Finite Model Theory
Lecture 16: Descriptive Complexity

Spring 2025

Announcement

- HW4 (the last one!!) is posted and due on May 30.
- No lectures on 5/26, 5/28, 6/4
- Next (and last) lecture: Monday, 6/2.
  We will do a review of the topics covered this quarter.
  Please come and participate.
  If you have a particular topic to discuss, post it on Ed or email me.

Review

Review: Encoding of a TM

Fix $M = (Q, \{0,1\}, \Delta, q_0, Q_F)$.
Define: $\sigma_M = (<, T_0(\cdot,\cdot), T_1(\cdot,\cdot), H(\cdot,\cdot), (S_q(\cdot))_{q \in Q})$

Meaning:

- $<$ is a total order
- $T_0(t,p), T_1(t,p)$: the tape content at time t position p is 0 or 1.
- $H(t,p)$: the head at time t is on position p.
- $S_q(t)$: the Turing Machine is is stated q at time t.

Review: Details of the Encoding

Fix $M = (Q, \{0,1\}, \Delta, q_0, Q_F)$.
Define: $\sigma_M = (<, T_0(\cdot,\cdot), T_1(\cdot,\cdot), H(\cdot,\cdot), (S_q(\cdot))_{q \in Q})$

The sentence φ_M asserts the following:

- General consistency: $<$ is a total order, every tape has exactly one symbol, the head is on exactly one position, etc.
- At time $t = \min$, the TM is in the initial configuration.
- At time $t = \max$, the TM is in an accepting configuration.
- The configuration at time t yields that at time $t + 1$

Review: the Proof of Trakhtenbrot's Theorem

φ_M is satisfiable iff M terminates. **Proof:**

- If M terminates, then there exists an accepting computation history $\boldsymbol{C} = c_1, c_2, \ldots, c_T$. From $\boldsymbol{C}$, we constructed a structure s.t. $\boldsymbol{A} \vDash \varphi_M$.
- If $\boldsymbol{A} \vDash \varphi_M$, then construct an accepting computation $c_1, \ldots, c_T$, by reading the configuration along the chain $0 < 1 < 2 \cdots$

Can we replace $<$ with $\mathtt{succ}$?

Review: the Proof of Trakhtenbrot's Theorem

φ_M is satisfiable iff M terminates. **Proof:**

- If M terminates, then there exists an accepting computation history $\boldsymbol{C} = c_1, c_2, \ldots, c_T$. From $\boldsymbol{C}$, we constructed a structure s.t. $\boldsymbol{A} \vDash \varphi_M$.
- If $\boldsymbol{A} \vDash \varphi_M$, then construct an accepting computation $c_1, \ldots, c_T$, by reading the configuration along the chain $0 < 1 < 2 \cdots$

Can we replace $<$ with $\texttt{succ}$?

YES The first item doesn't change, while for the second item we only use the (unique!) chain, and ignore any loops.

Descriptive Complexity

Descriptive Complexity

A *problem* is a function $P : \mathrm{STRUCT}[\sigma] \to \{0, 1\}$.

- A *computational complexity class* is the set of problems that can be answered within some fixed computational resources:. E.g. LOGSPACE, PTIME, PSPACE: Turing Machine M s.t. $P(\boldsymbol{A}) = \boldsymbol{T}$ iff M accepts $\boldsymbol{A}$.

Descriptive Complexity

A *problem* is a function $P : \text{STRUCT}[\sigma] \to \{0, 1\}$.

- A *computational complexity class* is the set of problems that can be answered within some fixed computational resources:. E.g. LOGSPACE, PTIME, PSPACE: Turing Machine M s.t. $P(\boldsymbol{A}) = \boldsymbol{T}$ iff M accepts $\boldsymbol{A}$.

- A *descriptive complexity class* is the set of problems that can be represented in some fixed logic language L. E.g. FO, LFP, ∃SO, SO: sentence φ s.t. $P(\boldsymbol{A}) = \boldsymbol{T}$ iff $\boldsymbol{A} \models \varphi$.

Descriptive Complexity

A *problem* is a function $P : \text{STRUCT}[\sigma] \to \{0, 1\}$.

- A *computational complexity class* is the set of problems that can be answered within some fixed computational resources:. E.g. LOGSPACE, PTIME, PSPACE: Turing Machine M s.t. $P(\boldsymbol{A}) = \boldsymbol{T}$ iff M accepts $\boldsymbol{A}$.

- A *descriptive complexity class* is the set of problems that can be represented in some fixed logic language L. E.g. FO, LFP, ∃SO, SO: sentence φ s.t. $P(\boldsymbol{A}) = \boldsymbol{T}$ iff $\boldsymbol{A} \vDash \varphi$.

- We say that a logic L captures a complexity class, if they can express precisely the same problems.

Descriptive Complexity: Overview of Results

- $\text{FO}(+, *) = \text{FO}(<, \text{BIT}) = AC^0$
- $\text{FO}(\text{det-TC}, <) = \text{LOGSPACE}$, and $\text{FO}(\text{TC}, <) = \text{NLOGSPACE}$;
- $LFP(<) = \text{PTIME}$
- $\text{FO}(\text{PartialFixpoint}, <) = \text{PSPACE}$
- $\exists\text{SO} = \text{NP}$

Encodings

- A Turing Machine, accepts a language $L \subseteq \{0,1\}^*$.
- A sentence φ defines a set of models $\subseteq \text{STRUCT}[\sigma]$.
- To compare them, we to encode structures as strings.

Encoding STRUCT[σ] to $\{0,1\}^*$

Encode $\boldsymbol{A} = ([n], R_1^A, R_2^A, \dots)$ as follows:

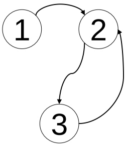
- Start with $0^n 1$.
- Encode R_i^A using "adjacency matrix", of length $n^{\text{arity}(R_i)}$
- Thus, $\boxed{\texttt{enc}(\boldsymbol{A}) = 01^n \texttt{enc}(R_1^A)\texttt{enc}(R_2^A)\cdots} \in \{0,1\}^*$
- Length of encoding: $1 + n + n^{\text{arity}(R_1)} + n^{\text{arity}(R_2)} + \cdots = n^{O(1)} = \text{poly}(n)$.

Encoding STRUCT[σ] to $\{0,1\}^*$

Encode $\boldsymbol{A} = ([n], R_1^A, R_2^A, \ldots)$ as follows:

- Start with $0^n 1$.
- Encode R_i^A using "adjacency matrix", of length $n^{\text{arity}(R_i)}$
- Thus, $\boxed{\texttt{enc}(\boldsymbol{A}) = 01^n \texttt{enc}(R_1^A)\texttt{enc}(R_2^A)\cdots} \in \{0,1\}^*$
- Length of encoding: $1 + n + n^{\text{arity}(R_1)} + n^{\text{arity}(R_2)} + \cdots = n^{O(1)} = \text{poly}(n)$.

Example:

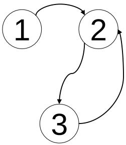


Encoding STRUCT[σ] to $\{0,1\}^*$

Encode $\boldsymbol{A} = ([n], R_1^A, R_2^A, \ldots)$ as follows:

- Start with $0^n 1$.
- Encode R_i^A using "adjacency matrix", of length $n^{\text{arity}(R_i)}$
- Thus, $\boxed{\mathtt{enc}(\boldsymbol{A}) = 01^n \mathtt{enc}(R_1^A)\mathtt{enc}(R_2^A)\cdots} \in \{0,1\}^*$
- Length of encoding: $1 + n + n^{\text{arity}(R_1)} + n^{\text{arity}(R_2)} + \cdots = n^{O(1)} = \text{poly}(n)$.

Example:



$$\mathtt{enc}(G) = \underbrace{0001}_{n=3}\underbrace{010001010}_{3\times 3 \text{ matrix}}$$

$\exists SO = NP$

∃SO and NP

∃SO consists of sentences $\exists S_1 \cdots \exists S_m \psi$, where $\psi \in$ FO over vocabulary $\sigma \cup \{S_1, \ldots, S_m\}$.

Theorem (Fagin)

$\exists SO = NP$.

∃SO and NP

∃SO consists of sentences $\exists S_1 \cdots \exists S_m \psi$, where $\psi \in$ FO over vocabulary $\sigma \cup \{S_1, \ldots, S_m\}$.

Theorem (Fagin)

$\exists SO = NP$.

We need to prove:

- $\exists$SO $\subseteq NP$: the data complexity of any $\varphi \in \exists$SO is in NP

∃SO and NP

∃SO consists of sentences $\exists S_1 \cdots \exists S_m \psi$, where $\psi \in$ FO over vocabulary $\sigma \cup \{S_1, \ldots, S_m\}$.

Theorem (Fagin)

$\exists SO = NP$.

We need to prove:

- $\exists$SO $\subseteq NP$: the data complexity of any $\varphi \in \exists$SO is in NP
- $NP \subseteq \exists$SO: for any problem in NP, there exists a sentence $\psi \in \exists$SO that expresses precisely that problem.

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

Given: $\varphi = \exists S_1 \cdots \exists S_m \psi$, input structure $\boldsymbol{A} \in \text{STRUCT}[\sigma]$.

$\exists$SO $\subseteq$ *NP*

Given: $\varphi = \exists S_1 \cdots \exists S_m \psi$, input structure $\boldsymbol{A} \in \text{STRUCT}[\sigma]$.

The TM starts with the input $\texttt{enc}(\boldsymbol{A}) = 0^n 1 \texttt{enc}(R_1^A) \cdots \texttt{enc}(R_k^A)$ on its tape, and does the following:

$\exists$SO $\subseteq$ *NP*

Given: $\varphi = \exists S_1 \cdots \exists S_m \psi$, input structure $\boldsymbol{A} \in \text{STRUCT}[\sigma]$.

The TM starts with the input $\texttt{enc}(\boldsymbol{A}) = 0^n 1 \texttt{enc}(R_1^A) \cdots \texttt{enc}(R_k^A)$ on its tape, and does the following:

- Guess the encodings of $S_1, S_2, \ldots$, and append them: $0^n 1 \texttt{enc}(R_1^A) \cdots \texttt{enc}(R_k^A) \texttt{enc}(S_1) \cdots \texttt{enc}(S_m)$

$\exists SO \subseteq NP$

Given: $\varphi = \exists S_1 \cdots \exists S_m \psi$, input structure $\boldsymbol{A} \in \mathrm{STRUCT}[\sigma]$.

The TM starts with the input $\mathtt{enc}(\boldsymbol{A}) = 0^n 1 \mathtt{enc}(R_1^A) \cdots \mathtt{enc}(R_k^A)$ on its tape, and does the following:

- Guess the encodings of $S_1, S_2, \ldots$, and append them: $0^n 1 \mathtt{enc}(R_1^A) \cdots \mathtt{enc}(R_k^A) \mathtt{enc}(S_1) \cdots \mathtt{enc}(S_m)$
- Evaluate the FO sentence ψ on this structure.

The plan for proving $NP \subseteq \exists SO$

For simplicity, consider the language of graphs $\sigma = (E)$.

The plan for proving $NP \subseteq \exists$SO

For simplicity, consider the language of graphs $\sigma = (E)$.

A graph problem P is in NP if $\exists$ non-deterministic TM M such that:

For any graph, $P(G)$ is true iff M has an accepting computation of polynomial length on input $\mathtt{enc}(G) \in \{0,1\}^*$.

We need to describe ψ_M s.t. $G \models \psi_M$ iff M accepts G.

Let's try this, like in the proof of Trakthenbrot's theorem:

$$\psi_M = \exists < \exists T_0(\cdot,\cdot) \exists T_1(\cdot,\cdot) \exists H(\cdot,\cdot) \exists S_{q_0}(\cdot) \exists S_{q_1}(\cdot) \cdots \varphi_M$$

The plan for proving $NP \subseteq \exists$SO

For simplicity, consider the language of graphs $\sigma = (E)$.

A graph problem P is in NP if $\exists$ non-deterministic TM M such that:

For any graph, $P(G)$ is true iff M has an accepting computation of polynomial length on input $\texttt{enc}(G) \in \{0,1\}^*$.

We need to describe ψ_M s.t. $G \models \psi_M$ iff M accepts G.

Let's try this, like in the proof of Trakthenbrot's theorem:

$$\psi_M = \exists < \exists T_0(\cdot,\cdot) \exists T_1(\cdot,\cdot) \exists H(\cdot,\cdot) \exists S_{q_0}(\cdot) \exists S_{q_1}(\cdot) \cdots \varphi_M$$

Now the tape needs to start with $\texttt{enc}(G)$.

The plan for proving $NP \subseteq \exists SO$

For simplicity, consider the language of graphs $\sigma = (E)$.

A graph problem P is in NP if $\exists$ non-deterministic TM M such that:

For any graph, $P(G)$ is true iff M has an accepting computation of polynomial length on input $\mathtt{enc}(G) \in \{0,1\}^*$.

We need to describe ψ_M s.t. $G \models \psi_M$ iff M accepts G.

Let's try this, like in the proof of Trakthenbrot's theorem:

$$\psi_M = \exists < \exists T_0(\cdot,\cdot) \exists T_1(\cdot,\cdot) \exists H(\cdot,\cdot) \exists S_{q_0}(\cdot) \exists S_{q_1}(\cdot) \cdots \varphi_M$$

Now the tape needs to start with $\mathtt{enc}(G)$.

And we need more time stamps than $|\mathrm{Dom}(G)|$.

Proof Details

The order relation $<$ can only provide us with a set of size $n \stackrel{\text{def}}{=} |\text{Dom}(G)|$.

The TM may run in time $T > n$. However, T is a polynomial in n.

Proof Details

The order relation $<$ can only provide us with a set of size $n \stackrel{\text{def}}{=} |\text{Dom}(G)|$.

The TM may run in time $T > n$. However, T is a polynomial in n.

For illustration, we will assume $T \leq n^3$.

Proof Details

The order relation $<$ can only provide us with a set of size $n \stackrel{\text{def}}{=} |\text{Dom}(G)|$.

The TM may run in time $T > n$. However, T is a polynomial in n.

For illustration, we will assume $T \leq n^3$.

Then we use triples $(x, y, z) \in (\text{Dom}(G))^3$ as our time stamps.

$(x, y, z) < (u, v, w)$ is the lexicographic order.

Proof Details

The order relation $<$ can only provide us with a set of size $n \stackrel{\text{def}}{=} |\text{Dom}(G)|$.

The TM may run in time $T > n$. However, T is a polynomial in n.

For illustration, we will assume $T \leq n^3$.

Then we use triples $(x, y, z) \in (\text{Dom}(G))^3$ as our time stamps.

$(x, y, z) < (u, v, w)$ is the lexicographic order.

$$
\begin{aligned}
\texttt{min}_3 =& (\text{min}, \text{min}, \text{min}) = (0, 0, 0) \\
\texttt{succ}(0, 0, 0) =& (0, 0, 1) \\
\texttt{succ}(0, 0, \text{max}) =& (0, 1, 0) \\
& \dots
\end{aligned}
$$

Proof Details

The relations encoding the TM use triples to represent time/space:

$$T_0(x, y, z, u, v, w), T_1(x, y, z, u, v, w)$$
$$S_{q_1}(x, y, z), S_{q_2}(x, y, z), \ldots$$

Remains to show:

- φ_M can check the constraints required for the TM,
- φ_M can initialize the tape with $\mathtt{enc}(G)$.

Proof Details: Checking the Constraints for the TM

- General consistency: every tape has exactly one symbol, the head is on exactly one position, etc: unchanged
- At time $t = (0, 0, 0)$, the TM is in the initial configuration. Need to replace empty tape with $\mathtt{enc}(G)$
- At time $t = (\max, \max, \max)$, the TM is in an accepting configuration.

Proof Details: Checking the Constraints for the TM

- General consistency: every tape has exactly one symbol, the head is on exactly one position, etc: unchanged
- At time $t = (0, 0, 0)$, the TM is in the initial configuration. Need to replace empty tape with $\texttt{enc}(G)$
- At time $t = (\max, \max, \max)$, the TM is in an accepting configuration.
- The configuration at time t yields that at time $t + 1$.
  Note that $t = (x, y, z)$ and $t + 1 = (u, v, w)$.

Proof Details: Computing the Successor

We need to be able to compute $t + 1$ from t.

Assuming $t = (x, y, z)$ and $t + 1 = (u, v, w)$:

$\mathrm{succ}_3(x,y,z,u,v,w) =$???

Proof Details: Computing the Successor

We need to be able to compute $t+1$ from t.

Assuming $t = (x, y, z)$ and $t+1 = (u, v, w)$:

$$\begin{aligned}\mathrm{succ}_3(x,y,z,u,v,w) = {} & ((x = u) \wedge (y = v) \wedge \mathrm{succ}(z, w)) \\ & \vee(x = u \wedge \mathrm{succ}(y, v) \wedge (z = \max) \wedge (w = \min)) \\ & \vee(\mathrm{succ}(x, u) \wedge (y = \max) \wedge (v = \min) \wedge (z = \max) \wedge (w = \min))\end{aligned}$$

Encoding of the Input

It remains show how to write the input $\texttt{enc}(G)$ to the tape at time $t = 0$

Recall that $\boxed{\texttt{enc}(G) = 0^n 1 \texttt{enc}(E)}$,
where $\texttt{enc}(E)$ is the $n \times n$ the adjacency matrix M.

Encoding of the Input

It remains show how to write the input $\texttt{enc}(G)$ to the tape at time $t = 0$

Recall that $\boxed{\texttt{enc}(G) = 0^n 1 \texttt{enc}(E)}$,
where $\texttt{enc}(E)$ is the $n \times n$ the adjacency matrix M.

If we didn't have the prefix 0^n1, then the encoding were this:

$$\begin{aligned} T_0(0,0,0, \quad 0,x,y) &= \quad \neg E(x,y) \\ T_1\underbrace{(0,0,0,}_{t=0} \quad \underbrace{0,x,y)}_{\text{tape position}} &= \quad E(x,y) \end{aligned}$$

because the bit M_{xy} is at location $nx + y$, whose representation is $(0, x, y)$

Encoding of the Input

It remains show how to write the input $\texttt{enc}(G)$ to the tape at time $t = 0$

Recall that $\boxed{\texttt{enc}(G) = 0^n 1 \texttt{enc}(E)}$,
where $\texttt{enc}(E)$ is the $n \times n$ the adjacency matrix M.

If we didn't have the prefix 0^n1, then the encoding were this:

$$T_0(0,0,0,\quad 0,x,y) = \quad \neg E(x,y)$$
$$T_1 \underbrace{(0,0,0,}_{t=0} \quad \underbrace{0,x,y)}_{\text{tape position}} = E(x,y)$$

because the bit M_{xy} is at location $nx + y$, whose representation is $(0, x, y)$

To add the prefix 0^n1 we "add" $n+1$ positions. At home

Discussion

- $\exists SO = NP$ gives a characterization of NP independent of computational resources (time/space).
- Generalization:
  - Σ_k^1 = sentences of the form $(\exists\cdots\exists)(\forall\cdots\forall)\cdots$
  - Π_k^1 = sentences of the form $(\forall\cdots\forall)(\exists\cdots\exists)\cdots$

  Then Σ_k^1 captures Σ_k^P and Π_k^1 captures Π_k^P.
- $\exists SO = \forall SO$ is equivalent to $NP = coNP$, hence an open problem. But we have seen $\exists MSO \neq \forall MSO$: Monadic NP$\neq$ Monadic coNP

$$LFP(<) = PTIME$$

Review

Fixpoint Logic, LFP extends FO with a least fixpoint predicate:

$$\boxed{\text{lfp}_{T,\boldsymbol{x}}\varphi}$$

where P is a fresh relational symbol that occurs positively, and φ is a formula that uses P has free variables $\boldsymbol{x}$.

Example: transitive closure $T(u, v)$ becomes:

$$\boxed{\text{lfp}_{T,x,y}(E(x, y) \vee \exists z(E(x, z) \wedge T(z, y)))(u, v)}$$

LFP and PTIME

Theorem (Immerman and Vardi)

(1) The data complexity of LFP is in PTIME.
(2) LFP(<) captures PTIME.

(1) remains true for LFP(<): data complexity is in PTIME.

Item (2) says that, any property that is in PTIME can be checked in LFP assuming that the input structure also contains a linear order on the domain.

Why do we need a linear order?

LFP and PTIME

Theorem (Immerman and Vardi)

(1) The data complexity of LFP is in PTIME.
(2) LFP(<) captures PTIME.

(1) remains true for LFP(<): data complexity is in PTIME.

Item (2) says that, any property that is in PTIME can be checked in LFP assuming that the input structure also contains a linear order on the domain.

Why do we need a linear order?
Because LFP cannot express EVEN, which is in PTIME.

Proof: LFP⊆PTIME

Let φ be an LFP sentence, and $\boldsymbol{A}$ an input structure.

We show that we can check $\boldsymbol{A} \models \varphi$ in PTIME in the size of $\boldsymbol{A}$.

Proof: LFP⊆PTIME

Let φ be an LFP sentence, and $\boldsymbol{A}$ an input structure.

We show that we can check $\boldsymbol{A} \models \varphi$ in PTIME in the size of $\boldsymbol{A}$.

Proceed by induction on φ

- If φ is a grounded atom $R(a, b, c)$ simply check if $(a, b, c) \in R^A$.

Proof: LFP⊆PTIME

Let φ be an LFP sentence, and $\boldsymbol{A}$ an input structure.

We show that we can check $\boldsymbol{A} \models \varphi$ in PTIME in the size of $\boldsymbol{A}$.

Proceed by induction on φ

- If φ is a grounded atom $R(a, b, c)$ simply check if $(a, b, c) \in R^A$.
- If $\varphi = \exists x \psi$, then for each $a \in \mathrm{Dom}(\boldsymbol{A})$ check $\boldsymbol{A} \models \psi[a/x]$.

Proof: LFP⊆PTIME

Let φ be an LFP sentence, and $\boldsymbol{A}$ an input structure.

We show that we can check $\boldsymbol{A} \vDash \varphi$ in PTIME in the size of $\boldsymbol{A}$.

Proceed by induction on φ

- If φ is a grounded atom $R(a,b,c)$ simply check if $(a,b,c) \in R^A$.
- If $\varphi = \exists x \psi$, then for each $a \in \mathrm{Dom}(\boldsymbol{A})$ check $\boldsymbol{A} \vDash \psi[a/x]$.
- If $\varphi = \varphi_1 \vee \varphi_2$ then ....

Proof: LFP⊆PTIME

Let φ be an LFP sentence, and $\boldsymbol{A}$ an input structure.

We show that we can check $\boldsymbol{A} \models \varphi$ in PTIME in the size of $\boldsymbol{A}$.

Proceed by induction on φ

- If φ is a grounded atom $R(a, b, c)$ simply check if $(a, b, c) \in R^A$.
- If $\varphi = \exists x \psi$, then for each $a \in \text{Dom}(\boldsymbol{A})$ check $\boldsymbol{A} \models \psi[a/x]$.
- If $\varphi = \varphi_1 \vee \varphi_2$ then ....

The only interesting case is when $\varphi = \text{lfp}$.

Proof: LFP⊆PTIME

Assume $\varphi = \text{lfp}_{T,\boldsymbol{x}}\psi(a,b,c)$. It is a sentence, hence a, b, c are constants.

We compute the lfp as the limit of the Kleene sequence:

$$\begin{aligned} T^0(a,b,c) &:= \boldsymbol{F} \\ T^1(a,b,c) &:= \psi[T^0](a,b,c) \\ T^2(a,b,c) &:= \psi[T^1](a,b,c) \\ &\dots \end{aligned}$$

Each line can be computed in PTIME.

If T has arity n, then the total number of steps is $\leq |\text{Dom}(\boldsymbol{A})|^n$.

Total runtime is in PTIME.

Proof: PTIME$\subseteq$ $LFP(<)$

Consider a problem P in PTIME.

There exists a deterministic TM M that, given the encoding $\texttt{enc}(\boldsymbol{A})$ of a structure $\boldsymbol{A} \in \text{STRUCT}[\sigma]$, runs in PTIME and accepts iff $P(\boldsymbol{A})$ is true.

Proof: PTIME$\subseteq LFP(<)$

Consider a problem P in PTIME.

There exists a deterministic TM M that, given the encoding $\texttt{enc}(\boldsymbol{A})$ of a structure $\boldsymbol{A} \in \mathrm{STRUCT}[\sigma]$, runs in PTIME and accepts iff $P(\boldsymbol{A})$ is true.

We need to describe φ_M s.t. $P(\boldsymbol{A})$ is true iff $\boldsymbol{A} \vDash \varphi_M$

Proof: PTIME$\subseteq LFP(<)$

Consider a problem P in PTIME.

There exists a deterministic TM M that, given the encoding $\texttt{enc}(\boldsymbol{A})$ of a structure $\boldsymbol{A} \in \text{STRUCT}[\sigma]$, runs in PTIME and accepts iff $P(\boldsymbol{A})$ is true.

We need to describe φ_M s.t. $P(\boldsymbol{A})$ is true iff $\boldsymbol{A} \vDash \varphi_M$

Proof idea: φ_M will be a fixpoint $\text{lfp}_{T_0, T_1, (S_q)_{q \in Q}}(\psi)$.

Proof: PTIME$\subseteq$ $LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q \in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.

Proof: PTIME$\subseteq LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q\in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.
- We cannot write $\text{lfp}_{T_0, T_1, (S_q)_{q\in Q}}$, but need the fixpoint of a single relation lfp_K.

Proof: PTIME $\subseteq LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q \in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.
- We cannot write $\mathrm{lfp}_{T_0, T_1, (S_q)_{q \in Q}}$, but need the fixpoint of a single relation lfp_K.
- We encode multiple relations T_0, T_1, H, S_q using one relation K:
$$K = T_0 \times T_1 \times H \times \prod_q S_q$$

Proof: PTIME$\subseteq LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q \in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.
- We cannot write $\text{lfp}_{T_0, T_1, (S_q)_{q \in Q}}$, but need the fixpoint of a single relation lfp_K.
- We encode multiple relations T_0, T_1, H, S_q using one relation K:
$$K = T_0 \times T_1 \times H \times \prod_q S_q$$
- Problem: if, say, $S_{q_5} = \emptyset$ then we have $K = \emptyset$.

Proof: PTIME$\subseteq$ $LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q \in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.
- We cannot write $\text{lfp}_{T_0, T_1, (S_q)_{q \in Q}}$, but need the fixpoint of a single relation lfp_K.
- We encode multiple relations T_0, T_1, H, S_q using one relation K:
$$K = T_0 \times T_1 \times H \times \prod_q S_q$$
- Problem: if, say, $S_{q_5} = \varnothing$ then we have $K = \varnothing$.
To fix this, add a spurious tuple, say $S_{q_5}(\max, \max, \cdots)$, assuming the time never reaches $(\max, \max, \cdots)$. Now the fixpoint is $\text{lfp}_K(\psi)$.

Proof: PTIME$\subseteq$ $LFP(<)$ Details

- The vocabulary $T_0, T_1, H, (S_q)_{q\in Q}$ is similar to that in the proof of Fagin's theorem, i.e. arity depends on the degree of the polynomial.
- We cannot write $\text{lfp}_{T_0,T_1,(S_q)_{q\in Q}}$, but need the fixpoint of a single relation lfp_K.
- We encode multiple relations T_0, T_1, H, S_q using one relation K:
$$K = T_0 \times T_1 \times H \times \prod_q S_q$$
- Problem: if, say, $S_{q_5} = \emptyset$ then we have $K = \emptyset$.
To fix this, add a spurious tuple, say $S_{q_5}(\max, \max, \cdots)$, assuming the time never reaches $(\max, \max, \cdots)$. Now the fixpoint is $\text{lfp}_K(\psi)$.
- ψ will (1) set the tape at time 0 to $\texttt{enc}(\boldsymbol{A})$, (2) for each time step t that occurs in K, computes configuration c_{t+1} yielded by c_t.

Discussion

- $LFP(<)$ =PTIME but $LFP \subsetneq PTIME$, e.g. cannot express EVEN.
- If we add counting quantifiers to LFP, does it capture PTIME?

Discussion

- $LFP(<)$ =PTIME but $LFP \subsetneq PTIME$, e.g. cannot express EVEN.
- If we add counting quantifiers to LFP, does it capture PTIME?
  No: $C^k_{\infty\omega}$ same expressive power as k-WL, hence cannot distinguish the Cai-Führer-Immerman graphs G_k, H_k

Discussion

- $LFP(<)$ =PTIME but $LFP \subsetneq PTIME$, e.g. cannot express EVEN.
- If we add counting quantifiers to LFP, does it capture PTIME?
  No: $C^k_{\infty\omega}$ same expressive power as k-WL, hence cannot distinguish the Cai-Führer-Immerman graphs G_k, H_k
- Major open problem: is there a logic for PTIME (without order)?

Discussion

- $LFP(<)$ =PTIME but $LFP \subsetneq PTIME$, e.g. cannot express EVEN.
- If we add counting quantifiers to LFP, does it capture PTIME?
  No: $C^k_{\infty\omega}$ same expressive power as k-WL, hence cannot distinguish the Cai-Führer-Immerman graphs G_k, H_k
- Major open problem: is there a logic for PTIME (without order)?
- Variation on LFP: the partial fixpoint $\text{pfp}_k(\psi)$ does not require ψ to be monotone in K. Defined only when the Kleene sequence converges.

$$\text{FO}(\text{pfp}, <) = PSPACE$$

Discussion

- $LFP(<)$ =PTIME but $LFP \subsetneq PTIME$, e.g. cannot express EVEN.
- If we add counting quantifiers to LFP, does it capture PTIME?
  No: $C^k_{\infty\omega}$ same expressive power as k-WL, hence cannot distinguish the Cai-Führer-Immerman graphs G_k, H_k
- Major open problem: is there a logic for PTIME (without order)?
- Variation on LFP: the partial fixpoint $\text{pfp}_k(\psi)$ does not require ψ to be monotone in K. Defined only when the Kleene sequence converges.

$$\text{FO}(\text{pfp}, <) = PSPACE$$

- Abiteboul and Vianu: FO(lfp) = FO(pfp) iff PTIME=PSAPCE.

$\mathrm{FO}(\mathrm{TC}, <) = \mathrm{NLOGSPACE}$

FO with Transitive Closure

If the language LFP is too powerful, one possibility is to restrict recursion to transitive closure

For any formula $\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ with free variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x}$ where $|\boldsymbol{x}| = |\boldsymbol{y}|$, the following is a formula with free variables $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{z}$:

$$\boxed{\mathrm{TC}(\psi, \boldsymbol{x}, \boldsymbol{y})(\boldsymbol{u}, \boldsymbol{v})}$$

The semantics is:

- Consider the graph with edges $E(\boldsymbol{x}, \boldsymbol{y})$ defined by ψ,
- The formula checks whether $(\boldsymbol{u}, \boldsymbol{v})$ is in the transitive closure of E.

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:

$$\text{TC}(E(x, y), x, y)(a, b)$$

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:
$$\mathrm{TC}(E(x,y), x, y)(a, b)$$

- Check if a, b are in the same generation.

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:

$$\mathrm{TC}(E(x,y), x, y)(a,b)$$

- Check if a, b are in the same generation.

$$\exists p\, (\mathrm{TC}((E(x,y) \wedge E(u,v)), x, u, y, v))\, (p, p, a, b)$$

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:
$$\mathrm{TC}(E(x, y), x, y)(a, b)$$
- Check if a, b are in the same generation.
$$\exists p\, (\mathrm{TC}((E(x, y) \wedge E(u, v)), x, u, y, v))\, (p, p, a, b)$$
- Assume the edges have a color: $(x, c, y) \in E$ has color c. Check if a, b are connected by a path where all edges have the same color.

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:
$$\mathrm{TC}(E(x,y), x, y)(a,b)$$

- Check if a, b are in the same generation.
$$\exists p\,(\mathrm{TC}((E(x,y) \wedge E(u,v)), x, u, y, v))\,(p, p, a, b)$$

- Assume the edges have a color: $(x, c, y) \in E$ has color c. Check if a, b are connected by a path where all edges have the same color.
$$\exists c(\mathrm{TC}(E(x,c,y), x, y)(a,b))$$

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:

$$\text{TC}(E(x,y), x, y)(a,b)$$

- Check if a, b are in the same generation.

$$\exists p\,(\text{TC}((E(x,y) \wedge E(u,v)), x, u, y, v))\,(p, p, a, b)$$

- Assume the edges have a color: $(x, c, y) \in E$ has color c. Check if a, b are connected by a path where all edges have the same color.

$$\exists c(\text{TC}(E(x,c,y), x, y)(a,b))$$

FO(TC) is in stratified datalog. why?

Examples

Graph $G = (V, E)$

- Check if a, b are connected by a path:
$$\text{TC}(E(x, y), x, y)(a, b)$$
- Check if a, b are in the same generation.
$$\exists p\,(\text{TC}((E(x, y) \wedge E(u, v)), x, u, y, v))\,(p, p, a, b)$$
- Assume the edges have a color: $(x, c, y) \in E$ has color c. Check if a, b are connected by a path where all edges have the same color.
$$\exists c(\text{TC}(E(x, c, y), x, y)(a, b))$$

FO(TC) is in stratified datalog. why? Cannot express WinMove game

$FO(TC, <) = NLOGSPACE$

Recall: NLOGSPACE is the set of problems that can be accepted by a non-deterministic TM with a read-only input tape and a working tape of logarithmic size.

Theorem

$FO(TC, <) = NLOGSPACE$

FO(TC, <) = *NLOGSPACE*

Proof (sketch) We show the inclusion NLOGSPACE ⊆ FO(TC, <).

$\mathrm{FO}(\mathrm{TC}, <) = \mathit{NLOGSPACE}$

Proof (sketch) We show the inclusion $\mathrm{NLOGSPACE} \subseteq \mathrm{FO}(\mathrm{TC}, <)$.

Let M be the Turing machine, input $\boldsymbol{A} \in \mathrm{STRUCT}[\sigma]$, and $n = |\mathrm{Dom}(\boldsymbol{A})|$.

$\mathrm{FO}(\mathrm{TC}, <) = \mathit{NLOGSPACE}$

Proof (sketch) We show the inclusion $\mathrm{NLOGSPACE} \subseteq \mathrm{FO}(\mathrm{TC}, <)$.

Let M be the Turing machine, input $\boldsymbol{A} \in \mathrm{STRUCT}[\sigma]$, and $n = |\mathrm{Dom}(\boldsymbol{A})|$.

The working tape of M has size $O(\log n)$, hence the total number of possible configurations is $2^{O(\log n)} = n^{O(1)}$ (polynomial).

$\text{FO}(\text{TC}, <) = \mathit{NLOGSPACE}$

Proof (sketch) We show the inclusion $\text{NLOGSPACE} \subseteq \text{FO}(\text{TC}, <)$.

Let M be the Turing machine, input $\boldsymbol{A} \in \text{STRUCT}[\sigma]$, and $n = |\text{Dom}(\boldsymbol{A})|$.

The working tape of M has size $O(\log n)$, hence the total number of possible configurations is $2^{O(\log n)} = n^{O(1)}$ (polynomial).

Construct the graph whose nodes are the configurations, and edges are the yield relation.

$\text{FO}(\text{TC}, <) = \textit{NLOGSPACE}$

Proof (sketch) We show the inclusion $\text{NLOGSPACE} \subseteq \text{FO}(\text{TC}, <)$.

Let M be the Turing machine, input $\boldsymbol{A} \in \text{STRUCT}[\sigma]$, and $n = |\text{Dom}(\boldsymbol{A})|$.

The working tape of M has size $O(\log n)$, hence the total number of possible configurations is $2^{O(\log n)} = n^{O(1)}$ (polynomial).

Construct the graph whose nodes are the configurations, and edges are the yield relation.

M accepts $\boldsymbol{A}$ iff there exists a path from the initial configuration to some final configuration.

$\text{FO}(\text{TC}, <) = \mathit{NLOGSPACE}$

Proof (sketch) We show the inclusion $\text{NLOGSPACE} \subseteq \text{FO}(\text{TC}, <)$.

Let M be the Turing machine, input $\boldsymbol{A} \in \text{STRUCT}[\sigma]$, and $n = |\text{Dom}(\boldsymbol{A})|$.

The working tape of M has size $O(\log n)$, hence the total number of possible configurations is $2^{O(\log n)} = n^{O(1)}$ (polynomial).

Construct the graph whose nodes are the configurations, and edges are the yield relation.

M accepts $\boldsymbol{A}$ iff there exists a path from the initial configuration to some final configuration.

This can be checked in $\text{FO}(\text{TC}, <)$.

FO(Det-TC, <) = *NLOGSPACE*

The deterministic transitive closure is restricted to outdegree ≤ 1:

$$\boxed{\text{Det-TC}(\psi, \boldsymbol{x}, \boldsymbol{y})(\boldsymbol{u}, \boldsymbol{v})}$$

where ψ satisfies: $\forall \boldsymbol{x} \forall \boldsymbol{y} \forall \boldsymbol{y}' (\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{z}) \Rightarrow \boldsymbol{y} = \boldsymbol{y}')$

FO(Det-TC, <) = *NLOGSPACE*

The deterministic transitive closure is restricted to outdegree ≤ 1:

$$\text{Det-TC}(\psi, \boldsymbol{x}, \boldsymbol{y})(\boldsymbol{u}, \boldsymbol{v})$$

where ψ satisfies: $\forall \boldsymbol{x} \forall \boldsymbol{y} \forall \boldsymbol{y}'(\psi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y}', \boldsymbol{z}) \Rightarrow \boldsymbol{y} = \boldsymbol{y}')$

Theorem

FO(Det-TC, <) = LOGSPACE

Proof: Immediate.

Summary

- FO$(+, *)$ $=$ FO$(<, \text{BIT})$ = AC^0 In class
- FO(det-TC$, <)$ =LOGSPACE, and FO$(\text{TC}, <)$ =NLOGSPACE;
- $LFP(<)$ =PTIME
- FO(PartialFixpoint$, <)$ =PSPACE
- ∃SO =NP