THE ROLE OF SAT, CONSTRAINT PROGRAMMING, AND LOGIC PROGRAMMING IN COMPUTATIONAL BIOLOGY

Agostino Dovier

Dept of Mathematics, Computer Science, and Physics
University of Udine
Udine (Italy)

Melbourne, Sept 1, 2017
Things that unify us:

- Declarative approach to
- Problem Solving
- Solving methods
- DPLL
- Plus search strategies, conflict driven learning, heuristics,
  (constrained) local search, automated configuration
  (portfolio), parallelism, . . .

Things that divide us:

- ?
A Computing Procedure for Quantification Theory*

MARTIN DAVIS
Rensselaer Polytechnic Institute, Hartford Division, East Windsor Hill, Conn.

AND

HILARY PUTNAM
Princeton University, Princeton, New Jersey

6. An Example

P. C. Gilmore\textsuperscript{18} tested his refutation-procedure on a number of formulas, including the following one:

\[(Ex)(Ey)(z)\left((F(x, y) \to (F(y, z) & F(z, z))) \\& ((F(x, y) & G(x, y)) \to (G(x, z) & G(z, z)))\right)\] (1)

We have selected this example for purposes of comparison because (a) it is not so long as to make hand computation immediately impractical (e.g., it is already in prenex form, and the matrix can easily be put into conjunctive normal form); yet (b) Gilmore's procedure did not lead to a refutation although an IBM 704 was employed for 21 minutes.

Our procedure, on the other hand, did lead to a refutation in under a half-hour of hand computation!

A Computing Procedure for Quantification Theory*

MARTIN DAVIS

Rensselaer Polytechnic Institute, Hartford Division, East Windsor Hill, Conn.

AND

HILARY PUTNAM

Princeton University, Princeton, New Jersey

6. An Example

P. C. Gilmore\textsuperscript{18} tested his refutation-procedure on a number of formulas, including the following one:

\begin{equation}
(\exists x)(\exists y)(z)((F(x, y) \rightarrow (F(y, z) \& F(z, z))) \& ((F(x, y) \& G(x, y)) \\
\rightarrow (G(x, z) \& G(z, z))))
\end{equation}

We have selected this example for purposes of comparison because (a) it is not so long as to make hand computation immediately impractical (e.g., it is already in prenex form, and the matrix can easily be put into conjunctive normal form); yet (b) Gilmore's procedure did \textit{not} lead to a refutation although an IBM 704 was employed for 21 minutes.

Our procedure, on the other hand, \textit{did} lead to a refutation in under a half-hour of \textit{hand} computation!

The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule. He noted that it was easy to save the current environment on tape, and retrieve it on backtracking. As for the other systems with splitting rules, this led to depth-first search. Thus, with the new program, instead of clause addition there was clause removal. Interspersed with applications of Rules I and II, the program recursed on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropriate
The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule. He noted that it was easy to save the current environment on tape, and retrieve it on backtracking. As for the other systems with splitting rules, this led to depth-first search. Thus, with the new program, instead of clause addition there was clause removal. Interspersed with applications of Rules I and II, the program recursed on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropri-
The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule. He noted that it was easy to save the current environment on tape, and retrieve it on backtracking. As for the other systems with splitting rules, this led to depth-first search. Thus, with the new program, instead of clause addition there was clause removal. Interspersed with applications of Rules I and II, the program recursed on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropri-
The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule, and retrieve it

**III Rule for eliminating atomic formulas.** $(A \lor p) \land (B \lor \neg p) \land R$ is unsatisfiable iff $(A \lor B) \land R$ is unsatisfiable.

Interspersed with applications of Rules I and II, the program recurred on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropri-
The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule. He noted that it was easy to save the current environment on tape, and retrieve it on backtracking. As for the other systems with splitting rules, this led to depth-first search. Thus, with the new program, instead of clause addition there was clause removal. Interspersed with applications of Rules I and II, the program recursed on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropri-
The task of implementing the DP procedure was split. Logemann undertook the parsing of the CNF formula, entered in Polish notation, the formula preparation having been done by hand. Logemann also handled the structuring of the set of clauses, while Loveland took on the testing of the clause set for consistency. The program was written in SAP, the Symbolic Assembler Program, the assembly language for the IBM 704. After the first runs, which quickly saturated the 32,768 36-bit words of available storage, George Logemann suggested that Rule III be replaced with a splitting rule. He noted that it was easy to save the current environment on tape, and retrieve it on backtracking. As for the other systems with splitting rules, this led to depth-first search. Thus, with the new program, instead of clause addition there was clause removal. Interspersed with applications of Rules I and II, the program recursed on Rule III*, saving the environment on tape for backtracking. This solved the space issue, at least until the input clause set overwhelmed the memory. Now very important, but not emphasized at the time, was the easy definition of a satisfying assignment should there be one. It is unclear why “turning off” and then “turning on” various clauses and literals on backtracking was not pursued, as writing to tape is a slow operation. Using the appropriate list structures, top level routines existed that quickly deleted all occurrences of a specified literal and eliminated the appropriate
Prop-Labeling Tree

- P
- P'
- P'', X=0
- P'', X=1
- P'''

Constraints and Propagation

Backtracking
ASP solver

- **P**
- **P'**
- **P'', X=f**
- **P'', X=t**
- **P''''**

Backtracking

- Expand
- Lookahead

Common (pre) history
A unifying special issue
The WCB experiment

Introduction
Genomics and Systems Biology
Structural Bioinformatics
Towards the conclusions

Agostino Dovier
SCALP and BIO
**The three communities**

- ICLP (1982)
- CLP (1986)
- ASP (1999)
- CP (1995)
- SAT (1996)
Michael J. Maher. Contractibility for open global constraints.

Mutsunori Banbara, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub. Clingcon: The next generation.


Andrew Reynolds, Cesare Tinelli, Clark Barrett. Constraint solving for finite model finding in SMT solvers.

Yuliya Lierler, Benjamin Susman. On relation between constraint answer set programming and satisfiability modulo theories.

Marcello Balduccini, Daniele Magazzeni, Marco Maratea, Emily LeBlanc. CASP solutions for planning in hybrid domains.

Tiep Le, Tran Cao Son, Enrico Pontelli, William Yeoh. Solving distributed constraint optimization problems using logic programming.

(Edited by AD)
Workshops on Constraint Based Methods for Bioinformatics

http://clp.dimi.uniud.it/wcb/

Agostino Dovier
SCALP and BIO
Biology is an incredible source of challenging problems for computer science

Problems are often hidden or vaguely defined and emerge only after several cycles of feedback with biologists, physicists, chemists, etc

Solving one of these problems can be of unpredictable importance for life sciences and medicine
Introduction

Bioinformatics

Bioinformatics deals with modeling and solving problems, analyzing and filtering data, from biology and related life sciences.

- Data availability is huge.
- Data is affected by experimental errors.
- Computer science tools should help in analyzing and filtering.
Bioinformatics applications are divided in three categories:

1) **Support Infrastructure for Analysis and Experiments**

Applications of computational methods for automated environments for workflow management, description and annotation of experiments, minimal reporting requirements, ...

2) **Polynomial Time Solvable Problems**

The input size is large: e.g. string matching problems over DNA sequences.

3) **Intractable Problems**

NP-complete or worse problems. Mainly covered by this lecture.
Areas of Bioinformatics

1. **Genomics.** Study of the genomes. Huge amount of data, fast algorithms (not always), limited to sequence analysis.

   ```
   ... G A T C T G T A C T G A G T ...
   ... G A T C T G T A C T G A A T ...
   ```

2. **Structural Bioinformatics.** Study of the folding process of bio-molecules. Less structural data than sequence data available.

3. **Systems Biology.** Study of complex interactions in biological systems. High level of representation.
Why SCALP?

- Models need to be designed/corrected/revised. Logic and Constraint Programming provide the level of elaboration-tolerance to support model modifications and incremental addition of new knowledge. Declarative formalism is elegant and concise!

- Linear (Integer) Programming is not enough (in particular for modeling energy models)

- Constraint Programming is perfect for some applications. For others it lacks in some KR capabilities (e.g., un-natural solutions due to lacking of a stability notion)

- SAT, MAXSAT fit perfectly with some of the emerging problems.

- Model execution can be later speed-up (symmetry breaking, search heuristics, learning, constraint based local search, parallelism, portfolio, ad-hoc global constraints, etc)

- Inductive Logic Programming can be used where input is huge and/or affected by many errors.
DNA (DeoxyriboNucleic Acid) is characterized by a string of nucleotides: A, C, G, and T (Adenine, Cytosine, Guanine, Thymine)

- Given a sequence $s \in \{A, C, G, T\}^*$ the complementary sequence $\bar{s}$ is deterministically obtained by reversing $s$ and substituting $A \leftrightarrow T$ and $C \leftrightarrow G$

- $s$ and $\bar{s}$ fold together forming the famous double helix

...ATGCGCTAGCTCATT...  
...AATAGCTAGGCAAT...
Some fragments of the DNA, called Genes, encode proteins.

The set of all genes of an individual is called Genome.

Human Genome Project: ≃ 16–20K protein-coding genes in human DNA.

Differences of some nucleotides in the same gene characterize a property of an individual w.r.t. another.
Haplotypes and Genotypes

- Genes are packaged in bundles called chromosomes. (Chromosomes are therefore regions of DNA)
- In diploid organisms (like humans) there are two copies of (almost all) chromosomes (23 in humans). Each pair is made of an inherited chromosome from the father and another from the mother.
- A haplotype is a DNA sequence that has been inherited from one parent.
- A genotype is a pairing of two corresponding haplotypes.
SNPs

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>A</td>
<td>T</td>
</tr>
</tbody>
</table>

In some typical positions, bases are subject to mutations.

Almost always, there is a Single Nucleotide Polymorphism (SNP):

Mutations are $C \leftrightarrow T$ and $A \leftrightarrow G$
Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

```
... G A T C T G T A C T G A G T ...
... G A T C T G T A C T G A A T ...
```

In some typical positions, bases are subject to mutations. Almost always, there is a Single Nucleotide Polymorphism (SNP):

Mutations are $C \leftrightarrow T$ and $A \leftrightarrow G$
SNPs

Each person inherits two haplotypes (from the mother and from the father) for most regions of the genome.

\[
\begin{array}{cccccccccccc}
\ldots & G & A & T & C & T & G & T & A & C & T & G & A & G & T & \ldots \\
\ldots & G & A & T & C & T & G & T & A & C & T & G & A & A & T & \ldots \\
\uparrow & \uparrow & & \uparrow & & * & \uparrow & * \\
\end{array}
\]

In some typical positions, bases are subject to mutations. Almost always, there is a Single Nucleotide Polymorphism (SNP):

Mutations are $C \leftrightarrow T$ and $A \leftrightarrow G$
Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

\[
\begin{array}{cccccccccccc}
G & A & A & T & C & T & T & C & G & T & A & C & T & G & A & G & T \\
G & A & T & C & T & T & C & G & T & A & C & T & G & A & A & T \\
\end{array}
\]

Let us focus on the SNPs:

\[
\begin{array}{cccc}
A & C & T & G \\
A & C & T & A \\
\end{array}
\]

Mutations:

<table>
<thead>
<tr>
<th>C $\leftrightarrow$ T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A $\leftrightarrow$ G</td>
</tr>
</tbody>
</table>

0 0 1 1
0 0 1 0
0 0 1 0
0 0 1 2

The genotype is set to 2 if there is a mismatch.
What we have

Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

| G | A | A | T | C | T | T | C | G | T | A | C | T | G | A | G | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | T | C | T | T | C | G | T | A | C | T | G | A | A | T |

Let us focus on the SNPs:

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>T</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Mutations:

- \( C \leftrightarrow T \)
- \( A \leftrightarrow G \)

But this is the situation of complete knowledge. In practice, we can detect a mismatch but not its single components.

The genotype is set to 2 if there is a mismatch.
What we have

Each person has two haplotypes (from the mother and from the father) for most regions of the genome:

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>A</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>G A A</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Let us focus on the SNPs:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A C T A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

But this is the situation of complete knowledge. In practice, we can detect a mismatch but not its single components.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The genotype is set to 2 if there is a mismatch.
Looking for an explanation

The smaller the better
Looking for an explanation

The smaller the better

Agostino Dovier  SCALP and BIO
Looking for an explanation

The smaller the better
Given a set $G$ of (equal length) genotypes (e.g., $G = \{0012, 0211, 0212\}$)

We look for a set $H$ of haplotypes that “explains” $G$ (e.g., $H = \{0010, 0011, 0111\}$ — with $|H| = 2|G|$ is trivial).

Namely, for each $g \in G$ there are $h_1, h_2 \in H$ such that $h_1 \oplus h_2 = g$ where the $\oplus$, bitwise, is $0 \oplus 0 = 0$, $1 \oplus 1 = 1$, $0 \oplus 1 = 2$, $1 \oplus 0 = 2$

Since genotypes are introduced in evolution, it is reasonable to look for sets of haplotypes of minimum cardinality explaining the known genotypes (Haplotype Inference by Pure Parsimony—HIPP)
Encoding HIPP is easy for SCALP people

- in Integer Linear Programming (Lancia et al, INFORMS J. on Comp. 16(4) 2004; Cussens WCB 10; 0-1 ILP Graça et al WCB 08)
- in ASP (Erdem et al, LPNMR 2009)
- in Constraint Programming (FD \{0, 1\} and \{0, 1, 2\}—see ACP summer school 2012)
- in SAT (Lynce and Marquez-Silva, SAT 2006)
- in (partial) MAXSAT (He et al, Bioinformatics 26(12), 2010)
- in Local Search (Di Gaspero et al WCB 08)

A link for benchmarks:
http://www.stats.ox.ac.uk/~marchini/phaseoff.html
But it is still hard to solve it!

Investigating *sets* of genotypes for a population helps in understanding the relationships between SNPs and physical features as well as medical information.

In human genetics, genome-wide association studies collect genotypes in

- **thousands** of individuals at
- between *200,000–5,000,000* SNPs (using microarrays)

Haplotype estimation methods are used in the analysis of these datasets and allow genotype imputation (i.e., statistical inference of unobserved genotypes)

Can be used in forensic investigation (even if simpler and faster techniques are used there to filter out possible culprits)
A phylogeny describes evolutionary relationships among entities.

More reliable than pattern matching

Provide insights into

**SPECIES TREES**: relationships among organisms
**GENE TREES**: history and evolution of genes

Applied outside biology: Epidemiology (development of pandemics, patterns of disease transmission), Ecology (population structure, population at risk), Indo-European languages, Forensics (e.g., detect smuggled animals)
A phylogeny is a tree, which estimates the historical connections between species or genes that they carry.

The tips of the tree are called Taxonomic Units or taxa — orders, species, populations, presence of a tail, etc.

The goal of phylogenetic analysis is to recover bifurcating trees.
Reconstruction

© 2008 Leonard Eisenberg. All rights reserved.

All major and many of the minor living branches of life are shown on this diagram, but only a few of those that have gone extinct are shown. Example: Dinosaurs - extinct.
**Example (from Linguistic — Erdem11)**

**How many binary trees with 5 leaves?**

<table>
<thead>
<tr>
<th>Character</th>
<th>English</th>
<th>German</th>
<th>French</th>
<th>Spanish</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hand&quot;</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>&quot;Father&quot;</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Taxa:** \{English, German, French, Spanish, Italian\}

**Characters:** \{Hand, Father\}

**Values (aka, states):** \(D = \{1, 2\}\)
A character $c$ is **compatible** with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character "Hand" is compatible with the above tree.
A character $c$ is compatible with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character Hand is compatible with the above tree.
A character $c$ is compatible with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character Hand is compatible with the above tree.
A character $c$ is compatible with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character Father is incompatible with the above tree.
A character $c$ is compatible with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character Father is incompatible with the above tree.
A character $c$ is **compatible** with a phylogeny (i.e., a labeled tree) if all the taxa that present the same value for $c$ are connected by a subtree.

Character Father is incompatible with the above tree.
**k-INCOMPATIBILITY PROBLEM (k-IP)**

Given sets \( L \) (taxa/leaves), \( C \) (characters), and \( D \) (states), a function \( f : L \times C \rightarrow D \), and \( k \in \mathbb{N} \), decide the existence of a phylogeny \((V, E, L, C, D, f)\) with at most \( k \) incompatible characters.

The problem is NP-complete. Rough upper bounds:

- The number of binary trees with \( n \) leaves is
  \[
  \text{Cat}(n - 1) = \frac{(2(n-1))!}{n!(n-1)!}
  \]

- Leaves can be labelled in \( n! \) ways

An alternative formulation is based on a precomputation of all the 4-tuples of taxa.
**k-INCOMPATIBILITY PROBLEM (k-IP)**

Given sets $L$ (taxa/leaves), $C$ (characters), and $D$ (states), a function $f : L \times C \rightarrow D$, and $k \in \mathbb{N}$, decide the existence of a phylogeny $(V, E, L, C, D, f)$ with at most $k$ incompatible characters.

The problem is NP-complete. Rough upper bounds:

- The number of binary trees with $n$ leaves is
  $$\text{Cat}(n - 1) = \frac{(2(n-1))!}{n!(n-1)!}$$

- Leaves can be labelled in $n!$ ways.

An alternative formulation is based on a precomputation of all the 4-tuples of taxa.
MQC - Maximum Quartet Consistency

Considering four taxa $a, b, c, d$ (all leaves), following the paths that lead to the least common ancestors in the tree, we always identify two pairs of independent paths: $(a, b), (c, d)$ or $(a, c), (b, d)$ or $(a, d), (b, c)$ (up to symmetries).

Given a precomputation of all the $\binom{n}{4}$ 4-tuples of taxa, the problem is now putting things together maximizing coherence.
**Encoding k-IP/MQC is easy for SCALP people**

- in pseudo-Boolean (Morgado and Marques-Silva WCB 08, FUIN 2010)
- in CP (Schiex et al — pedigree reconstructions — WCB 05, WCB 06, WCB 07, Constraints 2008; Moore and Prosser: The Ultrametric Constraint and its Application to Phylogenetics. WCB 06 JAIR 32, 2008)
- in Stochastic Local Search (e.g., Tria et al, Diachronica 27(2), 2010)
- in ASP, for different purposes (Pontelli et al — Phylotastic! Making tree-of-life knowledge accessible, reusable and convenient. BMC Bioinformatics 14, 2013)
Example: DINGO (Letnic et al 2014)

We can also conclusively say that the dingo is a distinctive Australian wild canid or member of the dog family in its own right, separate from dogs and wolves. The appropriate scientific classification is Canis dingo, as they appear not to be descended from wolves, are distinct from dogs and are not a subspecies.

Welcome back to your own branch of the phylogenetic tree, dingoes!
A cell contains complex systems of interacting components

E.g. small molecules, DNA, proteins

Each system can be modeled by means of networks
**Biological Networks**

- The problem is to model a network from biological knowledge
- The model has to be validated w.r.t. experimental data
- Data is incomplete, sometimes unreliable
- Models need to be modified, repaired and/or extended
- Models can guide the design of new experiments

<table>
<thead>
<tr>
<th>Molecules</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNA</td>
<td>Transcriptional regulatory network</td>
</tr>
<tr>
<td>mRNA</td>
<td>Gene regulatory network</td>
</tr>
<tr>
<td>Protein</td>
<td>Protein interaction network</td>
</tr>
<tr>
<td>Metabolite</td>
<td>Metabolic network</td>
</tr>
<tr>
<td>Heterogeneous components</td>
<td>Signaling network</td>
</tr>
</tbody>
</table>

- **transcription factor**
- **Gene**
- **A + C       AC     B + C**
- **mRNA**
- **Protein**
- **DNA**
- **Gene**
- **Metabolite**
- **Heterogeneous components**

---

**Introduction**

Genomics and Systems Biology

Structural Bioinformatics

Towards the conclusions

**Haplotype Inference**

Phylogenetics

Systems Biology
Gene Regulatory Network: Influence Graph

- Simplest type of Gene Regulatory Network
- **Nodes** are genes
- **Edges** show how a gene influence other genes
- The influence can be positive or negative

Operon Lactose in E. coli
From:
Gebser, Schaub, Thiele, Veber, 2011
Gene Regulatory Network

- Every positive node must be supported at least by one positive node and one positive entering edge or by one negative node and one negative entering edge.
- Dually for negative nodes (use rule of signs).

```
1 2 3 4 5 6 7 8
+ + + + + + + + NO (8)
+ + + + + - + - YES
+ - - - ? ? ? ? SAT
+ - - - - - + - + YES
+ - - - + + - + + YES
```
Gene Regulatory Network: Influence Graphs

Two main problems

- Checking consistency
- Repairing Networks

Easy mapping in ASP/SAT (MAXSAT in the case of repairing)

- Gebser et al. *Detecting Inconsistencies in Large Biological Networks with Answer Set Programming*. WCB 08, ICLP 2011.

Biocham is a software environment for modeling biochemical systems. (e.g., WCB 06, . . . , WCB 13) It allows the analysis and simulation of boolean, kinetic and stochastic models (using a rule-based language) and the formalization of biological properties in temporal logic (LTL/CTL). It uses CLP, SAT and other constraint-based techniques. Lots of successful experiments with real data


Model checking is also used by Parvu and Gilbert (PLOS One, 2016) to verify Biological Systems
Other SCALP contributions

(Gene) Regulatory network


- Emna Ben Abdallah et al. ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. *AMB 17*

- Fitime et al. Identification of Bifurcation Transitions in Biological Regulatory Networks using Answer-Set Programming *WCB 16* — *AMB 17*
Other SCALP contributions

Metabolic Networks

- Frioux et al. Hybrid Metabolic Network Completion (ASP with linear constraints over reals—fluto=clingo+cplex/lpsolve) LPNMR 17

Cell signaling networks

SCALP and WCB in Berlin

- Goldstein, Bockmayr: Double and multiple knockout simulations for genome-scale metabolic network reconstructions. Algorithms for Molecular Biology 10(1), 2015
- ...
RNA is a sequence of nucleotides (A,C,G,U) that (often) is just an intermediary between DNA and proteins.

DNA strands are transcribed to mRNA, in order to exit the cell’s nucleus.

Nucleotides replacement: DNA T $\rightarrow$ RNA U.
RNA Secondary Structure

- RNA folds according to favorable matchings: A–U, C–G, U~G
- The **secondary structure** is the set of its base pairings
- Secondary structure determines the 3D properties
**A** RNA sequence \( \vec{s} = s_1 s_2 \cdots s_n \) is a string in \( \{A, C, G, U\}^* \)

**A** RNA secondary structure is a (partial) **injective** function \( P \subseteq \{1, \ldots, n\}^2 \) such that

- \( (i, j) \in P \iff (j, i) \in P \)
- \( (i, j) \in P \) only if
  - \( (s_i, s_j) \in \{(A, U), (U, A), (C, G), (G, C), (U, G), (G, U)\} \)

**We are interested in a solution with maximal “good” pairs (and/or minimizing a more complex energy function)**
THE GENERAL PROBLEM IS NP-COMPLETE
ENCODING EASY FOR SCALP PEOPLE


- M. Bavarian and V. Dahl. Constraint Based Methods for Biological Sequence Analysis. WCB 05 and J. Universal Computer Science 12(11), 2006

- A. Dal Palù et al A Propagator for Maximum Weight String Alignment with Arbitrary Pairwise Dependencies. WCB 10 and CP 2010

- Alexander Bau et al RNA Design by Program Inversion via SAT Solving WCB 13
The translation phase starts from a mRNA sequence and associates a protein sequence.

Proteins are made of amino acids (20 common different types).

Amino acids are defined by letters \( \{A, \ldots, Z\} \setminus \{B, J, O, U, X, Z\} \).
The translation converts 3 RNA bases into 1 amino acid.

The translation rules are encoded in the universal code.

The code contains stop symbol and some redundant RNA triplets.
Proteins

Amino acids

- Proteins are molecules made of a linear sequence of amino acids.
- Amino acids are combined through peptide bond.

- The purple dots represent the side chains, that depend on the amino acid type
- Side chains have different shape, size, charge, polarity, etc.
- A side chain contains from 1 (Glycine) up to 18 (Tryptophan) atoms.
Proteins

Amino acids

- There are 2 degrees of freedom (black arrows) for each amino acid
- A protein with $n$ amino acids has $2n$ degrees of freedom (plus side chains)
- Typical size range from 50 to 500 amino acids
The structure prediction problem

- Given the **primary structure** of a protein (its amino acid sequence)
- For each amino acid, output its position in the space (tertiary structure of a protein)

```
A L F W K L R R ...
```

- **Secondary structures** are rigid subparts (helices, sheets) that can be “easily” predicted
**PROTEINS**

**Facts**

- Folding is **consistent** ⇒ same protein folds in the same way [Anfinsen74]
- Folding is **fast** ⇒ 1ms – 1s
- Driven by **non covalent** forces: electrostatic interactions, volume constraints, Hydrogen Bonding, van der Waals, Salt/disulfide Bridges
- Backbone is rigid, interaction with water, ions and ligands
- There is a fixed distance (3.8Å) between the $C\alpha$ atoms of consecutive aminoacids.
- There are several statistics on (bend/torsional) angles.
... and this is the hard part:

- In nature a protein has a unique/stable 3D conformation
- A cost function (that mimics physics laws) can be used to score each conformation
- Searching for the optimal score produces the best candidate is difficult (NP-complete even in extremely simplified modelings)
The protein structure prediction problem

- A first simplification (HP):
  - **Protein model**: only one atom per amino acid, only 2 classes of amino acids (hydrophobic and polar)

- A second simplification:
  - **Spatial model**: Lattices to represent amino acid positions
The protein structure prediction problem

- A first simplification (HP):
  - **Protein model**: only one atom per amino acid, only 2 classes of amino acids (hydrophobic and polar)

- A second simplification:
  - **Spatial model**: Lattices to represent amino acid positions
A simple (TOY) modeling

- The input is a list $S$ of amino acids $S = s_1, \ldots, s_n$, where $s_i \in \{h, p\}$.
- Each $s_i$ is placed on a 2D grid (square lattice) with integer coordinates, adjacent to $s_{i-1}$ and $s_{i+1}$ (a folding).
- Any pair of two amino acids can’t occupy the same position.
- If two amino acids are at distance 1, they are in contact.
- If they are both “h” (dark circles below) they contribute with -1 to the energy.
A simple (TOY) modeling

- Even in this simple setting, establishing whether there is a folding with energy $< k$ is NP-complete


- This formulation of the problem can be taught to children as a simple pencil/paper game (then suggest to play with http://fold.it/portal/).

- It is also useful to understand its empirical complexity (try modeling it. It is naive with SCALP languages). You’ll probably lose your patience for instances of length 25 (e.g., $h(pph)^8$)!

- But this is just a game ...
A simple (TOY) modeling

- Even in this simple setting, establishing whether there is a folding with energy $< k$ is NP-complete
- This formulation of the problem can be taught to children as a simple pencil/paper game (then suggest to play with http://fold.it/portal/).
- It is also useful to understand its *empirical* complexity (try modeling it. It is naive with SCALP languages). You’ll probably lose your patience for instances of length 25 (e.g., $h(pph)^8$)!
- But this is just a game . . .
3D Lattice models: Cube, FCC, Chess Knight
The Face Centered Cube lattice models the discrete space in which the protein can fold.

It is proved to allow realistic conformations (if small).

The cube has size 2.

Two points are connected (next) iff

\[ |x_i - x_j|^2 + |y_i - y_j|^2 + |z_i - z_j|^2 = 2, \]

Each point has 12 neighbors (but 60° and 180° can be removed).
The **FCC lattice** models the discrete space in which the protein can fold. It is proved to allow realistic conformations (if small). The cube has size 2.

Two points are connected (next) iff

\[|x_i - x_j|^2 + |y_i - y_j|^2 + |z_i - z_j|^2 = 2,\]

Each point has 12 neighbors (but 60° and 180° can be removed).
The Face Centered Cube lattice models the discrete space in which the protein can fold.

It is proved to allow realistic conformations (if small).

The cube has size 2.

Two points are connected (next) iff
\[ |x_i - x_j|^2 + |y_i - y_j|^2 + |z_i - z_j|^2 = 2, \]

Each point has 12 neighbors (but 60° and 180° can be removed).
The protein folding problem

HP on FCC

- Backofen and Will (e.g., Constraints 11(1):5-30, 2006) fold HP-proteins up to length 200 on FCC using CP
- Some precomputations, clever propagation, symmetry breaking, and some geometrical results on the lattice.
- Hoos et al approached it with local search (BMC Bioinformatics 2005 and 2007)
- Dotú et al (IEEE/ACM Trans. Comput. Biology Bioinform. 8(6) 2011) with Constraint based Local Search (extendible to other energy models)

*Drawbacks*: It is only an abstraction. The solutions obtained are far from reality. For instance, helices and sheets are never obtained.

*Problems*:
- Energy function too simple.
- Notion of Contact too strict.
THE PROTEIN FOLDING PROBLEM
HP on FCC

- Backofen and Will (e.g., Constraints 11(1):5-30, 2006) fold HP-proteins up to length 200 on FCC using CP.
- Some precomputations, clever propagation, symmetry breaking, and some geometrical results on the lattice.
- Hoos et al approached it with local search (BMC Bioinformatics 2005 and 2007).

**Drawbacks**: It is only an abstraction. The solutions obtained are far from reality. For instance, helices and sheets are never obtained.

**Problems**:
- Energy function too simple.
- Notion of Contact too strict.
The protein folding problem
A more realistic Energy function on FCC

- A $20 \times 20$ potential matrix $\text{Pot}$ storing the contribution for each pair of aminoacids is used.
- Values are either positive or negative.
- The notion of contact (easy) on lattice models is slightly extended:
  - if distance $(a_i, a_j) < k$ then $\text{Pot}(a_i, a_j)$ else $\frac{\text{Pot}(a_i, a_j)}{\text{distance}^2}$
The protein folding problem
A more realistic Energy function on FCC

- The encoding in CLP(FD) [Dal Palù et al, BMC Bioinformatics 2004] showed that the approach was feasible
- COLA (COnstraint solving on LAttices) [Dal Palù et al, SPE 2007] can predict on FCC proteins of length 100–120
- Arbelaez, Hamadi, Sebag WCB 10 used the (search options of) COLA solver for implementing a Porfolio approach
- Ullah and Steinhöfel used the COLA solver to build a CBLS tool (BMC Bioinformatics 2010(11))
- Similar approach in Shatabda, Newton, Sattar AAAI 2013
- Global constraints for lattice models are studied in Dal Palù et al, J. of Data Mining and Bioinformatics. 4(1), 2010
- I’ll skip on other global constraints for BIO in this talk http://sofdem.github.io/gccat/gccat/Kbioinformatics.html (apologies)
ESCAPING FROM LATTICES

- Small number of angles allowed by lattice models: large errors are unavoidable for long proteins.
- Difficult to reuse known information from deposited proteins (state-of-the-art methods are largely built upon this idea).
- We would like to model the PSP off-lattice, but using finite domain variables.
- We try to use statistics.
The Protein Data Bank contains more than $132K$ protein sequences with their observed 3D structures (X-ray/NMR).

For a sequence of 4 amino acids of our protein, we select their set of occurrences and cluster them into families.
The Protein Data Bank contains more than 132K protein sequences with their observed 3D structures (X-ray/NMR). For a sequence of 4 amino acids of our protein, we select their set of occurrences and cluster them into families.
COMBINING THE BLOCKS

How to assemble fragments?

\[
\begin{array}{cccccc}
F & Y & V & A & H & \ldots \\
F & Y & V & A \\
Y & V & A & H \\
V & A & H & \ldots \\
\end{array}
\]
Two fragments are *compatible* only if the 3 common amino acids have a low RMSD (similar bend angle)
Combining the blocks

Each compatible pair of fragments is stored as $\text{next}(F_i, F_j, M)$ with optimal rotation matrix $M$ (that rotates $F_j$ in the reference of $F_i$).

The protein is grown by attaching compatible fragments, checking spatial constraints, and minimizing its energy [Dal Palù et al, ICLP 2010, IJCAI 2011].
Enriching the model

- Given a $C\alpha$ 4-tuple in 3D, a small degree of freedom for the position of the side chain is allowed.
- Different amino acids have different occupation.
- A pure $C\alpha$-$C\alpha$ model does not keep into account these differences.
- We consider the positions of the centroids of the side chains.
- Roughly, a centroid is the expected center of mass of the side chain.
- We used a model with 4 (real) atoms, plus the centroid. Briefly, 5@$model.
F. Campeotto et al. A Constraint Solver for Flexible Protein Model. JAIR 48, 2013 (also CP 2012 and WCB 12)

The JM constraint is the formalization of the problem of finding a rigid body from a multi body that fulfills a set of spatial constraints.

Domain elements are either fragment (index) and spatial points. Propagation removes fragments that would lead to clashes.
F. Campeotto et al. A declarative concurrent system for protein structure prediction on GPU. JETAI 27(5), 2015
Protein Docking

- Standard methods (ClusPro) rely on a-posteriori filtering of good results (and of an idea of using FFT)
- BiGGER (Barahona and Kripphal) use constraint propagation and symmetry breaking (see Krippahl and Barahona contribution to WCB 15 — and many other publication of the group, including CP 2016)
Computational Protein Design

- Find a primary sequence that will fold in a desired way.
- Simplification: fix some parts (e.g., secondary structures) and replace some of the other amino acids in all possible ways: choose those that minimize the overall energy.
- Viricel, Simoncini, Allouche, de Givry, Barbe, and Schiex contribution to WCB 15 — and previous (many) works of the group.
- Hugo Bazille and Jacques Nicolas (WCB 14, with ASP)
A new challenge

- ASP has been used for “Haplotype inference”
- ASP has been used for “Phylogenetic reconstruction”
- Plenty of clinical data from cancer is becoming available

(Unfortunately)
Aim: building a phylogenesis of cancer in order to understand relationships between genes
Inductive Logic Programming (ILP)

ILP implements a “structured” form of ML, by inferring a first-order theory that “explains” a set of input ground facts.

Given two disjoint sets of positive/negative observations $O^+, O^-$ and a (possibly empty) background theory (program) $P$, ILP finds a set of hypotheses $H$ s.t. $P \land H |= O^+$ and such that for all $\ell \in O^-$ it holds that $P \land H \not|= \ell$ (see De Raedt, 2008).

In presence of data that might be affected from errors, one might accept a set $H$ that might be not complete ($P \land H |= G$, for $G \subseteq O^+$) and be not correct ($P \land H |= \ell$ for $\ell \in W \subseteq O^-$). $G$ should be maximized and $W$ minimized.

In PILP $O^+, O^- P$ can be annotated by a Probability. A semantics based on probabilistic inference rules to deal properly with uncertainty is developed (see Riguzzi et al 2014).
ILP has been used to analyze cancer data

- Muggleton et al participated to a world-wide carcinogenicity prediction competition in 1997 using Progol
- Qui et al (2013) inferred general properties on pancreatic cancer and to allow early detection of this kind of cancer using Progol

ILP and ASP can be interchanged in the core part of the tool, for different purposes.
CONCLUSIONS

- SCALP community has strong potential
- We are stronger together
- Let’s play a role in the future of Computer Science (in general, and in particular in AI and Bioinformatics)
- Submit to WCB 18 and to the Series of AMB (and to Constraints and to TPLP of course!)

A big thank to all of you, to the organizers, and to my coauthors (in particular Federico Campeotto, Alessandro Dal Palù, Federico Fogolari, Andrea Formisano, Enrico Pontelli), and all the WCB colleagues.

Apologies for those I’ve forgotten...
CONCLUSIONS

- SCALP community has strong potential
- We are stronger together
- Let’s play a role in the future of Computer Science (in general, and in particular in AI and Bioinformatics)
- Submit to WCB 18 and to the Series of AMB (and to Constraints and to TPLP of course!)

A big thank to all of you, to the organizers, and to my coauthors (in particular Federico Campeotto, Alessandro Dal Palù, Federico Fogolari, Andrea Formisano, Enrico Pontelli), and all the WCB colleagues.

Apologies for those I’ve forgotten
Questions?