Overview of Topics

Part 3: – Logic and Complexity

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

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.

- $\text{Ptime} \subseteq \{\text{NP}, \text{coNP}\} \subseteq \text{Pspace}$

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:
  \[
P = \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k).
  \]

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

**Complexity Class – NP**

- **NP** is the class of languages accepted by a nondeterministic TM in polynomial time:
  \[
  \text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k).
  \]

- **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).

- It is not known:
  - whether the containment is proper, and
  - whether NP equals coNP.

**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 \( \text{DSPACE}(f(n)) \) if \( L \) is decided by M and the running space of M is \( O(f(n)) \).

**Complexity Class – Pspace**

- **Pspace** is the class of languages decidable in polynomial-space on a deterministic TM:
  \[
  \text{Pspace} = \bigcup_{k \in \mathbb{N}} \text{DSPACE}(n^k).
  \]

- By Savitch’s theorem, the nondeterministic case collapses to the deterministic one, in the case of space complexity.
\[ P = \text{the class of languages for which membership can be decided quickly.} \]

\[ NP = \text{the class of languages for which membership can be verified quickly.} \]

**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:
  - Every query that can be evaluated in \( C \) over finite structures is definable in \( L \).
  - 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.
  - Fagin, 1974
    \[ \text{NP} \equiv \exists \text{SO on all finite structures} \]
  - Immerman and Vardi, 1982
    \[ \text{Ptime} \equiv \text{LFP on all ordered finite structures} \]
  - Abiteboul and Vianu, 1982
    \[ \text{Pspace} \equiv \text{PFP on all ordered finite structures} \]
  - \( \ldots \)

- 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 \( \varphi(X_1, \ldots, X_n) \) be a FO formula.

- **Existential second-order logic (\( \exists \text{SO} \))** has the form:
  \[ \exists X_1 \ldots \exists X_n \varphi(X_1, \ldots, X_n); \]
- **Universal second-order logic (\( \forall \text{SO} \))** has the form:
  \[ \forall X_1 \ldots \forall X_n \varphi(X_1, \ldots, X_n), \]

### Second-Order Logic - Examples

- **Hamiltonian graph**: The following formula can test if a graph contains a Hamiltonian cycle.
  \[ \exists L \exists S \text{ linear order}(L) \land \]
  \[ \forall x \exists y (L(x, y) \lor L(y, x)) \land \]
  \[ \forall x \forall y (S(x, y) \Rightarrow E(x, y)) \]

  \( L \) is a linear ordering relation.

  \[ (\forall x \neg L(x, x)) \land \]
  \[ (\forall x \forall y (L(x, y) \land L(y, z) \Rightarrow L(x, z))) \land \]
  \[ (\forall x \forall y ((x \neq y) \Rightarrow (L(x, y) \lor L(y, x)))) \]

  \( S \) is a circular successor relation:

  \[ \forall x \forall y S(x, y) \Leftrightarrow \]
  \[ ((L(x, y) \land \exists z (L(x, z) \land L(z, y)))) \land \]
  \[ (\neg \exists z L(x, z) \land \neg \exists z L(z, y))) \]

- **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.
  \[ \exists R \exists B \exists G \forall x (R(x) \lor B(x) \lor G(x)) \land \]
  \[ \forall x (\neg (R(x) \land B(x)) \land \neg (B(x) \land G(x)) \land \neg (R(x) \land G(x))) \land \]
  \[ \forall x \forall y (E(x, y) \Rightarrow (\neg (R(x) \land R(y))) \land \]
  \[ (\neg (B(x) \land B(y))) \land \]
  \[ \neg (G(x) \land G(y))) \land \]

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

  \[ \exists A \exists B \forall x \exists y (A(x, y) \land \]
  \[ \forall x \forall y \forall z (A(x, y) \land A(x, z) \Rightarrow y = z) \land \]
  \[ \forall x \forall y \forall z (A(x, z) \land A(y, z) \Rightarrow x = y) \land \]
  \[ \forall x \forall y \exists B(x) \land A(x, y) \Rightarrow \neg B(y) \land \]
  \[ \forall x \forall y \neg B(x) \land A(x, y) \Rightarrow B(y) \]
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 $\exists SO$.

$\exists SO \equiv NP$

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.

Lemma

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

Proof:

Let $Q$ be $\exists X_1 \ldots \exists X_n \varphi(X_1, \ldots, X_n)$, where $\varphi$ is a FO formula.

Given $\exists \alpha$, the non-deterministic machine first

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

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

Step 2 can be done in polynomial time in the length of the encoding string of $\exists \alpha$ plus the size of $R_1, \ldots, R_n$.

Hence, the computation is polynomially time bounded.

Lemma

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

Proof:

Suppose $M = (S, \Sigma, \Delta, q_0, S_a, S_r)$ is a nondeterministic polynomial time TM with a one-way infinite tape, where $\Sigma = \{0, 1\}$, $S = \{q_0, \ldots, q_{m-1}\}$ and $\Delta = \Sigma \cup \{-\}$.

The sentence describing acceptance by $M$ on encodings of structures has the form

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

where

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

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 NP $\neq$ coNP, it would suffice to exhibit a query definable in $\forall SO$ but not definable in $\exists SO$. Hence, we have:

$$\forall SO \neq \exists SO \Rightarrow NP \neq coNP \Rightarrow Ptime \neq NP$$

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

- Recall that we have seen some properties that cannot be expressed by first-order logic over finite structures:
  - evenness
  - acyclicity
  - connectivity
  - colorability
  - hamiltonicity

- Some properties can be expressed by second-order logic, such as, evenness, colorability and hamiltonicity.

- How about the other properties, such as connectivity? How can first-order logic be extended to express them?

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: $R^0, R^1, \ldots, R^n$,
  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 $F : \mathcal{P}(D) \rightarrow \mathcal{P}(D)$.

- 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 $X \in \mathcal{P}(X)$, i.e., the sequence $(F^n(X))_{n \in \mathbb{N}}$ is increasing in the sense that $X \subseteq F(X) \subseteq F(F(X)) \subseteq \ldots$.

- 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 $\text{lfp}(F)$, if it is a fixed point and $X \subseteq Y$ for every other fixed point $Y$ of $F$.

- Every monotone operator has a least fixed point, which is the union of relations in an increasing sequence $X^0, X^1, \ldots, X^{n+1}$, i.e.,
  \[
  \text{lfp}(F) = \bigcup_{n=0}^{\infty} X^n.
  \]

- However, not all the operators are monotone.
  - Inflationary fixed point
    \[
    \text{ifp}(F) = \bigcup_{n=0}^{n} X^n \cup F(X^n)
    \]
  - Partial fixed point
    \[
    \text{pfp}(F) = \begin{cases} X^n & \text{if } X^n = X^{n+1} \\ \emptyset & \text{if } X^n \neq X^{n+1} \text{ for all } n \leq \left\lceil \frac{\log \left( \frac{\|D\|}{\|X^n\|} \right)}{\log 2} \right\rceil. \end{cases}
    \]

- If $F$ is monotone, then $\text{lfp}(F) = \text{ifp}(F) = \text{pfp}(F)$.
Fixed Point Logics - IFP and PFP

- We now add formulas for computing fixed points into FO.
- Suppose \( \varphi(X, \bar{x}) \) be a formula of \( \sigma \cup \{X\} \), i.e., contains a SO variable X of arity k, and a tuple \( \bar{x} \) of k free FO variables. For each structure \( \mathcal{A} \), \( \varphi \) defines an operator
  \[
  F_\varphi : \mathcal{P}(A^k) \rightarrow \mathcal{P}(A^k)
  \]
  such that
  \[
  F_\varphi(X) = \{ \bar{a} \in A^k : \mathcal{A} \models \varphi(X/R, \bar{a}) \}.
  \]

- Syntax: If \( \varphi(X, \bar{x}) \) is a formula, then \( \lfp\varphi(X, \bar{x}) \) is also a formula, whose free variables are those of \( \bar{t} \).

- Semantics:
  - For IFP: \( \mathcal{A} \models [\lfp\varphi(X, \bar{x})](\bar{a}) \) iff \( \bar{a} \in \lfp(F_\varphi) \).
  - For PFP: \( \mathcal{A} \models [\lfp\varphi(X, \bar{x})](\bar{a}) \) iff \( \bar{a} \in \pfp(F_\varphi) \).

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

Fixed Point Logics - LFP

- Unfortunately, testing if \( F_\varphi \) is monotone is undecidable for formulas \( \varphi(X, \bar{x}) \).
- 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 \( S \)?
    1. \( \exists x \neg R(x) \lor \forall y \forall z (R(y) \land \neg R(z)) \)
    2. \( R(x, y) \land \neg \exists y \forall z (\neg R(x, y) \land S(x, y)) \)
- If \( \varphi(X, \bar{x}) \) is positive in \( X \), then \( F_\varphi \) is monotone (results from classical FO logic).

- Syntax: the same as IFP and PFP.

- Semantics:
  - For IFP: \( \mathcal{A} \models [\lfp\varphi(X, \bar{x})](\bar{a}) \) iff \( \bar{a} \in \lfp(F_\varphi) \).
  - For PFP: \( \mathcal{A} \models [\lfp\varphi(X, \bar{x})](\bar{a}) \) iff \( \bar{a} \in \pfp(F_\varphi) \).

Fixed Point Logics - Examples

- Consider a graph whose edge relation is \( E \).
  - Suppose that \( \varphi(R, x, y) \) is defined by:
    \[
    E(x, y) \lor \exists z(E(x, z) \land R(z, y))
    \]
  - Transitive closure:
    \[
    \lfp\varphi(R, x, y)(u, v)
    \]
  - Graph connectivity:
    \[
    \forall u \forall v [\lfp\varphi(R, x, y)](u, v)
    \]

- Consider a graph whose edge relation is \( E \), and \( \varphi_1(R, x, y) \) defined by:
  \[
  E(x, y) \lor \exists z(E(x, z) \land R(z, y))
  \]
  Are the least, inflationary and partial fixed points of \( \varphi_1 \) the same?
Consider a graph whose edge relation is $E$, and $\varphi_1(R, x, y)$ defined by:

$$E(x, y) \lor \exists z (E(x, z) \land R(z, y))$$

Are the least, inflationary and partial fixed points of $\varphi_1$ the same?

They all compute the **transitive closure** of the graph. In each case, $R^n = \{(u, v) |$ there is a path between $u$ and $v$ 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:

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

Are the least, inflationary and partial fixed points of $\varphi_2$ the same?

LFP and PFP return an empty set, i.e., $R^n = \emptyset$.

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

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

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

IFP still computes the **transitive closure** of the graph, i.e., $R^n = \{(u, v) |$ there is a path between $u$ and $v$ of length less than $n\}$.

**IFP vs LFP**

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!
IFP vs LFP vs PFP

- Expressiveness of fixed point logics over all structures:
  \[ \text{FO} \subset \text{LFP} \equiv \text{IFP} \subset \text{PFP} \]

- Adding an order can increase their expressiveness, i.e.,
  \[ \text{LFP} \subset \text{LFP +} < \]
  \[ \text{IFP} \subset \text{IFP +} < \]
  \[ \text{PFP} \subset \text{PFP +} < \]

- 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 LFP + <, IFP + < and PFP + <.

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:
  - **necessary** for enabling sufficient coding machinery;
    (algorithms/Turing machines always work with an ordered input representation)
  - **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!)

Logic summer school 34

- **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 \([\text{ifp}_{X,x}\vec{t}](\vec{a})\), \( F_{\varphi} : \mathcal{P}(A^k) \to \mathcal{P}(A^k) \) 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 \( \varphi \)).
    - Since the fixed point computation stops after at most \( |A^k| \) iteration, and in each iteration \([\text{ifp}_{X,x}\vec{t}](\vec{a})\) can be computed in polynomial time, every LFP formula can be evaluated in Ptime.

Logic summer school 35

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

  - **Proof**:
    (The proof can be done by simulating the accepting condition of a Turing machine using a LFP formula.)
**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.

$\text{PFP} \subseteq \equiv \text{Pspace}$

**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.)

**Lemma**

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

**Proof:**

We need to show that the evaluation of $F_{\forall}$ for a partial fixed point $[\text{pfp}_{X,\vec{x}}](t)$ with k-ary $X$ is in Pspace.

By the definition, one only need to check the following:

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

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 $\text{Ptime} \subseteq \text{Pspace}$ is one of the open problems in computational complexity theory.

- Based on their logical definability, we have:

  $\text{Ptime} \equiv \text{Pspace} \iff \text{LFP} \equiv \text{PFP}$ 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.

$\text{Ptime} \equiv \text{Pspace} \iff \text{LFP} \equiv \text{PFP}$ 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, $\text{LFP} \subseteq \text{Ptime}$.

- 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 $\mathcal{S}$ a class of finite structures. Then $L$ captures Ptime on $\mathcal{S}$ 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 $\varphi$ in $L$ a polynomial time algorithm that decides $P_{\varphi}$ on $\mathcal{S}$.
  3. For every polynomial time algorithm that decides a query $P$ on $\mathcal{S}$, there is a sentence $\varphi$ in $L$ that defines $P$ on $\mathcal{S}$.

- $L$ 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 $S$ a class of finite structures. Then $L$ captures Ptime on $S$ 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 $\varphi$ in $L$ a polynomial time algorithm that decides $P_\varphi$ on $\mathcal{S}$.
3. For every polynomial time algorithm that decides a query $P$ on $\mathcal{S}$, there is a sentence $\varphi$ in $L$ that defines $P$ on $\mathcal{S}$.

$L$ 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).

Some negative results

- Theorem (Cai et al., 1992)
  IFP + C does not express all polynomial time properties of graphs.

Some positive results

- 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.

A General Picture for Descriptive complexity

[Diagram]

1. The picture is taken from the cover of the book “Descriptive Complexity” by Neil Immerman