# Recent and ongoing developments of model-constructing satisfiability

Stéphane Graham-Lengrand
SRI International

FroCoS, September 2019

Recent and ongoing developments of model-constructing satisfiability . . . at SRI International

Overview of MCSAT (as implemented in Yices)

Application to the bitvector theory

Application to intuitionistic logic

Quantifier elimination: towards MCSAT for quantified problems?

# 1. Overview of MCSAT

# The model-constructing approach to SMT-solving 1/2

MCSAT introduced in [dMJ13, JBdM13, Jov17],
based on Conflict Resolution [KTV09] and other works on decision procedures such as

- LPSAT [WW99]
- Separation logic [WIGG05]
- Linear Rational Arithmetic [MKS09, KTV09, Cot10]
- Linear Integer Arithmetic [Jd11]
- Non-Linear Arithmetic [JdM12] (NLSAT)

# The model-constructing approach to SMT-solving 1/2

MCSAT introduced in [dMJ13, JBdM13, Jov17],
based on Conflict Resolution [KTV09] and other works on decision procedures such as

- LPSAT [WW99]
- Separation logic [WIGG05]
- Linear Rational Arithmetic [MKS09, KTV09, Cot10]
- Linear Integer Arithmetic [Jd11]
- Non-Linear Arithmetic [JdM12] (NLSAT)

MCSAT offers:

- a template for decision procedures
- an integration of such procedures with Boolean reasoning

# The model-constructing approach to SMT-solving 1/2

MCSAT introduced in [dMJ13, JBdM13, Jov17],
based on Conflict Resolution [KTV09] and other works on decision procedures such as

- LPSAT [WW99]
- Separation logic [WIGG05]
- Linear Rational Arithmetic [MKS09, KTV09, Cot10]
- Linear Integer Arithmetic [Jd11]
- Non-Linear Arithmetic [JdM12] (NLSAT)

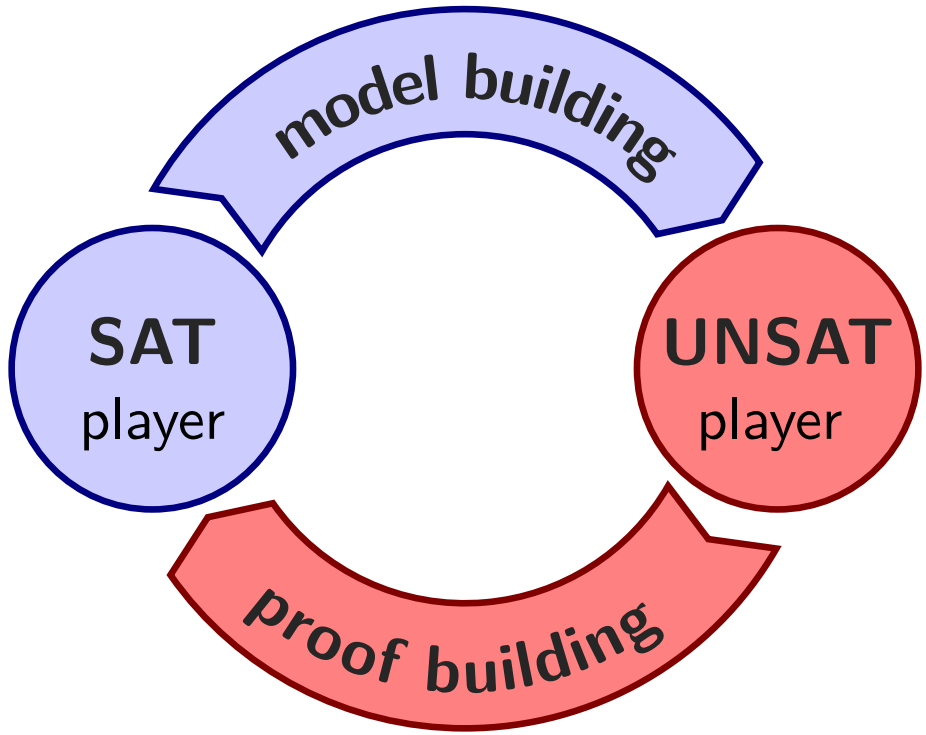
MCSAT offers:

- a template for decision procedures
- an integration of such procedures with Boolean reasoning

The template is a generalisation of how CDCL works.
It is an instance of conflict-driven reasoning.

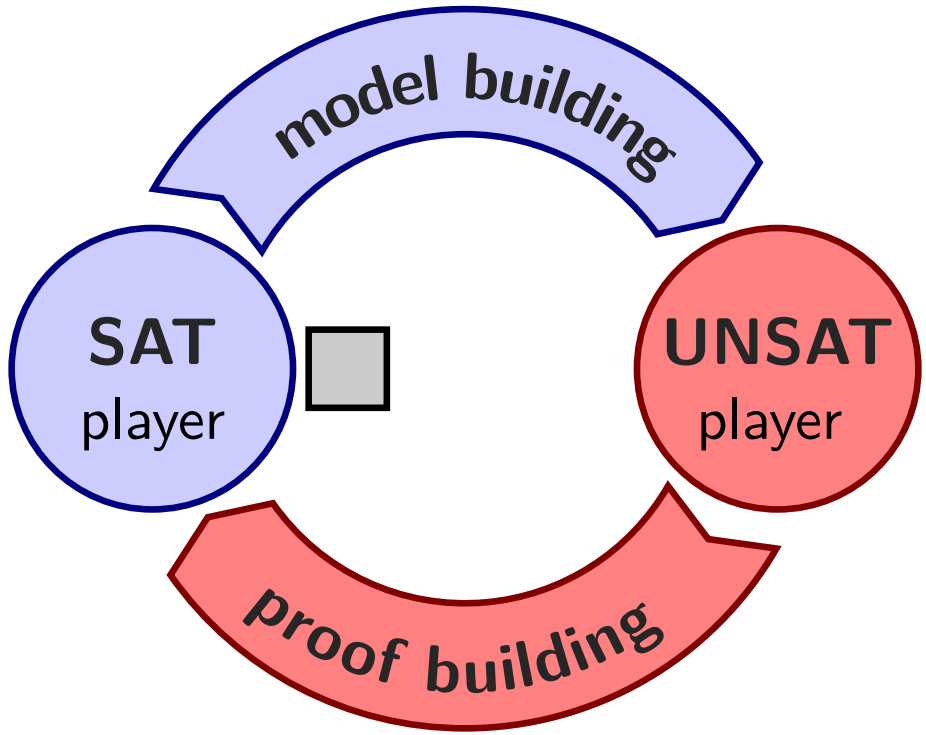
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



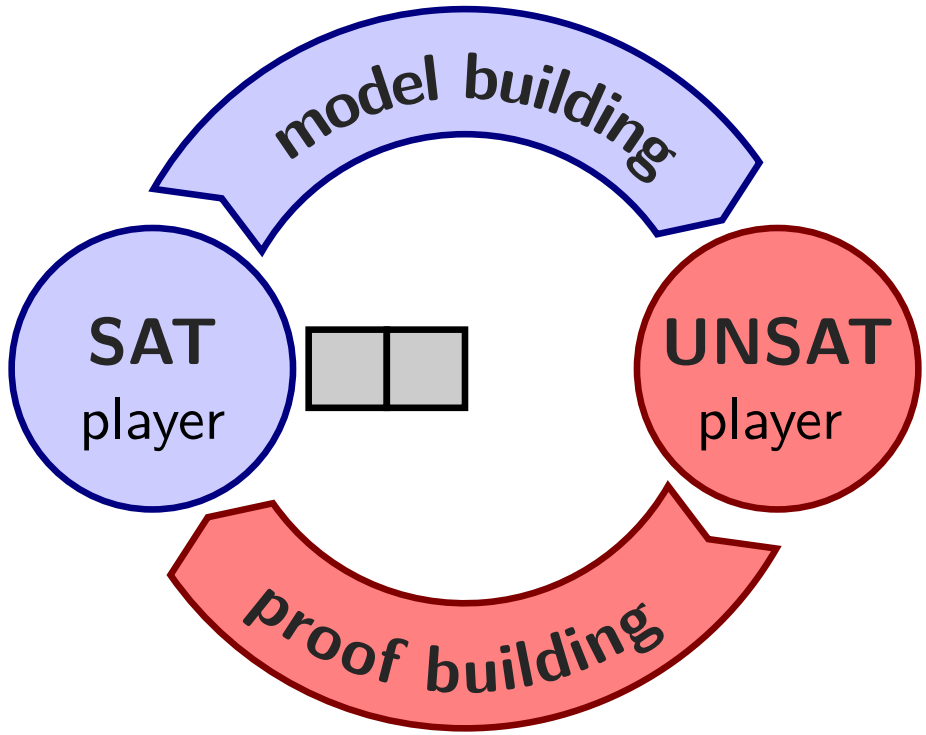
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



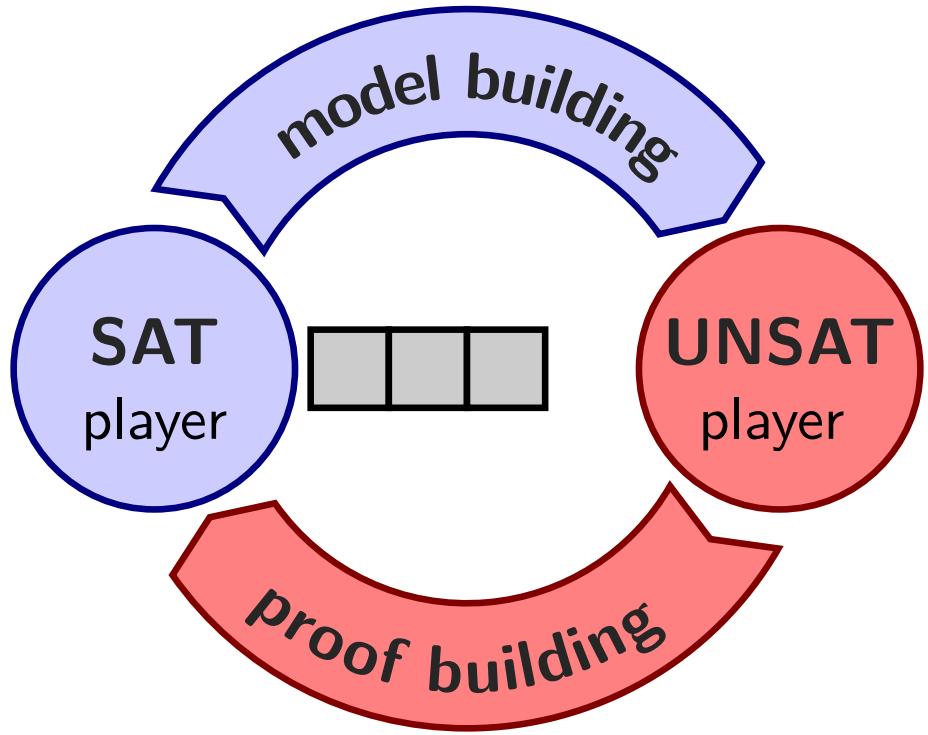
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



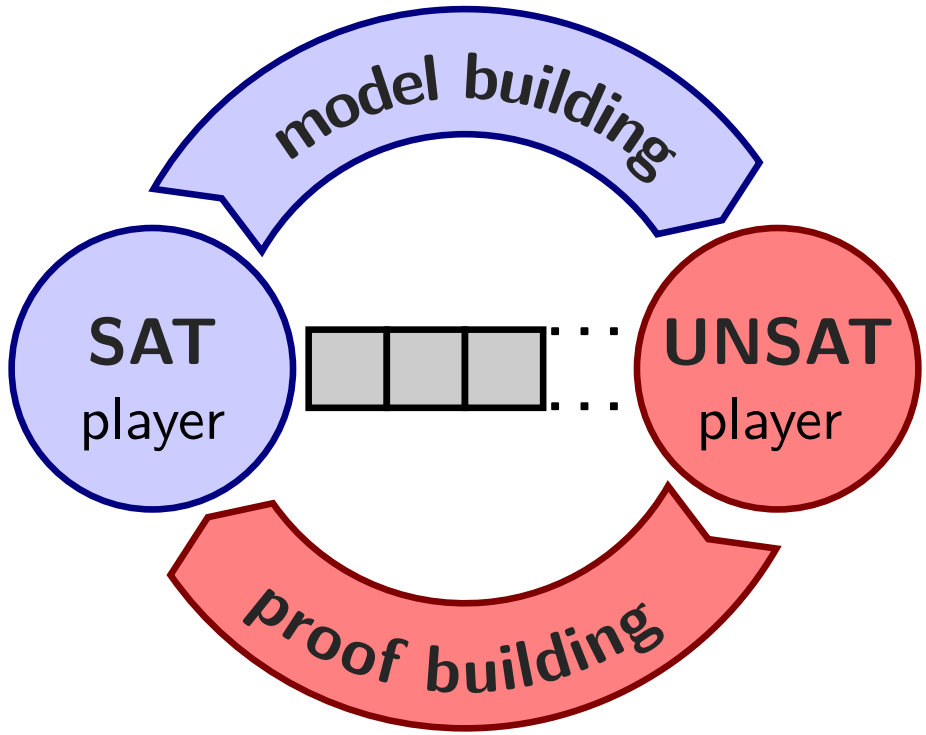
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



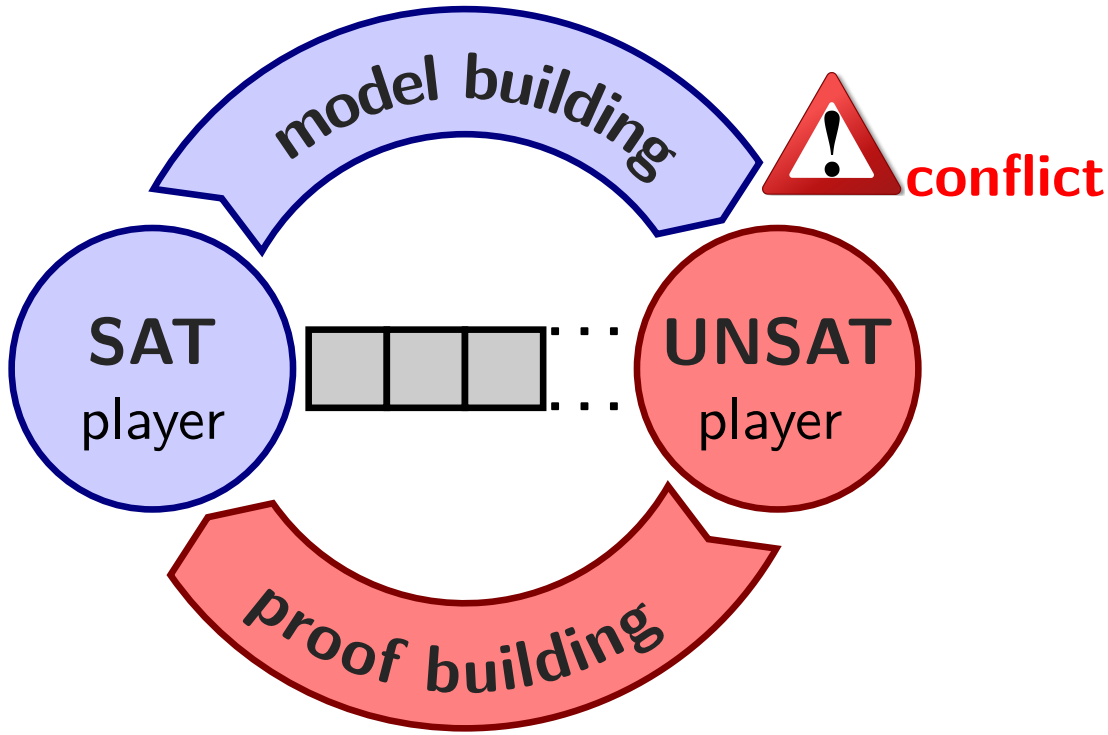
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



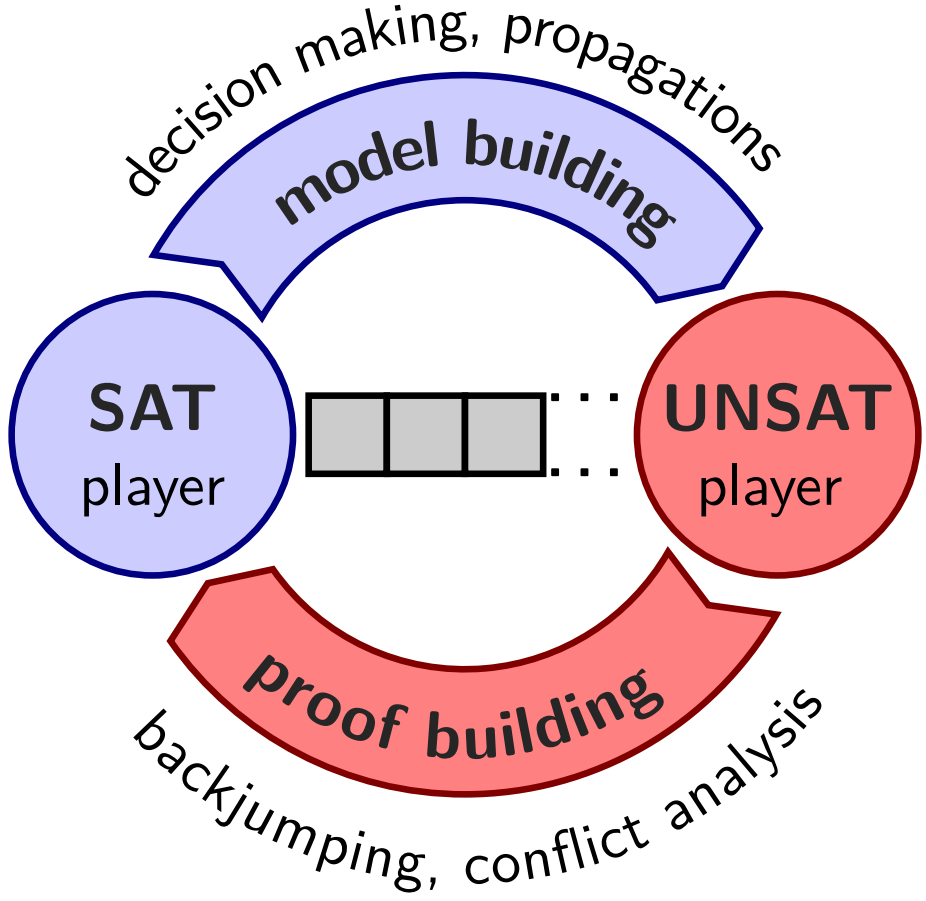
# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.



# Conflict-driven reasoning

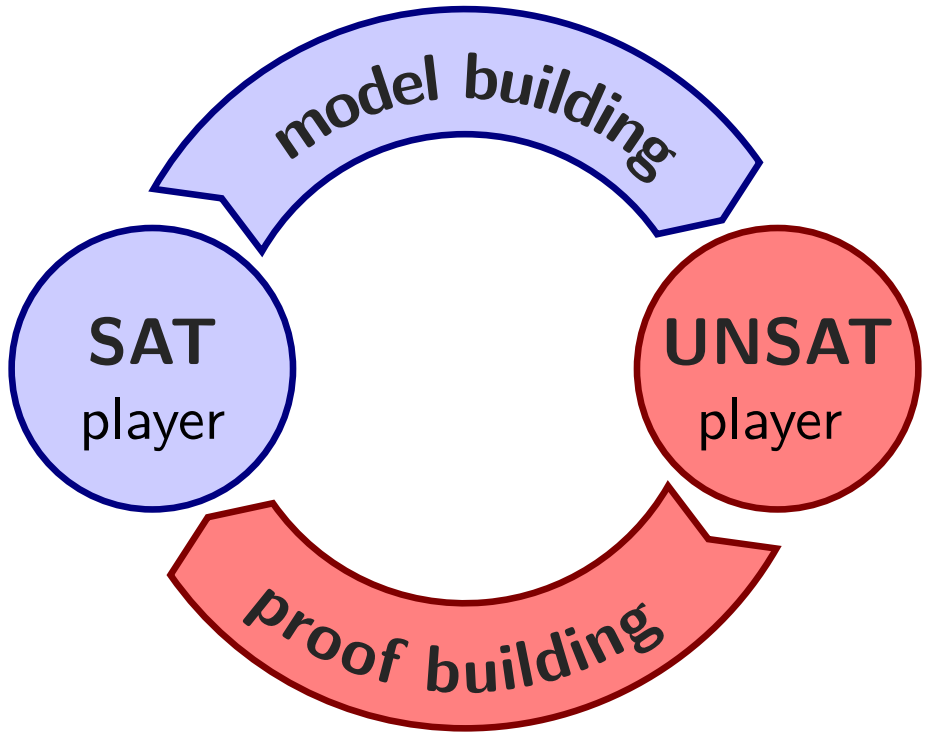
2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

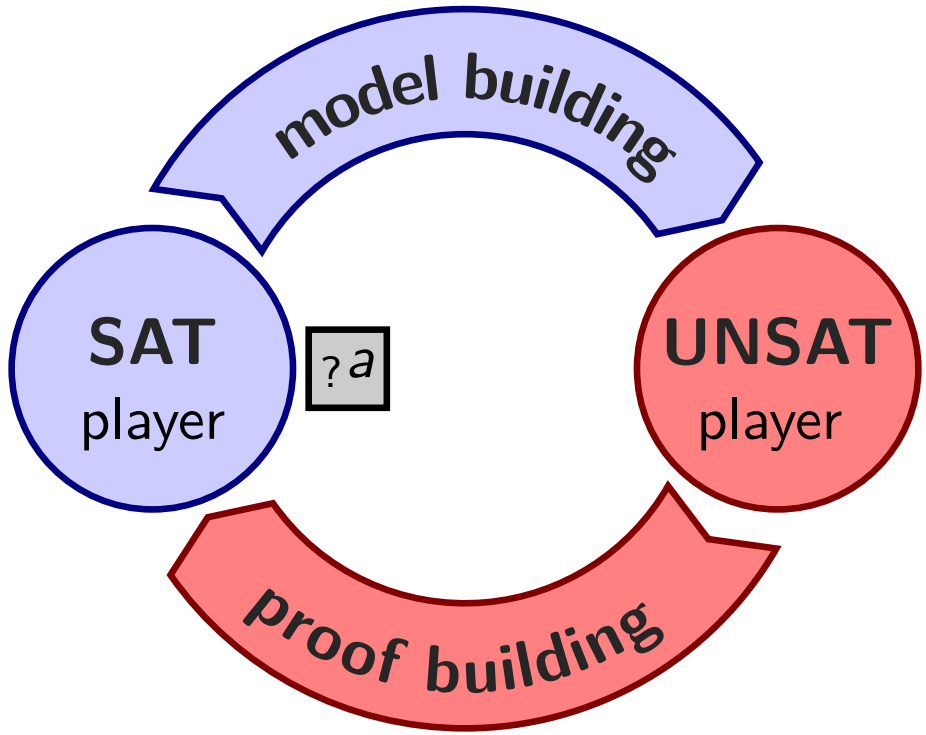
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

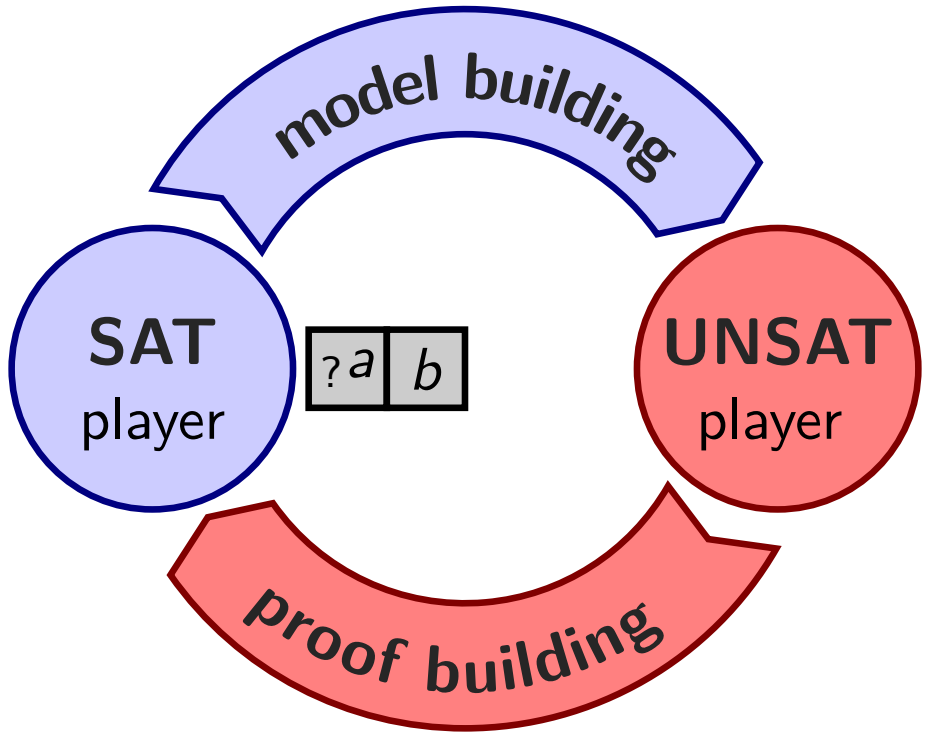
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

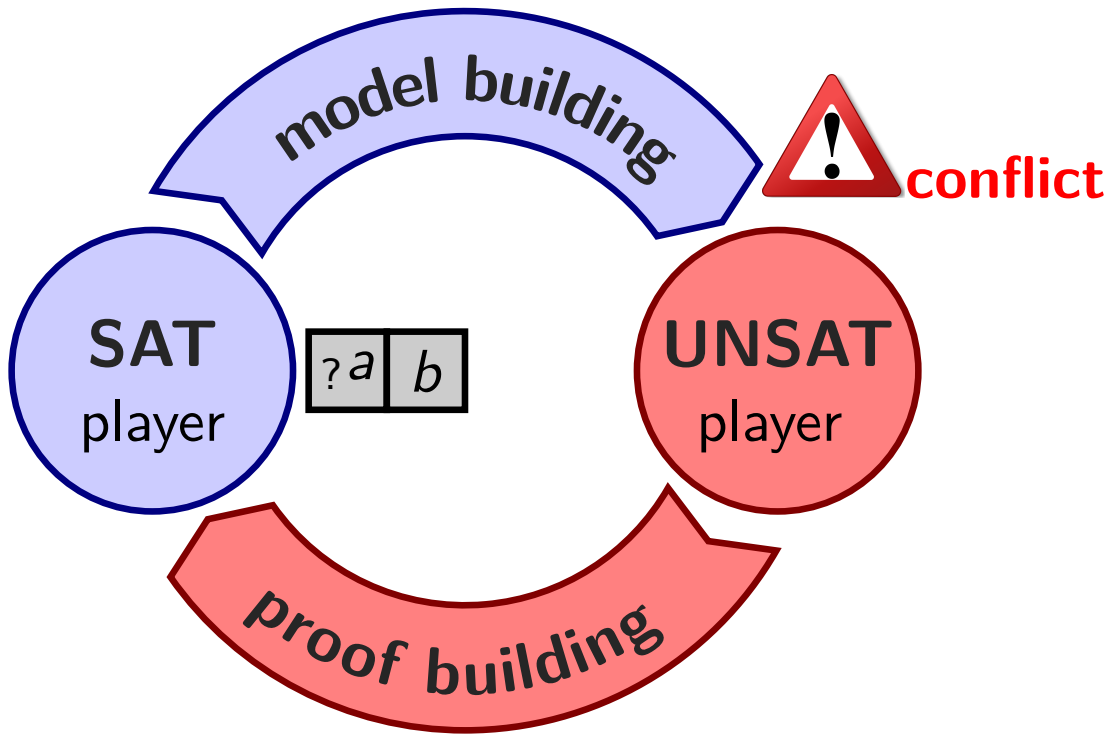
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

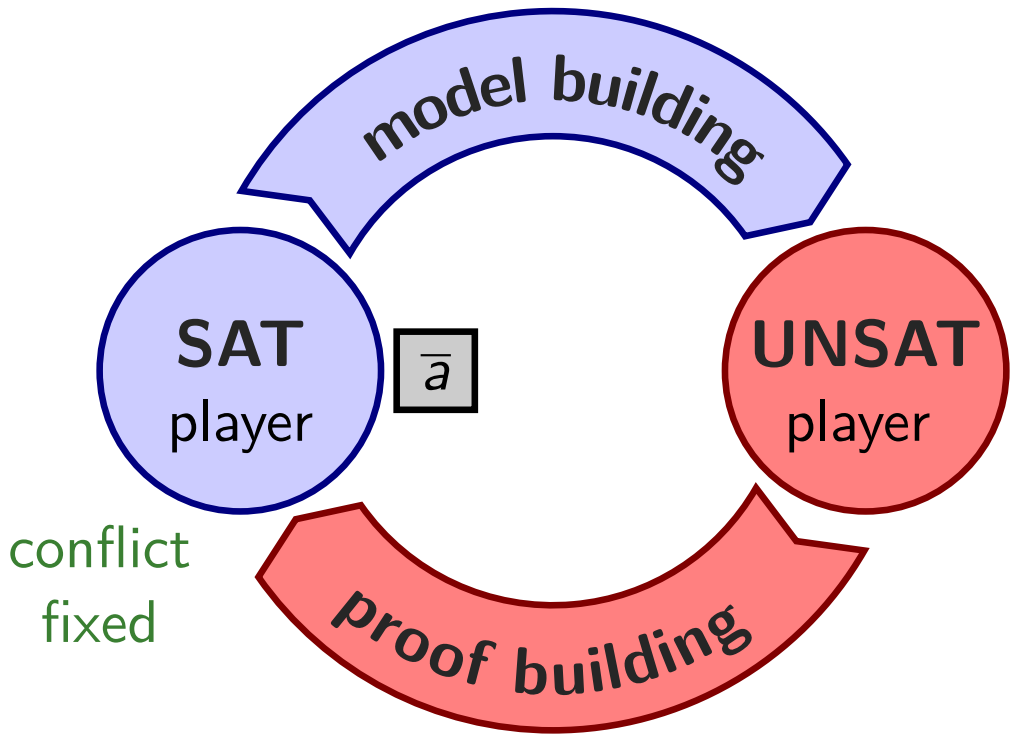
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

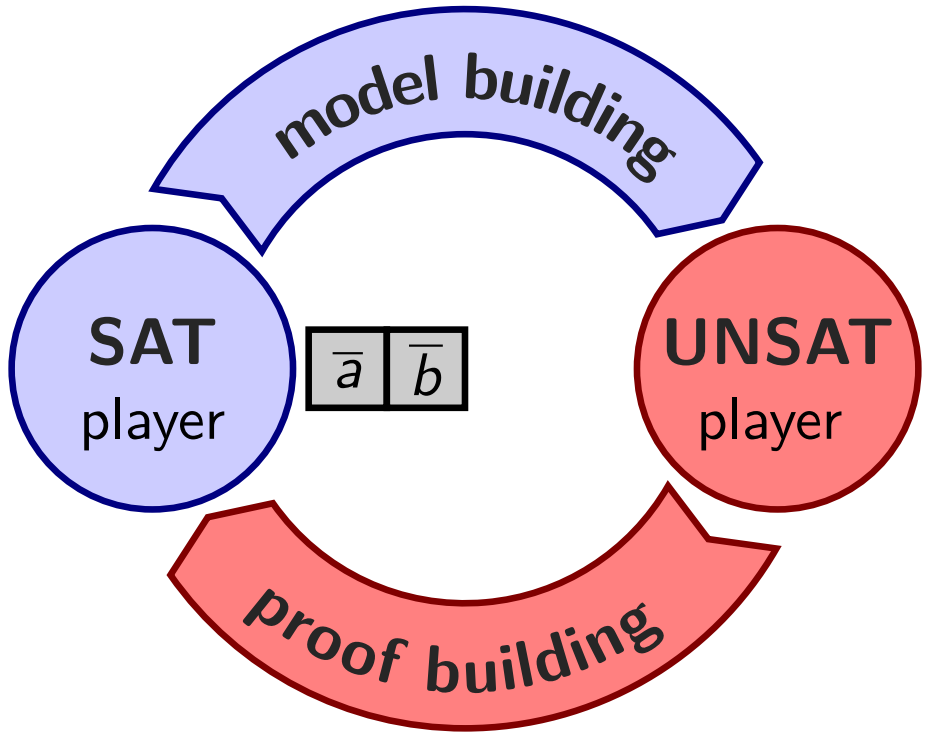
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$
$\overline{a}$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

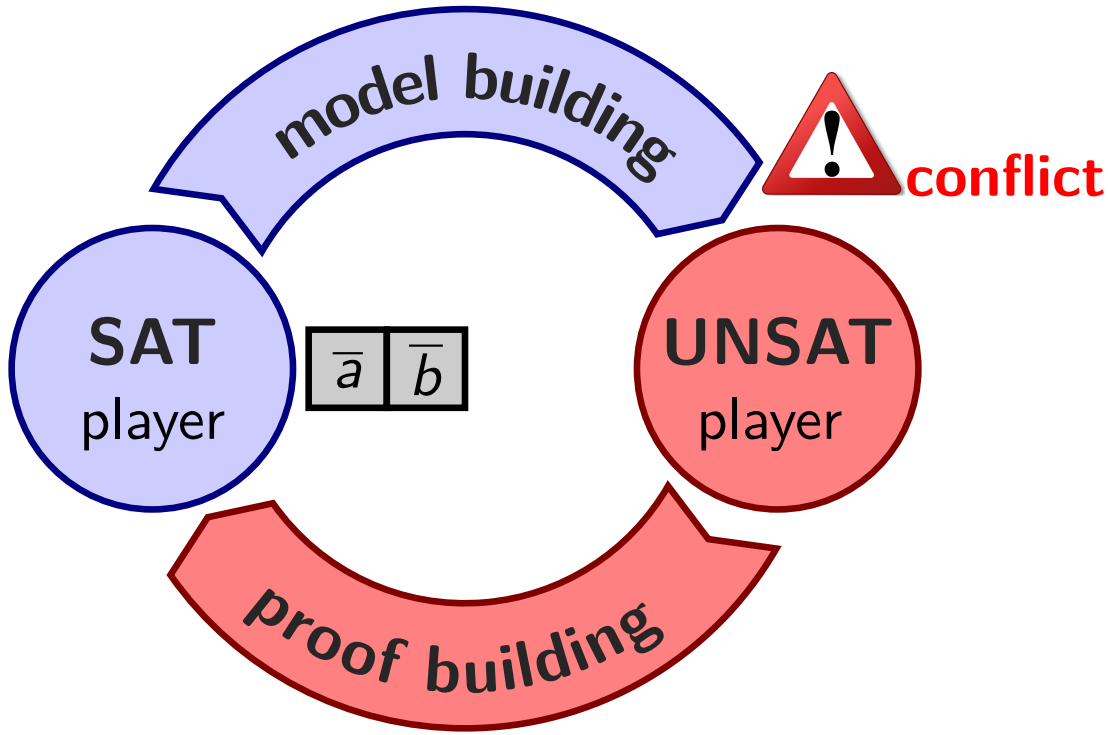
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$
$\overline{a}$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

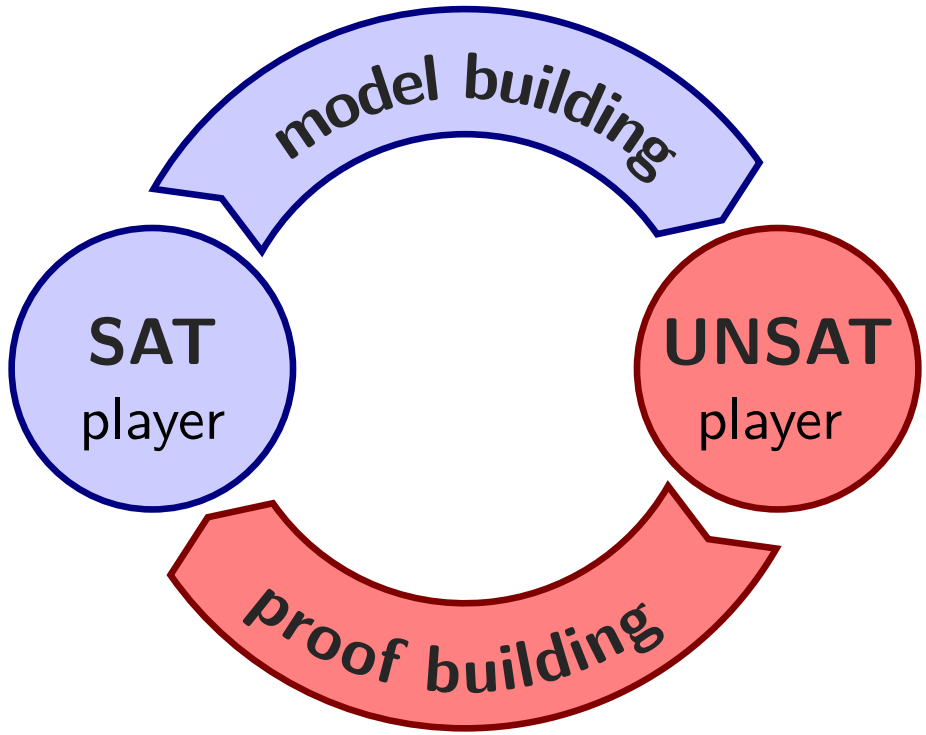
$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$
$\overline{a}$



# Conflict-driven reasoning

2-player game to determine whether a formula is satisfiable.
It involves a trail where a putative model is being specified.
It relies on a notion of conflict between the putative model and the formula it should satisfy.
Archetype of conflict-driven reasoning: **CDCL**
a conflict occurs when a clause is falsified

$a \Rightarrow b$
$b \Rightarrow \overline{a}$
$\overline{a} \Rightarrow \overline{b}$
$\overline{b} \Rightarrow a$
$\overline{a}$
$\bot$



# MCSAT vs CDSAT

MCSAT tailored to theories with a standard model used for evaluating constraints (example: arithmetic)

**Evaluation** is a key aspect of MCSAT

# MCSAT vs CDSAT

MCSAT tailored to theories with a standard model used for evaluating constraints (example: arithmetic)

**Evaluation** is a key aspect of MCSAT

Solving satisfiability problem

(set of constraints on variables $x_1, \ldots, x_n$)

= finding values for variables $x_1, \ldots, x_n$

(so that constraints evaluate to true)

# MCSAT vs CDSAT

MCSAT tailored to theories with a standard model used for evaluating constraints (example: arithmetic)

**Evaluation** is a key aspect of MCSAT

Solving satisfiability problem

(set of constraints on variables $x_1, \ldots, x_n$)

= finding values for variables $x_1, \ldots, x_n$

(so that constraints evaluate to true)

CDSAT [BGLS19] (for Conflict-Driven Satisfiability) is a more abstract framework where

- evaluation is not a mandatory ingredient of search
- theory reasoning is abstracted using inference systems
- theory reasoning can be performed in a union of theories
- Boolean theory can be given the same status as other theories.

# MCSAT vs CDSAT

MCSAT tailored to theories with a standard model used for evaluating constraints (example: arithmetic)

**Evaluation** is a key aspect of MCSAT

Solving satisfiability problem

(set of constraints on variables $x_1, \ldots, x_n$)

= finding values for variables $x_1, \ldots, x_n$

(so that constraints evaluate to true)

CDSAT [BGLS19] (for Conflict-Driven Satisfiability) is a more abstract framework where

- evaluation is not a mandatory ingredient of search
- theory reasoning is abstracted using inference systems
- theory reasoning can be performed in a union of theories
- Boolean theory can be given the same status as other theories.

As an abstract framework, it counts among its instances:

- Equality sharing scheme (Nelson-Oppen combinations)
- CDCL (with restarts, learning, etc)
- MCSAT (original [dMJ13] version)

# The model-constructing approach to SMT-solving 2/2

Run = alternation of search phases and conflict analysis phases

- It uses assignments to first-order variables (e.g., $x \leftarrow 3/4$) like CDCL uses Boolean assignments to Boolean variables;
- It may explain conflicts by introducing atoms that are not in the input.

# The model-constructing approach to SMT-solving 2/2

Run = alternation of search phases and conflict analysis phases

- It uses assignments to first-order variables (e.g., $x \leftarrow 3/4$) like CDCL uses Boolean assignments to Boolean variables;
- It may explain conflicts by introducing atoms that are not in the input.
- As in CDCL, it successively guesses values to assign to variables. . .

  . . . while maintaining the invariant: *given the assignments made so far, none of the constraints evaluates to false*

# The model-constructing approach to SMT-solving 2/2

Run = alternation of search phases and conflict analysis phases

- It uses assignments to first-order variables (e.g., $x \leftarrow 3/4$) like CDCL uses Boolean assignments to Boolean variables;
- It may explain conflicts by introducing atoms that are not in the input.
- As in CDCL, it successively guesses values to assign to variables...

  ...while maintaining the invariant: *given the assignments made so far, none of the constraints evaluates to false*
- To pick a value for variable $y$ after $x_1, \ldots, x_n$ were assigned values $v_1, \ldots, v_n$, simply worry about constraints over variables $x_1, \ldots, x_n, y$ (i.e. constraints that have become unit in $y$)

# The model-constructing approach to SMT-solving 2/2

Run = alternation of search phases and conflict analysis phases

- It uses assignments to first-order variables (e.g., $x \leftarrow 3/4$) like CDCL uses Boolean assignments to Boolean variables;
- It may explain conflicts by introducing atoms that are not in the input.
- As in CDCL, it successively guesses values to assign to variables...

  ...while maintaining the invariant: *given the assignments made so far, none of the constraints evaluates to false*
- To pick a value for variable $y$ after $x_1, \ldots, x_n$ were assigned values $v_1, \ldots, v_n$, simply worry about constraints over variables $x_1, \ldots, x_n, y$ (i.e. constraints that have become unit in $y$)
- If all variables get values while maintaining invariant $\Rightarrow$ SAT

# The model-constructing approach to SMT-solving 2/2

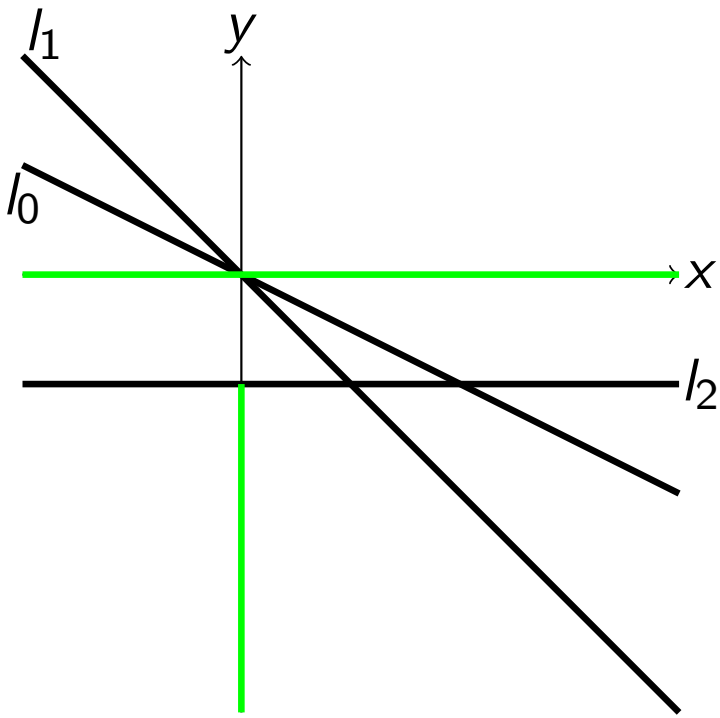
Run = alternation of search phases and conflict analysis phases

- It uses assignments to first-order variables (e.g., $x \leftarrow 3/4$) like CDCL uses Boolean assignments to Boolean variables;
- It may explain conflicts by introducing atoms that are not in the input.
- As in CDCL, it successively guesses values to assign to variables...
  ...while maintaining the invariant: *given the assignments made so far, none of the constraints evaluates to false*
- To pick a value for variable $y$ after $x_1, \ldots, x_n$ were assigned values $v_1, \ldots, v_n$, simply worry about constraints over variables $x_1, \ldots, x_n, y$ (i.e. constraints that have become unit in $y$)
- If all variables get values while maintaining invariant $\Rightarrow$ SAT
- If at any point the invariant cannot be maintained:
  There is a conflict.
  MCSAT performs a conflict analysis,
  backtracks over some of the assignments $x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n$
  and tries new ones

# An example in Linear Rational Arithmetic

$$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}, \qquad \overbrace{(y + x < 0)}^{l_1}, \qquad \overbrace{(y < -1)}^{l_2}$$

unsatisfiable in Linear Rational Arithmetic (LRA).

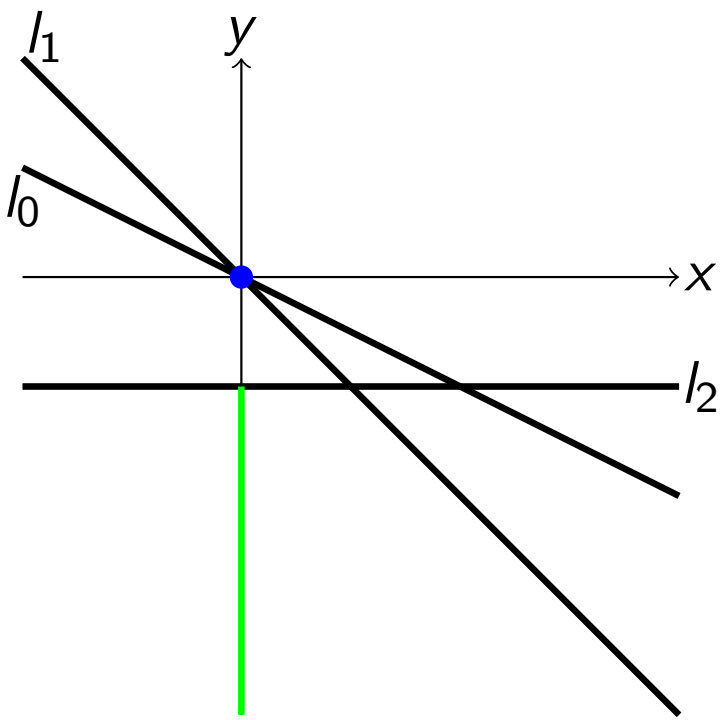


# An example in Linear Rational Arithmetic

$$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}, \quad \overbrace{(y + x < 0)}^{l_1}, \quad \overbrace{(y < -1)}^{l_2}$$

unsatisfiable in Linear Rational Arithmetic (LRA).

- Guess a value, e.g., $x \leftarrow 0$

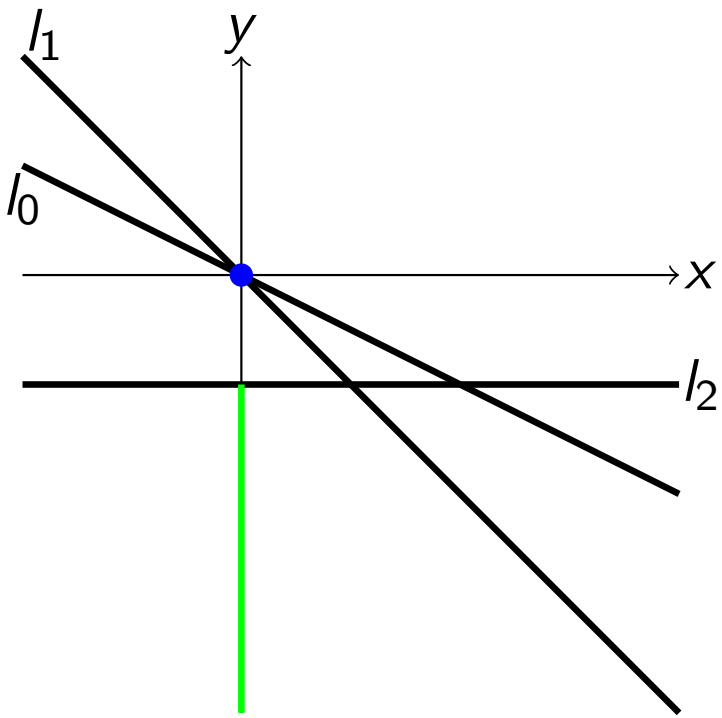


# An example in Linear Rational Arithmetic

$$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}, \quad \overbrace{(y + x < 0)}^{l_1}, \quad \overbrace{(y < -1)}^{l_2}$$

unsatisfiable in Linear Rational Arithmetic (LRA).

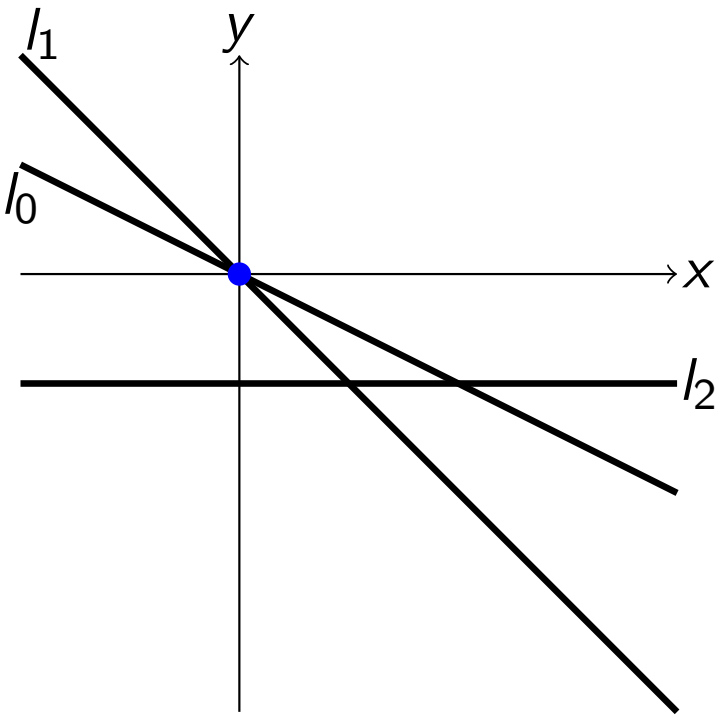
- Guess a value, e.g., $x \leftarrow 0$
  Then $l_0$ yields lower bound $y > 0$



# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}, \quad \overbrace{(y + x < 0)}^{l_1}, \quad \overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

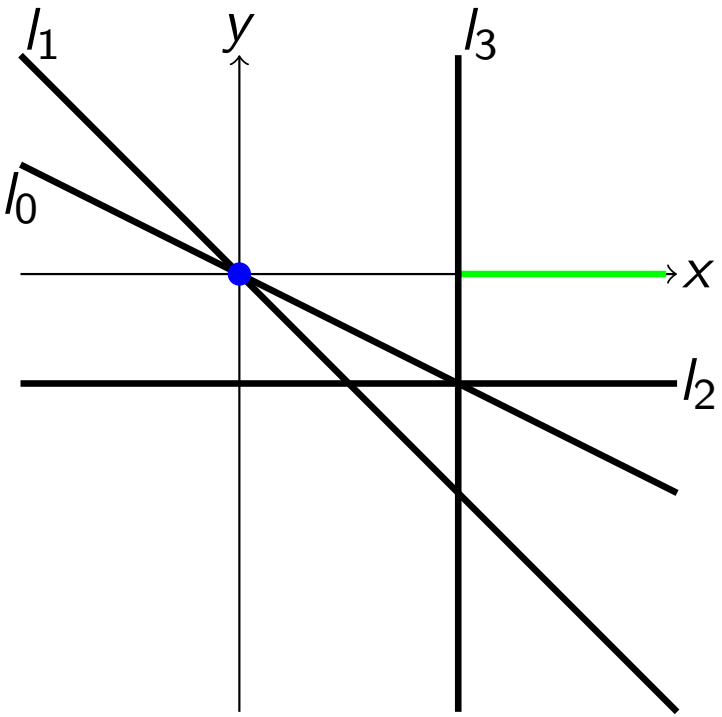



- Guess a value, e.g., $x \leftarrow 0$
  Then $l_0$ yields lower bound $y > 0$
  Together with $l_2$, range of possible values for $y$ is empty. What to do?
  Just undo $x \leftarrow 0$ & remember $x \neq 0$?

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).




- Guess a value, e.g., $x \leftarrow 0$
  Then $l_0$ yields lower bound $y > 0$
  Together with $l_2$, range of possible values for $y$ is empty. What to do?
  Just undo $x \leftarrow 0$ & remember $x \neq 0$?
- **No!** Clash of bounds suggests a better conflict explanation, by inferring

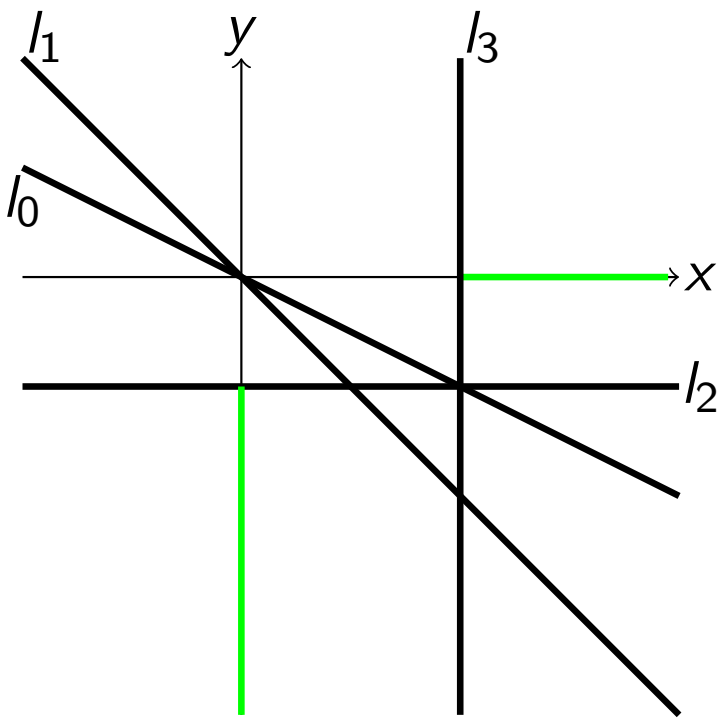
  $l_0 + 2l_2$, i.e., $\overbrace{(-x < -2)}^{l_3}$
  It rules out $x \leftarrow 0$, but also many values that would fail for the same reasons.

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$



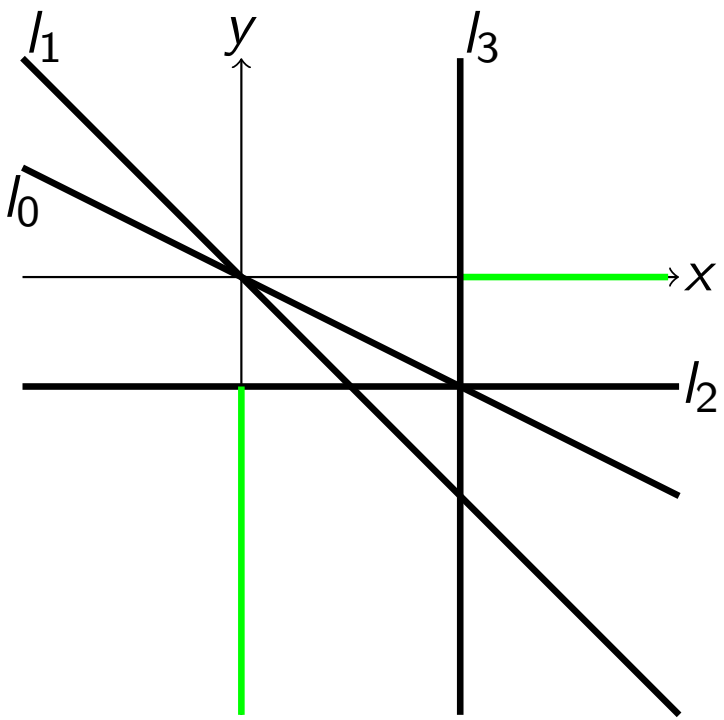

- Guess a value, e.g., $x \leftarrow 0$
  Then $l_0$ yields lower bound $y > 0$
  Together with $l_2$, range of possible values for $y$ is empty. What to do?
  Just undo $x \leftarrow 0$ & remember $x \neq 0$?
- **No!** Clash of bounds suggests a better conflict explanation, by inferring
  $l_0 + 2l_2$, i.e., $\overbrace{(-x < -2)}^{l_3}$
  It rules out $x \leftarrow 0$, but also many values that would fail for the same reasons.
- Now undo the guess but keep $l_3$.

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$




- Guess a value, e.g., $x \leftarrow 0$
  Then $l_0$ yields lower bound $y > 0$
  Together with $l_2$, range of possible values for $y$ is empty. What to do?
  Just undo $x \leftarrow 0$ & remember $x \neq 0$?
- **No!** Clash of bounds suggests a better conflict explanation, by inferring
  $l_0 + 2l_2$, i.e., $\overbrace{(-x < -2)}^{l_3}$
  It rules out $x \leftarrow 0$, but also many values that would fail for the same reasons.
- Now undo the guess but keep $l_3$.

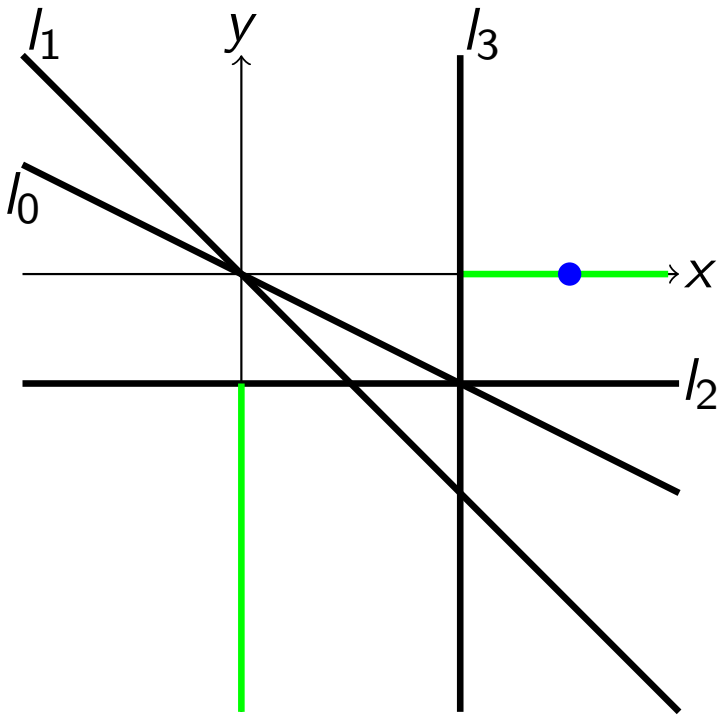
# An example in Linear Rational Arithmetic

$$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}, \quad \overbrace{(y + x < 0)}^{l_1}, \quad \overbrace{(y < -1)}^{l_2}$$

unsatisfiable in Linear Rational Arithmetic (LRA).

$$\overbrace{(-x < -2)}^{l_3}$$

- Guess a value, e.g., $x \leftarrow 3$

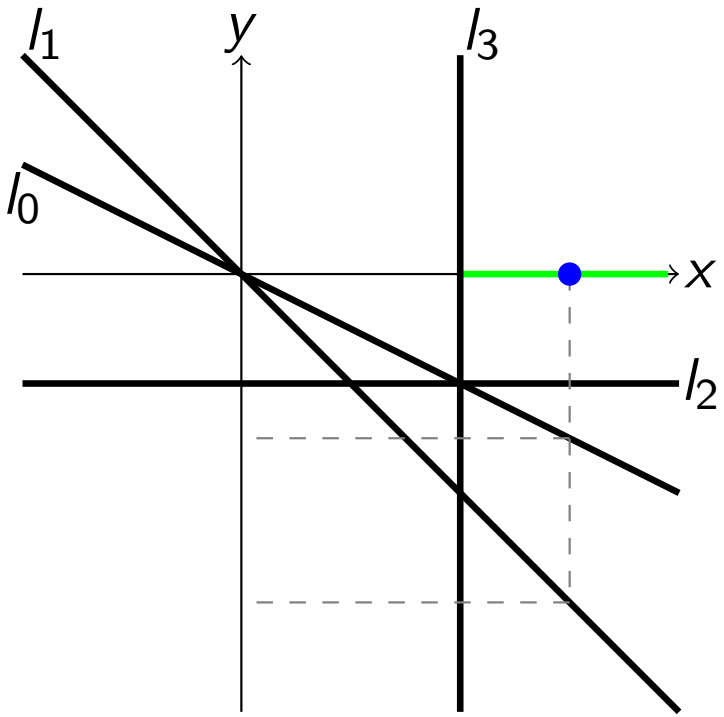


# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$



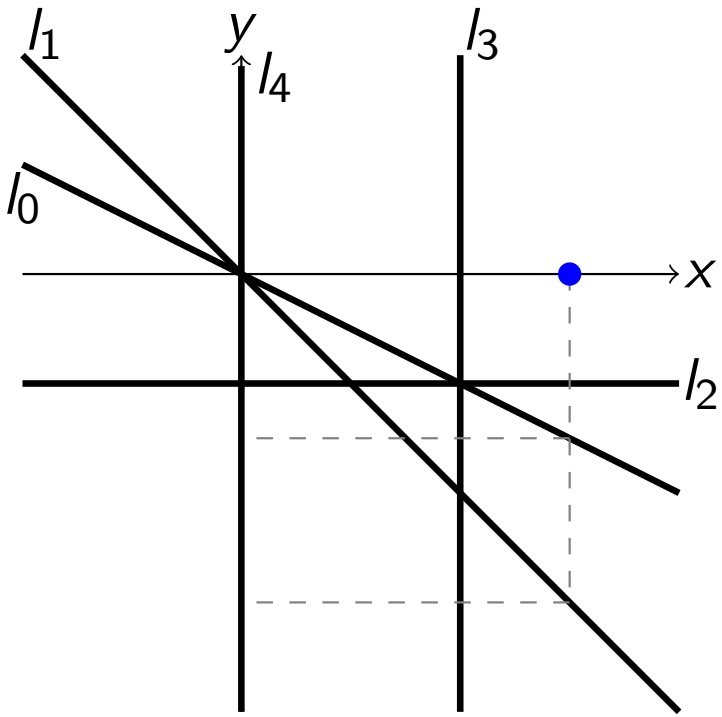

- Guess a value, e.g., $x \leftarrow 3$
  Then $l_0$ yields lower bound $y > -\frac{3}{2}$
  and $l_1$ yields upper bound $y < -3$

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$




- Guess a value, e.g., $x \leftarrow 3$
  Then $l_0$ yields lower bound $y > -\frac{3}{2}$
  and $l_1$ yields upper bound $y < -3$
- Clash of bounds suggests inferring

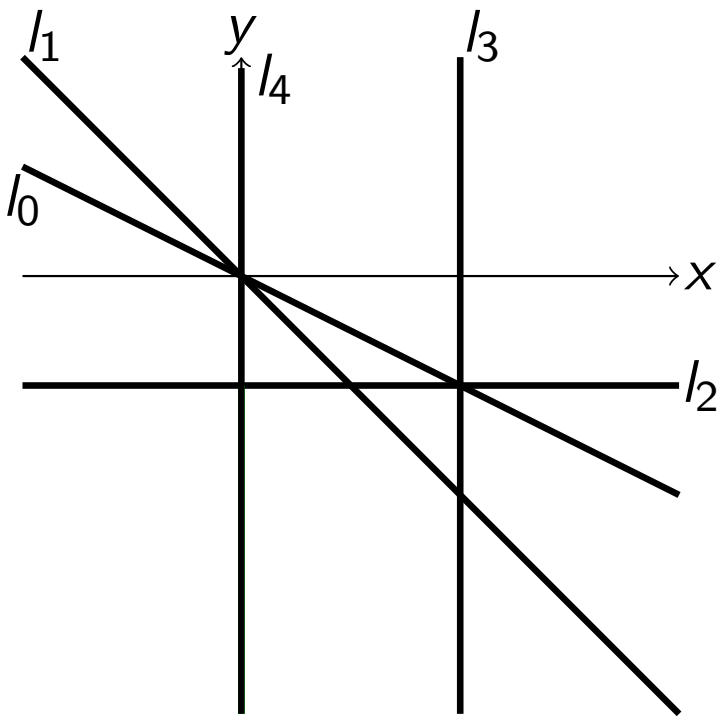
  $l_0 + 2l_1$, i.e., $\overbrace{(x < 0)}^{l_4}$.

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$ $\overbrace{(x < 0)}^{l_4}$



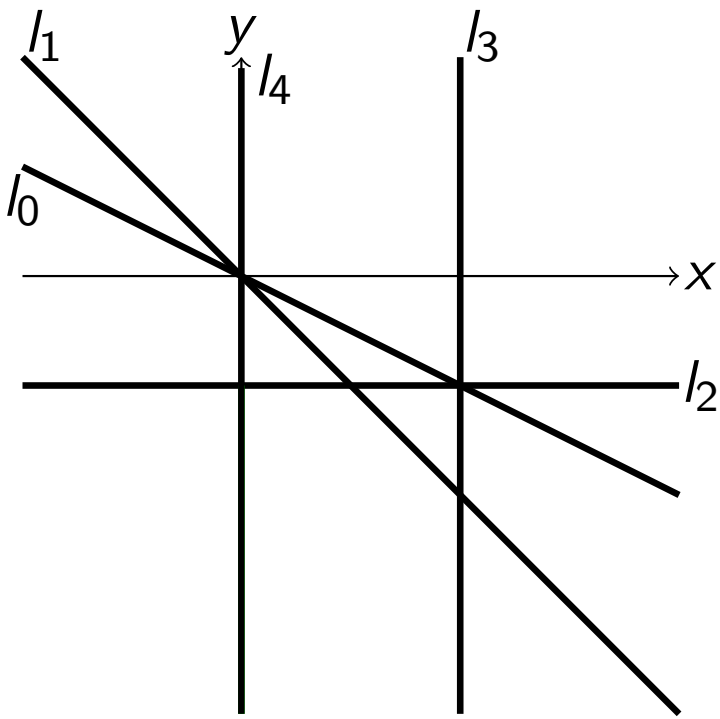

- Guess a value, e.g., $x \leftarrow 3$
  Then $l_0$ yields lower bound $y > -\frac{3}{2}$
  and $l_1$ yields upper bound $y < -3$
- Clash of bounds suggests inferring
  $l_0 + 2l_1$, i.e., $\overbrace{(x < 0)}^{l_4}$.
- Now undo the guess but keep $l_4$.

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$ $\overbrace{(x < 0)}^{l_4}$



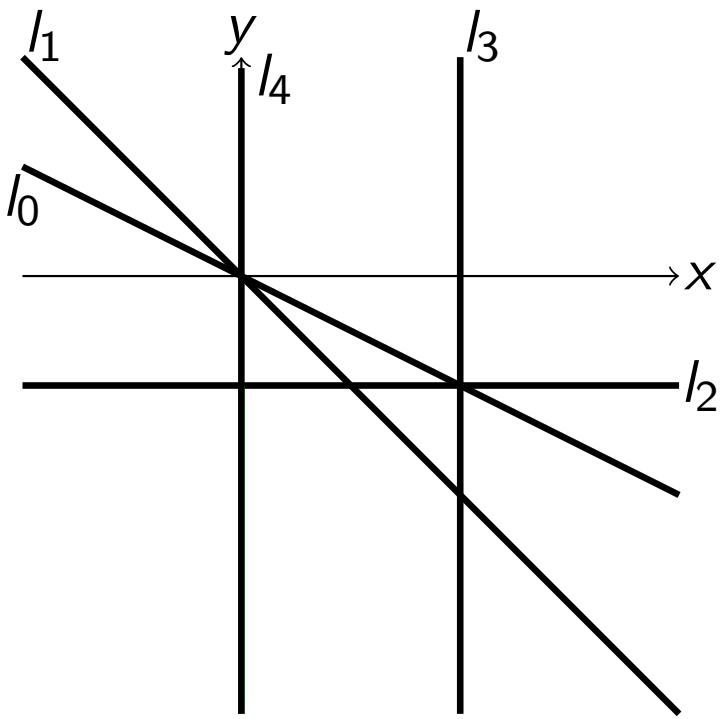

- Guess a value, e.g., $x \leftarrow 3$
  Then $l_0$ yields lower bound $y > -\frac{3}{2}$
  and $l_1$ yields upper bound $y < -3$
- Clash of bounds suggests inferring
  $l_0 + 2l_1$, i.e., $\overbrace{(x < 0)}^{l_4}$.
- Now undo the guess but keep $l_4$.
- Spot that $l_3$ and $l_4$ leave no value for $x$. Clash of bounds suggests inferring
  $l_3 + l_4$, i.e., $\overbrace{(0 < -2)}^{l_5}$.

# An example in Linear Rational Arithmetic

$\overbrace{(-2{\cdot}y - x < 0)}^{l_0}$, $\overbrace{(y + x < 0)}^{l_1}$, $\overbrace{(y < -1)}^{l_2}$

unsatisfiable in Linear Rational Arithmetic (LRA).

$\overbrace{(-x < -2)}^{l_3}$ $\overbrace{(x < 0)}^{l_4}$

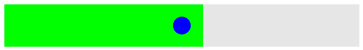



- Guess a value, e.g., $x \leftarrow 3$
  Then $l_0$ yields lower bound $y > -\frac{3}{2}$
  and $l_1$ yields upper bound $y < -3$
- Clash of bounds suggests inferring
  $l_0 + 2l_1$, i.e., $\overbrace{(x < 0)}^{l_4}$.
- Now undo the guess but keep $l_4$.
- Spot that $l_3$ and $l_4$ leave no value for $x$. Clash of bounds suggests inferring
  $l_3 + l_4$, i.e., $\overbrace{(0 < -2)}^{l_5}$.
- No guess to undo. UNSAT.

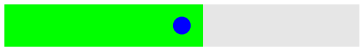
# Search phase (satisfiable case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

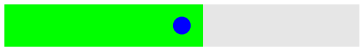
# Search phase (satisfiable case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

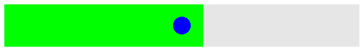
# Search phase (satisfiable case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

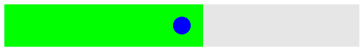
# Search phase (satisfiable case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

# Search phase (satisfiable case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

# Search phase (satisfiable case)

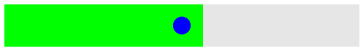
| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

SAT

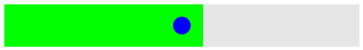
# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

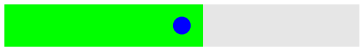
# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| $\ldots$ | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

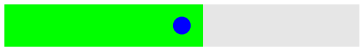
# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

# Search phase (conflict case)

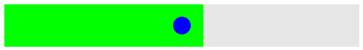
| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

Conflict

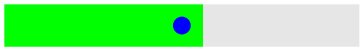
# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

Conflict

What to do now?

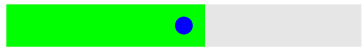
# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

Conflict

What to do now?
Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$
(i.e. try another $\Gamma'$)

# Search phase (conflict case)

| Free var within | Constraints (unit ones in red) | Feasible set | Var |
|---|---|---|---|
| $\{x_1\}$ | $C_1^1, \ldots, C_j^1, \ldots$ |  | $x_1$ |
| $\{x_1, x_2\}$ | $C_1^2, C_2^2, \ldots, C_j^2, \ldots$ |  | $x_2$ |
| $\{x_1, x_2, x_3\}$ | $C_1^3, C_2^3, \ldots, C_j^3, \ldots$ |  | $x_3$ |
| ... | | | |
| $\{x_1, \ldots, x_i\}$ | $C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$ |  | $x_i$ |

Conflict

What to do now?
Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$
(i.e. try another $\Gamma'$)

How do we avoid picking the same values (i.e. the same $\Gamma$)?
How do we avoid picking a $\Gamma'$ that fails for the same reason $\Gamma$ fails?

# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$

# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y)$

# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y)$
- detected that no value can be assigned to $y$ to extend $\mathfrak{M}$ into a model of those unit constraints: $\mathfrak{M} \not\models \exists y A$

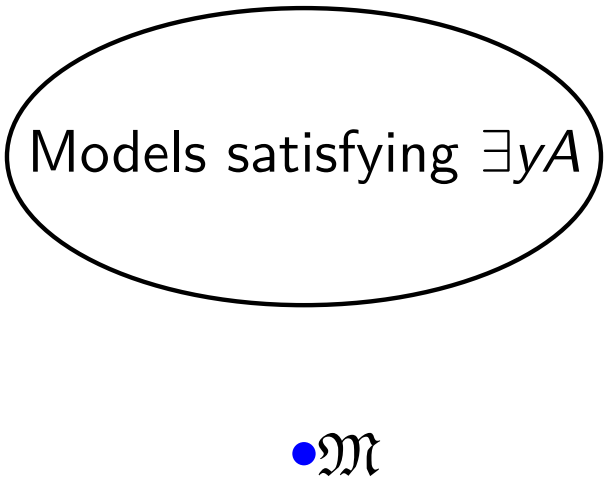
  where $\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$

# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y)$
- detected that no value can be assigned to $y$ to extend $\mathfrak{M}$ into a model of those unit constraints: $\mathfrak{M} \not\models \exists y A$

  where $\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$

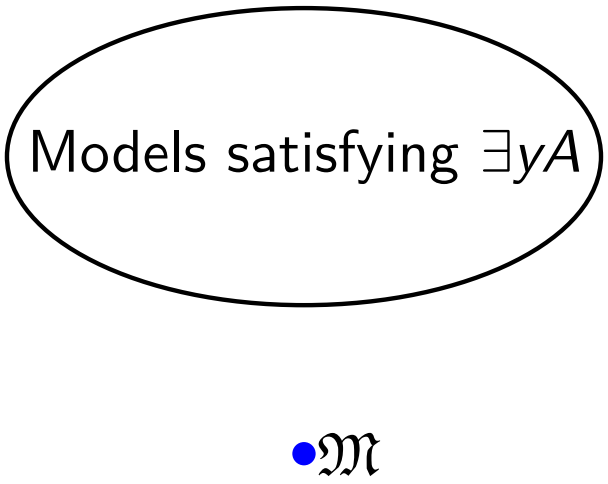


# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y)$
- detected that no value can be assigned to $y$ to extend $\mathfrak{M}$ into a model of those unit constraints: $\mathfrak{M} \not\models \exists y A$

  where $\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$

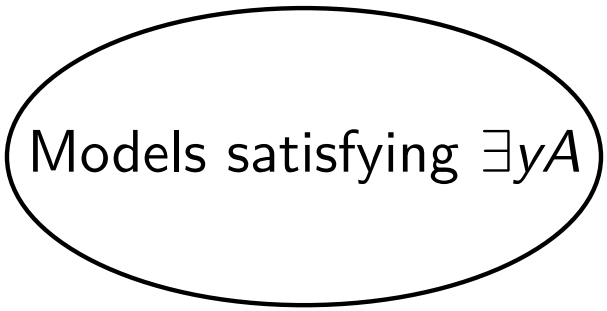


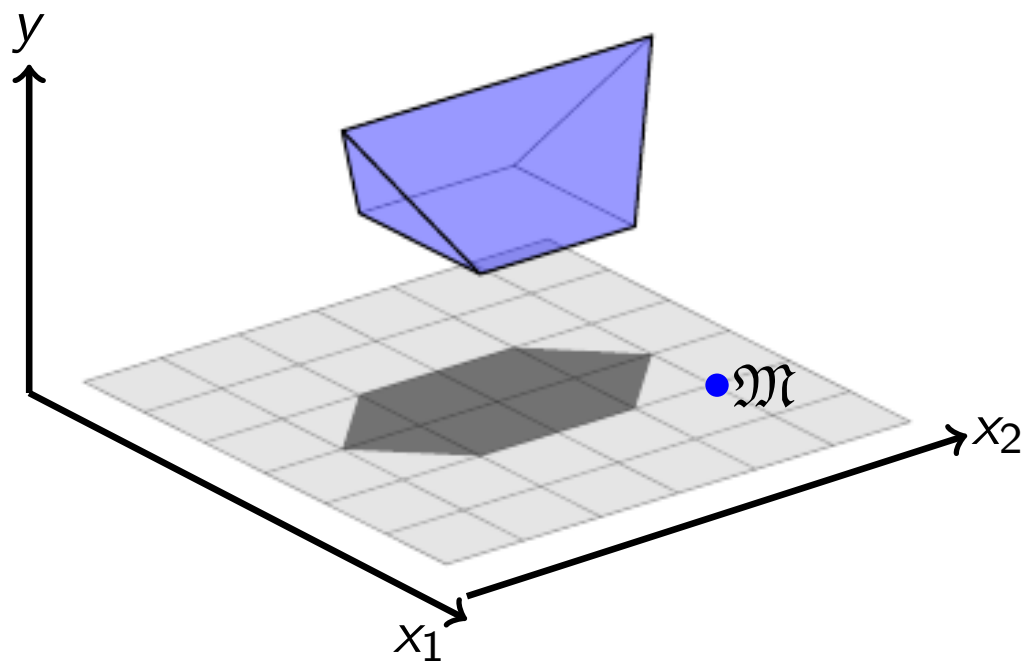
# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\vec{x}, y) \wedge \cdots \wedge l_m(\vec{x}, y)$
- detected that no value can be assigned to $y$ to extend $\mathfrak{M}$ into a model of those unit constraints: $\mathfrak{M} \not\models \exists y A$

  where $\exists y A$ is $\exists y(l_1(\vec{x}, y) \wedge \cdots \wedge l_m(\vec{x}, y))$



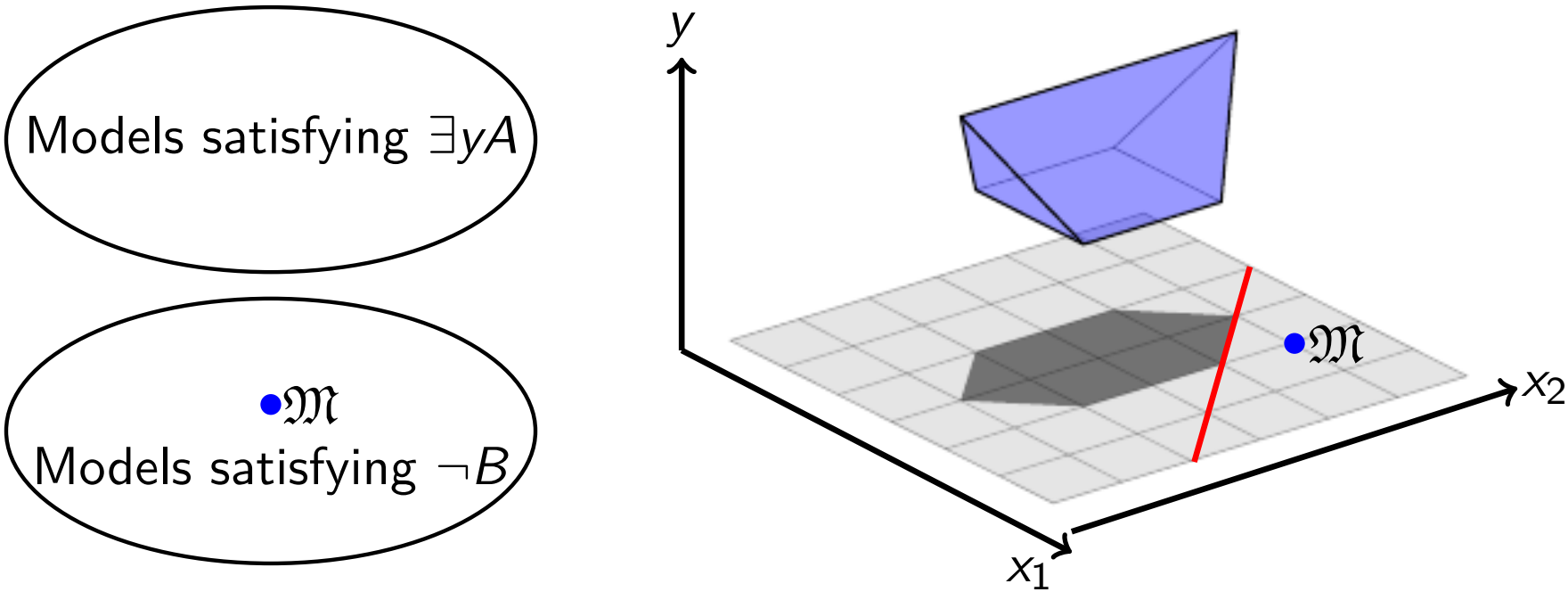

We seek to generalise $\mathfrak{M}$ into a class of models that do not satisfy $\exists y A$ "for the same reason" $\mathfrak{M}$ does not.

# Conflict analysis

In case of conflict we have

- assigned values $x_1 \mapsto v_1, \ldots, x_1 \mapsto v_n$, i.e., a model $\mathfrak{M}$
- a collection of unit constraints in $y$: $l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y)$
- detected that no value can be assigned to $y$ to extend $\mathfrak{M}$ into a model of those unit constraints: $\mathfrak{M} \not\models \exists y A$

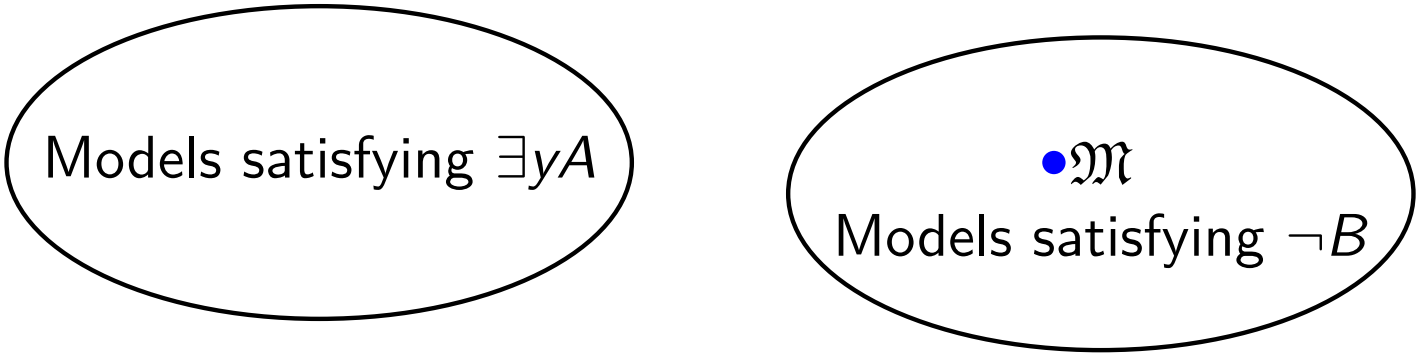
  where $\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$




We seek to generalise $\mathfrak{M}$ into a class of models that do not satisfy $\exists y A$ "for the same reason" $\mathfrak{M}$ does not.

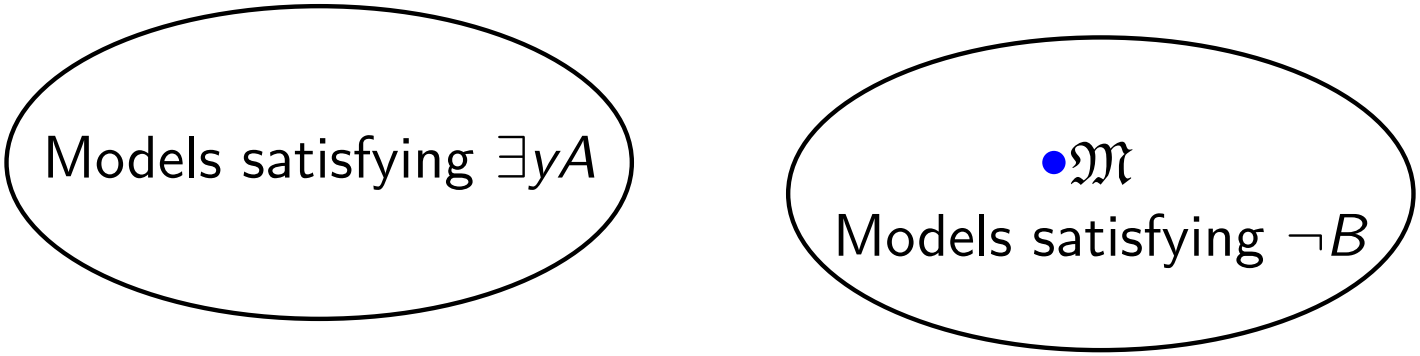
# The theory lemmas

$\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$



# The theory lemmas

$\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$



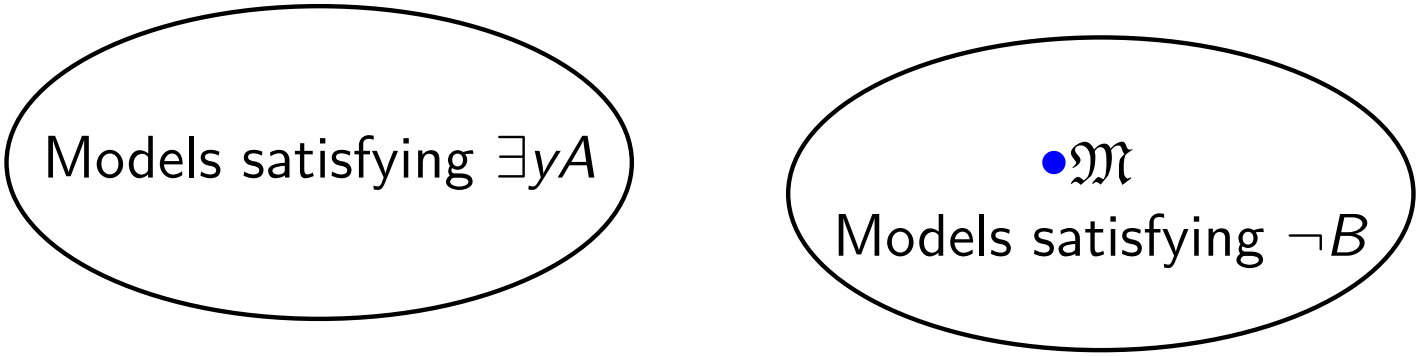

We characterise this class as those models not satisfying $B$, for some quantifier-free $B$ (with $\mathrm{fv}(B) \subseteq \{\overrightarrow{x}\}$) such that

- $\mathcal{T} \models (\exists y A) \Rightarrow B$
- $\mathfrak{M} \not\models B$

$B$ is an interpolant of $\exists y A$ at $\mathfrak{M}$.

# The theory lemmas

$\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$




We characterise this class as those models not satisfying $B$, for some quantifier-free $B$ (with $\mathrm{fv}(B) \subseteq \{\overrightarrow{x}\}$) such that

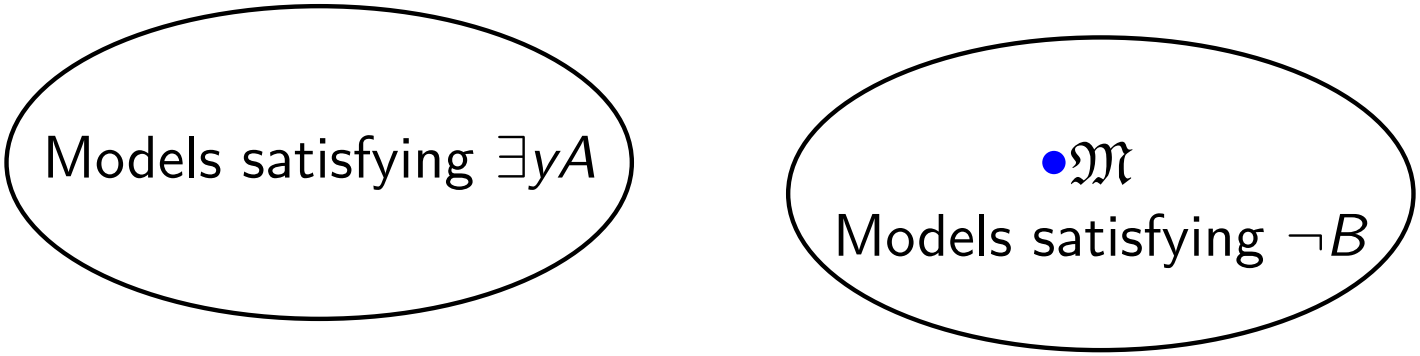
- $\mathcal{T} \models (\exists y A) \Rightarrow B$
- $\mathfrak{M} \not\models B$

$B$ is an interpolant of $\exists y A$ at $\mathfrak{M}$.

MCSAT considers the theory lemma $A \Rightarrow B$
that rules out not only $\mathfrak{M}$ but a set of similar models
(we impose that $B$ be a clause, so $A \Rightarrow B$ is a clause).

# The theory lemmas

$\exists y A$ is $\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$




We characterise this class as those models not satisfying $B$, for some quantifier-free $B$ (with $\mathrm{fv}(B) \subseteq \{\overrightarrow{x}\}$) such that

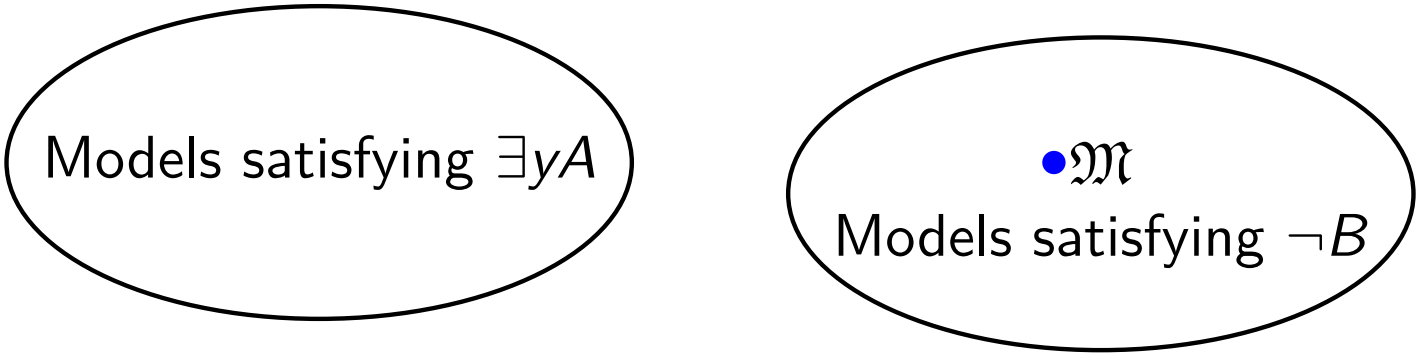
- $\mathcal{T} \models (\exists y A) \Rightarrow B$
- $\mathfrak{M} \not\models B$

$B$ is an interpolant of $\exists y A$ at $\mathfrak{M}$.

MCSAT considers the theory lemma $A \Rightarrow B$
that rules out not only $\mathfrak{M}$ but a set of similar models
(we impose that $B$ be a clause, so $A \Rightarrow B$ is a clause).
If $A$ results from Boolean reasoning,
it performs Boolean conflict analysis on $A$ (Boolean resolutions).

# The theory lemmas

$\exists y A$ is $\exists y(l_1(\vec{x}, y) \wedge \cdots \wedge l_m(\vec{x}, y))$




We characterise this class as those models not satisfying $B$, for some quantifier-free $B$ (with $\mathrm{fv}(B) \subseteq \{\vec{x}\}$) such that

- $\mathcal{T} \models (\exists y A) \Rightarrow B$
- $\mathfrak{M} \not\models B$

$B$ is an interpolant of $\exists y A$ at $\mathfrak{M}$.

MCSAT considers the theory lemma $A \Rightarrow B$
that rules out not only $\mathfrak{M}$ but a set of similar models
(we impose that $B$ be a clause, so $A \Rightarrow B$ is a clause).
If $A$ results from Boolean reasoning,
it performs Boolean conflict analysis on $A$ (Boolean resolutions).
It backtracks to a point where $A \Rightarrow B$ is no longer violated,
e.g., $B$ no longer evaluates (to false).

# MCSAT theories

Give me a theory $\mathcal{T}$ with

- a nice way of representing domains of feasible values,
  and how they are affected (i.e. reduced) by unit constraints;
- such an explanation mechanism

. . . and I'll give you an MCSAT calculus for $\mathcal{T}$,
using some adaptation of the 2-watched literals technique
for tracking unit constraints

# MCSAT theories

Give me a theory $\mathcal{T}$ with

- a nice way of representing domains of feasible values,
  and how they are affected (i.e. reduced) by unit constraints;
- such an explanation mechanism,
  producing $B$ as a clause (or $\neg B$ as a cube)

. . . and I'll give you an MCSAT calculus for $\mathcal{T}$,
using some adaptation of the 2-watched literals technique
for tracking unit constraints

# MCSAT theories

Give me a theory $\mathcal{T}$ with

- a nice way of representing domains of feasible values,
  and how they are affected (i.e. reduced) by unit constraints;
- such an explanation mechanism,
  producing $B$ as a clause (or $\neg B$ as a cube) ,
  satisfying some suitable conditions for termination;

. . . and I'll give you an MCSAT calculus for $\mathcal{T}$,
using some adaptation of the 2-watched literals technique
for tracking unit constraints

# MCSAT theories

Give me a theory $\mathcal{T}$ with

- a nice way of representing domains of feasible values,
  and how they are affected (i.e. reduced) by unit constraints;
- such an explanation mechanism,
  producing $B$ as a clause (or $\neg B$ as a cube) ,
  satisfying some suitable conditions for termination;
- optionally, a nice way to propagate a value for a variable
  whose domain has become a singleton set;

. . . and I'll give you an MCSAT calculus for $\mathcal{T}$,
using some adaptation of the 2-watched literals technique
for tracking unit constraints

# MCSAT theories

Give me a theory $\mathcal{T}$ with

- a nice way of representing domains of feasible values,
  and how they are affected (i.e. reduced) by unit constraints;
- such an explanation mechanism,
  producing $B$ as a clause (or $\neg B$ as a cube) ,
  satisfying some suitable conditions for termination;
- optionally, a nice way to propagate a value for a variable
  whose domain has become a singleton set;

. . . and I'll give you an MCSAT calculus for $\mathcal{T}$,
using some adaptation of the 2-watched literals technique
for tracking unit constraints

**In Yices**: Boolean, non-linear arithmetic, EUF (can be mixed),
. . . and now bitvectors

# On non-linear arithmetic

For producing lemma:
Where in linear arithmetic, Fourier-Motzkin resolution is used to eliminate a variable:

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

with $\lessdot_1, \lessdot_2, \lessdot_3 \in \{\leq, <\}$ such that...

# On non-linear arithmetic

For producing lemma:
Where in linear arithmetic, Fourier-Motzkin resolution is used to eliminate a variable:

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

with $\lessdot_1, \lessdot_2, \lessdot_3 \in \{\leq, <\}$ such that...

in non-linear arithmetic,
Yices uses Cylindrincal Algebraic Decomposition (CAD).

# On non-linear arithmetic

For producing lemma:
Where in linear arithmetic, Fourier-Motzkin resolution is used to eliminate a variable:

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

with $\lessdot_1, \lessdot_2, \lessdot_3 \in \{\leq, <\}$ such that...

in non-linear arithmetic,
Yices uses Cylindrincal Algebraic Decomposition (CAD).

At the SMT-comp 2019, the winner of the QF_NRA (single query track) was...

# On non-linear arithmetic

For producing lemma:
Where in linear arithmetic, Fourier-Motzkin resolution is used to eliminate a variable:

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

with $\lessdot_1, \lessdot_2, \lessdot_3 \in \{\leq, <\}$ such that...

in non-linear arithmetic,
Yices uses Cylindrincal Algebraic Decomposition (CAD).

At the SMT-comp 2019, the winner of the QF_NRA (single query track) was...
...a portfolio made of the four best 2018 solvers.
But that portfolio aside, Yices-MCSAT was best.

# 2. Application to the bitvector theory

# Bitvectors

# Bitvectors

Just like Boolean logic, bitvectors talk about a finite number of models.

# Bitvectors

Just like Boolean logic, bitvectors talk about a finite number of models.

Traditional approach to bitvectors in SMT-solving:
Bitvector formulae can be encoded into Boolean logic (one Boolean variable for each bit of each variable): bit blasting.

# Bitvectors

Just like Boolean logic, bitvectors talk about a finite number of models.

Traditional approach to bitvectors in SMT-solving:
Bitvector formulae can be encoded into Boolean logic (one Boolean variable for each bit of each variable): bit blasting.

Describing complex operations (typically $\times$) as a CNF can be expensive.

# A trivial example

```
(set-info :smt-lib-version 2.6)
(set-logic QF_BV)
(set-info :source |
We verify that (x < y) -> (x + 1 <= y)
...
|)
(set-info :status unsat)
(declare-fun x () (_ BitVec 29980))
(declare-fun y () (_ BitVec 29980))
(assert (bvult x y))
(assert (bvugt (bvadd x (_ bv1 29980)) y))
(check-sat)
(exit)
```

The best 2 solvers of the SMT-comp 2019 (which use bitblasting) cannot solve this.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)

  $x_1$

  $x_2$

  $x_3$

  $\ldots$

  $x_n$

  $\ldots$

- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)

  $x_1$

  $x_2$

  $x_3$

  $\ldots$

  $x_n$

  $\ldots$

- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)

  $x_1$

  $x_2$

  $x_3$

  $\ldots$

  $x_n$

  $\ldots$

- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)

  $x_1$

  $x_2$

  $x_3$

  $\ldots$

  $x_n$

  $\ldots$

- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)



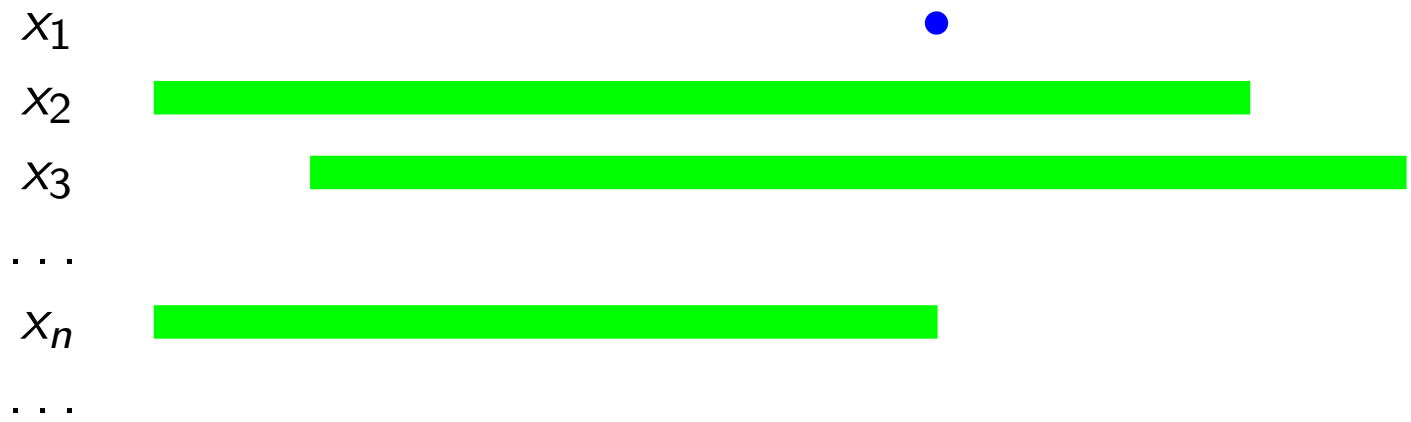

- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)




- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)




- Some proposals for producing theory lemmas explaining conflicts.

# MCSAT for bitvectors

First explored in [ZWR16] based on aggressive propagation mechanisms instead of interpolation-based conflict-explanations.

In [GLJ17] we described a conflict-driven approach to bitvectors with

- A nice way of representing domains of feasible values using Binary Decision Diagrams ([ZWR16] uses intervals+bit masks)




- Some proposals for producing theory lemmas explaining conflicts.

# Explanation mechanism for bitvectors

If $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\mathfrak{M} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$ (i.e., if $v_1, \ldots, v_n$ are the values picked for $x_1, \ldots, x_n$, and $C_1, \ldots, C_m$ are the constraints that leave no feasible values for $y$)

- **Naive explanation mechanism**: Take

  $\neg B \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$

# Explanation mechanism for bitvectors

If $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\mathfrak{M} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$
(i.e., if $v_1, \ldots, v_n$ are the values picked for $x_1, \ldots, x_n$, and
$C_1, \ldots, C_m$ are the constraints that leave no feasible values for $y$)

- **Naive explanation mechanism**: Take
  $\neg B \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out $\mathfrak{M}$)

# Explanation mechanism for bitvectors

If $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\mathfrak{M} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$
(i.e., if $v_1, \ldots, v_n$ are the values picked for $x_1, \ldots, x_n$, and
$C_1, \ldots, C_m$ are the constraints that leave no feasible values for $y$)

- **Naive explanation mechanism**: Take
  $\neg B \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out $\mathfrak{M}$)
- **Default explanation mechanism**:
  Bitblast the unsat formula $C_1 \wedge \cdots \wedge C_m \wedge x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$
  get an *unsat core* identifying the bits of $x_1, \ldots, x_n$ that mattered

# Explanation mechanism for bitvectors

If $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\mathfrak{M} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$ (i.e., if $v_1, \ldots, v_n$ are the values picked for $x_1, \ldots, x_n$, and $C_1, \ldots, C_m$ are the constraints that leave no feasible values for $y$)

- **Naive explanation mechanism**: Take
  $\neg B \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out $\mathfrak{M}$)
- **Default explanation mechanism**:
  Bitblast the unsat formula $C_1 \wedge \cdots \wedge C_m \wedge x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$
  get an *unsat core* identifying the bits of $x_1, \ldots, x_n$ that mattered
  Drawbacks:
  - 1-bit extracts are re-injected into the problem via the explanation, for the rest of the run

# Explanation mechanism for bitvectors

If $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\mathfrak{M} = \{x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n\}$ (i.e., if $v_1, \ldots, v_n$ are the values picked for $x_1, \ldots, x_n$, and $C_1, \ldots, C_m$ are the constraints that leave no feasible values for $y$)

- **Naive explanation mechanism**: Take
  $\neg B \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out $\mathfrak{M}$)
- **Default explanation mechanism**:
  Bitblast the unsat formula $C_1 \wedge \cdots \wedge C_m \wedge x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ get an *unsat core* identifying the bits of $x_1, \ldots, x_n$ that mattered
  Drawbacks:
  - 1-bit extracts are re-injected into the problem via the explanation, for the rest of the run
  - explanations close to the bit level do not give understandable explanations of what was wrong about $\mathfrak{M}$; we'd rather have explanations close to the word level

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice explanation mechanisms
(otherwise we resort to bitblasting + unsat core for an explanation)

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice explanation mechanisms
(otherwise we resort to bitblasting + unsat core for an explanation)

In [GLJ17], we treated equality with concat+extract
Explanation lemma built by slicing

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice explanation mechanisms
(otherwise we resort to bitblasting + unsat core for an explanation)

In [GLJ17], we treated equality with concat+extract
Explanation lemma built by slicing

In [GLJ19], we treated (a fragment of) linear bitvector arithmetic

| | | | |
|---|---|---|---|
| Constraints (literals) | $C$ | $::=$ | $a \mid \neg a$ |
| Atoms | $a$ | $::=$ | $t_1^w \leq^s t_2^w \mid t_1^w \leq^u t_2^w \mid t_1^w \simeq t_2^w$ |
| Terms of bitwidth w | $t^w$ | $::=$ | $c_0^w + \sum_{i=1}^n c_i^w \cdot x_i^w$ |

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice explanation mechanisms
(otherwise we resort to bitblasting + unsat core for an explanation)

In [GLJ17], we treated equality with concat+extract
Explanation lemma built by slicing

In [GLJ19], we treated (a fragment of) linear bitvector arithmetic

| | | | |
|---|---|---|---|
| Constraints (literals) | $C$ | $::=$ | $a \mid \neg a$ |
| Atoms | $a$ | $::=$ | $t_1 \leq^s t_2 \mid t_1 \leq^u t_2 \mid t_1 \simeq t_2$ |
| Terms of bitwidth w | $t$ | $::=$ | $c_0 + \sum_{i=1}^{n} c_i \cdot x_i$ |

# Word-level explanations

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice explanation mechanisms
(otherwise we resort to bitblasting + unsat core for an explanation)

In [GLJ17], we treated equality with concat+extract
Explanation lemma built by slicing

In [GLJ19], we treated (a fragment of) linear bitvector arithmetic

| | | | |
|---|---|---|---|
| Constraints (literals) | $C$ | $::=$ | $a \mid \neg a$ |
| Atoms | $a$ | $::=$ | $t_1 \leq^s t_2 \mid t_1 \leq^u t_2 \mid t_1 \simeq t_2$ |
| Terms of bitwidth w | $t$ | $::=$ | $c_0 + \sum_{i=1}^{n} c_i \cdot x_i$ |

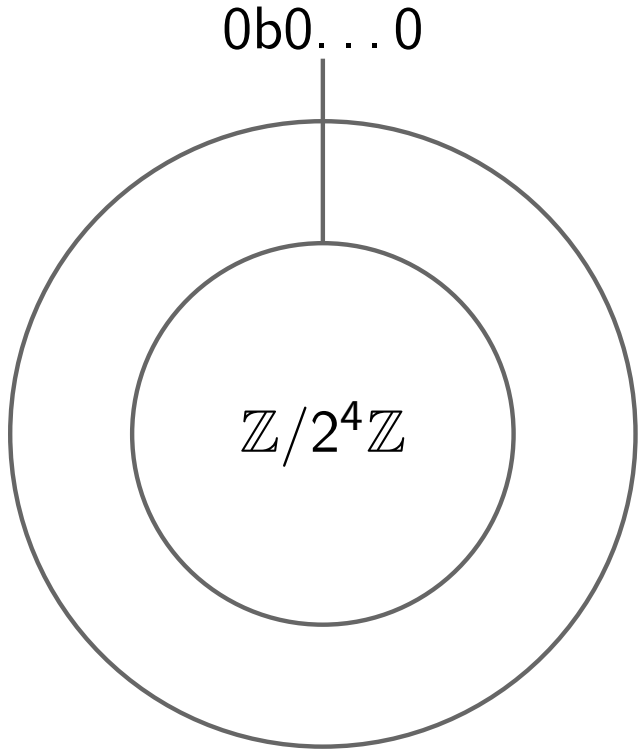
Furthermore, when interpolating $\exists y(C_1 \wedge \cdots \wedge C_m)$, for each $C$ among $C_1, \ldots, C_m$, coefficient $c$ (resp. $c'$) of $y$ in $t_1$ (resp. $t_2$) are in $\{-1, 0, 1\}$ such that $c \cdot c' \in \{0, 1\}$.

# Concrete example

$\mathfrak{M} = \{x_1 \leftarrow 0b1100, x_2 \leftarrow 0b1101, x_3 \leftarrow 0b0000\}$

| | | | | |
|---|---|---|---|---|
| $C_1$ is | $\neg(y \simeq x_1)$ | $I_1$ is | $[x_1; x_1 + 1[$ | as $(0 \not\simeq -1)$ |
| $C_2$ is | $(x_1 \leq^u x_3 + y)$ | $I_2$ is | $[-x_3; x_1 - x_3[$ | as $(x_1 \not\simeq 0)$ |
| $C_3$ is | $\neg(y - x_2 \leq^u x_3 + y)$ | $I_3$ is | $[x_2; -x_3[$ | as $(-x_2 \not\simeq x_3)$ |

Space of values for $y$ (feasible ones in white, forbidden ones in red):

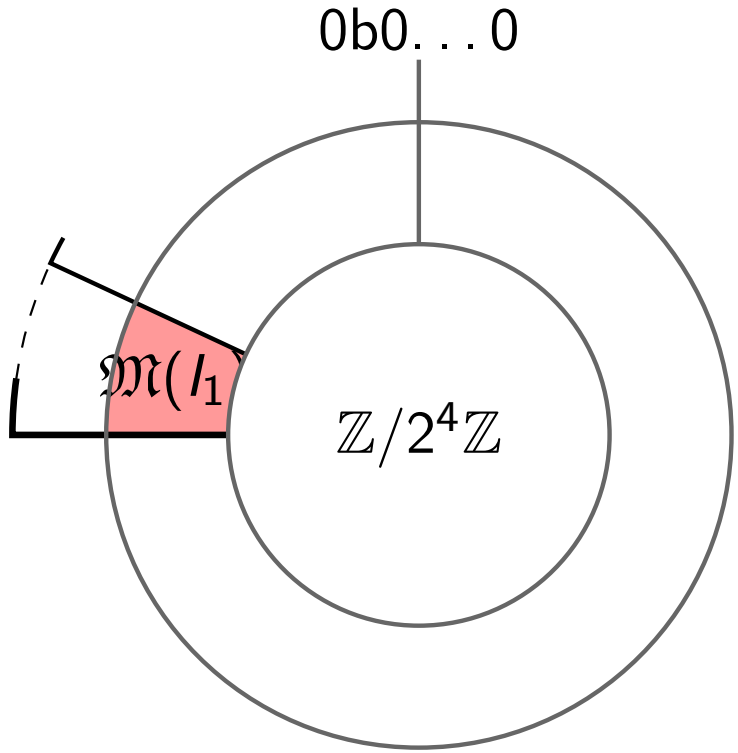


# Concrete example

$\mathfrak{M} = \{x_1 \leftarrow 0b1100, x_2 \leftarrow 0b1101, x_3 \leftarrow 0b0000\}$

| | | | | |
|---|---|---|---|---|
| $C_1$ is | $\neg(y \simeq x_1)$ | $I_1$ is | $[x_1; x_1 + 1[$ | as $(0 \not\simeq -1)$ |
| $C_2$ is | $(x_1 \leq^u x_3 + y)$ | $I_2$ is | $[-x_3; x_1 - x_3[$ | as $(x_1 \not\simeq 0)$ |
| $C_3$ is | $\neg(y - x_2 \leq^u x_3 + y)$ | $I_3$ is | $[x_2; -x_3[$ | as $(-x_2 \not\simeq x_3)$ |

Space of values for $y$ (feasible ones in white, forbidden ones in red):

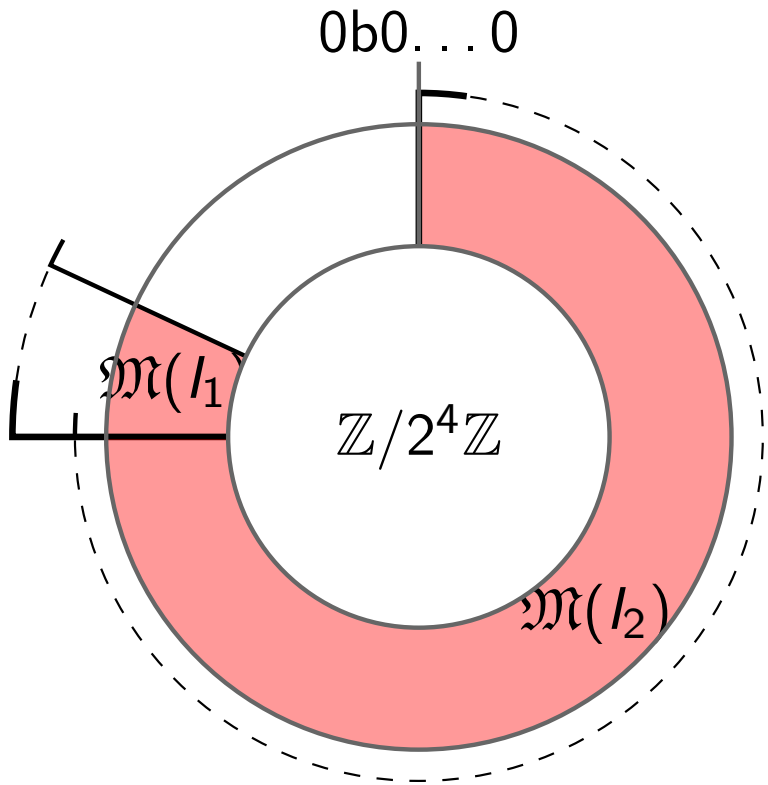


# Concrete example

$\mathfrak{M} = \{x_1 \leftarrow 0b1100, x_2 \leftarrow 0b1101, x_3 \leftarrow 0b0000\}$

| | | | | |
|---|---|---|---|---|
| $C_1$ is | $\neg(y \simeq x_1)$ | $I_1$ is | $[x_1; x_1 + 1[$ | as $(0 \not\simeq -1)$ |
| $C_2$ is | $(x_1 \leq^u x_3 + y)$ | $I_2$ is | $[-x_3; x_1 - x_3[$ | as $(x_1 \not\simeq 0)$ |
| $C_3$ is | $\neg(y - x_2 \leq^u x_3 + y)$ | $I_3$ is | $[x_2; -x_3[$ | as $(-x_2 \not\simeq x_3)$ |

Space of values for $y$ (feasible ones in white, forbidden ones in red):

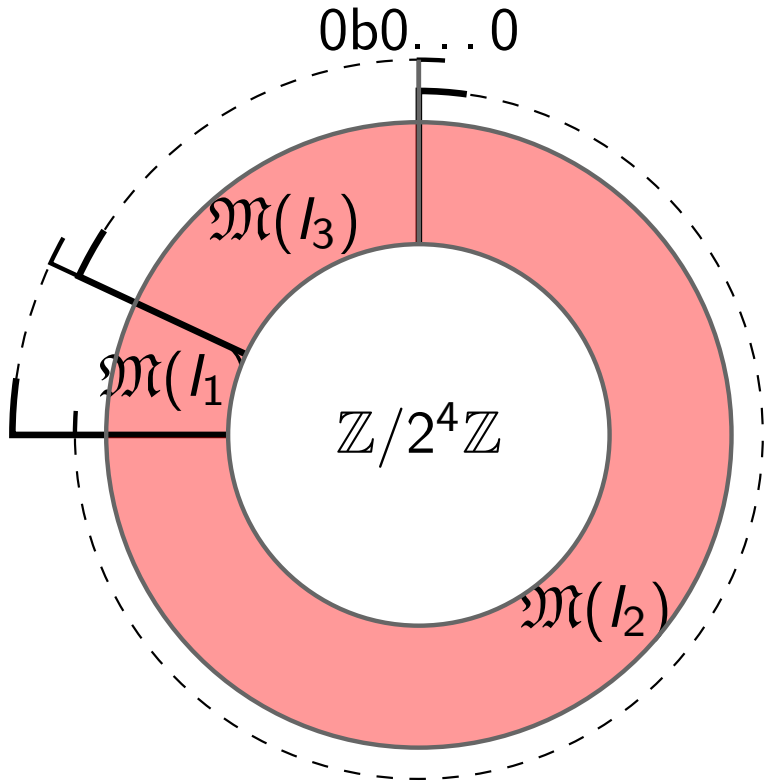


# Concrete example

$\mathfrak{M} = \{x_1 \leftarrow 0b1100, x_2 \leftarrow 0b1101, x_3 \leftarrow 0b0000\}$

| | | | | |
|---|---|---|---|---|
| $C_1$ is | $\neg(y \simeq x_1)$ | $I_1$ is | $[x_1; x_1 + 1[$ | as $(0 \not\simeq -1)$ |
| $C_2$ is | $(x_1 \leq^u x_3 + y)$ | $I_2$ is | $[-x_3; x_1 - x_3[$ | as $(x_1 \not\simeq 0)$ |
| $C_3$ is | $\neg(y - x_2 \leq^u x_3 + y)$ | $I_3$ is | $[x_2; -x_3[$ | as $(-x_2 \not\simeq x_3)$ |

Space of values for $y$ (feasible ones in white, forbidden ones in red):

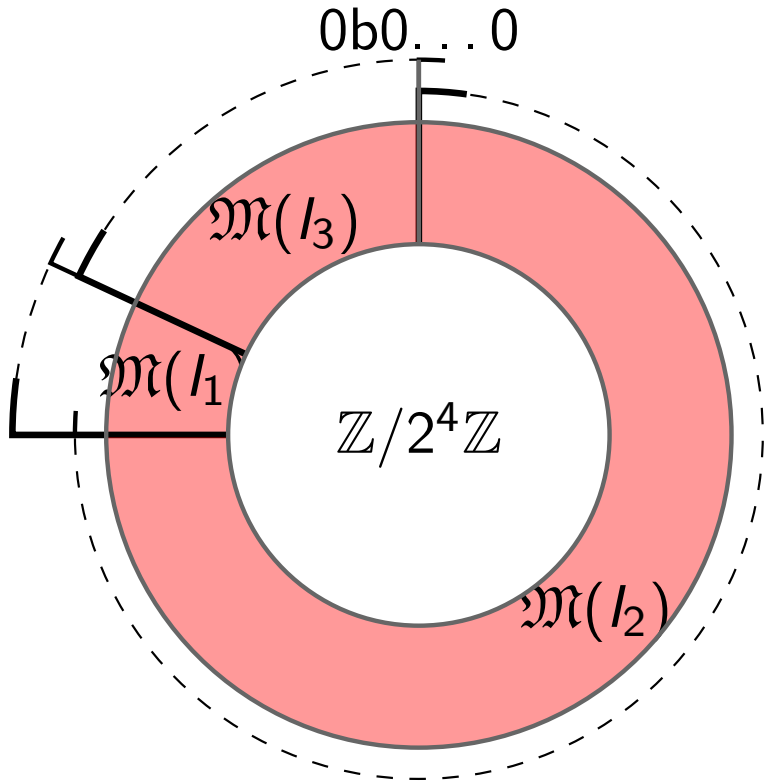


# Concrete example

$\mathfrak{M} = \{x_1 \leftarrow 0b1100, x_2 \leftarrow 0b1101, x_3 \leftarrow 0b0000\}$

| | | | | |
|---|---|---|---|---|
| $C_1$ is | $\neg(y \simeq x_1)$ | $I_1$ is | $[x_1; x_1 + 1[$ | as $(0 \not\simeq -1)$ |
| $C_2$ is | $(x_1 \leq^u x_3 + y)$ | $I_2$ is | $[-x_3; x_1 - x_3[$ | as $(x_1 \not\simeq 0)$ |
| $C_3$ is | $\neg(y - x_2 \leq^u x_3 + y)$ | $I_3$ is | $[x_2; -x_3[$ | as $(-x_2 \not\simeq x_3)$ |

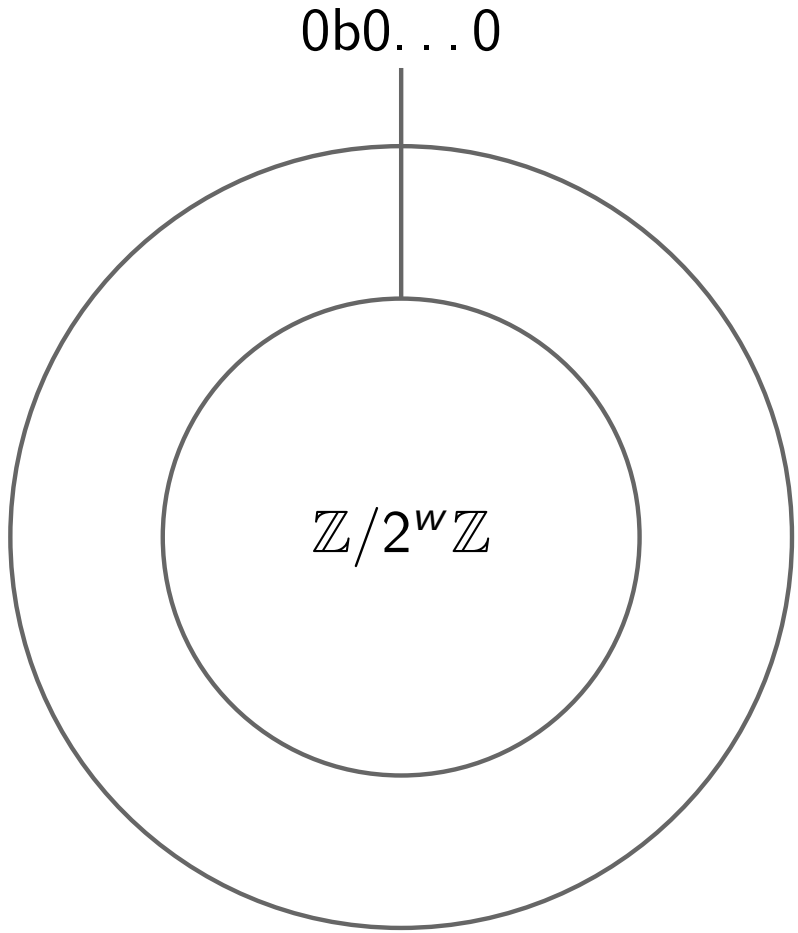
Space of values for $y$ (feasible ones in white, forbidden ones in red):




The explanation is $(x_1+1) \in I_3 \ \wedge (-x_3) \in I_2 \wedge (x_1 - x_3) \in I_1$

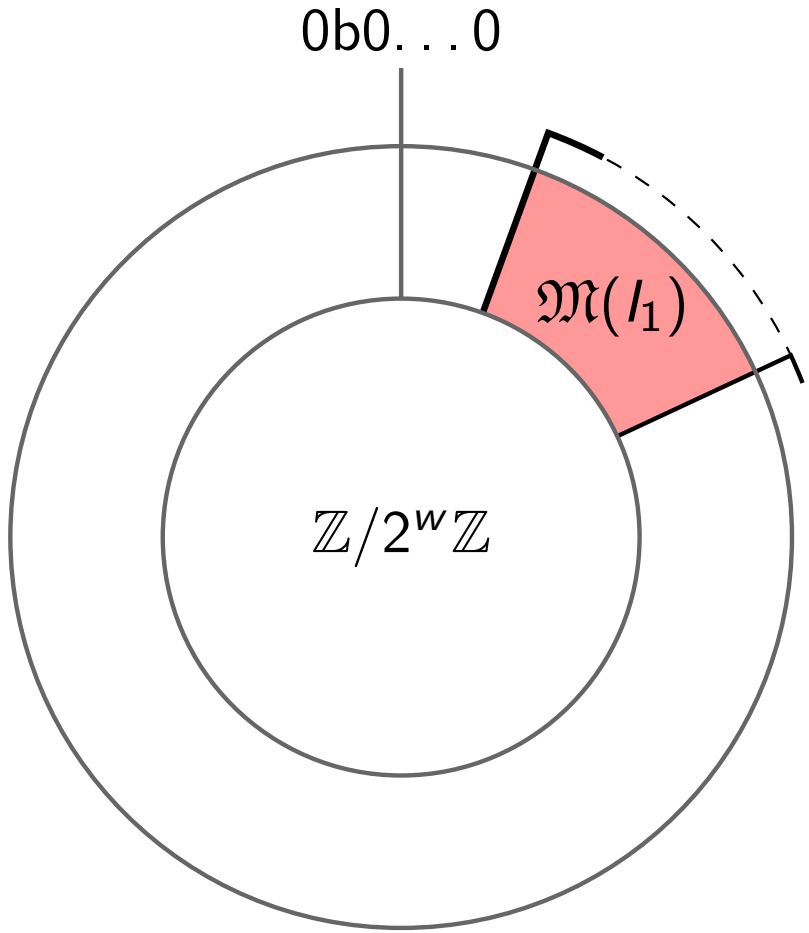
# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$



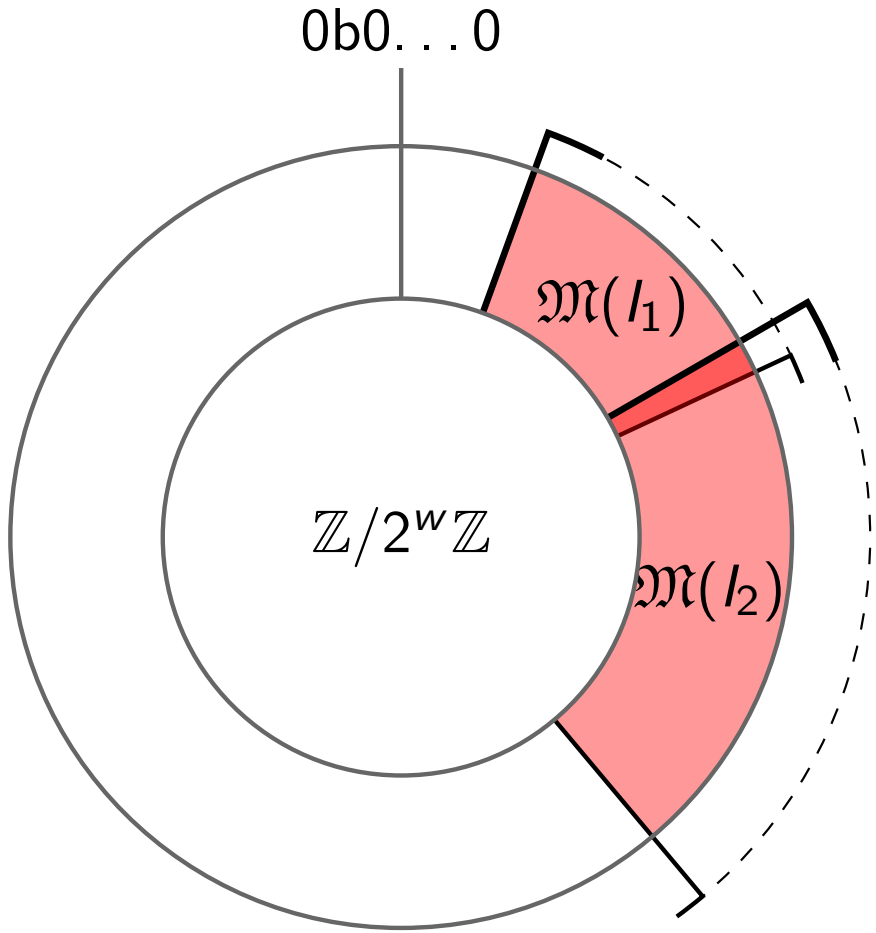
# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$



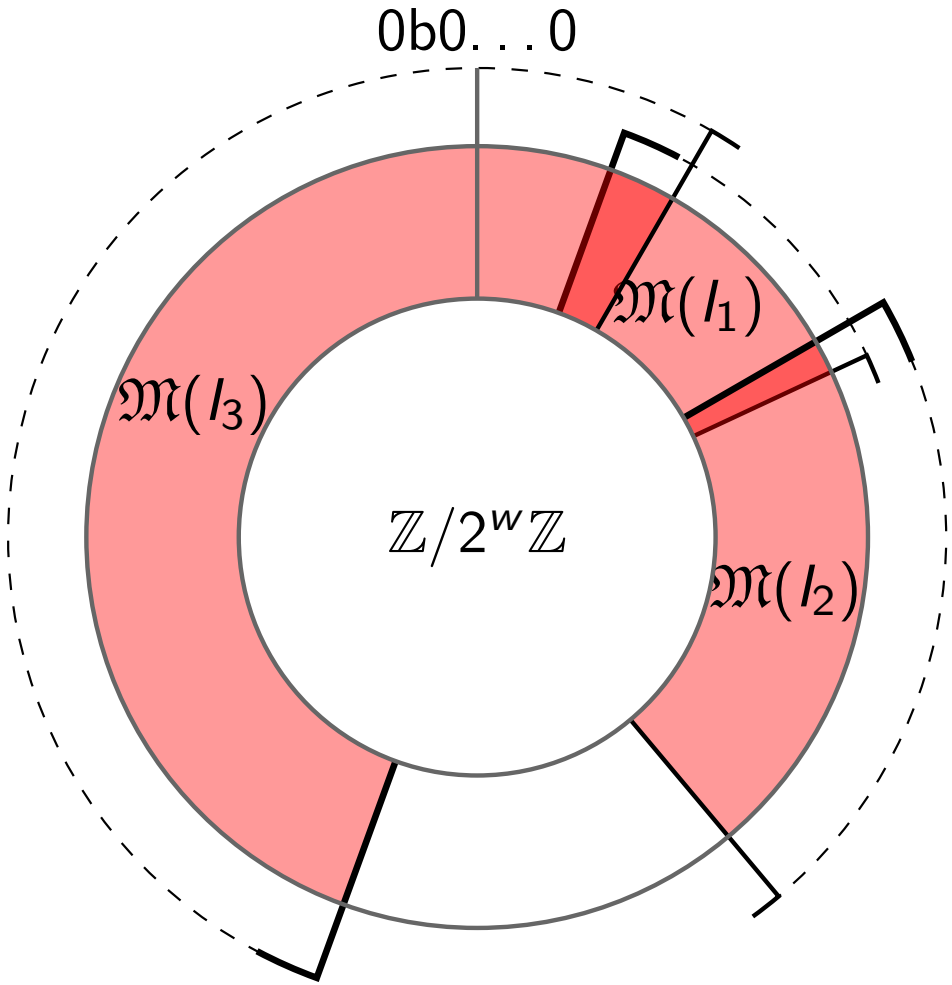
# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$



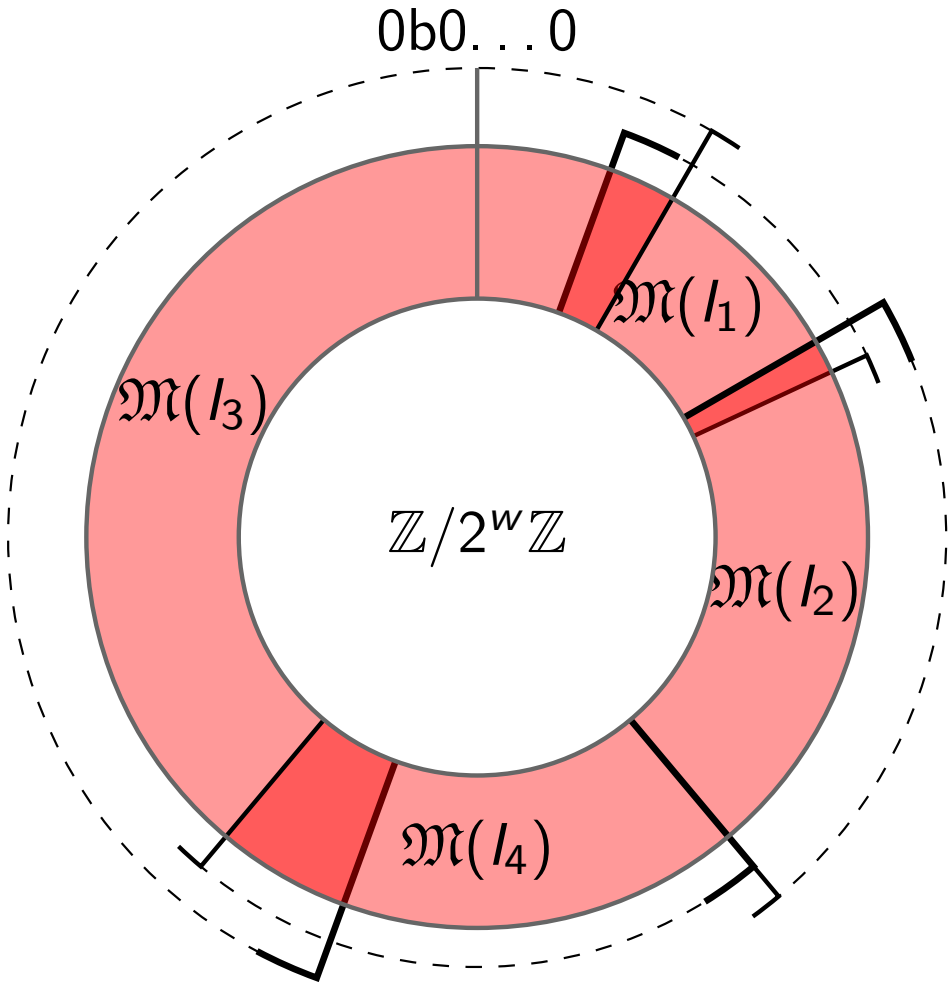
# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$



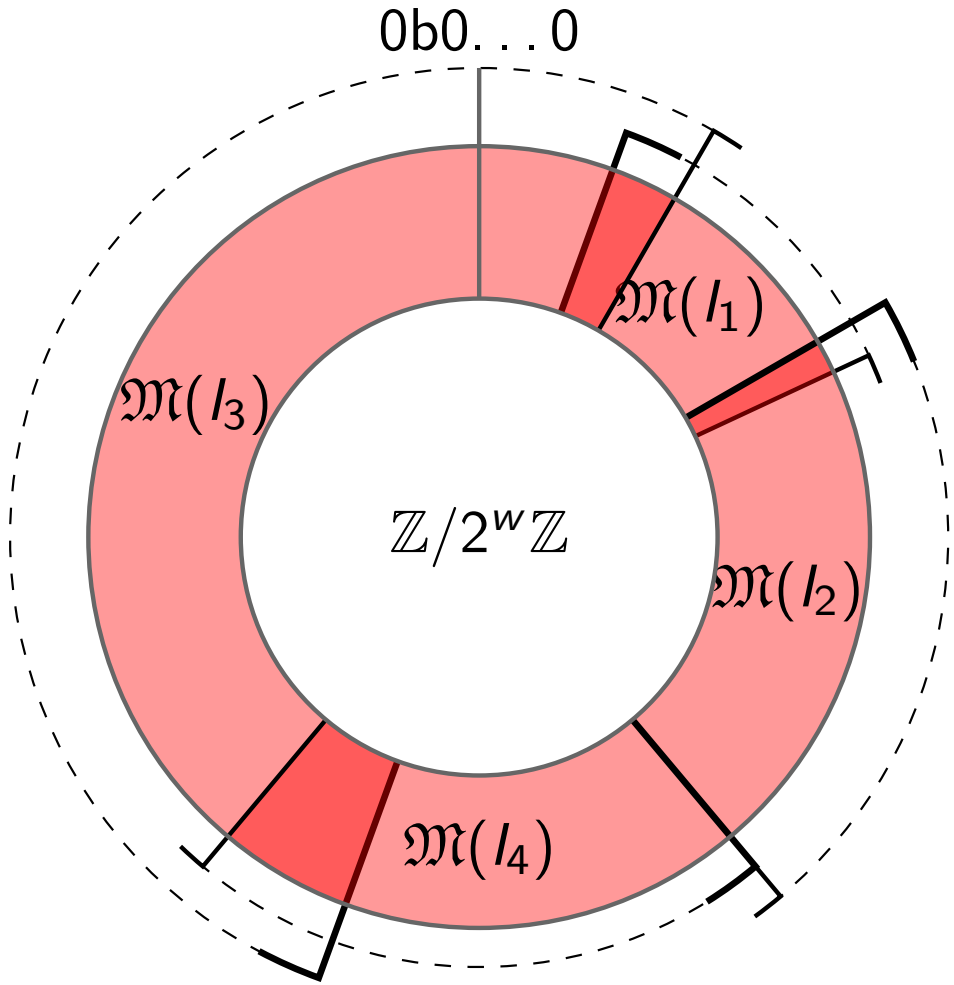
# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$



# More generally on bitwidth $w$

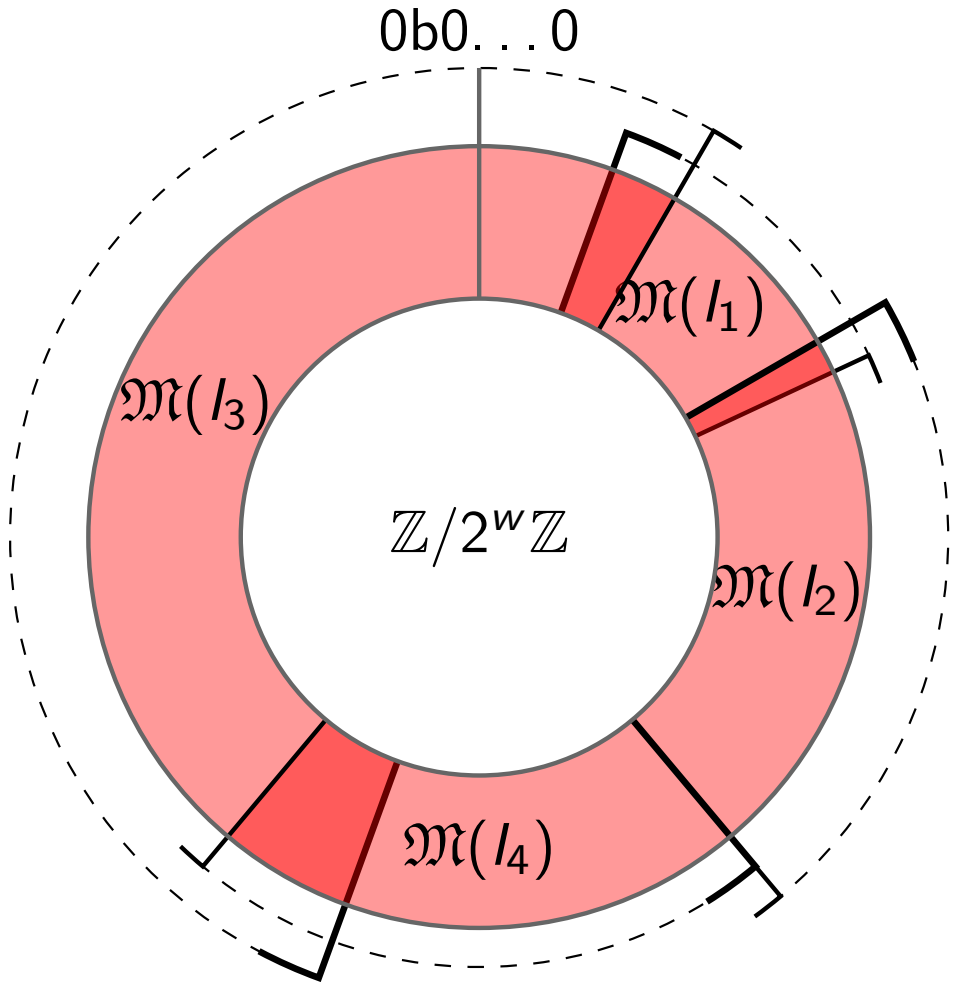
Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$




All values in $\mathbb{Z}/2^w\mathbb{Z}$ end up being forbidden because:
$\mathfrak{M}(u_1) \in \mathfrak{M}(I_2)$ and $\mathfrak{M}(u_2) \in \mathfrak{M}(I_4)$ and $\mathfrak{M}(u_4) \in \mathfrak{M}(I_3)$ and $\mathfrak{M}(u_3) \in \mathfrak{M}(I_1)$

# More generally on bitwidth $w$

Each constraint $C_i$ forbids an interval $I_i$ with interpretation $\mathfrak{M}(I_i)$




All values in $\mathbb{Z}/2^w\mathbb{Z}$ end up being forbidden because:

$\mathfrak{M}(u_1) \in \mathfrak{M}(I_2)$ and $\mathfrak{M}(u_2) \in \mathfrak{M}(I_4)$ and $\mathfrak{M}(u_4) \in \mathfrak{M}(I_3)$ and $\mathfrak{M}(u_3) \in \mathfrak{M}(I_1)$

The explanation is $(u_1 \in I_2) \wedge (u_2 \in I_4) \wedge (u_4 \in I_3) \wedge (u_3 \in I_1)$

# Things to do in practice

- For each constraint $C_i$, compute the forbidden interval $I_i$ (there are finitely many cases to consider)

# Things to do in practice

- For each constraint $C_i$, compute the forbidden interval $I_i$ (there are finitely many cases to consider)
- From the set $\{I_1, \ldots, I_m\}$ of intervals corresponding to constraints $C_1, \ldots, C_m$,
  extract a sequence $I_{\pi(1)}, \ldots, I_{\pi(q)}$ covering $\mathbb{Z}/2^w\mathbb{Z}$ in model $\mathfrak{M}$, two consecutive intervals being hooked together.

# Things to do in practice

- For each constraint $C_i$, compute the forbidden interval $I_i$ (there are finitely many cases to consider)
- From the set $\{I_1, \ldots, I_m\}$ of intervals corresponding to constraints $C_1, \ldots, C_m$,
  extract a sequence $I_{\pi(1)}, \ldots, I_{\pi(q)}$ covering $\mathbb{Z}/2^w\mathbb{Z}$ in model $\mathfrak{M}$, two consecutive intervals being hooked together.
  In the example, the sequence is $I_1, I_3, I_2$

# Things to do in practice

- For each constraint $C_i$, compute the forbidden interval $I_i$ (there are finitely many cases to consider)
- From the set $\{I_1, \ldots, I_m\}$ of intervals corresponding to constraints $C_1, \ldots, C_m$,
  extract a sequence $I_{\pi(1)}, \ldots, I_{\pi(q)}$ covering $\mathbb{Z}/2^w\mathbb{Z}$ in model $\mathfrak{M}$, two consecutive intervals being hooked together.
  In the example, the sequence is $I_1, I_3, I_2$
- Express constraints "$a \in [l; u[$" in the language of linear bv-arithmetic: $a - l <^u u - l$

# Things to do in practice

- For each constraint $C_i$, compute the forbidden interval $I_i$ (there are finitely many cases to consider)
- From the set $\{I_1, \ldots, I_m\}$ of intervals corresponding to constraints $C_1, \ldots, C_m$,
  extract a sequence $I_{\pi(1)}, \ldots, I_{\pi(q)}$ covering $\mathbb{Z}/2^w\mathbb{Z}$ in model $\mathfrak{M}$, two consecutive intervals being hooked together.
  In the example, the sequence is $I_1, I_3, I_2$
- Express constraints "$a \in [l; u[$" in the language of linear bv-arithmetic: $a{-}l <^u u{-}l$
  In the example, the formula $\mathcal{C}$,
  namely $(x_1{+}1) \in I_3 \ \wedge (-x_3) \in I_2 \wedge (x_1 - x_3) \in I_1$, becomes

$$(x_1{+}1{-}x_2 <^u -x_3{-}x_2) \wedge (0 <^u x_1) \wedge (-x_3 <^u 1)$$

# Producing the forbidden intervals: preprocessing

- Step 1: only look at $\leq^u$, expressing $\leq^s$ and $\simeq$ in terms of $\leq^u$:

$$t_1 \leq^s t_2 \quad \rightsquigarrow \quad t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1}$$
$$t_1 \simeq t_2 \quad \rightsquigarrow \quad t_1 - t_2 \leq^u 0$$

# Producing the forbidden intervals: preprocessing

- Step 1: only look at $\leq^u$, expressing $\leq^s$ and $\simeq$ in terms of $\leq^u$:

$$t_1 \leq^s t_2 \quad \rightsquigarrow \quad t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1}$$
$$t_1 \simeq t_2 \quad \rightsquigarrow \quad t_1 - t_2 \leq^u 0$$

- Step 2: assume the coefficients of $y$ are positive; otherwise:

$$t_1 \leq^u t_2 \quad \rightsquigarrow \quad -(t_2 + 1) \leq^u -(t_1 + 1)$$

# Producing the forbidden intervals: preprocessing

- Step 1: only look at $\leq^u$, expressing $\leq^s$ and $\simeq$ in terms of $\leq^u$:

$$t_1 \leq^s t_2 \quad \rightsquigarrow \quad t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1}$$
$$t_1 \simeq t_2 \quad \rightsquigarrow \quad t_1 - t_2 \leq^u 0$$

- Step 2: assume the coefficients of $y$ are positive; otherwise:

$$t_1 \leq^u t_2 \quad \rightsquigarrow \quad -(t_2 + 1) \leq^u -(t_1 + 1)$$

Then in order to compute a forbidden interval $I$ from a constraint $t_1 \leq^u t_2$ where $y$ has positive coefficients, we took inspiration from [JW16], but working with intervals modulo $2^w$.

# Producing the forbidden intervals

Coefficients of $y$ in $C$ are in $\{0, 1\}$ leaves 4 cases:

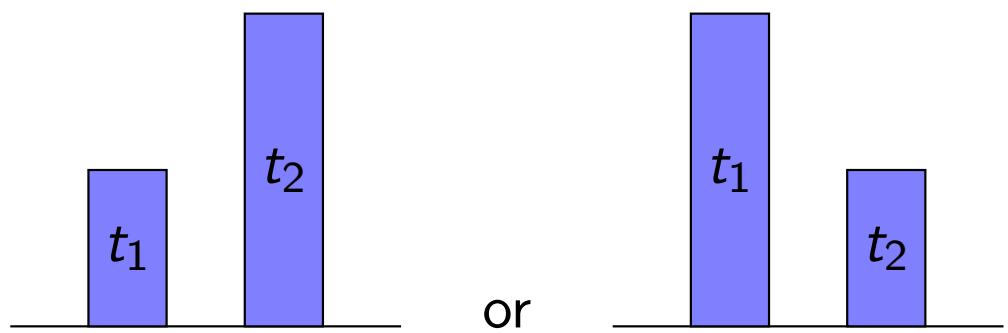
| **Normalised atom** a | **Forbidden interval that** $a$ **(resp.** $\neg a$**) specifies for** $y$ | | |
|---|---|---|---|
| | $I_a$ | $I_{\neg a}$ | **Condition** |
| $t_1 + y \leq^u t_2 + y$ | $[-t_2; -t_1[$ | $[-t_1; -t_2[$ | $t_1 \not\simeq t_2$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \simeq t_2$ |
| $t_1 \leq^u t_2 + y$ | $[-t_2; t_1 - t_2[$ | $[t_1 - t_2; -t_2[$ | $t_1 \not\simeq 0$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \simeq 0$ |
| $t_1 + y \leq^u t_2$ | $[t_2 - t_1 + 1; -t_1[$ | $[-t_1; t_2 - t_1 + 1[$ | $t_2 \not\simeq -1$ |
| | $\emptyset$ | $[0; 0[$ | $t_2 \simeq -1$ |
| $t_1 \leq^u t_2$ | $[0; 0[$ | $\emptyset$ | $t_2 <^u t_1$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \leq^u t_y 2$ |

# Producing the forbidden intervals

Coefficients of $y$ in $C$ are in $\{0, 1\}$ leaves 4 cases:

| **Normalised atom** a | **Forbidden interval that** $a$ **(resp.** $\neg a$**) specifies for** $y$ | | |
|---|---|---|---|
| | $I_a$ | $I_{\neg a}$ | **Condition** |
| $t_1 + y \leq^u t_2 + y$ | $[-t_2; -t_1[$ | $[-t_1; -t_2[$ | $t_1 \not\simeq t_2$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \simeq t_2$ |
| $t_1 \leq^u t_2 + y$ | $[-t_2; t_1 - t_2[$ | $[t_1 - t_2; -t_2[$ | $t_1 \not\simeq 0$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \simeq 0$ |
| $t_1 + y \leq^u t_2$ | $[t_2 - t_1 + 1; -t_1[$ | $[-t_1; t_2 - t_1 + 1[$ | $t_2 \not\simeq -1$ |
| | $\emptyset$ | $[0; 0[$ | $t_2 \simeq -1$ |
| $t_1 \leq^u t_2$ | $[0; 0[$ | $\emptyset$ | $t_2 <^u t_1$ |
| | $\emptyset$ | $[0; 0[$ | $t_1 \leq^u t_y 2$ |

Illustrating the first line:



# Also newly implemented in Yices

- Extraction of lower bits

  particular forms of extraction, $t[(m-1):0]$, denoted $t\langle m\rangle$,
  can be used to project $w$-bit integers to $m$-bit integers ($m \leq w$)

# Also newly implemented in Yices

- Extraction of lower bits

  particular forms of extraction, $t[(m-1):0]$, denoted $t\langle m\rangle$,
  can be used to project $w$-bit integers to $m$-bit integers ($m \leq w$)
  It behaves well with arithmetic operations:
  $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle$ $\qquad$ $(a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$

# Also newly implemented in Yices

- Extraction of lower bits

  particular forms of extraction, $t[(m-1):0]$, denoted $t\langle m\rangle$, can be used to project $w$-bit integers to $m$-bit integers ($m \leq w$)

  It behaves well with arithmetic operations:

  $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle \qquad (a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$

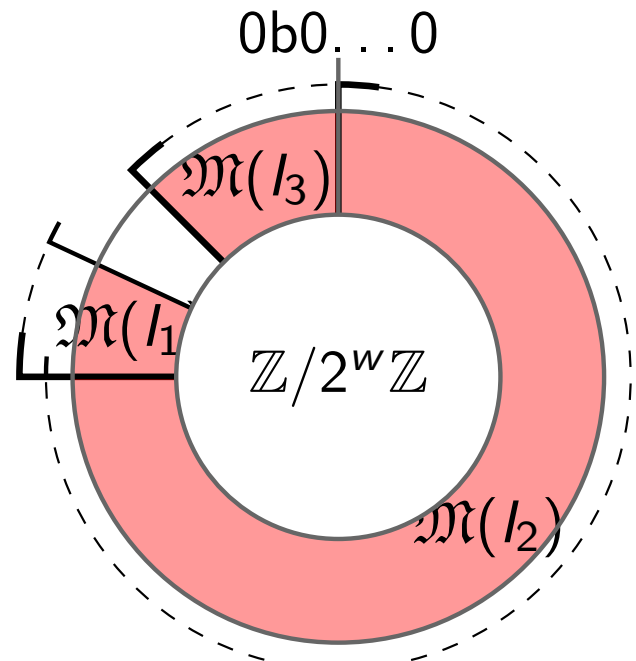
- *0-extensions* and *sign-extensions* of bitvectors

  add heavy bits to a bitvector, not changing its interpretation as an integer, can be used to inject $w$-bit integers to $m$-bit integers ($m \geq w$) and to compare terms $t_1$ and $t_2$ with $\leq^u$, $<^u$, $\simeq$ when $t_1$ and $t_2$ have different bitwidths

# Also newly implemented in Yices

- Extraction of lower bits
  particular forms of extraction, $t[(m-1):0]$, denoted $t\langle m\rangle$,
  can be used to project $w$-bit integers to $m$-bit integers ($m \leq w$)
  It behaves well with arithmetic operations:
  $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle \qquad (a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$
- *0-extensions* and *sign-extensions* of bitvectors
  add heavy bits to a bitvector, not changing its interpretation as an integer, can be used to inject $w$-bit integers to $m$-bit integers ($m \geq w$) and to compare terms $t_1$ and $t_2$ with $\leq^u$, $<^u$, $\simeq$ when $t_1$ and $t_2$ have different bitwidths
- Value propagation mechanism

In case only one value remains feasible, we can propagate the assignement of that value to $y$, with an explanation for it.
We extended MCSAT's core calculus for this and added support for it in all theory modules, including the bitvector theory:



# Experimentation

Interpolation procedure is rather insensitive to big bitwidth.
Performances are kind of predictable:
If the whole problem lies within this fragment of arithmetic. . .
. . . it performs very well!
If not, the first conflict that we bitblast shoots us in the foot for the rest of the run.

# Experimentation

Interpolation procedure is rather insensitive to big bitwidth.
Performances are kind of predictable:
If the whole problem lies within this fragment of arithmetic. . .
. . . it performs very well!
If not, the first conflict that we bitblast shoots us in the foot for the rest of the run.
Where do we go from here?

- Extend the fragment little by little, e.g., handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable $y$ are not necessarily in $\{-1, 0, 1\}$

# Experimentation

Interpolation procedure is rather insensitive to big bitwidth.
Performances are kind of predictable:
If the whole problem lies within this fragment of arithmetic. . .
. . . it performs very well!
If not, the first conflict that we bitblast shoots us in the foot for the rest of the run.
Where do we go from here?

- Extend the fragment little by little, e.g., handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable $y$ are not necessarily in $\{-1, 0, 1\}$
- Figure out how to not resort to bitblasting when the conflict does not lie 100% within 1 nicely behaved fragment.

# Experimentation

Interpolation procedure is rather insensitive to big bitwidth.
Performances are kind of predictable:
If the whole problem lies within this fragment of arithmetic. . .
. . . it performs very well!
If not, the first conflict that we bitblast shoots us in the foot for the rest of the run.
Where do we go from here?

- Extend the fragment little by little, e.g., handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable $y$ are not necessarily in $\{-1, 0, 1\}$
- Figure out how to not resort to bitblasting when the conflict does not lie 100% within 1 nicely behaved fragment.
  Each explanation mechanism is a theory reasoner
  for a subtheory of the bitvectors

# Experimentation

Interpolation procedure is rather insensitive to big bitwidth.
Performances are kind of predictable:
If the whole problem lies within this fragment of arithmetic. . .
. . . it performs very well!
If not, the first conflict that we bitblast shoots us in the foot for the rest of the run.
Where do we go from here?

- Extend the fragment little by little, e.g., handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable $y$ are not necessarily in $\{-1, 0, 1\}$
- Figure out how to not resort to bitblasting when the conflict does not lie 100% within 1 nicely behaved fragment.
  Each explanation mechanism is a theory reasoner
  for a subtheory of the bitvectors
  Sounds like a theory combination problem for MCSAT!
  . . . but they are not disjoint
  (they share the predicates "The ith bit of $x$ is 0/1")

# 3. Application to intuitionistic logic

THE JOURNAL OF SYMBOLIC LOGIC
Volume 57, Number 3, Sept. 1992

CONTRACTION-FREE SEQUENT CALCULI
FOR INTUITIONISTIC LOGIC

ROY DYCKHOFF

**§0. Prologue.** Gentzen's sequent calculus LJ, and its variants such as G3 [21], are (as is well known) convenient as a basis for automating proof search for IPC (intuitionistic propositional calculus). But a problem arises: that of detecting loops, arising from the use (in reverse) of the rule $\supset\Rightarrow$ for implication introduction on the left. We describe below an equivalent calculus, yet another variant on these systems, where the problem no longer arises: this gives a simple but effective decision procedure for IPC.

The underlying method can be traced back forty years to Vorob′ev [33], [34]. It has been rediscovered recently by several authors (the present author in August 1990, Hudelmaier [18], [19], Paulson [27], and Lincoln et al. [23]). Since the main idea is not plainly apparent in Vorob′ev's work, and there are mathematical applications [28], it is desirable to have a simple proof. We present such a proof, exploiting the Dershowtiz-Manna theorem [4] on multiset orderings.

**§1. Introduction.** Consider the task of constructing proofs in Gentzen's sequent calculus LJ of intuitionistic sequents $\Gamma \Rightarrow G$, where $\Gamma$ is a set of assumption

# SAT Modulo Intuitionistic Implications

Koen Claessen(✉) and Dan Rosén(✉)

Chalmers University of Technology, Gothenburg, Sweden
{koen,danr}@chalmers.se

**Abstract.** We present a new method for solving problems in intuitionistic propositional logic, which involves the use of an incremental SAT-solver. The method scales to very large problems, and fits well into an SMT-based framework for interaction with other theories.

## 1 Introduction

Let us take a look at *intuitionistic propositional logic*. Its syntax looks just like classical propositional logic:

$$
\begin{array}{lll}
A ::= a \mid b \mid c \mid .. \mid q & & \text{-- atoms} \\
\quad \mid A_1 \wedge A_2 & & \text{-- conjunction} \\
\quad \mid A_1 \vee A_2 & & \text{-- disjunction} \\
\quad \mid A_1 \rightarrow A_2 & & \text{-- implication} \\
\quad \mid \bot \mid \top & & \text{-- false/true}
\end{array}
$$

However, its definition of truth is considerably weaker than for classical logic. In Fig. 1, we show a Hilbert-style proof system for intuitionistic propositional

In [CR15], Claessen and Rosén showed how to decide Intuitionistic Propositional Logic (IPL) with an SMT-solving approach, using

- a SAT-solver as a black box
- a mechanism generalising (c.f., [FGGL19]) the G4ip sequent calculus rule [Dyc92]

$$\frac{A, (B \Rightarrow C), \Gamma \vdash B \quad C, \Gamma \vdash D}{((A \Rightarrow B) \Rightarrow C), \Gamma \vdash D}$$

# Working on a special form of sequents

Using a variant of the Tseitin encoding, every formula $F$ can be transformed into an (intuitionistically) equiprovable sequent $\Gamma_{\mathsf{imp}}, \Gamma_{\mathsf{flat}} \vdash d$ with

- $d$ an atom
- $\Gamma_{\mathsf{flat}}$ made of *flat clauses*: $(a_1 \wedge \cdots \wedge a_n) \Rightarrow (b_1 \vee \cdots \vee b_m)$
- $\Gamma_{\mathsf{imp}}$ made of *implication clauses*: $((a \Rightarrow b) \Rightarrow c)$

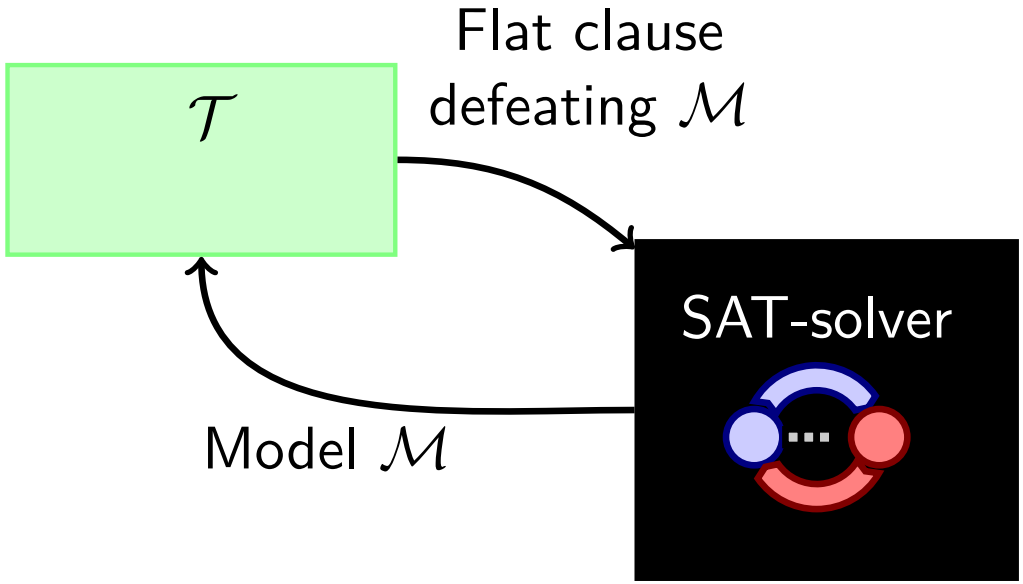
If $\Gamma_{\mathsf{imp}}$ is empty,
$\Gamma_{\mathsf{flat}} \vdash d$ is intuitionistically valid iff it is classically valid

Idea for proof-search:

- flat clauses are treated eagerly,
  to see if, by chance, $\Gamma_{\mathsf{flat}} \vdash d$ is provable,
  using e.g., a SAT-solver.
- implication clauses treated lazily,
  using the (generalised) G4ip rule.

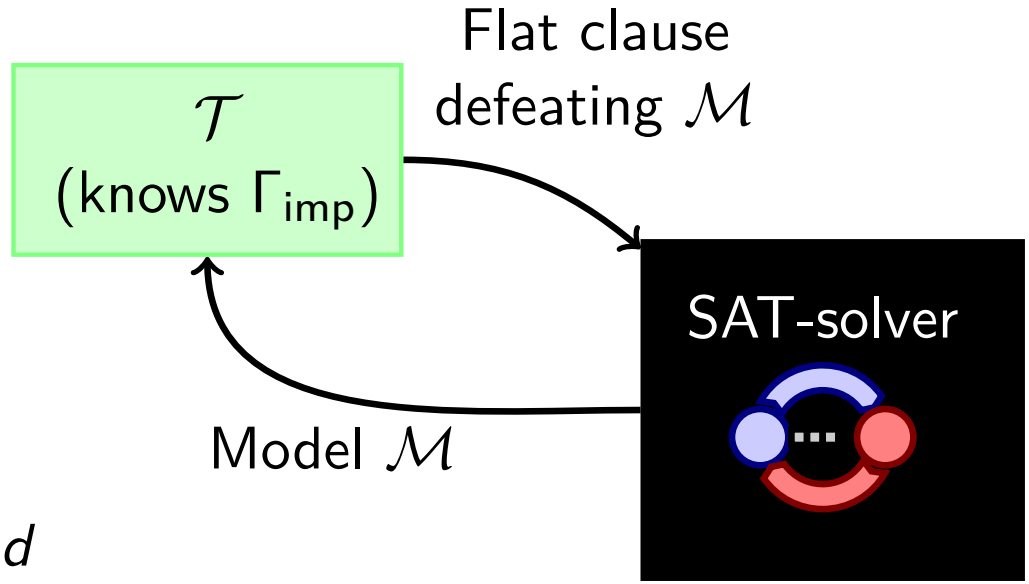
# From G4ip to SMT solving

[CR15] applies the traditional SMT scheme DPLL($\mathcal{T}$) / CDCL($\mathcal{T}$).



# From G4ip to SMT solving

[CR15] applies the traditional SMT scheme DPLL($\mathcal{T}$) / CDCL($\mathcal{T}$).
Theory $\mathcal{T}$ is that of "intuitionistic entailment"

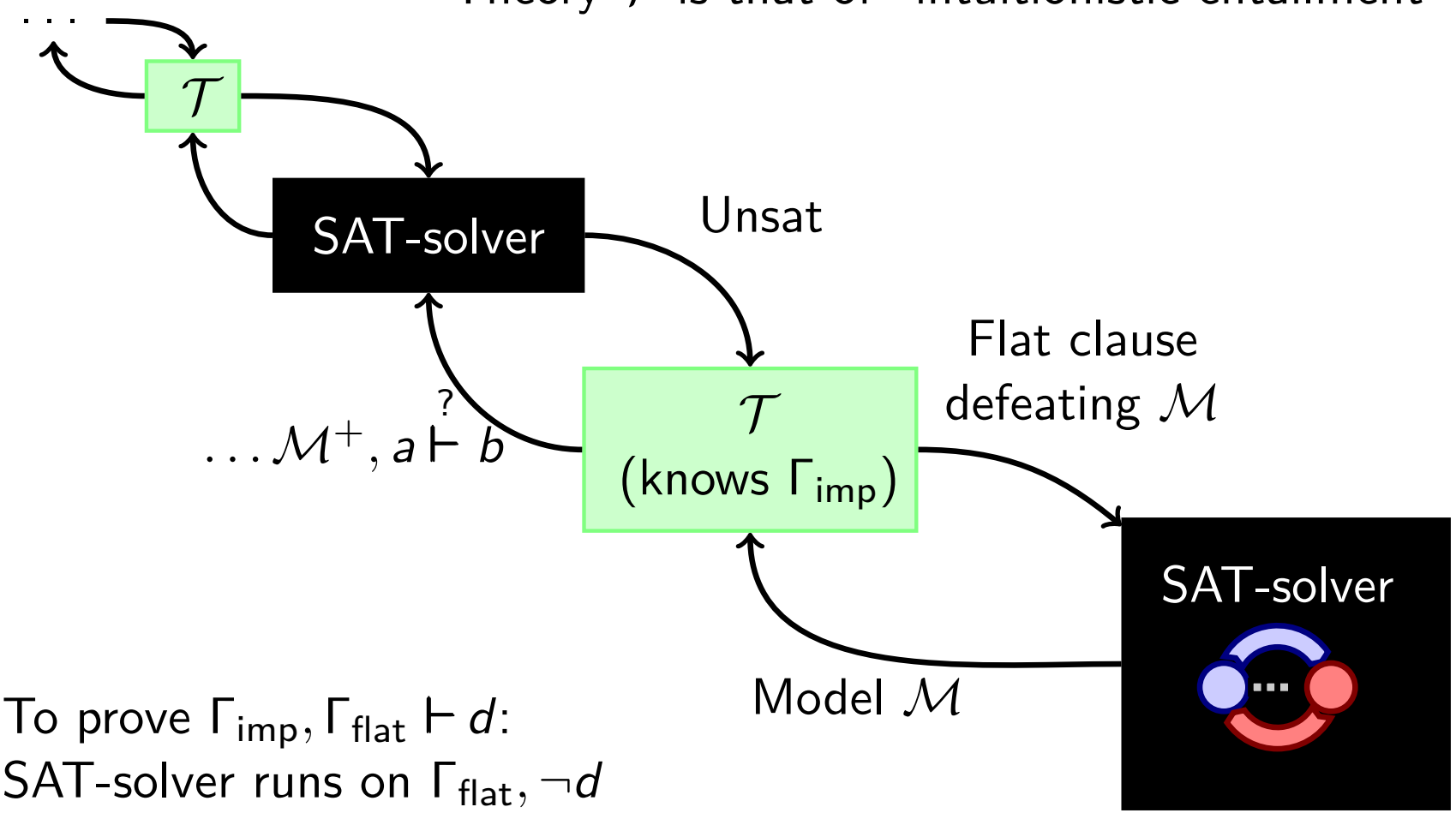



To prove $\Gamma_{\mathsf{imp}}, \Gamma_{\mathsf{flat}} \vdash d$:
SAT-solver runs on $\Gamma_{\mathsf{flat}}, \neg d$
If it returns a classical model $\mathcal{M}$:

# From G4ip to SMT solving

[CR15] applies the traditional SMT scheme DPLL($\mathcal{T}$) / CDCL($\mathcal{T}$).

Theory $\mathcal{T}$ is that of "intuitionistic entailment"




To prove $\Gamma_{\text{imp}}, \Gamma_{\text{flat}} \vdash d$:
SAT-solver runs on $\Gamma_{\text{flat}}, \neg d$
If it returns a classical model $\mathcal{M}$:
$\mathcal{T}$-reasoner picks $(a \Rightarrow b) \Rightarrow c$ in $\Gamma_{\text{imp}}$, & recursively tries to prove

$$\Gamma'_{\text{imp}}, \mathcal{M}^+, a, (b \Rightarrow c), \Gamma_{\text{flat}} \vdash b$$

# From a semantics point of view

The procedure is both a proof-constructing procedure and a model-constructing procedure, where model = Kripke structure

# From a semantics point of view

The procedure is both a proof-constructing procedure and a model-constructing procedure, where model = Kripke structure

Each recursive call, decomposing an implication clause $(a \Rightarrow b) \Rightarrow c$, is exploring a new world in the Kripke structure being constructed as a putative counter-model to the sequent, a world where $a$ is true and $b$ is false.

# From a semantics point of view

The procedure is both a proof-constructing procedure and a model-constructing procedure, where model = Kripke structure

Each recursive call, decomposing an implication clause $(a \Rightarrow b) \Rightarrow c$, is exploring a new world in the Kripke structure being constructed as a putative counter-model to the sequent, a world where $a$ is true and $b$ is false.

World = sequence of implication clauses
(identifying the state of the call stack)

# MCSAT as an intuitionistic CDCL?

Instead of using the SAT-solver as a black box,
can MCSAT integrate the CDCL mechanisms with $\mathcal{T}$-reasoning?
(where $\mathcal{T}$ = "intuitionistic entailment")

# MCSAT as an intuitionistic CDCL?

Instead of using the SAT-solver as a black box,
can MCSAT integrate the CDCL mechanisms with $\mathcal{T}$-reasoning?
(where $\mathcal{T}$ = "intuitionistic entailment")

What would be the shape of $\mathcal{T}$-assignments?

# MCSAT as an intuitionistic CDCL?

Instead of using the SAT-solver as a black box,
can MCSAT integrate the CDCL mechanisms with $\mathcal{T}$-reasoning?
(where $\mathcal{T}$ = "intuitionistic entailment")

What would be the shape of $\mathcal{T}$-assignments?
Not: $a \leftarrow$true or $a \leftarrow$false
But: $a \leftarrow$true@$w$ or $a \leftarrow$false@$w$ ("true"/"false" at world $w$)
with worlds $w$ being sequences of implication clauses

# MCSAT as an intuitionistic CDCL?

Instead of using the SAT-solver as a black box,
can MCSAT integrate the CDCL mechanisms with $\mathcal{T}$-reasoning?
(where $\mathcal{T}$ = "intuitionistic entailment")

What would be the shape of $\mathcal{T}$-assignments?
Not: $a \leftarrow$true or $a \leftarrow$false
But: $a \leftarrow$true@$w$ or $a \leftarrow$false@$w$ ("true"/"false" at world $w$)
with worlds $w$ being sequences of implication clauses

**Challenge**: what would be the conflict explanation mechanisms?

# MCSAT as an intuitionistic CDCL?

Instead of using the SAT-solver as a black box,
can MCSAT integrate the CDCL mechanisms with $\mathcal{T}$-reasoning?
(where $\mathcal{T}$ = "intuitionistic entailment")

What would be the shape of $\mathcal{T}$-assignments?
Not: $a \leftarrow$true or $a \leftarrow$false
But: $a \leftarrow$true@$w$ or $a \leftarrow$false@$w$ ("true"/"false" at world $w$)
with worlds $w$ being sequences of implication clauses

**Challenge**: what would be the conflict explanation mechanisms?

On-going work.

# Some propagation mechanisms

$a \leftarrow \mathsf{true}@[w, ((a \Rightarrow b) \Rightarrow c)]$
$b \leftarrow \mathsf{false}@[w, ((a \Rightarrow b) \Rightarrow c)]$
If $d \leftarrow \mathsf{true}@w$ then $d \leftarrow \mathsf{true}@[w, ((a \Rightarrow b) \Rightarrow c)]$

# Some propagation mechanisms

$a \leftarrow \mathsf{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$
$b \leftarrow \mathsf{false@}[w, ((a \Rightarrow b) \Rightarrow c)]$
If $d \leftarrow \mathsf{true@}w$ then $d \leftarrow \mathsf{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$

In presence of a flat clause $(a_1 \wedge \cdots \wedge a_n) \Rightarrow (b_1 \vee \cdots \vee b_m)$:

# Some propagation mechanisms

$a \leftarrow \text{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$
$b \leftarrow \text{false@}[w, ((a \Rightarrow b) \Rightarrow c)]$
If $d \leftarrow \text{true@}w$ then $d \leftarrow \text{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$

In presence of a flat clause $(a_1 \wedge \cdots \wedge a_n) \Rightarrow (b_1 \vee \cdots \vee b_m)$:

If $a_1 \leftarrow \text{true@}w, \ldots, a_n \leftarrow \text{true@}w, b_1 \leftarrow \text{false@}w, \ldots, b_{m-1} \leftarrow \text{false@}w$
then $b_m \leftarrow \text{true@}w$

# Some propagation mechanisms

$a \leftarrow \text{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$
$b \leftarrow \text{false@}[w, ((a \Rightarrow b) \Rightarrow c)]$
If $d \leftarrow \text{true@}w$ then $d \leftarrow \text{true@}[w, ((a \Rightarrow b) \Rightarrow c)]$

In presence of a flat clause $(a_1 \wedge \cdots \wedge a_n) \Rightarrow (b_1 \vee \cdots \vee b_m)$:

If $a_1 \leftarrow \text{true@}w, \ldots, a_n \leftarrow \text{true@}w, b_1 \leftarrow \text{false@}w, \ldots, b_{m-1} \leftarrow \text{false@}w$
then $b_m \leftarrow \text{true@}w$

If $a_1 \leftarrow \text{true@}w, \ldots, a_{n-1} \leftarrow \text{true@}w, b_1 \leftarrow \text{false@}w, \ldots, b_m \leftarrow \text{false@}w$
then $a_n \leftarrow \text{false@}w$

# 4. Quantifier elimination: towards MCSAT for quantified problems?

# Quantifier elimination

A theory $\mathcal{T}$ admits quantifier elimination if for every formula $A$ there is a quantifier-free formula $B$ such that $\mathcal{T} \models A \Leftrightarrow B$ (with $\mathrm{fv}(B) \subseteq \mathrm{fv}(A)$)

# Quantifier elimination

A theory $\mathcal{T}$ admits quantifier elimination if for every formula $A$ there is a quantifier-free formula $B$ such that $\mathcal{T} \models A \Leftrightarrow B$ (with $\mathrm{fv}(B) \subseteq \mathrm{fv}(A)$)

The property is useful for showing the decidability of provability/satisfiability in $\mathcal{T}$:
If

1. it can be shown constructively that $\mathcal{T}$ admits quantifier elimination (there is an algorithm that produces $B$ from $A$),
2. and provability/satisfiability in $\mathcal{T}$ of closed quantifier-free formulae is decidable,

then provability/satisfiability in $\mathcal{T}$ of all formulae is decidable.

# The interesting quantified form

$\mathcal{T}$ admits quantifier elimination as soon as for every formula $\exists y A$, of the form

$$\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$$

there is a quantifier-free formula $B(\overrightarrow{x})$ such that

- $\mathcal{T} \models (\exists y A) \Leftrightarrow B$

(Indeed, quantifiers can then be eliminated from any formula $C$, by structural induction on $C$.)

# The interesting quantified form

$\mathcal{T}$ admits quantifier elimination as soon as for every formula $\exists y A$, of the form

$$\exists y(l_1(\overrightarrow{x}, y) \wedge \cdots \wedge l_m(\overrightarrow{x}, y))$$

there is a quantifier-free formula $B(\overrightarrow{x})$ such that

- $\mathcal{T} \models (\exists y A) \Leftrightarrow B$

(Indeed, quantifiers can then be eliminated from any formula $C$, by structural induction on $C$.)

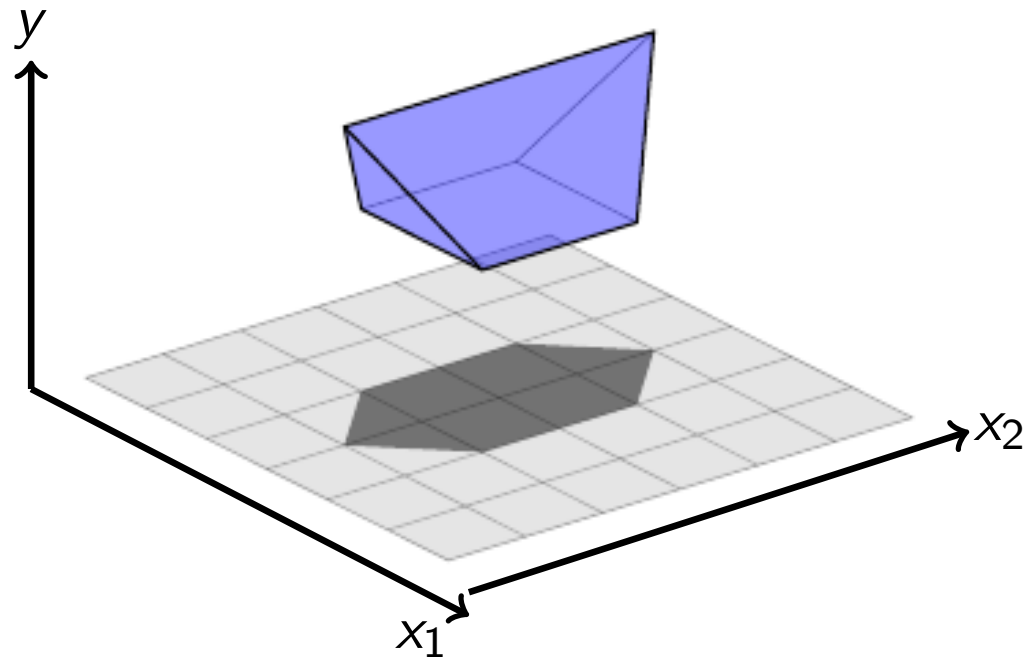
This looks similar to an MCSAT interpolant $B$ explaining a conflict between $\exists y A$ and a model $\mathfrak{M}$ such that $\mathfrak{M} \not\models \exists y A$, namely a $B$ with

- $\mathcal{T} \models (\exists y A) \Rightarrow B$
- $\mathfrak{M} \not\models B$

# Graphically:

Models satisfying $\exists y A$

Models satisfying $\neg B$

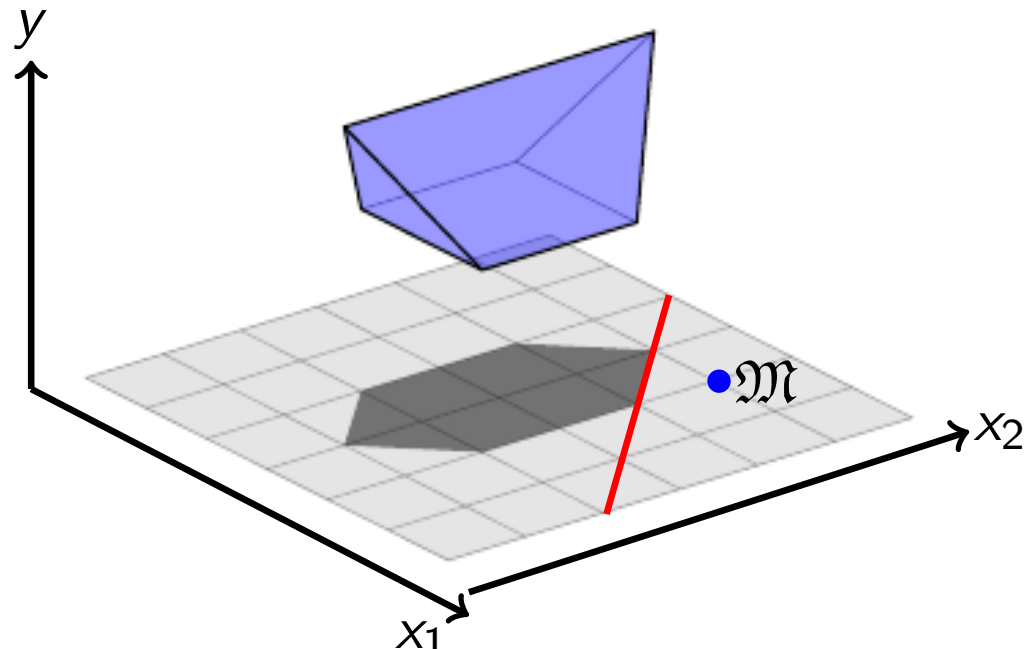



---

Models satisfying $\exists y A$

$\bullet\mathfrak{M}$

Models satisfying $\neg B$



# Main difference with standard QE

We only require $\mathcal{T} \models (\exists y A) \Rightarrow B$, not $\mathcal{T} \models (\exists y A) \Leftrightarrow B$

# Main difference with standard QE

We only require $\mathcal{T} \models (\exists y A) \Rightarrow B$, not $\mathcal{T} \models (\exists y A) \Leftrightarrow B$
This is because we do not seek to remove $y$ from the problem, we simply add $B$ to the problem

# Main difference with standard QE

We only require $\mathcal{T} \models (\exists y A) \Rightarrow B$, not $\mathcal{T} \models (\exists y A) \Leftrightarrow B$
This is because we do not seek to remove $y$ from the problem, we simply add $B$ to the problem . . . in order to "defeat" model $\mathfrak{M}$ (so that our next model construction attempt avoids building models like $\mathfrak{M}$)

# Main difference with standard QE

We only require $\mathcal{T} \models (\exists y A) \Rightarrow B$, not $\mathcal{T} \models (\exists y A) \Leftrightarrow B$
This is because we do not seek to remove $y$ from the problem, we simply add $B$ to the problem . . . in order to "defeat" model $\mathfrak{M}$ (so that our next model construction attempt avoids building models like $\mathfrak{M}$)
The construction of the proofs (e.g. Boolean/FM resolution steps)

- is local to the current model $\mathfrak{M}$
- is guided by model construction attempts
- is lazy

# Main difference with standard QE

We only require $\mathcal{T} \models (\exists y A) \Rightarrow B$, not $\mathcal{T} \models (\exists y A) \Leftrightarrow B$
This is because we do not seek to remove $y$ from the problem, we simply add $B$ to the problem . . . in order to "defeat" model $\mathfrak{M}$ (so that our next model construction attempt avoids building models like $\mathfrak{M}$)
The construction of the proofs (e.g. Boolean/FM resolution steps)

- is local to the current model $\mathfrak{M}$
- is guided by model construction attempts
- is lazy

In fact

- $B$ may not take all of the constraints in $A$ into account (just enough to defeat $\mathfrak{M}$)
- $A$ itself does not contain all constraints involving $y$ in the full problem, only those that $\mathfrak{M}$ has made unit in $y$

An objective is to make the production of $B$ as cheap as possible

# In practice

In non-linear arithmetic, computing the projections for quantifier elimination is done by Cylindrical Algebraic Decomposition (CAD).
Rather expensive if full projections are computed
(i.e., completely eliminate a variable $y$ from the entire problem)

# In practice

In non-linear arithmetic, computing the projections for quantifier elimination is done by Cylindrical Algebraic Decomposition (CAD).
Rather expensive if full projections are computed
(i.e., completely eliminate a variable $y$ from the entire problem)

Using partial / model-guided projections (i.e., lazy quantifier elimination) turns out to be a good approach in practice
(c.f. NLSAT / Yices-MCSAT on NRA).

# Eager vs Lazy

Cylindrincal Algebraic Decomposition give QE for NRA.
Fourier-Motzkin resolution give QE for LRA.

# Eager vs Lazy

Cylindrincal Algebraic Decomposition give QE for NRA.
Fourier-Motzkin resolution give QE for LRA.
. . . like Boolean resolution for Boolean logic.

# Eager vs Lazy

Cylindrincal Algebraic Decomposition give QE for NRA.
Fourier-Motzkin resolution give QE for LRA.
. . . like Boolean resolution for Boolean logic.

Boolean resolution

$$\frac{C_1 \vee \neg y \quad C_2 \vee y}{C_1 \vee C_2}$$

Fourier-Motzkin resolution

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

# Eager vs Lazy

Cylindrincal Algebraic Decomposition give QE for NRA.
Fourier-Motzkin resolution give QE for LRA.
. . . like Boolean resolution for Boolean logic.

Boolean resolution

$$\frac{C_1 \vee \neg y \quad C_2 \vee y}{C_1 \vee C_2}$$

Fourier-Motzkin resolution

$$\frac{e_1 - y \lessdot_1 0 \quad e_2 + y \lessdot_2 0}{e_1 + e_2 \lessdot_3 0}$$

On ground satisfiability problems (i.e. existential problems), QE can be used as a strategy

- to produce Bool. resolution proofs for Boolean CNF problems
- to produce F.-M. resolution profs for LRA problems

. . . but a rather non-efficient one.
MCSAT / conflict resolution gives a much better strategy (a conflict-driven one), the same way CDCL gives a much better strategy for Boolean CNF problems.

# What about arbitrarily quantified problems?

After all, being able to eliminate a quantifier $\mathcal{T} \models (\exists y A) \Leftrightarrow B$ allows the treatment of an arbitrarily quantified formula.

# What about arbitrarily quantified problems?

After all, being able to eliminate a quantifier $\mathcal{T} \models (\exists y A) \Leftrightarrow B$ allows the treatment of an arbitrarily quantified formula.

If we are only interested in determining satisfiability or provability, can we avoid the expensive full computation of equivalent quantifier-free formulae?
Can we have a lazier approach as in the existential case?

# What about arbitrarily quantified problems?

After all, being able to eliminate a quantifier $\mathcal{T} \models (\exists y A) \Leftrightarrow B$ allows the treatment of an arbitrarily quantified formula.

If we are only interested in determining satisfiability or provability, can we avoid the expensive full computation of equivalent quantifier-free formulae?
Can we have a lazier approach as in the existential case?

The basic quantifier elimination algorithm, which treats $\forall$ as $\neg\exists\neg$, heavily relies on the elimination of a quantified variable being an equivalence.
We can anticipate that the weaker condition $\mathcal{T} \models (\exists y A) \Rightarrow B$ required in MCSAT may be insufficient when treating alternations of $\forall$ and $\exists$

# The approach by Bjørner and Janota 1/3

In [**?**], Bjørner and Janota propose an algorithm for "playing with quantified satisfaction", inspired by QBF.

# The approach by Bjørner and Janota 1/3

In [**?**], Bjørner and Janota propose an algorithm for "playing with quantified satisfaction", inspired by QBF.

The SAT and UNSAT players from MCSAT exchange roles at every quantifier alternation:

- The universal player (UP) chooses values for universally quantified variables, trying to make the formula false;
- The existential player (EP) chooses values for existentially quantified variables, trying to make the formula true.

# The approach by Bjørner and Janota 1/3

In [**?**], Bjørner and Janota propose an algorithm for "playing with quantified satisfaction", inspired by QBF.

The SAT and UNSAT players from MCSAT exchange roles at every quantifier alternation:

- The universal player (UP) chooses values for universally quantified variables, trying to make the formula false;
- The existential player (EP) chooses values for existentially quantified variables, trying to make the formula true.

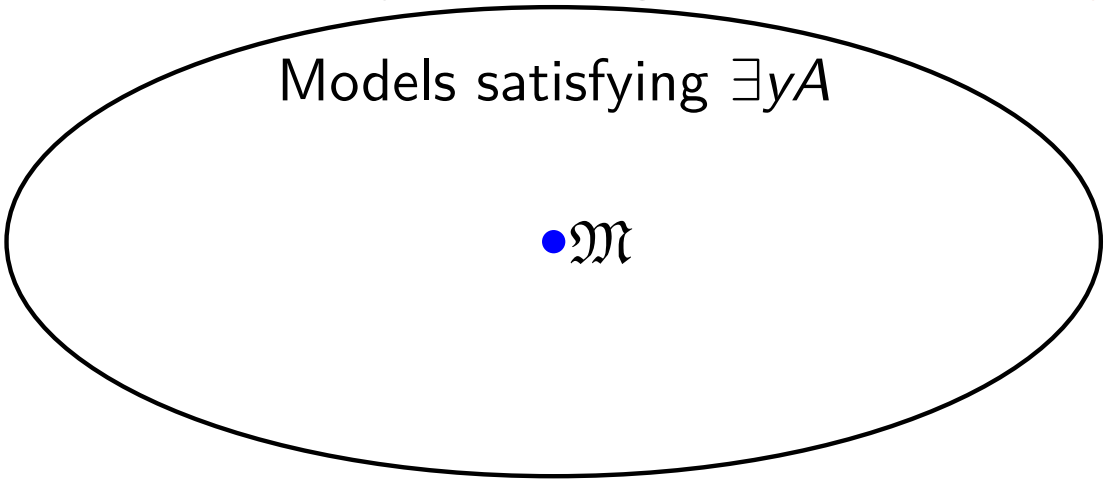
In order to determine whether $\forall x \exists y A$ is satisfiable:

- UP first chooses a value for $x$;
- If EP cannot find a value for $y$ to make $A$ true, UP has won.
- If on the contrary EP can find a value, UP needs to understand what was wrong about its choice for $x$.

Relies on theory $\mathcal{T}$ having model-based projection

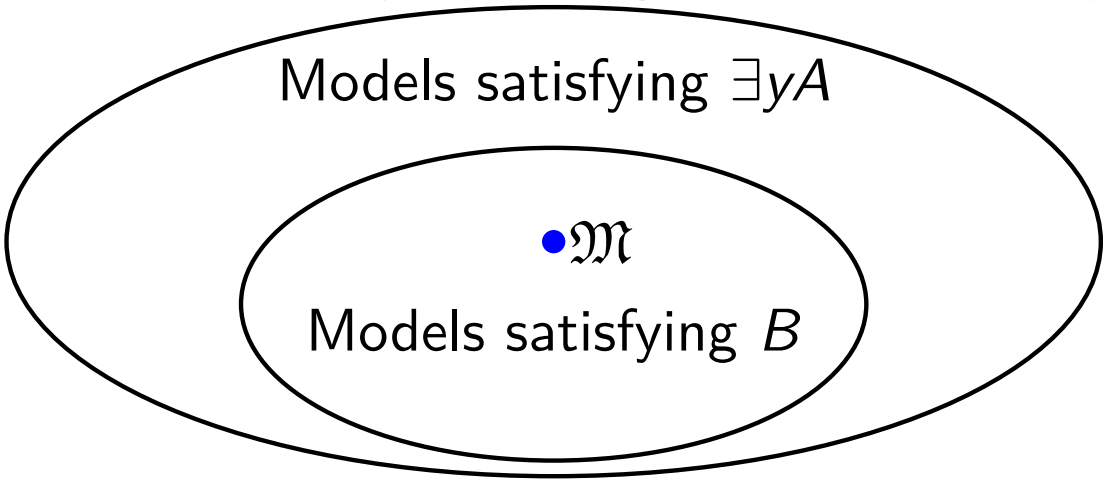
# The approach by Bjørner and Janota 2/3

Relies on theory $\mathcal{T}$ having model-based projection



# The approach by Bjørner and Janota 2/3

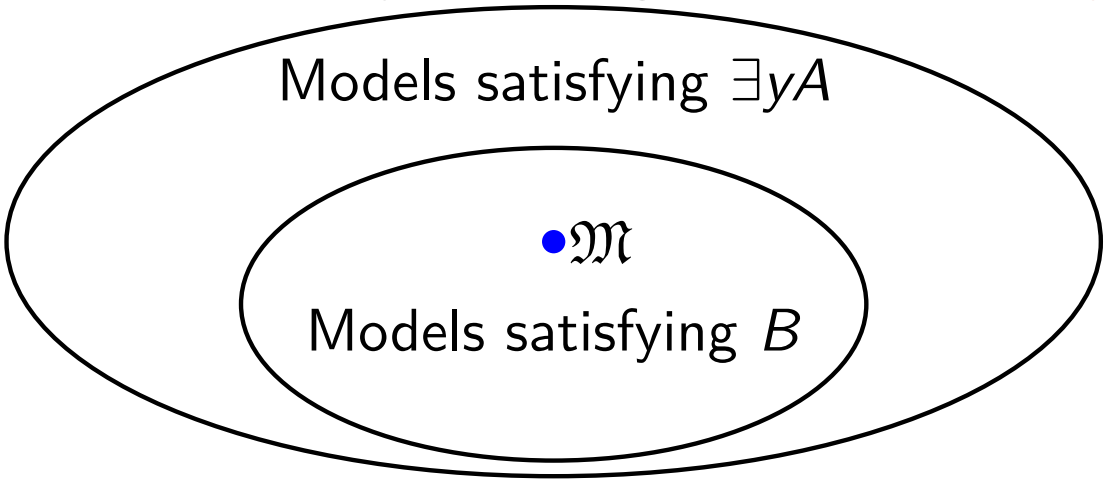
Relies on theory $\mathcal{T}$ having model-based projection



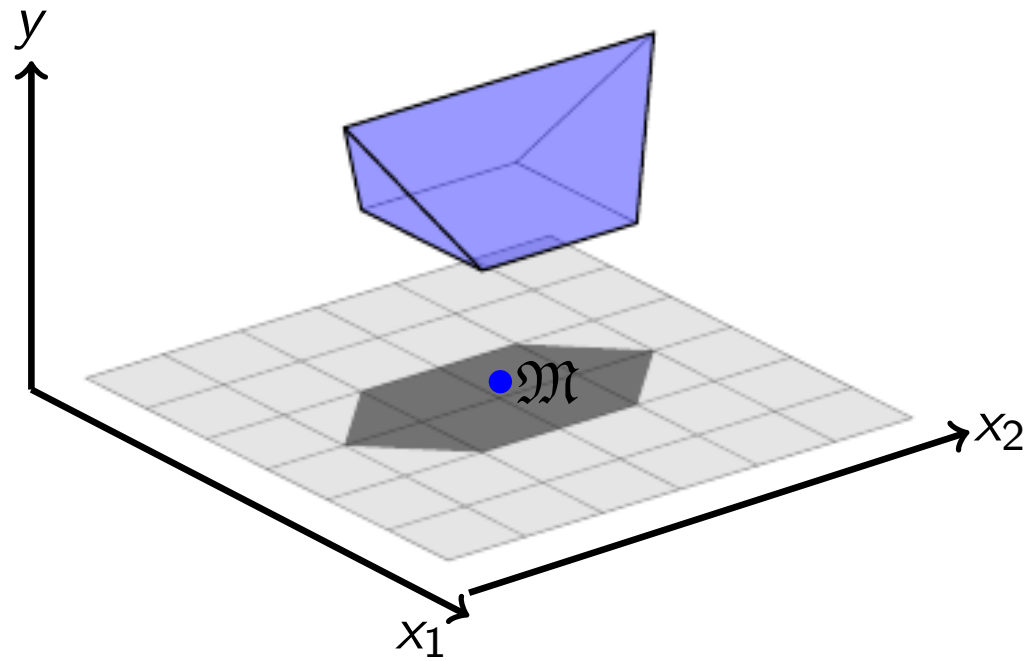

From $\mathfrak{M} \models \exists yA$,
produce quantifier-free
$B = mbp(\mathfrak{M}, \exists yA)$
such that
$\mathcal{T} \models B \Rightarrow \exists yA$.

# The approach by Bjørner and Janota 2/3

Relies on theory $\mathcal{T}$ having model-based projection

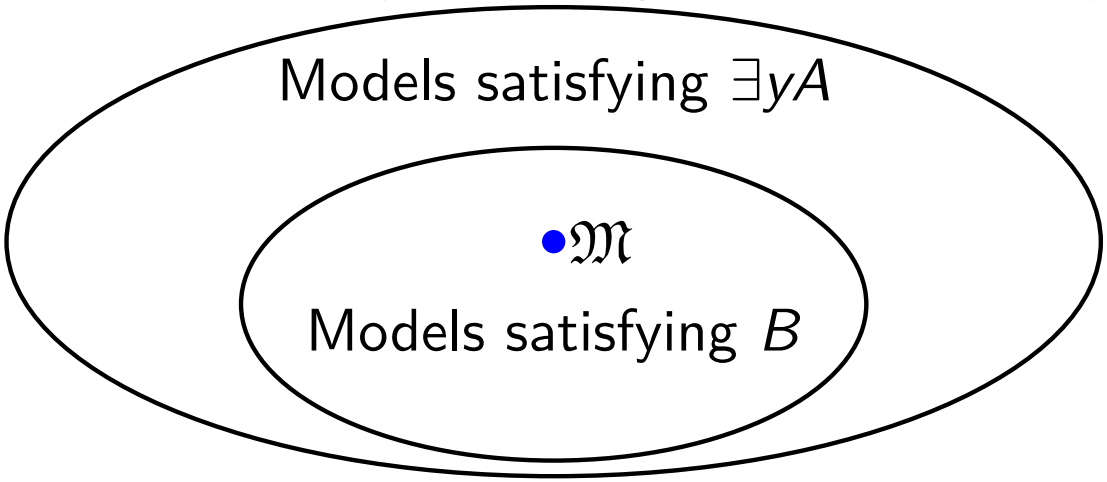



From $\mathfrak{M} \models \exists yA$,
produce quantifier-free
$B = mbp(\mathfrak{M}, \exists yA)$
such that
$\mathcal{T} \models B \Rightarrow \exists yA$.

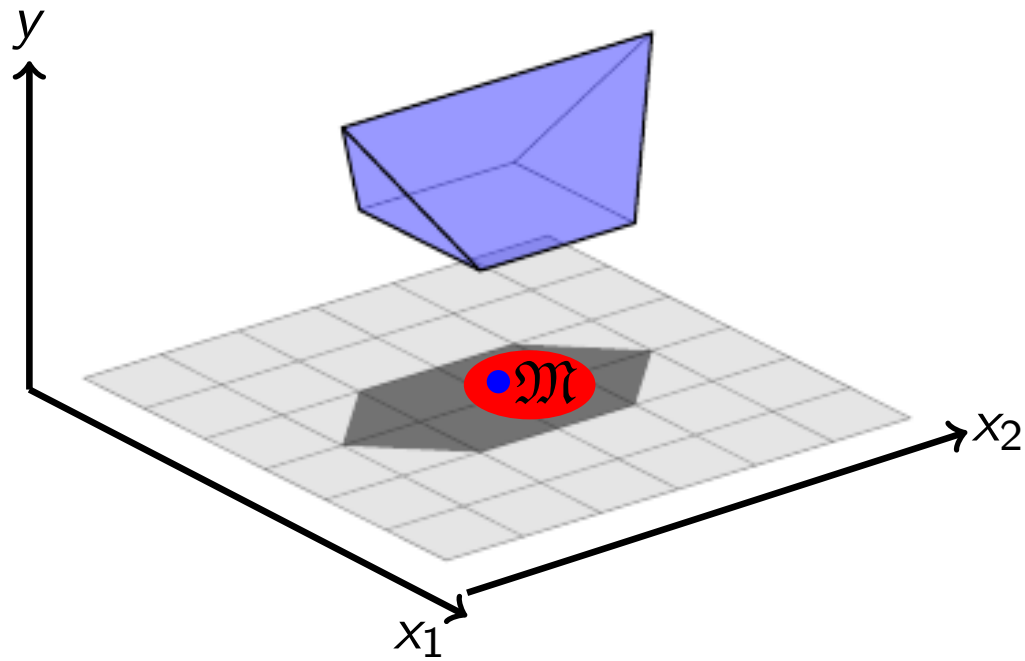


# The approach by Bjørner and Janota 2/3

Relies on theory $\mathcal{T}$ having model-based projection




From $\mathfrak{M} \models \exists yA$,
produce quantifier-free
$B = mbp(\mathfrak{M}, \exists yA)$
such that
$\mathcal{T} \models B \Rightarrow \exists yA$.



# The approach by Bjørner and Janota 3/3

# The approach by Bjørner and Janota 3/3

And $\exists y A$ should be covered by finitely many $B$'s:
$\mathcal{T} \models (\exists y A) \Leftrightarrow \bigvee_i mbp(\mathfrak{M}_i, \exists y A)$ for a finite number of models $\mathfrak{M}_i$.

specific form of quantifier elimination

# The approach by Bjørner and Janota 3/3

And $\exists y A$ should be covered by finitely many $B$'s:
$\mathcal{T} \models (\exists y A) \Leftrightarrow \bigvee_i mbp(\mathfrak{M}_i, \exists y A)$ for a finite number of models $\mathfrak{M}_i$.

specific form of quantifier elimination

From there, an algorithm is given that applies the mbp computations lazily.

# Questions:

On purely existential problems, does the Bjørner-Janota algorithm collapse to MCSAT?
I suspect not: the form of laziness seems distinct, but it also seems complementary.

# Questions:

On purely existential problems, does the Bjørner-Janota algorithm collapse to MCSAT?
I suspect not: the form of laziness seems distinct, but it also seems complementary.

Ongoing work:

- extend MCSAT with the Bjørner-Janota approach for handling arbitrary alternations of quantifiers.
- what is the notion of proof production in this approach?

# Related work and future work

Investigate related approaches:

- The ANR Decert work on Linear Integer arithmetic, which extends Fourier-Motzkin with simplex-based techniques [BCC+12]
- Monniaux's work on quantifier elimination [Mon08, Mon10]. It uses a ground SMT-solver as a black box (for purely existential problems), and also performs some QE-elimination steps (e.g., FM resolutions) independently from the SMT-solver.
- Dutertre's work on solving "EF problems" ($\exists\forall$) in Yices, also relying on a ground SMT-solver considered as a black box.

How would the Bjørner-Janota approach work in a combination of theories?
Just as our CDSAT system generalises MCSAT to a combination of theories, what would be the equivalent for the Bjørner-Janota approach?

Questions?

F. Bobot, S. Conchon, E. Contejean, M. Iguernelala, A. Mahboubi, A. Mebsout, and G. Melquiond.
A simplex-based extension of fourier-motzkin for solving linear integer arithmetic.
In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning*, pages 67–81. Springer Berlin Heidelberg, 2012.

M. P. Bonacina, S. Graham-Lengrand, and N. Shankar.
Conflict-driven satisfiability for theory combination: Transition system and completeness.
*J. of Automated Reasoning*, in press:1–31, 2019.

S. Cotton.
Natural domain SMT: A preliminary assessment.
In K. Chatterjee and T. A. Henzinger, editors, *Proc. of the 8th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 6246 of *LNCS*, pages 77–91. Springer-Verlag, 2010.