Database Theory

VU 181.140, WS 2024

4. Trakhtenbrot's Theorem

Matthias Lanzinger
(Based on slides of Reinhard Pichler)

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

29 October, 2024

TU WIEN Informatics

Outline

Perfect Query Optimization

A legitimate question:

Question

Given a query Q in RA, does there exist at least one database $\mathcal{A}$ such that $Q(\mathcal{A}) \neq \emptyset$?

- If there is no such database, then the query Q makes no sense and we can directly replace it by the empty result.
- Could save much run-time!
- We shall show that this problem is undecidable!

Turing Machines

Turing machines are a formal model of algorithms to solve problems:

Definition

A Turing machine is a quadruple $M = (Q, \Sigma, \delta, q_0)$ with a finite set of states Q, a finite set of symbols Σ (alphabet of M) so that $\sqcup, \triangleright \in \Sigma$, a transition function δ:

$$Q \times \Sigma \to (Q \cup \{q_{yes}, q_{no}\}) \times \Sigma \times \{+1, -1, 0\},$$

an accepting state q_{yes}, a rejecting state q_{no},
and R/W head directions: $+1$ (right), -1 (left), and 0 (stay).

- Function δ is the “program” of the machine.

- Function δ is the "program" of the machine.
- For the current state $q \in Q$ and the current symbol $\sigma \in \Sigma$,
  - $\delta(q, \sigma) = (p, \rho, D)$ where p is the new state,
  - ρ is the symbol to be overwritten on σ, and
  - $D \in \{+1, -1, 0\}$ is the direction in which the R/W head will move.

- Function δ is the "program" of the machine.
- For the current state $q \in Q$ and the current symbol $\sigma \in \Sigma$,
  - $\delta(q, \sigma) = (p, \rho, D)$ where p is the new state,
  - ρ is the symbol to be overwritten on σ, and
  - $D \in \{+1, -1, 0\}$ is the direction in which the R/W head will move.
- For any states p and q, $\delta(q, \triangleright) = (p, \rho, D)$ with $\rho = \triangleright$ and $D = +1$.

  In other words: The delimiter $\triangleright$ is never overwritten by another symbol, and the R/W head never moves off the left end of the tape.

- Function δ is the "program" of the machine.
- For the current state $q \in Q$ and the current symbol $\sigma \in \Sigma$,
  - $\delta(q, \sigma) = (p, \rho, D)$ where p is the new state,
  - ρ is the symbol to be overwritten on σ, and
  - $D \in \{+1, -1, 0\}$ is the direction in which the R/W head will move.
- For any states p and q, $\delta(q, \triangleright) = (p, \rho, D)$ with $\rho = \triangleright$ and $D = +1$.

  In other words: The delimiter $\triangleright$ is never overwritten by another symbol, and the R/W head never moves off the left end of the tape.
- The machine starts as follows:

  (i) the initial state of $M = (Q, \Sigma, \delta, q_0)$ is q_0,

  (ii) the tape is initialized to the infinite string $\triangleright I \sqcup \sqcup \ldots$, where I is a finitely long string in $(\Sigma - \{\sqcup\})^*$ (I is the *input* of the machine) and

  (iii) the R/W head points to $\triangleright$.

- Function δ is the "program" of the machine.
- For the current state $q \in Q$ and the current symbol $\sigma \in \Sigma$,
  - $\delta(q, \sigma) = (p, \rho, D)$ where p is the new state,
  - ρ is the symbol to be overwritten on σ, and
  - $D \in \{+1, -1, 0\}$ is the direction in which the R/W head will move.
- For any states p and q, $\delta(q, \triangleright) = (p, \rho, D)$ with $\rho = \triangleright$ and $D = +1$.

  In other words: The delimiter $\triangleright$ is never overwritten by another symbol, and the R/W head never moves off the left end of the tape.
- The machine starts as follows:

  (i) the initial state of $M = (Q, \Sigma, \delta, q_0)$ is q_0,

  (ii) the tape is initialized to the infinite string $\triangleright I \sqcup \sqcup \ldots$, where I is a finitely long string in $(\Sigma - \{\sqcup\})^*$ (I is the *input* of the machine) and

  (iii) the R/W head points to $\triangleright$.
- The machine halts iff q_{yes}, or q_{no} has been reached.
  If q_{yes} has been reached, then the machine accepts the input.
  If q_{no} has been reached, then the machine rejects the input.

Halting Problem

HALTING

INSTANCE: A Turing machine M, an input string I.
QUESTION: Does M halt on I?

Theorem

HALTING *is undecidable, i.e. there does not exist a Turing machine that decides* **HALTING**.

Halting Problem

HALTING

INSTANCE: A Turing machine M, an input string I.
QUESTION: Does M halt on I?

Theorem

HALTING *is undecidable, i.e. there does not exist a Turing machine that decides* **HALTING**.

Undecidability applies already to the following variant of **HALTING**:

HALTING-ϵ

INSTANCE: A Turing machine M.
QUESTION: Does M halt on the empty string ϵ, i.e. does M reach q_{yes}, or q_{no} when run on the initial tape contents $\triangleright \sqcup \sqcup \ldots$?

Trakhtenbrot's Theorem

Theorem (Trakhtenbrot's Theorem, 1950)

Finite Satisfiability of First-Order Logic is undecidable, i.e.: given an FO formula φ, it is undecidable if φ has a finite model.

Remark. The problem remains undecidable even if we restrict the formulae to a relational vocabulary with a single binary relation symbol.

Trakhtenbrot's Theorem

Theorem (Trakhtenbrot's Theorem, 1950)

Finite Satisfiability of First-Order Logic is undecidable, i.e.: given an FO formula φ, it is undecidable if φ has a finite model.

Remark. *The problem remains undecidable even if we restrict the formulae to a relational vocabulary with a single binary relation symbol.*

This theorem rules out perfect query optimization. Translated into database terminology, it reads:

Theorem

For a database schema σ with at least one binary relation, it is undecidable whether a Boolean FO or RA query Q over σ has a non-empty answer for at least one database.

Idea to prove Trakhtenbrot's Theorem

- Define a relational signature σ suitable for encoding finite computations of a TM.
- Given an arbitrary TM M, we construct an FO formula φ_M "encoding" the computation of M and a halting condition, such that:

> φ_M has a finite model iff M halts on ϵ.

- The undecidability of **HALTING**-ϵ together with the reduction proves Trakhtenbrot's Theorem!

Proof of Trakhtenbrot's Theorem

Assume a machine $M = (Q, \Sigma, \delta, q_{start})$.

Simplifying assumptions:

- σ may have several unary and binary relations
  Exercise. We could easily encode them into a single binary relation.
- Tape alphabet of M is $\Sigma = \{0, 1, \triangleright, \sqcup\}$
  - Can always be obtained by simple binary encoding, e.g., let $\Sigma = \{a_1, \ldots, a_k\}$ with $k \leq 8$, then we use the following encoding:
    $a_0 \to 000$, $a_1 \to 001$, $a_2 \to 010$, $a_3 \to 011$, etc.

We use the following relations:

- Binary $<$ will encode a linear order (as usual, we'll write $x < y$ instead of $<(x, y)$). The elements of this linear order will be used to simulate both time instants and tape positions ($=$ cell numbers).

We use the following relations:

- Binary $<$ will encode a linear order (as usual, we'll write $x < y$ instead of $<(x, y)$). The elements of this linear order will be used to simulate both time instants and tape positions (= cell numbers).
- Unary *Min* will denote the smallest element of $<$.
  Note: instead of a relation *Min* we can use a constant *min*.
- Binary *Succ* will encode the successor relation w.r.t. the linear order.

We use the following relations:

- Binary $<$ will encode a linear order (as usual, we'll write $x < y$ instead of $<(x, y)$). The elements of this linear order will be used to simulate both time instants and tape positions (= cell numbers).
- Unary *Min* will denote the smallest element of $<$.
  Note: instead of a relation *Min* we can use a constant *min*.
- Binary *Succ* will encode the successor relation w.r.t. the linear order.
- Binary $T_0, T_1, T_\triangleright, T_\sqcup$ are tape predicates: $T_\alpha(p, t)$ indicates that cell number p at time t contains α.

We use the following relations:

- Binary $<$ will encode a linear order (as usual, we'll write $x < y$ instead of $<(x, y)$). The elements of this linear order will be used to simulate both time instants and tape positions (= cell numbers).
- Unary *Min* will denote the smallest element of $<$.
  Note: instead of a relation *Min* we can use a constant *min*.
- Binary *Succ* will encode the successor relation w.r.t. the linear order.
- Binary $T_0, T_1, T_\triangleright, T_\sqcup$ are tape predicates: $T_\alpha(p, t)$ indicates that cell number p at time t contains α.
- Binary H will store the head position: $H(p, t)$ indicates that the R/W head at time t is at position p (i.e., at cell number p).

We use the following relations:

- Binary $<$ will encode a linear order (as usual, we'll write $x < y$ instead of $<(x, y)$). The elements of this linear order will be used to simulate both time instants and tape positions ($=$ cell numbers).
- Unary *Min* will denote the smallest element of $<$.
  Note: instead of a relation *Min* we can use a constant *min*.
- Binary *Succ* will encode the successor relation w.r.t. the linear order.
- Binary $T_0, T_1, T_\triangleright, T_\sqcup$ are tape predicates: $T_\alpha(p, t)$ indicates that cell number p at time t contains α.
- Binary H will store the head position: $H(p, t)$ indicates that the R/W head at time t is at position p (i.e., at cell number p).
- Binary S will store the state: $S(q, t)$ indicates that at time instant t the machine is in state q.

We let φ_M be the conjunction $\varphi_M = \varphi_< \land \varphi_{Min} \land \varphi_{comp}$ that is explained next:

We let φ_M be the conjunction $\varphi_M = \varphi_< \wedge \varphi_{Min} \wedge \varphi_{comp}$ that is explained next:

- $<$ must be a strict linear order (a total, transitive, antisymmetric, irreflexive relation). Thus $\varphi_<$ is the conjunction of:

$$\forall x, y.(x \neq y \leftrightarrow (x < y \vee y < x))$$

$$\forall x, y, z.((x < y \wedge y < z) \rightarrow x < z)$$

$$\forall x, y.\neg(x < y \wedge y < x)$$

We axiomatize the successor relation based on $<$ as follows:

$$\forall x, y.(Succ(x, y) \leftrightarrow (x < y) \wedge \neg\exists z.(x < z \wedge z < y))$$

- *Min* must contain the minimal element of $<$. Thus φ_{Min} is:

$$\forall x, y.(Min(x) \leftrightarrow (x = y \vee x < y))$$

- The formula φ_{comp} is defined as

$$\varphi_{comp} \equiv \exists y_0, y_1, ..., y_k(\varphi_{states} \wedge \varphi_{rest}),$$

where each variable y_i corresponds to the state q_i of M (we assume the TM has $k+1$ states), and

$$\varphi_{states} \equiv \bigwedge_{0 \leq i < j \leq k} y_i \neq y_j.$$

Intuitively, using the $\exists y_0, y_1, ..., y_k$ prefix and φ_{states} we associate to each state of M a distinct domain element.

- The formula φ_{rest} is the conjunction of several formulas defined next (R1-R6) to describe the behaviour of M.

(R1) Formula defining the initial configuration of M with $\triangleright \sqcup \sqcup \ldots$ on its input tape.

- At time instant 0 the tape has $\triangleright$ in the first cell of the tape:

$$\forall p.(Min(p) \rightarrow T_{\triangleright}(p, p))$$

- All other cells contain $\sqcup$ at time 0:

$$\forall p, t.((Min(t) \land \neg Min(p)) \rightarrow T_{\sqcup}(p, t))$$

- The head is initially at the start position 0:

$$\forall t(Min(t) \rightarrow H(t, t))$$

- The machine is initially in state q_{start}:

$$\forall t(Min(t) \rightarrow S(y_{start}, t))$$

(R2) Formulas stating that in every configuration, each cell of the tape contains exactly one symbol:

$$\forall p, t.(T_0(p,t) \vee T_1(p,t) \vee T_{\triangleright}(p,t) \vee T_{\sqcup}(p,t)),$$

$$\forall p, t.(\neg T_{\sigma_1}(p,t) \vee \neg T_{\sigma_2}(p,t)), \qquad \text{for all } \sigma_1 \neq \sigma_2 \in \Sigma$$

(R2) Formulas stating that in every configuration, each cell of the tape contains exactly one symbol:

$$\forall p, t.(T_0(p,t) \vee T_1(p,t) \vee T_{\triangleright}(p,t) \vee T_{\sqcup}(p,t)),$$

$$\forall p, t.(\neg T_{\sigma_1}(p,t) \vee \neg T_{\sigma_2}(p,t)), \qquad \text{for all } \sigma_1 \neq \sigma_2 \in \Sigma$$

(R3) A formula stating that at any time the machine is in exactly one state:

$$\forall t.((\bigvee_{0 \leq i \leq k} S(y_i, t)) \wedge \bigwedge_{0 \leq i < j \leq k} \neg(S(y_i, t) \wedge S(y_j, t)))$$

(R2) Formulas stating that in every configuration, each cell of the tape contains exactly one symbol:

$$\forall p, t.(T_0(p,t) \vee T_1(p,t) \vee T_{\triangleright}(p,t) \vee T_{\sqcup}(p,t)),$$

$$\forall p, t.(\neg T_{\sigma_1}(p,t) \vee \neg T_{\sigma_2}(p,t)), \qquad \text{for all } \sigma_1 \neq \sigma_2 \in \Sigma$$

(R3) A formula stating that at any time the machine is in exactly one state:

$$\forall t.((\bigvee_{0 \leq i \leq k} S(y_i, t)) \wedge \bigwedge_{0 \leq i < j \leq k} \neg(S(y_i, t) \wedge S(y_j, t)))$$

(R4) A formula stating that at any time the head is at exactly one position:

(R2) Formulas stating that in every configuration, each cell of the tape contains exactly one symbol:

$$\forall p, t.(T_0(p,t) \lor T_1(p,t) \lor T_{\triangleright}(p,t) \lor T_{\sqcup}(p,t)),$$

$$\forall p, t.(\neg T_{\sigma_1}(p,t) \lor \neg T_{\sigma_2}(p,t)), \qquad \text{for all } \sigma_1 \neq \sigma_2 \in \Sigma$$

(R3) A formula stating that at any time the machine is in exactly one state:

$$\forall t.((\bigvee_{0 \leq i \leq k} S(y_i, t)) \land \bigwedge_{0 \leq i < j \leq k} \neg(S(y_i, t) \land S(y_j, t)))$$

(R4) A formula stating that at any time the head is at exactly one position:

$$\forall t.\big([\exists p.(H(p,t)] \land \forall p, p'.[H(p,t) \land H(p',t) \rightarrow p = p']\big)$$

(R5) Formulas describing the transitions. In particular, for each tuple $(q_1, \sigma_1, q_2, \sigma_2, D)$ such that $\delta(q_1, \sigma_1) = (q_2, \sigma_2, D)$, we have the formula:

$$
\begin{aligned}
\forall p, t \Big((H(p,t) \land T_{\sigma_1}(p,t) \land S(y_1, t)) \to \exists p', t'. \big(& \mathit{FollowTo}(p,p') \land \mathit{Succ}(t,t') \land \\
& H(p',t') \land S(y_2,t') \land T_{\sigma_2}(p,t') \land \\
& \forall r. (r \neq p \land T_0(r,t) \to T_0(r,t')) \land \\
& \forall r. (r \neq p \land T_1(r,t) \to T_1(r,t')) \land \\
& \forall r. (r \neq p \land T_\triangleright(r,t) \to T_\triangleright(r,t')) \land \\
& \forall r. (r \neq p \land T_\sqcup(r,t) \to T_\sqcup(r,t'))) \big) \Big)
\end{aligned}
$$

where:

$$
\mathit{FollowTo}(p,p') \equiv \begin{cases} \mathit{Succ}(p,p') & \text{if } D = +1, \\ \mathit{Succ}(p',p) & \text{if } D = -1, \\ p = p' & \text{if } D = 0. \end{cases}
$$

(R6) A formula φ_{halt} saying that M halts on input I:

$$\exists t.(S(y_{yes}, t) \vee S(y_{no}, t)).$$

This completes the description of the formula φ_M, which faithfully describes the computation of M on the empty word ϵ.

By construction of φ_M, we have:

φ_M has a finite model iff M halts on ϵ

This completes the reduction from **HALTING**-ϵ and proves Trakhtenbrot's Theorem.

Further Consequences of Trakhtenbrot's Theorem

The following problems can now be easily shown undecidable:

- checking whether an FO query is domain independent,
- checking query containment of two FO (or RA) queries;
  recall that this means: $\forall \mathcal{A} : Q_1(\mathcal{A}) \subseteq Q_2(\mathcal{A})$;
- checking equivalence of two FO (or RA) queries.

Proof Sketches

Undecidability of Domain Independence

By reduction from finite unsatisfiability:
Let φ be an arbitrary instance of finite unsatisfiability.
Construct the following instance ψ of Domain Independence:
w.l.o.g. let x be a variable not occurring in φ;
then we set $\psi = \exists x. \neg R(x) \wedge \varphi$.

Proof Sketches

Undecidability of Domain Independence

By reduction from finite unsatisfiability:
Let φ be an arbitrary instance of finite unsatisfiability.
Construct the following instance ψ of Domain Independence:
w.l.o.g. let x be a variable not occurring in φ;
then we set $\psi = \exists x. \neg R(x) \wedge \varphi$.

Undecidability of Query Containment and Query Equivalence

By reduction from finite unsatisfiability:
Let φ be an arbitrary instance of finite unsatisfiability; w.l.o.g., suppose that φ has no free variables (i.e., simply add existential quantifiers).
Let χ be a trivially unsatisfiable query, e.g., $\chi = (\exists x)(R(x) \wedge \neg R(x))$.
Define the instance (φ, χ) of Query Containment or Query Equivalence.

Finite vs. Infinite Domain

Motivation

Recall the following property of the formula φ_M in the proof of Trakhtenbrot's Theorem: φ_M has a finite model iff M halts on ϵ.

Question. What about arbitrary models (with possibly infinite domain)?

It turns out that the ("$\Rightarrow$" direction of the) equivalence
"φ_M has an arbitrary model iff M halts on ϵ"
does not hold. Indeed, suppose that M does not terminate on input ϵ.
Then φ_M has the following (infinite) model:

- Choose as domain D the natural numbers $\{0, 1, \ldots, \}$ plus some additional element a.
- Choose the ordering such that a is greater than all natural numbers.
- By assumption, M runs "forever" and we set $S(\_, n)$, $T_{\sigma_i}(n, m)$, and $H(n, m)$ according to the intended meaning of these predicates.
- Moreover, we set $S(q_{halt}, a)$ to true. This is consistent with the rest since, intuitively, time instant a is "never reached".

Finite vs. Infinite Domain (2)

Question. How should we modify the problem reduction to prove undecidability of the Entscheidungsproblem (i.e. validity or, equivalently, unsatisfiability of FO without the restriction to finite models)?

Undecidability of the Entscheidungsproblem

We modify the problem reduction as follows: Transform the formula φ_M into φ'_M as follows: we replace the subformula φ_{halt} in φ_M by $\neg\varphi_{halt}$. Then we have: φ'_M has no model at all iff M halts on ϵ.

In other words, we have reduced **HALTING-ϵ** to Unsatisfiability.

Question. Does this reduction also work for finite unsatisfiability?

The answer is "no", because of the the "$\Rightarrow$" direction.
Indeed, suppose that M does not terminate on input ϵ. Then, by the above equivalence, φ'_M has a model – but no finite model! Intuitively, since M does not halt, any model refers to infinitely many time instants.

Learning Objectives

- Short recapitulation of
  - Turing machines,
  - undecidability (the **HALTING** problem).
- Formulation of Trakhtenbrot's Theorem in terms of FO logic and databases.
- Proof of Trakhtenbrot's Theorem.
- Further undecidability results.
- Differences between finite and infinite domain.