Satisfiability Modulo Theories
Lezione 4 - The Lazy Approach

(slides revision: Saturday 14$^{\text{th}}$ March, 2015, 11:46)

Roberto Bruttomesso

Seminario di Logica Matematica
(Corso Prof. Silvio Ghilardi)

10 Novembre 2011

Outline

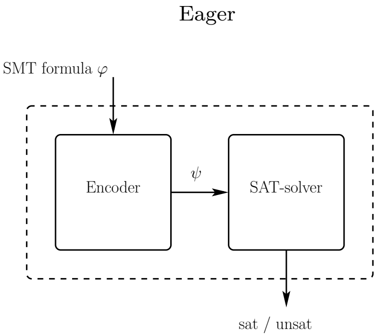
1. The Lazy Approach
   - Intro
   - Lazy Approach as Abstraction Refinement
   - CDCL($\mathcal{T}$)
   - $\mathcal{T}$-solver Features
2. Implementation Details
   - OpenSMT

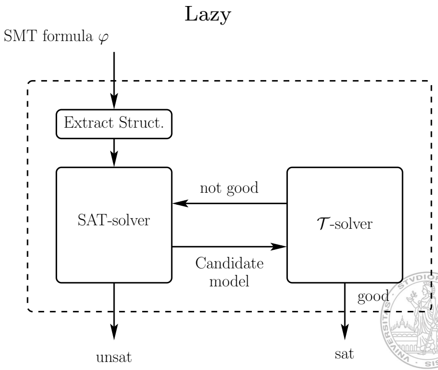
Eager and Lazy

SMT can be reduced to SAT, but requires discovering and adding incompatibilities between $\mathcal{T}$-atoms

Eager and Lazy refers to the time in which these incompatibilities are added to the Boolean structure of the problem

- eager: immediately, before SAT-solver is called as black-box
- **lazy**: on demand, during SAT-solver's search





Lazy Approach

The Lazy Approach builds on top of SAT and of available and well known **decision procedures**, which we call **theory solvers** ($\mathcal{T}$-solvers)

Examples of these $\mathcal{T}$-solvers are the Union-Find procedure for equality, and the Simplex Algorithm for Linear Rational Arithmetic

These procedures are very efficient in handling **conjunctions** of $\mathcal{T}$-atoms, but they don't know how to handle arbitrary Boolean operators

Lazy Approach

The Lazy Approach builds on top of SAT and of available and well known **decision procedures**, which we call **theory solvers** ($\mathcal{T}$-solvers)

Examples of these $\mathcal{T}$-solvers are the Union-Find procedure for equality, and the Simplex Algorithm for Linear Rational Arithmetic

These procedures are very efficient in handling **conjunctions** of $\mathcal{T}$-atoms, but they don't know how to handle arbitrary Boolean operators

Lazy SMT can be seen as an efficient mechanism to extend these procedures to handle generic Boolean combinations of $\mathcal{T}$-atoms

This is achieved with a tight integration between a SAT-solver and the $\mathcal{T}$-solver

In the following we assume that

(i) $\mathcal{T}$ is decidable, and that

(ii) a $\mathcal{T}$-solver for conjunctions of $\mathcal{T}$-atoms exists

A bit of notation

We will use the following notation

Symbol	Meaning
φ	original formula, in some background theory $\mathcal{T}$
$\varphi^{\mathcal{B}}$	the Boolean abstraction of φ
μ	an assignment for φ
$\mu^{\mathcal{B}}$	the assignment for $\varphi^{\mathcal{B}}$ induced by μ

A bit of notation

We will use the following notation

Symbol	Meaning
φ	original formula, in some background theory $\mathcal{T}$
$\varphi^{\mathcal{B}}$	the Boolean abstraction of φ
μ	an assignment for φ
$\mu^{\mathcal{B}}$	the assignment for $\varphi^{\mathcal{B}}$ induced by μ

E.g., where $\mathcal{T}$ is $\mathcal{LIA}$ (Linear Integer Arithmetic)

$$
\begin{array}{rccccccc}
\varphi \equiv & (x+y \leq 0) & \wedge & (x=0) & \wedge & (\neg(y=1) & \vee & (x=1)) \\
\varphi^{\mathcal{B}} \equiv & a_1 & \wedge & a_2 & \wedge & (\neg a_3 & \vee & a_4) \\
\mu \equiv & \multicolumn{7}{l}{\{x \mapsto 0, y \mapsto 0\}} \\
\mu^{\mathcal{B}} \equiv & \multicolumn{7}{l}{\{a_1 \mapsto \top, a_2 \mapsto \top, a_3 \mapsto \bot, a_4 \mapsto \bot\}}
\end{array}
$$

A bit of notation

$$\begin{aligned}
\varphi &\equiv (x + y \leq 0) \wedge (x = 0) \wedge (\neg(y = 1) \vee (x = 1)) \\
\varphi^{\mathcal{B}} &\equiv a_1 \wedge a_2 \wedge (\neg a_3 \vee a_4) \\
\mu &\equiv \{x \mapsto 0, y \mapsto 0\} \\
\mu^{\mathcal{B}} &\equiv \{a_1 \mapsto \top, a_2 \mapsto \top, a_3 \mapsto \bot, a_4 \mapsto \bot\}
\end{aligned}$$

Notice that

$$\mu^{\mathcal{B}} \equiv \{a_1 \mapsto \top, a_2 \mapsto \top, a_3 \mapsto \bot, a_4 \mapsto \bot\} \equiv \{a_1, a_2, \neg a_3, \neg a_4\}$$

is nothing but

$$\{\ (x + y \leq 0),\ (x = 0),\ \neg(y = 1),\ \neg(x = 1)\ \}$$

i.e., it is a **conjunction** of constraints, whose satisfiability can be checked with a $\mathcal{T}$-solver

A bit of notation

$$
\begin{aligned}
\varphi &\equiv (x+y\leq 0) \quad \wedge \quad (x=0) \quad \wedge \quad (\neg(y=1) \quad \vee \quad (x=1))\\
\varphi^{\mathcal{B}} &\equiv \quad a_1 \quad \wedge \quad a_2 \quad \wedge \quad (\neg a_3 \quad \vee \quad a_4)\\
\mu &\equiv \{x\mapsto 0, y\mapsto 0\}\\
\mu^{\mathcal{B}} &\equiv \{a_1\mapsto\top, a_2\mapsto\top, a_3\mapsto\bot, a_4\mapsto\bot\}
\end{aligned}
$$

Notice that

$$\mu^{\mathcal{B}} \equiv \{a_1\mapsto\top, a_2\mapsto\top, a_3\mapsto\bot, a_4\mapsto\bot\} \equiv \{a_1, a_2, \neg a_3, \neg a_4\}$$

is nothing but

$$\{\ (x+y\leq 0),\ (x=0),\ \neg(y=1),\ \neg(x=1)\ \}$$

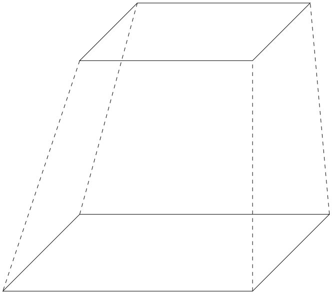
i.e., it is a **conjunction** of constraints, whose satisfiability can be checked with a $\mathcal{T}$-solver

In other words, the $\mathcal{T}$-solver can tell if $\mu^{\mathcal{B}}$ is $\mathcal{T}$-satisfiable

If so, then there is a model μ, that induces $\mu^{\mathcal{B}}$, and if $\mu^{\mathcal{B}}$ is a model for $\varphi^{\mathcal{B}}$ then μ is also a model for φ (take some time to think about it at home)

Assigment relations

$\varphi^{\mathcal{B}}$

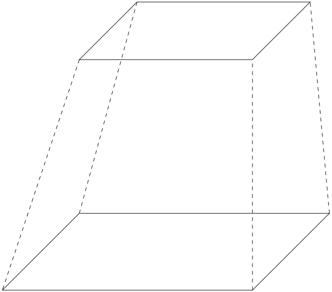


φ

Abstraction

Assigment relations

$\varphi^{\mathcal{B}}$



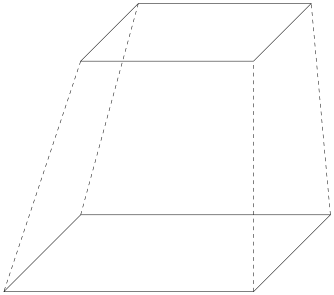
∞
(can be)

φ

Abstraction

Assigment relations

$\varphi^{\mathcal{B}}$



2^n
$(n = |\{a_i\}|)$

φ

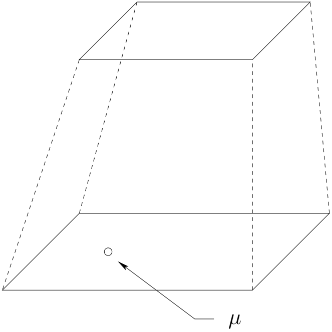
∞
(can be)

Abstraction

Assigment relations

$\varphi^{\mathcal{B}}$

2^n
$(n = |\{a_i\}|)$



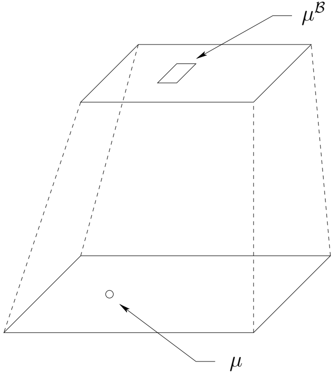
φ

∞
(can be)

Abstraction

Assigment relations

$\varphi^{\mathcal{B}}$



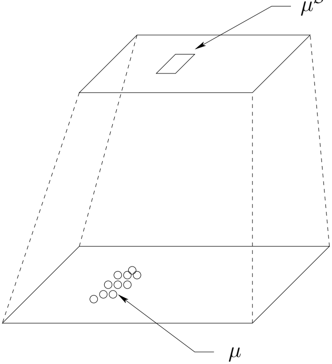
2^n
$(n = |\{a_i\}|)$

φ

∞
(can be)

Assigment relations

$\varphi^{\mathcal{B}}$



2^n

$(n = |\{a_i\}|)$

φ

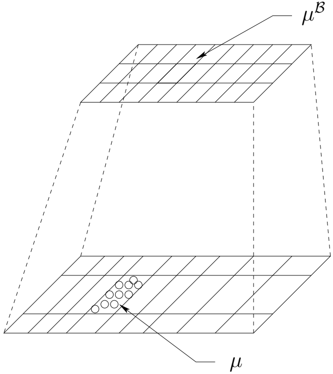
∞

(can be)

Abstraction

Assigment relations

$\varphi^{\mathcal{B}}$



2^n
$(n = |\{a_i\}|)$

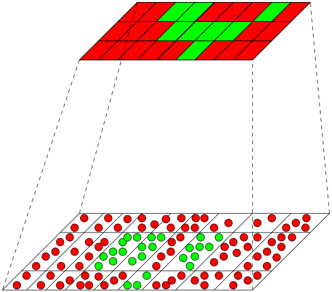
φ

∞
(can be)

Abstraction

Model relations

$\varphi^{\mathcal{B}}$



φ

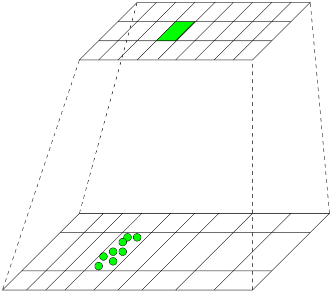
Abstraction

Model relations

- if μ is a model for φ, then $\mu^{\mathcal{B}}$ is a model for $\varphi^{\mathcal{B}}$

$\varphi^{\mathcal{B}}$

φ

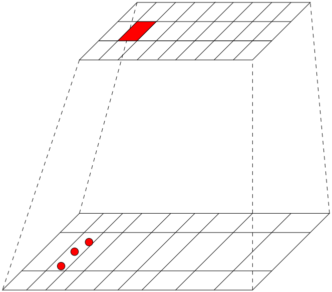


Abstraction

Model relations

- if μ is a model for φ, then $\mu^{\mathcal{B}}$ is a model for $\varphi^{\mathcal{B}}$
- if $\mu^{\mathcal{B}}$ is not a model for $\varphi^{\mathcal{B}}$, then there is no μ that is a model for φ

$\varphi^{\mathcal{B}}$



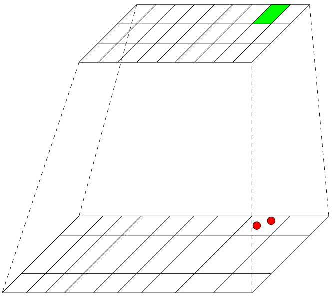
φ

Abstraction

Model relations

- if μ is a model for φ, then $\mu^{\mathcal{B}}$ is a model for $\varphi^{\mathcal{B}}$
- if $\mu^{\mathcal{B}}$ is not a model for $\varphi^{\mathcal{B}}$, then there is no μ that is a model for φ
- there may be some model $\mu^{\mathcal{B}}$ for $\varphi^{\mathcal{B}}$ that does not map to any model μ for φ

$\varphi^{\mathcal{B}}$



φ

Abstraction Refinement

Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

Abstraction Refinement

Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)

Abstraction Refinement

Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)
- Check if $\mu^{\mathcal{B}}$ is satisfiable using the $\mathcal{T}$-solver. If so $\mu^{\mathcal{B}}$ can be extended to a model μ of φ, and so we are done ! (φ is satisfiable)

Abstraction Refinement

Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)
- Check if $\mu^{\mathcal{B}}$ is satisfiable using the $\mathcal{T}$-solver. If so $\mu^{\mathcal{B}}$ can be extended to a model μ of φ, and so we are done ! (φ is satisfiable)
- It not, we tell the SAT-solver not to enumerate $\mu^{\mathcal{B}}$ again, thus **cutting away systematically an infinite number** of assignments for φ (refinement)

Abstraction Refinement

Notice that

- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)
- Check if $\mu^{\mathcal{B}}$ is satisfiable using the $\mathcal{T}$-solver. If so $\mu^{\mathcal{B}}$ can be extended to a model μ of φ, and so we are done ! (φ is satisfiable)
- It not, we tell the SAT-solver not to enumerate $\mu^{\mathcal{B}}$ again, thus **cutting away systematically an infinite number** of assignments for φ (refinement)
- It can be blocked by adding a clause $\neg\mu^{\mathcal{B}}$. Go up

Abstraction Refinement

Notice that

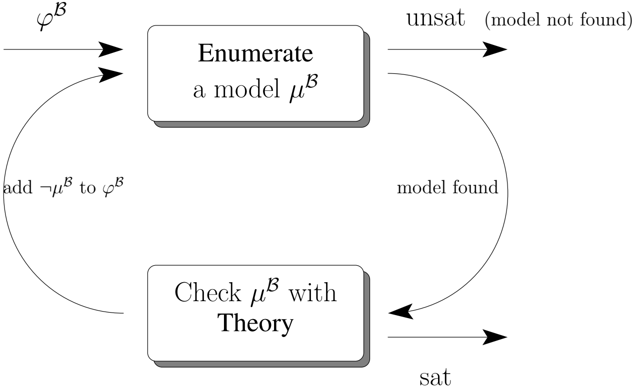
- Assignments μ of φ are many (potentially ∞), infeasible to check if any of them is a model **systematically**
- Models $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ are finite in number, and easy to enumerate with a SAT-solver
- A model $\mu^{\mathcal{B}}$ is nothing but a **conjunction of $\mathcal{T}$-atoms**, can be checked efficiently with a $\mathcal{T}$-solver

These observations suggest us a methodology to tackle the SMT($\mathcal{T}$) problem

- Enumerate a Boolean model $\mu^{\mathcal{B}}$ of $\varphi^{\mathcal{B}}$ (abstraction). If no model exist we are done (φ is unsatisfiable)
- Check if $\mu^{\mathcal{B}}$ is satisfiable using the $\mathcal{T}$-solver. If so $\mu^{\mathcal{B}}$ can be extended to a model μ of φ, and so we are done ! (φ is satisfiable)
- It not, we tell the SAT-solver not to enumerate $\mu^{\mathcal{B}}$ again, thus **cutting away systematically an infinite number** of assignments for φ (refinement)
- It can be blocked by adding a clause $\neg\mu^{\mathcal{B}}$. Go up
- It terminates because there are finite Boolean models

Abstraction Refinement

The lazy approach falls into the so-called **abstraction-refinement** paradigm



CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \end{array}$$

$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: { }

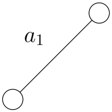
SAT-solver:	Idle
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \end{array}$$



$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{\ a_1\ \}$

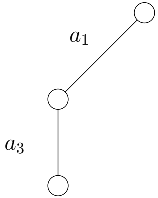
SAT-solver:	Decision
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \end{array}$$



$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{ \; a_1, a_3 \; \}$

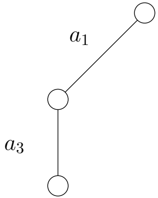
SAT-solver: BCP
$\mathcal{T}$-solver: Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \end{array}$$



$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{\ a_1, a_3\ \}$

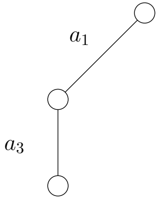
SAT-solver: Idle
$\mathcal{T}$-solver: Is $\mu^{\mathcal{B}}$ $\mathcal{T}$-satisfiable ?

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \end{array}$$



$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{\ a_1, a_3\ \}$

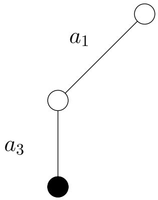
SAT-solver: Idle
$\mathcal{T}$-solver: Is $\mu^{\mathcal{B}}$ $\mathcal{T}$-satisfiable ? NO

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \ \wedge \ (x = 3 \vee \neg(x > 3)) \ \wedge \ (x > 3 \vee \neg(x < 3)) \ \wedge \ (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \quad \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \end{array}$$



$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{ \ a_1, a_3 \ \}$

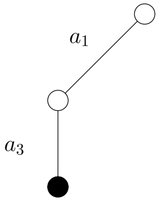
SAT-solver: Learn
$\mathcal{T}$-solver: Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \end{array}$$



$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: { }

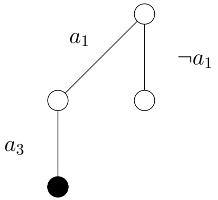
SAT-solver:	Conf. Analysis, Backtrack
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \end{array}$$



$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{ \neg a_1 \}$

SAT-solver:	BCP
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

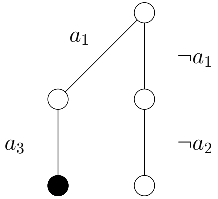
$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \quad \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \end{array}$$

$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{ \neg a_1, \neg a_2 \}$

SAT-solver:	BCP
$\mathcal{T}$-solver:	Idle

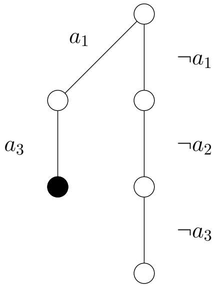


CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \end{array}$$



$$\begin{array}{l} a_1 \equiv x = 3 \\ a_2 \equiv x < 3 \\ a_3 \equiv x > 3 \end{array}$$

$\mu^{\mathcal{B}}$: $\{ \neg a_1, \neg a_2, \neg a_3 \}$

SAT-solver:	BCP
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

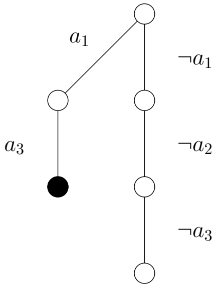
The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\begin{aligned}\varphi^{\mathcal{B}} \equiv \quad & (a_1 \vee \neg a_2)\\ & (a_1 \vee \neg a_3)\\ & (a_3 \vee \neg a_2)\\ & (a_3 \vee \neg a_1)\\ & (\neg a_1 \vee \neg a_3)\\ & (\neg a_1)\end{aligned}$$

$$\begin{aligned} a_1 &\equiv x = 3\\ a_2 &\equiv x < 3\\ a_3 &\equiv x > 3\end{aligned}$$

$\mu^{\mathcal{B}}$: $\{\ \neg a_1, \neg a_2, \neg a_3\ \}$



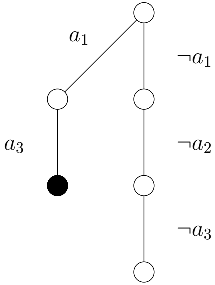
SAT-solver: Idle
$\mathcal{T}$-solver: Is $\mu^{\mathcal{B}}$ $\mathcal{T}$-satisfiable ?

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \end{array}$$



$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{\ \neg a_1, \neg a_2, \neg a_3\ \}$

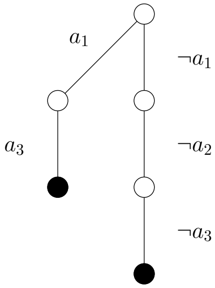
SAT-solver: Idle
$\mathcal{T}$-solver: Is $\mu^{\mathcal{B}}$ $\mathcal{T}$-satisfiable ? NO

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\varphi^{\mathcal{B}} \equiv \begin{array}{l} (a_1 \vee \neg a_2) \\ (a_1 \vee \neg a_3) \\ (a_3 \vee \neg a_2) \\ (a_3 \vee \neg a_1) \\ (\neg a_1 \vee \neg a_3) \\ (\neg a_1) \\ (a_1 \vee a_2 \vee a_3) \end{array}$$



$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{ \neg a_1, \neg a_2, \neg a_3 \}$

SAT-solver:	Learn
$\mathcal{T}$-solver:	Idle

CDCL($\mathcal{T}$)

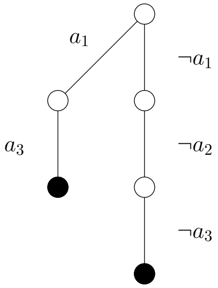
The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\begin{aligned}\varphi^{\mathcal{B}} \equiv \quad & (a_1 \vee \neg a_2)\\ & (a_1 \vee \neg a_3)\\ & (a_3 \vee \neg a_2)\\ & (a_3 \vee \neg a_1)\\ & (\neg a_1 \vee \neg a_3)\\ & (\neg a_1)\\ & (a_1 \vee a_2 \vee a_3)\\ & (\ )\end{aligned}$$

$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{\ \ \}$



SAT-solver: Conf. Analysis, Backtrack
$\mathcal{T}$-solver: Idle

CDCL($\mathcal{T}$)

The interaction described naturally falls within the CDCL style, enriched with a $\mathcal{T}$-solver

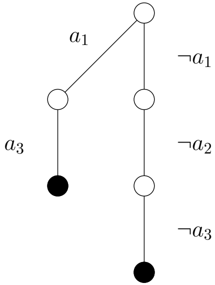
$$\varphi \equiv (x = 3 \vee \neg(x < 3)) \;\wedge\; (x = 3 \vee \neg(x > 3)) \;\wedge\; (x > 3 \vee \neg(x < 3)) \;\wedge\; (x > 3 \vee \neg(x = 3))$$

$$\begin{aligned} \varphi^{\mathcal{B}} \equiv \quad & (a_1 \vee \neg a_2) \\ & (a_1 \vee \neg a_3) \\ & (a_3 \vee \neg a_2) \\ & (a_3 \vee \neg a_1) \\ & (\neg a_1 \vee \neg a_3) \\ & (\neg a_1) \\ & (a_1 \vee a_2 \vee a_3) \\ & (\ ) \end{aligned}$$

$a_1 \equiv x = 3$
$a_2 \equiv x < 3$
$a_3 \equiv x > 3$

$\mu^{\mathcal{B}}$: $\{\ \ \}$

SAT-solver: UNS
$\mathcal{T}$-solver: Idle



Early Pruning

Notice that there is no need to wait until a (partial) Boolean model is found to call $\mathcal{T}$-solver

Suppose that the first Boolean model is

$$\mu^{\mathcal{B}} = \{x < y, x = y, \ldots (\text{ 1000 other constraints })\}$$

This Boolean model is $\mathcal{T}$-unsatisfiable already at $\{x < y, x = y\}$, and could have been stopped much earlier

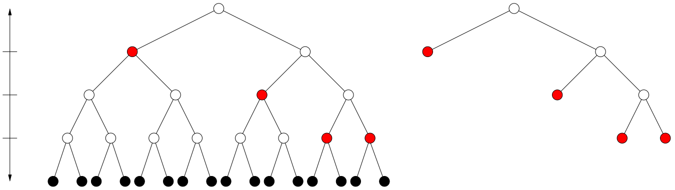
Early Pruning

Notice that there is no need to wait until a (partial) Boolean model is found to call $\mathcal{T}$-solver

Suppose that the first Boolean model is

$$\mu^{\mathcal{B}} = \{x < y, x = y, \ldots (\text{ 1000 other constraints })\}$$

This Boolean model is $\mathcal{T}$-unsatisfiable already at $\{x < y, x = y\}$, and could have been stopped much earlier



Position in the search in which $\mu^{\mathcal{B}}$ is already $\mathcal{T}$-unsatisfiable

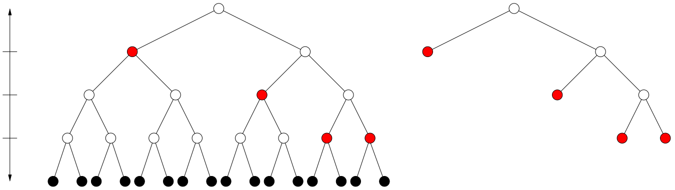
Early Pruning

Notice that there is no need to wait until a (partial) Boolean model is found to call $\mathcal{T}$-solver

Suppose that the first Boolean model is

$$\mu^{\mathcal{B}} = \{x < y, x = y, \ldots (\text{ 1000 other constraints })\}$$

This Boolean model is $\mathcal{T}$-unsatisfiable already at $\{x < y, x = y\}$, and could have been stopped much earlier



Position in the search in which $\mu^{\mathcal{B}}$ is already $\mathcal{T}$-unsatisfiable

However, do not call $\mathcal{T}$-solver too often, as it may slow things down

Good heuristic: call $\mathcal{T}$-solver just after any BCP

The CDCL Procedure

```
dl = 0; trail = { };                                   // Decision level, assignment

while ( true )

  if ( BCP( ) == conflict )                            // Do BCP until possible
    if ( dl == 0 ) return unsat;                       // Unresolvable conflict
    C, dl = AnalyzeConflict( );                        // Compute conf. clause, and dec. level
    AddClause( C );                                    // Add C to clause database
    BacktrackTo( dl );                                 // Backtracking (shrinks trail)
  else
    if ( "all variables assigned" ) return sat;        // trail holds satisfying assignment
    l = Decision( );                                   // Do another decision
    trail = trail ∪ {l}
    dl = dl + 1;                                       // Increase decision level
```

The (basic) CDCL($\mathcal{T}$) Procedure

```
class Theory;                                        // T-solver

dl = 0; trail = { };                                 // Decision level, assignment

while ( true )

  if ( BCP( ) == conflict )                          // Do BCP until possible
    if ( dl == 0 ) return unsat;                     // Unresolvable conflict
    C, dl = AnalyzeConflict( );                      // Compute conf. clause, and dec. level
    AddClause( C );                                  // Add C to clause database
    BacktrackTo( dl );                               // Backtracking (shrinks trail)
  else if ( Theory.Check( trail ) == unsat )         // T-solver check
    AddClause( ¬trail );                             // Add clause that is now unsat
  else
    if ( "all variables assigned" ) return sat;      // trail holds satisfying assignment
    l = Decision( );                                 // Do another decision
    trail = trail ∪ {l}
    dl = dl + 1;                                     // Increase decision level
```

Incrementality and Backtrackability

$\mathcal{T}$-solver is asked to check consistency on sets of constraints that evolve **incrementally** because of BCP and Decide actions

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$$

and can be backtracked because of Conflict Analysis and Backtracking

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0, x \overset{0}{<} y\ \}$$

Incrementality and Backtrackability

$\mathcal{T}$-solver is asked to check consistency on sets of constraints that evolve **incrementally** because of BCP and Decide actions

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$$

and can be backtracked because of Conflict Analysis and Backtracking

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0, x \overset{0}{<} y\ \}$$

Everything happens in a stack-based fashion: constraints are pushed and popped on $\mu^{\mathcal{B}}$ in a LIFO order

For efficiency, $\mathcal{T}$-solver should be able to

- reason incremetally: check of consistency of $\{x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$ should reuse as much as possible the computation already spent for checking $\{x \overset{0}{=} 0, x \overset{0}{<} y\}$

Incrementality and Backtrackability

$\mathcal{T}$-solver is asked to check consistency on sets of constraints that evolve **incrementally** because of BCP and Decide actions

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$$

and can be backtracked because of Conflict Analysis and Backtracking

$$\mu^{\mathcal{B}} = \{\ x \overset{0}{=} 0, x \overset{0}{<} y\ \}$$

Everything happens in a stack-based fashion: constraints are pushed and popped on $\mu^{\mathcal{B}}$ in a LIFO order

For efficiency, $\mathcal{T}$-solver should be able to

- reason incremetally: check of consistency of $\{x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$ should reuse as much as possible the computation already spent for checking $\{x \overset{0}{=} 0, x \overset{0}{<} y\}$
- backtrack efficiently: going back from $\{x \overset{0}{=} 0,\ x \overset{0}{<} y,\ x + \overset{1}{y} = z\ \}$ to $\{x \overset{0}{=} 0, x \overset{0}{<} y\}$ should be done quickly and without losing information

Minimal Conflicts

It is desirable for the $\mathcal{T}$-solver to return **minimal conflicts**

Consider the assignment

$$\mu^{\mathcal{B}} = \{x - y \leq 0,\ y - z \leq 0,\ \ldots(1000 \text{ other constratints }) \ldots, z - x \leq -1,\ \}$$

according to our basic procedure we would add the clause $\neg\mu^{\mathcal{B}}$ to the SAT-solver

Minimal Conflicts

It is desirable for the $\mathcal{T}$-solver to return **minimal conflicts**

Consider the assignment

$$\mu^{\mathcal{B}} = \{x - y \le 0,\ y - z \le 0,\ \ldots (1000 \text{ other constratints }) \ldots, z - x \le -1,\ \}$$

according to our basic procedure we would add the clause $\neg\mu^{\mathcal{B}}$ to the SAT-solver

However we see that

$$\nu^{\mathcal{B}} = \{x - y \le 0,\ y - z \le 0, z - x \le -1\}$$

is already a minimal reason for the $\mathcal{T}$-unsatisfiability, and $\neg\nu^{\mathcal{B}}$ is a clause with 3 literals instead of 1003. We call these reasons $\mathcal{T}$-conflicts

A $\mathcal{T}$-conflict is **minimal** if it does not contain redundant $\mathcal{T}$-atoms

Minimal Conflicts

It is desirable for the $\mathcal{T}$-solver to return **minimal conflicts**

Consider the assignment

$$\mu^{\mathcal{B}} = \{x - y \leq 0,\ y - z \leq 0,\ \ldots(1000 \text{ other constratints })\ldots, z - x \leq -1,\ \}$$

according to our basic procedure we would add the clause $\neg\mu^{\mathcal{B}}$ to the SAT-solver

However we see that

$$\nu^{\mathcal{B}} = \{x - y \leq 0,\ y - z \leq 0, z - x \leq -1\}$$

is already a minimal reason for the $\mathcal{T}$-unsatisfiability, and $\neg\nu^{\mathcal{B}}$ is a clause with 3 literals instead of 1003. We call these reasons $\mathcal{T}$-conflicts

A $\mathcal{T}$-conflict is **minimal** if it does not contain redundant $\mathcal{T}$-atoms

Small clauses are more retrictive than big clauses, and they therefore reduce SAT search

Theory Propagation

So far we have seen that $\mathcal{T}$-solver is **passive** as far as the search is concerned:

- SAT is the master that drives the search
- $\mathcal{T}$-solver is queried to confirm that the search is correct from the point of view of $\mathcal{T}$

However, consider the following scenario

$$\mu^{\mathcal{B}} = \{\ldots, x \overset{2}{>} 0,\ y \overset{3}{>} 0\}$$

and assume that

- BCP has completed
- There is a $\mathcal{T}$-atom $x + y > 0$ that is currently not assigned (i.e., it is not in $\mu^{\mathcal{B}}$)

Also we know that in $\mathcal{T}$ the following implication holds

$$(x > 0) \wedge (y > 0) \rightarrow (x + y > 0)$$

Then we can do a **Theory Propagation**, i.e., we can expand the assignment as

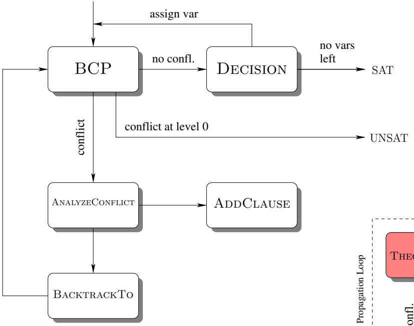
$$\mu^{\mathcal{B}} = \{\ldots, x \overset{2}{>} 0,\ y \overset{3}{>} 0, (x + y \overset{3}{>} 0)\}$$

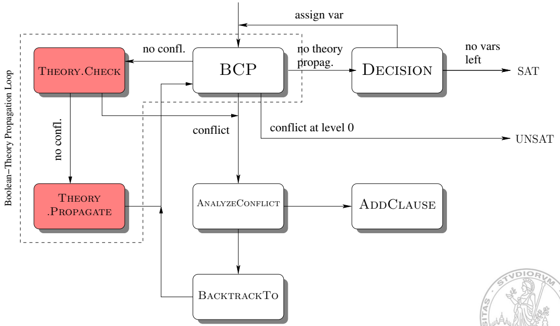
thus avoiding a Decision in SAT

The CDCL($\mathcal{T}$) Procedure

```
class THEORY;                                   // T-solver

dl = 0; trail = { };                            // Decision level, assignment

while ( true )

  if ( BCP( ) == conflict )                     // Do BCP until possible
    if ( dl == 0 ) return unsat;                // Unresolvable conflict
    C, dl = ANALYZECONFLICT( );                 // Compute conf. clause, and dec. level
    ADDCLAUSE( C );                             // Add C to clause database
    BACKTRACKTO( dl );                          // Backtracking (shrinks trail)
  else if ( THEORY.CHECK( trail ) == unsat )    // T-solver check
    ν^B = THEORY.GETCONFLICT( );                // Retrieve T-conflict
    ADDCLAUSE( ¬ν^B );                          // Add clause that is now unsat
  else if ( THEORY.CANPROPAGATE( ) )            // Can do some propagations ?
    ρ = THEORY.PROPAGATE( );                    // Retrieve propagations
    trail = trail ∪ ρ;                          // Extend assignment
  else
    if ( "all variables assigned" ) return sat; // trail holds satisfying assignment
    l = DECISION( );                            // Do another decision
    trail = trail ∪ {l};
    dl = dl + 1;                                // Increase decision level
```

The CDCL($\mathcal{T}$) Procedure





Outline

1. The Lazy Approach
   - Intro
   - Lazy Approach as Abstraction Refinement
   - CDCL($\mathcal{T}$)
   - $\mathcal{T}$-solver Features
2. Implementation Details
   - OpenSMT

OpenSMT

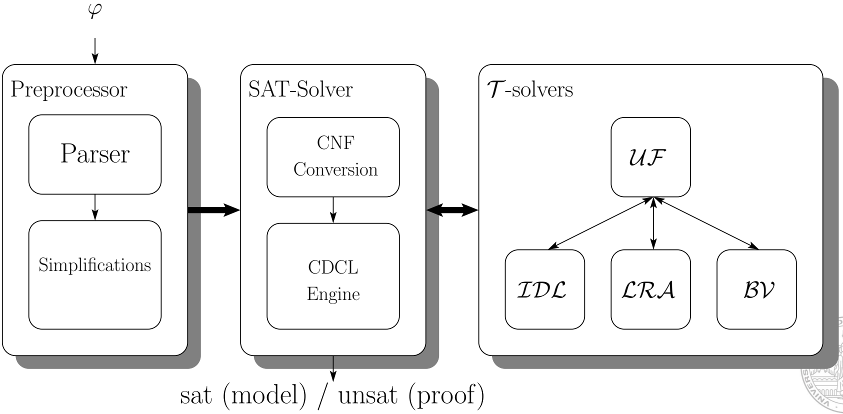
OpenSMT is an open-source SMT-solver that implements the lazy approach

It uses MiniSAT 2.0 as SAT-solver, and it has its own implementations for $\mathcal{T}$-solvers

OpenSMT

OpenSMT is an open-source SMT-solver that implements the lazy approach

It uses MiniSAT 2.0 as SAT-solver, and it has its own implementations for $\mathcal{T}$-solvers



$\mathcal{T}$-solver interface (src/tsolvers/TSolver.h)

OpenSMT features a minimalistic API for $\mathcal{T}$-solvers

```
class TSolver
{
  [...]

  lbool      inform                  ( Enode * );
  bool       assertLit               ( Enode *, bool = false );
  bool       check                   ( bool );
  void       pushBacktrackPoint      ( );
  void       popBacktrackPoint       ( );

  [...]
};
```

`Enode`: a data-structure that represents formulæ and $\mathcal{T}$-atoms

`inform`: tells the $\mathcal{T}$-solver that a $\mathcal{T}$-atom exists

`assertLit`: tells the $\mathcal{T}$-solver that a $\mathcal{T}$-atom is assigned

`check`: consistency check

`push/pop`: set/restore backtrack points

Exercizes

Assuming $\mathcal{T} = \mathcal{LIA}$

1. For the example on slide 11, write the resolution steps of the two calls to conflict analysis

2. Find two different minimal $\mathcal{T}$-conflicts for

   $\mu^{\mathcal{B}} = \{x+y \leq 0,\ 2x+y \leq -1,\ -x+y \leq 5,\ x+2y \leq 2,\ -3x-3y \leq -3\,\}$

   (notice that minimal $\mathcal{T}$-conflicts may have different size)

3. Suppose that $\mu^{\mathcal{B}} = \{x = 0, \neg(x + y > 0)\}$ and that other $\mathcal{T}$-atoms $y \leq 0$, $x = 1$, $x + y \leq -10$ are currently unassigned. What is $\mu^{\mathcal{B}}$ after an exhaustive application of theory-propagation ?