

Practical SAT Solving

Lecture 2

Markus Iser, Dominik Schreiber, Tomáš Balyo | April 22, 2024



Overview

Recap. Lecture 1

- Satisfiability: Propositional Logic, CNF Formulas, NP-completeness, Applications
- Examples: Pythagorean Triples, Arithmetic Progressions, k-Colorability
- Incremental SAT: IPASIR, Sample Code

Today's Topics

- Tractable Subclasses
- Constraint Encodings
- Encoding Techniques

Tractable Subclasses

Tractable Subclasses

Tractable Subclasses (cf. Schaefer, 1978)

- **2-SAT**
Exactly two literals per clause
- **HORN-SAT**
At most one positive literal per clause
- **Inverted HORN-SAT**
At most one negative literal per clause
- **Positive / Negative**
Literals occur only pure (either positive or negative)
- **XOR-SAT**
No clauses, only XOR constraints

2-SAT

aka. Binary or Quadratic SAT

Each clause has exactly two literals.

Example (2-SAT Formulas)

$$F_5 = \{\{x_1, x_2\}, \{\overline{x_1}, x_2\}, \{x_1, \overline{x_2}\}, \{\overline{x_1}, \overline{x_2}\}\}$$

$$F_7 = \{\{\overline{x_1}, x_2\}, \{\overline{x_2}, x_3\}, \{\overline{x_3}, x_1\}, \{x_2, x_4\}, \{x_3, x_4\}, \{x_1, x_3\}\}$$

2-SAT

aka. Binary or Quadratic SAT

Each clause has exactly two literals.

Example (2-SAT Formulas)

$$F_5 = \{\{x_1, x_2\}, \{\overline{x_1}, x_2\}, \{x_1, \overline{x_2}\}, \{\overline{x_1}, \overline{x_2}\}\}$$

$$F_7 = \{\{\overline{x_1}, x_2\}, \{\overline{x_2}, x_3\}, \{\overline{x_3}, x_1\}, \{x_2, x_4\}, \{x_3, x_4\}, \{x_1, x_3\}\}$$

Linear Time Algorithm for 2-SAT (cf. Aspvall et al., 1979)

- Construct Implication Graph:

Directed graph with a vertex for each literal and two edges $\bar{l}_1 \rightarrow l_2$ and $\bar{l}_2 \rightarrow l_1$ for each clause $\{l_1, l_2\}$

- Find Strongly Connected Components (SCC)

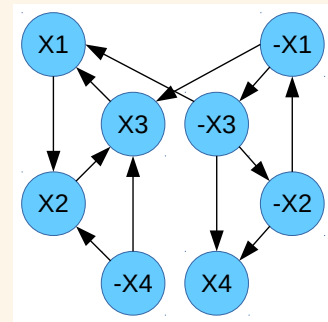
SCC: There is a path from every vertex to every other vertex

- Check for Complementary Literals in the same SCC

Implication Graph

Example (Implication Graph)

$$F_7 = \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_2}, x_3 \}, \{ \overline{x_3}, x_1 \}, \{ x_2, x_4 \}, \{ x_3, x_4 \}, \{ x_1, x_3 \} \}$$

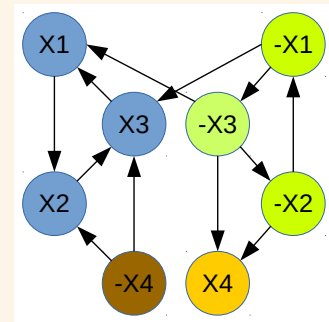


Implication Graph

Example (Implication Graph)

$$F_7 = \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_2}, x_3 \}, \{ \overline{x_3}, x_1 \}, \{ x_2, x_4 \}, \{ x_3, x_4 \}, \{ x_1, x_3 \} \}$$

- Find Strongly Connected Components (SCC)
- Tarjan's algorithm finds SCCs in $\mathcal{O}(|V| + |E|)$
- Complexity: $\mathcal{O}(n + m)$, where m is the number of clauses

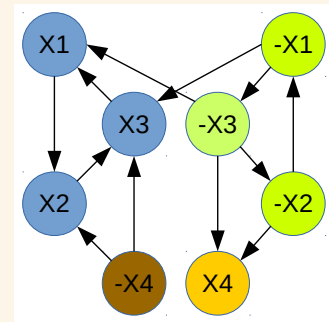


Implication Graph

Example (Implication Graph)

$$F_7 = \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_2}, x_3 \}, \{ \overline{x_3}, x_1 \}, \{ x_2, x_4 \}, \{ x_3, x_4 \}, \{ x_1, x_3 \} \}$$

- If an SCC contains both x and \overline{x} , the formula is **UNSAT**
 - Why?

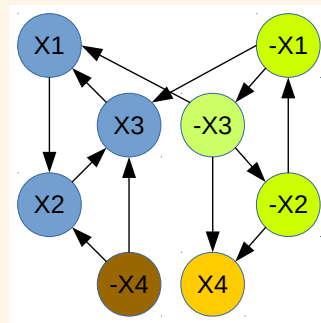


Implication Graph

Example (Implication Graph)

$$F_7 = \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_2}, x_3 \}, \{ \overline{x_3}, x_1 \}, \{ x_2, x_4 \}, \{ x_3, x_4 \}, \{ x_1, x_3 \} \}$$

- If an SCC contains both x and \overline{x} , the formula is **UNSAT**
 - x implies its own negation!
 - Literals in an SCC must be either all true or all false
- What about **SAT**? How to get a solution?

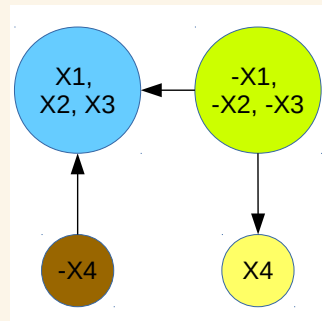


Implication Graph

Example (Implication Graph)

$$F_7 = \{\{\bar{x}_1, x_2\}, \{\bar{x}_2, x_3\}, \{\bar{x}_3, x_1\}, \{x_2, x_4\}, \{x_3, x_4\}, \{x_1, x_3\}\}$$

- If an SCC contains both x and \bar{x} , the formula is **UNSAT**
 - x implies its own negation!
 - Literals in an SCC must be either all true or all false
- What about **SAT**? How to get a solution?
 - Contract each SCC into one vertex
 - In reverse topological order, set unassigned literals to true.



HornSAT

Each clause contains at most one positive literal.

Example (Horn Formula)

Each clause can be written as an implication with **positive literals only** and a single consequent:

$$\begin{aligned} F_6 &= \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_1}, \overline{x_2}, x_3 \}, \{ x_1 \} \} \\ &\equiv (x_1 \rightarrow x_2) \wedge ((x_1 \wedge x_2) \rightarrow x_3) \wedge (\top \rightarrow x_1) \end{aligned}$$

HornSAT

Each clause contains at most one positive literal.

Example (Horn Formula)

Each clause can be written as an implication with **positive literals only** and a single consequent:

$$\begin{aligned} F_6 &= \{ \{ \overline{x_1}, x_2 \}, \{ \overline{x_1}, \overline{x_2}, x_3 \}, \{ x_1 \} \} \\ &\equiv (x_1 \rightarrow x_2) \wedge ((x_1 \wedge x_2) \rightarrow x_3) \wedge (\top \rightarrow x_1) \end{aligned}$$

Solving Horn Formulas

- Propagate until fixpoint
- If $\top \rightarrow \perp$ then the formula is **UNSAT**. Otherwise it is **SAT**.
- Construct a satisfying assignment by setting the remaining variables to false

Hidden Horn

aka. Renamable or Disguised Horn

A CNF formula is **Hidden Horn** if it can be made **Horn** by flipping the polarity of some of its variables.

Example (Hidden Horn Formula)

$$F_8 = \{ \{x_1, x_2, x_4\}, \{x_2, \overline{x_4}\}, \{x_1\} \}$$
$$\rightsquigarrow \{ \{ \overline{x_1}, \overline{x_2}, x_4 \}, \{ \overline{x_2}, \overline{x_4} \}, \{ \overline{x_1} \} \}$$

How to recognize a Hidden Horn formula? And how to hard is it?

Hidden Horn

aka. Renamable or Disguised Horn

A CNF formula is **Hidden Horn** if it can be made **Horn** by flipping the polarity of some of its variables.

Example (Hidden Horn Formula)

$$F_8 = \{\{x_1, x_2, x_4\}, \{x_2, \overline{x_4}\}, \{x_1\}\}$$
$$\rightsquigarrow \{\{\overline{x_1}, \overline{x_2}, x_4\}, \{\overline{x_2}, \overline{x_4}\}, \{\overline{x_1}\}\}$$

How to recognize a Hidden Horn formula? And how to hard is it?

Recognizing Hidden Horn Formula F

Construct 2-SAT formula R_F that contains the clause $\{l_1, l_2\}$ iff there is a clause $C \in F$ such that $\{l_1, l_2\} \subseteq C$.

- Example: $R_{F_8} = \{\{x_1, x_2\}, \{x_1, x_4\}, \{x_2, x_4\}, \{x_2, \overline{x_4}\}\}$
- If the 2-SAT formula is satisfiable, then F is Hidden Horn
- If $x_i = \text{true}$ in ϕ , then x_i needs to be renamed to $\overline{x_i}$

Mixed Horn

A CNF formula is **Mixed Horn** if it contains only binary and Horn clauses.

Example (Mixed Horn Formula)

$$F_9 = \{ \{ \overline{x_1}, \overline{x_7}, x_3 \}, \{ \overline{x_2}, \overline{x_4} \}, \{ x_1, x_5 \}, \{ x_3 \} \}$$

How to solve a Mixed Horn formula? **And how to hard is it?**

Mixed Horn

A CNF formula is **Mixed Horn** if it contains only binary and Horn clauses.

Example (Mixed Horn Formula)

$$F_9 = \{ \{ \overline{x_1}, \overline{x_7}, x_3 \}, \{ \overline{x_2}, \overline{x_4} \}, \{ x_1, x_5 \}, \{ x_3 \} \}$$

How to solve a Mixed Horn formula? **And how hard is it?**

Mixed Horn is NP-complete

Proof: Reduce SAT to Mixed Horn SAT

For each non-Horn non-binary clause $C = \{l_1, l_2, l_3, \dots\}$,

- for each but one positive $l_i \in C$ introduce a new variable l'_i and replace l_i in C by $\overline{l'_i}$
- add clauses $\{l'_i, l_i\}, \{\overline{l'_i}, \overline{l_i}\}$ to establish $l_i = \overline{l'_i}$

Next up: CNF Encodings

Elementary Encodings

- Tseitin Transformation
- Cardinality Constraints
- Finite Domain Encodings

Properties of Encodings

- Size: Number of Variables and Clauses
- Propagation consistency: Can the encoding ensure consistency through propagation?

Encoding Circuits

Given a propositional formula F with operations \wedge , \vee , and \neg , how can it be encoded in CNF?

Example (CNF Conversion)

$$F = \neg((\neg x \vee y) \wedge (\neg z \wedge \neg(x \wedge \neg w)))$$

(Given Formula)

Naive / Direct Conversion

Encoding Circuits

Given a propositional formula F with operations \wedge , \vee , and \neg , how can it be encoded in CNF?

Example (CNF Conversion)

$$F = \neg((\neg x \vee y) \wedge (\neg z \wedge \neg(x \wedge \neg w)))$$

(Given Formula)

$$= (x \wedge \neg y) \vee z \vee (x \wedge \neg w)$$

(Negation Normal Form)

Naive / Direct Conversion

- Convert to Negation Normal Form (NNF)

Encoding Circuits

Given a propositional formula F with operations \wedge , \vee , and \neg , how can it be encoded in CNF?

Example (CNF Conversion)

$$F = \neg((\neg x \vee y) \wedge (\neg z \wedge \neg(x \wedge \neg w))) \quad \text{(Given Formula)}$$

$$= (x \wedge \neg y) \vee z \vee (x \wedge \neg w) \quad \text{(Negation Normal Form)}$$

$$= (x \vee z) \wedge (x \vee z \vee \neg w) \wedge (\neg y \vee z \vee x) \wedge (\neg y \vee z \vee \neg w) \quad \text{(Conjunctive Normal Form)}$$

Naive / Direct Conversion

- Convert to Negation Normal Form (NNF)
- Apply distributive law to get CNF
- Problem: Applying the distributive law may result in an exponential blow-up.

Tseitin Encoding

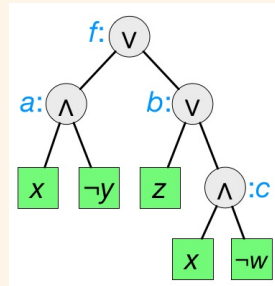
Idea: Introduce new variables for subformulas.

Example (Tseitin Conversion)

$$F = (x \wedge \neg y) \vee z \vee (x \wedge \neg w) \quad (\text{Negation Normal Form})$$

$$\stackrel{\text{SAT}}{=} (c \leftrightarrow x \wedge \neg w) \wedge \dots \wedge (f \leftrightarrow a \vee b) \wedge f \quad (\text{Tseitin Encoding})$$

- Define new variables: $a \leftrightarrow x \wedge \neg y$, $f \leftrightarrow a \vee b$, ...
- Encode definitions in CNF: $(\bar{f} \vee a \vee b) \wedge (f \vee \bar{a}) \wedge (f \vee \bar{b}) \wedge \dots$
- One additional clause (f) to assert that F must be true



Tseitin Encoding

The Tseitin-Encoding $\mathcal{T}(F)$ of a propositional formula F over connectives $\{\wedge, \vee, \neg\}$ is specified as follows.

Definition of Tseitin Encoding

$$\mathcal{T}(F) = d_F \wedge \mathcal{T}^*(F) \quad \text{(Root Formula)}$$

$$\mathcal{T}^*(F) = \begin{cases} \mathcal{T}_{\text{def}}(F) \wedge \mathcal{T}^*(G) \wedge \mathcal{T}^*(H), & \text{if } F = G \circ H \text{ and } \circ \in \{\wedge, \vee\} \\ \mathcal{T}_{\text{def}}(F) \wedge \mathcal{T}^*(G), & \text{if } F = \neg G \\ \text{True}, & \text{if } F \in \mathcal{V} \end{cases} \quad \text{(Recursion)}$$

$$\mathcal{T}_{\text{def}}(F) = \begin{cases} (\overline{d_F} \vee d_G) \wedge (\overline{d_F} \vee d_H) \wedge (d_F \vee \overline{d_G} \vee \overline{d_H}), & \text{if } F = G \wedge H \\ (\overline{d_F} \vee d_G \vee d_H) \vee (d_F \vee \overline{d_G}) \wedge (d_F \vee \overline{d_H}), & \text{if } F = G \vee H \\ (\overline{d_F} \vee \overline{d_G}) \wedge (d_F \vee d_G), & \text{if } F = \neg G \end{cases} \quad \text{(Definitions)}$$

$\mathcal{T}(F)$ introduces a new variable d_S for each subformula S of F and **is satisfiable iff F is satisfiable**.

Tseitin Encoding

Example (Tseitin Encoding)

$$F = \underbrace{\overbrace{(x \wedge \neg y)}^{a, S_a} \vee \underbrace{(z \vee \overbrace{(x \wedge \neg w)}^{c, S_c})}_{b, S_b}}_f$$

(Encoding / Auxiliary Variables)

$$\stackrel{\text{SAT}}{=} \mathcal{T}_{\text{def}}(S_c) \wedge \mathcal{T}_{\text{def}}(S_b) \wedge \mathcal{T}_{\text{def}}(S_a) \wedge \mathcal{T}_{\text{def}}(F) \wedge f$$

$$\stackrel{\text{SAT}}{=} \dots \wedge \underbrace{(f \vee \bar{a}) \wedge (f \vee \bar{b}) \wedge (\bar{f} \vee a \vee b)}_{\mathcal{T}_{\text{def}}(F)} \wedge f$$

(Tseitin Encoding)

$$\stackrel{\text{SAT}}{=} (c \leftrightarrow x \wedge \neg w) \wedge \dots \wedge (f \leftrightarrow a \vee b) \wedge f$$

Simplification: treat negative literals like variables in $\mathcal{T}(F)$

Tseitin Encoding: Plaisted-Greenbaum Optimization

Example (Plaisted-Greenbaum Optimization)

$$\begin{aligned}
 \mathcal{T}(F) &= f \wedge (f \leftrightarrow a \vee b) \wedge (a \leftrightarrow x \wedge \neg y) \wedge (b \leftrightarrow z \vee c) \wedge (c \leftrightarrow x \wedge \neg w) \\
 &= f \wedge (\bar{f} \vee a \vee b) \wedge (f \vee \bar{a}) \wedge (f \vee \bar{b}) \\
 &\quad \wedge (\bar{a} \vee x) \wedge (\bar{a} \vee \bar{y}) \wedge (a \vee \bar{x} \vee y) \\
 &\quad \wedge (\bar{b} \vee z \vee c) \wedge (b \vee \bar{z}) \wedge (b \vee \bar{c}) \\
 &\quad \wedge (\bar{c} \vee x) \wedge (\bar{c} \vee \bar{w}) \wedge (c \vee \bar{x} \vee w)
 \end{aligned}$$

Relaxed Transformation: Exploit *Don't Cares* in monotonic functions

Model Duplication: Unconstrained encoding variables introduce additional models

Semantic Coupling: $\mathcal{T}(F) \models \mathcal{T}^{PG}(F) \models F$

Tseitin Encoding: Plaisted-Greenbaum Optimization

Example (Plaisted-Greenbaum Optimization)

$$\begin{aligned}
 \mathcal{T}^{PG}(F) &= f \wedge (f \rightarrow a \vee b) \wedge (a \rightarrow x \wedge \neg y) \wedge (b \rightarrow z \vee c) \wedge (c \rightarrow x \wedge \neg w) \\
 &= \cancel{f} \wedge (\bar{\cancel{f}} \vee a \vee b) \wedge (\cancel{f \vee \bar{a}}) \wedge (\cancel{f \vee \bar{b}}) \\
 &\quad \wedge (\bar{a} \vee x) \wedge (\bar{a} \vee \bar{y}) \wedge (\cancel{a \vee \bar{x} \vee y}) \\
 &\quad \wedge (\bar{b} \vee z \vee c) \wedge (\cancel{b \vee \bar{z}}) \wedge (\cancel{b \vee \bar{c}}) \\
 &\quad \wedge (\bar{c} \vee x) \wedge (\bar{c} \vee \bar{w}) \wedge (\cancel{c \vee \bar{x} \vee w}) \\
 &\stackrel{\text{SAT}}{=} (a \vee b) \wedge (\bar{a} \vee x) \wedge (\bar{a} \vee \bar{y}) \wedge (\bar{b} \vee z \vee c) \wedge (\bar{c} \vee x) \wedge (\bar{c} \vee \bar{w})
 \end{aligned}$$

Relaxed Transformation: Exploit *Don't Cares* in monotonic functions

Model Duplication: Unconstrained encoding variables introduce additional models

Semantic Coupling: $\mathcal{T}(F) \models \mathcal{T}^{PG}(F) \models F$

Tseitin Encoding: Plaisted-Greenbaum Optimization

Definition of Plaisted Greenbaum Encoding

$$\mathcal{T}(F) = d_F \wedge \mathcal{T}^1(F)$$

$$\mathcal{T}^p(F) = \begin{cases} \mathcal{T}_{\text{def}}^p(F) \wedge \mathcal{T}^p(G) \wedge \mathcal{T}^p(H), & \text{if } F = G \circ H \text{ and } \circ \in \{\wedge, \vee\} \\ \mathcal{T}_{\text{def}}^p(F) \wedge \mathcal{T}^{p \oplus 1}(G), & \text{if } F = \neg G \\ \text{True}, & \text{if } F \in \mathcal{V} \end{cases}$$

$$\mathcal{T}_{\text{def}}^1(F) = \begin{cases} (\overline{d_F} \vee d_G) \wedge (\overline{d_F} \vee d_H), & \text{if } F = G \wedge H \\ (\overline{d_F} \vee d_G \vee d_H), & \text{if } F = G \vee H \\ (\overline{d_F} \vee \overline{d_G}), & \text{if } F = \neg G \end{cases}$$

$$\mathcal{T}_{\text{def}}^0(F) = \begin{cases} (d_F \vee \overline{d_G} \vee \overline{d_H}), & \text{if } F = G \wedge H \\ (d_F \vee \overline{d_G}) \wedge (d_F \vee \overline{d_H}), & \text{if } F = G \vee H \\ (d_F \vee d_G), & \text{if } F = \neg G \end{cases}$$

Recap

Elementary Encodings

- Tseitin Transformation
 - Tseitin encoding allows to carry over structure to CNF
 - Formula size linear in the number of subformulas (of bounded arity)
- Cardinality Constraints
- Finite Domain Encodings

Recap

Elementary Encodings

- Tseitin Transformation
 - Tseitin encoding allows to carry over structure to CNF
 - Formula size linear in the number of subformulas (of bounded arity)
- Cardinality Constraints
- Finite Domain Encodings

Next Up

Cardinality Constraints

At-Most-One Constraints

Notation: $\text{AtMostOne}(x_1, \dots, x_n)$ **or** $\leq 1 (x_1, \dots, x_n)$ **or** $\sum_i^n x_i \leq 1$

Not more than one literal from x_1, \dots, x_n is set to True.

Naive / Pairwise Encoding

$$\mathcal{E}[\leq 1 (x_1, \dots, x_n)] = \{ \{ \overline{x_i}, \overline{x_j} \} \mid 1 \leq i < j \leq n \}$$

$$\text{Size: } \binom{n}{2} = \frac{n \cdot (n-1)}{2} \text{ clauses}$$

Size Complexity

Encoding	Clauses	Enc. Variables	Consistency
Pairwise Encoding	$\mathcal{O}(n^2)$	0	direct
Tree Encoding	$\mathcal{O}(n \log n)$	$\log n$	propagate
Ladder Encoding	$\mathcal{O}(n)$	n	propagate

Cardinality Constraints

Notation: $\leq k (x_1, \dots, x_n)$ or $\sum_i^n x_i \leq k$

Not more than k literals from x_1, \dots, x_n are set to True.

Naive Encoding

$$\mathcal{E}[\leq k (x_1, \dots, x_n)] = \{ \{ \overline{x_{i_1}}, \dots, \overline{x_{i_{k+1}}} \} \mid 1 \leq i_1 < \dots < i_{k+1} \leq n \}$$

Size: $\binom{n}{k+1}$ clauses^a

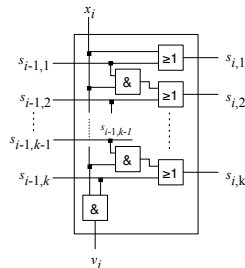
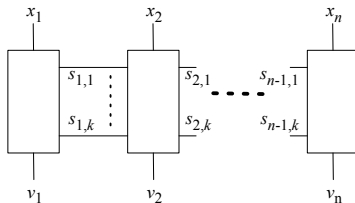
^a $\approx 2^n / \sqrt{n}$ by Stirling's Approx. for the worst case $k = \lceil n/2 \rceil$

Size Complexity

Encoding	Clauses	Enc. Variables	Consistency
Naive Encoding	$\binom{n}{k+1}$	0	direct
Sequential Counter Encoding	$\mathcal{O}(n \cdot k)$	$\mathcal{O}(n \cdot k)$	propagate
Parallel Counter Encoding	$\mathcal{O}(n)$	$\mathcal{O}(n)$	search

Cardinality Constraints: Sequential Counter Encoding

Idea: encode count-and-compare hardware circuit (cf. [Sinz, 2005](#))



$$\left. \begin{array}{l}
 (\neg x_1 \vee s_{1,1}) \\
 (\neg s_{1,j}) \quad \text{for } 1 < j \leq k \\
 (\neg x_i \vee s_{i,1}) \\
 (\neg s_{i-1,1} \vee s_{i,1}) \\
 (\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \\
 (\neg s_{i-1,j} \vee s_{i,j}) \\
 (\neg x_i \vee \neg s_{i-1,k}) \\
 (\neg x_n \vee \neg s_{n-1,k})
 \end{array} \right\} \text{ for } 1 < j \leq k \quad \left. \vphantom{\begin{array}{l} (\neg x_1 \vee s_{1,1}) \\ (\neg s_{1,j}) \\ (\neg x_i \vee s_{i,1}) \\ (\neg s_{i-1,1} \vee s_{i,1}) \\ (\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \\ (\neg s_{i-1,j} \vee s_{i,j}) \\ (\neg x_i \vee \neg s_{i-1,k}) \\ (\neg x_n \vee \neg s_{n-1,k}) \end{array}} \right\} \text{ for } 1 < i < n$$

Recap

Elementary Encodings

- Tseitin Transformation
 - Tseitin encoding allows to carry over structure to CNF
 - Formula size linear in the number of subformulas (of bounded arity)
- Cardinality Constraints
 - Size of encoding vs. Complexity of consistency
 - Choice of encoding matters
- Finite Domain Encodings

Recap

Elementary Encodings

- Tseitin Transformation
 - Tseitin encoding allows to carry over structure to CNF
 - Formula size linear in the number of subformulas (of bounded arity)
- Cardinality Constraints
 - Size of encoding vs. Complexity of consistency
 - Choice of encoding matters
- Finite Domain Encodings

Next Up

Finite Domain Encodings

Finite-Domain Variables

Common in combinatorial problems: finite domain variables, e.g.: $x \in \{v_1, \dots, v_n\}$

Relationships between them expressed as equality-formulas, e.g.: $x = v_3 \Rightarrow y \neq v_2$.

Direct encoding / “one-hot-encoding”

- Boolean variables x_v : “x takes value v”
- Must encode that each variable takes exactly one value from its domain (using at-least-one/at-most-one constraints)
- Encoding of variables’ constraints simple

Finite-Domain Variables

Common in combinatorial problems: finite domain variables, e.g.: $x \in \{v_1, \dots, v_n\}$

Relationships between them expressed as equality-formulas, e.g.: $x = v_3 \Rightarrow y \neq v_2$.

Log-encoding / binary encoding

- Boolean variables b_i^x for $0 \leq i < \lceil \log_2 n \rceil$
- Each value gets assigned a binary number, e.g. $v_1 \rightarrow 00, v_2 \rightarrow 01, v_3 \rightarrow 10$
- Inadmissible values must be excluded, e.g.:
 $x \in \{v_1, v_2, v_3\}$ requires $(\overline{b_0^x} \vee \overline{b_1^x})$
- Encoding of constraints can become complicated

Recap

Elementary Encodings

- Tseitin Transformation
 - Tseitin encoding allows to carry over structure to CNF
 - Formula size linear in the number of subformulas (of bounded arity)
- Cardinality Constraints
 - Size of encoding vs. Complexity of consistency
 - Choice of encoding matters
- Finite Domain Encodings
 - One-hot encoding vs. Log encoding
 - One-hot often simpler w.r.t. interaction between encodings