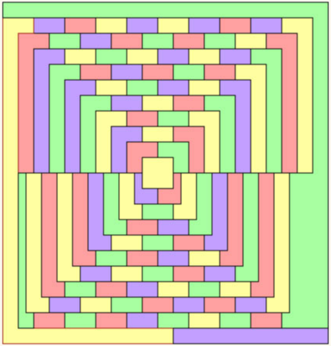
SAT-based algorithms
(in railway infrastructure design)

Bjørnar Luteberget

Visit to SINTEF Digital, 28 Oct 2019



Curriculum Vitae

- Bjørnar Luteberget

- (2005-2010) M.Sc. from NTNU: industrial mathematics.
- (2011-2014) Subsea engineering: Finite element analysis.
- (2014-2019) Ph.D. from UiO: software development for railway engineering.

Numerical conformal mappings (M.Sc. thesis 2010)

- *Numerical approximation of conformal mappings*

Conformal mapping:

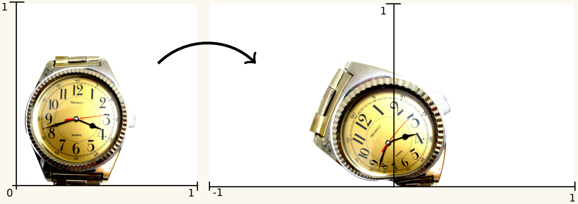


Figure 1.7: The map $f(z) = z^2$ applied to a photo of a clock. See appendix A.1 for program code.

Numerical conformal mappings (M.Sc. thesis 2010)

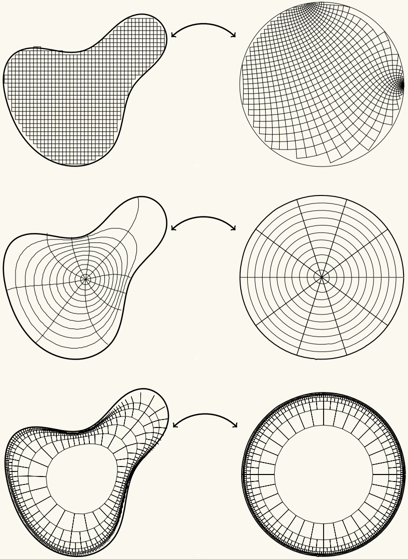


Figure 1.5: Examples of different grid types used to graphically present conformal maps. The first one is a rectangular grid mapped to the disc. The second one is a polar grid mapped from the disc to the desired domain. The third one is a Carleson grid mapped from the disc to the desired domain.

Numerical conformal mappings (M.Sc. thesis 2010)

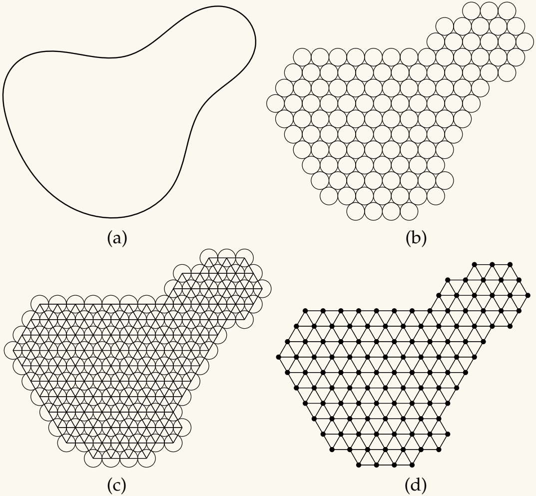


Figure 6.1: (a) The domain Ω. (b) Small circles that cover Ω as well as possible. (c) Graph edges between tangent circles. (d) Approximate representation of Ω as the intersection graph between the circles.

Numerical conformal mappings (M.Sc. thesis 2010)

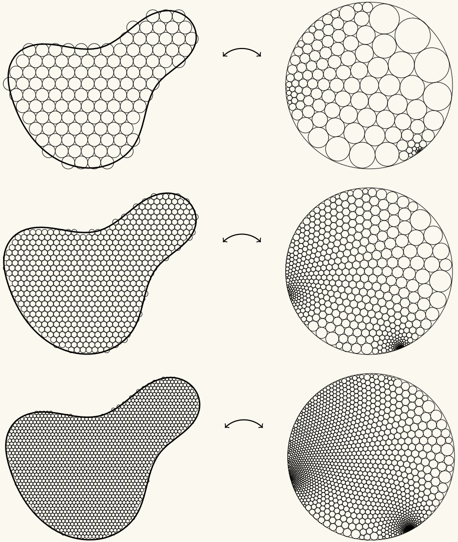
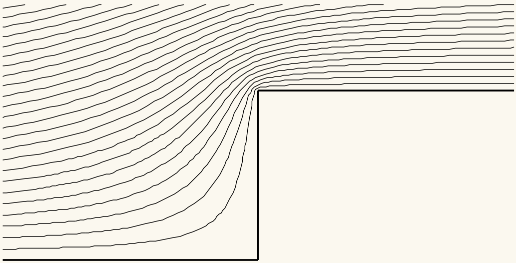
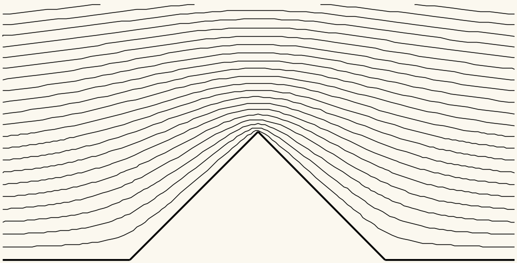


Figure 6.3: Circle packings for the domain Ω with decreasing radius in the hexagonal grid

Numerical conformal mappings (M.Sc. thesis 2010)

Application: planar potential flows (e.g. steady fluid flow).





Numerical conformal mappings (M.Sc. thesis 2010)

Application: flat maps of the brain surface.

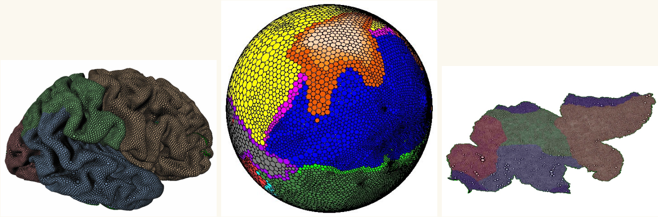
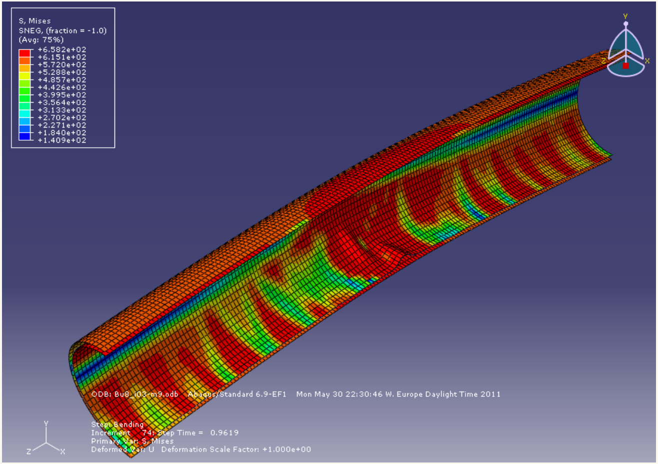


Figure 7.2: Examples from Hurdal's research on flat maps of the brain. Figures from [8].

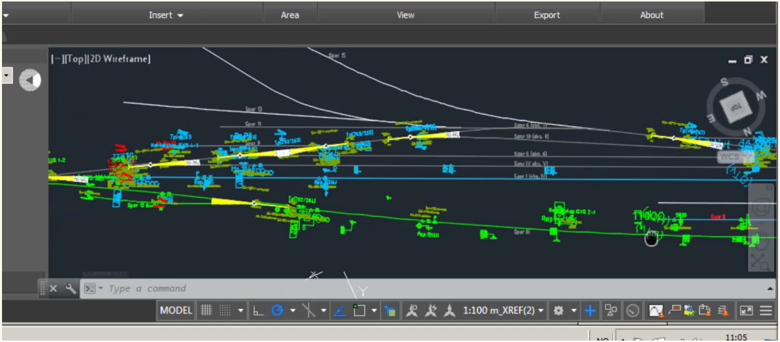
IKM Ocean Design AS: finite element analysis

- Analysis of structural/mechanical, thermal properties.
- Wave statistics, material fatigue.
- Construction, installation planning.



Railcomplete AS: software for railway engineering

- Complex design dependencies between disciplines.
- Software support is not highly developed.
- AutoCAD plugin (C#/.NET): railway object model and analysis tools.



Ph.D. research (2015-2019)

Ph.D. studies at Informatics, UiO funded by NFR and Railcomplete AS (industry Ph.D.).

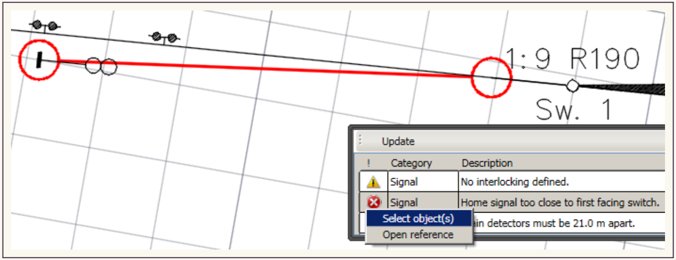
Automated reasoning for railway construction planning:

- Static analysis using Datalog
  (published in iFM '16, FM '16, FMSD)
- Controlled natural language
  (published in SEFM '17, journ. u/review)
- Dynamic analysis (SAT-based verification/synthesis)
  (published in FMCAD '18, FM '19, journ. u/review)
- Drawing railway schematics (SAT-based optimization)
  (published in iFM '19).

Ph.D. research (2015-2019)

- **Best paper award** at iFM '16 for static railway infrastructure verification.
- Rule base in Datalog syntax with structured comments:

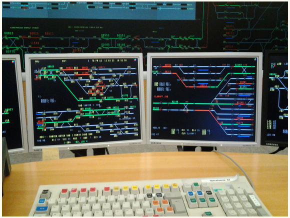
```
%| rule: Home signal too close to first facing switch.
%| type: technical
%| severity: error
homeSignalBeforeFacingSwitchError(S,SW) :-
    firstFacingSwitch(B,SW,DIR),
    homeSignalBetween(S,B,SW),
    distance(S,SW,DIR,L), L < 200.
```

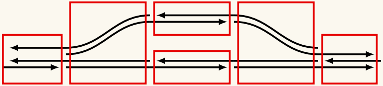


Ph.D. research (2015-2019)

- **Best paper award** at FMCAD '18 for local railway capacity verification.
- Split the planning work into two separate points of view:

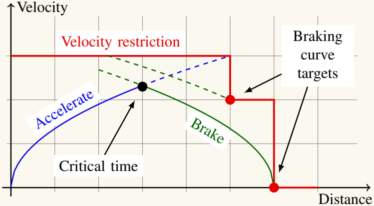
Dispatcher (discrete planning)





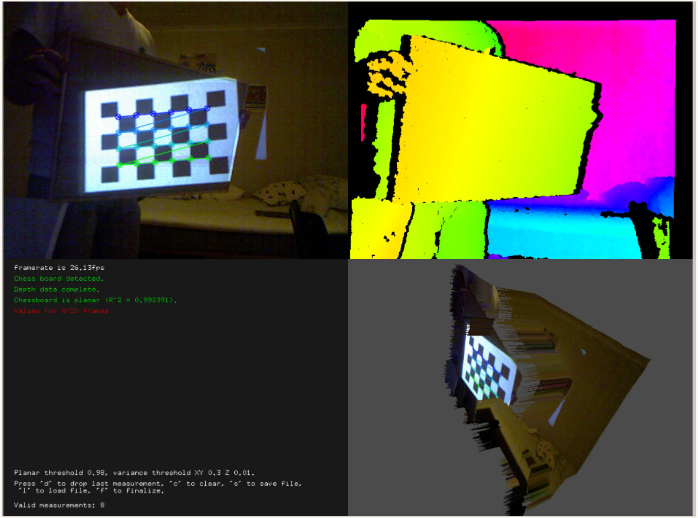
Elementary routes and their conflicts

Train driver (simulation)



Hobby projects: Dynamic projection mapping

- C++ programming, computer vision library, least squares, ...
- Tom Nærland and Bjørnar Luteberget



Hobby projects: Dynamic projection mapping

Hobby projects: III – audio-visual performance

- Supercollider audio programming, C++ / GLSL graphics, guitar and electronics.
- Øyvind Mellbye, Tom Nærland, Markus Dvergastein, and Bjørnar Luteberget.

Hobby projects: Ill – audio-visual performance

Production Network for Electronic Art, Norway

HOME ABOUT PNEK NETWORK PUBLICATIONS UPCOMING EVENTS PROJECTS GALLERY CONTACT ARCHIVE

DUSK TILL DAWN ART PRIZE

The winner of From Dusk Till Dawn Art Prize: "Ill"

About Ill:
Ill is an audiovisual and anti-fascist noise project started in 2013 in Oslo. While desperately seeking to balance real-time graphics with analog and digital audio, Ill simultaneously aims to challenge the perception of their own bodies and their submission to structures of authority (material, social, spiritual).

HVA SKJER I NETTVERKET:

Atelier Nord

FAEN – FEMALE ARTISTIC EXPERIMENTS NORWAY

Open Call – Apply for a Chance to Exhibit in 2020

BEK

Retrodemo/demoworkshop

Open call for prosjekter

Dans for Voksne

Hong Chulki, Skiftende Skydekke og AnnarkieLene Grenager, spilt av ensemblene Parkour, Krock og...

Notam

Verksteder for viderekomne brukere

Verksted for avanserte brukere: MuBu med Diemo Schwarz

Research interests

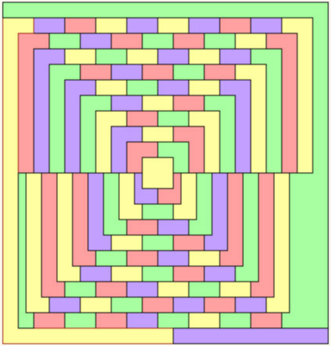
Bjørnar's main personal research interests:

- Mathematical/scientific programming in a broad sense.
- Automating and optimizing in design and engineering using mathematical modelling and algorithms.
- The interface between general solvers and specific problem domains.

SAT-based algorithms (in railway infrastructure design)

Bjørnar Luteberget

Visit to SINTEF Digital, 28 Oct 2019



Outline

- **Part I: SAT and SMT**
  a. SAT solvers
  b. SAT-based algorithms and SMT
- **Part II: SAT-based algorithms in railway engineering**
  a. Local capacity verification using planning and simulation.
  b. Schematic drawings using difference logic and optimization.

Propositional logic

Propositions are statements that are either true or false:

- Example: *it is raining*
- Example: *it is sunny*
- Mathematical model: the statements $x_1, x_2, \ldots$ have no further semantics than being either true or false.

Operations (logical connectives):

- AND ($\bigwedge$): $x_1 \wedge x_2$.
- OR ($\bigvee$): $x_1 \vee x_2$.
- NOT ($\neg$): $\neg x_1$.

- IMPLIES ($\longrightarrow$), $a \to b \equiv b \vee \neg a$.
- EQUIV ($\longleftrightarrow$) $a \leftrightarrow b \equiv (b \vee \neg a) \wedge (a \vee \neg b)$.
- ...

The Boolean satisfiability problem

Boolean algebra calculations:

- $a = T$, $b = F$
- $a \wedge b = F$
- $a \vee b = T$

The Boolean satisfiability problem (SAT):

- Given a propositional logic formula $\phi(x_1, x_2, \ldots)$, does there exist an assignment to the variables such that $\phi(x_1, x_2, \ldots) = T$?
- NP-complete. ("The" NP-complete problem).

The Boolean satisfiability problem

- Conjunctive normal form:
  - express SAT problem as a conjunction of clauses.
  - Each clause is a disjunction of literals.
  - A literal is a variable or a negated variable.
- Example:

$$
\begin{aligned}
(x_1 \vee \neg x_2) &\quad \wedge \\
(x_2 \vee x_3) &\quad \wedge \\
(x_4 \vee x_5 \vee \neg x_8) &\quad \wedge \\
(x_5 \vee \neg x_6 \vee x_7) &\quad \wedge \\
(x_2 \vee \neg x_6 \vee \neg x_7) &\quad \wedge \\
x_8 &
\end{aligned}
$$

DPLL

The DPLL algorithm
(Davis, Putnam, Logemann, Loveland, 1962)

- Main idea: backtracking + unit propagation.
- Still basis for most efficient and complete solvers today.
- Let's solve the following SAT problem:

$$
\begin{aligned}
(x_1 \vee x_2) &\quad \wedge \\
(x_1 \vee x_3 \vee x_8) &\quad \wedge \\
(\neg x_2 \vee \neg x_3 \vee x_4) &\quad \wedge \\
(\neg x_4 \vee x_5 \vee x_7) &\quad \wedge \\
(\neg x_4 \vee x_6 \vee x_8) &\quad \wedge \\
(\neg x_5 \vee \neg x_6) &\quad \wedge \\
(x_7 \vee \neg x_8) &\quad \wedge \\
(x_7 \vee \neg x_9 \vee x_{10}) &\quad \wedge
\end{aligned}
$$

DPLL run (example by Jon Smock)

Decisions:

Formula:

x_1, x_2

x_1, x_3, x_8

$\neg x_2, \neg x_3, x_4$

$\neg x_4, x_5, x_7$

$\neg x_4, x_6, x_8$

$\neg x_5, \neg x_6$

$x_7, \neg x_8$

$x_7, \neg x_9, x_{10}$

DPLL run

Decisions:

$x_7 = F$.

Formula:

$x_1,\ x_2$

$x_1,\ x_3,\ x_8$

$\neg x_2,\ \neg x_3,\ x_4$

$\neg x_4,\ x_5,\ \cancel{x_7}$

$\neg x_4,\ x_6,\ x_8$

$\neg x_5,\ \neg x_6$

$\cancel{x_7},\ \neg x_8$

$\cancel{x_7},\ \neg x_9,\ x_{10}$

DPLL run

Decisions:

$x_7 = F.$

$x_8 = F.$

Formula:

x_1, x_2

$x_1, x_3, \cancel{x_8}$

$\neg x_2, \neg x_3, x_4$

$\neg x_4, x_5, \cancel{x_7}$

$\neg x_4, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{x_7, \neg x_8}$

$\cancel{x_7}, \neg x_9, x_{10}$

DPLL run

Decisions:

$x_7 = F$.

  $x_8 = F$.

$x_9 = T$.

Formula:

x_1, x_2

$x_1, x_3, \cancel{x_8}$

$\neg x_2, \neg x_3, x_4$

$\neg x_4, x_5, \cancel{x_7}$

$\neg x_4, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{\cancel{x_7}, \neg x_8}$

$\cancel{x_7}, \cancel{\neg x_9}, x_{10}$

DPLL run

Decisions:

$x_7 = F.$

$x_8 = F.$

$x_9 = T.$

$x_{10} = T.$

Formula:

x_1, x_2

$x_1, x_3, \cancel{x_8}$

$\neg x_2, \neg x_3, x_4$

$\neg x_4, x_5, \cancel{x_7}$

$\neg x_4, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{x_7, \neg x_8}$

$\cancel{x_7, \neg x_9, x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

  $x_8 = F.$

$x_9 = T.$

  $x_{10} = T.$

$x_1 = F.$

Formula:

$\cancel{x_1}, x_2$

$\cancel{x_1}, x_3, \cancel{x_8}$

$\neg x_2, \neg x_3, x_4$

$\neg x_4, x_5, \cancel{x_7}$

$\neg x_4, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{\cancel{x_7}, \neg x_8}$

$\cancel{\cancel{x_7}, \cancel{\neg x_9}, x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$\quad x_8 = F.$

$x_9 = T.$

$\quad x_{10} = T.$

$x_1 = F.$

$\quad x_2 = T.$

$\quad x_3 = T.$

Formula:

$\cancel{\cancel{x_1},\ x_2}$

$\cancel{\cancel{x_1},\ x_3,\ \cancel{x_8}}$

$\cancel{\neg x_2},\ \cancel{\neg x_3},\ x_4$

$\neg x_4,\ x_5,\ \cancel{x_7}$

$\neg x_4,\ x_6,\ \cancel{x_8}$

$\neg x_5,\ \neg x_6$

$\cancel{\cancel{x_7},\ \neg x_8}$

$\cancel{\cancel{x_7},\ \cancel{\neg x_9},\ x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$\quad x_8 = F.$

$x_9 = T.$

$\quad x_{10} = T.$

$x_1 = F.$

$\quad x_2 = T.$

$\quad x_3 = T.$

$\quad x_4 = T.$

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, x_8}$

$\cancel{\neg x_2, \neg x_3, x_4}$

$\cancel{\neg x_4}, x_5, \cancel{x_7}$

$\cancel{\neg x_4}, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{x_7, \neg x_8}$

$\cancel{x_7, \neg x_9, x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

  $x_8 = F.$

$x_9 = T.$

  $x_{10} = T.$

$x_1 = F.$

  $x_2 = T.$

  $x_3 = T.$

  $x_4 = T.$

  $x_5 = T.$

  $x_6 = T.$

Formula:

$\cancel{x_1,\ x_2}$

$\cancel{x_1,\ x_3,\ x_8}$

$\cancel{\neg x_2,\ \neg x_3,\ x_4}$

$\cancel{\neg x_4,\ x_5,\ x_7}$

$\cancel{\neg x_4,\ x_6,\ x_8}$

$\cancel{\neg x_5},\ \cancel{\neg x_6} \leftarrow$ Conflict!

$\cancel{x_7,\ \neg x_8}$

$\cancel{x_7,\ \neg x_9,\ x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$\quad x_8 = F.$

$x_9 = T.$

$\quad x_{10} = T.$

$x_1 = T.$

Formula:

$\cancel{x_1,\ x_2}$

$\cancel{x_1,\ x_3,\ x_8}$

$\neg x_2,\ \neg x_3,\ x_4$

$\neg x_4,\ x_5,\ \cancel{x_7}$

$\neg x_4,\ x_6,\ \cancel{x_8}$

$\neg x_5,\ \neg x_6$

$\cancel{x_7,\ \neg x_8}$

$\cancel{x_7,\ \neg x_9,\ x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$\quad x_8 = F.$

$x_9 = T.$

$\quad x_{10} = T.$

$x_1 = T.$

$x_2 = T.$

Formula:

$\cancel{x_1, \ x_2}$

$\cancel{x_1, \ x_3, \ x_8}$

$\cancel{\neg x_2}, \ \neg x_3, \ x_4$

$\neg x_4, \ x_5, \ \cancel{x_7}$

$\neg x_4, \ x_6, \ \cancel{x_8}$

$\neg x_5, \ \neg x_6$

$\cancel{x_7, \ \neg x_8}$

$\cancel{x_7, \ \neg x_9, \ x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$x_8 = F.$

$x_9 = T.$

$x_{10} = T.$

$x_1 = T.$

$x_2 = T.$

$x_3 = F.$

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, \cancel{x_8}}$

$\cancel{\cancel{\neg x_2}, \neg x_3, x_4}$

$\neg x_4, x_5, \cancel{x_7}$

$\neg x_4, x_6, \cancel{x_8}$

$\neg x_5, \neg x_6$

$\cancel{\cancel{x_7}, \neg x_8}$

$\cancel{\cancel{x_7}, \cancel{\neg x_9}, x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$x_8 = F.$

$x_9 = T.$

$x_{10} = T.$

$x_1 = T.$

$x_2 = T.$

$x_3 = F.$

$x_4 = F.$

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, x_8}$

$\cancel{\neg x_2, \neg x_3, x_4}$

$\cancel{\neg x_4, x_5, x_7}$

$\cancel{\neg x_4, x_6, x_8}$

$\neg x_5, \neg x_6$

$\cancel{x_7, \neg x_8}$

$\cancel{x_7, \neg x_9, x_{10}}$

DPLL run

Decisions:

$x_7 = F.$

$x_8 = F.$

$x_9 = T.$

$x_{10} = T.$

$x_1 = T.$

$x_2 = T.$

$x_3 = F.$

$x_4 = F.$

$x_5 = T.$

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, x_8}$

$\cancel{\neg x_2, \neg x_3, x_4}$

$\cancel{\neg x_4, x_5, x_7}$

$\cancel{\neg x_4, x_6, x_8}$

$\cancel{\neg x_5}, \neg x_6$

$\cancel{x_7, \neg x_8}$

$\cancel{x_7, \neg x_9, x_{10}}$

DPLL run

Decisions:

$x_7 = F$.

$\quad x_8 = F$.

$x_9 = T$.

$\quad x_{10} = T$.

$x_1 = T$.

$x_2 = T$.

$x_3 = F$.

$x_4 = F$.

$x_5 = T$.

$\quad x_6 = F$.

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, \cancel{x_8}}$

$\cancel{\cancel{\neg x_2}, \neg x_3, x_4}$

$\cancel{\neg x_4, x_5, \cancel{x_7}}$

$\cancel{\neg x_4, x_6, \cancel{x_8}}$

$\cancel{\cancel{\neg x_5}, \neg x_6}$

$\cancel{\cancel{x_7}, \neg x_8}$

$\cancel{\cancel{x_7}, \cancel{\neg x_9}, x_{10}}$

Solved!

Advances since DPLL

Many advances have been made in SAT solving since 1962:

- Conflict-driven clause learning in GRASP (Silva, 1996)
- Two-watched literals in zChaff (Chaff, 2001)
- VSIDS (variable state independent decaying sum) in zChaff
- Random restarts
- Locality based search (Chaff, Berkmin, MiniSAT)

Conflict-driven clause learning (CDCL) run

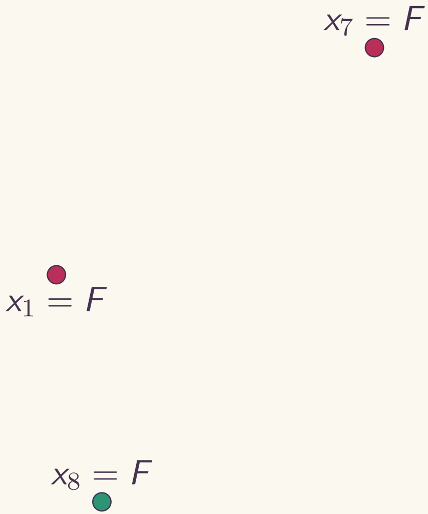
$$x_7 = F$$

Conflict-driven clause learning (CDCL) run

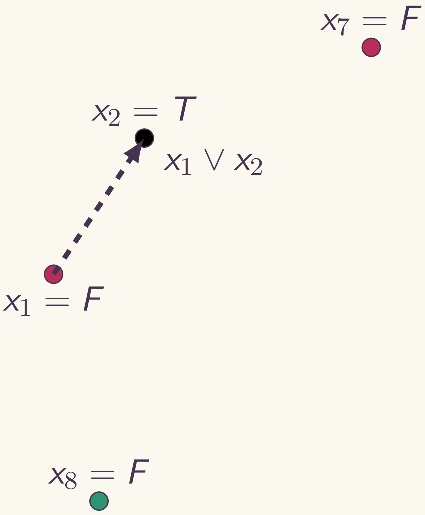
$x_7 = F$

$x_8 = F$

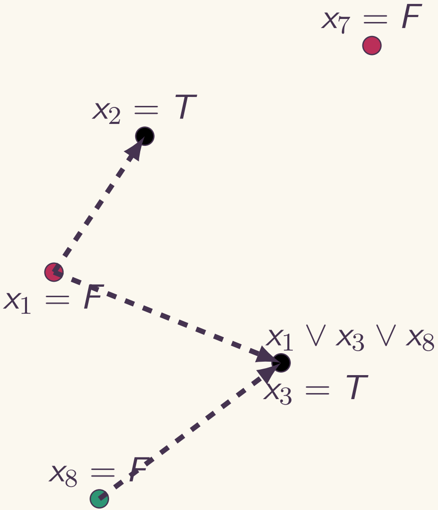
Conflict-driven clause learning (CDCL) run



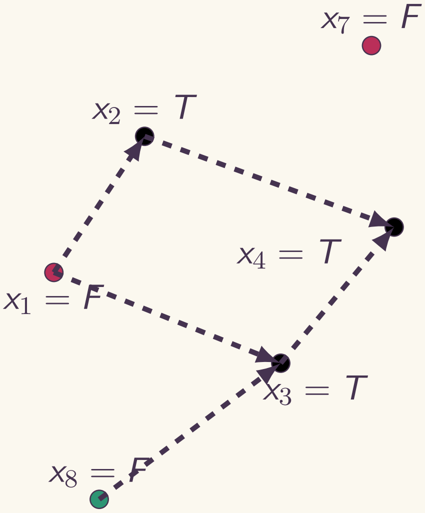
Conflict-driven clause learning (CDCL) run



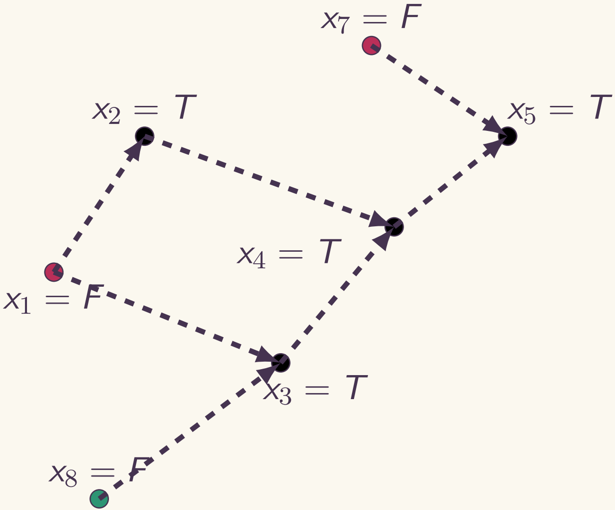
Conflict-driven clause learning (CDCL) run



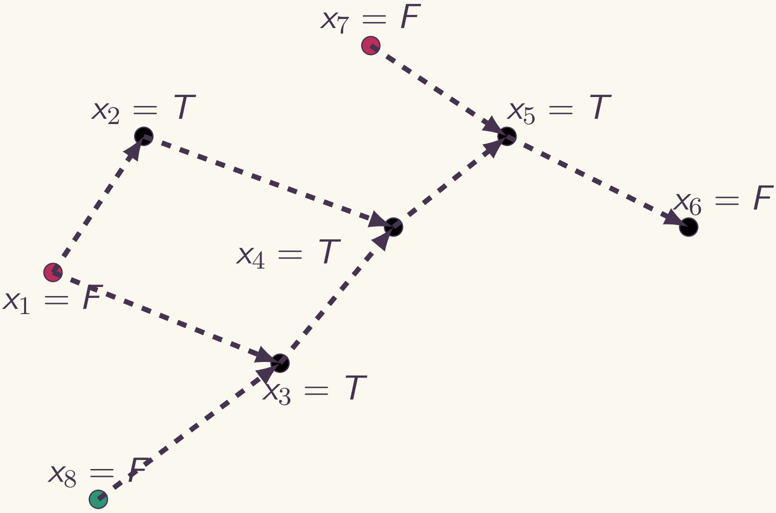
Conflict-driven clause learning (CDCL) run



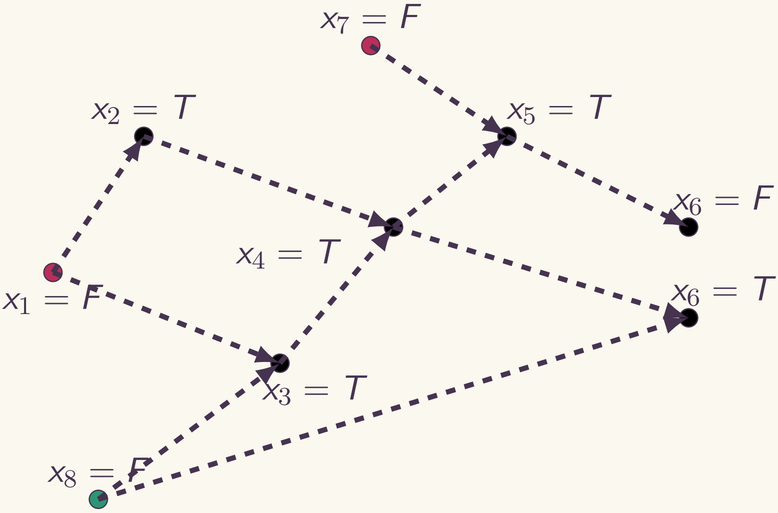
Conflict-driven clause learning (CDCL) run



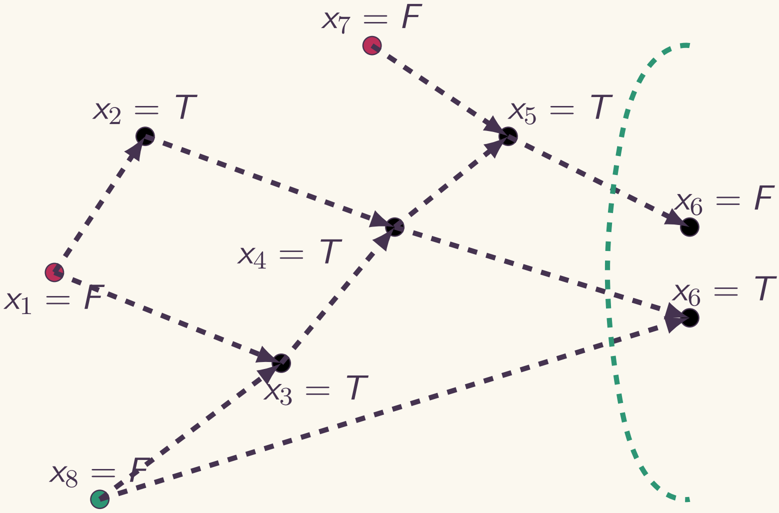
Conflict-driven clause learning (CDCL) run



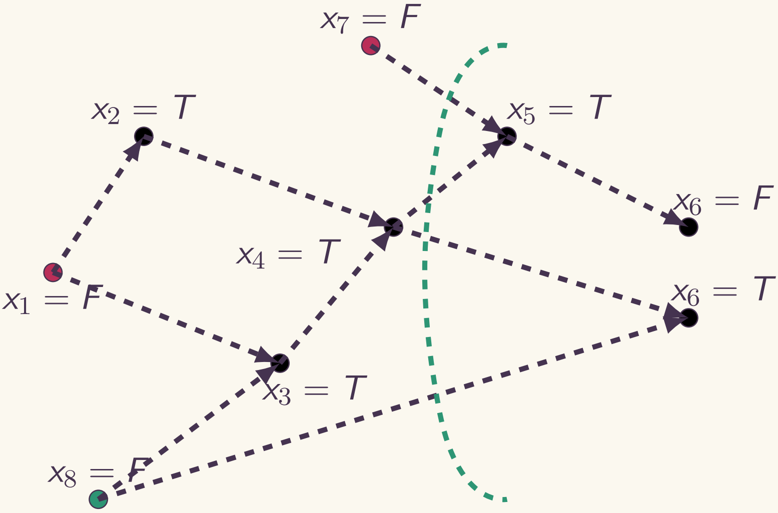
Conflict-driven clause learning (CDCL) run



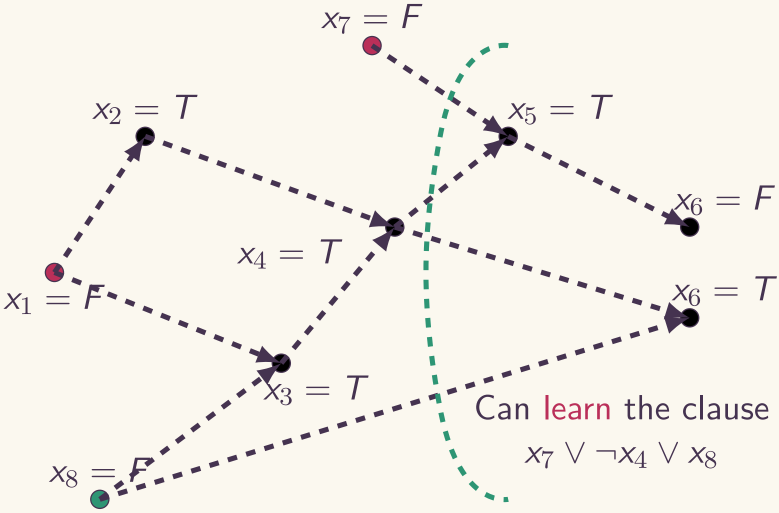
Conflict-driven clause learning (CDCL) run



Conflict-driven clause learning (CDCL) run



Conflict-driven clause learning (CDCL) run



Can learn the clause

$x_7 \vee \neg x_4 \vee x_8$

DPLL run

Decisions:

$x_7 = F$. $\leftarrow$ **Backtrack**

$\quad x_8 = F$.

$x_9 = T$.

$\quad x_{10} = T$.

$x_1 = F$.

$\quad x_2 = T$.

$\quad x_3 = T$.

$\quad x_4 = T$.

$\quad x_5 = T$.

$\quad x_6 = T$.

Formula:

$\cancel{x_1, x_2}$

$\cancel{x_1, x_3, x_8}$

$\cancel{\neg x_2, \neg x_3, x_4}$

$\cancel{\neg x_4, x_5, x_7}$

$\cancel{\neg x_4, x_6, x_8}$

$\cancel{\neg x_5}, \cancel{\neg x_6}$ $\leftarrow$ Conflict!

$\cancel{x_7, \neg x_8}$

$\cancel{x_7, \neg x_9, x_{10}}$

$x_7, \neg x_4, x_8$ $\leftarrow$ **Learned**

Conflict-driven clause learning

After having found a conflict clause, we can:

- Add the clause to our problem, hoping to gain unit propagation from it in other situations.
- Backtrack to the highest decision level of the variables in the clause: non-chronological backtracking.

SAT solver performance progression

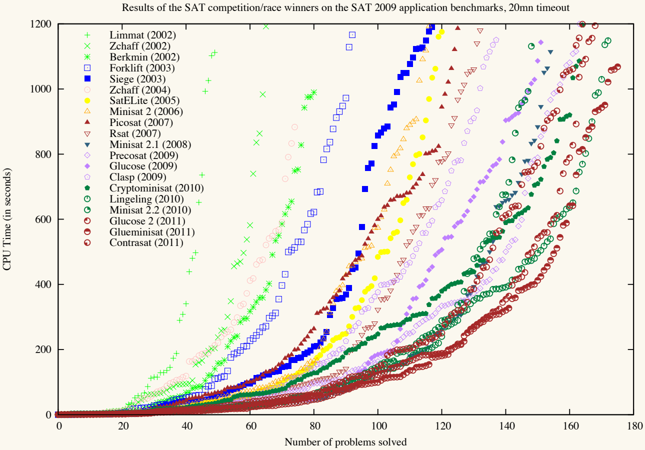


Figure 1: Evolution of the best solvers from 2002 to 2010 on the application benchmarks from the SAT 2009 competition using the cumulative number of problems solved (x axis) within a specific amount of time (y axis). The farther to the right the data points are, the better the solver.

Figure from M. Järvisalo, D. Le Berre, O. Roussel, L. Simon: *The international SAT solver competitions*, AI Magazine, 2012.

Applications of SAT technology

- Formal methods:
  - Hardware model checking, software model checking, model-based testing.
- Artificial intelligence:
  - Planning, knowledge representation, games.
- Bioinformatics
  - Haplotype inference, pedigree checking, genetic regulatory networks.
- Design automation
  - Equivalence checking, delay computation, fault diagnosis, noise analysis.
- Security
  - Cryptanalysis, inverting hash functions.

(from D. Le Berre: *Introduction to SAT*, SAT-SMT summer school 2014 slides)

Applications of SAT technology

- Computationally hard problems
  - Graph coloring, traveling salesperson.
- Mathematical problems
  - van der Waerden numbers, open problems in quasigroups.

- Core engine for other solvers: 0-1 ILP / pseudo-boolean, QBF, #SAT, SMT, MaxSAT.
- Integrated with theorem provers: HOL, Isabelle.
- Integrated into software: SuSe Linux package dependency manager, Eclipse provisioning system.

(from D. Le Berre: *Introduction to SAT*, SAT-SMT summer school 2014 slides)

Incremental SAT

Solver interface from MiniSAT: (Eén, Sörensson, 2003)

```
public interface SATSolver {
  public Literal NewVariable();
  public void AddClause(List<Literal> clause);
  public Model SolveUnderAssumptions(
                  List<Literal> assumptions);
}
```

- Allows solving many related SAT problems, reusing decisions and learnt clauses!
- Basis for a wide variety of solvers.

Properties of transition systems

- System state: a vector of Booleans s.
- System transitions: a Boolean formula $T(s_j, s_{j+1})$.
- The system starts in an initial state $I(s_0)$.
- Verify that a property $p(s_j)$ holds in all states.
- ... and we're willing to limit the number of transitions to k.

Bounded model checking (Biere et al., 1999):

$$\mathrm{BMC}(S, I, T, p, k) = I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(s_k)$$

Properties of transition systems

BMC gives a heavy formula. Incremental SAT helps:

$$BMC(k=1) = I(s_0) \land T(s_0, s_1) \land \neg p(s_1)$$
$$BMC(k=2) = I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land \neg p(s_2)$$
$$BMC(k=3) = I(s_0) \land T(s_0, s_1) \land T(s_1, s_2) \land T(s_2, s_3) \land \neg p(s_3)$$
$$\dots$$

... and the same idea applies to planning!
Planning as satisfiability (Kautz, Selman, 1992)

Satisfiability modulo theories

What if the propositions themselves had meaning in some other theory with a corresponding decision procedure?

Satisfiability modulo theories

What if the propositions themselves had meaning in some other theory with a corresponding decision procedure?

$$\phi = (a < b) \wedge (b < c) \wedge (c < a \vee a < c)$$

Satisfiability modulo theories

What if the propositions themselves had meaning in some other theory with a corresponding decision procedure?

$$\phi = (a < b) \wedge (b < c) \wedge (c < a \vee a < c)$$

Boolean abstraction:

$$\phi = x_1 \wedge x_2 \wedge (x_3 \vee x_4)$$

Satisfiability modulo theories

What if the propositions themselves had meaning in some other theory with a corresponding decision procedure?

$$\phi = (a < b) \wedge (b < c) \wedge (c < a \vee a < c)$$

Boolean abstraction:

$$\phi = x_1 \wedge x_2 \wedge (x_3 \vee x_4)$$

SAT solver finds $x_1 = T$, $x_2 = T$, $x_3 = T$.
Theory solver (difference logic) learns $\neg x_1 \vee \neg x_2 \vee \neg x_3$.

Satisfiability modulo theories

Approaches to SMT:

1. Eager SMT: represent or approximate domain by Booleans ("bit-blasting").
   - Encoding techinques: one-hot, unary, binary (logarithmic). (see Björk, 2009)
2. Fully lazy SMT: wait for an assignment from the SAT solver, use it as an assumption in the theory solver.
3. Lazy SMT: wait for a partial assignment, and search for constraints that can be deduced from the partial assignment.

Satisfiability modulo theories

$$\phi = (g(a) = c) \land (f(g(a)) \neq f(c) \lor g(a) = d) \land (c \neq d)$$

Boolean abstraction:

$$\phi = x_1 \land (\neg x_2 \lor x_3) \land \neg x_4$$

SAT solver suggests: $x_1 = T$, $x_2 = F$, $x_4 = F$.

Uninterpreted functions solver finds conflict:
$f(g(a)) = f(c) \neq f(c)$, add new clause:

$$\neg x_1 \lor x_2$$

Satisfiability modulo theories

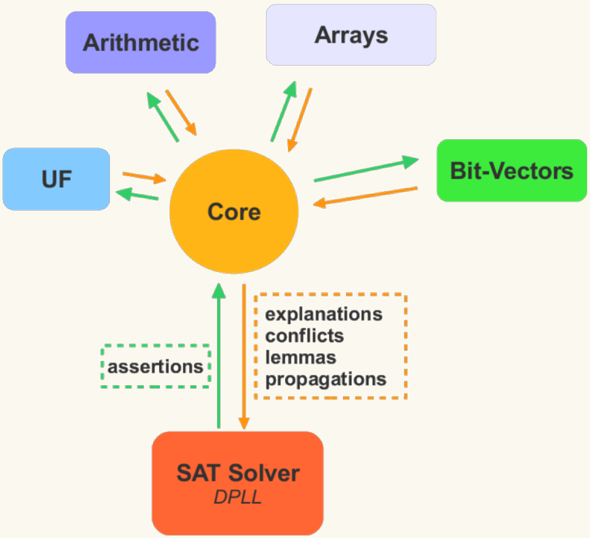


Figure from Clark Barrett, Summer School on Formal Techniques slides, 2016

Satisfiability modulo theories

Desirable properties in a theory solver:

- Speed/efficiency.
- Incrementality.
- Backtracking.
- Concise expression of conflicts.

The Z3 theorem prover

A highly popular and successful solver: Z3 from Microsoft Research.

Supports many theories:

- Linear integer arithmetic, mixed linear/real arithmetic, ...
- Real difference logic, integer difference logic, ..
- Non-linear real arithmetic.
- Fixed size bit vectors
- Uninterpreted functions
- Arrays
- ... and can select automatically between them.

Other popular SMT solvers include MathSAT, Yices, CVC4.
Heavily used in program analysis, and interactive theorem provers.

Satisfiability modulo theories

Two perspectives on SMT:

1. General-purpose logic engines. Large, ambitious automated reasoning programs (Z3, MathSAT, Yices, CVC4). Common standardized input language SMT-LIB2. Z3 has $\approx$ 400k LOC.

2. Extend SAT solvers with domain-specific reasoning when needed. MiniSAT has $\approx$ 3k LOC. Diff. logic 0.3k LOC.

Satisfiability modulo theories

Two perspectives on SMT:

1. General-purpose logic engines. Large, ambitious automated reasoning programs (Z3, MathSAT, Yices, CVC4). Common standardized input language SMT-LIB2. Z3 has $\approx$ 400k LOC.

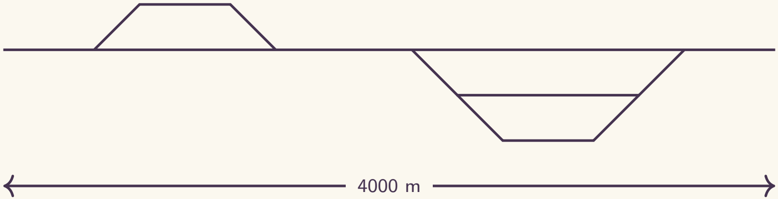
2. Extend SAT solvers with domain-specific reasoning when needed. MiniSAT has $\approx$ 3k LOC. Diff. logic 0.3k LOC.

   $\uparrow$ **Will present two case studies from railway**

Part II:
SAT-based algorithms in railway engineering

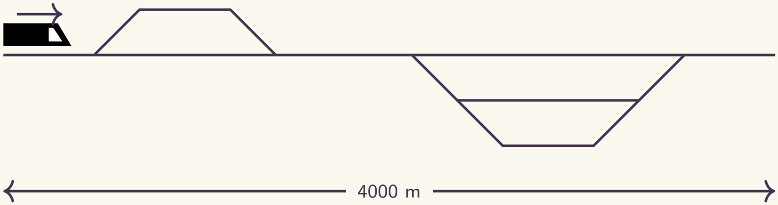
Railway control systems

Constructing a new railway line starts with a track plan:



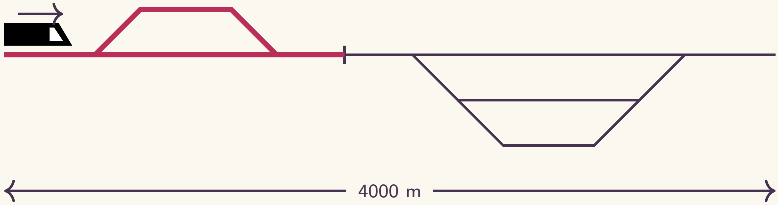
Railway control systems

Constructing a new railway line starts with a track plan:



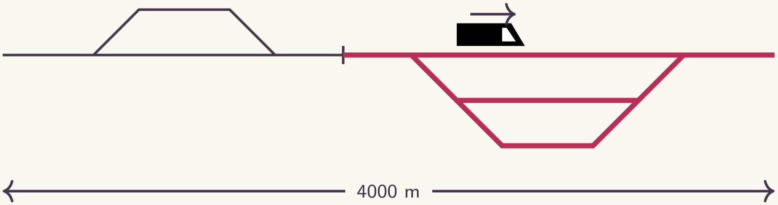
Railway control systems

By adding detectors, we can allocate smaller pieces of tracks to the train:



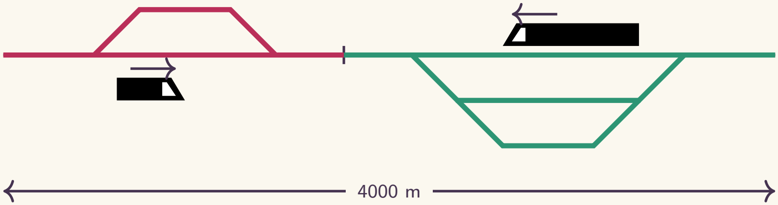
Railway control systems

By adding detectors, we can allocate smaller pieces of tracks to the train:



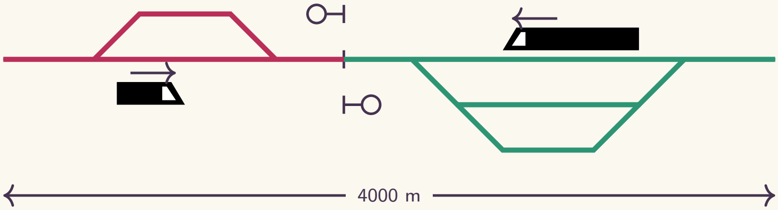
Railway control systems

Now, other trains can occupy different sections.



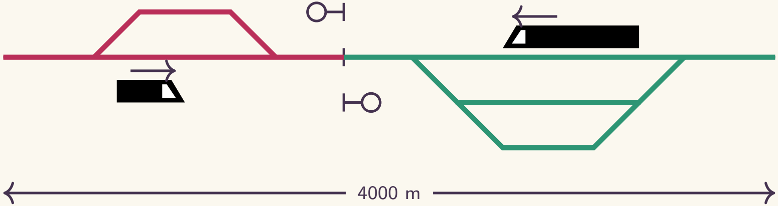
Railway control systems

We add signals to indicate to drivers when they can proceed.



Railway control systems

This situation is in principle safe, but is it a good design?



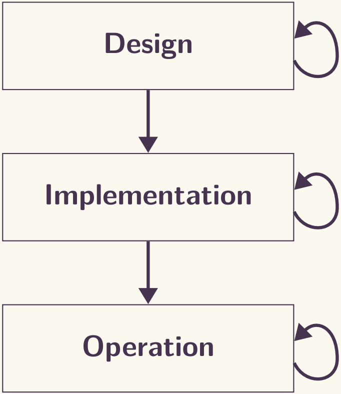
Requirements

Will my station design handle the actual traffic?

Two methods used in practice:

1. Whole-network time table analysis: a whole discipline in itself – complicated theory and software
2. Manual, ad-hoc analysis: varying quality, little documentation, low repeatability.

Design-implementation-operation



?

Formal methods for verifying correctness (safety) [3, 2].

Railway optimization for network-wide timetables [1, 4].

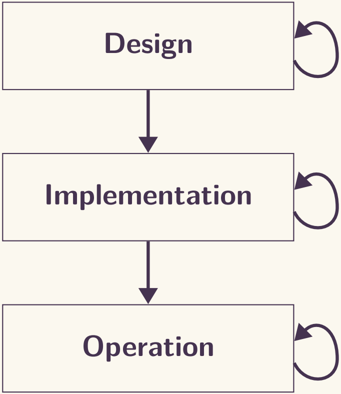
[1] M. Abril, F. Barber, L. Ingolotti, M.A. Salido, P. Tormos, and A. Lova. An assessment of railway capacity. *Transportation Research*, 44(5):774 – 806, 2008.

[2] Arne Borälv and Gunnar Stålmarck. Formal verification in railways. In *Industrial-Strength Formal Methods in Practice*, pages 329–350. Springer, 1999.

[3] A. Fantechi, W. Fokkink, and A. Morzenti. Some trends in formal methods applications to railway signalling. In *Formal Methods for Industrial Crit Sys.*, 2012.

[4] Alex Landex. *Methods to est. railway cap. and passenger delays*. PhD thesis, 2008.

Design-implementation-operation



Agile, fast verification methods with suitable, small specifications.

Formal methods for verifying correctness (safety).

Railway optimization for network-wide timetables.

Specification capture

Railway engineers gave us examples of performance properties that governed their designs.

Typical categories:

1. Running time (get from A to B)
   - Similar to a simulation test, but smaller specification.
2. Frequency (several consecutive trains)
   - Route trains into alternate tracks.
3. Overtaking
4. Crossing
   - Let one train wait on a side track while another train passes.

Capacity specifications

Local requirements suitable for construction projects.

- Operational scenario $S = (V, M, C)$:
- Vehicle types $V = \{(l_i, v_i^{\max}, a_i, b_i)\}$, defined by length, max velocity, max accel, max braking.
- Movements $M = \{(v_i, \langle q_i \rangle)\}$, defined by vehicle type v and ordered sequence of visits $\langle q_i \rangle$.

  - Each visit $q_i = (\{l_i\}, t_d)$ is a set of alternative locations l_i and an optional dwelling time t_d.

- Timing constraints $C = \{(q_a, q_b, t_c)\}$ which orders two visits and sets a maximum time from the first to the second $t_{q_a} < t_{q_b} < t_{q_a} + t_c$. The maximum time constraint can be omitted ($t_c = \infty$).

Constraints

Verification of these specifications would involve finding satisfying train trajectories and control system state:

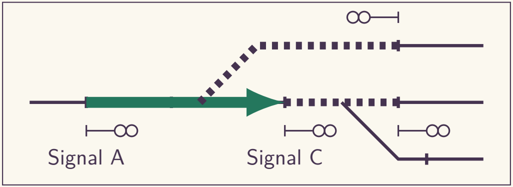
$$\exists p : \mathrm{spec}(p)$$

Also, constrained by:

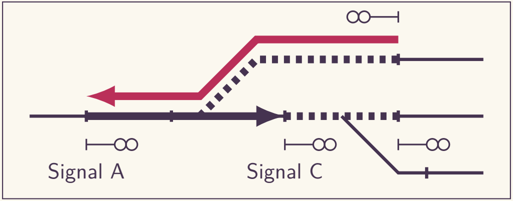
- 1 - Physical infrastructure
- 2 - Allocation of resources (collision safety)
- 3 - Limited communication
- 4 - Laws of motion

Constraints (2) Allocation of resources

An elementary route is a set of resources allocated together.

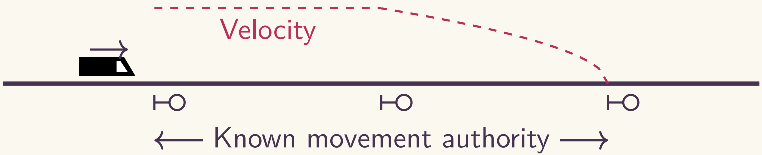


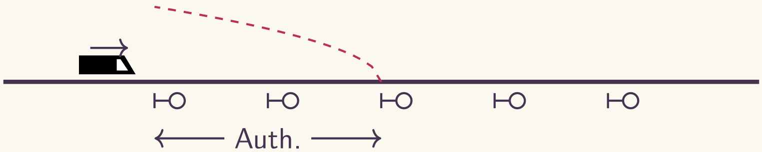
Routes are conflicting if they use any of the same resources.



Constraints (3) Limited communication

Signal information only carries across two signals (”pre-signalling”).

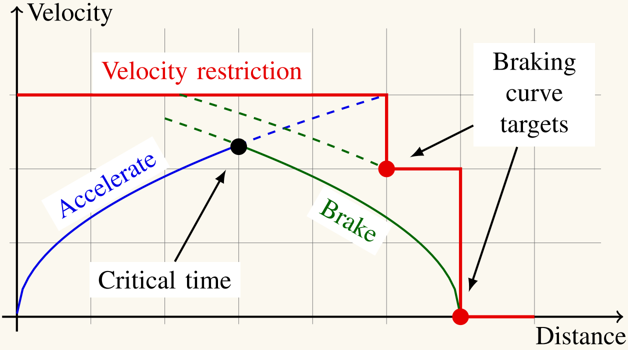




Constraints (4) Laws of motion

Trains move within the limits of given maximum acceleration and braking power. Train drivers need to plan ahead for braking so that the train respects its given movement authority and speed restrictions at all times.

$$v - v_0 \leq a\Delta t, \qquad v^2 - v_i^2 \leq 2bs_i.$$



Automated verification

Design-time capacity verification amounts to planning in a mixed discrete/continuous space.

Some suggestions:

- PDDL+, planning domain description language for mixed discrete-continuous planning domains [1].
- SMT with non-linear real arithmetic [2, 4].
- dReal: δ-complete decision proc. for FOL with reals [3].

Using these tools/techinques and straight-forward modeling did not make our problem manageable on relevant scales.

[1] M. Fox and D. Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.

[2] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *J. SAT*, 1:209–236, 2007.

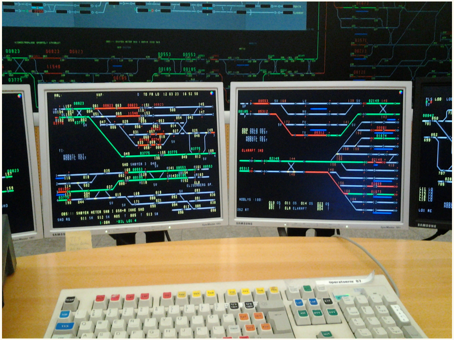
[3] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. CADE-24 vol. 7898 of *LNCS*, pages 208–214. Springer, 2013.

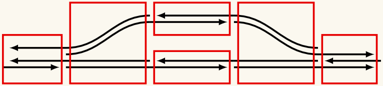
[4] D. Jovanovic and L. de Moura. Solving non-linear arithmetic. *ACM Comm. Computer Algebra*, 46(3/4):104–105, 2012.

Dispatch vs. driver

Split the planning work into two separate points of view:

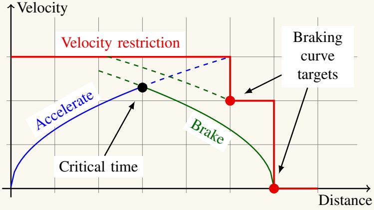
Dispatcher



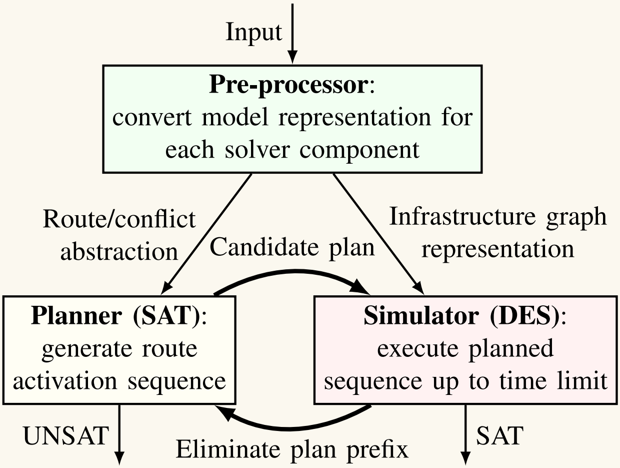


Elementary routes and their conflicts

Train driver

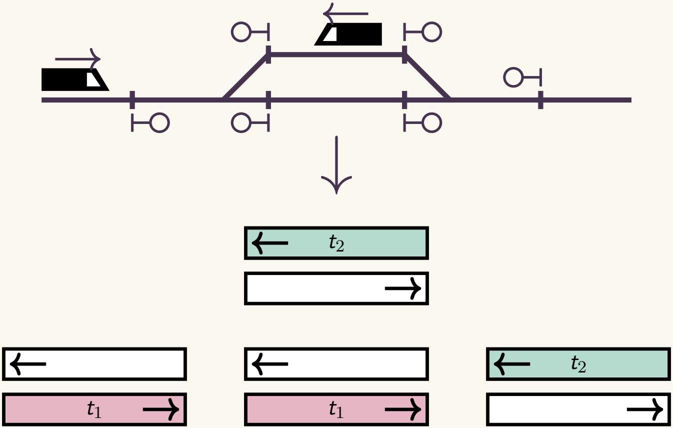


Local Capacity Solver architecture



SAT encoding of dispatch planning

General idea: represent which train occupies which elementary route in each of a sequence of steps.



SAT encoding

Planning as bounded model checking (BMC [1,2]). Build planning steps as needed using incremental SAT solver interface.

Movement correctness:

- Conflicting routes are not active simultaneously
  $\mathrm{conflict}(r_1, r_2) \Rightarrow o^i_{r_1} = \mathrm{Free} \vee o^i_{r_2} = \mathrm{Free}$.
- Elementary route allocation is consistent with train movement: $(o^i_r \neq t \wedge o^{i+1}_t = t) \Rightarrow$
  $\bigvee \left\{ o^{i+1}_{r_x} = t \mid \mathrm{route}(r_x), \mathrm{entry}(r) = \mathrm{exit}(r_x) \right\}$

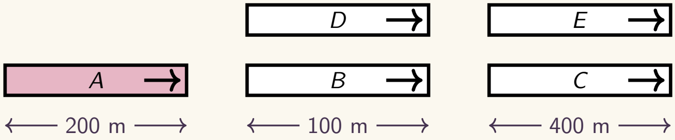
Satisfy specification:

- Visits happen in order (timing requirement is measured on simulation).

[1] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. Formal Methods in System Design, 19:7–34, 2001.

[2] J. F. Groote, S. F. M. van Vlijmen, and J. W. C. Koorn. The safety guaranteeing system at station Hoorn-Kersenboogerd. COMPASS '95, p. 57–68. IEEE, 1995.

Freeing



If A holds a train t of length 200.0 m, freeing A is constrained by:

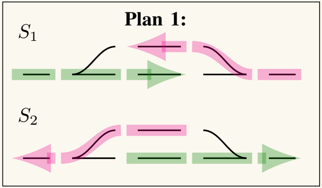
$$A^i \Rightarrow \left(A^{i+1} \vee (B^i \wedge C^i) \vee (D^i \wedge E^i)\right).$$

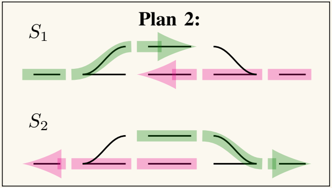
Eliminate equivalent solutions

- Can free $\implies$ must free
- Can allocate $\implies$ must allocate

- Exception to allocation: deferred progress
  a train may be waiting for a conflict to be resolved, even if the conflict starts in the future.

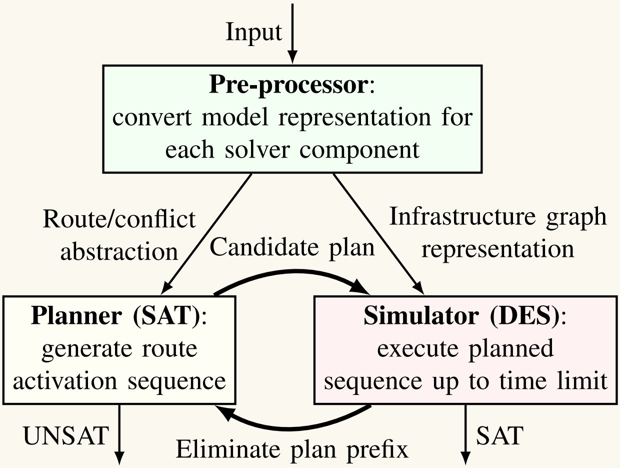
  Crossing example: **exactly two** solutions:



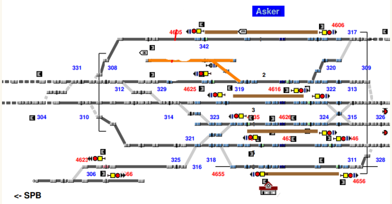


- Overlaps. Partial release.
- Loops in the infrastructure / loops in the dispatch.

Local Capacity Solver architecture



Case studies

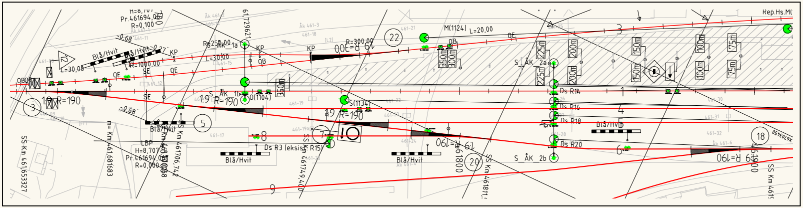


Infrastructure	Property	Result	n_{DES}	t_{SAT}	t_{DES}	t_{total}
Simple	Run.time	Sat.	1	0.00	0.00	0.00
(3 elem.)	Crossing	Unsat.	0	0.00	0.00	0.00
Two track	Run.time	Sat.	1	0.01	0.00	0.01
(14 elem.)	Frequency	Sat.	1	0.01	0.00	0.01
	Overtaking 2	Sat.	1	0.00	0.00	0.01
	Overtaking 3	Unsat.	0	0.01	0.00	0.01
	Crossing 3	Unsat.	0	0.01	0.00	0.01
Kolbotn (BN)	Run. time	Sat.	2	0.01	0.00	0.02
(56 elem.)	Overtake 4	Sat.	1	0.05	0.00	0.06
	Overtake 3	Unsat.	0	0.05	0.00	0.06
Eidsvoll (BN)	Run. time	Sat.	2	0.01	0.00	0.02
(64 elem.)	Overtake 2	Sat.	1	0.08	0.00	0.08
	Crossing 3	Sat.	1	0.04	0.00	0.04
	Crossing 4	Unsat.	0	0.21	0.00	0.21
Asker (BN)	Overtaking 2	Sat.	1	0.20	0.00	0.21
(170 elem.)	Overtaking 3	Unsat.	1	0.73	0.00	0.74
	Crossing 4	Sat.	0	0.75	0.00	0.77
Arna (CAD)	Run. time	Sat.	1	0.02	0.00	0.04
(258 elem.)	Overtaking 2	Sat.	1	0.50	0.00	0.51
	Overtaking 3	Sat.	1	1.43	0.00	1.45
	Crossing 4	Sat.	1	1.73	0.00	1.74
Gen. 3x3	High time	Sat.	1	0.01	0.00	0.01
(74 elem.)	Low time	Unsat.	27	0.18	0.01	0.19
Gen. 4x4	High time	Sat.	1	0.01	0.00	0.03
(196 elem.)	Low time	Unsat.	256	2.08	0.26	2.34
Gen. 5x5	High time	Sat.	1	0.06	0.00	0.09
(437 elem.)	Low time	Unsat.	3125	38.89	4.35	43.24

TABLE I: Verification performance on test cases, including Bane NOR (BN) and RailCOMPLETE (CAD) infrastructure models. The number of elementary routes (*elem.*) is shown for each infrastructure to indicate the model's size. n_{DES} is the number simulator runs, t_{SAT} the time in seconds spent in SAT solver, t_{DES} the time in seconds spent in DES, and t_{total} the total calculation time in seconds.

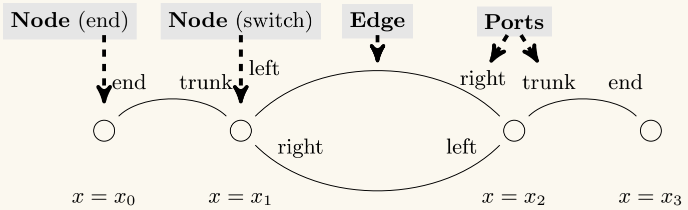
Schematic drawings: background

- Schematics used for visualizing operations, communicate system specifications, construction blueprints.
- Engineers need to coordinate 2D, 3D, and schematic drawings.
- Automated drawing from geographical and/or topological models can help engineers produce and update schematics efficiently.

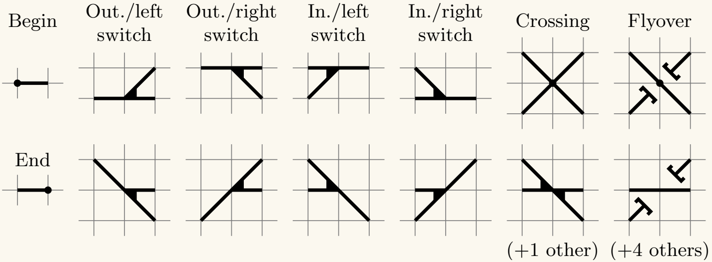


Schematic drawings: model

Topological representation extracted from CAD:



Node type variants:



Schematic drawings: constraints

Hard constraints:

- **Octilinearity**: 45 degree lines only.
- **Linear order**: nodes are ordered horizontally by "mileage".
- **Node shapes**: left/right branches recognizable.
- **Uniform vertical spacing**.

Soft constraints / optimization criteria:

- **Height / width** of the drawing
- Length of **diagonal** lines (non-horizontal lines)
- Number of **bends** (direction changes on lines)

Schematic drawings: encoding

- Horizontal distance between consecutive nodes: $\Delta x \in \{0, 1, \geq 2\}$.
- Short edge up/down indicator boolean q_j^{up}, q_j^{down}.
- Node vertical y_i and edge level l_j: unbounded integers in the theory of difference constraints.
- Node variant selection r_i.
- Edge direction values $d_i^{\text{begin}}, d_i^{\text{end}} \in \{\text{Up}, \text{Down}, \text{Straight}\}$

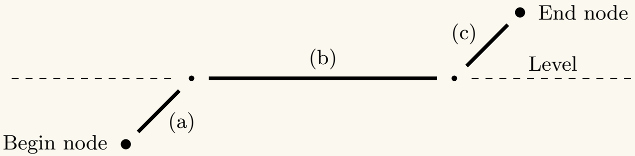
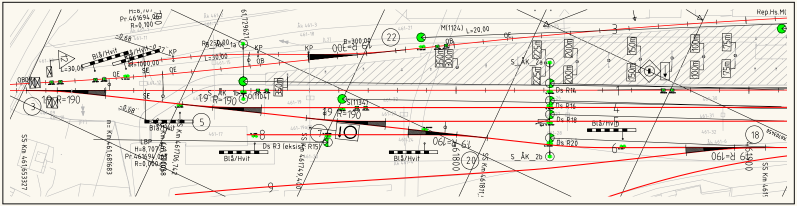
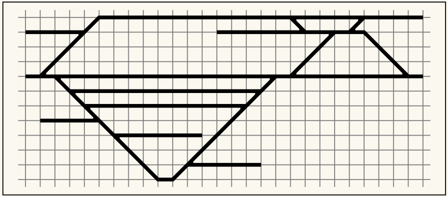


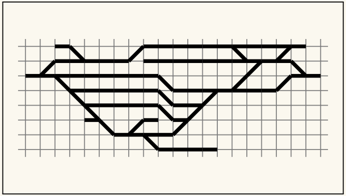
Fig. 6. The *edge level model* divides the edge into three sections on the horizontal axis: (a) the initial diagonal section from the left-most node to the edge level, (b) the middle horizontal section connecting the two diagonal sections, (c) the final diagonal section reaching the right-most node from the edge level. Any of these may have zero length.

Schematic drawings: optimization

For a set of constraints ϕ, we can perform numerical optimization on some number x by solving the sequence of formulas $\phi \wedge (x < m_1)$, $\phi \wedge (x < m_2)$, $\dots$, where the sequence m_i is a linear or binary search over the range of x, locating the smallest value that satisfies the constraints.







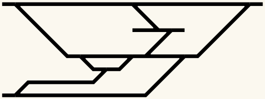
Schematic drawings: tool performance

Model	Src.	Size	Direct/SAT		Levels/SAT		Cross-sec./SAT			
			hwb	size (v/c)	bhw	size (v/c)	hwb	hbw	bhw	size (v/c)
Eidsvoll	[19]	35	60.7	57k/153k	0.02	2.3k/0.7k	0.05	0.06	0.33	4.0k/28k
Arna	RC	57	294	167k/493k	0.03	4.9k/1.3k	0.26	0.65	1.06	11k/100k
Asker	[19]	64	T/O	104k/295k	0.04	5.6k/2.0k	0.61	1.02	0.87	14k/124k
Weert	[6]	102	T/O	304k/969k	0.18	11k/4.0k	0.72	19.3	21.4	29k/327k
5x10	T	228	T/O	2.8M/13M	0.58	35k/2.7k	5.83	7.48	8.08	46k/364k
5x20	T	478	T/O	2.8M/12M	3.37	97k/7.7k	279	299	T/O	265k/4.2M
10x5	T	203	T/O	3.0M/14M	0.40	28k/2.0k	0.52	0.59	1.08	20k/83k
20x5	T	403	T/O	3.0M/14M	1.73	70k/4.0k	1.95	2.50	3.36	44k/165k
10x10	T	453	T/O	2.6M/12M	2.74	86k/5.5k	21.9	22.4	40.7	96k/727k
15x15	T	1053	T/O	2.3M/10M	22.7	255k/15k	T/O	T/O	T/O	N/A

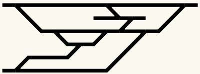
Table 1. Running times in seconds on a mid-range workstation. Time-outs (T/O) indicate exceeding 300 s. Model sizes are given as the sum of the number of nodes and edges. Models were obtained from BaneNOR [19], a RailCOMPLETE CAD project (RC), and adapted from [6]. Scaling test models (T) named $n \times m$ consist of n serially connected stations, each spreading out to m parallel tracks. Optimization criteria are height (h), width (w) and bends (b). The size columns show the number of SAT variables and clauses (v/c).

Schematic drawings: output examples (1/2)

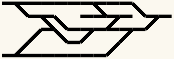
Model: Eidsvoll, imported from BaneNOR railML [19]



Levels/Lin.Prog.

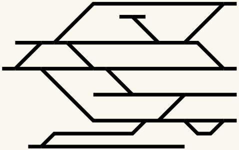


Levels/SAT

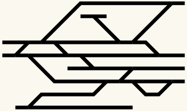


Cross-sec./SAT,
opt. width/height

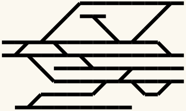
Model: Asker, imported from BaneNOR railML [19]



Levels/Lin.Prog.



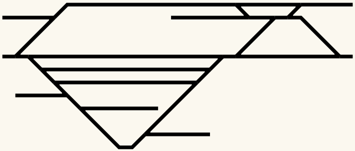
Levels/SAT



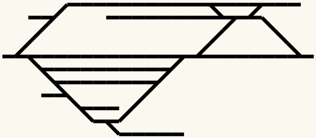
Cross-sec./SAT,
opt. height/bends

Schematic drawings: output examples (2/2)

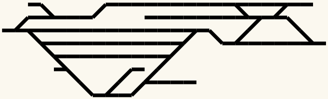
Model: Arna, imported from RailCOMPLETE CAD project



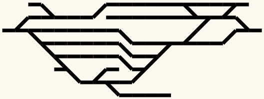
Levels/SAT



Cross-sec./SAT, opt. bends/width

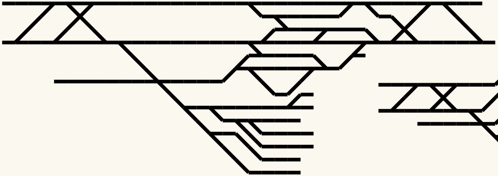


Cross-sec./SAT, opt. height/bends

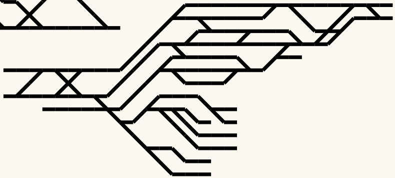


Cross-sec./SAT, opt. height/width

Model: Weert, remodeled from figures in [6]



Cross-sec./SAT, opt. height/bends



Cross-sec./SAT, opt. height/width

Conclusions?

Consider SAT/SMT for operations research if your constraint/optimization problem is:

- Program-like domain: lists, arrays, etc.
- Real/integer arithmetic with complex Boolean structure.
- Integer problems with small domains.
- Lexicographical objectives.

Advanced free solvers available.

Solvers can be taken apart and tailored to your problem..

Competetive with CPLEX on LGDB problems (MILP + big-M) (Sebastiani, Tomasi, 2012).