What do gambling, leukemia treatment, database design, and computer security have in common?

Wong Limsoon

21 April 2012

NUS
National University of Singapore

Invariants:
The Golden Thread of Science

Science is characterized by

- **Observing an invariant, a law, etc…**
- **Proving that it is true**
- **Exploiting it to solve problems**

Biology/Chemistry is no more about Petri dish & test tube than Computer Science is about programming

Plan

- **What is an invariant?**
  - Bet on color of the bean
  - Efficiency of PTPs
- Problem solving by logical reasoning on invariants

- **Design a good database**
- Fixing db design by rectifying violation of invariants

- **Diagnose leukemia**
- Guilt by association of invariants

- **Make computers safer**
- Rootkit detection by monitoring violation of invariants

What is an invariant?

- **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?

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?**

- **When the parity of # of green beans (y) is odd, …**
- **Start with y=2n+1**
- **y=2n+1 → y=2n-1**
- **y=2n+1 → y=2n+1**
- **y=2n+1 → y=2n+1**
- **y remains odd**

⇒ **Last bean must be green!**

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 → y=2n-2**
- **y=2n → y=2n**
- **y=2n → 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 w/ 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**

Science is characterized by …

Observing an invariant:
Parity of green beans is invariant

Proving it:

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 → y=2n-2
- y=2n → y=2n
- y=2n → y=2n
- y remains even

⇒ Last bean must be red!

Exploit it to solve problems:
Predict colour of the last bean

Why are some PTPs inefficient?

National University of Singapore

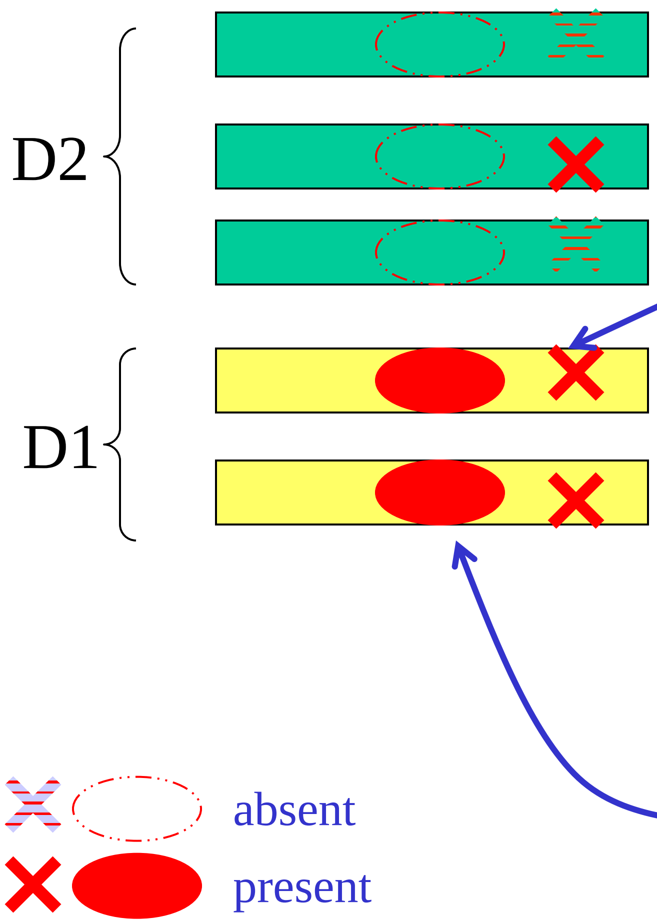
Protein Tyrosine Phosphatase

Sequence from a typical PTP

```
>gi|00000|PTPA-D2
EEEFKKLTSIKIQNDKMRTGNLPANMKKNRVLQIIPYEFNRVIIPVKRGEENTDYVNASF
IDGYRQKDSYIASQGPLLHTIEDFWRMIWEWKSCSIVMLTELEERGQEKCAQYWPSDGLV
SYGDITVELKKEEECESYTVRDLLVTNTRENKSRQIRQFHFHGWPEVGIPSDGKGMISII
AAVQKQQQQSGNHPITVHCSAGAGRTGTFCALSTVLERVKAEGILDVFQTVKSLRLQRPH
MVQTLEQYEFCYKVVQEYIDAFSDYANFK
```

- **Some PTPs are much less efficient than others**
- **Why? And how do you figure out which mutations cause the loss of efficiency?**

Reasoning based on an invariant…



This site is conserved in D1, but is not consistently missing in D2
⇒ Not a likely cause of D2's loss of function

This site is conserved in D1, but is consistently missing in D2
⇒ Possible cause of D2's loss of function

Key Mutation Site: PTP D1 vs D2

```
                    ?   !  ?         ?                 ?             ? ??
gi|00000|P  D2  QFHFHGWPEVGIPSDGKGMISIIAAVQKQQQQ-SGNHPITVHCSAGAGRTGTFCALSTVL
gi|126467|      QFHFTSWPDFGVPFTPIGMLKFLKKVKACNP--QYAGAIVVHCSAGVGRTGTFVVIDAML
gi|2499753      QFHFTGWPDHGVPYHATGLLSFIRRVKLSNP--PSAGPIVVHCSAGAGRTGCYIVIDIML
gi|462550|      QYHYTQWPDMGVPEYALPVLTFVRRSSAARM--PETGPVLVHCSAGVGRTGTYIVIDSML
gi|2499751      QFHFTSWPDHGVPDTTDLLINFRYLVRDYMKQSPPESPILVHCSAGVGRTGTFIAIDRLI
gi|1709906  D1  QFQFTAWPDHGVPEHPTPFLAFLRRVKTCNP--PDAGPMVVHCSAGVGRTGCFIVIDAML
gi|126471|      QLHFTSWPDFGVPFTPIGMLKFLKKVKTLNP--VHAGPIVVHCSAGVGRTGTFIVIDAMM
gi|548626|      QFHFTGWPDHGVPYHATGLLSFIRRVKLSNP--PSAGPIVVHCSAGAGRTGCYIVIDIML
gi|131570|      QFHFTGWPDHGVPYHATGLLGFVRQVKSKSP--PNAGPLVVHCSAGAGRTGCFIVIDIML
gi|2144715      QFHFTSWPDHGVPDTTDLLINFRYLVRDYMKQSPPESPILVHCSAGVGRTGTFIAIDRLI
                * ..  **. *.*      .                  . ****** ****..  .  ..
```

- **Positions marked by “!” and “?” are likely places responsible for reduced PTP activity**
  - All PTP D1 agree on them
  - All PTP D2 disagree on them

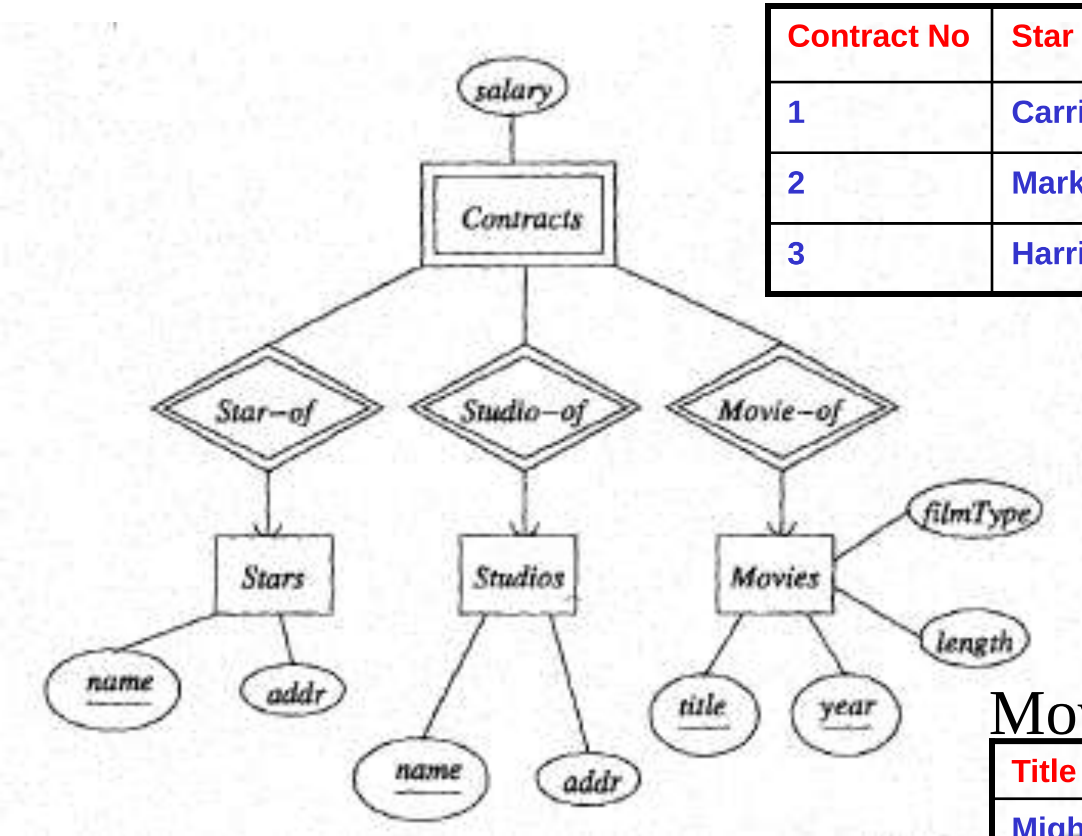
Confirmation by Mutagenesis Expt

- **Wet expts to confirm the prediction**
  - Mutate D $\rightarrow$ E in D1
    - **i.e., check if D $\rightarrow$ E can cause efficiency loss**
  - Mutate E $\rightarrow$ D in D2
    - **i.e., show D $\rightarrow$ E is the cause of efficiency loss**

Impact:
Hundreds of mutagenesis expts saved by simple reasoning on (violation of) invariants!

What is a good database design?

Relational Data Model



Contracts

Contract No	Star	Studio	Title	Salary
1	Carrie Fisher	Fox	Star Wars	$$$
2	Mark Hamill	Fox	Star Wars	$$$
3	Harrison Ford	Fox	Star Wars	$$$

Stars

Name	Address
Carrie Fisher	Hollywood
Mark Hamill	Brentwood
Harrison Ford	Beverly Hills

Movies

Title	Year	Length	Film Type
Mighty Ducks	1991	104	Color
Wayne’s World	1992	95	Color
Star Wars	1977	124	Color

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

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

Anomalies

- **What's wrong with the "Wrong Movies" table?**

Wrong Movies

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

- **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**

Some Interesting Questions

- **How to differentiate a good database design from a bad one?**
- **How to produce a good database design automatically from a bad one?**

Functional Dependency

- **Functional dependency ($A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$)**
  - If two rows of a table R agree on attributes $A_1, \ldots, A_n$, then they must also agree on attributes $B_1, \ldots, B_m$

  $\Rightarrow$ Values of B's depend on values of A's
- **FD ($A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$) is trivial if a B_i is an A_j**

Wrong Movies

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

- **Example: Title, Year $\rightarrow$ Length, Film Type, Studio**

Keys

- **Key** is a minimal set of attributes $\{A_1, \ldots, A_n\}$ that functionally determine all other attributes of a table
- **Superkey** is a set of attributes that contains a key

Wrong Movies

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

- Example superkey: Any set of attributes that contains {Title, Year, 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 (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?

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

- **A nontrivial FD:**
  - Title, Year → Length, Film Type, Studio
  - The LHS not superset of the key {Title,Year, 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**
  - $R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$
  - $R_2(A_1, \ldots, A_n, C_1, \ldots, C_h)$

No update anomaly

No redundant info

Title	Year	Length	Film Type	Studio
Star Wars	1997	124	Color	Fox
Mighty Ducks	1991	104	Color	Disney

No deletion anomaly

Title	Year	Star
Star Wars	1997	Carrie Fisher
Star Wars	1997	Mark Hamill
Star Wars	1997	Harrison Ford
Mighty Ducks	1991	Emilio Estevez

Wrong Movies

Title	Year	Length	Film Type	Studio	Star
Star Wars	1997	124	Color	Fox	Carrie Fisher
Star Wars	1997	124	Color	Fox	Mark Hamill
Star Wars	1997	124	Color	Fox	Harrison Ford
Mighty Ducks	1991	104	Color	Disney	Emilio Estevez

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

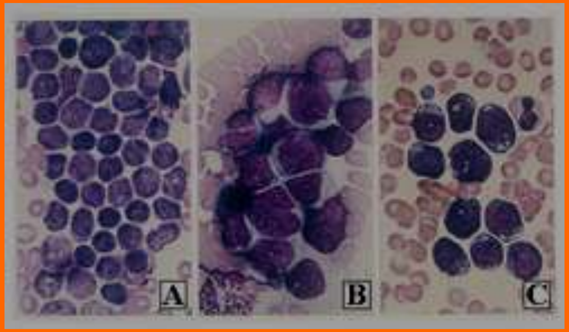
Impact

ORACLE CORPORATION

Q3 FISCAL 2010 FINANCIAL RESULTS
CONDENSED CONSOLIDATED STATEMENTS OF OPERATIONS
($ in millions, except per share data)

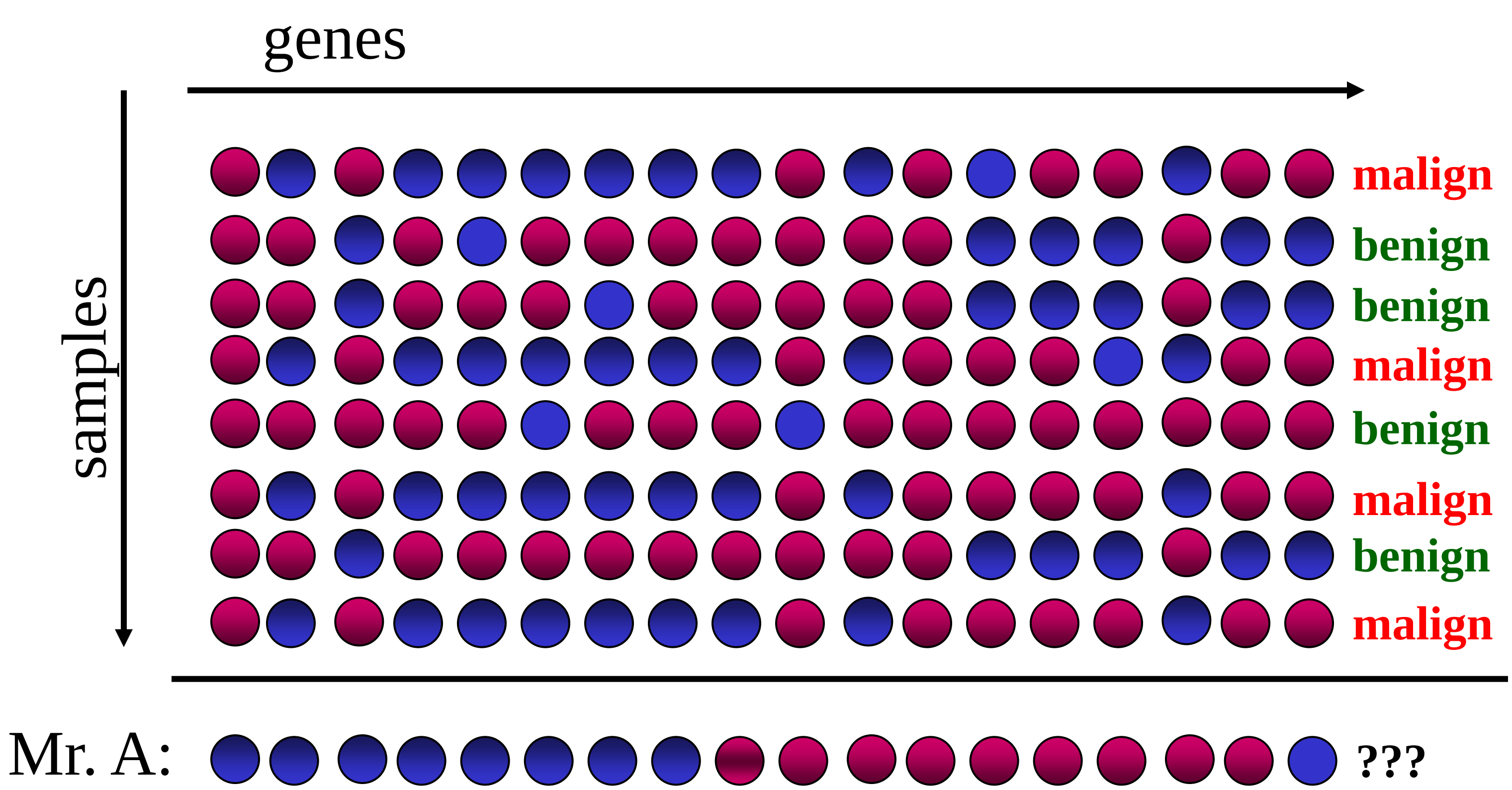
	Three Months Ended February 28,				% Increase (Decrease) in US $
	2010	% of Revenues	2009	% of Revenues	
REVENUES					
New software licenses	$ 1,718	27%	$ 1,516	28%	13%
Software license updates and product support	3,297	51%	2,917	53%	13%
Software Revenues	5,015	78%	4,433	81%	13%
Hardware systems products	273	4%	-	0%	*
Hardware systems support	185	3%	-	0%	*
Hardware Systems Revenues	458	7%	-	0%	*
Services	931	15%	1,020	19%	(9%)
Total Revenues	6,404	100%	5,453	100%	17%

Diagnosing Leukemias



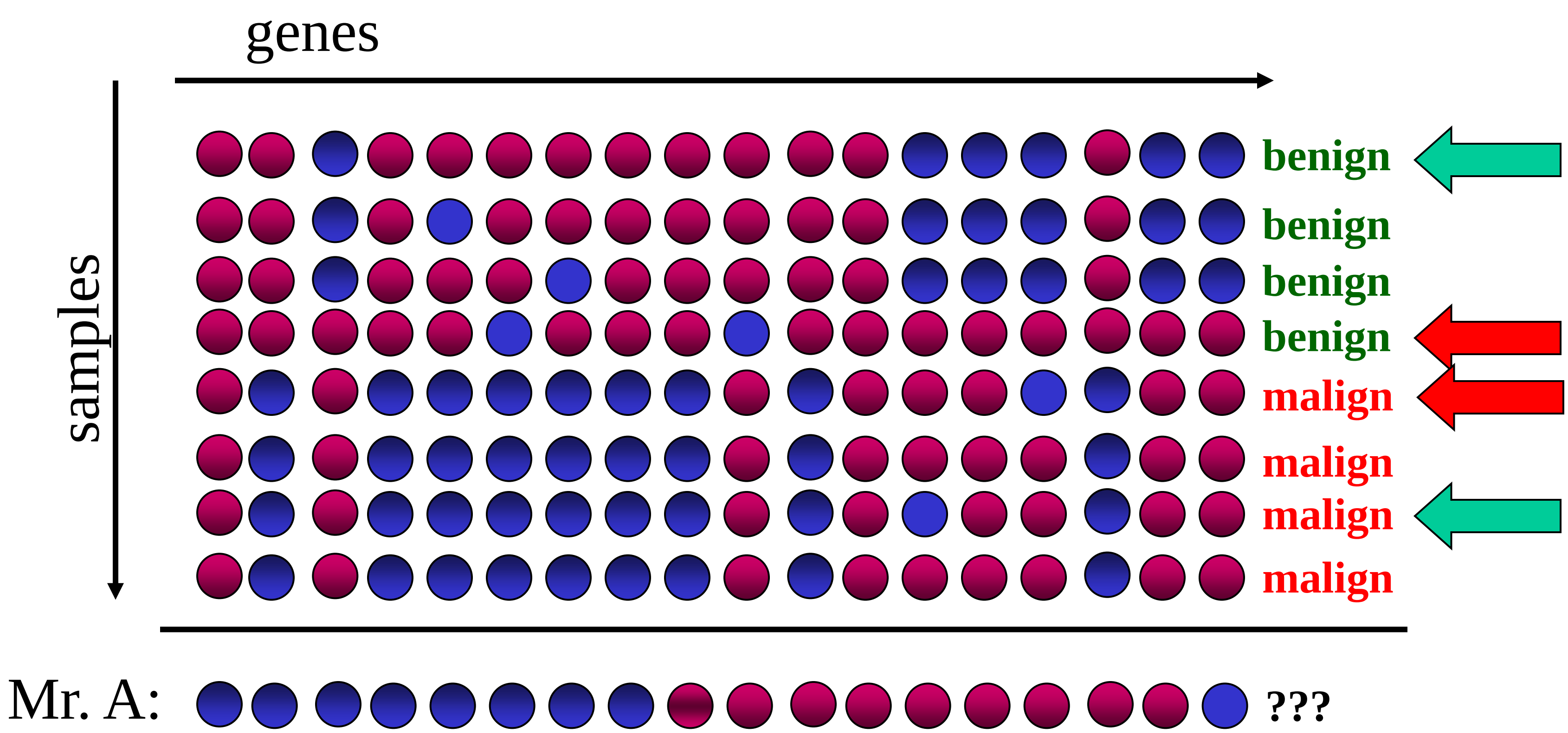
NUS
National University of Singapore

Some Patient Samples



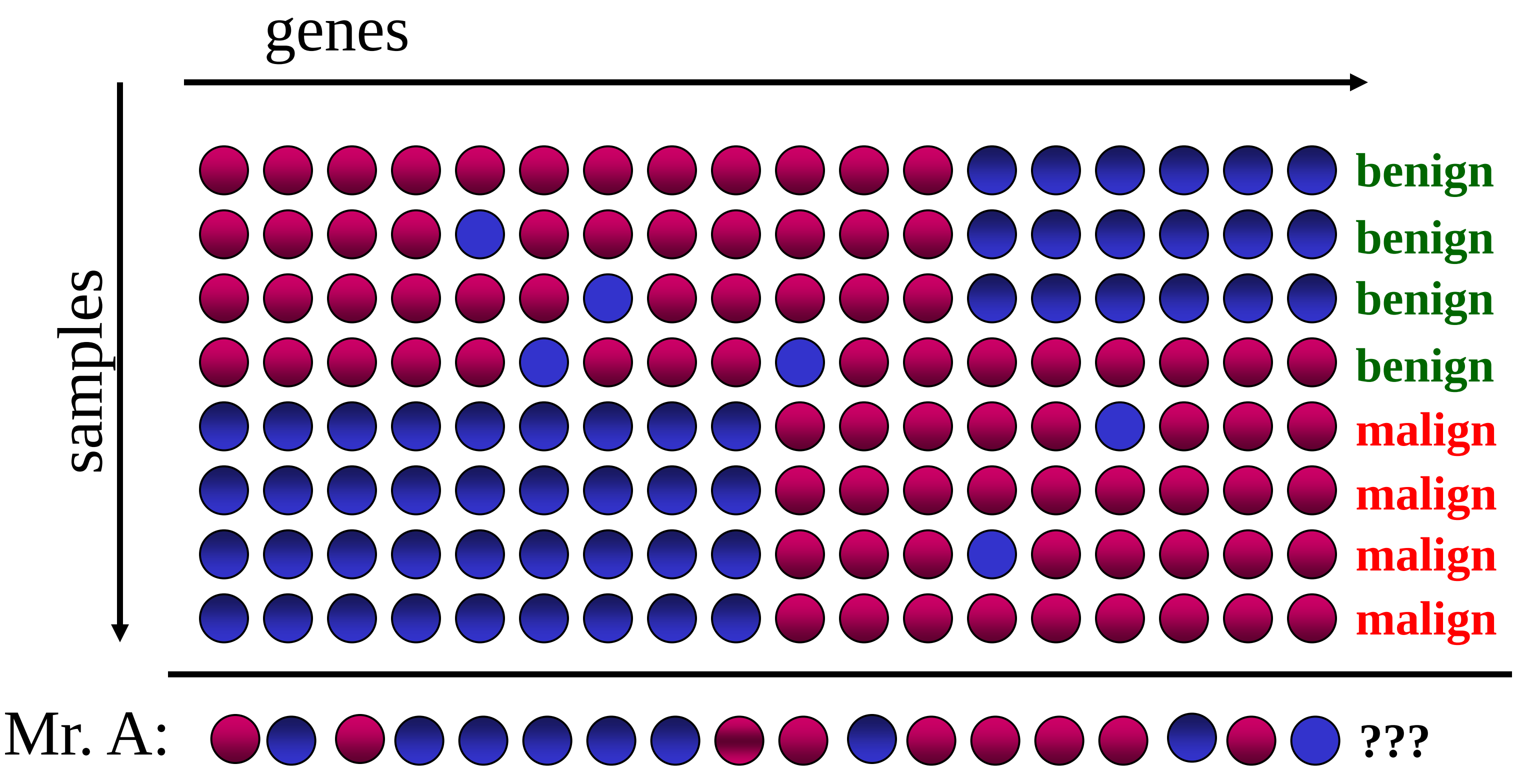
- **Does Mr. A have cancer?**

Let's rearrange the rows…

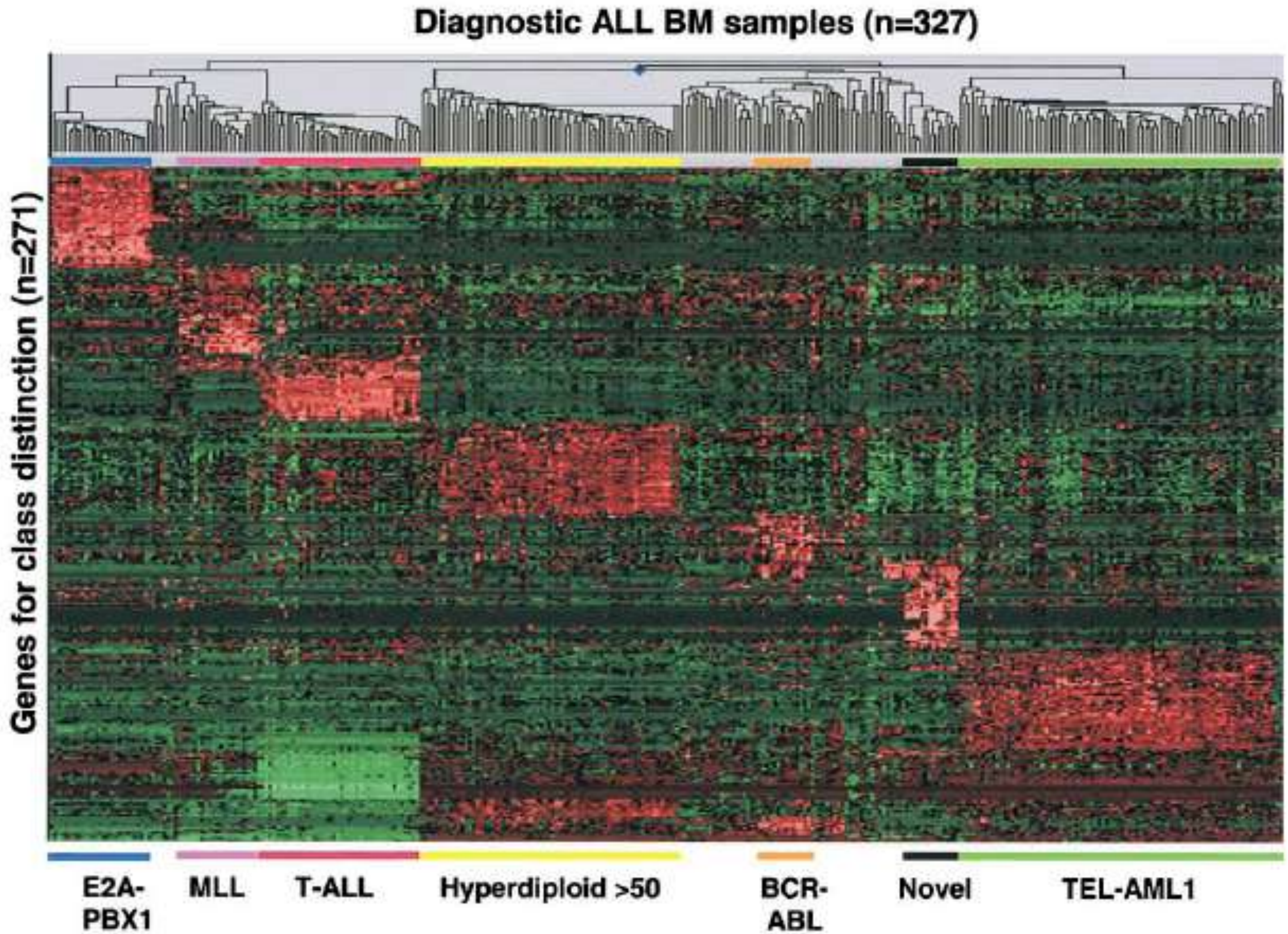


- **Does Mr. A have cancer?**

and the columns too…



Invariant Profile of Leukemia Subtypes



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

Exploit Invariant Gene Expr Profiles

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

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

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

- Total US$98.4m/yr

⇒ Save US$51.6m/yr, compared to applying intermediate-intensity treatment to everyone

Yeoh et al, Cancer Cell 2002

How to make computers safer?

National University of Singapore

COMPUTERWORLD An IDG company

RSA: Microsoft on 'rootkits': Be afraid, be very afraid

Rootkits are a new generation of powerful system-monitoring programs

News Story by Paul Roberts

FEBRUARY 17, 2005 (IDG NEWS SERVICE) - Microsoft Corp. security researchers are warning about a new generation of powerful system-monitoring programs, or "rootkits," that are almost impossible to detect using current security products and could pose a serious risk to corporations and individuals…….the only reliable way to remove kernel rootkits is to completely erase an infected hard drive and reinstall the operating system from scratch……

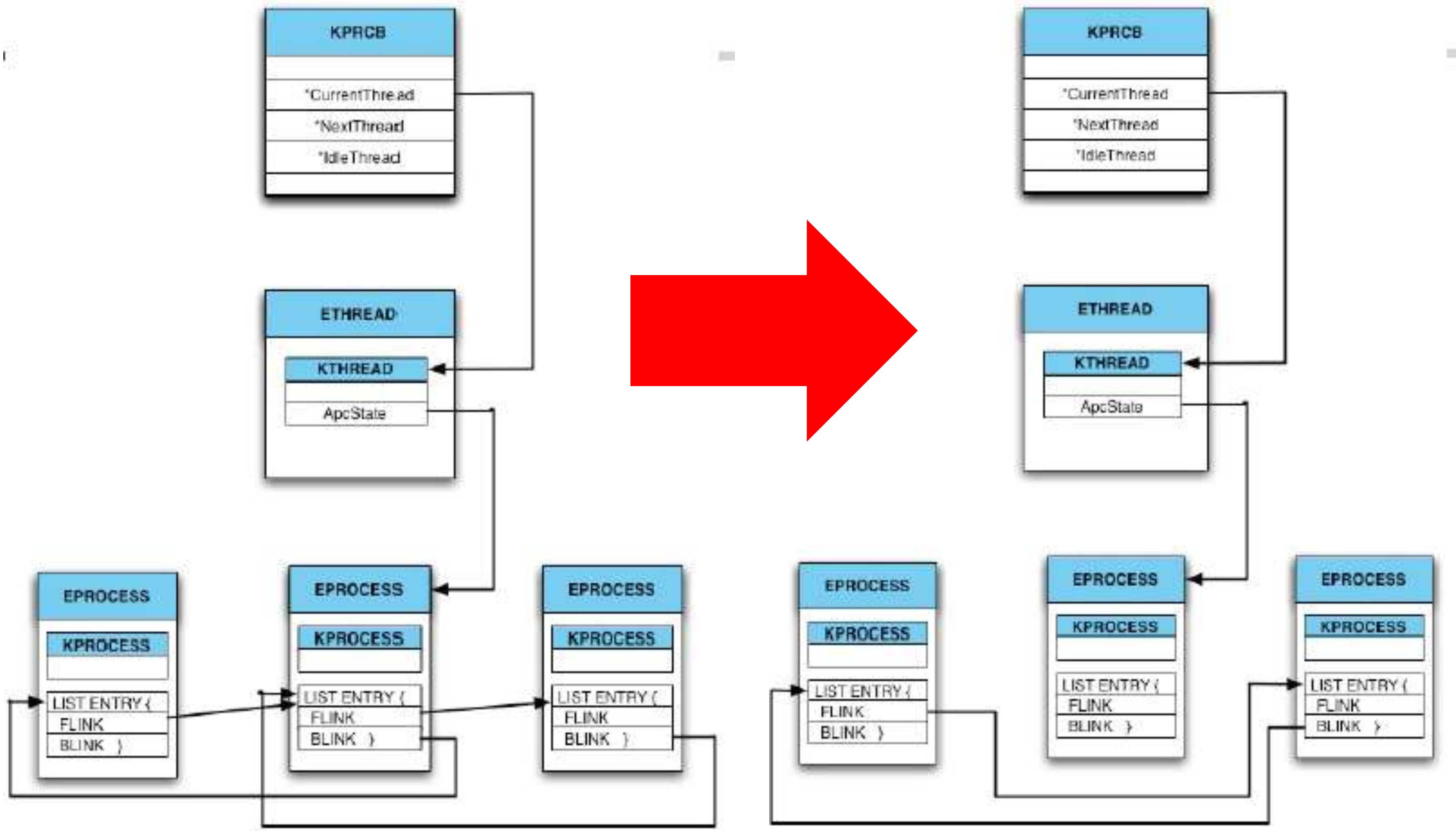
Credit: Bill Arbaugh

Rootkit Problem

- **Traditional rootkits**
  - Modify static scalar invariants in OS
    - **kernel text**
    - **interrupt table**
    - **syscall table**
- **Modern rootkits**
  - Direct Kernel Object Manipulation (DKOM)
  - Rather than modify scalar invariants in OS, dynamic data of kernel are modified to:
    - **Hide processes**
    - **Increase privilege level**

Credit: Bill Arbaugh

Hiding a window process



Semantic integrity

- **Current integrity monitoring systems focus on the scalar / static nature of the monitored data**
  – Don't work for non-scalar / dynamic 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!

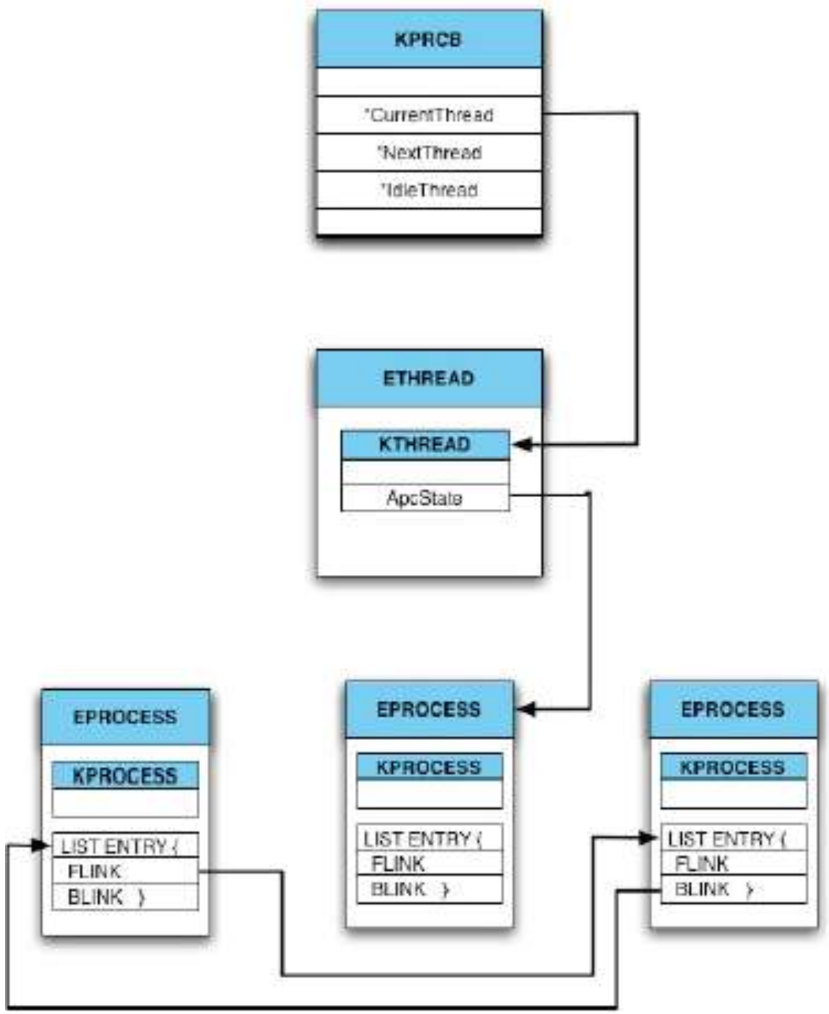
Credit: Bill Arbaugh

DKOM Example

- **Semantic integrity predicate (ie., dynamic invariant) is**

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

⇒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 Gate

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

Computer Science is no more about programming than Biology/Chemistry is about Petri dish & test tube