Formal Methods in Software Development

Modeling with Propositional Logic

Mădălina Erașcu

West University of Timișoara
Faculty of Mathematics and Informatics
Department of Computer Science

Based on slides of the lecture Satisfiability Checking (Erika Ábrahám), RTWH Aachen

WS 2019/2020

Before we solve this problem...

- Suppose we can solve the satisfiability problem... how can this help us?

Before we solve this problem...

- Suppose we can solve the satisfiability problem... how can this help us?

- There are numerous problems in the industry that are solved via the satisfiability problem of propositional logic
  - Logistics
  - Planning
  - Electronic Design Automation industry
  - Cryptography
  - ...

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.
  - *Case 2*: To check whether $F_1, ..., F_n \models G$, check the satisfiability of $F_1 \wedge ... \wedge F_n \wedge \neg G$. If it is unsatisfiable, then $F_1, ..., F_n \models G$, otherwise $F_1, ..., F_n \not\models G$.

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.
  - *Case 2*: To check whether $F_1, ..., F_n \models G$, check the satisfiability of $F_1 \wedge ... \wedge F_n \wedge \neg G$. If it is unsatisfiable, then $F_1, ..., F_n \models G$, otherwise $F_1, ..., F_n \not\models G$.
  - Complexity $O(2^n)$

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.
  - *Case 2*: To check whether $F_1, \ldots, F_n \models G$, check the satisfiability of $F_1 \wedge \ldots \wedge F_n \wedge \neg G$. If it is unsatisfiable, then $F_1, \ldots, F_n \models G$, otherwise $F_1, \ldots, F_n \not\models G$.
  - Complexity $O(2^n)$
  - Can we do better?

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.
  - *Case 2*: To check whether $F_1, ..., F_n \models G$, check the satisfiability of $F_1 \wedge ... \wedge F_n \wedge \neg G$. If it is unsatisfiable, then $F_1, ..., F_n \models G$, otherwise $F_1, ..., F_n \not\models G$.
  - Complexity $O(2^n)$
  - Can we do better?

SAT was the first problem shown to be NP-complete: all of the problems in the class NP can be solved by translating them (in polynomial time) into SAT.

SAT Solving

- A SAT solver solves the Boolean satisfiability problem.
- The SAT problem can be determined using truth table algorithm:
  - *Case 1*: To check whether F is satisfiable, compute the truth table for F. If there is a row in which *true* appears as the value for F, then F is satisfiable. Otherwise, F is unsatisfiable.
  - *Case 2*: To check whether $F_1, ..., F_n \models G$, check the satisfiability of $F_1 \wedge ... \wedge F_n \wedge \neg G$. If it is unsatisfiable, then $F_1, ..., F_n \models G$, otherwise $F_1, ..., F_n \not\models G$.
  - Complexity $O(2^n)$
  - Can we do better?

SAT was the first problem shown to be NP-complete: all of the problems in the class NP can be solved by translating them (in polynomial time) into SAT.

Goal: Build a fast solver for SAT, hence it could be used to solve lots of problems.

Example 0

Consider the following formula (in CNF!):

$$
\begin{aligned}
&(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge \\
&(\neg C \vee F) \wedge (\neg F \vee C) \wedge \\
&(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge \\
&(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge \\
&(G \vee H \vee \neg I) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge \\
&I
\end{aligned}
$$

Example 0

Consider the following formula (in CNF!):

$$
\begin{aligned}
&(\neg A \vee \neg B \vee E) \wedge (\neg E \vee A) \wedge (\neg E \vee B) \wedge \\
&(\neg C \vee F) \wedge (\neg F \vee C) \wedge \\
&(\neg D \vee \neg E \vee G) \wedge (\neg G \vee D) \wedge (\neg G \vee E) \wedge \\
&(\neg E \vee \neg F \vee H) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge \\
&(G \vee H \vee \neg I) \wedge (\neg H \vee E) \wedge (\neg H \vee F) \wedge \\
&I
\end{aligned}
$$

Check if it SAT or not.

Example 0 (cont'd)

Notation: DIMACS standard which is also the input language of SAT solvers.

- Each variable is represented by a positive integer.

Example 0 (cont'd)

Notation: DIMACS standard which is also the input language of SAT solvers.

- Each variable is represented by a positive integer.
- A negative integer refers to the negation of the variable.

Example 0 (cont'd)

Notation: DIMACS standard which is also the input language of SAT solvers.

- Each variable is represented by a positive integer.
- A negative integer refers to the negation of the variable.
- Clauses are given as sequences of integers separated by spaces.

Example 0 (cont'd)

Notation: DIMACS standard which is also the input language of SAT solvers.

- Each variable is represented by a positive integer.
- A negative integer refers to the negation of the variable.
- Clauses are given as sequences of integers separated by spaces.
- 0 (zero) terminates the clause.

Example 0 (cont'd)

Notation: DIMACS standard which is also the input language of SAT solvers.

- Each variable is represented by a positive integer.
- A negative integer refers to the negation of the variable.
- Clauses are given as sequences of integers separated by spaces.
- 0 (zero) terminates the clause.

For example above, we have:

$$
\begin{array}{l}
-1\ -2\ 5\ 0 \quad -5\ 1\ 0 \quad -5\ 2\ 0 \\
-3\ 6\ 0 \quad -6\ 3\ 0 \\
-4\ -5\ 7\ 0 \quad -7\ 4\ 0 \quad -7\ 5\ 0 \\
-5\ -6\ 8\ 0 \quad -8\ 5\ 0 \quad -8\ 6\ 0 \\
7\ 8\ -9\ 0 \quad -8\ 5\ 0 \quad -8\ 6\ 0 \\
9\ 0
\end{array}
$$

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.
- Nowadays, SAT solvers, which are extremely fast being able to solve formulae with more than 1000K clauses and tens of thousands of variables, are based on the:

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.
- Nowadays, SAT solvers, which are extremely fast being able to solve formulae with more than 1000K clauses and tens of thousands of variables, are based on the:
  - Conflict-Driven Clause Learning algorithm (modern variant of the DPLL algorithm – e.g. the SAT solver Chaff)

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.
- Nowadays, SAT solvers, which are extremely fast being able to solve formulae with more than 1000K clauses and tens of thousands of variables, are based on the:
  - Conflict-Driven Clause Learning algorithm (modern variant of the DPLL algorithm – e.g. the SAT solver Chaff)
  - stochastic local search algorithms (e. g. SAT solver WalkSAT).

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.
- Nowadays, SAT solvers, which are extremely fast being able to solve formulae with more than 1000K clauses and tens of thousands of variables, are based on the:
  - Conflict-Driven Clause Learning algorithm (modern variant of the DPLL algorithm – e.g. the SAT solver Chaff)
  - stochastic local search algorithms (e. g. SAT solver WalkSAT).

We will use an online SAT solver:
`https://msoos.github.io/cryptominisat_web/`

Example 0 (cont'd)

- The first SAT solvers were based on the Davis-Putnam method.
- Nowadays, SAT solvers, which are extremely fast being able to solve formulae with more than 1000K clauses and tens of thousands of variables, are based on the:
  - Conflict-Driven Clause Learning algorithm (modern variant of the DPLL algorithm – e.g. the SAT solver Chaff)
  - stochastic local search algorithms (e. g. SAT solver WalkSAT).

We will use an online SAT solver:

`https://msoos.github.io/cryptominisat_web/`

Experimenting with more advanced SAT solvers (see http://www.satcompetition.org/ for the most competitive SAT solvers) is strongly encouraged.

Example 1

Example 1

- Notation:

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

Introduce a propositional variable for each pair (person, chair):

$x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

Example 1

- Notation: Aunt $= 1$, Sister $= 2$, Father $= 3$
  Left chair $= 1$, Middle chair $= 2$, Right chair $= 3$

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

  Sister doesn't want to sit to the right of Father:

Example 1

- Notation: Aunt = 1, Sister = 2, Father = 3
  Left chair = 1, Middle chair = 2, Right chair = 3

  Introduce a propositional variable for each pair (person, chair):

  $x_{p,c}$ = "person p is sited in chair c" for $1 \leq p, c \leq 3$
- Constraints:

  Aunt doesn't want to sit near Father:

  $$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

  Aunt doesn't want to sit in the left chair:

  $$\neg x_{1,1}$$

  Sister doesn't want to sit to the right of Father:

  $$(x_{3,1} \rightarrow \neg x_{2,2}) \wedge (x_{3,2} \rightarrow \neg x_{2,3})$$

Example 1 (continued)

Example 1 (continued)

Each person is placed:

Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3}) \quad \Leftrightarrow$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3}) \quad \Leftrightarrow$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

No person is placed in more than one chair:

Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3}) \quad \Leftrightarrow$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

No person is placed in more than one chair:

$$\bigwedge_{p=1}^{3} \bigwedge_{c1=1}^{3} \bigwedge_{c2=c1+1}^{3} (\neg x_{p,c1} \vee \neg x_{p,c2})$$

Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3}) \quad \Leftrightarrow$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

No person is placed in more than one chair:

$$\bigwedge_{p=1}^{3} \bigwedge_{c1=1}^{3} \bigwedge_{c2=c1+1}^{3} (\neg x_{p,c1} \vee \neg x_{p,c2})$$

At most one person per chair:

Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3}) \quad \Leftrightarrow$$

$$\bigwedge_{p=1}^{3} \bigvee_{c=1}^{3} x_{p,c}$$

No person is placed in more than one chair:

$$\bigwedge_{p=1}^{3} \bigwedge_{c1=1}^{3} \bigwedge_{c2=c1+1}^{3} (\neg x_{p,c1} \vee \neg x_{p,c2})$$

At most one person per chair:

$$\bigwedge_{p1=1}^{3} \bigwedge_{p2=p1+1}^{3} \bigwedge_{c=1}^{3} (\neg x_{p1,c} \vee \neg x_{p2,c})$$

Example 2: Assignment of frequencies

- n radio stations
- For each station assign one of k transmission frequencies, $k < n$.
- E – set of pairs of stations, that are too close to have the same frequency.

Example 2: Assignment of frequencies

- n radio stations
- For each station assign one of k transmission frequencies, $k < n$.
- E – set of pairs of stations, that are too close to have the same frequency.

- Q: Can we assign to each station a frequency, such that no station pairs from E have the same frequency?

Example 2 (continued)

- Notation:

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} (\neg x_{s,f1} \vee \neg x_{s,f2})$$

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} (\neg x_{s,f1} \vee \neg x_{s,f2})$$

  Close stations are not assigned the same frequency:

Example 2 (continued)

- Notation:
  $x_{s,f}$ = "station s is assigned frequency f" for $1 \leq s \leq n$, $1 \leq f \leq k$
- Constraints:
  Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^{n} \left(\bigvee_{f=1}^{k} x_{s,f} \right)$$

  Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^{n} \bigwedge_{f1=1}^{k-1} \bigwedge_{f2=f1+1}^{k} (\neg x_{s,f1} \vee \neg x_{s,f2})$$

  Close stations are not assigned the same frequency:
  For each $(s1, s2) \in E$,

$$\bigwedge_{f=1}^{k} (\neg x_{s1,f} \vee \neg x_{s2,f})$$

Example 3: Seminar topic assignment

- n participants
- n topics
- Set of preferences $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$
  $(p, t) \in E$ means: participant p would take topic t

Example 3: Seminar topic assignment

- n participants
- n topics
- Set of preferences $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$
  $(p, t) \in E$ means: participant p would take topic t
- Q: Can we assign to each participant a topic which he/she is willing to take?

Example 3 (continued)

- Notation:

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

  Each participant is assigned at most one topic:

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

  Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t1=1}^{n-1} \bigwedge_{t2=t1+1}^{n} (\neg x_{p,t1} \vee \neg x_{p,t2})$$

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

  Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t1=1}^{n-1} \bigwedge_{t2=t1+1}^{n} (\neg x_{p,t1} \vee \neg x_{p,t2})$$

  Each participant is willing to take his/her assigned topic:

Example 3 (continued)

- Notation: $x_{p,t}$ = "participant p is assigned topic t"
- Constraints:

  Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^{n} \left(\bigvee_{t=1}^{n} x_{p,t} \right)$$

  Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{t1=1}^{n-1} \bigwedge_{t2=t1+1}^{n} (\neg x_{p,t1} \vee \neg x_{p,t2})$$

  Each participant is willing to take his/her assigned topic:

$$\bigwedge_{p=1}^{n} \bigwedge_{(p,t)\notin E} \neg x_{p,t}$$

Example 3 (continued)

Example 3 (continued)

Each topic is assigned to at most one participant:

Example 3 (continued)

Each topic is assigned to at most one participant:

$$\bigwedge_{t=1}^{n} \bigwedge_{p1=1}^{n} \bigwedge_{p2=p1+1}^{n} (\neg x_{p1,t} \vee \neg x_{p2,t})$$

Example 4

The circuit satisfiability problem (CSP) is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output true. Consider the CSP for following circuit using a SAT solver.