Practical arrangements

Tommi Junttila and Tomi Janhunen

Aalto University
School of Science
Department of Computer Science

CS-E3220 Declarative Programming
Spring 2019

First things first...

- Exercises, changes in schedule, announcements etc are in MyCourses

  https://mycourses.aalto.fi/course/view.php?id=20570

- In order to see all these, please

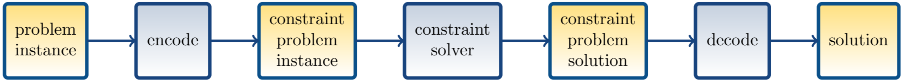
  register to the course in Oodi

  immediately if you plan to take the course

Introduction

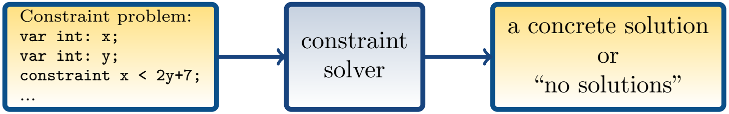
Declarative and constraint programming

- In declarative programming, one declares what a program should accomplish, not how this is done (as in imperative programming)
- Declarative programming is an “umbrella term” covering many paradigms
- We’ll focus on constraint programming, where the problem at hand is described with **variables** and **constraints** so that any assignment to the variables that respects the constraint is a solution to the problem[1]
- The figure below shows a typical flow in constraint programming:
  - The problem instance is encoded to a constraint problem instance,
  - which is then solved by some highly-optimised constraint solver, and
  - a solution to the problem instance is decoded from the solver output



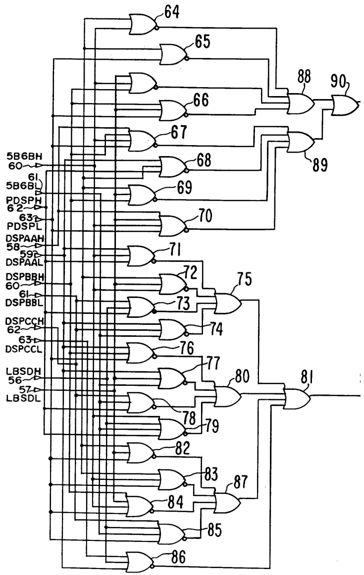
[1] The course could (and perhaps should) be called “constraint programming” but this term historically refers more strongly to one approach (CSPs, round 4) than to some others (SAT, ASP, SMT) that we also cover

- Constraint programming is usually used to solve *intractable*, NP-hard or harder, problems
- Such problems could be solved with custom backtracking search or other algorithms as well ...
- but using highly-optimised **constraint solvers** makes the task easier as one only declares the constraints and the solver then performs the actual search
- Basically, a constraint solver is a tool that
  - takes a constraint problem instance as input,
  - finds whether the constraints have a solution, and
  - outputs such a solution if one exists or “no solutions” if the constraints cannot be satisfied



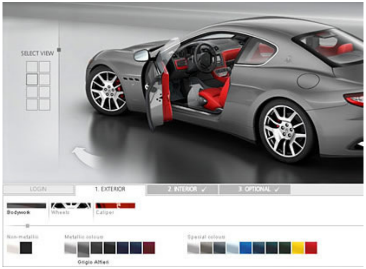
- Constraint problem types to be covered in this course
  - Propositional satisfiability (SAT)
  - Constraint satisfaction problems (CSP)
  - Answer set programming (ASP)
  - Satisfiability modulo theories (SMT)
- Practice: solving problems with these
- Theory: (a glimpse of) how the solvers for these formalisms work

Application example 1: Correctness of integrated circuits



- T. Larrabee: "*Test pattern generation using Boolean satisfiability*", 1992.
- A. Biere et al.: "*Symbolic model checking without BDDs*", 1999.
- ...

Application example 2: Product/Software Configuration



Product configuration:
A combination of components is chosen based on requirements and information about dependencies and incompatibilities.

- T. Soininen and I. Niemelä: Developing a Declarative Rule Language for Applications in Product Configuration, 1999.
- M. Gebser et al.: aspcud: A Linux Package Configuration Tool Based on Answer Set Programming, 2011.

Application example 3: solving mathematical problems

- Boolean Pythagorean Triples problem

  Is there an n such that in every partitioning of $\{1, 2, ..., n\}$ into two parts, either part contains three numbers a, b, and c such that $a^2 + b^2 = c^2$?

- In 2017, Heule and Kullmann showed that such n exists: 7825
- A 200TB machine checkable proof of this was also produced
- Heule, Kullmann, and Marek: Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer, 2016
- Also see Heule and Kullmann: The science of brute force, 2017

Practical arrangements

Tentative lecture schedule

Ten rounds, lectures on the following dates:

1	February 25: Intro and propositional SAT recap	Tommi
2	March 4: CDCL SAT solvers	Tomi
3	March 11: Constraint satisfaction problems (CSP)	Tommi
4	March 18: Binary decision diagrams	Tomi
5	March 25: Advanced normal forms	Tomi
6	April 1: Answer set programming (ASP), part I	Tomi
7	April 15: Answer set programming (ASP), part II	Tomi
8	April 29: Satisfiability modulo theories (SMT), part I	Tommi
9	May 6: Satisfiability modulo theories (SMT), part II	Tommi
10	May 13: Satisfiability modulo theories (SMT), part III	Tommi

Personnel

- Lecturers:
  - Senior university lecturer Tomi Janhunen
  - University lecturer Tommi Junttila
- Email: firstname dot lastname at aalto dot fi
  (No email consultance is possible, attend the exercise sessions)

Prerequisites

- This is a Master’s level course in Computer Science
- The prerequisites are:
  - Basics of propositional logic covered, e.g., by the course CS-E4800 Artificial Intelligence
    A sufficient recap will be provided in Round 1
  - Programming skills in some procedural language such as Python, Java, Scala or C++
    We use Python in some exercises and you should be able to learn the syntax of Python quickly if you don’t know it already
  - Fundamental data structures and algorithms, e.g., the CS-A1140 Data Structures and Algorithms course
  - Basics on discrete mathematics

Passing and grading of the course

- To pass the course, one has to pass
  - obligatory online exercises (see MyCourses)
  - an exam (one on May 31st, another one before the beginning of the Autumn semester)
- The total grade is obtained by the following scheme:

	exam grade					
exercise points	0	1	2	3	4	5
0–199	0	0	0	0	0	0
200–349	0	1	2	2	3	3
350–499	0	2	2	3	3	4
500–649	0	2	3	3	4	4
650–799	0	3	3	4	4	5
800–1000	0	3	4	4	5	5

- For instance, if one gets 600 points from the online exercises and grade 4 from the exam, the total grade will be 4
- As usual, total grade 0 means "not passed" or "failed"

Online exercises

- See MyCourses
- Both "theory" and "practice"
  - Theory: check that the topics, algorithms etc are understood.
    No programming.
  - Practice: solving problems with some actual state-of-the-art tools.
    Some programming required but the emphasis is on modelling the problem with the tool.
- Ten rounds, 100 points available on each $\Rightarrow$ max. 1000 points
  - roughly half of the points from no-programming theory exercises
  - roughly half of the points from programming practice exercises
  - some points for feedback
- Exercise sessions (getting help for the online exercises, see MyCourses for exceptions):
  - Wednesdays at 14-16 in T7 of the CS-building
- The exercise points gathered in Spring 2019 will be valid in all exams before the next course instance starts in Autumn 2019 (but *not* after that)
- **The exercises are personal assignments!**

Questions?