PROPOSITIONAL LOGIC

VL Logik: WS 2017/18

(Version 2017.1)

Martina Seidl (martina.seidl@jku.at),
Armin Biere (biere@jku.at)
Institut für Formale Modelle und Verifikation

JOHANNES KEPLER UNIVERSITY LINZ

BOX Game: Rules

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

$\underline{\heartsuit}\quad\underline{\diamondsuit}$	$\heartsuit\quad\underline{\diamondsuit}$	$\underline{\heartsuit}\quad\square$	$\diamondsuit\quad\underline{\square}$

BOX Game: Rules

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

$\underline{\heartsuit}$ $\underline{\diamondsuit}$	$\heartsuit$ $\underline{\diamondsuit}$	$\underline{\heartsuit}$ $\square$	$\diamondsuit$ $\underline{\square}$

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:

$\underline{\heartsuit}$ $\underline{\diamondsuit}$	$\heartsuit$ $\underline{\diamondsuit}$	$\underline{\heartsuit}$ $\square$	$\diamondsuit$ $\underline{\square}$

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

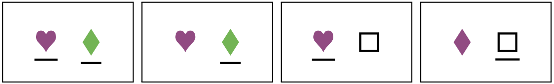
1. The game board consists of boxes with symbols. For example:

$\underline{\heartsuit}$ $\underline{\diamondsuit}$	$\heartsuit$ $\underline{\diamondsuit}$	$\underline{\heartsuit}$ $\square$	$\diamondsuit$ $\underline{\square}$

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.

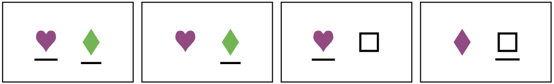
Some Examples

- Wrong coloring!

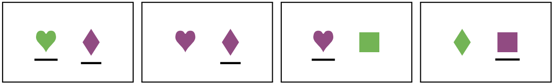


Some Examples

- Wrong coloring!

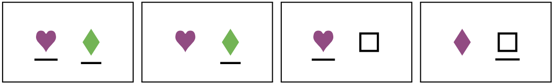


- Again a wrong coloring!

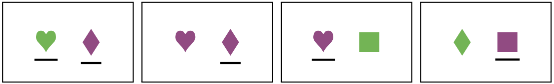


Some Examples

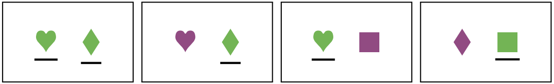
- Wrong coloring!



- Again a wrong coloring!

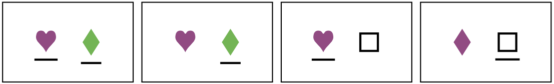


- Lost!

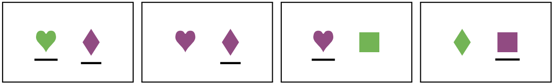


Some Examples

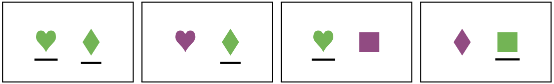
- Wrong coloring!



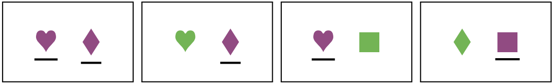
- Again a wrong coloring!



- Lost!



- Won!



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**.

$\Rightarrow$ 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. ♥, ♦

How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
    1. ${\color{purple}\heartsuit}, {\color{purple}\blacklozenge}, {\color{purple}\blacksquare}$
    2. ${\color{purple}\heartsuit}, {\color{green}\blacklozenge}, {\color{purple}\blacksquare}$
    3. ${\color{green}\heartsuit}, {\color{purple}\blacklozenge}, {\color{purple}\blacksquare}$
    4. ${\color{green}\heartsuit}, {\color{green}\blacklozenge}, {\color{purple}\blacksquare}$
    5. ${\color{purple}\heartsuit}, {\color{purple}\blacklozenge}, {\color{green}\blacksquare}$
    6. ${\color{purple}\heartsuit}, {\color{green}\blacklozenge}, {\color{green}\blacksquare}$
    7. ${\color{green}\heartsuit}, {\color{purple}\blacklozenge}, {\color{green}\blacksquare}$
    8. ${\color{green}\heartsuit}, {\color{green}\blacklozenge}, {\color{green}\blacksquare}$

How Many Possibilities?

- 1 symbol, 2 possibilities
- 2 symbols, 4 possibilities
- 3 symbols, 8 possibilities
- 4 symbols, 16 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 symbols
$\Rightarrow$
2^n 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

$$(\neg x \vee \neg y) \wedge (x \vee \neg y) \wedge (\neg x \vee z) \wedge (y \vee \neg z).$$

- 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 (...) 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

$$(\neg x \vee \neg y) \wedge (x \vee \neg y) \wedge (\neg x \vee z) \wedge (y \vee \neg z).$$

- 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 (...) 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

$$\boxed{\underline{\heartsuit}\quad\underline{\diamondsuit}}\ \boxed{\heartsuit\quad\underline{\diamondsuit}}\ \boxed{\underline{\heartsuit}\quad\square}\ \boxed{\diamondsuit\quad\underline{\square}}$$

and

$$(\neg x \vee \neg y) \wedge (x \vee \neg y) \wedge (\neg x \vee z) \wedge (y \vee \neg z):$$

Relating BOX and SAT

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

$$\boxed{\underline{\heartsuit}\quad\underline{\diamondsuit}}\ \boxed{\heartsuit\quad\underline{\diamondsuit}}\ \boxed{\underline{\heartsuit}\quad\square}\ \boxed{\diamondsuit\quad\underline{\square}}$$

and

$(\neg x \lor \neg y) \land (x \lor \neg y) \land (\neg x \lor z) \land (y \lor \neg z)$:

- $x \cong \heartsuit$ and $\neg x \cong \underline{\heartsuit}$
- $y \cong \diamondsuit$ and $\neg y \cong \underline{\diamondsuit}$
- $z \cong \square$ and $\neg z \cong \underline{\square}$
- 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

$\underline{\heartsuit}\quad\underline{\diamondsuit}$	$\heartsuit\quad\underline{\diamondsuit}$	$\underline{\heartsuit}\quad\square$	$\diamondsuit\quad\underline{\square}$

and

$(\neg x \vee \neg y) \wedge (x \vee \neg y) \wedge (\neg x \vee z) \wedge (y \vee \neg z)$:

- $x \cong \heartsuit$ and $\neg x \cong \underline{\heartsuit}$
- $y \cong \diamondsuit$ and $\neg y \cong \underline{\diamondsuit}$
- $z \cong \square$ and $\neg z \cong \underline{\square}$
- purple/green coloring $\cong$ assignment to true/false (**1**/**0**)

$\Rightarrow$ if we can solve SAT, we can solve BOX (and vice versa)

Practical Applications of SAT Solving

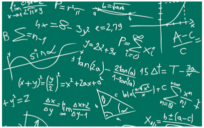
formal verification

bioinformatics

train safety

planning & scheduling

security



theorem proving

encode → $\boxed{\text{SAT solver}}$ → decode

Logics in this Lectures

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

- <u>propositional logic (SAT)</u>
  - simple language: only atoms and connectives
  - low expressiveness, low complexity
  - very successful in industry (e.g., verification)
- <u>first-order logic (predicate logic)</u>
  - rich language: predicates, functions, terms, quantifiers
  - great power of expressiveness, high complexity
  - many applications in mathematics and verification
- <u>satisfiability modulo theories (SMT)</u>
  - 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 $\vee, \wedge, \neg, \top, \bot, \forall, \exists \ldots$)
  - a set of variables (like $x, y, z, \ldots$)
  - 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 ($\neg$), and ($\wedge$), or ($\vee$), ...
    - operators for construction of composite propositions
    - concise meaning
    - argument(s) and return value from Boolean domain
  - parenthesis

example: formula of propositional logic: $(\neg \mathbf{t} \vee \mathbf{s}) \wedge (\mathbf{t} \vee \mathbf{s}) \wedge (\neg \mathbf{t} \vee \neg \mathbf{s})$

atoms: **t**, **s**, connectives: $\neg$, $\vee$, $\wedge$, 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 $(\vee)$ of literals
  - $x \vee y$
  - $x \vee y \vee \neg z$
  - z
  - $\top$

Syntax: Structure of Propositional Formulas

A propositional formula is a conjunction ($\wedge$) of clauses.

examples of formulas:

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

Remark: For the moment, we consider formulas of a restricted structure called CNF (e.g., we do not consider formulas like $(x \wedge y) \vee (\neg x \wedge z)$). 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
- ϕ, ψ, γ denote arbitrary formulas
- C, D denote clauses
- clauses are also written as sets
  - $(l_1 \vee \ldots \vee l_n) = \{l_1, \ldots l_n\}$
  - to add a literal l to clause C, we write $C \cup \{l\}$
  - to remove a literal l from clause C, we write $C \backslash \{l\}$
- formulas in CNF are also written as sets of sets
  - $((l_{11} \vee \ldots \vee l_{1m_1}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm_n})) = \{\{l_{11}, \ldots l_{1m_1}\}, \ldots, \{l_{n1}, \ldots l_{nm_n}\}\}$
  - to add a clause C to CNF ϕ, we write $\phi \cup \{C\}$
  - to remove a clause C from CNF ϕ, we write $\phi \backslash \{C\}$

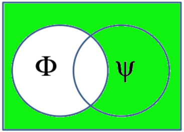
Negation

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

truth table:

ϕ	$\neg\phi$
0	**1**
1	**0**

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.

JKU

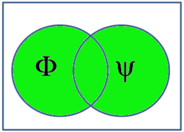
Disjunction

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

truth table:

ϕ	ψ	$\phi \vee \psi$
0	0	0
0	1	1
1	0	1
1	1	1

set view:



example:

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

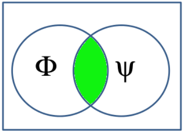
Conjunction

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

truth table:

ϕ	ψ	$\phi \wedge \psi$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

set view:



example:

- $(a \wedge \neg a)$ is always false.
- $(\top \wedge a)$ is true if a is true. $(\perp \wedge \phi)$ is always false.
- If $(a \vee b)$ is true and $(\neg c \vee d)$ is true then $(a \vee b) \wedge (\neg c \vee d)$ is true.

Properties of Connectives

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

  <u>example</u>
  - $\neg a \vee b \wedge \neg c \vee d$ 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

  <u>example</u>
  - $(a \wedge b) \wedge \neg c$ 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 $\mathbb{B} = \{\mathbf{1}, \mathbf{0}\}$
- the mapping $\nu : \mathcal{P} \to \mathbb{B}$ is called <u>assignment</u>, where $\mathcal{P}$ is the set of atomic propositions
- we sometimes write an assignment ν as set V with $V \subseteq \mathcal{P} \cup \{\neg x | x \in \mathcal{P}\}$ such that
  - $x \in V$ iff $\nu(x) = \mathbf{1}$
  - $\neg x \in V$ iff $\nu(x) = \mathbf{0}$
- for n variables, there are 2^n assignments possible
- an assignment corresponds to one line in the truth table

Assignment: Example

x	y	z	$x \vee y$	$\neg z$	$(x \vee y) \wedge \neg z$
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**

- one assignment: $\nu(x) = \mathbf{1}, \nu(y) = \mathbf{0}, \nu(z) = \mathbf{1}$
- 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:

- $[\top]_\nu = \mathbf{1}$, $[\bot]_\nu = \mathbf{0}$
- if $x \in \mathcal{P}$ then $[x]_\nu = \nu(x)$
- $[\neg\phi]_\nu = \mathbf{1}$ iff $[\phi]_\nu = \mathbf{0}$
- $[\phi \vee \psi]_\nu = \mathbf{1}$ iff $[\phi]_\nu = \mathbf{1}$ or $[\psi]_\nu = \mathbf{1}$
- $[\phi \wedge \psi]_\nu = \mathbf{1}$ iff $[\phi]_\nu = \mathbf{1}$ and $[\psi]_\nu = \mathbf{1}$

Satisfying/Falsifying Assignments

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

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:[3]
  - variables are strings over letters, digits and – _ . [ ] $ @
  - negation symbol $\neg$ is `!`
  - disjunction symbol $\lor$ is `|`
  - conjunction symbol $\land$ is &

<u>example</u>

$(a \lor b \lor \neg c) \land (\neg a \lor b) \land c$ is represented as

`(a | b | !c) & (!a | b) & c`

[3] For now, we will only use subset of the language supported by Limboole.

Properties of Propositional Formulas (1/2)

- formula ϕ is <u>satisfiable</u> iff
  there exists interpretation $[.]_\nu$ with $[\phi]_\nu = \mathbf{1}$
  check with `limboole -s`
- formula ϕ is <u>valid</u> iff
  for all interpretations $[.]_\nu$ it holds that $[\phi]_\nu = \mathbf{1}$
  check with `limboole`
- formula ϕ is <u>refutable</u> iff
  exists interpretation $[.]_\nu$ with $[\phi]_\nu = \mathbf{0}$
  check with `limboole`
- formula ϕ is <u>unsatisfiable</u> iff
  $[\phi]_\nu = \mathbf{0}$ 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.
- $a \vee \neg a$ is a tautology.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ is refutable.
- $\bot$ is unsatisfiable.
- $a \wedge \neg a$ is a contradiction.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ is satisfiable.

SAT: The Boolean Satisfiability Problem

Given a propositional formula ϕ.
Is there an assignment that satisfies ϕ?

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:

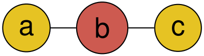
1. propositional variables: v_j ... node $v \in V$ has color j $(1 \leq j \leq k)$
2. each node has a color:
$$\bigwedge_{v \in V} (\bigvee_{1 \leq j \leq k} v_j)$$
3. each node has just one color: $(\neg v_i \vee \neg v_j)$ with $v \in V, 1 \leq i < j \leq k$
4. neighbors have different colors: $(\neg v_i \vee \neg w_i)$ with $(v, w) \in E, 1 \leq i \leq k$

JKU

Encoding the k-Coloring Problem: Example

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

possible solution:

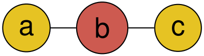


encoding in SAT:

Encoding the k-Coloring Problem: Example

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

possible solution:



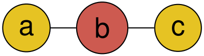
encoding in SAT:

- 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:



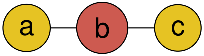
encoding in SAT:

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

Encoding the k-Coloring Problem: Example

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

possible solution:



encoding in SAT:

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