

Open Problem – A Logic for P?

- Let L be a logic and \mathfrak{C} a class of finite structures. Then L captures Ptime on \mathfrak{C} if it satisfies the following conditions:
 - 1 The set of sentences in L is decidable.
 - 2 There is an algorithm that associates with every sentence φ in L a polynomial time algorithm that decides P_φ on \mathfrak{C} .
 - 3 For every polynomial time algorithm that decides a query P on \mathfrak{C} , there is a sentence φ in L that defines P on \mathfrak{C} .
- L captures Ptime if L captures Ptime on the class of **all finite structures**.

Open Problem – A Logic for P?

- Let L be a logic and \mathfrak{G} a class of finite structures. Then L captures Ptime on \mathfrak{G} if it satisfies the following conditions:
 - The set of sentences in L is decidable.
 - There is an algorithm that associates with every sentence φ in L a polynomial time algorithm that decides P_φ on \mathfrak{G} .
 - For every polynomial time algorithm that decides a query P on \mathfrak{G} , there is a sentence φ in L that defines P on \mathfrak{G} .
- L captures Ptime if L captures Ptime on the class of **all finite structures**.
- FO and IFP only satisfy (1) and (2) for the class of all finite structures, but does not satisfy (3).

Open Problem – A Logic for P?

- Let L be a logic and \mathfrak{G} a class of finite structures. Then L captures Ptime on \mathfrak{G} if it satisfies the following conditions:
 - The set of sentences in L is decidable.
 - There is an algorithm that associates with every sentence φ in L a polynomial time algorithm that decides P_φ on \mathfrak{G} .
 - For every polynomial time algorithm that decides a query P on \mathfrak{G} , there is a sentence φ in L that defines P on \mathfrak{G} .
- L captures Ptime if L captures Ptime on the class of **all finite structures**.
- Without (2), we could cheat ... Let \mathfrak{P} be polynomial time decidable queries over all finite structures. Then we could have a “logic”:
 - $L := \{\varphi \mid P_\varphi \in \mathfrak{P}\}$;
 - $\mathfrak{A} \models \varphi$ if $\mathfrak{A} \in P_\varphi$ for every structure \mathfrak{A} and $\varphi \in L$.

Open Problem – A Logic for P?

- A candidate logic*: IFP + C (the extension of FO with inflationary fixed-points and counting quantifiers).
- Some negative results**
 - Theorem** (Cai et al., 1992)
IFP + C does not express all polynomial time properties of graphs.
- Some positive results**

IFP + C does capture Ptime on specific classes of finite structures (the key idea is to find an IFP+C-definable linear order on restricted classes).

 - Theorem** (Grohe, 1998)
IFP + C captures Ptime on all planar graphs.
 - Theorem** (Grohe and J. Mariño, 1999)
IFP + C captures Ptime on all graphs with bounded tree-width.
 - Theorem** (Grohe, 2010))
IFP + C captures Ptime on every class of graphs that exclude a minor.