What do human evolution, leukemia treatment, database design (and a few other things) have in common?

by

Prof. Wong Limsoon

Plan

• What is an invariant?
  – Bet on color of the bean
  – 21 cards
  • Problem solving by logical reasoning on invariants

• Origin of Polynesians

• Make a list sorted
• Design a good database
• Diagnose leukemia
• Make exponentiation faster

• Guilt by association of invariants
• Solution optimization by preserving invariants

Bet on the last green bean

• Suppose you have a bag of x red beans and y green beans
• Repeat the following:
  – Remove 2 beans
  – If both green, discard both
  – If both red, discard one, put back one
  – If one green and one red, discard red, put back green
• If one bean is left behind, can you predict its colour?

Shall we bet on the color of the bean that is left behind?

• Suppose you have a bag of x red beans and y green beans
• Repeat the following:
  – Remove 2 beans
  – If both green, discard both
  – If both red, discard one, put back one
  – If one green and one red, discard red, put back green
• If one bean is left behind, can you predict its colour?

• When the parity of # of green beans (y) is odd, ...
  • Start with y=2n+1
    • y=2n+1 \rightarrow y=2n-1
    • y=2n+1 \rightarrow y=2n+1
    • y=2n+1 \rightarrow y=2n+1
    • y remains odd \Rightarrow Last bean must be green!

Fun With Invariants

Limsoon Wong
Bet on the last red bean

- Suppose you have a bag of x red beans and y green beans
- Repeat the following:
  - Remove 2 beans
    - If both green, discard both
    - If both red, discard one, put back one
    - If one green and one red, discard red, put back green
- If one bean is left behind, can you predict its colour?

- When the parity of # of green beans (y) is even, ...
- Start with y=2n

- y=2n \rightarrow y=2n+2
- y=2n \rightarrow y=2n
- y=2n \rightarrow y=2n
- y remains even \Rightarrow Last bean must be red!

Bet on color of the last bean ... and win!

- Suppose you have a bag of x red beans and y green beans
- Repeat the following:
  - Remove 2 beans
    - If both green, discard both
    - If both red, discard one, put back one
    - If one green and one red, discard red, put back green
- If one bean is left behind, can you predict its colour?

- If you start with odd # (even #) of green beans, there will always be an odd # (even #) of green beans in the bag
\Rightarrow Parity of green beans is invariant
\Rightarrow Bean left behind is green iff you start with odd # of green beans

What have we just seen?

- Problem solving by logical reasoning on invariants

Welcome to the Magical World...

The 21-Card Trick

1. Magician asks you to remember any one card from a deck of 21 cards as your card. Do not tell him what the card is
2. He deals the 21 cards face down, from top to bottom and left to right, into 3 equal piles
3. Next, he fans the piles to you and asks you to look for the pile of cards which contains your card and pass the pile back to him
4. Again, he stacks up the 3 piles on top of each other and redistribute, from top to bottom and left to right, into 3 equal piles
5. He repeats step (3) and (4) 2 more times
6. Finally, he deals your card right out from the rest of the 21 cards!

How does he manage that?!
The Trick

- The pile containing the card is being placed in the middle of the other 2 piles
- Imposing constraints on where the card can move to...

Assuming the chosen card is in the first pile.

<table>
<thead>
<tr>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
</tbody>
</table>

After the first distribution, ...

<table>
<thead>
<tr>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

After the second distribution, ...

<table>
<thead>
<tr>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

After the third distribution, ...

What have we just seen?

- Problem solving by logical reasoning on invariants (with a twist of injecting the invariant)

Where do Polynesians come from?

In the course of evolution...

What is the invariant?

- Mitochondrial DNA accumulates 1 mutation about every 10,000 years
- Human history is not so long relative to this

⇒ When a nucleotide in mitochondrial DNA is mutated it stays mutated through future generations
Do Polynesians come from Asia or America?

Origin of Polynesians

- Common mitochondrial control seq from Rarotonga have variants at positions 189, 217, 247, 261. Less common ones have 189, 217, 261.
- Seq from Taiwan natives have variants 189, 217.
- Seq from regions in between have variants 189, 217, 261.
- More 189, 217 closer to Taiwan. More 189, 217, 261 closer to Rarotonga.
- 247 not found in America.⇒ Polynesians came from Taiwan!
- Taiwan seq sometimes have extra mutations not found in other parts.⇒ These are mutations that happened since Polynesians left Taiwan!

The “Invariant” Perspective

- The invariant:
  When a nucleotide in mitochondrial DNA is mutated it stays mutated through future generations.
- The lesson learned:
  Figure out origins of Polynesians by logical reasoning on invariant.

How to get a list sorted?

- What is a sorted list?
  A list L is sorted iff L[i] ≤ L[j] for all adjacent positions i < j.
- So how do you make a list M become sorted?
  While M[i] > M[j] for some adjacent positions i < j {
    swap M[i], M[j]
  }

Sorting a list

- Invariant of sorted lists
  A list L is sorted iff L[i] ≤ L[j] for all adjacent positions i < j.
- Making a list M become sorted:
  While M[i] > M[j] for some adjacent positions i < j {
    swap M[i], M[j]
  }⇒ Find violation of the invariant
  swap M[i], M[j]⇒ Fix it
  }⇒ When no more violation, the list must be sorted!
• What have we just seen?

• Problem solving by rectifying violation of invariants

What is a good database design?

Relational Data Model

Contracts

<table>
<thead>
<tr>
<th>Contract No</th>
<th>Star</th>
<th>Studio</th>
<th>Title</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Carrie Fisher</td>
<td>Fox</td>
<td>Star Wars</td>
<td>$98</td>
</tr>
<tr>
<td>2</td>
<td>Mark Hamill</td>
<td>Fox</td>
<td>Star Wars</td>
<td>$98</td>
</tr>
<tr>
<td>3</td>
<td>Harrison Ford</td>
<td>Fox</td>
<td>Star Wars</td>
<td>$98</td>
</tr>
</tbody>
</table>

Stars

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrie Fisher</td>
<td>Hollywood</td>
</tr>
<tr>
<td>Mark Hamill</td>
<td>Brentwood</td>
</tr>
<tr>
<td>Harrison Ford</td>
<td>Beverly Hills</td>
</tr>
</tbody>
</table>

Movies

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

Design Issues

• How many possible alternate ways to represent movies using tables?
• Why this particular set of tables to represent movies?
• Indeed, why not use this alternative single table below to represent movies?

Wrong Movies

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

Anomalies

• What's wrong with the “Wrong Movies” table?

Wrong Movies

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

• Redundancy: Unnecessary repetition of info
• Update anomalies: If Star Wars is 125 min, we might carelessly update row 1 but not rows 2 & 3
• Deletion anomalies: If Emilio Estevez is deleted from stars of Mighty Ducks, we lose all info on that movie

Functional Dependency

• Functional dependency \((A_1, ..., A_n \rightarrow B_1, ..., B_m)\)
  – If two rows of a table \(R\) agree on attributes \(A_1, ..., A_n\), then they must also agree on attributes \(B_1, ..., B_m\)
  \(\Rightarrow\) Values of \(B\)’s depend on values of \(A\)’s

• Example: Title, Year \(\rightarrow\) Length, Film Type, Studio
• FD \((A_1, ..., A_n \rightarrow B_1, ..., B_m)\) is trivial if a \(B_i\) is an \(A_j\)
Can you identify the FD’s here?

**Wrong Movies**

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1997</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

- Some FD’s:
  - Title, Year → Length
  - Title, Year → Film Type
  - Title, Year → Studio

**Keys**

- **Key**
  - A minimal set of attributes \( \{A_1, \ldots, A_n\} \) that functionally determine all other attributes of a table
  - A key is trivial if it comprises the entire set of attributes of a table

- **Superkey**
  - A set of attributes that contains a key

Can you identify the superkeys here?

**Wrong Movies**

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1997</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

- Superkeys:
  - Any set of attributes that contains \( \{\text{Title}, \text{Year}, \text{Star}\} \) as a subset

**Boyce-Codd Normal Form**

- A relation \( R \) is in **Boyce-Codd Normal Form** iff whenever there is a nontrivial FD \( \{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\} \) for \( R \), it is the case that \( \{A_1, \ldots, A_n\} \) is a superkey for \( R \)

**Theorem A1 (Codd, 1972)**

A database design has no anomalies due to FD iff all its relations are in Boyce-Codd Normal Form

How is BCNF violated here?

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Film Type</th>
<th>Studio</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Carrie Fisher</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Mark Hamill</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1997</td>
<td>124</td>
<td>Color</td>
<td>Fox</td>
<td>Harrison Ford</td>
</tr>
<tr>
<td>Mighty Ducks</td>
<td>1997</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

- A nontrivial FD:
  - Title, Year → Length, Film Type, Studio
  - The LHS not superset of the key \( \{\text{Title}, \text{Year}, \text{Star}\} \)
  - Violate BCNF!

- Anomalies are due to FD’s whose LHS is not superkey

Towards a Better Design

- Use an offending FD \( \{A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m\} \) to decompose \( R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_h) \) into 2 tables
  - \( R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \)
  - \( R_2(A_1, \ldots, A_n, C_1, \ldots, C_h) \)
The "Invariant" Perspective

• The invariants:
  BCNF is an invariant of a good database design

• The lesson learned:
  Deliver a better database design by fixing violated invariants

Impact

<table>
<thead>
<tr>
<th>ORACLE CORPORATION</th>
<th>Q3 FISCAL 2015 FINANCIAL RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDENSED CONSOLIDATED STATEMENTS OF OPERATIONS</td>
<td>($ in millions, except per share data)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Thru Nov 30, 2014</th>
<th>% of</th>
<th>Thru Nov 30, 2013</th>
<th>% of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenues</td>
<td></td>
<td>Revenues</td>
<td></td>
</tr>
<tr>
<td>Hardware licenses</td>
<td>$1,718</td>
<td>$1,710</td>
<td>100%</td>
</tr>
<tr>
<td>Software revenue</td>
<td>2,383</td>
<td>2,817</td>
<td>196%</td>
</tr>
<tr>
<td>Hardware/Software</td>
<td>5,101</td>
<td>4,430</td>
<td>81%</td>
</tr>
<tr>
<td>Inquiries</td>
<td>213</td>
<td>39</td>
<td>19%</td>
</tr>
<tr>
<td>Hardware systems</td>
<td>181</td>
<td>10</td>
<td>9%</td>
</tr>
<tr>
<td>Hardware systems</td>
<td>181</td>
<td>10</td>
<td>9%</td>
</tr>
<tr>
<td>Total/Revenues</td>
<td></td>
<td>$20,972</td>
<td>100%</td>
</tr>
</tbody>
</table>


Data from US

• Data from US

  - 3% of unfilled ICT vacancies are in db mgmt

Data from Singapore

Some Patient Samples

- Does Mr. A have cancer?

Let's rearrange the rows...

- Does Mr. A have cancer?
and the columns too...

Mr. A: 

- What have we just seen?
- Guilt by association of invariants

Childhood Acute Lymphoblastic Leukemia

- Major subtypes: T-ALL, E2A-PBX, TEL-AML, BCR-ABL, MLL, genome rearrangements, Hyperdiploid>50
- Diff subtypes respond differently to same Tx
- Over-intensive Tx
  - Development of secondary cancers
  - Reduction of IQ
- Under-intensive Tx
  - Relapse

- The subtypes look similar

- Conventional diagnosis
  - Immunophenotyping
  - Cytogenetics
  - Molecular diagnostics

- Unavailable in developing countries

Invariant Profile of Leukemia Subtypes

Making an Impact: Leukemia Diagnosis Revisited

Patient Profiles & Treatment Costs

- Treatment for childhood ALL over 2 yrs
  - Intermediate intensity: US$60k
  - Low intensity: US$36k
  - High intensity: US$72k
- Treatment for relapse: US$150k
- Cost for side-effects: Unquantified

2000 new cases a year in ASEAN countries
Why not high/low intensity to everyone?

- **High-intensity Tx**
  - Over intensive for 90% of patients, thus a lot more side effects
  - US$144m (US$72k * 2000) for high-intensity tx

  ⇒ Total US$144m/yr plus un-quantified costs for dealing with side effects

- **Low-intensity Tx**
  - Under intensive for 50% of patients, thus a lot more relapse
  - US$72m (US$36k * 2000) for low-intensity tx
  - US$150m (US$75k * 2000 * 50%) for relapse tx

  ⇒ Total US$222m/yr

Current Situation

- **Intermediate intensity** conventionally applied in ASEAN countries

  - Over intensive for 50% of patients, thus more side effects
  - Under intensive for 10% of patients, thus more relapse

  - US$120m (US$60k * 2000)
  - US$30m (US$150k * 2000 * 10%) for relapse tx

  Total US$150m/yr plus un-quantified costs for dealing with side effects

Exploit Invariant Gene Expr Profiles

- Low intensity applied to 50% of patients
- Intermediate intensity to 40% of patients
- High intensity to 10% of patients

  - US$36m (US$36k * 2000 * 50%)
  - US$48m (US$60k * 2000 * 40%)
  - US$14.4m (US$72k * 2000 * 10%) for high intensity

  ⇒ Reduced side effects
  ⇒ Reduced relapse
  ⇒ 75-80% cure rates

  ⇒ Total US$98.4m/yr

  ⇒ Save US$51.6m/yr

Expontiation

- What does this program do?
  
  \[ F(a, 0) = 1 \]
  
  \[ F(a, n+1) = a \times F(a, n) \]

- We see that
  
  \[ F(a, n) = a^n \]
Playing the invariant…

What does this program do?

F(a, 0) = 1
F(a, n+1) = a * F(a, n)

Then

F(a, 2^n) = a^(2^n) = a * a^n = y * y where y = F(a, n)

We see that

F(a, n) = a^n

So we get...

What’s the difference?

Original program:

F(a, 0) = 1
F(a, n+1) = a * F(a, n)

New program:

F(a, 0) = 1
F(a, 1) = a
F(a, n) = if n is odd
then a * y * y
else y * y
where y = F(a, n div 2)

Cost of F(a, n) = n

Cost of F(a, n) = log_2 n

New program:

F(a, 0) = 1
F(a, n+1) = a * F(a, n)

Original program:

F(a, 0) = 1
F(a, n+1) = a * F(a, n)

Cost of F(a, n) = n

Cost of F(a, n) = log_2 n

What have we just seen?

Optimizing a solution by preserving invariant

How to make computers safer?

Rootkit Problem

Traditional rootkits
– Modify scalar invariants in OS
  • kernel table
  • interrupt table
  • syscall table

Modern rootkits
– Direct Kernel Object Manipulation (DKOM)
– Rather than modify scalar invariants in OS, data of kernel are modified to:
  • Hide processes
  • Increase privilege level
Hiding a window process

Semantic integrity

- Current integrity monitoring systems focus on the scalar nature of the monitored data
  - Work for scalar (i.e., invariant) data
  - Don't work for non-scalar data

- Semantic integrity
  - Monitor non-invariant portions of a system via predicates that remain valid during the proper operation of the system
  - I.e., monitor invariant dynamic properties!

DKOM Example

- Semantic integrity predicate (i.e., dynamic invariant) is

  - There is no thread such that its parent process is not on the process list

  \[ \neg \text{kHIVE (contains 20k other predicates)} \]

What have we just seen?

- Maintain computer safety by checking violation of invariants!

Impact

- 2008: Komoku (kHIVE) acquired by Microsoft
- 2009: Put into MS Security Essentials (~4m hosts)
- 2010: Put into Windows Update (~500m hosts)

“There is no other field out there where you can get right out of university and define substantial aspects of a product that is going to go out and over 100 million people are going to use it”. ---Bill Gates

Remarks
What have we learned?

- Invariant is a fundamental property of many problems

- Paradigms of problem solving
  - Problem solving by logical reasoning on invariants
  - Problem solving by rectifying/monitoring violation of invariants
  - Guilt by association of invariants
  - Solution optimization by preserving invariants

I didn’t get to telling you yet, but …

- Every time you write a loop in a program, it involves an invariant

- Every time you do a recursive function call, it involves an invariant

- Every time you do an induction proof, it involves an invariant

- … Computing is about discovering, understanding, exploiting, and having fun with invariants!

A Test: “Restoring” Historic Documents

Suggest a way to digitally restore damaged historic documents

Enjoy!