Optimization Problems for Recursive Database Queries

Moshe Y. Vardi

Computer Science

RELATIONAL MODEL

Relational model (Codd, 1970): Data organized in tables

CUSTOMERS	NAME	ADDRESS	BALANCE

DATABASE QUERY LANGUAGES

SQL:

```
SELECT NAME
FROM CUSTOMERS
WHERE BALANCE < 0
```

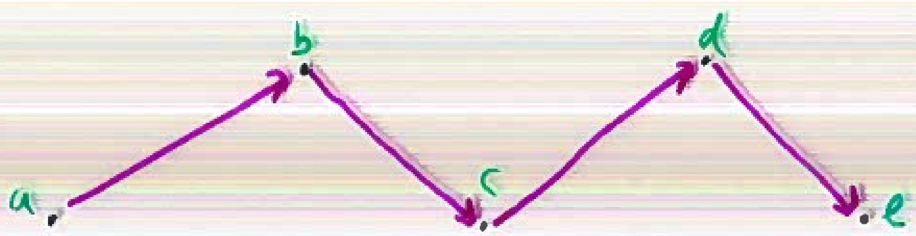
- SQL is ***first-order***.
- Codd, 1970: 1st-order languages are expressive enough.
- Aho and Ullman, 1978: 1st-order languages are **not** expressive enough.

TRANSITIVE CLOSURE

FLIGHTS	ORIGIN	DESTINATION

Query TC:

SELECT ORIGIN, DESTINATION
FROM FLIGHTS
WHERE there is a route from ORIGIN
to DESTINATION



Aho+Ullman: TC is not expressible in SQL.

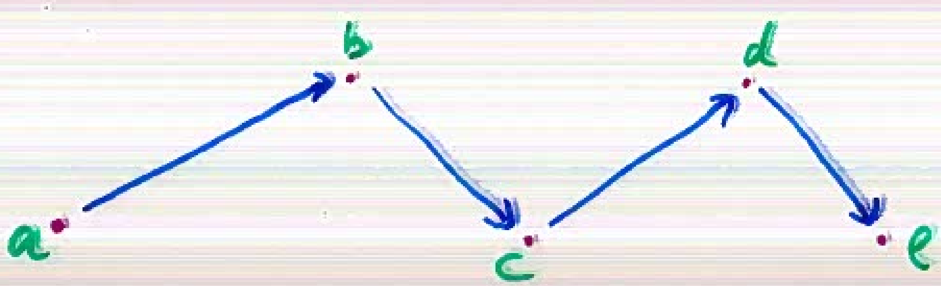
RECURSION

$$Route(X, Y) \leftarrow Flights(X, Y)$$

$$Route(X, Y) \leftarrow Flights(X, Z), Route(Z, Y)$$

Flights - ***base***

Route - ***derived***



Flights(a, b), Flights(b, c), Flights(c, d) . . .

Route (a, b), Route (b, c), Route (c, d) . . .

Route (a, c), Route (b, d) . . .

Route (a, d) . . .

COMPUTATIONAL COMPLEXITY

- Expressiveness costs ***money***: recursive query are harder to evaluate.
  - ***1st-order queries***: speed-up by parallel processing.
  - ***Recursive queries***: No speed-up.
- ***Remedy:*** Automated optimization.

COMPUTER SCIENCE THEMES

- Trade-off between expressiveness and computational complexity.
- What can be automated?

OPTIMIZATION

$Buys(X, Y) \leftarrow Cheap(Y), Likes(X, Y)$

$Buys(X, Y) \leftarrow Cheap(Y), Knows(X, Z), Buys(Z, Y)$

$Cheap(Y)$ is ***redundant***.

$Buys(X, Y) \leftarrow Cheap(Y), Likes(X, Y)$

$Buys(X, Y) \leftarrow Rich(X), Knows(X, Z), Buys(Z, Y)$

$Rich(X)$ is ***not*** redundant.

THE SHOPPERS

The Trendy Shopper:

$$Buys(X, Y) \leftarrow Likes(X, Y)$$

$$Buys(X, Y) \leftarrow Trendy(X), Buys(Z, Y)$$

The Impressionable Shopper:

$$Buys(X, Y) \leftarrow Likes(X, Y)$$

$$Buys(X, Y) \leftarrow Knows(X, Z), Buys(Z, Y)$$

BOUNDEDNESS

The Trendy shopper:

$Buys(X, Y) \leftarrow Likes(X, Y)$

$Buys(X, Y) \leftarrow Trendy(X), Buys(Z, Y)$

Program is ***bounded***: 2 iterations suffices.

Equivalently:

$Buys(X, Y) \leftarrow Likes(X, Y)$

$Buys(X, Y) \leftarrow Trendy(X), \underline{Likes(Z, Y)}$

Boundedness $\equiv$ Eliminable recursion.

UNSOLVABILITY

Theorem: The boundedness problem is unsolvable. That is, there is no program that will always identify bounded queries!

***Proof*:**

1. ***Reduction:*** an automated way of using a program for a problem *B* to solve a problem A.

2. If *A* is unsolvable and *A* is ***reducible*** to *B*, then *B* is unsolvable.

3. The ***halting problem*** is unsolvable.

4. The halting problem is reducible to boundedness.

UNARY QUERIES

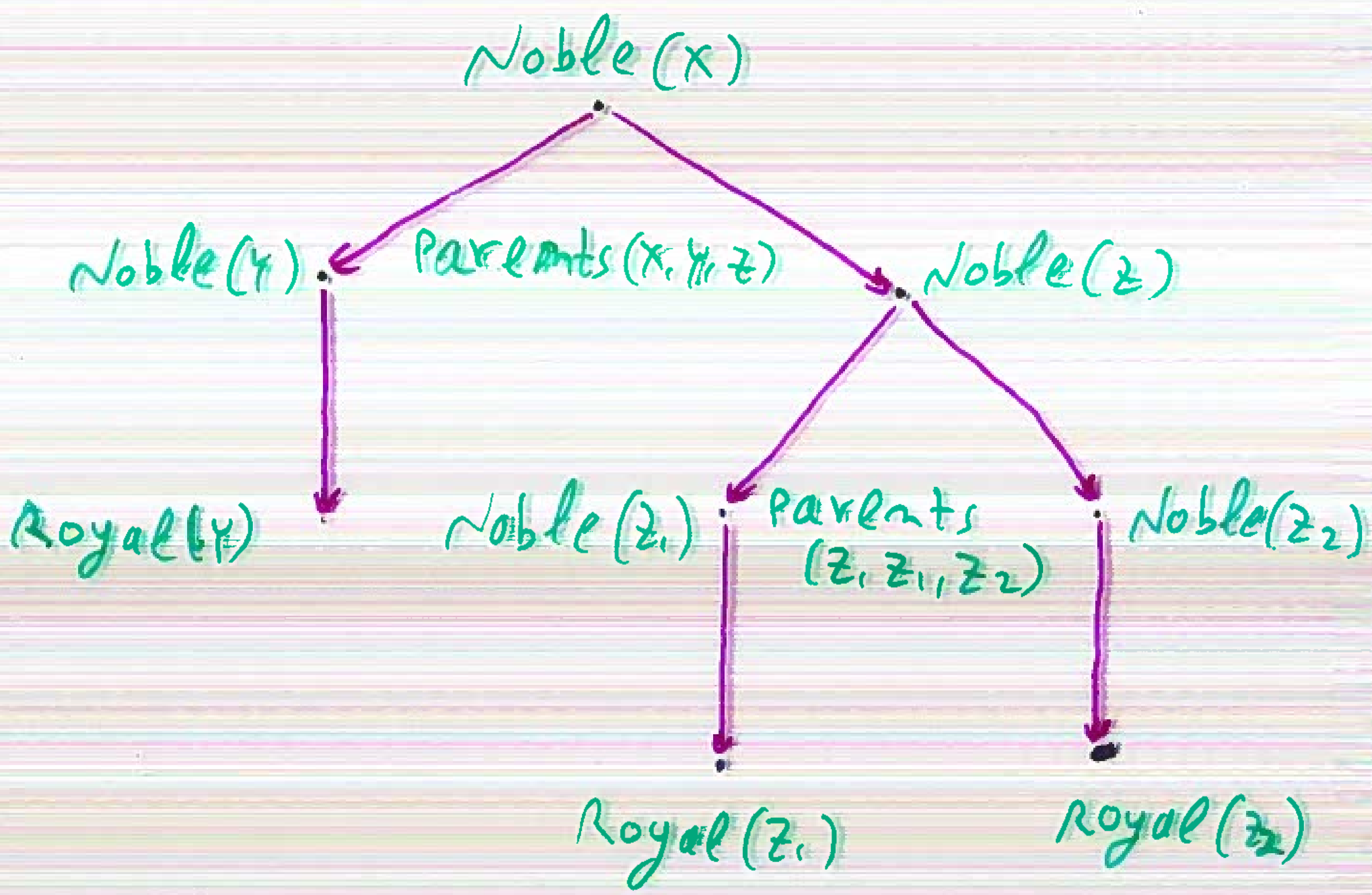
$Noble(X) \leftarrow Royal(X)$

$Noble(X) \leftarrow Noble(Y), Noble(Z), Parents(X, Y, Z)$

Royal, Parents - ***base***
Noble - ***derived***

Theorem: The boundedness problem for unary queries is solvable.

PROOF TREES



Noble(x)
Noble(y)
Parents(x,y,z)
Noble(z)
Royal(y)
Noble(z1)
Parents (z,z1,z2)
Noble(z2)
Royal(z1)
Royal(z2)

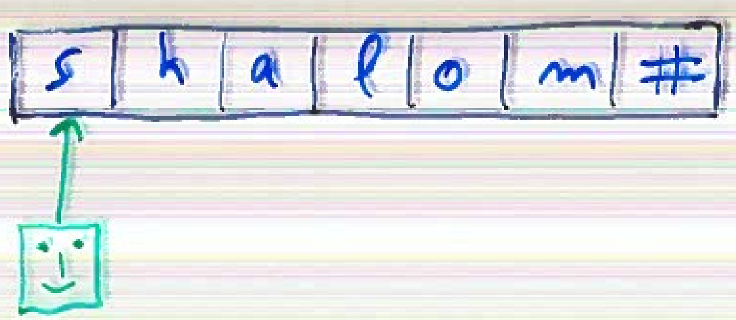
FORMAL LANAGUAGE THEORY

- ***Alphabet*** - a finite set of symbols.
- ***Word*** - a finite sequence of symbols from the alphabet.
- ***Language*** - a collection of words over the alphabet.

WORD AUTOMATA

Automaton: $A = (\Sigma, S, I, F, \rho)$

- Σ: alphabet
- S: finite set of states
- I: initial states
- F: accepting states
- ρ: transition relation - collection of triples (s, a, t), meaning that A can make a transition from s to t upon reading a.



APPLYING AUTOMATA

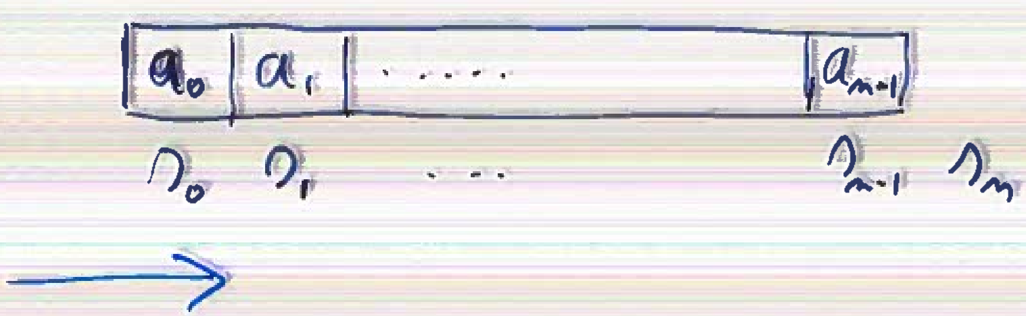
- **Word** w: $a_0, \ldots, a_{n-1}$

- **Run** r: $s_0, \ldots, s_n$

    s_0 in I, (s_i, a_i, s_{i+1}) in ρ

- **Acceptance**: s_n in F.

- $L(A)$ - the set of words accepted by A.



FINITENESS PROBLEM FOR AUTOMATA

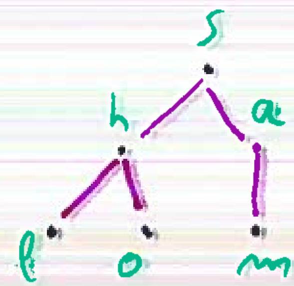
- ***Finiteness Problem***: Given an automaton A, determine if $L(A)$ is finite or not.

- Rabin+Scott, 1959: The finiteness problem is solvable.

TREE LANAGUAGE THEORY

- ***Alphabet*** - a finite set of symbols.

- ***Tree*** -



- ***Tree Language*** - a collection of trees over the alphabet.

- ***Tree automata*** - $T(A)$: the set of trees accepted by A.

- The finiteness problem for tree automata is solvable.

UNARY BOUNDEDNESS

Reduction: From a unary query $\mathcal{P}$ we can construct a tree automaton $A_{\mathcal{P}}$ such that $\mathcal{P}$ is bounded iff $A_{\mathcal{P}}$ accepts only finitely many trees.

Corollary: Boundedness for unary queries is solvable.

CONCLUDING REMARKS

- ***Disappointment***: General optimization of recursive queries is ***very*** difficult.

- ***Hope***: Some optimization of recursive queries is possible.

Such is life in computer science!!!