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!
- We first recall some basic notions and results from the lecture “Formale Methoden der Informatik”.

Turing Machines

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

**Definition**

A **Turing machine** is a quadruple $M = (K, \Sigma, \delta, s)$ with a finite set of states $K$, a finite set of symbols $\Sigma$ (alphabet of $M$) so that $\sqcup, \triangledown \in \Sigma$, a transition function $\delta$:

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

a halting state $q_{\text{halt}}$, an accepting state $q_{\text{yes}}$, a rejecting state $q_{\text{no}}$, and R/W head directions: $+1$ (right), $-1$ (left), and 0 (stay).
Function $\delta$ is the “program” of the machine.

For the current state $q \in K$ and the current symbol $\sigma \in \Sigma$,

- $\delta(q, \sigma) = (p, \rho, D)$ where $p$ is the new state,
- $\rho$ is the symbol to be overwritten on $\sigma$, and
- $D \in \{+1, -1, 0\}$ is the direction in which the R/W head will move.

For any states $p$ and $q$,

$$\delta(q, \triangle) = (p, \rho, D)$$

with $\rho = \triangle$ and $D = +1$.

In other words: The delimiter $\triangle$ 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 $M = (K, \Sigma, \delta, s)$ is $s$,
(ii) the tape is initialized to the infinite string $\triangle I \sqcup \ldots$, 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 $\triangle$.

The machine halts iff $q_{\text{halt}}$, $q_{\text{yes}}$, or $q_{\text{no}}$ has been reached.

If $q_{\text{yes}}$ has been reached, then the machine accepts the input.

If $q_{\text{no}}$ has been reached, then the machine rejects the input.

If $q_{\text{halt}}$ has been reached, then the machine produces output.

### Church-Turing Thesis

**Church-Turing Thesis**

Any “reasonable” attempt to model mathematically computer algorithms ends up with a model of computation that is equivalent to Turing machines.

**Evidence for this thesis**

All of the following models can be shown to have precisely the same expressive power as Turing machines:

- Random access machines
- $\mu$-recursive functions
- any conventional programming language (Java, C, …)

**Strengthening of the Church-Turing Thesis**

Turing machines are not less efficient than other models of computation!

### 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-$\epsilon$**

INSTANCE: A Turing machine $M$.

QUESTION: Does $M$ halt on the empty string $\epsilon$, i.e. does $M$ reach $q_{\text{halt}}$, $q_{\text{yes}}$, or $q_{\text{no}}$ when run on the initial tape contents $\triangle \sqcup \sqcup \ldots$?

### Trakhtenbrot’s Theorem

**Theorem (Trakhtenbrot’s Theorem, 1950)**

For every relational vocabulary $\sigma$ with at least one binary relation symbol, it is undecidable to check whether an FO sentence $\varphi$ over $\sigma$ is finitely satisfiable (i.e. has a finite model).

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

**Theorem**

For a database schema $\sigma$ with at least one binary relation, it is undecidable whether a Boolean FO or RA query $Q$ over $\sigma$ is satisfied by at least one database.
Idea to prove Traktenbrot’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:
  \[ \varphi_M \text{ has a finite model iff } M \text{ halts on } \epsilon. \]
- The undecidability of HALTING-ε together with the reduction proves Trakhenbrot’s Theorem!

Proof of Trakhenbrot’s Theorem

Assume a machine \( M = (K, \Sigma, \delta, q_{start}) \).
Simplifying assumptions:
- \( \sigma \) may have several unary and binary relations
  - Exercise. We could easily encode them into a single binary relation.
- Tape alphabet of \( M \) is \( \Sigma = \{0, 1, \text{>, <}\} \)
  - 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:
    \[ a_0 \to 000, a_1 \to 001, a_2 \to 010, a_3 \to 011, \ldots, a_k \to 111, \ldots \]

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

- \( < \) must be a strict linear order (a total, transitive, antisymmetric, irreflexive relation). Thus \( \varphi_< \) is the conjunction of:
  \[
  \forall x, y. (x < y \iff (x < y \land y < x))
  \]
  \[
  \forall x, y, z. ((x < y \land y < z) \to x < z)
  \]
  \[
  \forall x, y. \neg (x < y \land y < x)
  \]
- We axiomatize the successor relation based on \( < \) as follows:
  \[
  \forall x, y. (\text{Succ}(x, y) \iff (x < y) \land \neg \exists z. (x < z \land z < y))
  \]
Min must contain the minimal element of \(<\). Thus \(\varphi_{\text{Min}}\) is:
\[
\forall x, y. (\text{Min}(x) \leftrightarrow (x = y \lor x < y))
\]

The formula \(\varphi_{\text{comp}}\) is defined as:
\[
\varphi_{\text{comp}} \equiv \exists y_0, y_1, \ldots, y_k (\varphi_{\text{states}} \land \varphi_{\text{rest}}),
\]
where each variable \(y_i\) corresponds to the state \(q_i\) of \(M\) (we assume the TM has \(k + 1\) states), and
\[
\varphi_{\text{states}} \equiv \bigwedge_{0 \leq i < j \leq k} y_i \neq y_j.
\]
Intuitively, using the \(\exists y_0, y_1, \ldots, y_k\) prefix and \(\varphi_{\text{states}}\) we associate to each state of \(M\) a distinct domain element.

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

(R2) Formulas stating that in every configuration, each cell of the tape contains exactly one symbol:
\[
\forall p, t. (T_0(p, t) \lor T_1(p, t) \lor T_p(p, t) \lor T_{\perp}(p, t)),
\]
\[
\forall p, t. (\lnot T_{\sigma_1}(p, t) \lor \lnot T_{\sigma_2}(p, t)), \quad \text{for all } \sigma_1 \neq \sigma_2 \in \Sigma
\]

(R3) A formula stating that at any time the machine is in exactly one state:
\[
\forall t. [(\bigwedge_{0 \leq i \leq k} S(y_i(t)) \land \bigwedge_{0 \leq i < j \leq k} \lnot (S(y_i(t)) \land S(y_j(t)))]
\]

(R4) A formula stating that at any time the head is at exactly one position:
\[
\forall t. (\exists p. (H(p, t) \land \forall p', p'. (H(p, t) \land H(p', t) \rightarrow p = p'))
\]

(R1) Formula defining the initial configuration of \(M\) with \(\triangledown\triangledown\triangledown\ldots\) on its input tape:
- At time instant 0 the tape has \(\triangledown\) in the first cell of the tape:
\[
\forall p. (\text{Min}(p) \rightarrow T_0(p, p))
\]
- All other cells contain \(\sqcup\) at time 0:
\[
\forall p, t. ((\text{Min}(t) \land \neg \text{Min}(p)) \rightarrow T_\perp(p, t))
\]
- The head is initially at the start position 0:
\[
\forall t. (\text{Min}(t) \rightarrow H(t, t))
\]
- The machine is initially in state \(q_{\text{start}}\):
\[
\forall t. (\text{Min}(t) \rightarrow S(y_{\text{start}}, t))
\]

(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:
\[
\forall p, t. [(H(p, t) \land T_{\sigma_1}(p, t) \land S(y_1, t)) \rightarrow \exists p', t'. (\text{FollowTo}(p, p') \land \text{Succ}(t, t') \land H(p', t') \land S(y_2, t') \land T_{\sigma_2}(p, t') \land \forall r. (r \neq p \land T_0(r, t) \rightarrow T_0(r, t')) \land \forall r. (r \neq p \land T_1(r, t) \rightarrow T_1(r, t')) \land \forall r. (r \neq p \land T_\perp(r, t) \rightarrow T_\perp(r, t')) \land \forall r. (r \neq p \land T_{\perp}(r, t) \rightarrow T_{\perp}(r, t'))]
\]
where:
\[
\text{FollowTo}(p, p') \equiv \begin{cases} 
\text{Succ}(p, p') & \text{if } D = +1, \\
\text{Succ}'(p, p') & \text{if } D = -1, \\
p = p' & \text{if } D = 0.
\end{cases}
\]
(R6) A formula \( \varphi_{\text{halt}} \) saying that \( M \) halts on input \( I \):

\[
\exists t. (S(y_{\text{halt}}, t) \lor S(y_{\text{yes}}, t) \lor S(y_{\text{no}}, t)).
\]

This completes the description of the formula \( \varphi_M \), which faithfully describes the computation of \( M \) on the empty word \( \epsilon \).

By construction of \( \varphi_M \), we have:

\[
\varphi_M \text{ has a finite model iff } M \text{ halts on } \epsilon.
\]

This completes the reduction from \textsc{HALTING}-\( \epsilon \) 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: \( \forall A : Q_1(A) \subseteq Q_2(A) \);
- checking equivalence of two FO (or RA) queries.

**Proof Sketches**

**Undecidability of Domain Independence**

By reduction from finite unsatisfiability:
Let \( \varphi \) be an arbitrary instance of finite unsatisfiability. Construct the following instance \( \psi \) of Domain Independence:

w.l.o.g. let \( x \) be a variable not occurring in \( \varphi \);
then we set \( \psi = \neg R(x) \land \varphi \).

**Undecidability of Query Containment and Query Equivalence**

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

**Finite vs. Infinite Domain**

**Motivation**

Recall the following property of the formula \( \varphi_M \) in the proof of Trakhtenbrot’s Theorem: \( \varphi_M \) has a finite model iff \( M \) halts on \( \epsilon \).

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

It turns out that the ("\( \Rightarrow \" \) direction of the) equivalence “\( \varphi_M \) has an arbitrary model iff \( M \) halts on \( \epsilon \)” does not hold. Indeed, suppose that \( M \) does not terminate on input \( \epsilon \). Then \( \varphi_M \) 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 \( S(\_, n), T_{\sigma}(n, m), \) and \( H(n, m) \) according to the intended meaning of these predicates.
- Moreover, we set \( S(q_{\text{halt}}, a) \) to true. This is consistent with the rest since, intuitively, time instant \( a \) is “never reached”.
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 $\varphi_M$ into $\varphi'_M$ as follows: we replace the subformula $\varphi_{\text{halt}}$ in $\varphi_M$ by $\neg \varphi_{\text{halt}}$.

Then we have: $\varphi'_M$ has no model at all iff $M$ halts on $\epsilon$.

In other words, we have reduced HALTING-$\epsilon$ 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 $\epsilon$. Then, by the above equivalence, $\varphi'_M$ has a model – but no finite model! Intuitively, since $M$ does not halt, any model refers to infinitely many time instants.

Semi-Decidability (2)

Recall that satisfiability of FO is not semi-decidable. In contrast, we now show that finite satisfiability is semi-decidable.

Proof idea

- The evaluation of an FO formula in an interpretation is defined by a recursive algorithm. This algorithm terminates over finite domains.
- Hence, it is decidable if a given formula $\varphi$ is satisfied by a finite interpretation $\mathcal{I}$.
- Hence, for finite signatures, the problem whether an FO formula has a model with a given finite cardinality is decidable.
- Therefore, for finite signatures, finite satisfiability of FO is semi-decidable.