This document shows the ingredients of a good homework report for the CP course. The \LaTeX source code of this document exemplifies almost everything you need to know about \LaTeX in order to typeset a professional-looking homework report (for the CP course). Use it as a starting point for imitation and delete everything irrelevant. The usage of \LaTeX is optional, but highly recommended, for reasons that will soon become clear to those who have never used it before; any learning time is outside the budget of this course, but will hugely pay off, if not in this course then in the next course(s) you take and when writing your BSc, MSc, or PhD thesis.

1 The Sudoku Problem

1.1 Model

Instance Data and Derived Constants.

- \( n \) designates the number of cells along one side of the generalised Sudoku board. Hence there are \( n^2 \) cells in total. We assume that \( \sqrt{n} \) is an integer.
- \( N \) designates the set \( \{1, 2, 3, \ldots, n\} \).
- \( N' \) designates the set \( \{0, \sqrt{n}, 2 \cdot \sqrt{n}, \ldots, n - \sqrt{n}\} \).
- \( \text{Hints} \) designates the set of hints of the Sudoku board that is to be completed. Each hint takes the form \( \langle r, c, v \rangle \), meaning that the cell at row \( r \) and column \( c \) takes value \( v \), with \( r, c, v \in N \).

Decision Variables.

- \( \text{Sudoku}[r, c] \) designates the value of the cell at row \( r \) and column \( c \) of the Sudoku board, with \( r, c \in N \). Hence \( \text{dom}(\text{Sudoku}[r, c]) = N \).

Problem Constraints.

- \( \text{Row constraints} \) require the values in each row of the Sudoku board to be pairwise different:
  \[
  \text{for all } r \in N : \text{DISTINCT}(\text{Sudoku}[r, \star]) \quad (1)
  \]
- \( \text{Column constraints} \) require the values in each column of the Sudoku board to be pairwise different:
  \[
  \text{for all } c \in N : \text{DISTINCT}(\text{Sudoku}[\star, c]) \quad (2)
  \]
• **Block constraints** require the values in each block of the Sudoku board to be pairwise different:

\[
\text{for all } r, c \text{ in } N' : \text{DISTINCT}(Sudoku[r + 1 \ldots \sqrt{n}, c + 1 \ldots \sqrt{n}]) \quad (3)
\]

• **Hint constraints** require the given hints to be satisfied:

\[
\text{for all } \langle r, c, v \rangle \text{ in } \text{Hints} : Sudoku[r, c] = v \quad (4)
\]

None of the problem constraints is enforced automatically by the choice of decision variables. We advocate enforcing domain consistency on the constraints (1) to (3), because ... The hint constraints (4) are subsumed upon their first run under any consistency level.

**Objective Function.** The Sudoku puzzle is a constraint satisfaction problem, hence there is no objective function to be minimised or maximised.

**Redundant Decision Variables.** Our model has no redundant decision variables, because ...

**Channelling Constraints.** Our model has no channelling constraints, because ...

**Implied Constraints.** Our model has no implied constraints, because ...

**Symmetry-Breaking Constraints.** Our model has no symmetry-breaking constraints, because ...

**Branching Heuristics.** We will experiment with two variable selection heuristics:

- **Size Min**: Branch on an unassigned decision variable \(Sudoku[r, c]\) with the smallest domain, ties being broken by the lexicographically smallest coordinate \(\langle r, c \rangle\).

- **None**: Branch on the unassigned decision variable \(Sudoku[r, c]\) with the lexicographically smallest coordinate \(\langle r, c \rangle\).

In both cases, we use the **Min** value selection heuristic to pick the smallest value, say \(m\), in the domain of the chosen variable \(Sudoku[r, c]\). We branch with \(Sudoku[r, c] = m\) in the left branch and \(Sudoku[r, c] \neq m\) in the right branch. We advocate these branching heuristics, because ...

**Exploration Order.** We advocate exploring the search tree in depth-first order, because ...

### 1.2 Implementation

A *Gecode* implementation of the described model is attached as file `sudoku.cpp`.

**Compilation and Running Instructions.** ... (Explain how to run the implementation for a particular problem instance \(\langle n, \text{Hints} \rangle\) as well as particular consistency, branching, and exploration choices. This will help the teachers grade your program.)

**Sample Test-Run Commands.** ... (Show sample inputs and outputs, and check whether these test runs are reproducible by the program you submit!)
<table>
<thead>
<tr>
<th>$n$</th>
<th>$\text{Size}_\text{Min} + \text{Min}$</th>
<th>$\text{None} + \text{Min}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>Failures</td>
</tr>
<tr>
<td>9</td>
<td>23.45</td>
<td>67</td>
</tr>
<tr>
<td>49</td>
<td>56.78</td>
<td>901</td>
</tr>
<tr>
<td>100</td>
<td>123.45</td>
<td>6789</td>
</tr>
</tbody>
</table>

Table 1: Total runtime (in seconds) and number of failures in terms of $n$, for 100 $\text{Hints}$ sets each.

### 1.3 Experiments

Our experiments were run under Linux OpenSuse 11.3 (64 bit) on an Intel Core i7 950 of 3.07 GHz with an 8 MB L2 cache and a 3 GB RAM.

**Hint:** First do `uname -a` (under Linux or Solaris) and then do `more /proc/cupping` (Linux) or `/usr/platform/X/sbin/prtdiag` (Solaris, where $X$ is the last field in the reply of the previous command) to find this information. Mac OS X users find this information via “About this Mac” in the Apple menu.

Table 1 gives the total runtime (in seconds) and number of failures for 100 $\text{Hints}$ sets (available at . . . ) each for various values of $n$, under the two branching heuristics. The timeout was 600.00 seconds. We observe that $\text{Size}_\text{Min} + \text{Min}$ outperforms $\text{None} + \text{Min}$ on all instances. The reason for this is that . . . .

**Hint:** In order to save a lot of time, it is very important that you write programs that conduct the experiments for you and directly generate result tables (see the LATEX source code of Table 1) or plots, which are automatically imported (rather than copied) into your report.

### Intellectual Property

We certify that this report and all its attachments are solely produced by us, except where explicitly stated otherwise and clearly referenced, and that we can individually explain any part of it at the moment of submitting this report.

### References


Checklist before Submitting

In order to protect yourself against an unnecessary loss of points, and in order to show both self-respect and respect for the human reader of your report, please use the following checklist before submitting:

- Crosscheck your report against the homework instructions.
- Crosscheck your model against the modelling instructions below.
- Crosscheck against the technical writing and LaTeX advice below.
- Spellcheck all documents, including the comments in the source code.
- Proofread, if not grammar-check, your report at least once per teammate.

Modelling Instructions

Any constraint model should be clear and comprehensible, say such that your classmates can understand and implement it without difficulty. Write it in pseudo-code, such as (but not necessarily identical to) the language proposed below and used in the lecture slides, or in MiniZinc \[2\] (which you would have to learn by yourself, but which is interfaced with Gecode).

In any case, the instance data, as well as the decision variables and their domains, must be declared and their semantics must be given **in English**, and every constraint must be annotated with an **English paraphrase**.

You may use standard mathematical notation and logical notation, such as (but not limited to) the following:

- \(M[i,j]\), to designate the element in row \(i\) and column \(j\) of a matrix \(M\); similarly for arrays and matrices of any other number of dimensions. You may use \(*\) or intervals to extract an entire slice of a matrix.

  If some index is an integer decision variable, then you must also model the constraint using the \texttt{ELEMENT} constraint. For instance, the constraint \(c(M[x,j])\), where \(x\) is an integer decision variable and \(j\) an integer constant, is modelled via the conjunction of \texttt{ELEMENT}(\(M[*,j]\), \(x\), \(Mxj\)) and \(c(Mxj)\), where \(Mxj\) is a new decision variable.

- \(\sum(i \in S) f(i)\), to designate the sum over all \(i\) in set \(S\) of the numerical expressions \(f(i)\).

- \(\text{for all } i \in S : c(i)\), to express that for all \(i\) in set \(S\) the constraint \(c(i)\) must hold; we refer to the whole statement as a **quantified constraint**.

- \(\wedge\) or \& (logical and); you may assume an implicit logical and between any two (quantified) constraints.

- \(\Leftrightarrow\) (is logically equivalent to): you may only use this connective for the reification of a constraint \(c(\ldots)\) by a Boolean decision variable \(b\) (denoted by \(c(\ldots) \Leftrightarrow b\)).

Note that you may **not** use full logic: you may neither use \(\lor\) (logical or), \(\Rightarrow\) (logically implies), or \(\Leftrightarrow\) (is logically equivalent to) between two (quantified) constraints, nor use \texttt{exists } i \in S : c(i)\) to express that there must exist at least one \(i\) in set \(S\) such that the (quantified) constraint \(c(i)\) holds, nor apply \(\neg\) (logical negation) to a (quantified) constraint.

If you wrap the implicitly reifying Iverson brackets around a constraint \(c(\ldots)\) in order to formulate a higher-order constraint \(\gamma([c(\ldots)])\), then you must also model the latter using explicit reification of \(c(\ldots)\) by a Boolean decision variable \(b\), via \(c(\ldots) \Leftrightarrow b \land \gamma(b)\).
You may use the following global constraints, as well as any others seen in the course or necessary for a homework:

- **DISTINCT**\(\{x_1, \ldots, x_n\}\), also known as **ALLDIFFERENT**\(\{x_1, \ldots, x_n\}\), requires that any two decision variables \(x_i\) and \(x_j\) with distinct indices take distinct values.

- **ELEMENT**\([a_1, \ldots, a_n], x, y\), where \(a_1, \ldots, a_n, x, y\) are integers or integer decision variables, requires that \(y\) be equal to the element at position \(x\) (counting from 1) of the array \(\langle a_1, \ldots, a_n \rangle\), that is \(a_x = y\).

- **GCC**\(\{x_1, \ldots, x_n\}, [v_1, \ldots, v_m], [\ell_1, \ldots, \ell_m], [u_1, \ldots, u_m]\) requires that the number of decision variables among \(\{x_1, \ldots, x_n\}\) that take the constant value \(v_j\) be between the integer lower bound \(\ell_j\) and integer upper bound \(u_j\) inclusive, for all \(j \in \{1, \ldots, m\}\).

- \([x_1, \ldots, x_n] \leq_{\text{lex}} [y_1, \ldots, y_n]\) requires that the decision-variable array \([x_1, \ldots, x_n]\) be lexicographically smaller than or equal to the decision-variable array \([y_1, \ldots, y_n]\).

- **LINEAR**\([a_1, \ldots, a_n], [x_1, \ldots, x_n], R, d\) requires that the scalar product of the integer array \([a_1, \ldots, a_n]\) with the decision-variable array \([x_1, \ldots, x_n]\) be in relation \(R\) with the integer \(d\), where \(R \in \{<, \leq, =, \neq, \geq, >\}\), that is \((\sum_{i=1}^{n} a_i \cdot x_i) \ R d\).

as well as **all** non-global constraints.
More \LaTeX\ and Technical Writing Advice

Unnumbered itemisation (only to be used when the order of the items does \textit{not} matter):

- Unnumbered displayed formula:
  \[ E = m \cdot c^2 \]

- Numbered displayed formula (which is normally cross-referenced somewhere):
  \[ E = m \cdot c^2 \tag{5} \]

- Formula — the same as formula \textsuperscript{5} — spanning more than one line:
  \[
  E = m \cdot c^2
  \]

Numbered itemisation (only to be used when the order of the items \textit{does} matter):

1. First do this.
2. Then do that.
3. If we are not finished, then go back to Step \textsuperscript{2} else stop.

Tables and elementary mathematics are typeset as exemplified in Table \textsuperscript{2} see \url{ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf} for many more details.

Do \textit{not} use programming-language-specific lower-ASCII notation (such as ! for negation, && for conjunction, || for disjunction, and the equality sign = for assignment) in algorithms, formulas, or models (but rather \texttt{not}, \texttt{and}, \texttt{or}, and \texttt{← or :=}, respectively), as this testifies to a very strong confusion of concepts.

Figures can be imported with \texttt{\includegraphics} or drawn inside the \LaTeX\ source code using the highly declarative notation of the \texttt{tikz} package (see Figure \textsuperscript{1} for sample drawings).

Algorithms can be typeset as pseudo-code as exemplified in Algorithm \textsuperscript{1} study its \LaTeX\ source code.

If you are not sure whether you will stick to your current choice of notation or terminology, then introduce a new (possibly parametric) command. For instance, upon

\begin{verbatim}
\newcommand{\Cardinality}[1]{\left\lvert#1\right\rvert}
\end{verbatim}

the formula $\text{Cardinality}(S)$ typesets the cardinality of set $S$ as $\lvert S \rvert$ with autosized vertical bars and proper spacing, but upon changing the definition of that parametric command to

\begin{verbatim}
\newcommand{\Cardinality}[1]{\# #1}
\end{verbatim}

and recompiling, the formula $\text{Cardinality}(S)$ typesets the cardinality of set $S$ as $#S$.

Similarly, upon

\begin{verbatim}
\newcommand{\Gecode}{\textit{Gecode}}
\end{verbatim}

the text \texttt{\Gecode} typesets into \textit{Gecode}, but upon changing the definition of that command to

\begin{verbatim}
\newcommand{\Gecode}{\textsc{Gecode}}
\end{verbatim}

\footnote{Use footnotes very sparingly, and note that footnote pointers are \textit{never} preceded by a space and always glued immediately \textit{behind} the punctuation, if there is any.}
Figure 1: A binary search tree, a binary min-heap, and a binomial tree of rank 3

<table>
<thead>
<tr>
<th>1: function $f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: if $n &lt; 0$ then</td>
</tr>
<tr>
<td>3: $n ← -2 \cdot n$</td>
</tr>
<tr>
<td>4: else</td>
</tr>
<tr>
<td>5: $n ← 3 \cdot n$</td>
</tr>
<tr>
<td>6: while $n &gt; 0$ do</td>
</tr>
<tr>
<td>7: $n ← n - 1$</td>
</tr>
<tr>
<td>8: return $n$</td>
</tr>
</tbody>
</table>

**Algorithm 1:** Silly algorithm

and recompiling, the text \texttt{Gecode} typesets into GECODE. You can thus obtain an arbitrary number of changes in the document with a constant-time change in its source code, rather than having to perform a linear-time find-and-replace operation within the source code, which is painstaking and error-prone. The imported file macros.tex has a lot of useful predefined commands about mathematics, CP, Gecode, modelling, and algorithms.

Use commands on positioning (such as \texttt{hspace}, \texttt{vspace}, and \texttt{noindent}) and appearance (such as \texttt{small} for reducing the font size, and \texttt{textit} for italics) very sparingly, and ideally only in (parametric) commands, as the very idea of mark-up languages such as \LaTeX is to let the class designer (usually a trained professional typesetter) decide on where things appear and how they look. For instance, \texttt{emph} (for emphasis) compiles (outside italicised environments, such as \texttt{theorem}) into italics under the \texttt{article} class used for this document, but it may compile into \texttt{boldface} under some other class. If you do not (need to) worry about how things look, then you can fully focus on what you are trying to express!

Note that no absolute numbers are used in the \LaTeX source code for any of the cross-references inside this document. For ease of maintenance, \texttt{label} and \texttt{ref} are used for giving a label to something that is automatically numbered (such as an algorithm, equation, figure, footnote, item, line, section, subsection, or table) and referring to a label, respectively. An item in a bibliography is labelled by \texttt{bibitem} and referred to by \texttt{cite} instead. Upon reshuffling the text, adding text, or deleting text, it suffices to recompile twice in order to update all cross-references consistently.

Prefer \texttt{Section~\ref{sec:sudoku}} over \texttt{Section \ref{sec:sudoku}}, using the non-breaking space (typeset as ~) instead of the space, as this gives “Section 1” instead of “Section 1” and thereby avoids that a cross-reference is spread across a line break, as happened in the previous line: this is considered poor typesetting.
<table>
<thead>
<tr>
<th>Topic</th>
<th>\LaTeX code</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greek letter</td>
<td>$\Theta, \Omega, \epsilon$</td>
<td>(\Theta, \Omega, \epsilon)</td>
</tr>
<tr>
<td>multiplication</td>
<td>$m \cdot n$</td>
<td>(m \cdot n)</td>
</tr>
<tr>
<td>division</td>
<td>$\frac{m}{n}, m \div n$</td>
<td>(\frac{m}{n}, m \div n)</td>
</tr>
<tr>
<td>rounding down</td>
<td>$\lfloor n \rfloor$</td>
<td>(\lfloor n \rfloor)</td>
</tr>
<tr>
<td>rounding up</td>
<td>$\lceil n \rceil$</td>
<td>(\lceil n \rceil)</td>
</tr>
<tr>
<td>binary modulus</td>
<td>$m \mod n$</td>
<td>(m \mod n)</td>
</tr>
<tr>
<td>unary modulus</td>
<td>$m = n \mod \ell$</td>
<td>(m = n \mod \ell)</td>
</tr>
<tr>
<td>root</td>
<td>$\sqrt[n]{n}, \sqrt[3]{n}$</td>
<td>(\sqrt[n]{n}, \sqrt[3]{n})</td>
</tr>
<tr>
<td>exponentiation, superscript</td>
<td>$n^i$</td>
<td>(n^i)</td>
</tr>
<tr>
<td>subscript</td>
<td>$n_{i}$</td>
<td>(n_i)</td>
</tr>
<tr>
<td>overline</td>
<td>$\overline{n}$</td>
<td>(\overline{n})</td>
</tr>
<tr>
<td>base 2 logarithm</td>
<td>$\log_2 n$</td>
<td>(\log_2 n)</td>
</tr>
<tr>
<td>base b logarithm</td>
<td>$\log_b n$</td>
<td>(\log_b n)</td>
</tr>
<tr>
<td>binomial</td>
<td>$\binom{n}{k}$</td>
<td>(\binom{n}{k})</td>
</tr>
<tr>
<td>sum</td>
<td>$\sum_{i=1}^n i$</td>
<td>(\sum_{i=1}^n i)</td>
</tr>
<tr>
<td>numeric comparison</td>
<td>$\leq, &lt;, =, \neq, &gt;, \geq$</td>
<td>(\leq, &lt;, =, \neq, &gt;, \geq)</td>
</tr>
<tr>
<td>non-numeric comparison</td>
<td>$\prec, \nprec, \preceq, \succeq$</td>
<td>(\prec, \nprec, \preceq, \succeq)</td>
</tr>
<tr>
<td>extremum</td>
<td>$\min, \max, +, \infty, \bot, \top$</td>
<td>(\min, \max, +, \infty, \bot, \top)</td>
</tr>
<tr>
<td>function</td>
<td>$f: A \to B, \circ, \mapsto$</td>
<td>(f: A \to B, \circ, \mapsto)</td>
</tr>
<tr>
<td>tuple</td>
<td>$\langle a, b, c \rangle$</td>
<td>(\langle a, b, c \rangle)</td>
</tr>
<tr>
<td>set</td>
<td>${a, b, c}, \emptyset, \mathbb{N}$</td>
<td>({a, b, c}, \emptyset, \mathbb{N})</td>
</tr>
<tr>
<td>set membership</td>
<td>$\in, \notin$</td>
<td>(\in, \notin)</td>
</tr>
<tr>
<td>set comprehension</td>
<td>${i \mid 1 \leq i \leq n}$</td>
<td>({i \mid 1 \leq i \leq n})</td>
</tr>
<tr>
<td>set operation</td>
<td>$\cup, \cap, \setminus, \times$</td>
<td>(\cup, \cap, \setminus, \times)</td>
</tr>
<tr>
<td>set comparison</td>
<td>$\subseteq, \subset, \not\subset$</td>
<td>(\subseteq, \subset, \not\subset)</td>
</tr>
<tr>
<td>logic quantifier</td>
<td>$\forall, \exists, \neg, \Rightarrow$</td>
<td>(\forall, \exists, \neg, \Rightarrow)</td>
</tr>
<tr>
<td>logic connective</td>
<td>$\land, \lor, \neg, \rightarrow$</td>
<td>(\land, \lor, \neg, \rightarrow)</td>
</tr>
<tr>
<td>logic</td>
<td>$\models, \equiv, \dashv$</td>
<td>(\models, \equiv, \dashv)</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>$&amp;, #, \approx, \sim, \ell$</td>
<td>(&amp;, #, \approx, \sim, \ell)</td>
</tr>
<tr>
<td>dots</td>
<td>$\ldots, \vdots$</td>
<td>(\ldots, \vdots)</td>
</tr>
<tr>
<td>dots (context-sensitive)</td>
<td>$1, \ldots, n; 1 + \ldots + n$</td>
<td>(1, \ldots, n; 1 + \ldots + n)</td>
</tr>
<tr>
<td>parentheses (autosizing)</td>
<td>$\left(m^{-n-k}\right), (m^{-n-k})$</td>
<td>(\left(m^{-n-k}\right), (m^{-n-k}))</td>
</tr>
<tr>
<td>identifier of &gt; 1 character</td>
<td>$\mathtt{identifier}$</td>
<td>(\mathtt{identifier})</td>
</tr>
<tr>
<td>hyphen, n-dash, m-dash, minus</td>
<td>$-, --, --$</td>
<td>(-, --, --)</td>
</tr>
</tbody>
</table>

Table 2: The typesetting of elementary mathematics. Note very carefully when italics are used by \LaTeX and when not, as well as all the horizontal and vertical spacing performed by \LaTeX.