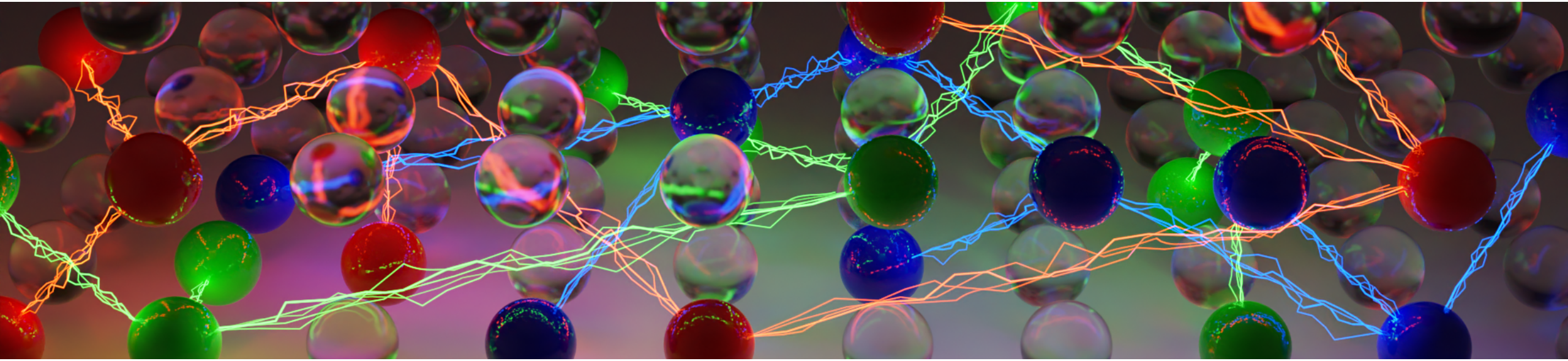
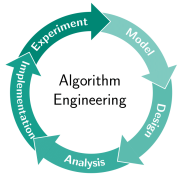


Practical SAT Solving



Lecture 7 – Redundancy Notions and Proofs of Unsatisfiability

Markus Iser, Dominik Schreiber | June 2, 2025

SAtRes
Scalable Automated Reasoning

Conflict-driven Clause Learning (CDCL)

Recap: Preprocessing

- Subsumption
- Self-subsuming Resolution
- Bounded Variable Elimination
- Blocked Clause Elimination
- Relationship with Tseitin Encoding
- Scheduling
  - Heuristic Bounds
  - Interleaving of Techinques (f.ex. Round Robin)
  - Inprocessing: Interleave with solving

Algorithm 1: CDCL(CNF Formula F, &Assignment A $\leftarrow \emptyset$)

```
1  if not PREPROCESSING then return UNSAT
2  while A is not complete do
3      UNIT PROPAGATION
4      if A falsifies a clause in F then
5          if decision level is 0 then return UNSAT
6          else
7              (clause, level) ← CONFLICT-ANALYSIS
8              add clause to F and backtrack to level
9              continue
10     if RESTART then backtrack to level 0
11     if CLEANUP then forget some learned clauses
12     if not INPROCESSING then return UNSAT
13     BRANCHING
14 return SAT
```

Propagation-based Redundancy Notions

Let a formula F, and a literal x be given.

Failed Literal Probing

If $F \wedge x \vdash_{UP} \bot$, then $F \models \neg x$

$\implies$ add $\{\neg x\}$ to F

Example (Failed Literal Probing)

Let $F := \big\{\{a, b\}, \{a, \neg b\}\big\}$.

Probing with $\neg a$ results in a conflict, i.e., $F \wedge \neg a \vdash_{UP} \bot$.

Ergo, we can deduce $F \equiv F \wedge a$.

Propagation-based Redundancy Notions

Let a formula F, a clause $C \in F$, and a literal $x \in C$ be given.

Asymmetric Literal Elimination (ALE)

If $F \setminus C \wedge \overline{C \setminus \{x\}} \vdash_{UP} \overline{x}$, then $F \models C \setminus \{x\}$.

$\implies$ strengthen C to $C \setminus \{x\}$

Example (Asymmetric Literal Elimination (ALE))

Let $F := \{\{a, b\}, \{\neg b, \neg c\}, \{a, c, d\}\}$, $C := \{a, c, d\}$, and $x := c$.

ALE results in the following propagation: $\{\{a, b\}, \{\neg b, \neg c\}, \{\neg a\}, \{\neg d\}\} \vdash_{UP} \neg c$.

Ergo, we can deduce $F \models \{a, d\}$.

F can not have a model which satisfies $\{a, c, d\}$, but not $\{a, d\}$.

Propagation-based Redundancy Notions

Let a formula F, a clause $C \in F$, and a literal $x \in C$ be given.

Asymmetric Tautology Elimination (ATE)

If $F \setminus C \wedge \overline{C} \vdash_{UP} \bot$, then $F \models C$.

$\Longrightarrow$ remove C from F

Example (Asymmetric Tautology Elimination (ATE))

Let $F := \{\{a, b, c\}, \{\neg b, d\}, \{a, c, d\}\}$, and $C := \{a, c, d\}$.

ATE results in the following propagation: $\{\{a, b, c\}, \{\neg b, d\}, \{\neg a\}, \{\neg c\}, \{\neg d\}\} \vdash_{UP} \bot$,

Ergo, we can deduce $F \equiv F \setminus \{a, c, d\}$.

$\{a, c, d\}$ follows from the other clauses in F.

Propagation-based Redundancy Notions

Variants: Efficient Algorithms and Implementations

- **Hidden Tautology Elimination (HTE) / Hidden Literal Elimination (HLE)**
  Restricted forms of ATE/ALE which only propagate over binary clauses.
  Efficient HLE algorithm based on randomized DFS and application of paranthesis theorem: **Unhiding**[1]
- **Distillation / Vivification**
  Interleave assignment and propagation to detect ATs / ALs early on.
- **Avoidance of Redundant Propagations**
  Sort literals and clauses in a formula to simulate a trie, and reuse propagations that share the same prefix.

[1] 2011, Heule et al., Efficient CNF Simplification Based on Binary Implication Graphs

Recap.

Propagation-based Preprocessing

- Propagation-based Redundancy Notions:

  Failed Literal Probing, Asymmetric Literal Elimination, Asymmetric Tautology Elimination
- Efficient Implementation of Propagation-based Redundancy Removal
- Autarkies: Partial assignments that satisfy all touched clauses

Next Up

Proofs of Unsatisfiability

Relationship with Proof Checking

Generalizations of Blocked Clauses

- **Reverse Unit Propagation (RUP)**
  A clause has the property RUP if and only if it is an Asymmetric Tautology (AT).

  In CDCL, learned clauses are RUP at the moment of their learning.

- **Resolution Asymetric Tautologies (RATs)**
  A clause C is a RAT in a formula F if it contains a literal x such that
  each resolvent in $C \otimes_x F_{\overline{x}}$ is an asymmetric tautology.

  Blocked Sets in particular are RATs.

Proof Checking

SAT Solvers are complex software systems, and bugs are not uncommon.

Trustworthiness of SAT Solvers

- For satisfiable instances, SAT solvers can output the found assignment
- For unsatisfiable instances, SAT solvers can output a proof of unsatisfiability
- Both can be checked independently from the solver by much simpler, and **formally verified program**.

Example (Applications of Proof Checking)

Unsatisfiability of a formula might proof important properties of the problem at hand, such as:

- Absence of certain bugs in a hardware design or software verification,
- Optimality of a certain makespan in planning
- etc. pp.

Feasibility of Proof Checking

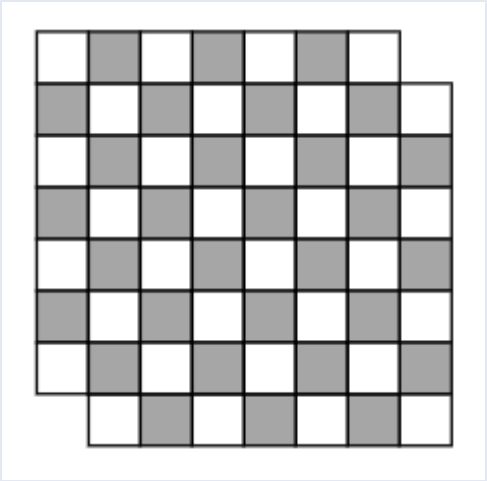
RAT proof checking is polynomial in the proof-size, but proof-size is worst-case exponential in the formula-size.

Pragmatics: if we could generate it, we can also check it.

Proof Systems: Motivating Example

Example (Mutilated Chessboards)

Let a chessboard be given with two diagonally opposite corners removed.
Is it possible to cover the remaining board with dominoes?

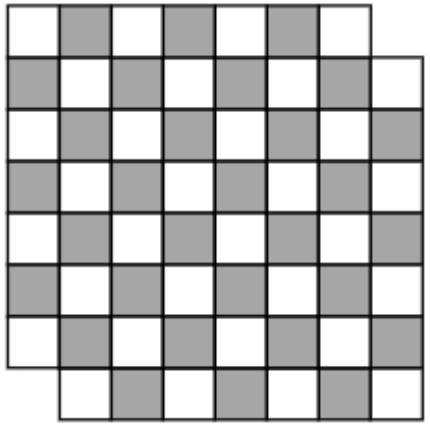


Proof Systems: Motivating Example

Example (Mutilated Chessboards)

Let a chessboard be given with two diagonally opposite corners removed.
Is it possible to cover the remaining board with dominoes?

Human: No, because each dominoe covers exactly one black and one white field, and there are two more black fields than white fields.

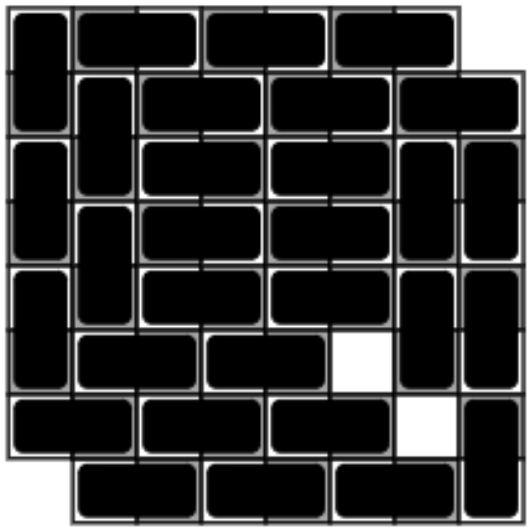


Proof Systems: Motivating Example

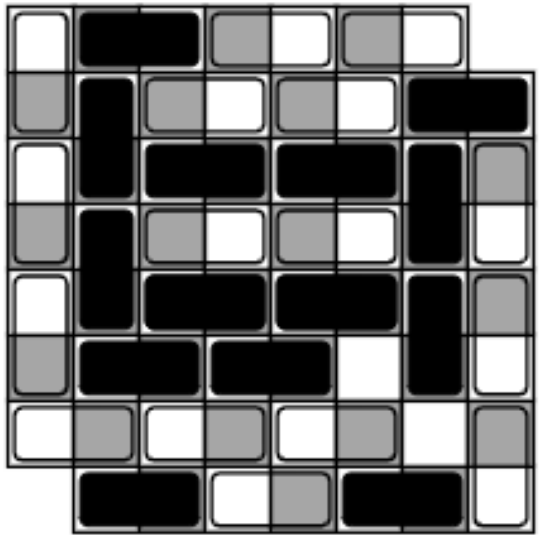
Example (Mutilated Chessboards)

Let a chessboard be given with two diagonally opposite corners removed.
Is it possible to cover the remaining board with dominoes?

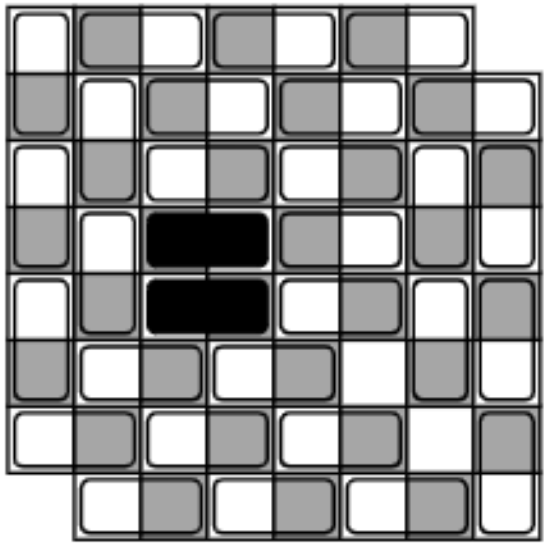
SAT Solver: Let's try and learn.



Impasse Detected



→ Resolution



→ More powerful Proof-system[2]

[2]Example taken from SAT lecture of Marijn J.H. Heule

Proof Complexity

Proof complexity is the study of the size of proofs in different proof systems.

Relationship with SAT Solving

- Static analysis without algorithmic considerations
- Questions of automizability not addressed
- Lower bounds on proof size tell us how good a SAT solver can be in the best case
- Upper bounds on proof size tell us how good a SAT solver should be

Resolution

Resolution

- Preserves equivalence
- CDCL is as powerful as General Resolution
- Well-known Exponential Lower Bounds:

  For example, proofs of unsatisfiabilty of Pigeon Hole formulas,
  are necessarily of exponential length in the resolution proof system (Haken 1985)
- Heuristic for Finding Resolution Proofs: CDCL

Blocked Clauses

Extended Resolution (Tseitin 1966)

- Preserves satisfiability
- Extended Resolution:
  incorporate extension rule $x \leftrightarrow a \wedge b$ for some a, b in formula and a new variable x
- No Lower Bounds known
- (Cook 1967) polynomial sized ER proof for PH formulas

Blocked Clauses (Kullmann 1999)

- Preserves satisfiability
- Generalization of ER Proof systems
- Allows the addition and removal of blocked clauses
- No Lower Bounds known

Stronger Proof-Systems without new Variables

Can we get stronger without introducing new variables?
In the following: C is redundant with respect to F means that F and $F \wedge C$ are equisatisfiable.

Example (Implication-based Redundancy)

Given $F := \{\{x, y, z\}, \{\neg x, y, z\}, \{x, \neg y, z\}, \{\neg x, \neg y, z\}\}$, and $G := \{\{z\}\}$,
G is at least as satisfiable as F since $F \models G$

Implication-based Redundancy Notion

A clause C is redundant w.r.t. formula F iff there exists an assignment ω such that $F \wedge \neg C \models (F \wedge C)|_\omega$[3]

In other words: Potential models of F falsifiying C are still models of F and C modulo an assignment ω

In Practice: Propagation Redundancy

Approximate $F \wedge \neg C \models (F \wedge C)|_\omega$ with unit propagation: $F \wedge \neg C \vdash_{UP} (F \wedge C)|_\omega$

$\rightarrow$ Efficiently Checkable Proofs: Tan et al., Verified Propagation Redundancy Checking in CakeML, TACAS 21

[3] Given an assignment α, the formula $F|_\alpha$ is the formula after removing from F all clauses that are satisfied and all literals falsified by α

Theorem: Clause Redundancy via Implication

Let F be a formula, C a non-empty clause, and α the assignment blocked by C.
C is redundant with respect to F *if and only if* there exists an assignment ω such that ω satisfies C and $F|_\alpha \models F|_\omega$.[4]

Proof "only if"

Assume F and $F \wedge C$ are equisatisfiable. Show that there exists an ω satisfying C and $F|_\alpha \models F|_\omega$.

If $F|_\alpha$ is unsatisfiable, then the semantic implication trivially holds.

Assume now that $F|_\alpha$ is satisfiable, implying that F is satisfiable.

Since F and $F \wedge C$ are equisatisfiable, there exists an assignment ω that satisfies both F and C.

Since ω satisfies F, it holds that $F|_\omega = \emptyset$ and so $F|_\alpha \models F|_\omega$.

[4] Heule et al., 2019, Strong Extension-Free Proof Systems

Theorem: Clause Redundancy via Implication

Let F be a formula, C a non-empty clause, and α the assignment blocked by C.
C is redundant with respect to F *if and only if* there exists an assignment ω such that ω satisfies C and $F|_\alpha \models F|_\omega$.[4]

Proof "if"

Assume there exists an assignment ω satisfying C and $F|_\alpha \models F|_\omega$. Show that F and $F \wedge C$ are equisatisfiable.

Let γ be a (total) assignment that satisfies F and falsifies C.
Since γ satisfies F, it must satisfy $F|_\alpha$ and since $F|_\alpha \models F|_\omega$, it must also satisfy $F|_\omega$.

Now, we can turn γ into a satisfying assignment γ' for $F \wedge C$ as follows:

$$\gamma'(x) = \begin{cases} \omega(x) & \text{if } x \in \text{vars}(\omega) \\ \gamma(x) & \text{otherwise} \end{cases}$$

Since ω satisfies C, γ' satisfies C.

Since γ satisfies $F|_\omega$, and $\text{vars}(F|_\omega) \subseteq \text{vars}(\gamma) \setminus \text{vars}(\omega)$, γ' satisfies F.

$\Rightarrow \gamma'$ satisfies $F \wedge C$.

[4] Heule et al., 2019, Strong Extension-Free Proof Systems

Autarkies

Autarky

Let a formula F and a partial assignment A be given. A clause $C \in F$ is ...

- ... touched by A if it contains a variable assigned in A
- ... satisfied by A if it contains a literal assigned to *True* by A

An autarky is a partial assignment A such that all touched clauses are satisfied.

Let α be an autarky of F. Then, F and $F|_\alpha$ are equisatisfiable.[5]
$\Rightarrow$ All clauses touched by an autarky can be removed.

Edge Cases: Pure Literals and Satisfying Assignments are Autarkies

Application in Inprocessing: Kissat analyses assignments found by sprints of local search to find autarkies.

Example

The partial assignment $A = \{\neg a, \neg c\}$ is an autarky for $F := \big\{\{\neg a, b\}, \{b, \neg c\}, \{a, \neg b, \neg c\}\big\}$

[5] The notion of *autark assignments* dates back to Monien and Speckenmeyer, Solving satisfiability in less than 2^n steps, 1985

Pruning Branches with Conditional Autarkies

Conditional Autarkies

An assignment $\alpha = \alpha_{con} \cup \alpha_{aut}$ is a conditional autarky of F if α_{aut} is an autarky of $F|_{\alpha_{con}}$

Then F and $F \wedge (\alpha_{con} \rightarrow \alpha_{aut})$ are equisatisfiable.

Example (Pruning Branches with a Conditional Autarky)

Let $F := \{\{x, y\}, \{x, \neg y\}, \{\neg y, \neg z\}\}$, and let $\alpha_{con} = \{x\}$ and $\alpha_{aut} = \{\neg y\}$.

Then $F|_{\alpha_{con}} = \{\{\neg y, \neg z\}\}$ and $\alpha_{aut} = \{\neg y\}$ is an autarky of $F|_{\alpha_{con}}$, such that $\{x\} \cup \{\neg y\}$ is a conditional autarky of F.

We can thus learn the clause $\{\neg x, \neg y\}$.

Satisfaction Driven Clause Learning (SDCL)

Idea: Also learn clauses if no conflict is detected, but a positive reduct is satisfiable.

Positive Reduct

Let a formula F and a partial, non-conflicting assignment α be given.

The positive reduct $p(F, \alpha)$ is the formula that contains all clauses of F satisfied by α and a clause $C := \neg\alpha$.

A satisfying assignment ω of the positive reduct $p(F, \alpha)$ is a conditional autarky of F.

$\Rightarrow$ if the positive reduct is satisfiable, then the branch α can be pruned.

Key Idea of SDCL:

While solving a formula, check the positive reducts of current assignments α for satisfiability.

If $p(F, \alpha)$ is satisfiable, prune the branch α.

Positive reducts are typically very easy to solve.[6]

[6] Heule et al., 2017, PRuning Through Satisfaction

From Modern to “Post-modern” SAT Solving

Problem: Automizability, how to find such short proofs?

Solving the Problem of Automizability: Practical Approaches

- **Resolution**: Clause Learning

  → *any classic CDCL implementation*

- **Extended Resolution**: Structural Boundend Variable Addiction (SBVA).

  → `SBVA-CaDiCaL`: Winner of SAT Competition 2023

- **PReLearning**: Preprocessing adds specific Propagation Redundant (PR) clauses

  → `KissatMAB-Prop`: Winner of SAT Competition 2023 on UNSAT instances

- **Symmetry Breaking Predicates**: Exclusion of Symmetric Solutions

  → `BreakId-Kissat`: Special Price at SAT Competition 2023

Recap.

Recap.

- Propagation-based Redundancy Notions
- Proof Systems: Resolution, Extended Resolution, Blocked Clauses, Implication-based Redundancy
- Autarkies, Conditional Autarkies, and Satisfaction Driven Clause Learning (SDCL)