BOX Game: Rules

1. The game board consists of boxes with symbols. For example:

```
♥ ♦♥
□ •
```

2. We chose two colors, for example purple and green.

3. One color is the winning color, for example purple.

4. Then the non-winning color is green.

5. Now we can play: assign the two colors to each symbol such that its underlined and non-underlined occurrences have a different color.

6. If each box contains a symbol in purple you won.
BOX Game: Rules

1. The game board consists of boxes with symbols. For example:

   - ♠  ♦
   - ♠  ♦
   - ♠  □
   - □  □

2. We chose two colors, for example purple and green.
   - One color is the winning color, for example purple.
   - Then the non-winning color is green.
BOX Game: Rules

1. The game board consists of boxes with symbols. For example:

   ![Box Game Example](image)

2. We chose two colors, for example purple and green.
   - One color is the winning color, for example purple.
   - Then the non-winning color is green.

3. Now we can play: assign the two colors to each symbol such that its underlined and non-underlined occurrences have a different color.
BOX Game: Rules

1. The game board consists of boxes with symbols. For example:

```
♥ ♦ 
□
```

2. We chose two colors, for example purple and green.
   - One color is the winning color, for example purple.
   - Then the non-winning color is green.

3. Now we can play: assign the two colors to each symbol such that its underlined and non-underlined occurrences have a different color.

4. If each box contains a symbol in purple you won.
Some Examples

- Wrong coloring!

```
[Heart] [Diamond]  [Heart] [Diamond]  [Heart] [Square]  [Diamond] [Square]
```
Some Examples

- Wrong coloring!

  

- Again a wrong coloring!
Some Examples

- Wrong coloring!

- Again a wrong coloring!

- Lost!
Some Examples

- Wrong coloring!
  ![Examples](image1)

- Again a wrong coloring!
  ![Examples](image2)

- Lost!
  ![Examples](image3)

- Won!
  ![Examples](image4)
Some Terminology

- From now on, we call a box a clause.

- We call a clause with at least one purple symbol satisfied.

- We call a clause with all symbols in green falsified.

- We call a clause with green and uncolored symbols undecided.

⇒ The game is won if all clauses are satisfied.
How Many Possibilities?

- 1 symbol, 2 possibilities
  1. 💖
  2. 💚
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
  1.  ❤️, ♡
  2.  ❤️, ♡
  3.  ❤️, ♡
  4.  ❤️, ♡
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1,048,576 possibilities
- 30 symbols, 1,073,741,824 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
  1. ♠, ♦, ♣
  2. ♠, ♦, ♣
  3. ♠, ♦, ♣
  4. ♠, ♦, ♣
  5. ♠, ♦, ♣
  6. ♠, ♦, ♣
  7. ♠, ♦, ♣
  8. ♠, ♦, ♣
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1,048,576 possibilities
- 30 symbols, 1,073,741,824 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1,048,576 possibilities
- 30 symbols, 1,073,741,824 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...

...
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1,048,576 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1,048,576 possibilities
- 30 symbols, 1,073,741,824 possibilities
How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 possibilities
- 5 symbols, 32 possibilities
- 6 symbols, 64 possibilities
- ...
- 20 symbols, 1.048.576 possibilities
- 30 symbols, 1.073.741.824 possibilities

\[ n \text{ symbols} \quad \Rightarrow \quad 2^n \text{ possibilities} \]
Guess & Check Problems

observation in our BOX game:

- finding a solution is hard
  - $2^n$ solution candidates have to be considered
  - a good oracle is needed for guessing

- verifying a given candidate solution is easy
  - check that each box contains a purple symbol
Guess & Check Problems

observation in our BOX game:

■ finding a solution is hard
  □ $2^n$ solution candidates have to be considered
  □ a good oracle is needed for guessing

■ verifying a given candidate solution is easy
  □ check that each box contains a purple symbol

fundamental question in computer science:

The P = NP Question

Is searching for a solution harder than verifying a solution?
(unfortunately, the answer is not known)
Famous Guess & Check Problem: SAT

SAT is the decision problem of propositional logic:

- Given a Boolean formula, for example

\[ (\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z). \]

- Question: is the formula satisfiable?
  i.e., is there an assignment of truth values 1 (true), 0 (false) to the variables \( x, y, z \) such that

  - \( \forall v \in \{ x, y, z \}^1 \): the truth value of \( v \) and \( \neg v \) is different
  - each clause (\( \ldots \) ) contains at least one true literal\(^2\)

---

\(^1\)forall variables it holds that ...

\(^2\)literals of the formula: \( x, \neg x, y, \neg y, z, \neg z \)
Famous Guess & Check Problem: SAT

SAT is the decision problem of propositional logic:

- Given a Boolean formula, for example

\[ (\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z). \]

- Question: is the formula satisfiable?
  I.e., is there an assignment of truth values \( 1 \) (true), \( 0 \) (false) to the variables \( x, y, z \) such that
  \[ \forall v \in \{x, y, z\}^1 : \text{the truth value of } v \text{ and } \neg v \text{ is different} \]
  \[ \forall \text{ each clause } (...) \text{ contains at least one true literal}^2 \]

**Cook-Levin Theorem [71]**: SAT is NP-complete

Searching is as easy as checking if and only if it is for SAT.

---

1 forall variables it holds that ...
2 literals of the formula: \( x, \neg x, y, \neg y, z, \neg z \)
Relating BOX and SAT

There is a correspondence between BOX and SAT, for example between

\[
\begin{array}{cccc}
\heartsuit & \spadesuit & \heartsuit & \spadesuit \\
\heartsuit & \spadesuit & \heartsuit & \spadesuit \\
\heartsuit & \spadesuit & \spadesuit & \spadesuit \\
\spadesuit & \spadesuit & \spadesuit & \spadesuit \\
\end{array}
\]

and

\[
(\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z) : 
\]
Relating BOX and SAT

There is a correspondence between BOX and SAT, for example between

\[
\begin{array}{cccc}
\heartsuit & \blacklozenge & \heartsuit & \lozenge \\
\heartsuit & \blacklozenge & \heartsuit & \square \\
\heartsuit & \lozenge & \heartsuit & \square \\
\lozenge & \square & \lozenge & \square
\end{array}
\]

and

\[
(\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z)
\]

- \( x \equiv \heartsuit \) and \( \neg x \equiv \heartsuit \)
- \( y \equiv \blacklozenge \) and \( \neg y \equiv \blacklozenge \)
- \( z \equiv \square \) and \( \neg z \equiv \square \)
- purple/green coloring \( \cong \) assignment to true/false (1/0)
Relating BOX and SAT

There is a correspondence between BOX and SAT, for example between

\[
\begin{align*}
\heartsuit & \quad \lozenge \\
\heartsuit & \quad \lozenge \\
\heartsuit & \quad \Box \\
\lozenge & \quad \Box
\end{align*}
\]

and

\[
(\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z) : \\
\begin{align*}
\heartsuit & \quad \heartsuit \\
\lozenge & \quad \lozenge \\
\Box & \quad \Box \\
\Box & \quad \Box
\end{align*}
\]

- \( x \cong \heartsuit \) and \( \neg x \cong \heartsuit \)
- \( y \cong \lozenge \) and \( \neg y \cong \lozenge \)
- \( z \cong \Box \) and \( \neg z \cong \Box \)
- purple/green coloring \( \cong \) assignment to true/false (1/0)

⇒ if we can solve SAT, we can solve BOX (and vice versa)
Practical Applications of SAT Solving

- formal verification
- bioinformatics
- train safety
- planning & scheduling
- security
- theorem proving

encode

SAT solver

decode
Logics in this Lectures

In this lecture, we consider different logic-based languages:

- **propositional logic (SAT)**
  - simple language: only atoms and connectives
  - low expressiveness, low complexity
  - very successful in industry (e.g., verification)

- **first-order logic (predicate logic)**
  - rich language: predicates, functions, terms, quantifiers
  - great power of expressiveness, high complexity
  - many applications in mathematics and verification

- **satisfiability modulo theories (SMT)**
  - customizable language: user decides
  - as much expressiveness as required
  - as much complexity as necessary
  - very popular and successful in industry
Logic-Based Languages (Logics)

- A logic consists of
  - a set of symbols (like $\lor$, $\land$, $\neg$, $\top$, $\bot$, $\forall$, $\exists$...)
  - a set of variables (like $x$, $y$, $z$, ...)
  - concise syntax: well-formedness of expressions
  - concise semantics: meaning of expressions

- Logics support reasoning for
  - derivation of “new” knowledge
  - proving the truth/falsity of a statement (satisfiability checking)

- Different logics differ in their
  - truth values: binary (true, false), multi-valued (true, false, unknown), fuzzy (between 0 and 1, e.g., $[0, 1]$ as subset of the real numbers)
  - expressiveness (what can be formulated in the logic?)
  - complexity (how expensive is reasoning?)
PROPOSITIONAL LOGIC
Propositions

A proposition is an atomic statement that is either true or false.

**Example:**
- Alice comes to the party.
- It rains.

With connectives, propositions can be combined.

**Example:**
- Alice comes to the party, Bob as well, but not Cecile.
- If it rains, the street is wet.
Propositional Logic

- two truth values (Boolean domain): true/false, verum/falsum, on/off, 1/0
- language elements
  - atomic propositions (atoms, variables)
    - no internal structure
    - either true or false
  - logic connectives: not (¬), and (∧), or (∨), ...
    - operators for construction of composite propositions
    - concise meaning
    - argument(s) and return value from Boolean domain
  - parenthesis

**example:** formula of propositional logic: \((\neg t \lor s) \land (t \lor s) \land (\neg t \lor \neg s)\)

atoms: \(t, s\), connectives: \(\neg, \lor, \land\), parenthesis for structuring the expression
Background

- historical origins: ancient Greeks
- in philosophy, mathematics, and computer science
- two very basic principles:
  - Law of Excluded Middle: a proposition is true or its negation is true
  - Law of Contradiction: no expression is both true and false at the same time
- very simple language
  - no objects, no arguments to propositions
  - no functions, no quantifiers
- solving is easy (relative to other logics)
- many applications in industry
Syntax: Structure of Propositional Formulas

we build a propositional formula using the following components:

- **literals:**
  - variables $x, y, z, \ldots$
  - negated variables $\neg x, \neg y, \neg z, \ldots$
  - truth constants: $\top$ (verum) and $\bot$ (falsum)
  - negated truth constants: $\neg \top$ and $\neg \bot$

- **clauses**: disjunction ($\lor$) of literals
  - $x \lor y$
  - $x \lor y \lor \neg z$
  - $z$
  - $\top$
A propositional formula is a **conjunction** $(\land)$ of clauses.

examples of formulas:

- $\top$
- $\bot$
- $x$
- $\neg y$
- $x \land y \land z$
- $(\neg x \lor y \lor \neg z) \land z$
- $(x \lor \neg y) \land (x \lor \neg y \lor z) \land (y \lor \neg z)$
- $((l_{11} \lor \ldots \lor l_{1m_1}) \land \ldots \land (l_{n1} \lor \ldots \lor l_{nm_n}))$

Remark: For the moment, we consider formulas of a restricted structure called CNF (e.g., we do not consider formulas like $(x \land y) \lor (\neg x \land z)$). Any propositional formula can be translated into this structure. We will relax this restriction later.
Conventions

we use the following conventions unless stated otherwise:

- $a, b, c, x, y, z$ denote variables and $l, k$ denote literals
- $\phi, \psi, \gamma$ denote arbitrary formulas
- $C, D$ denote clauses
- clauses are also written as sets
  - $(l_1 \lor \ldots \lor l_n) = \{l_1, \ldots, l_n\}$
  - to add a literal $l$ to clause $C$, we write $C \cup \{l\}$
  - to remove a literal $l$ from clause $C$, we write $C \setminus\{l\}$
- formulas in CNF are also written as sets of sets
  - $((l_{11} \lor \ldots \lor l_{1m_1}) \land \ldots \land (l_{n1} \lor \ldots \lor l_{nm_n})) = \{\{l_{11}, \ldots, l_{1m_1}\}, \ldots, \{l_{n1}, \ldots, l_{nm_n}\}\}$
  - to add a clause $C$ to CNF $\phi$, we write $\phi \cup \{C\}$
  - to remove a clause $C$ from CNF $\phi$, we write $\phi \setminus \{C\}$
Negation

- unary connective $\neg$ (operator with exactly one argument)
- negating the truth value of its argument
- alternative notation: $!\phi$, $\bar{\phi}$, $\neg \phi$, $NOT \phi$

Truth table:

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\neg \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Set view:

Example:

- If the atom "It rains." is true then the negation "It does not rain." is false.
- If the propositional variable $a$ is true then $\neg a$ is false.
- If the propositional variable $a$ is false then $\neg a$ is true.
Disjunction

- A disjunction is true iff at least one of the arguments is true.
- Alternative notation for $\phi \lor \psi$: $\phi \mid \psi$, $\phi + \psi$, $\phi OR \psi$.
- For $(\phi_1 \lor \ldots \lor \phi_n)$ we also write $\bigvee_{i=1}^n \phi_i$.

$$
\begin{array}{c|c|c}
\phi & \psi & \phi \lor \psi \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
$$

**Truth table:**

**Set view:**

**Example:**

- $(a \lor \neg a)$ is always true.
- $(\top \lor a)$ is always true. $(\bot \lor a)$ is true if $a$ is true.
Conjunction

- a conjunction is true iff both arguments are true
- alternative notation for $\phi \land \psi$: $\& \phi \psi$, $\phi \psi$, $\phi \ast \psi$, $\phi \cdot \psi$, $\phi AND \psi$
- For $(\phi_1 \land \ldots \land \phi_n)$ we also write $\land_{i=1}^{n} \phi_i$.

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \land \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

truth table:

example:

- $(a \land \neg a)$ is always false.
- $(\top \land a)$ is true if $a$ is true. $(\bot \land \phi)$ is always false.
- If $(a \lor b)$ is true and $(\neg c \lor d)$ is true then $(a \lor b) \land (\neg c \lor d)$ is true.
Properties of Connectives

- rules of precedence:
  - \( \neg \) binds stronger than \( \wedge \)
  - \( \wedge \) binds stronger than \( \vee \)

example

- \( \neg a \vee b \wedge \neg c \vee d \) is the same as \( (\neg a) \vee (b \wedge (\neg c)) \vee d \), but not \( ((\neg a) \vee b) \wedge ((\neg c) \vee d) \)

\( \Rightarrow \) put clauses into parentheses!

- associativity:
  - \( \wedge \) is associative and commutative
  - \( \vee \) is associative and commutative

example

- \( (a \wedge b) \wedge \neg c \) is the same as \( a \wedge (b \wedge \neg c) \)
- \( (a \vee b) \vee \neg c \) is the same as \( a \vee (b \vee \neg c) \)
Assignment

- A variable can be assigned one of two values from the two-valued domain \( \mathbb{B} \), where \( \mathbb{B} = \{1, 0\} \).

- The mapping \( \nu : \mathcal{P} \rightarrow \mathbb{B} \) is called assignment, where \( \mathcal{P} \) is the set of atomic propositions.

- We sometimes write an assignment \( \nu \) as set \( V \) with \( V \subseteq \mathcal{P} \cup \{\neg x \mid x \in \mathcal{P}\} \) such that
  - \( x \in V \) iff \( \nu(x) = 1 \)
  - \( \neg x \in V \) iff \( \nu(x) = 0 \)

- For \( n \) variables, there are \( 2^n \) assignments possible.

- An assignment corresponds to one line in the truth table.
Assignment: Example

\[
\begin{array}{cccc|cccc}
 x & y & z & x \lor y & \neg z & (x \lor y) \land \neg z \\
 \hline
 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 1 \\
 1 & 0 & 1 & 1 & 0 & 0 \\
 1 & 1 & 0 & 1 & 1 & 1 \\
 1 & 1 & 1 & 1 & 0 & 0 \\
\end{array}
\]

- one assignment: \( \nu(x) = 1, \nu(y) = 0, \nu(z) = 1 \)
- alternative notation: \( V = \{x, \neg y, z\} \)
- observation: A variable assignment determines the truth value of the formulas containing these variables.
Semantics of Propositional Logic

Let $\mathcal{P}$ be the set of atomic propositions (variables) and $\mathcal{L}$ be the set of all propositional formulas over $\mathcal{P}$ that are syntactically correct (i.e., all possible conjunctions of clauses over $\mathcal{P}$).

Given assignment an $\nu : \mathcal{P} \rightarrow \mathbb{B}$, the interpretation $[.]_\nu : \mathcal{L} \rightarrow \mathbb{B}$ is defined by:

- $[\top]_\nu = 1$, $[\bot]_\nu = 0$
- if $x \in \mathcal{P}$ then $[x]_\nu = \nu(x)$
- $[\neg \phi]_\nu = 1$ iff $[\phi]_\nu = 0$
- $[\phi \lor \psi]_\nu = 1$ iff $[\phi]_\nu = 1$ or $[\psi]_\nu = 1$
- $[\phi \land \psi]_\nu = 1$ iff $[\phi]_\nu = 1$ and $[\psi]_\nu = 1$
Satisfying/Falsifying Assignments

- an assignment is called
  - satisfying a formula $\phi$ iff $[\phi]_\nu = 1$
  - falsifying a formula $\phi$ iff $[\phi]_\nu = 0$
- a satisfying assignment for $\phi$ is a **model** of $\phi$
- a falsifying assignment for $\phi$ is a **counter-model** of $\phi$

**example:**

For formula $((x \lor y) \land \neg z)$,
- $\{x, y, z\}$ is a counter-model,
- $\{x, y, \neg z\}$ is a model.
SAT-Solver Limboole

- available at http://fmv.jku.at/limboole
- input:
  - variables are strings over letters, digits and – _ . [ ] $ @
  - negation symbol \( \neg \) is !
  - disjunction symbol \( \lor \) is |
  - conjunction symbol \( \land \) is &

example

\[(a \lor b \lor \neg c) \land (\neg a \lor b) \land c\]
is represented as
\[(a \mid b \mid !c) \& (\neg a \mid b) \& c\]

\(^3\)For now, we will only use subset of the language supported by Limboole.
Properties of Propositional Formulas (1/2)

- formula $\phi$ is **satisfiable** iff there exists interpretation $[.]_\nu$ with $[\phi]_\nu = 1$
  - check with limboole -s

- formula $\phi$ is **valid** iff for all interpretations $[.]_\nu$ it holds that $[\phi]_\nu = 1$
  - check with limboole

- formula $\phi$ is **refutable** iff exists interpretation $[.]_\nu$ with $[\phi]_\nu = 0$
  - check with limboole

- formula $\phi$ is **unsatisfiable** iff $[\phi]_\nu = 0$ for all interpretations $[.]_\nu$
  - check with limboole -s
Properties of Propositional Formulas (2/2)

- a valid formula is called **tautology**
- an unsatisfiable formula is called **contradiction**

**example:**

- $\top$ is valid.
- $a \lor \neg a$ is a tautology.
- $(a \lor \neg b) \land (\neg a \lor b)$ is refutable.
- $\bot$ is unsatisfiable.
- $a \land \neg a$ is a contradiction.
- $(a \lor \neg b) \land (\neg a \lor b)$ is satisfiable.
SAT: The Boolean Satisfiability Problem

Given a propositional formula $\phi$. Is there an assignment that satisfies $\phi$?

different formulation: can we find an assignment such that each clause contains at least one true literal?
Encoding the k-Coloring Problem

Given graph \((V, E)\) with vertices \(V\) and edges \(E\). Color each node with one of \(k\) colors, such that there is no edge \((v, w) \in E\), with vertices \(v\) and \(w\) colored in the same color.

encoding:

1. **propositional variables**: \(v_j\) ... node \(v \in V\) has color \(j\) \((1 \leq j \leq k)\)

2. each node has a color:

\[
\bigwedge_{v \in V} \left( \bigvee_{1 \leq j \leq k} v_j \right)
\]

3. each node has just one color: \((\neg v_i \lor \neg v_j)\) with \(v \in V, 1 \leq i < j \leq k\)

4. neighbors have different colors: \((\neg v_i \lor \neg w_i)\) with \((v, w) \in E, 1 \leq i \leq k\)
Encoding the k-Coloring Problem: Example

task: find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\} \) with SAT

possible solution:

encoding in SAT:
Encoding the k-Coloring Problem: Example

task: find 2-coloring of graph \(\{a, b, c\}, \{(a, b), (b, c)\}\) with SAT

possible solution:

encoding in SAT:

- variables: \(a_1, a_2, b_1, b_2, c_1, c_2\)
Encoding the k-Coloring Problem: Example

task: find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\} \) with SAT

possible solution:

encoding in SAT:

- variables: \( a_1, a_2, b_1, b_2, c_1, c_2 \)
- clauses:
  1. each node has a color: \( (a_1 \lor a_2), (b_1 \lor b_2), (c_1 \lor c_2) \)
  2. no node has two colors: \( (\neg a_1 \lor \neg a_2), (\neg b_1 \lor \neg b_2), (\neg c_1 \lor \neg c_2) \)
  3. connected nodes have a different color:
     \[ (\neg a_1 \lor \neg b_1), (\neg a_2 \lor \neg b_2), (\neg b_1 \lor \neg c_1), (\neg b_2 \lor \neg c_2) \]
Encoding the k-Coloring Problem: Example

**task**: find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\} \) with SAT

**possible solution**:

```
a b c
```

**encoding in SAT**:

- **variables**: \( a_1, a_2, b_1, b_2, c_1, c_2 \)
- **clauses**:
  1. each node has a color: \((a_1 \lor a_2), (b_1 \lor b_2), (c_1 \lor c_2)\)
  2. no node has two colors: \((\neg a_1 \lor \neg a_2), (\neg b_1 \lor \neg b_2), (\neg c_1 \lor \neg c_2)\)
  3. connected nodes have a different color:
      \((\neg a_1 \lor \neg b_1), (\neg a_2 \lor \neg b_2), (\neg b_1 \lor \neg c_1), (\neg b_2 \lor \neg c_2)\)
- **full formula**:
  \((a_1 \lor a_2) \land (b_1 \lor b_2) \land (c_1 \lor c_2) \land (\neg a_1 \lor \neg a_2) \land (\neg b_1 \lor \neg b_2) \land (\neg c_1 \lor \neg c_2) \land (\neg a_1 \lor \neg b_1) \land (\neg a_2 \lor \neg b_2) \land (\neg b_1 \lor \neg c_1) \land (\neg b_2 \lor \neg c_2)\)