Combinatorial Decision Making and Optimization (Module 2)

Last update: 18 April 2024

Academic Year 2023 – 2024

Alma Mater Studiorum · University of Bologna

Contents

1 Satisfiability modulo theory

Satisfiability modulo theory (SMT) Satisfiability of a formula with respect to some background formal theory/theories.

SMT extends SAT and exploits domain-specific reasoning (possibly with infinite domains).

Satisfiability modulo theory (SMT)

1.1 First-order logic for SMT

1.1.1 Syntax

Remark. Only quantifier-free formulas (q.f.f.) are considered in SMT.

Functions The set of all the functions is denoted as $\Sigma^F = \bigcup_{k \geq 0} \Sigma_k^F$ where Σ_k^F denotes the set of k-ary functions.

Functions

Constants Σ_0^F

Predicates The set of all the predicates is denoted as $\Sigma^P = \bigcup_{k \geq 0} \Sigma_k^P$ where Σ_k^P denotes the set of k-ary predicates.

Predicates

Propositional symbols Σ_0^P

Signature The set of the non-logical symbols of FOL is denoted as:

Signature

$$\Sigma = \Sigma^F \cup \Sigma^P$$

Terms The set of terms over Σ is denoted as $\mathbb{T}^\Sigma$:

Terms

$$\begin{aligned}
\mathbb{T}^\Sigma = & \Sigma_0^F \cup \\
& \{f(t_1, \dots, t_k) \mid f \in \Sigma_k^F \land t_1, \dots, t_k \in \mathbb{T}^\Sigma\} \cup \\
& \{\texttt{ite}(\varphi, t_1, t_2) \mid \varphi \in \mathbb{F}^\Sigma \land t_1, t_2 \in \mathbb{T}^\Sigma\}
\end{aligned}$$

Remark. `ite` is an auxiliary function to capture the if-then-else construct.

Formulas The set of formulas over Σ is denoted as $\mathbb{F}^\Sigma$:

Formulas

$$\begin{aligned}
\mathbb{F}^\Sigma = & \{\bot, \top\} \cup \Sigma_0^P \cup \\
& \{t_1 = t_2 \mid t_1, t_2 \in \mathbb{T}^\Sigma\} \cup \\
& \{p(t_1, \dots, t_k) \mid p \in \Sigma_k^P \land t_1, \dots, t_k \in \mathbb{T}^\Sigma\} \cup \\
& \{\neg\varphi \mid \varphi \in \mathbb{F}^\Sigma\} \cup \\
& \{(\varphi_1 \Rightarrow \varphi_2), (\varphi_1 \iff \varphi_2), (\varphi_1 \land \varphi_2), (\varphi_1 \lor \varphi_2) \mid \varphi_1, \varphi_2 \in \mathbb{F}^\Sigma\}
\end{aligned}$$

1.1.2 Semantics

Σ-model Pair $\mathcal{M} = \langle M, (\cdot)^{\mathcal{M}} \rangle$ defined on a given signature Σ where: Σ-model

- M is the universe of $\mathcal{M}$.
- $(\cdot)^{\mathcal{M}}$ is a mapping such that:
  - $\forall f \in \Sigma_k^F : f^{\mathcal{M}} \in \{\varphi \mid \varphi : M^k \to M\}$.
  - $\forall p \in \Sigma_k^P : p^{\mathcal{M}} \in \{\varphi \mid \varphi : M^k \to \{\texttt{true}, \texttt{false}\}\}$.

Interpretation Extension of the mapping function $(\cdot)^{\mathcal{M}}$ to terms and formulas: Interpretation

- $\top^{\mathcal{M}} = \texttt{true}$ and $\bot^{\mathcal{M}} = \texttt{false}$.
- $(f(t_1, \dots, t_k))^{\mathcal{M}} = f^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_k^{\mathcal{M}})$ and $(p(t_1, \dots, t_k))^{\mathcal{M}} = p^{\mathcal{M}}(t_1^{\mathcal{M}}, \dots, t_k^{\mathcal{M}})$.
- $\texttt{ite}(\varphi, t_1, t_2)^{\mathcal{M}} = \begin{cases} t_1^{\mathcal{M}} & \text{if } \varphi^{\mathcal{M}} = \texttt{true} \\ t_2^{\mathcal{M}} & \text{if } \varphi^{\mathcal{M}} = \texttt{false} \end{cases}$.

1.1.3 Σ-theory

Satisfiability A model $\mathcal{M}$ satisfies a formula $\varphi \in \mathbb{F}^{\Sigma}$ if $\varphi^{\mathcal{M}} = \texttt{true}$. Satisfiability

Σ-theory Possibly infinite set $\mathcal{T}$ of Σ-models. Σ-theory

$\mathcal{T}$-satisfiability A formula $\varphi \in \mathbb{F}^{\Sigma}$ is $\mathcal{T}$-satisfiable if there exists a model $\mathcal{M} \in \mathcal{T}$ that satisfies it. $\mathcal{T}$-satisfiability

$\mathcal{T}$-consistency A set of formulas $\{\varphi_1, \dots, \varphi_k\} \subseteq \mathbb{F}^{\Sigma}$ is $\mathcal{T}$-consistent iff $\varphi_1 \land \dots \land \varphi_k$ is $\mathcal{T}$-satisfiable. $\mathcal{T}$-consistency

$\mathcal{T}$-entailment A set of formulas $\Gamma \subseteq \mathbb{F}^{\Sigma}$ $\mathcal{T}$-entails a formula $\varphi \in \mathbb{F}^{\Sigma}$ ($\Gamma \models_{\mathcal{T}} \varphi$) iff in every model $\mathcal{M} \in \mathcal{T}$ that satisfies Γ, φ is also satisfied. $\mathcal{T}$-entailment

Remark. Γ is $\mathcal{T}$-consistent iff $\Gamma \not\models_{\mathcal{T}} \bot$.

$\mathcal{T}$-validity A formula $\varphi \in \mathbb{F}^{\Sigma}$ is $\mathcal{T}$-valid iff $\varnothing \models_{\mathcal{T}} \varphi$. $\mathcal{T}$-validity

Remark. φ is $\mathcal{T}$-consistent iff $\neg\varphi$ is not $\mathcal{T}$-valid.

Theory lemma $\mathcal{T}$-valid clause $c = l_1 \lor \dots \lor l_k$. Theory lemma

Σ-expansion Given a Σ-model $\mathcal{M} = \langle M, (\cdot)^{\mathcal{M}} \rangle$ and $\Sigma' \supseteq \Sigma$, an expansion $\mathcal{M}' = \langle M', (\cdot)^{\mathcal{M}'} \rangle$ over Σ' is any Σ'-model such that: Σ-expansion

- $M' = M$.
- $\forall s \in \Sigma : s^{\mathcal{M}'} = s^{\mathcal{M}}$

Remark. Given a Σ-theory $\mathcal{T}$, we implicitly consider it to be the theory $\mathcal{T}'$ defined as:

$$\mathcal{T}' = \{\mathcal{M}' \mid \mathcal{M}' \text{ is an expansion of a } \Sigma\text{-model } \mathcal{M} \text{ in } \mathcal{T}\}$$

Ground $\mathcal{T}$-satisfiability Given a Σ-theory $\mathcal{T}$, determine if a ground formula is $\mathcal{T}$-satisfiable over a Σ-expansion $\mathcal{T}'$. Ground $\mathcal{T}$-satisfiability

Axiomatically defined theory Given a minimal set of formulas (axioms) $\Lambda \subseteq \mathbb{F}^{\Sigma}$, its corresponding theory is the set of all the models that respect Λ. Axiomatically defined theory

Example. Let Σ be defined as:

$$\Sigma_0^F = \{a, b, c, d\} \qquad \Sigma_1^F = \{f, g\} \qquad \Sigma_2^P = \{p\}$$

A Σ-model $\mathcal{M} = \langle [0, 2\pi[, (\cdot)^\mathcal{M} \rangle$ can be defined as follows:

$$a^\mathcal{M} = 0 \qquad b^\mathcal{M} = \frac{\pi}{2} \qquad c^\mathcal{M} = \pi \qquad d^\mathcal{M} = \frac{3\pi}{2}$$
$$f^\mathcal{M} = \sin \qquad g^\mathcal{M} = \cos \qquad p^\mathcal{M}(x, y) \iff x > y$$

To determine if $p(g(x), f(d))$ is $\mathcal{M}$-satisfiable, we have to expand $\mathcal{M}$ as there are free variables (x). Let $\Sigma' = \Sigma \cup \{x\}$. The expansion $\mathcal{M}'$ such that $x^{\mathcal{M}'} = \frac{\pi}{2}$ makes the formula satisfiable.

1.1.4 Theories of interest

Equality with Uninterpreted Functions theory (EUF) Theory $\mathcal{T}_\text{EUF}$ containing all the possible Σ-models.

Equality with Uninterpreted Functions theory (EUF)

Remark. Also called empty theory as its axiom set is $\varnothing$ (i.e. allows any model).

Remark. Useful to deal with black-box functions (i.e. prove satisfiability without a specific theory).

Example. The following formula can be proved to be unsatisfiable by only using syntactic manipulations of basic FOL concepts:

$$\big(a * (f(b) + f(c)) = d\big) \land \big(b * (f(a) + f(c)) \neq d\big) \land \underline{(a = b)}$$
$$\big(\underline{a * (f(a) + f(c))} = d\big) \land \big(\underline{a * (f(a) + f(c))} \neq d\big)$$
$$\big(g(a, c) = d\big) \land \big(g(a, c) \neq d\big)$$

Arithmetic theories Theories with $\Sigma = (0, 1, +, -, \leq)$.

Arithmetic theories

Presburger arithmetic Theory $\mathcal{T}_\mathbb{Z}$ that interprets Σ-symbols over integers.

- Ground $\mathcal{T}_\mathbb{Z}$-satisfiability is **NP**-complete.
- Extended with multiplication, $\mathcal{T}_\mathbb{Z}$-satisfiability becomes undecidable.

Real arithmetic Theory $\mathcal{T}_\mathbb{R}$ that interprets Σ-symbols over reals.

- Ground $\mathcal{T}_\mathbb{R}$-satisfiability is in **P**.
- Extended with multiplication, $\mathcal{T}_\mathbb{R}$-satisfiability becomes doubly-exponential.

Remark. In floating points, commutativity still holds, but associativity and distributivity are not guaranteed.

Array theory Let $\Sigma_\mathcal{A}$ be the signature containing two functions:

Array theory

$\texttt{read}(a, i)$ Reads the value of a at index i.

$\texttt{write}(a, i, v)$ Returns an array a' where the value v is at the index i of a.

The theory $\mathcal{T}_\mathcal{A}$ is the set of all models respecting the following axioms:

- $\forall a \, \forall i \, \forall v : \texttt{read}(\texttt{write}(a, i, v), i) = v$.
- $\forall a \, \forall i \, \forall j \, \forall v : (i \neq j) \Rightarrow \Big(\texttt{read}\big(\texttt{write}(a, i, v), j\big) = \texttt{read}(a, j)\Big)$.
- $\forall a \, \forall a' : \big(\forall i : \texttt{read}(a, i) = \texttt{read}(a', i)\big) \Rightarrow (a = a')$.

Remark. The full $\mathcal{T}_\mathcal{A}$ theory is undecidable but there are decidable fragments.

Bit-vectors theory Theory $\mathcal{T}_{\mathcal{BV}}$ with vectors of bits of fixed length as constants and operations such as: Bit-vectors theory

- String-like operations (e.g. slicing, concatenation, ...).
- Logical operations (e.g. bit-wise operators).
- Arithmetic operations (e.g. $+, -, \dots$).

String theory Theory to handle strings of unbounded length. String theory

Theory of word equations Given an alphabet $\mathcal{S}$, a word equation has form $L = R$ where L and R are concatenations of string constants over $\mathcal{S}^*$.

Remark. The general theory of word equations is undecidable.

Remark. The quantifier-free theory of word equations is decidable.

Remark. In practice, many theories are often combined.

1.2 Encoding to SAT

1.2.1 Eager approaches

All the information on the formal theory is used from the beginning to encode an SMT formula φ into an equisatisfiable SAT formula φ' (i.e. SMT is compiled into SAT).

Equisatisfiability Given a Σ-theory $\mathcal{T}$, two formulas φ and φ' are equisatisfiable iff: Equisatisfiability

$$\varphi \text{ is } \mathcal{T}\text{-satisfiable} \iff \varphi' \text{ is } \mathcal{T}\text{-satisfiable}$$

Eager approaches have the following advantages:

- Does not require an SMT solver.
- Once encoded, whichever SAT solver can be used.

Eager approaches have the following disadvantages:

- An ad-hoc encoding is needed for all the theories.
- The resulting SAT formula might be huge.

Algorithm Given an EUF formula φ, to determine if it is $\mathcal{T}_{\text{EUF}}$-satisfiable, the following steps are taken:

1. Replace functions and predicates with constant equalities. Given the terms $f(t_1), \dots, f(t_k)$, possible approaches are:

   Ackermann approach Ackermann approach

   - Each $f(t_i)$ is encoded into a new constant A_i.
   - Add the constraints $(t_i = t_j) \Rightarrow (A_i = A_j)$ for each $i < j$.

   Bryant approach Bryant approach

   - $f(t_1)$ is encoded as A_1.
   - $f(t_2)$ is encoded as $\texttt{ite}(t_2 = t_1, A_1, A_2)$.

- $f(t_3)$ is encoded as $\texttt{ite}\big(t_3 = t_1, A_1, \texttt{ite}(t_3 = t_2, A_2, A_3)\big)$.
- $f(t_i)$ is encoded as:

$$\texttt{ite}\Big(t_i = t_1, A_1, \texttt{ite}\Big(t_i = t_2, A_2, \texttt{ite}\big(\dots, \texttt{ite}(t_i = t_{i-1}, A_{i-1}, A_i)\big)\Big)\Big)$$

2. Remove equalities to reduce φ into propositional logic. Possible encodings are:

   Small-domain encoding If φ has n distinct variables $\{c_1, \dots, c_n\}$, a possible model $\mathcal{M} = \langle M, (\cdot)^{\mathcal{M}} \rangle$ that satisfies it must have $|M| \leq n$.

   Therefore, each $c_i^{\mathcal{M}}$ can be associated to a value in $\{1, \dots, n\}$. In SAT, this mapping from $c_i^{\mathcal{M}}$ to $\{1, \dots, n\}$ can be encoded using $O(\log n)$ bits. Finally, an equality $c_i = c_j$ (or $c_i \neq c_j$) can be encoded by adding bitwise constraints.

   Direct encoding Encode each equality $a = b$ with a propositional symbol $P_{a,b}$ and add transitivity constraints of form $(P_{a,b} \wedge P_{b,c}) \Rightarrow P_{a,c}$.

1.2.2 Lazy approaches

Integrate SAT solvers with theory-specific decision procedures.
These approaches are more flexible and modular and avoid an explosion of SAT clauses.
On the other hand, the search becomes SAT-driven and not theory-driven.

Remark. Most SMT solvers follow a lazy approach.

Algorithm Let $\mathcal{T}$ be a theory. Given a conjunction of $\mathcal{T}$-literals $\varphi = \varphi_1 \wedge \dots \wedge \varphi_n$, to determine its $\mathcal{T}$-satisfiability, a generic lazy solver does the following:

1. Each SMT literal φ_i is encoded (abstracted) into a SAT literal l_i to form the abstraction $\Phi = \{l_1, \dots, l_n\}$ of φ.
2. The $\mathcal{T}$-solver sends Φ to the SAT-solver.
   - If the SAT-solver determines that Φ is unsatisfiable, then φ is $\mathcal{T}$-unsatisfiable.
   - Otherwise, the SAT-solver returns a model $\mathcal{M} = \{a_1, \dots, a_n\}$ (an assignment of the literals, possibly partial).
3. The $\mathcal{T}$-solver determines if $\mathcal{M}$ is $\mathcal{T}$-consistent.
   - If it is, then φ is $\mathcal{T}$-satisfiable.
   - Otherwise, update $\Phi = \Phi \cup \neg\mathcal{M}$ and go to Point 2.

Example. Consider the EUF formula φ:

$$\big(g(a) = c\big) \wedge \big((f(g(a)) \neq f(c)) \vee (g(a) = d)\big) \wedge \big(c \neq d\big)$$

- φ abstracted into SAT is:

$$\underbrace{\big(g(a) = c\big)}_{l_1} \wedge \big(\neg \underbrace{(f(g(a)) = f(c))}_{l_2} \vee \underbrace{(g(a) = d)}_{l_3}\big) \wedge \neg \underbrace{(c = d)}_{l_4}$$

$$l_1 \wedge (\neg l_2 \vee l_3) \wedge \neg l_4$$

  Therefore, $\Phi = \{l_1, (\neg l_2 \vee l_3), \neg l_4\}$

- The $\mathcal{T}$-solver sends Φ to the SAT-solver. Let's say that it return $\mathcal{M} = \{l_1, \neg l_2, \neg l_4\}$.

- The $\mathcal{T}$-solver checks if $\mathcal{M}$ is consistent. Let's say it is not. Let $\Phi' = \Phi \cup \neg\mathcal{M} = \{l_1, (\neg l_2 \vee l_3), \neg l_4, (\neg l_1 \vee l_2 \vee l_4)\}$.
- The $\mathcal{T}$-solver sends Φ' to the SAT-solver. Let's say that it return $\mathcal{M}' = \{l_1, l_2, l_3, \neg l_4\}$.
- The $\mathcal{T}$-solver checks if $\mathcal{M}'$ is consistent. Let's say it is not. Let $\Phi'' = \Phi' \cup \neg\mathcal{M}' = \{l_1, (\neg l_2 \vee l_3), \neg l_4, (\neg l_1 \vee l_2 \vee l_4), (\neg l_1 \vee \neg l_2 \vee \neg l_3 \vee l_4)\}$.
- The $\mathcal{T}$-solver sends Φ'' to the SAT-solver and it detects the unsatisfiability. Therefore, φ is $\mathcal{T}$-unsatisfiable.

Optimizations

- Check $\mathcal{T}$-consistency on partial assignments.
- Given a $\mathcal{T}$-inconsistent assignment μ, find a smaller $\mathcal{T}$-inconsistent assignment $\eta \subseteq \mu$ and add $\neg\eta$ to Φ instead of $\neg\mu$.
- When reaching $\mathcal{T}$-inconsistency, backjump to a $\mathcal{T}$-consistent point in the computation.

1.3 CDCL($\mathcal{T}$)

Lazy solver based on CDCL for SAT extended with a $\mathcal{T}$-solver. The $\mathcal{T}$-solver does the following: CDCL($\mathcal{T}$)

- Checks the $\mathcal{T}$-consistency of a conjunction of literals.
- Performs deduction of unassigned literals.
- Explains $\mathcal{T}$-inconsistent assigments.
- Allows to backtrack.

State transition Transition system to describe the reasoning of SAT or SMT solvers. A transition has form: State transition

$$(\mu \| \varphi) \to (\mu' \| \varphi')$$

where:

- φ and φ' are $\mathcal{T}$-formulas.
- μ and μ' are (partial) boolean assignments to atoms of φ and φ', respectively.
- $(\mu \| \varphi)$ and $(\mu' \| \varphi')$ are states.

Transition rule Determine the possible transitions.

Derivation Sequence of transitions.

Initial state $(\varnothing \| \varphi)$.

$\mathcal{T}$-consistency Given a $\mathcal{T}$-formula φ and a full assignment μ of φ, φ is $\mathcal{T}$-consistent ($\mu \models_\mathcal{T} \varphi$) if there is a derivation from $(\varnothing \| \varphi)$ to $(\mu \| \varphi)$.

$\mathcal{T}$-propagation Deduce the assignment of an unassigned literal l using some knowledge of the theory. $\mathcal{T}$-propagation

$\mathcal{T}$-consequence If: $\mathcal{T}$-consequence

- $\mu \models_\mathcal{T} l$,

- l or $\neg l$ occur in φ,
- l and $\neg l$ do not occur in μ,

then:

$$(\mu \| \varphi) \to (\mu \cup \{l\} \| \varphi)$$

Example. Given the formula φ:

$$\big(g(a) = c\big) \land \Big(\big(f(g(a)) \neq f(c)\big) \lor \big(g(a) = d\big)\Big) \land \big(c \neq d\big)$$

A possible derivation for some theory $\mathcal{T}$ (i.e. $\mathcal{T}$-propagation are decided arbitrarily) is:

1. $\varnothing \| \varphi$ (initial state).
2. $\varnothing \| \varphi \to \{l_1\} \| \varphi$ (Unit propagation).
3. $\{l_1\} \| \varphi \to \{l_1, l_2\} \| \varphi$ ($\mathcal{T}$-propagation).
4. $\{l_1, l_2\} \| \varphi \to \{l_1, l_2, l_3\} \| \varphi$ (Unit propagation).
5. $\{l_1, l_2, l_3\} \| \varphi \to \{l_1, l_2, l_3, l_4\} \| \varphi$ ($\mathcal{T}$-propagation).
6. $\{l_1, l_2, l_3, l_4\} \| \varphi \to$ `fail` (Failure).

As we are at decision level 0 (as no decision literal has been fixed), we can conclude that φ is unsatisfiable.

Remark. Unit and theory propagation are alternated (see algorithm description).

Algorithm Given a $\mathcal{T}$-formula φ and a (partial) $\mathcal{T}$-assignment μ (i.e. initial decisions), CDCL($\mathcal{T}$) does the following:

Algorithm 1 CDCL($\mathcal{T}$)

```
def cdclT(φ, μ):
    if preprocess(φ, μ) == CONFLICT: return UNSAT
    φ^p, μ^p = SMT_to_SAT(φ), SMT_to_SAT(μ)
    level = 0
    l = None

    while True:
        status = propagate(φ^p, μ^p, l)
        if status == SAT:
            return SAT_to_SMT(μ^p)
        elif status == UNSAT:
            η^p, jump_level = analyzeConflict(φ^p, μ^p)
            if jump_level < 0: return UNSAT
            backjump(jump_level, φ^p ∧ ¬η^p, μ^p)
        elif status == UNKNOWN:
            l = decideNextLiteral(φ^p, μ^p)
            level += 1
```

Where:

`preprocess` Preprocesses φ with μ through operations such as simplifications, $\mathcal{T}$-specific rewritings, ...

`SMT_TO_SAT` Provides the boolean abstraction of an SMT formula.

`SAT_TO_SMT` Reverses the boolean abstraction of an SMT formula.

`propagate` Iteratively apply:

- Unit propagation,
- $\mathcal{T}$-consistency check,
- $\mathcal{T}$-propagation.

Returns `SAT`, `UNSAT` or `UNKNOWN` (when no deductions are possible and there are still free variables).

`analyzeConflict` Performs conflict analysis:

- If the conflict is detected by SAT boolean propagation ($\mu^p \land \varphi^p \models_p \bot$), a boolean conflict set η^p is outputted (as in standard CDCL).
- If the conflict is detected by $\mathcal{T}$-propatation ($\mu \land \phi \models_\mathcal{T} \bot$), a theory conflict η is produced and its boolean abstraction η^p is outputted.

Moreover, the earliest decision level at which a variable of η^p is unassigned is returned.

As in standard CDCL, $\neg\eta^p$ is added to φ^p and the algorithm backjumps to a previous decision level (if possible).

`decideNextLiteral` Decides the assignment of an unassigned variable (as in standard CDCL). Theory information might be exploited.

Implication graph As in the standard CDCL algorithm, an implication graph is used to explain conflicts. Implication graph

Nodes Decisions, derived literals or conflicts.

Edges If v allows to unit/theory propagate w, then there is an edge $v \to w$.

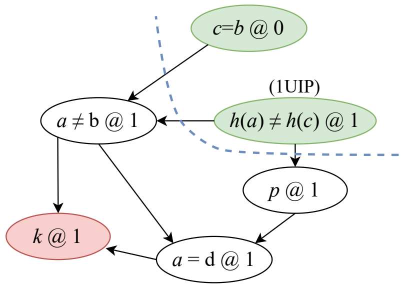
Example. Given the $\mathcal{T}$-formula φ:

$$\big(h(a) = h(c) \lor p\big) \land \big(a = b \lor \neg p \lor a = d\big) \land \big(a \neq d \lor a = b\big)$$

and an initial decision $(c = b) \in \mu$, CDCL($\mathcal{T}$) does the following:

1. As no propagation is possible, the decision $h(a) \neq h(c)$ is added to μ.
2. Unit propagate p due to the clause $\big(h(a) = h(c) \lor p\big)$ and the decision at the previous step.
3. $\mathcal{T}$-propagate $(a \neq b)$ due to the current assigments: $\{c = b, h(a) \neq h(c)\} \models_\mathcal{T} a \neq b$.
4. Unit propagate $(a = d)$ due to the clause $\big(a = b \lor \neg p \lor a = d\big)$ and the current knowledge base (p and $a \neq b$).
5. There is a conflict between $(a \neq d)$ and $(a = d)$.

By building the conflict graph, one can identify the 1UIP as the decision $h(a) \neq h(c)$.



A cut in front of the 1UIP that separates decision nodes and the conflict node (as in standard CDCL) is made to obtain the conflict set $\eta = \{h(a) \neq h(c), c = b\}$. $\big((h(a) = h(c)) \vee (c \neq b)\big)$ is added as a clause and the algorithm backjumps at the decision level 0.