Finite Model Theory
Lecture 15: Logic and Turing Machines

Spring 2025

Announcement

- HW4 (the last one!!) is posted and due on May 30.
- Topic for today and Wednesday: connection between logic and Turing machines.
- No lectures next week

Trakhtenbrot's Theorem

Trakhtenbrot's Theorem

A sentence φ is finitely satisfiable if there exists a structure $\boldsymbol{A}$ such that:

$$\boldsymbol{A} \models \varphi$$

Theorem (Trakhtenbrot)

Suppose the vocabulary σ has at least one relation with arity ≥ 2. Then the problem given φ check if it is finitely satisfiable is undecidable.

Thus, the set of sentences φ that are finitely satisfiable is recursively enumerable, but not decidable.

Why Binary Relation

What happens if the vocabulary σ has only unary relations $U_1, \ldots, U_m$?

Is finite satisfiability decidable?

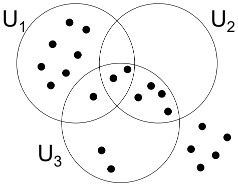
Why Binary Relation

What happens if the vocabulary σ has only unary relations $U_1, \ldots, U_m$?

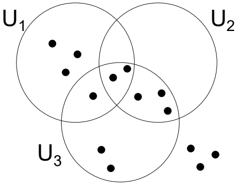
Is finite satisfiability decidable?

YES: Satisfiability and finite satisfiability coincide, and are decidable. HW3

Structure $\boldsymbol{A}$, s.t. $\boldsymbol{A} \vDash \varphi$



Small structure $\boldsymbol{B}$, $\boldsymbol{A} \sim_3 \boldsymbol{B}$



Before we prove Trakhtenbrot's theorem, we discuss 3 consequences.

Trakthenbrot's Theorem: Consequence 1

Let L be some logic; e.g. $L \subseteq FO$.

Definition

We say that L has the small model property if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ s.t. the following holds:

$$\forall \varphi \in L,\ \varphi \text{ has a model iff } \varphi \text{ has a finite model of size} \le f(|\varphi|)$$

[1]Or take $k = |\varphi|$.

Trakthenbrot's Theorem: Consequence 1

Let L be some logic; e.g. $L \subseteq FO$.

Definition

We say that L has the small model property if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ s.t. the following holds:

$$\forall \varphi \in L, \ \varphi \text{ has a model iff } \varphi \text{ has a finite model of size} \leq f(|\varphi|)$$

Example: if σ is unary, then FO has the small model property, where $f(\varphi) = (k+1)^{2^n}$, where k = number of variables[1] in φ.

[1] Or take $k = |\varphi|$.

Trakthenbrot's Theorem: Consequence 1

Corollary

If the vocabulary σ has at least one relation with arity ≥ 2, then FO does not have the small model property. above.

Trakthenbrot's Theorem: Consequence 1

Corollary

If the vocabulary σ has at least one relation with arity ≥ 2, then FO does not have the small model property. above.

Proof: Assume the contrary, that $f : \mathbb{N} \to \mathbb{N}$ exists such that φ has a model iff φ has a finite model of size $\leq f(|\varphi|)$.

Then we can check finite satisfiability as follows. Given φ, compute $n = f(|\varphi|)$, and try out all structures of size $\leq n$:

- If one of the structures is a model then answer YES.
- Otherwise answer NO.

Trakthenbrot's Theorem: Consequence 2

Denote $\varphi \equiv_{\text{fin}} \psi$ if φ, ψ are equivalent on all finite structures:

Corollary

If the vocabulary σ has at least one relation with arity ≥ 2, then the following problem is undecidable:

Given two sentences φ, ψ, check whether $\varphi \equiv_{fin} \psi$

Trakthenbrot's Theorem: Consequence 2

Denote $\varphi \equiv_{\text{fin}} \psi$ if φ, ψ are equivalent on all finite structures:

Corollary

If the vocabulary σ has at least one relation with arity ≥ 2, then the following problem is undecidable:

Given two sentences φ, ψ, check whether $\varphi \equiv_{fin} \psi$

Proof

Reduce it to UNSAT. Assume we have an oracle for $\varphi \equiv_{\text{fin}} \psi$.
Then we can check UNSAT by checking if $\varphi \equiv_{\text{fin}} \boldsymbol{F}$.

Trakhtenbrot's Theorem: Consequence 3

We say that φ is finitely valid, $\models_{\text{fin}} \varphi$, if it holds in every finite model $\boldsymbol{A}$.

Corollary

The set of finitely valid sentences it not recursively enumerable.

Trakhtenbrot's Theorem: Consequence 3

We say that φ is finitely valid, $\models_{\text{fin}} \varphi$, if it holds in every finite model $\boldsymbol{A}$.

Corollary

The set of finitely valid sentences it not recursively enumerable.

Proof φ is finitely satisfiable iff $\neg\varphi$ is not finitely valid.

Assume finitely valid sentences are r.e.
Then we could check if φ is finitely satisfiable by enumerating in parallel:

- All finite structures $\boldsymbol{A}$, checking if $\boldsymbol{A} \models \varphi$
- All finitely valid sentences, checking if $\neg\varphi$ shows up.

Trakhtenbrot's Theorem: Consequence 3

We say that φ is finitely valid, $\models_{\text{fin}} \varphi$, if it holds in every finite model $\boldsymbol{A}$.

Corollary

The set of finitely valid sentences it not recursively enumerable.

Proof φ is finitely satisfiable iff $\neg\varphi$ is not finitely valid.

Assume finitely valid sentences are r.e.
Then we could check if φ is finitely satisfiable by enumerating in parallel:

- All finite structures $\boldsymbol{A}$, checking if $\boldsymbol{A} \models \varphi$
- All finitely valid sentences, checking if $\neg\varphi$ shows up.

Corollary (Finiteness not axiomatizable)

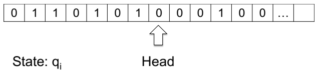
There is no r.e. set of axioms Σ such that $\Sigma \vdash \varphi$ iff $\models_{fin} \varphi$.

Review: Turing Machines

Review: Turing Machines Basics

$M = (Q, \Sigma, \Delta, q_0, Q_F)$ where:

- $Q = \{q_0, q_1, \ldots, q_m\}$ are the states;
  q_0 is the initial state;
  $Q_F \subseteq Q$ are the final states.
- Σ is the tape alphabet; we take $\Sigma = \{0, 1\}$
- $\Delta \subseteq Q \times \Sigma \times \Sigma \times \{\text{Left}, \text{Right}\} \times Q$ are the transitions.



Review: Turing Machines Basics

A configuration is a triple $c = (w, h, q)$ where:

- $w \subseteq \Sigma^*$ is a tape content.
- $h \in \mathbb{N}$ is the head position.
- $q \in Q$ is a state.

The usual notation of a configuration is: $\boxed{uqv}$,
where $uv = w$ and $|u| = h$.

Review: Turing Machine Basics

A configuration c yields a configuration c' if one of the following two conditions hold:

- $c = uqav$, $c' = ubq'v$, and $(q, a, b, \text{Right}, q') \in \Delta$, or
- $c = uaqv$, $c' = uq'bv$, and $(q, a, b, \text{Left}, q') \in \Delta$.

Review: Turing Machine Basics

A configuration c yields a configuration c' if one of the following two conditions hold:

- $c = uqav$, $c' = ubq'v$, and $(q, a, b, \text{Right}, q') \in \Delta$, or
- $c = uaqv$, $c' = uq'bv$, and $(q, a, b, \text{Left}, q') \in \Delta$.

An computation history is a sequence $\boldsymbol{C} = c_1, c_2, \ldots, c_T$ where:

- c_1 is the initial configuration: $c_1 = q_0$.
- c_{i-1} yields c_i, for $i = 2, T$.

Review: Turing Machine Basics

A configuration c yields a configuration c' if one of the following two conditions hold:

- $c = uqav$, $c' = ubq'v$, and $(q, a, b, \text{Right}, q') \in \Delta$, or
- $c = uaqv$, $c' = uq'bv$, and $(q, a, b, \text{Left}, q') \in \Delta$.

An computation history is a sequence $\boldsymbol{C} = c_1, c_2, \ldots, c_T$ where:

- c_1 is the initial configuration: $c_1 = q_0$.
- c_{i-1} yields c_i, for $i = 2, T$.

A computation history is accepting if it ends in a final configuration: $c_T = uqv$, with $q \in Q_F$.

Proof of Trakhtenbrot's Theorem

Proof of Trakhtenbrot's Theorem

"Given φ check if it is finitely satisfiable" is undecidable.

By reduction from the Halting Problem:

- Given a Turing Machine M, does M halt on the empty input?

The proof consist of the following: given M we will construct a sentence φ_M s.t. M halts iff φ_M is finitely satisfiable.

Proof Plan

M halts iff

$\exists \boldsymbol{C}$, $\boldsymbol{C}$ is an accepting computation of M.

Proof Plan

M halts iff

$$\boxed{\exists \boldsymbol{C},\ \boldsymbol{C} \text{ is an accepting computation of } M.}$$

φ is finitely satisfiable iff

$$\boxed{\exists \boldsymbol{A} \text{ such that } \boldsymbol{A} \models \varphi.}$$

Proof Plan

M halts iff

$$\boxed{\exists \boldsymbol{C},\ \boldsymbol{C} \text{ is an accepting computation of } M.}$$

φ is finitely satisfiable iff

$$\boxed{\exists \boldsymbol{A} \text{ such that } \boldsymbol{A} \models \varphi.}$$

This suggests the proof plan:

- Computation history $\boldsymbol{C}$ = structure $\boldsymbol{A}$.
- $\boldsymbol{C}$ is an accepting computation iff $\boldsymbol{A}$ is a model of φ.

Proof Details

Fix a Turing Machine M.

- Describe a binary vocabulary σ_M and sentence φ_M whose models correspond precisely to accepting computations of M.
- Later: describe an FO encoding of σ_M and φ_M into the language of graphs $\sigma = (E)$.

Proof Details

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

Proof Details

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

Intended meaning:

- $<$ is a total order

Proof Details

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

Intended 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.

Proof Details

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

Intended 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.

Proof Details

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

Intended 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 Turning Machine is is stated q at time t.

Proof Details

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

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.

Proof Details

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

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.

Proof Details

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

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 $t+1$

Proof Details: General Consistency

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

- $<$ is a total order.
- Exactly one tape symbol:
  $\forall t, \forall p (T_0(t,p) \vee T_1(t,p)) \wedge \neg(T_0(t,p) \wedge T_1(t,p))$
- Exactly one head position at each time: ...
- Exactly one state at each time: ...

Proof Details: Initial Configuration

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

At time $t = \min$, the TM is in the initial configuration:

$$\forall p T_0(\min, p) \wedge H(\min, \min) \wedge S_{q_0}(\min)$$

Note that we can name min by $\exists x \neg \exists y (y < x)$; similarly max.

Proof Details: Final Configuration

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

At time $t = \max$, the TM is in the final configuration:

$$\bigvee_{q\in Q_F} S_q(\max)$$

Proof Details: All Transitions are Correct

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

Each transition from t to $t+1$ corresponds to one valid $\delta \in \Delta$:

$$\forall t(t < \max \to \bigvee_{\delta\in\Delta} \mathrm{CHECK}_\delta(t))$$

Proof Details: All Transitions are Correct (Detail)

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

Example transition: $\delta = (q_5, 1, 0, \text{Left}, q_3)$
("If in state q_5 and the tape is 1, then write 0, move Left, enter q_3")

$\text{CHECK}_\delta(t) =$	$S_{q_5}(t)$	Check we are in q_5
	$\wedge \forall s(\neg H(t,s) \to (T_0(t,s) \leftrightarrow T_0(t+1,s)))$	Leave non-head symbols unchaged
	$\wedge \forall s(H(t,s) \to T_1(t,s) \wedge T_0(t+1,s))$	the head was 1 set it to 0
	$\wedge H(t+1, s-1)$	move to the left
	$\wedge S_{q_3}(t+1)$	enter q_3

Proof Details: Finalizing the Proof

Claim: φ_M is satisfiable iff M terminates.

Proof Details: Finalizing the Proof

Claim: 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 construct a structure $\boldsymbol{A}$ as follows:

Proof Details: Finalizing the Proof

Claim: 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 construct a structure $\boldsymbol{A}$ as follows:
  - The domain is $[T]$, and $<$ is the standard order.
  - The relations T_0, T_1, H, S_q are defined according to $\boldsymbol{C}$.

  Then $\boldsymbol{A} \models \varphi_M$.

Proof Details: Finalizing the Proof

Claim: 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 construct a structure $\boldsymbol{A}$ as follows:
  - The domain is $[T]$, and $<$ is the standard order.
  - The relations T_0, T_1, H, S_q are defined according to $\boldsymbol{C}$.

  Then $\boldsymbol{A} \vDash \varphi_M$.

- If $\boldsymbol{A} \vDash \varphi_M$, then construct an accepting computation $c_1, \ldots, c_T$:

Proof Details: Finalizing the Proof

Claim: 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 construct a structure $\boldsymbol{A}$ as follows:
  - The domain is $[T]$, and $<$ is the standard order.
  - The relations T_0, T_1, H, S_q are defined according to $\boldsymbol{C}$.

  Then $\boldsymbol{A} \models \varphi_M$.

- If $\boldsymbol{A} \models \varphi_M$, then construct an accepting computation $c_1, \ldots, c_T$:
  - Set $T \stackrel{\text{def}}{=} |\text{Dom}(\boldsymbol{A})|$.
  - Configuration c_i defined based on $T_0(i, \cdot), T_1(i, \cdot), H(i, \cdot), (S_q(i))_q$.

Proof Details: Finalizing the Proof

Claim: 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 construct a structure $\boldsymbol{A}$ as follows:
  - The domain is $[T]$, and $<$ is the standard order.
  - The relations T_0, T_1, H, S_q are defined according to $\boldsymbol{C}$.

  Then $\boldsymbol{A} \models \varphi_M$.

- If $\boldsymbol{A} \models \varphi_M$, then construct an accepting computation $c_1, \ldots, c_T$:
  - Set $T \stackrel{\text{def}}{=} |\text{Dom}(\boldsymbol{A})|$.
  - Configuration c_i defined based on $T_0(i, \cdot), T_1(i, \cdot), H(i, \cdot), (S_q(i))_q$.

  Since $\boldsymbol{A} \models \varphi_M$ it follows that $\boldsymbol{C}$ is an accepting computation for M.

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\mathrm{Dom}(A)|$?

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\text{Dom}(A)|$? The number of time steps needed by M to terminate.

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\text{Dom}(A)|$? The number of time steps needed by M to terminate.
- Is $\boldsymbol{A}$ unique?

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\text{Dom}(A)|$? The number of time steps needed by M to terminate.
- Is $\boldsymbol{A}$ unique? Not necessarily. But it is unique when M is deterministic.

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\mathrm{Dom}(A)|$? The number of time steps needed by M to terminate.
- Is $\boldsymbol{A}$ unique? Not necessarily. But it is unique when M is deterministic.
- Is succ enough, or do we need $<$?

Discussion

- A structure s.t. $\boldsymbol{A} \vDash \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\mathrm{Dom}(A)|$? The number of time steps needed by M to terminate.
- Is $\boldsymbol{A}$ unique? Not necessarily. But it is unique when M is deterministic.
- Is $\mathtt{succ}$ enough, or do we need $<$? ~~We need $<$ because $\mathtt{succ}$ is not finitely axiomatizable.~~ $\mathtt{succ}$ is enough. (Needs axioms to say that indegree and outdegree of each element is 1, except for two elements that have indegree 0 and outdegree 0 respectively.)

Discussion

- A structure s.t. $\boldsymbol{A} \models \varphi_M$ is precisely an accepting computation $\boldsymbol{C}$ of the Turing Machine M.
- how large is $|\mathrm{Dom}(A)|$? The number of time steps needed by M to terminate.
- Is $\boldsymbol{A}$ unique? Not necessarily. But it is unique when M is deterministic.
- Is succ enough, or do we need $<$? ~~We need $<$ because succ is not finitely axiomatizable.~~ succ is enough. (Needs axioms to say that indegree and outdegree of each element is 1, except for two elements that have indegree 0 and outdegree 0 respectively.)
- We still need to reduce the vocabulary σ_M to a vocabulary with a single binary relation E. We do this next

FO Reductions

FO Reduction

Let $\sigma = \{S_1, \ldots, S_m\}, \tau = \{T_1, \ldots, T_n\}$ be two relational vocabularies.

FO Reduction

Let $\sigma = \{S_1, \ldots, S_m\}, \tau = \{T_1, \ldots, T_n\}$ be two relational vocabularies.

A query from σ to τ is a function $Q : \text{STRUCT}[\sigma] \to \text{STRUCT}[\tau]$.

FO Reduction

Let $\sigma = \{S_1, \ldots, S_m\}, \tau = \{T_1, \ldots, T_n\}$ be two relational vocabularies.

A query from σ to τ is a function $Q : \text{STRUCT}[\sigma] \to \text{STRUCT}[\tau]$.

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

FO Reduction

Let $\sigma = \{S_1, \ldots, S_m\}, \tau = \{T_1, \ldots, T_n\}$ be two relational vocabularies.

A query from σ to τ is a function $Q : \text{STRUCT}[\sigma] \to \text{STRUCT}[\tau]$.

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

A First Order Query Q consists of n formulas, $Q = (q_1, \ldots, q_n)$, where each q_j has arity(T_j) free variables; it defines the mapping $Q(\boldsymbol{A}) \stackrel{\text{def}}{=} \boldsymbol{B}$ where:

$$B \stackrel{\text{def}}{=} A \qquad \text{same domain}$$

$$\forall j : \quad T_j^B \stackrel{\text{def}}{=} \{\boldsymbol{b} \mid \boldsymbol{A} \models q_j(\boldsymbol{b})\}$$

FO Reduction

Let $\sigma = \{S_1, \ldots, S_m\}, \tau = \{T_1, \ldots, T_n\}$ be two relational vocabularies.

A query from σ to τ is a function $Q : \text{STRUCT}[\sigma] \to \text{STRUCT}[\tau]$.

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

A First Order Query Q consists of n formulas, $Q = (q_1, \ldots, q_n)$, where each q_j has arity(T_j) free variables; it defines the mapping $Q(\boldsymbol{A}) \stackrel{\text{def}}{=} \boldsymbol{B}$ where:

$$B \stackrel{\text{def}}{=} A \qquad \text{same domain}$$

$$\forall j : \quad T_j^B \stackrel{\text{def}}{=} \{\boldsymbol{b} \mid \boldsymbol{A} \models q_j(\boldsymbol{b})\}$$

Q maps problems on STRUCT$[\tau]$ to problems on STRUCT$[\sigma]$ ("in reverse"): $P \mapsto P \circ Q$, i.e. $\hat{P}(\boldsymbol{A}) \stackrel{\text{def}}{=} P(Q(\boldsymbol{A}))$.

FO Reduction

Query STRUCT$[\sigma] \to$ STRUCT$[\tau]$
(Problems on STRUCT$[\tau]$) $\to$ (Problems on STRUCT$[\sigma]$)

Definition

A First Order Reduction is an FO query Q from σ to τ.
It "reduces" a problem P' on τ from the problem $P \stackrel{\text{def}}{=} P' \circ Q$ on σ.

Obviously, P' is at least as hard as P.

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

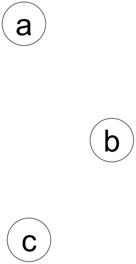
Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$



R

a	b
a	c
c	a

S

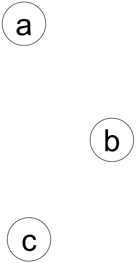
a	c
c	c

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$



R

a	b
a	c
c	a

S

a	c
c	c

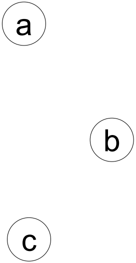
Use small gadgets for R and S

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$



R

a	b
a	c
c	a

S

a	c
c	c

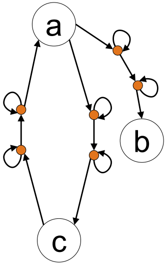
Use small gadgets for R and S

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$

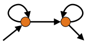


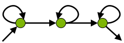
R

a	b
a	c
c	a

S

a	c
c	c





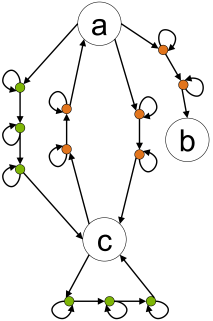
Use small gadgets for R and S

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$



R

a	b
a	c
c	a

S

a	c
c	c

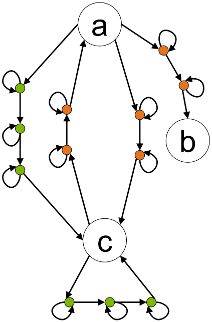
Use small gadgets for R and S

Every Structure is FO-Reducible to a Graph

$\sigma = \{E\}$ a graph.

τ = any vocabulary. For simplicity, assume $\tau = \{R(\cdot,\cdot), S(\cdot,\cdot)\}$.

Question: Given a τ-structure $\boldsymbol{A} = (R^A, S^A)$, encode it as a graph G s.t. you can decode it: $R^A = Q_1(\boldsymbol{G})$, $S^A = Q_2(\boldsymbol{G})$



R

a	b
a	c
c	a

S

a	c
c	c

Use small gadgets for R and S

The query (1) first checks that G is a correct encoding how?, then (2) decodes R and S how?

Discussion

Main take away: we can encode an accepting computation of a Turing Machine as a structure, and verify its correctness using a sentence.

Next lecture we use small variants of this encoding to prove:

- Fagins's Theorem: ∃SO captures NP.
- Immerman and Vardi's Theorem: $LFP(<)$ ("least fixpoint logic over ordered structures") captures PTIME
- Church-Turing's Undecidability Theorem: validity (in all models) is undecidable.
- Gödel's Incompleteness Theorem: $(\mathbb{N}, +, *)$ is not axiomatizable.