What is Computer Science about?

Theory

Graphics

Data Bases

Programming Languages

Algorithms

Hardware

Bioinformatics

Verification
What is Automated Deduction about?

Generic Problem Solving by a Computer Program.
Introductory Example: Solving $4 \times 4$ Sudoku

```
2 1

3 1

1 2

Start
```
### Introductory Example: Solving $4 \times 4$ Sudoku

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

**Solution**
Formal Model

Represent board by a function $f(x, y)$ mapping cells to their value.

\[
\begin{array}{ccc}
2 & 1 & \\
 & & \\
 & & \\
 & & \\
1 & 2 & \\
\end{array}
\]

Start

\[N = f(1, 1) \approx 2 \land f(1, 2) \approx 1\land\]
\[f(3, 3) \approx 3 \land f(3, 4) \approx 1\land\]
\[f(4, 1) \approx 1 \land f(4, 3) \approx 2\]

$\land$ is conjunction and $\top$ the empty conjunction.
Formal Model

A state is described by a triple \((N; D; r)\) where

- \(N\) contains the equations for the starting Sudoku
- \(D\) a conjunction of further equations computed by the algorithm
- \(r \in \{\top, \bot\}\)

Initial state is \((N; \top; \top)\).
A square $f(x, y)$ where $x, y \in \{1, 2, 3, 4\}$ is called defined by $N \land D$ if there is an equation $f(x, y) \approx z$, $z \in \{1, 2, 3, 4\}$ in $N$ or $D$. For otherwise $f(x, y)$ it is called undefined.
**Rule-Based Algorithm**

**Deduce** \((N; D; \top) \rightarrow (N; D \land f(x, y) \approx 1; \top)\)

provided \(f(x, y)\) is undefined in \(N \land D\), for any \(x, y \in \{1, 2, 3, 4\}\).

**Conflict** \((N; D; \top) \rightarrow (N; D; \bot)\)

provided for \(y \neq z\) (i) \(f(x, y) = f(x, z)\) for \(f(x, y), \ f(x, z)\) defined in \(N \land D\) for some \(x, y, z\) or (ii) \(f(y, x) = f(z, x)\) for \(f(y, x), \ f(z, x)\) defined in \(N \land D\) for some \(x, y, z\) or (iii) \(f(x, y) = f(x', y')\) for \(f(x, y), \ f(x', y')\) defined in \(N \land D\) and \([x, x' \in \{1, 2\}\) or \(x, x' \in \{3, 4\}]\) and \([y, y' \in \{1, 2\}\) or \(y, y' \in \{3, 4\}]\) and \(x \neq x'\) or \(y \neq y'\).
Rule-Based Algorithm

Backtrack \((N; D' \land f(x, y) \approx z \land D''; \bot) \rightarrow (N; D' \land f(x, y) \approx z + 1; \top)\)

provided \(z < 4\) and \(D'' = \top\) or \(D''\) contains only equations of the form \(f(x', y') \approx 4\).

Fail \((N; D; \bot) \rightarrow (N; \top; \bot)\)

provided \(D \neq \top\) and \(D\) contains only equations of the form \(f(x, y) \approx 4\).
Rule-Based Algorithm

Properties: Rules are applied don’t care non-deterministically.

An algorithm (set of rules) is *sound* if whenever it declares having found a solution it actually has computed a solution.

It is *complete* if it finds a solution if one exists.

It is *terminating* if it never runs forever.
Proposition 0.1 (Soundness):
The rules Deduce, Conflict, Backtrack and Fail are sound. Starting from an initial state \((N; \top; \top)\):
(i) for any final state \((N; D; \top)\), the equations in \(N \land D\) are a solution, and,
(ii) for any final state \((N; \top; \bot)\) there is no solution to the initial problem.
Rule-Based Algorithm

Proposition 0.2 (Completeness):
The rules Deduce, Conflict, Backtrack and Fail are complete. For any solution $N \land D$ of the Sudoku there is a sequence of rule applications such that $(N; D; \top)$ is a final state.
Rule-Based Algorithm

Proposition 0.3 (Termination):
The rules Deduce, Conflict, Backtrack and Fail terminate on any input state \((N; \top; \top)\).
Another important property for don’t care non-deterministic rule based definitions of algorithms is *confluence*.

It means that whenever several sequences of rules are applicable to a given states, the respective results can be rejoined by further rule applications to a common problem state.