Computing is no more about programming than biology is about test tubes

Wong Limsoon
Golden thread of science

Science is characterized by

• Observing an invariant
• Proving that it is true, i.e., a law
• Exploiting it to solve problems **logically**
Three types of logical inferences

- **Induction**
  - Socrates is a man
  - Socrates is mortal
  \[ \Rightarrow \text{All men are mortal, provided there is no counter example} \]

- **Deduction**
  - All men are mortal
  - Socrates is a man
  \[ \Rightarrow \text{Socrates is mortal} \]

- **Abduction**
  - All men are mortal
  - Socrates is mortal
  \[ \Rightarrow \text{Socrates is a man, provided there is no other explanation of Socrates’ mortality} \]

And two simple tactics

- **Fixing violation of invariants**
- **Guilt by association**
INVARIANT & SCIENCE
• 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?
You can always 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

⇒ Parity of green beans is invariant

⇒ Bean left behind is green iff you start with odd # of green beans
• What have we just seen?

• Problem solving by (deductive) 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
- 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!

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

REMOVING NOISE FROM PPI EXPERIMENTS
Protein-protein interaction detection

• Many high-throughput assays for PPIs

Generating *large amounts* of expt data on PPIs can be done with ease

• But …

High-throughput approaches sacrifice quality for *quantity*: (a) limited or biased coverage: *false negatives*, & (b) high error rates: *false positives*
### Noise in PPI networks

<table>
<thead>
<tr>
<th>Experimental method category</th>
<th>Number of interacting pairs</th>
<th>Co-localization (%)</th>
<th>Co-cellular-role (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All: All methods</td>
<td>9347</td>
<td>64</td>
<td>49</td>
</tr>
<tr>
<td>A: Small scale Y2H</td>
<td>1861</td>
<td>73</td>
<td>62</td>
</tr>
<tr>
<td>A0: GY2H Uetz et al. (published results)</td>
<td>956</td>
<td>66</td>
<td>45</td>
</tr>
<tr>
<td>A1: GY2H Uetz et al. (unpublished results)</td>
<td>516</td>
<td>53</td>
<td>33</td>
</tr>
<tr>
<td>A2: GY2H Ito et al. (core)</td>
<td>798</td>
<td>64</td>
<td>40</td>
</tr>
<tr>
<td>A3: GY2H Ito et al. (all)</td>
<td>3655</td>
<td>41</td>
<td>15</td>
</tr>
<tr>
<td>B: Physical methods</td>
<td>71</td>
<td>98</td>
<td>95</td>
</tr>
<tr>
<td>C: Genetic methods</td>
<td>1052</td>
<td>77</td>
<td>75</td>
</tr>
<tr>
<td>D1: Biochemical, <em>in vitro</em></td>
<td>614</td>
<td>87</td>
<td>79</td>
</tr>
<tr>
<td>D2: Biochemical, chromatography</td>
<td>648</td>
<td>93</td>
<td>88</td>
</tr>
<tr>
<td>E1: Immunological, direct</td>
<td>1025</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>E2: Immunological, indirect</td>
<td>34</td>
<td>100</td>
<td>93</td>
</tr>
<tr>
<td>2M: Two different methods</td>
<td>2360</td>
<td>87</td>
<td>85</td>
</tr>
<tr>
<td>3M: Three different methods</td>
<td>1212</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>4M: Four different methods</td>
<td>570</td>
<td>95</td>
<td>93</td>
</tr>
</tbody>
</table>

Sprinzak et al., *JMB*, 327:919-923, 2003

Large disagreement betw methods

- **High level of noise**
  ⇒ Need to clean up
Time for Exercise #1

Can you think of things a biologist can do to remove PPIs that are likely to be noise?

De-noising PPI networks using Reproducibility

- A PPI reported in several independent experiments is more reliable than those reported in only one experiment.

Good idea. But you need to do more expts ➔ More time & more $ has to be spent.

\[
r_{u,v} = 1 - \prod_{i \in E_{u,v}} (1 - r_i)
\]

- \(r_i\) is reliability of expt source i,
- \(E_{u,v}\) is the set of expt sources in which interaction betw u and v is observed.
De-noising PPI networks using localization coherence

• Two proteins should be in the same place to interact. Agree?

Good idea. But the two proteins in the PPI you are looking at may not have localization annotation.
Time for Exercise #2

Do you really need to know where two proteins are, in order to know whether they are in the same place? If not, how?
Topology of neighbourhood of real PPIs

• Suppose 20% of putative PPIs are noise
⇒ ≥ 3 purple proteins are real partners of both A & B
⇒ A and B are likely localized to the same cellular compartment (Why?)
⇒ A and B are more likely PPI than not
Czekanowski-Dice distance

Given a pair of proteins \((u, v)\) in a PPI network

\(N_u = \) the set of neighbors of \(u\)
\(N_v = \) the set of neighbors of \(v\)

\[
CD(u,v) = \frac{2 | N_u \cap N_v |}{| N_u | + | N_v |}
\]

See also Liu et al. (Bioinformatics 2009, 25:1891-1897) for a simple modification of CD to make it more robust for biological & power law-like networks
Identifying false-positive PPIs

- CD-distance and its variations correlate very well with functional homogeneity and localization coherence

Cf. ave localization coherence of protein pairs in DIP < 5%
ave localization coherence of PPI in DIP < 55%

The triumph of logic

Two proteins should be in same place to interact

Impact:
PPI networks can be cleansed based on topological info, w/o needing location etc info on proteins

\[ CD(u,v) = \frac{2 | N_u \cap N_v |}{| N_u | + | N_v |} \]
Deduction / induction

IDENTIFYING HOMOLOGOUS PROTEINS
A protein is a ...

- A protein is a large complex molecule made up of one or more chains of amino acids
- Proteins perform a wide variety of activities in the cell
In the course of evolution…
Time for Exercise #3

Let \( a = \) AFP HQH RVP  
Let \( b = \) PQV YNI MKE

Suppose each generation differs from the previous by 1 residue

What is the average difference between the 2\(^{nd}\) generation of \( a \)?

What is the average difference between the 2\(^{nd}\) generation of \( a \) and \( b \)?
In the course of evolution...

\[ a = \text{AFP HQH RVP} \]
\[ b = \text{PQV YNI MKE} \]

Each gen differs from its parent by 1 residue

Each 2\textsuperscript{nd}-gen of \( a \) differs from \( a \) by 2 residues and two 2\textsuperscript{nd}-gen of \( a \) differ by at most 4 residues

\( a \) and \( b \) differ in 9 residues

Each 2\textsuperscript{nd}-gen of \( b \) differs from \( b \) by 2 residues and so differs from \( a \) by at least 7 residues; thus each 2\textsuperscript{nd}-gen of \( b \) differs from each 2\textsuperscript{nd}-gen of \( a \) by at least 5 residues
The triumph of logic

In the course of evolution...

Two proteins (not) inheriting their function from a common ancestor (don’t) have very similar amino acid sequences.
Abduction

PROTEIN FUNCTION PREDICTION
Function assignment to a protein seq

SPSTNRKYPPLPVDKLEEENRRMADDNKLFREEFNALPACPIQATCEAAASKEENKEKNRYVNILPYDHRTVHLTPVEGVPDSDYINASFINGYQEKNKFIAAQAQGPKKEETVNDFWRMIWEQNTATIVMVTNLKERKECKCAQYWDQGCWYTGNVRSVEDVTVLVDYTVRKFCIQQVGDVTNRKPQRLITQFHFTSWPDGFVPFTPIGMLKFLKKVKACNPQYAGAIVVHCSAGVGRTGTFTVVIDAMLDMHSERKVDVFVGFSRIRAQRCQMVTDMQYVFITYQALLEHYLYGDTELEVT

• How do we attempt to assign a function to a new protein sequence?
Time for Exercise #4

How can we guess the function of a protein?
Abductive reasoning

• Law: Two proteins (not) inheriting their function from a common ancestor (don’t) have very similar amino acid sequences

• Observation: Proteins X and Y are very similar in their sequence

• Abduction: Proteins X and Y are descended from the same ancestor and inherit their function from this ancestor

⇒ Proteins X and Y have a common function
Guilt by association

Compare $T$ with seqs of known function in a db

Assign to $T$ same function as homologs

Confirm with suitable wet experiments

Discard this function as a candidate
Earliest research in seq comparison

Source: Ken Sung

- Doolittle et al. (*Science*, July 1983) searched for platelet-derived growth factor (PDGF) in his own DB. He found that PDGF is similar to v-sis oncogene

```
PDGF-2  1  SLGSLTIAEPAMIAECKTREEVFCICRRL?DR??  34
p28sis 61  LARGKRSLSLSVAEPAMIAECKTRTEVFIESRRLIDRTN  100
```
Violation of invariant

MAKING COMPUTER SYSTEMS MORE SECURE
RSA: Microsoft on 'rootkits': Be afraid, be very afraid
Rootkits are a new generation of powerful system-monitoring programs

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
Hiding a window process
Semantic integrity

• Earlier integrity monitoring systems focus on the scalar / static nature of the monitored data
  – Don’t work for non-scalar / dynamic data

• Current systems rely on 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

⇒ kHIVE (contains 20k other predicates)
• What have we just seen?

• Maintain computer safety by checking violation of invariants!
Violation of invariant

IMPROVING 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>$$$</td>
</tr>
<tr>
<td>2</td>
<td>Mark Hamill</td>
<td>Fox</td>
<td>Star Wars</td>
<td>$$$</td>
</tr>
<tr>
<td>3</td>
<td>Harrison Ford</td>
<td>Fox</td>
<td>Star Wars</td>
<td>$$$</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>
</tr>
</thead>
<tbody>
<tr>
<td>Mighty Ducks</td>
<td>1991</td>
<td>104</td>
<td>Color</td>
</tr>
<tr>
<td>Wayne’s World</td>
<td>1992</td>
<td>95</td>
<td>Color</td>
</tr>
<tr>
<td>Star Wars</td>
<td>1977</td>
<td>124</td>
<td>Color</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>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>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>
Exercise #5

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>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>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?

<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>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
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 \(B_1, \ldots, B_m\)
  - 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

<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>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

- **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

<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>1991</td>
<td>104</td>
<td>Color</td>
<td>Disney</td>
<td>Emilio Estevez</td>
</tr>
</tbody>
</table>

- **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?

<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>1991</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 {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
  - \(R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)\)
  - \(R_2(A_1, \ldots, A_n, C_1, \ldots, C_h)\)
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
Induction / fixing violated invariants

INFERRING KEY MUTATIONS: WHY SOME PTP IS INEFFICIENT
Protein tyrosine phosphatase

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

Sequence from a typical PTP

>gi|00000|PTPA-D2
EEFKKLTSIKIQRNDKMRTGNLPAKMKKNRVLQIIPYFNRVIIPVRGEEENTDYVNASF
IDGYRQKDSYIASQGPLLHTIEDFWRMIWEWKCSIVMLTELEERQEQKCAQYUPEDSGLV
SYGDITIVELKKEECESYTVRDLVTNTRENKSRQIRQFHFHWPEVGIPSDGKGMISII
AAVQKQQQQSGNHPITVHCASAGAGRTGTFCALSTVLERVKAEGILDVFQTVKSLRLQRPH
MVQTLEQYEFcyKVvQyEYIDAFSDYANFK
Exercise #6

Protein tyrosine phosphatase

Sequence from a typical PTP

>gi|00000|PTPA-D2
EEEFKKTINSKQNDKRTGNLPANMKKNRLQIIPIPYEFNRVIIPVKRGEENTDYVNASF
IDGVRQKDSYIASQGPLLHTIEDFURMIWEUKCSIVMLTELEERGQEKAQYUPSDDLGV
SYGDITVELKKEECESYTVRDLLVTNRTENKSRQIRQFHFGWPEVGIIGPSDGKGMISII
AAVQRQQQQSGNHPITVHC3AGAGRTGTFCAL3TVERVKAEGILDVFQTVK3LRLOQPH
MVQTLEQYEFCYKVQYEYIDAFAFSDYANFK

- Some PTPs are much less efficient than others

How do you figure out which mutations cause the loss of efficiency?
Some sites are impt for PTP function
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

- 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 exps.

- **Wet experiments confirm the predictions**
  - Mutate $D \rightarrow E$ in $D_1$
    - i.e., check if $D \rightarrow E$ can cause efficiency loss
  - Mutate $E \rightarrow D$ in $D_2$
    - 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!
The triumph of logic

- Induction/hypothesis: A site that is critical for PTP efficiency is present in all efficient PTPs and absent in all inefficient PTPs

- Observation: A site X is present in all efficient PTPs and absent in all inefficient PTPs

- Abduction: Site X is critical for PTP efficiency
Bioengineering more efficient PTPs

• Replace an inefficient PTP in the organism by an efficient version
  – Mutate E → D in D2

• What have we just seen?

• Create a more efficient PTP by fixing a violated invariant!
SUMMARY
What have we learned?

• Three types of logical reasoning

• Invariant is a fundamental property of many problems

• Tactics of logical problem solving
  – Problem solving by logical reasoning on invariants
  – Problem solving by rectifying/monitoring violation of invariants
  – Guilt by association of invariants