DiMo – Discrete Modelling Using Propositional Logic

Norbert Hundeshagen Martin Lange Georg Siebert

University of Kassel, Germany

24th Int. Conf. on Theory and Applications of Satisfiability Testing, Barcelona, ES
07/07/2021

Discrete Modelling: An Example

Proposition 1

Formal modelling in propositional logic (or something similar) is an essential skill for computer science graduates.

n-Queens Problem

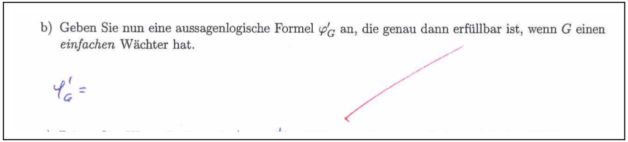
Place n queens on $n \times n$ chess board such that none can capture any other.

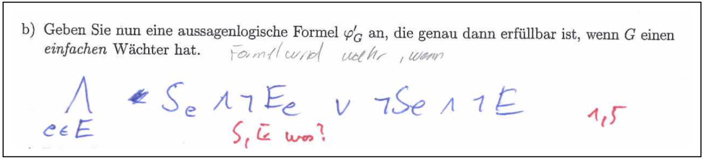
easy to do for $n = 1, 2, 3, 4$, maybe even 5 $\rightsquigarrow$ typical problem that should be solved using computers

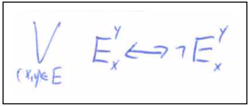
note: problem is parameterised by $n \in \mathbb{N}$; modelled as family $(\varphi_n)_{n \geq 1}$ of satisfiability problems

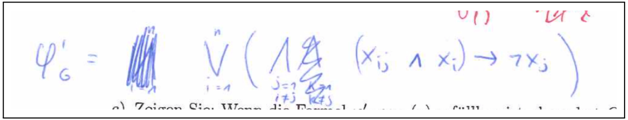
big challenge for students: separating concepts
(e.g. propositional variable vs. parameter)

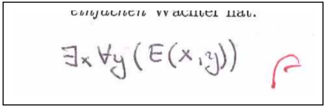
Typical Examples of What Goes Wrong











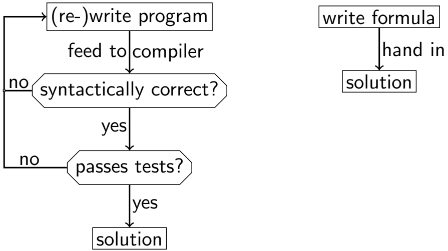
clearly ... satisfiable!

What Can Be Done About it?

programming essentially harder than modelling, but why do students have less problems there?

availability of compilers! they . . .

- support learning loops ⇝ learning by repetition, greater exposure
- make syntactic and semantic learning level explicit



Proposition 2

Discrete Modelling is learnt more easily with compiler support.

Example: the *n*-Queens Problem as a DiMo program

```
SATISFIABLE NQueens(n)
PROPOSITIONS D
PARAMETERS n: {1,..}
FORMULAS
  NQueens(n) = AtLeastOneInEachRow(n) & AtMostOneInEachRow(n) & AtMostOneInEachColumn(n)
             & AtMostOneInEachDiagUp(n) & AtMostOneInEachDiagDown(n)
  AtLeastOneInEachRow(n)     =
     FORALL i: {1,..,n}. FORSOME j: {1,..,n}. D(i,j)
  AtMostOneInEachRow(n)      =
     FORALL i: {1,..,n}. FORALL j: {1,..,n-1}.
        D(i,j) -> FORALL k: {j+1,..,n}. -D(i,k)
  AtMostOneInEachColumn(n)   =
     FORALL i: {1,..,n-1}. FORALL j: {1,..,n}.
        D(i,j) -> FORALL k: {i+1,..,n}. -D(k,j)
  AtMostOneInEachDiagUp(n)   =
     FORALL i: {1,..,n-1}. FORALL j: {1,..,n-1}.
        D(i,j) -> FORALL k: {1,..,MIN {n-i,n-j}}. -D(i+k,j+k)
  AtMostOneInEachDiagDown(n) =
     FORALL i: {1,..,n-1}. FORALL j: {2,..,n}.
        D(i,j) -> FORALL k: {1,..,MIN {n-i,j-1}}. -D(i+k,j-k)
```

- `SATISFIABLE NQueens(n)` ← defines a family of (here: satisfiability) *problems*
- `PROPOSITIONS D` ← declares the non-auxiliary *propositions*
- `PARAMETERS n: {1,..}` ← defines *parameters* and their *ranges*

DiMo in action ...

https://dumbarton.fm.cs.uni-kassel.de

Technology

- supported problems: satisfiability, validity, model enumeration, (upto-)equivalence
- backend written in OCaml, frontend accessible via web
- technology for propositional logic:
  - instantiation of formula schemes to propositional formulas
  - transformation into CNF
  - call to embedded incremental SAT solver (currently: MiniSat)

note: upto-equivalence ...

- is "didactic" equivalence (Π_2^p) rather than logical equivalence (Π_1^p)
- can be used to check students' homework exercises

Further Work

- extension of programs for problem-specific output
- QBF rather than SAT solver for (upto-)equivalence
- other data types as parameters: strings, graphs, . . .
- integration into learning platform
- . . .

The End