Introduction

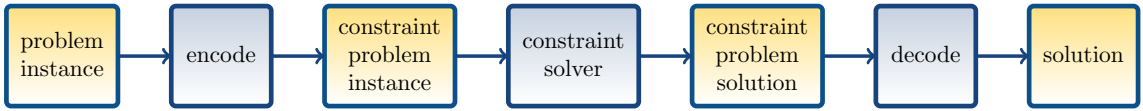
Tommi Junttila and Jussi Rintanen

Aalto University
School of Science
Department of Computer Science

CS-E3220 Declarative Programming
Autumn 2021

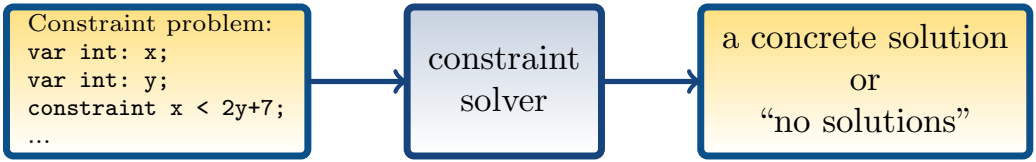
Declarative and constraint programming

- Imperative programming: how something is done
- Declarative programming: declare what a program should accomplish, but not how
- 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 3) than to some others (SAT, SMT) that we also cover

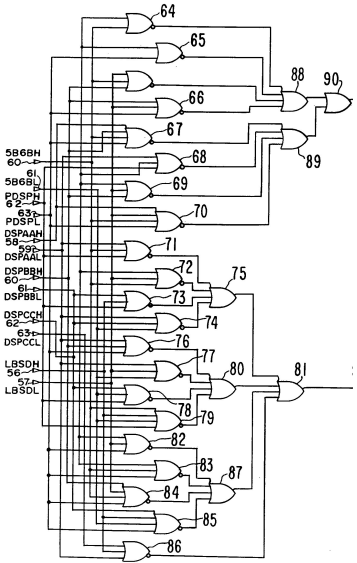
- Constraint programming is usually applied to *intractable*, NP-hard or harder, problems
- Such problems could be solved with custom backtracking search, too...
- but using highly-optimised **constraint solvers** makes the task easier as one only declares the constraints and the solver then performs 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



This course

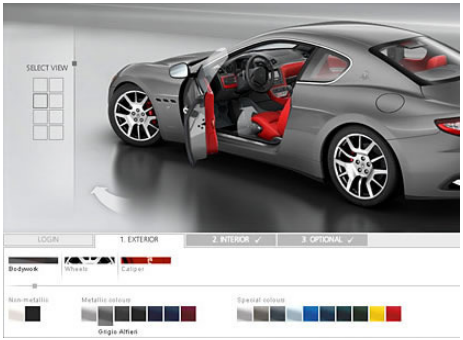
- Constraint problem types covered in this course
  - Propositional satisfiability (SAT)
  - Constraint satisfaction problems (CSP)
  - Satisfiability modulo theories (SMT)
- Practice: solving problems with these
- Theory: (a glimpse of) how the solvers for these formalisms work
- Applications: Where is all this applied

Application 1: Integrated circuits correctness, testing, diagnosis



- T. Larrabee: Test pattern generation using Boolean satisfiability, IEEE CADICS 11(1):4–15, 1992
- A. Biere et al.: Symbolic model checking without BDDs, Proc. TACAS 1999
- E. I. Goldberg, M. R. Prasad and R. K. Brayton, Using SAT for combinational equivalence checking, Proc. IEEE DATE 2001
- J. R. Burch, E. H. Clarke, K. L. McMillan and D. L. Dill, Sequential circuit verification using symbolic model checking, Proc. DAC 1990
- A. Smith, A. Veneris, M. F. Ali, and A. Viglas, Fault diagnosis and logic debugging using Boolean satisfiability, IEEE CADICS 24(10):1606–1621, 2005

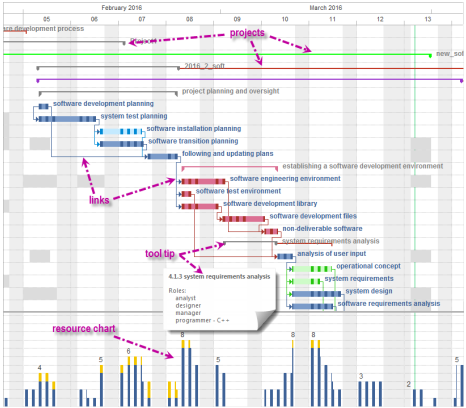
Application 2: Product/Software Configuration



- Configuration: Choose components based on requirements and inter-component dependencies (A requires B; C and D are incompatible)
- Product configuration: cars, all kinds of machinery, ...
- Software package configuration (operating systems)

- T. Soininen and I. Niemelä: Developing a Declarative Rule Language for Applications in Product Configuration, Proc. PADL 1999
- F. Mancinelli et al, Managing the Complexity of Large Free and Open Source Package-Based Software Distributions, Proc. ASE 2006
- P. Trezentos, I. Lynce, A. L. Oliveira, Apt-pbo: solving the software dependency problem using pseudo-boolean optimization, Proc. ASE 2010

Application 3: Planning, Scheduling, Timetabling



- Scheduling of courses/classes for schools, universities
- Project scheduling
- Production scheduling (manufacturing)
- Timetables/schedules for vehicles (trains, buses, airplanes)
- Staff/crew scheduling (airlines, trains, buses)
  E.g. an Aalto CS teaching assistant scheduler

- P. Baptiste, C. Le Pape, and W. Nuijten, Constraint-based scheduling: applying constraint programming to scheduling problems, 2001
- Companies and products: Quintiq, IBM ILOG CP Optimizer

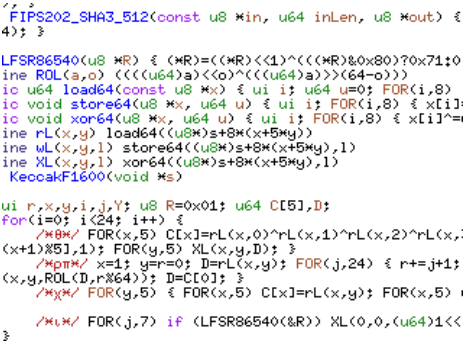
Application 4: solving mathematical problems

- 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, Proc. SAT 2016
- Also see Heule and Kullmann: The science of brute force, CACM 60(8):70–79, 2017
- Brakensiek, Heule, Mackey, and Narváez: The Resolution of Keller's Conjecture, Proc. IJCAI 2020

Application 5: Software Model Checking



- Test if program satisfies a given property
- Safety-critical applications
- Concurrency problems in multi-threaded programs

- R. Jhala, R. Majumdar, Software model checking, ACM Comp. Surv. 41(4):1–54, 2009.
- L. Cordeiro, B. Fischer, and J. Marques-Silva, SMT-Based Bounded Model Checking for Embedded ANSI-C Software, IEEE TASE 38(4):957–974, 2011
- F. Merz, S. Falke, C. Sinz: LLBMC: Bounded Model Checking of C and C++ Programs Using a Compiler IR, Proc. VSTTE 2021

Application 6: Software Synthesis

- Project at Aalto U since 2016 (AISS research group)
- Synthesis of full-stack program code from declarative specifications of software functionalities
- Domain: web apps, information systems, and other UI and DB intensive SW
- Specification for a change in the state of an application
  - User inputs $x_1, x_2, \ldots, x_n$ (any data types)
  - Condition $\Phi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)$ on the inputs and data $y_1, \ldots, y_n$ in DB
  - Program code to change DB according to the inputs
- Constraint satisfaction problem: $x_1, \ldots, x_n$ must satisfy $\Phi(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)$
- Automated synthesis of full stack code (DB, app logic, UI functionality) for whole application