Interpolating bitvector arithmetic constraints in MCSAT (preliminary report)

Stéphane Graham-Lengrand and Dejan Jovanović

SMT workshop, 7th July 2019

Outline

1. What is this about?
2. Why are we interested in this?
3. How do we do this?

1. What is this about?

A particular form of interpolation

Interpolation is usually considered between

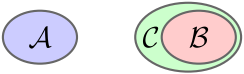
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,

in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

A particular form of interpolation

Interpolation is usually considered between

- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,

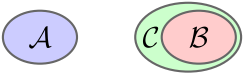


in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

A particular form of interpolation

Interpolation is usually considered between

- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

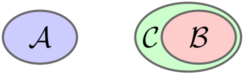
- a formula $\mathcal{A}$
- a model Γ such that $\Gamma \not\models \mathcal{A}$

in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\Gamma \models \mathcal{C}$.

A particular form of interpolation

Interpolation is usually considered between

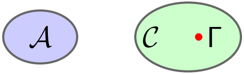
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

- a formula $\mathcal{A}$
- a model Γ such that $\Gamma \not\models \mathcal{A}$

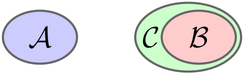


in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\Gamma \models \mathcal{C}$.

A particular form of interpolation

Interpolation is usually considered between

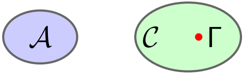
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in “the common language of $\mathcal{A}$ and $\mathcal{B}$” such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

- a formula $\mathcal{A}$
- a model Γ such that $\Gamma \not\models_{\mathcal{T}} \mathcal{A}$

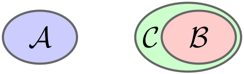


in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models_{\mathcal{T}} \bot$ and $\Gamma \models_{\mathcal{T}} \mathcal{C}$.

A particular form of interpolation

Interpolation is usually considered between

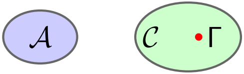
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

- a formula $\mathcal{A}$ of the form[†] $\exists y(C_1 \wedge \cdots \wedge C_m)$
- a model Γ such that $\Gamma \not\models_{\mathcal{T}} \mathcal{A}$



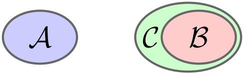
in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models_{\mathcal{T}} \bot$ and $\Gamma \models_{\mathcal{T}} \mathcal{C}$.

[†] where $C_1, \ldots, C_m$ are (quantifier-free) constraints of a theory $\mathcal{T}$.

A particular form of interpolation

Interpolation is usually considered between

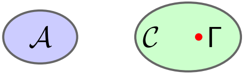
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

- a formula $\mathcal{A}$ of the form† $\exists y(C_1 \wedge \cdots \wedge C_m)$
- a model Γ such that $\Gamma \not\models_{\mathcal{T}} \mathcal{A}$



in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models_{\mathcal{T}} \bot$ and $\Gamma \models_{\mathcal{T}} \mathcal{C}$.

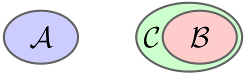
† where $C_1, \ldots, C_m$ are (quantifier-free) constraints of a theory $\mathcal{T}$.

Where do we find such problems?

A particular form of interpolation

Interpolation is usually considered between

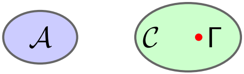
- a formula $\mathcal{A}$
- a formula $\mathcal{B}$ such that $\mathcal{A}, \mathcal{B} \models \bot$,



in the form of a formula $\mathcal{C}$ in "the common language of $\mathcal{A}$ and $\mathcal{B}$" such that $\mathcal{A}, \mathcal{C} \models \bot$ and $\mathcal{B} \models \mathcal{C}$.

In this talk, interpolation is between

- a formula $\mathcal{A}$ of the form† $\exists y(C_1 \wedge \cdots \wedge C_m)$
- a model Γ such that $\Gamma \not\models_{\mathcal{T}} \mathcal{A}$



in the form of a formula $\mathcal{C}$ that is quantifier-free, such that $\mathcal{A}, \mathcal{C} \models_{\mathcal{T}} \bot$ and $\Gamma \models_{\mathcal{T}} \mathcal{C}$.

† where $C_1, \ldots, C_m$ are (quantifier-free) constraints of a theory $\mathcal{T}$.

Where do we find such problems?
… in model-constructing satisfiability (MCSAT)

2. Why are we interested in this?

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

MCSAT introduced in [dMJ13, JBdM13, Jov17],
following work on specific decision procedures for theories such as non-linear arithmetic [JdM12].

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

MCSAT introduced in [dMJ13, JBdM13, Jov17], following work on specific decision procedures for theories such as non-linear arithmetic [JdM12].

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

Evaluation is a key aspect of MCSAT

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

MCSAT introduced in [dMJ13, JBdM13, Jov17],
following work on specific decision procedures for theories such as non-linear arithmetic [JdM12].

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)

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

MCSAT introduced in [dMJ13, JBdM13, Jov17],
following work on specific decision procedures for theories such as non-linear arithmetic [JdM12].

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 offers:

- a template for decision procedures
- an integration of such procedures with Boolean reasoning
- new possibilities for combining procedures [JBdM13, BGLS19]

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

MCSAT introduced in [dMJ13, JBdM13, Jov17],
following work on specific decision procedures for theories such as non-linear arithmetic [JdM12].

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 offers:

- a template for decision procedures
- an integration of such procedures with Boolean reasoning
- new possibilities for combining procedures [JBdM13, BGLS19]

The template is a generalisation of how CDCL works.

Run = alternation of search phases and conflict analysis phases

Boolean theory can be given the same status as other theories.

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

- **Method**: as in CDCL for Boolean logic,
  successively guess 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

- **Method**: as in CDCL for Boolean logic,
  successively guess 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

- **Method**: as in CDCL for Boolean logic, successively guess 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

- **Method**: as in CDCL for Boolean logic, successively guess 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, it means:

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

- **Method**: as in CDCL for Boolean logic, successively guess 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, it means:
  - Some variables $x_1, \ldots, x_n$ have already been assigned values $v_1, \ldots, v_n$: this constitutes a partial model Γ

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

- **Method**: as in CDCL for Boolean logic, successively guess 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, it means:
  - Some variables $x_1, \ldots, x_n$ have already been assigned values $v_1, \ldots, v_n$: this constitutes a partial model Γ
  - There is a variable y such that, for all possible value v, there is a constraint $C(x_1, \ldots, x_n, y)$ that evaluates to false in $\Gamma, y \mapsto v$

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

- **Method**: as in CDCL for Boolean logic, successively guess 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, it means:
  - Some variables $x_1, \ldots, x_n$ have already been assigned values $v_1, \ldots, v_n$: this constitutes a partial model Γ
  - There is a variable y such that, for all possible value v, there is a constraint $C(x_1, \ldots, x_n, y)$ that evaluates to false in $\Gamma, y \mapsto v$
    In other words: for a subset $\{C_1, \ldots, C_m\}$ of the constraints, formula $\mathcal{A} = \exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ

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

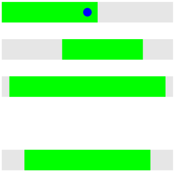
- **Method**: as in CDCL for Boolean logic, successively guess values to assign to variables …while maintaining the invariant: *given the assignments made so far, none of the constraints evaluates to false*
- To pick a value for variable y after $x_1, \ldots, x_n$ were assigned values $v_1, \ldots, v_n$, 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, it means:
  - Some variables $x_1, \ldots, x_n$ have already been assigned values $v_1, \ldots, v_n$: this constitutes a partial model Γ
  - There is a variable y such that, for all possible value v, there is a constraint $C(x_1, \ldots, x_n, y)$ that evaluates to false in $\Gamma, y \mapsto v$
    In other words: for a subset $\{C_1, \ldots, C_m\}$ of the constraints, formula $\mathcal{A} = \exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ

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

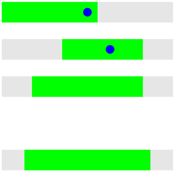
Search phase (satisfiable case)

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



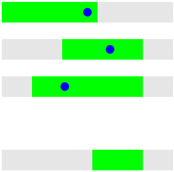
Search phase (satisfiable case)

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

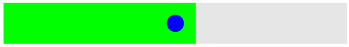


Search phase (satisfiable case)

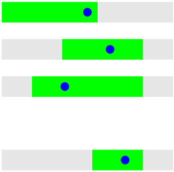
Free var within	Constraints (unit ones in red)	Feasible set	Var
$\{x_1\}$	$C_1^1, \ldots, C_j^1, \ldots$		x_1
$\{x_1, x_2\}$	$C_1^2, C_2^2, \ldots, C_j^2, \ldots$		x_2
$\{x_1, x_2, x_3\}$	$C_1^3, C_2^3, \ldots, C_j^3, \ldots$		x_3
$\ldots$			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots,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

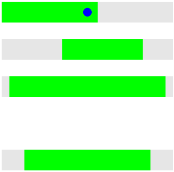
SAT

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
$\ldots$			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$		x_i

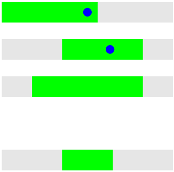
Search phase (conflict case)

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
$\ldots$			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$		x_i



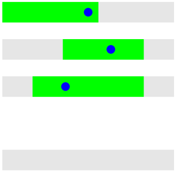
Search phase (conflict case)

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
$\ldots$			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots, C_j^i, \ldots$		x_i



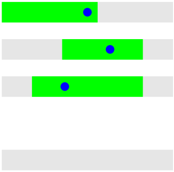
Search phase (conflict case)

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)

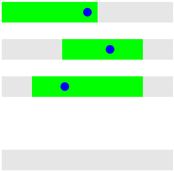
Free var within	Constraints (unit ones in red)	Feasible set	Var
$\{x_1\}$	$C_1^1, \ldots, C_j^1, \ldots$		x_1
$\{x_1, x_2\}$	$C_1^2, C_2^2, \ldots, C_j^2, \ldots$		x_2
$\{x_1, x_2, x_3\}$	$C_1^3, C_2^3, \ldots, C_j^3, \ldots$		x_3
$\ldots$			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots,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



Conflict

Search phase (conflict case)

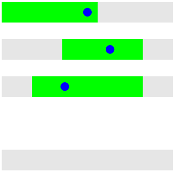
Free var within	Constraints (unit ones in red)	Feasible set	Var
$\{x_1\}$	$C_1^1, \ldots, C_j^1, \ldots$		x_1
$\{x_1, x_2\}$	$C_1^2, C_2^2, \ldots, C_j^2, \ldots$		x_2
$\{x_1, x_2, x_3\}$	$C_1^3, C_2^3, \ldots, C_j^3, \ldots$		x_3
...			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots,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



Conflict

What to do now? (when $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ)

Search phase (conflict case)

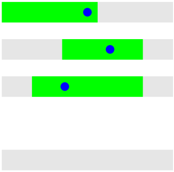
Free var within	Constraints (unit ones in red)	Feasible set	Var
$\{x_1\}$	$C_1^1, \ldots, C_j^1, \ldots$		x_1
$\{x_1, x_2\}$	$C_1^2, C_2^2, \ldots, C_j^2, \ldots$		x_2
$\{x_1, x_2, x_3\}$	$C_1^3, C_2^3, \ldots, C_j^3, \ldots$		x_3
...			
$\{x_1, \ldots, x_i\}$	$C_1^i, C_2^i, \ldots, C_{42}^i, \ldots,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

Conflict

What to do now? (when $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ)
Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$
(i.e. try another Γ')

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



Conflict

What to do now? (when $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ)

Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$

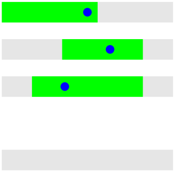
(i.e. try another Γ')

How do we avoid picking the same values (i.e. the same Γ)?

How do we avoid picking a Γ' that fails for the same reason Γ fails?

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



Conflict

What to do now? (when $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ)

Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$

(i.e. try another Γ')

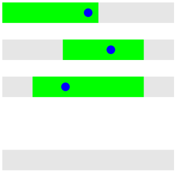
How do we avoid picking the same values (i.e. the same Γ)?

How do we avoid picking a Γ' that fails for the same reason Γ fails?

Learn a lemma that rules out not only Γ but a set of similar models

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



Conflict

What to do now? (when $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in Γ)

Backtrack and try new values $v_1', \ldots, v_n'$ for $x_1, \ldots, x_n$

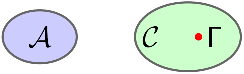
(i.e. try another Γ')

How do we avoid picking the same values (i.e. the same Γ)?

How do we avoid picking a Γ' that fails for the same reason Γ fails?

Learn a lemma that rules out not only Γ but a set of similar models

$\mathcal{A} \wedge \mathcal{C} \Rightarrow \bot$ (or equivalently quantif.-free $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$)



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 interpolation 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 interpolation mechanism, producing $\mathcal{C}$ as a conjunction of literals (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause)

... 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 interpolation mechanism,
  producing $\mathcal{C}$ as a conjunction of literals
  (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause),
  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 interpolation mechanism, producing $\mathcal{C}$ as a conjunction of literals (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause), 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 interpolation mechanism, producing $\mathcal{C}$ as a conjunction of literals (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause), 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)

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 interpolation mechanism, producing $\mathcal{C}$ as a conjunction of literals (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause), 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)

Now we are developing the case of the bitvector theory for $\mathcal{T}$

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 interpolation mechanism,
  producing $\mathcal{C}$ as a conjunction of literals
  (so that the theory lemma $C_1 \wedge \cdots \wedge C_m \wedge \mathcal{C} \Rightarrow \bot$ is a clause),
  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)

Now we are developing the case of the bitvector theory for $\mathcal{T}$

Bitvectors in MCSAT first looked at in [ZWR16]

"*Interpolants do have applications in mcBV, e.g., for conflict generalization, but we do not currently employ such methods.*"

This is what we do now.

3. How do we do this?

Bitvectors

- a nice way of representing domains of feasible values:
- an interpolation mechanism,
  satisfying some suitable conditions for termination;

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;
  - **naive interpolation mechanism**:
    if $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\Gamma = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$
    take $\mathcal{C} \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;
  - **naive interpolation mechanism**:
    if $\exists y (C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\Gamma = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$
    take $\mathcal{C} \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out Γ)

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;
  - **naive interpolation mechanism**:
    if $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\Gamma = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$
    take $\mathcal{C} \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out Γ)
  - **default interpolation 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

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;
  - **naive interpolation mechanism**:
    if $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\Gamma = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$
    take $\mathcal{C} \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out Γ)
  - **default interpolation 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 interpolant, for the rest of the run

Bitvectors

- a nice way of representing domains of feasible values:
  We use Binary Decision Diagrams (see [GLJ17])
  ([ZWR16] uses intervals + bit masks)
- an interpolation mechanism,
  satisfying some suitable conditions for termination;
  - **naive interpolation mechanism**:
    if $\exists y(C_1 \wedge \cdots \wedge C_m)$ evaluates to false in $\Gamma = \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\}$
    take $\mathcal{C} \;=\; x_1 \simeq v_1 \wedge \cdots \wedge x_n \simeq v_n$ (only rules out Γ)
  - **default interpolation 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 interpolant, for the rest of the run
    - interpolants close to the bit level do not give understandable explanations of what was wrong about Γ; we'd rather have interpolants close to the word level

Word-level interpolation

Difficulty is the diversity of word-level bitvector operations

Word-level interpolation

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice interpolation mechanisms

Word-level interpolation

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice interpolation mechanisms

In [GLJ17]: equality with concat+extract
Interpolant built by slicing

Word-level interpolation

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice interpolation mechanisms

In [GLJ17]: equality with concat+extract
Interpolant built by slicing

Today: (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	$::=$	$c_0^w + \Sigma_{i=1}^n c_i^w \cdot x_i^w$

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 $c \cdot c' \in \{0, 1\}$.

Word-level interpolation

Difficulty is the diversity of word-level bitvector operations

We identify fragments of the bitvector theory for which we can design nice interpolation mechanisms

In [GLJ17]: equality with concat+extract
Interpolant built by slicing

Today: (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	$::=$	$c_0 + \Sigma_{i=1}^{n} c_i \cdot x_i$

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 $c \cdot c' \in \{0, 1\}$.

Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \mathsf{false}$

Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y, l_i is the (included) lower bound, u_i the (excluded) upper bound, both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y, l_i is the (included) lower bound, u_i the (excluded) upper bound, both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

  Intuitively, $\mathcal{A}$ is equivalent to $\exists y(y \notin I_1 \wedge \cdots \wedge y \notin I_m)$, i.e., $\exists y(y \notin \bigcup_{i=1}^{m} I_i)$, i.e., $(\bigcup_{i=1}^{n} I_i) \neq (\mathbb{Z}/2^w\mathbb{Z})$
  $[\![\mathcal{A}]\!]_\Gamma = \text{false}$ means that $\bigcup_{i=1}^{m}[\![I_i]\!]_\Gamma = (\mathbb{Z}/2^w\mathbb{Z})$.

Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y, l_i is the (included) lower bound, u_i the (excluded) upper bound, both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

  Intuitively, $\mathcal{A}$ is equivalent to $\exists y(y \notin I_1 \wedge \cdots \wedge y \notin I_m)$, i.e., $\exists y(y \notin \bigcup_{i=1}^{m} I_i)$, i.e., $(\bigcup_{i=1}^{n} I_i) \neq (\mathbb{Z}/2^w\mathbb{Z})$
  $[\![\mathcal{A}]\!]_\Gamma = \text{false}$ means that $\bigcup_{i=1}^{m} [\![I_i]\!]_\Gamma = (\mathbb{Z}/2^w\mathbb{Z})$.
- We produce a series of constraints $E_1, \ldots, E_p$ such that
  - $\Gamma \models E_1, \ldots, E_p$
  - $E_1, \ldots, E_p \models$ "$(\bigcup_{i=1}^{n} I_i) = (\mathbb{Z}/2^w\mathbb{Z})$"

Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y,
  l_i is the (included) lower bound, u_i the (excluded) upper bound,
  both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

  Intuitively, $\mathcal{A}$ is equivalent to $\exists y(y \notin I_1 \wedge \cdots \wedge y \notin I_m)$, i.e.,
  $\exists y(y \notin \bigcup_{i=1}^m I_i)$, i.e., $(\bigcup_{i=1}^n I_i) \neq (\mathbb{Z}/2^w\mathbb{Z})$
  $[\![\mathcal{A}]\!]_\Gamma = \text{false}$ means that $\bigcup_{i=1}^m [\![I_i]\!]_\Gamma = (\mathbb{Z}/2^w\mathbb{Z})$.

- We produce a series of constraints $E_1, \ldots, E_p$ such that
  - $\Gamma \models E_1, \ldots, E_p$
  - $E_1, \ldots, E_p \models$ "$(\bigcup_{i=1}^n I_i) = (\mathbb{Z}/2^w\mathbb{Z})$" i.e., $\mathcal{A}, E_1, \ldots, E_p \models \bot$

Overview of the interpolation algorithm

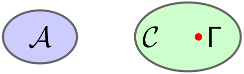
$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y,
  l_i is the (included) lower bound, u_i the (excluded) upper bound,
  both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

  Intuitively, $\mathcal{A}$ is equivalent to $\exists y(y \notin I_1 \wedge \cdots \wedge y \notin I_m)$, i.e.,
  $\exists y(y \notin \bigcup_{i=1}^{m} I_i)$, i.e., $(\bigcup_{i=1}^{n} I_i) \neq (\mathbb{Z}/2^w\mathbb{Z})$
  $[\![\mathcal{A}]\!]_\Gamma = \text{false}$ means that $\bigcup_{i=1}^{m}[\![I_i]\!]_\Gamma = (\mathbb{Z}/2^w\mathbb{Z})$.

- We produce a series of constraints $E_1, \ldots, E_p$ such that
  - $\Gamma \models E_1, \ldots, E_p$
  - $E_1, \ldots, E_p \models$ "$(\bigcup_{i=1}^{n} I_i) = (\mathbb{Z}/2^w\mathbb{Z})$" i.e., $\mathcal{A}, E_1, \ldots, E_p \models \perp$

  We take $\mathcal{C}$ to be $E_1 \wedge \cdots \wedge E_p$



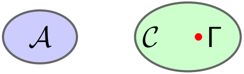
Overview of the interpolation algorithm

$\mathcal{A}$ is $\exists y(C_1 \wedge \cdots \wedge C_m)$ and $\Gamma \not\models \mathcal{A}$, i.e., $[\![\mathcal{A}]\!]_\Gamma = \text{false}$

- Each C_i turns into a forbidden interval $I_i = [l_i; u_i[$ for y,
  l_i is the (included) lower bound, u_i the (excluded) upper bound,
  both are terms not mentioning y, and $C_i \Leftrightarrow (y \notin I_i)$.

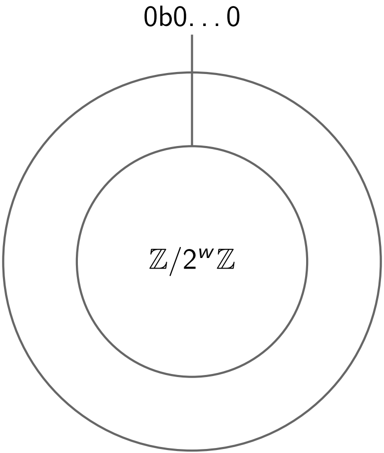
  Intuitively, $\mathcal{A}$ is equivalent to $\exists y(y \notin I_1 \wedge \cdots \wedge y \notin I_m)$, i.e.,
  $\exists y(y \notin \bigcup_{i=1}^{m} I_i)$, i.e., $(\bigcup_{i=1}^{n} I_i) \neq (\mathbb{Z}/2^w\mathbb{Z})$
  $[\![\mathcal{A}]\!]_\Gamma = \text{false}$ means that $\bigcup_{i=1}^{m} [\![I_i]\!]_\Gamma = (\mathbb{Z}/2^w\mathbb{Z})$.
- We produce a series of constraints $E_1, \ldots, E_p$ such that
  - $\Gamma \models E_1, \ldots, E_p$
  - $E_1, \ldots, E_p \models$ "$(\bigcup_{i=1}^{n} I_i) = (\mathbb{Z}/2^w\mathbb{Z})$" i.e., $\mathcal{A}, E_1, \ldots, E_p \models \bot$

  We take $\mathcal{C}$ to be $E_1 \wedge \cdots \wedge E_p$

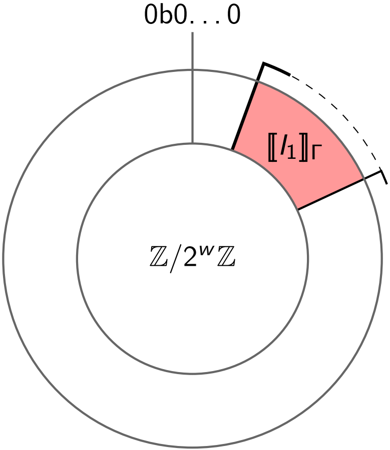


Our intervals are taken modulo 2^w (i.e., they may overflow):
$[0b1111; 0b0001[$ contains two values, namely $0b1111$ and $0b0000$

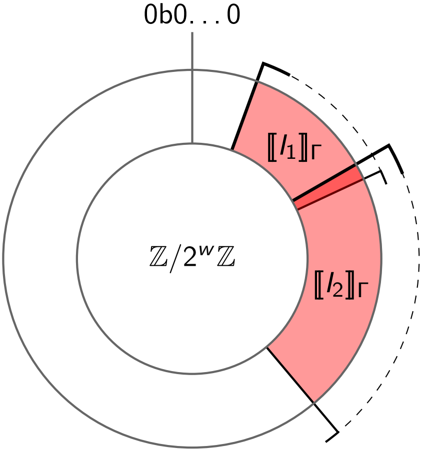
Example



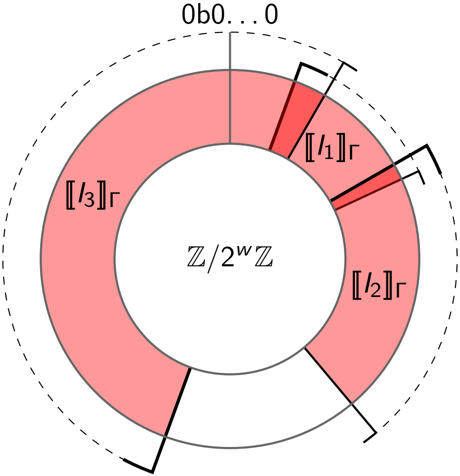
Example



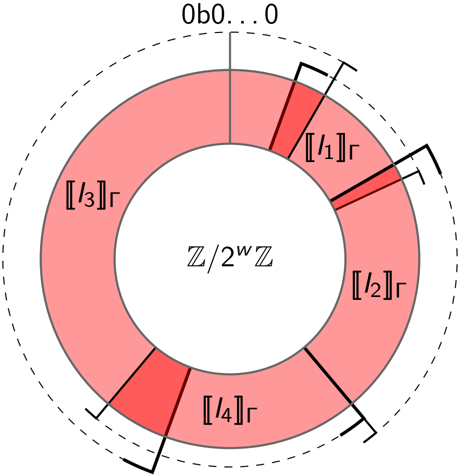
Example



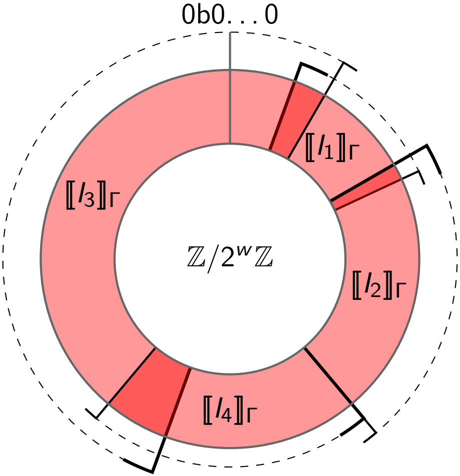
Example



Example

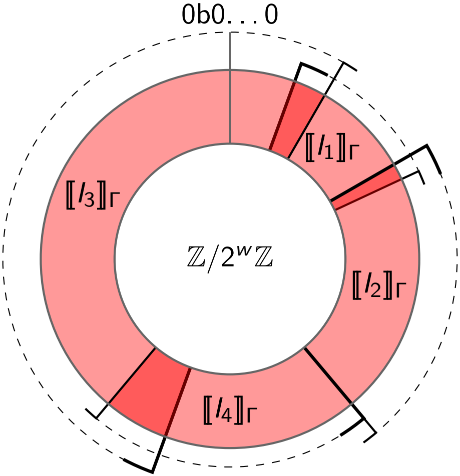


Example



Coverage of $\mathbb{Z}/2^w\mathbb{Z}$ is full because:
$[\![u_1]\!]_\Gamma \in [\![l_2]\!]_\Gamma$ and $[\![u_2]\!]_\Gamma \in [\![l_4]\!]_\Gamma$ and $[\![u_4]\!]_\Gamma \in [\![l_3]\!]_\Gamma$ and $[\![u_3]\!]_\Gamma \in [\![l_1]\!]_\Gamma$

Example



Coverage of $\mathbb{Z}/2^w\mathbb{Z}$ is full because:
$[\![u_1]\!]_\Gamma \in [\![I_2]\!]_\Gamma$ and $[\![u_2]\!]_\Gamma \in [\![I_4]\!]_\Gamma$ and $[\![u_4]\!]_\Gamma \in [\![I_3]\!]_\Gamma$ and $[\![u_3]\!]_\Gamma \in [\![I_1]\!]_\Gamma$
We take $\mathcal{C}$ to be $(u_1 \in I_2) \wedge (u_2 \in I_4) \wedge (u_4 \in I_3) \wedge (u_3 \in I_1)$

Producing the forbidden intervals: preprocessing

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

$$\begin{array}{lcl} t_1 \leq^s t_2 & \rightsquigarrow & t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1} \\ t_1 \simeq t_2 & \rightsquigarrow & t_1 - t_2 \leq^u 0 \end{array}$$

Producing the forbidden intervals: preprocessing

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

$$
\begin{array}{lcl}
t_1 \leq^s t_2 & \rightsquigarrow & t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1} \\
t_1 \simeq t_2 & \rightsquigarrow & t_1 - t_2 \leq^u 0
\end{array}
$$

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

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

Producing the forbidden intervals: preprocessing

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

$$\begin{array}{lcl} t_1 \leq^s t_2 & \rightsquigarrow & t_1 + 2^{w-1} \leq^u t_2 + 2^{w-1} \\ t_1 \simeq t_2 & \rightsquigarrow & t_1 - t_2 \leq^u 0 \end{array}$$

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

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

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:

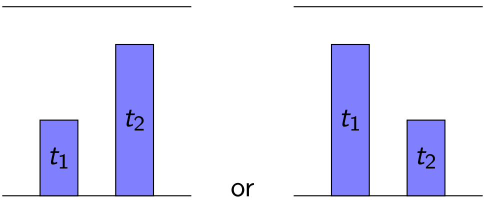
Normalised atom a	Forbidden interval that a (resp. $\neg a$) specifies for y		
	I_a	$I_{\neg a}$	**Condition**
$t_1 + y \leq^u t_2 + y$	$[-t_2; -t_1[$	$[-t_1; -t_2[$	$t_1 \not\simeq t_2$
	$\emptyset$	$[0; 0[$	$t_1 \simeq t_2$
$t_1 \leq^u t_2 + y$	$[-t_2; t_1 - t_2[$	$[t_1 - t_2; -t_2[$	$t_1 \not\simeq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

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
$t_1 + y \leq^u t_2 + y$	$[-t_2; -t_1[$	$[-t_1; -t_2[$	$t_1 \not\simeq t_2$
	$\emptyset$	$[0; 0[$	$t_1 \simeq t_2$
$t_1 \leq^u t_2 + y$	$[-t_2; t_1 - t_2[$	$[t_1 - t_2; -t_2[$	$t_1 \not\simeq 0$
	$\emptyset$	$[0; 0[$	$t_1 \simeq 0$
$t_1 + y \leq^u t_2$	$[t_2 - t_1 + 1; -t_1[$	$[-t_1; t_2 - t_1 + 1[$	$t_2 \not\simeq -1$
	$\emptyset$	$[0; 0[$	$t_2 \simeq -1$
$t_1 \leq^u t_2$	$[0; 0[$	$\emptyset$	$t_2 <^u t_1$
	$\emptyset$	$[0; 0[$	$t_1 \leq^u t_y2$

Illustrating the first line:



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
$t_1 + y \leq^u t_2 + y$	$[-t_2; -t_1[$	$[-t_1; -t_2[$	$t_1 \not\simeq t_2$
	$\emptyset$	$[0; 0[$	$t_1 \simeq t_2$
$t_1 \leq^u t_2 + y$	$[-t_2; t_1 - t_2[$	$[t_1 - t_2; -t_2[$	$t_1 \not\simeq 0$
	$\emptyset$	$[0; 0[$	$t_1 \simeq 0$
$t_1 + y \leq^u t_2$	$[t_2 - t_1 + 1; -t_1[$	$[-t_1; t_2 - t_1 + 1[$	$t_2 \not\simeq -1$
	$\emptyset$	$[0; 0[$	$t_2 \simeq -1$
$t_1 \leq^u t_2$	$[0; 0[$	$\emptyset$	$t_2 <^u t_1$
	$\emptyset$	$[0; 0[$	$t_1 \leq^u t_y 2$

Example 1: $\Gamma = \{x_1 \mapsto 0b0000\}$ and C_1 is literal $\neg(x_1 \leq^u y)$

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
$t_1 + y \leq^u t_2 + y$	$[-t_2; -t_1[$	$[-t_1; -t_2[$	$t_1 \not\simeq t_2$
	$\emptyset$	$[0; 0[$	$t_1 \simeq t_2$
$t_1 \leq^u t_2 + y$	$[-t_2; t_1 - t_2[$	$[t_1 - t_2; -t_2[$	$t_1 \not\simeq 0$
	$\emptyset$	$[0; 0[$	$t_1 \simeq 0$
$t_1 + y \leq^u t_2$	$[t_2 - t_1 + 1; -t_1[$	$[-t_1; t_2 - t_1 + 1[$	$t_2 \not\simeq -1$
	$\emptyset$	$[0; 0[$	$t_2 \simeq -1$
$t_1 \leq^u t_2$	$[0; 0[$	$\emptyset$	$t_2 <^u t_1$
	$\emptyset$	$[0; 0[$	$t_1 \leq^u t_y2$

Example 1: $\Gamma = \{x_1 \mapsto 0b0000\}$ and C_1 is literal $\neg(x_1 \leq^u y)$
$I_1 = [0; 0[$ (full interval $\mathbb{Z}/2^4\mathbb{Z}$), with condition $x_1 \simeq 0$
We take $\mathcal{C}$ to be the condition itself $x_1 \simeq 0$.
The lemma is $\neg(x_1 \leq^u y), x_1 \simeq 0 \models \bot$

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
$t_1 + y \leq^u t_2 + y$	$[-t_2; -t_1[$	$[-t_1; -t_2[$	$t_1 \not\simeq t_2$
	$\emptyset$	$[0; 0[$	$t_1 \simeq t_2$
$t_1 \leq^u t_2 + y$	$[-t_2; t_1 - t_2[$	$[t_1 - t_2; -t_2[$	$t_1 \not\simeq 0$
	$\emptyset$	$[0; 0[$	$t_1 \simeq 0$
$t_1 + y \leq^u t_2$	$[t_2 - t_1 + 1; -t_1[$	$[-t_1; t_2 - t_1 + 1[$	$t_2 \not\simeq -1$
	$\emptyset$	$[0; 0[$	$t_2 \simeq -1$
$t_1 \leq^u t_2$	$[0; 0[$	$\emptyset$	$t_2 <^u t_1$
	$\emptyset$	$[0; 0[$	$t_1 \leq^u t_y 2$

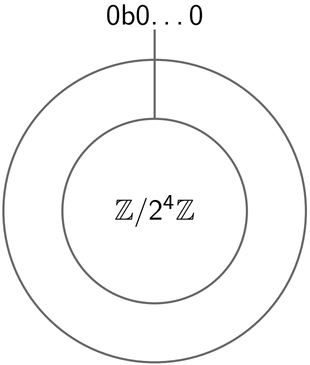
Example 2: $\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$ and

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 $(-x_2 \not\simeq x_3)$

Example 2, continued

$$\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$$

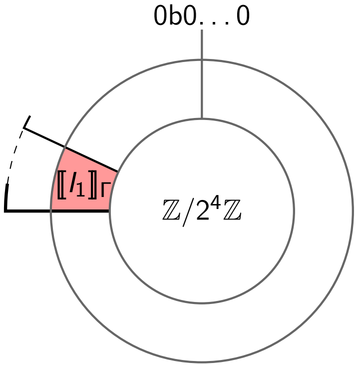
C_1 is $\neg(y \simeq x_1)$	I_1 is $[x_1; x_1 + 1[$	as $(0 \not\simeq -1)$
C_2 is $(x_1 \leq^u x_3 + y)$	I_2 is $[-x_3; x_1 - x_3[$	as $(x_1 \not\simeq 0)$
C_3 is $\neg(y - x_2 \leq^u x_3 + y)$	I_3 is $[x_2; -x_3[$	as $(-x_2 \not\simeq x_3)$



Example 2, continued

$\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$

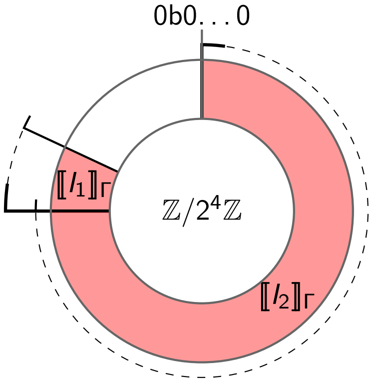
C_1 is $\neg(y \simeq x_1)$	I_1 is	$[x_1; x_1 + 1[$	as $(0 \not\simeq -1)$
C_2 is $(x_1 \leq^u x_3 + y)$	I_2 is	$[-x_3; x_1 - x_3[$	as $(x_1 \not\simeq 0)$
C_3 is $\neg(y - x_2 \leq^u x_3 + y)$	I_3 is	$[x_2; -x_3[$	as $(-x_2 \not\simeq x_3)$



Example 2, continued

$$\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$$

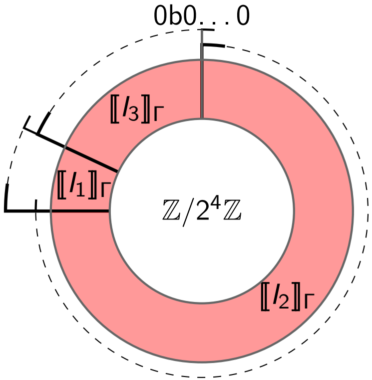
C_1 is	$\neg(y \simeq x_1)$	I_1 is	$[x_1; x_1 + 1[$	as $(0 \not\preceq -1)$
C_2 is	$(x_1 \leq^u x_3 + y)$	I_2 is	$[-x_3; x_1 - x_3[$	as $(x_1 \not\preceq 0)$
C_3 is	$\neg(y - x_2 \leq^u x_3 + y)$	I_3 is	$[x_2; -x_3[$	as $(-x_2 \not\preceq x_3)$



Example 2, continued

$\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$

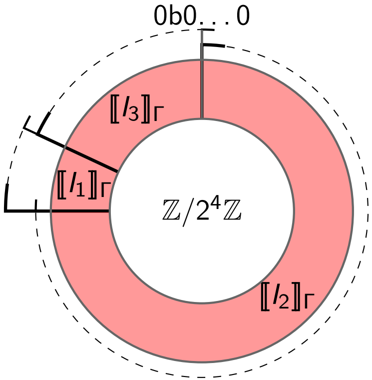
C_1 is	$\neg(y \simeq x_1)$	I_1 is	$[x_1; x_1 + 1[$	as $(0 \not\simeq -1)$
C_2 is	$(x_1 \leq^u x_3 + y)$	I_2 is	$[-x_3; x_1 - x_3[$	as $(x_1 \not\simeq 0)$
C_3 is	$\neg(y - x_2 \leq^u x_3 + y)$	I_3 is	$[x_2; -x_3[$	as $(-x_2 \not\simeq x_3)$



Example 2, continued

$\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$

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 $(-x_2 \not\simeq x_3)$



We take $\mathcal{C}$ to be $(x_1{+}1) \in I_3 \;\wedge (-x_3) \in I_2 \wedge (x_1 - x_3) \in I_1$

Things to do in practice

- 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 Γ.

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

$$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models \left(u_{\pi(i)} \in I_{\pi((i+1) \bmod q)}\right)$$

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

  $$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models \left(u_{\pi(i)} \in I_{\pi((i+1) \bmod q)}\right)$$

  In Example 2, the sequence is I_1, I_3, I_2

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

  $$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models \left(u_{\pi(i)} \in I_{\pi((i+1) \bmod q)}\right)$$

  In Example 2, the sequence is I_1, I_3, I_2
  Note: extract a *minimal* sequence, the interpolant will generalise Γ more, i.e., eliminate more models.

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

  $$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models (u_{\pi(i)} \in I_{\pi((i+1) \bmod q)})$$

  In Example 2, the sequence is I_1, I_3, I_2
  Note: extract a *minimal* sequence, the interpolant will generalise Γ more, i.e., eliminate more models.
- Express constraints "$a \in [l; u[$" in the language of linear bv-arithmetic

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

  $$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models (u_{\pi(i)} \in I_{\pi((i+1) \bmod q)})$$

  In Example 2, the sequence is I_1, I_3, I_2

  Note: extract a *minimal* sequence, the interpolant will generalise Γ more, i.e., eliminate more models.
- Express constraints "$a \in [l; u[$" in the language of linear bv-arithmetic

  Hint: do not take $(l \leq^u a <^u u)$, it is not robust to overflows
  (remember $[l; u[$ is a circular interval),
  take $a - l <^u u - l$

Things to do in practice

- 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 Γ.
  More precisely, the sequence should satisfy:

  $$\forall i \text{ with } 1 \leq i \leq q, \qquad \Gamma \models (u_{\pi(i)} \in I_{\pi((i+1) \bmod q)})$$

  In Example 2, the sequence is I_1, I_3, I_2
  Note: extract a *minimal* sequence, the interpolant will generalise Γ more, i.e., eliminate more models.
- Express constraints "$a \in [l; u[$" in the language of linear bv-arithmetic
  Hint: do not take $(l \leq^u a <^u u)$, it is not robust to overflows
  (remember $[l; u[$ is a circular interval),
  take $a{-}l <^u u{-}l$
  In Example 2, 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

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

Algorithm for extracting a covering sequence

function $\text{SEQ\_EXTRACT}(\{I_1, \ldots, I_m\}, \Gamma)$
  output $\leftarrow$ () ▷ output initialised with the empty sequence of intervals
  longest $\leftarrow$ $\text{LONGEST}(\{I_1, \ldots, I_m\}, \Gamma)$ ▷ longest interval identified
  baseline $\leftarrow$ longest.upper ▷ where to extend the coverage from
  while $[\![\text{baseline}]\!]_\Gamma \notin [\![\text{longest}]\!]_\Gamma$ **do**
    $I \leftarrow \text{FURTHEST\_EXTEND}(\text{baseline}, \{I_1, \ldots, I_m\}, \Gamma)$
    output $\leftarrow$ output, I ▷ adding I to the output sequence
    baseline $\leftarrow$ I.upper ▷ updating the baseline for the next interval pick
  if $[\![\text{baseline}]\!]_\Gamma \in [\![\text{output.first}]\!]_\Gamma$ **then**
    return output ▷ the circle is closed without the help of longest
  return output, longest ▷ longest is used to close the circle

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m{-}1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ $(m{\leq}w)$

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m-1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ $(m \leq w)$
- Why is this generalisation interesting?

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m-1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ ($m \leq w$)
- Why is this generalisation interesting?
  - because it behaves well with arithmetic operations:
    $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle$
    $(a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m-1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ ($m \leq w$)

- Why is this generalisation interesting?
  - because it behaves well with arithmetic operations:
    $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle$
    $(a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$
  - because it can be used to express *0-extensions* or *sign-extensions*, which are very often used in the SMTLib library, e.g., to compare terms t_1 and t_2 with $\leq^u$, $<^u$, $\simeq$ when t_1 and t_2 have different bitwidths

Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m-1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ ($m \leq w$)

- Why is this generalisation interesting?
  - because it behaves well with arithmetic operations:
    $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle$
    $(a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$
  - because it can be used to express *0-extensions* or *sign-extensions*, which are very often used in the SMTLib library, e.g., to compare terms t_1 and t_2 with $\leq^u$, $<^u$, $\simeq$ when t_1 and t_2 have different bitwidths

- Example?

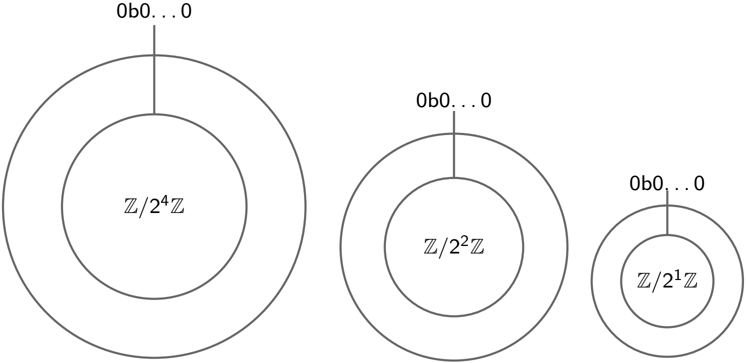
Also in the paper: generalisation with lower bits extraction

- What is lower bits extraction?
  particular forms of extraction, $t[(m-1){:}0]$, denoted here $t\langle m\rangle$
  Mathematically, it is a projection of $\mathbb{Z}/2^w\mathbb{Z}$ to $\mathbb{Z}/2^m\mathbb{Z}$ ($m \leq w$)

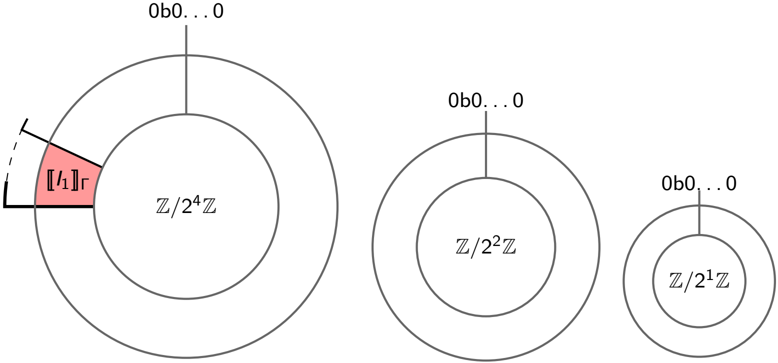
- Why is this generalisation interesting?
  - because it behaves well with arithmetic operations:
    $(a+b)\langle m\rangle \simeq a\langle m\rangle + b\langle m\rangle$
    $(a \cdot b)\langle m\rangle \simeq a\langle m\rangle \cdot b\langle m\rangle$
  - because it can be used to express *0-extensions* or *sign-extensions*, which are very often used in the SMTLib library, e.g., to compare terms t_1 and t_2 with $\leq^u$, $<^u$, $\simeq$ when t_1 and t_2 have different bitwidths

- Example? Variant of Example 2
  with model $\Gamma = \{x_1 \mapsto 0b1100, x_2 \mapsto 0b1101, x_3 \mapsto 0b0000\}$,
  and constraints

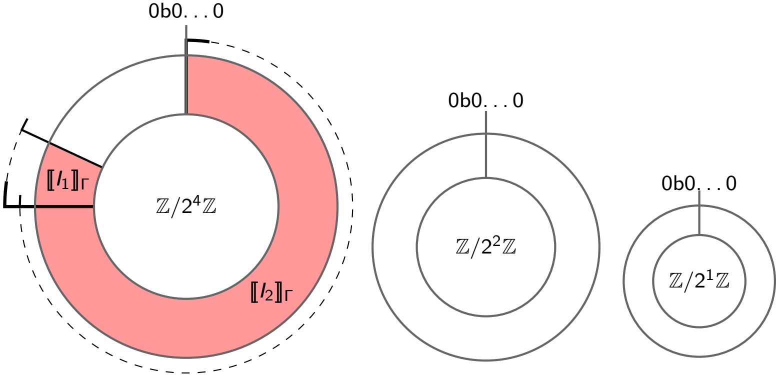
C_1	C_2	C_3	C_4
$\neg(y \simeq x_1)$	$(x_1 \leq^u x_3 + y)$	$(y\langle 2\rangle \leq^u x_2\langle 2\rangle)$	$(y\langle 1\rangle \simeq 0)$


0b0. . . 0
$\mathbb{Z}/2^4\mathbb{Z}$
0b0. . . 0
$\mathbb{Z}/2^2\mathbb{Z}$
0b0. . . 0
$\mathbb{Z}/2^1\mathbb{Z}$

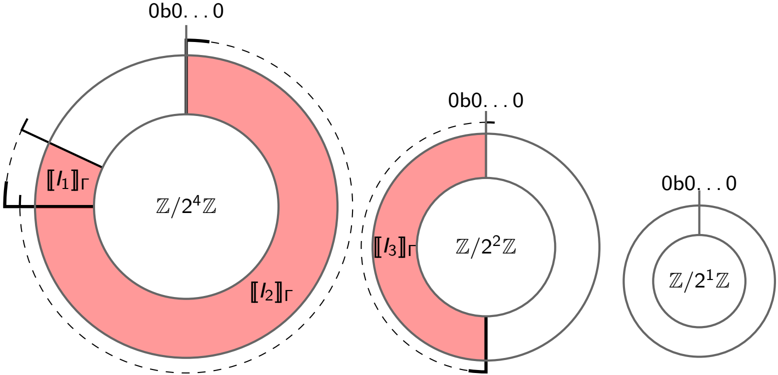
Constraint C	C_1 $\neg(y \simeq x_1)$	C_2 $(x_1 \leq^u x_3 + y)$	C_3 $(y\langle 2\rangle \leq^u x_2\langle 2\rangle)$	C_4 $(y\langle 1\rangle \simeq 0)$
Forbidden interval I_C	$[x_1; x_1 + 1[$	$[-x_3; x_1 - x_3[$	$[x_2\langle 2\rangle + 1; 0[$	$[1; 0[$
Condition c	$(0 \not\simeq -1)$	$(x_1 \not\simeq 0)$	$(x_2\langle 2\rangle \not\simeq -1)$	$(0 \not\simeq -1)$
Concrete interval $[\![I_C]\!]_\Gamma$	$[0b1100; 0b1101[$	$[0b0000; 0b1100[$	$[0b10; 0b00[$	$[0b1; 0b0[$
Bit-width w_i	$w_1 = 4$		$w_2 = 2$	$w_3 = 1$
Forbidding values for	y		$y\langle 2\rangle$	$y\langle 1\rangle$



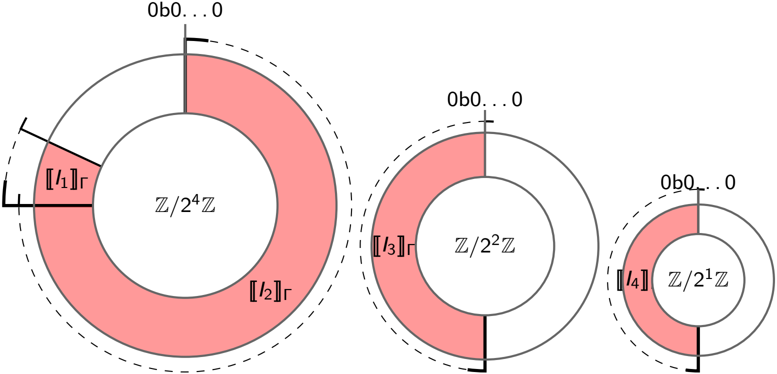
Constraint C	C_1 $\neg(y \simeq x_1)$	C_2 $(x_1 \leq^u x_3 + y)$	C_3 $(y\langle 2\rangle \leq^u x_2\langle 2\rangle)$	C_4 $(y\langle 1\rangle \simeq 0)$
Forbidden interval I_C	$[x_1; x_1+1[$	$[-x_3; x_1 - x_3[$	$[x_2\langle 2\rangle + 1; 0[$	$[1; 0[$
Condition c	$(0 \not\simeq -1)$	$(x_1 \not\simeq 0)$	$(x_2\langle 2\rangle \not\simeq -1)$	$(0 \not\simeq -1)$
Concrete interval $[\![I_C]\!]_\Gamma$	$[0b1100; 0b1101[$	$[0b0000; 0b1100[$	$[0b10; 0b00[$	$[0b1; 0b0[$
Bit-width w_i	$w_1 = 4$		$w_2 = 2$	$w_3 = 1$
Forbidding values for	y		$y\langle 2\rangle$	$y\langle 1\rangle$



Constraint C	C_1 $\neg(y \simeq x_1)$	C_2 $(x_1 \leq^u x_3 + y)$	C_3 $(y\langle 2\rangle \leq^u x_2\langle 2\rangle)$	C_4 $(y\langle 1\rangle \simeq 0)$
Forbidden interval I_C	$[x_1; x_1 + 1[$	$[-x_3; x_1 - x_3[$	$[x_2\langle 2\rangle + 1; 0[$	$[1; 0[$
Condition c	$(0 \not\simeq -1)$	$(x_1 \not\simeq 0)$	$(x_2\langle 2\rangle \not\simeq -1)$	$(0 \not\simeq -1)$
Concrete interval $[\![I_C]\!]_\Gamma$	$[0b1100; 0b1101[$	$[0b0000; 0b1100[$	$[0b10; 0b00[$	$[0b1; 0b0[$
Bit-width w_i	$w_1 = 4$		$w_2 = 2$	$w_3 = 1$
Forbidding values for	y		$y\langle 2\rangle$	$y\langle 1\rangle$



Constraint C	C_1 $\neg(y \simeq x_1)$	C_2 $(x_1 \leq^u x_3 + y)$	C_3 $(y\langle 2\rangle \leq^u x_2\langle 2\rangle)$	C_4 $(y\langle 1\rangle \simeq 0)$
Forbidden interval I_C	$[x_1; x_1 + 1[$	$[-x_3; x_1 - x_3[$	$[x_2\langle 2\rangle + 1; 0[$	$[1; 0[$
Condition c	$(0 \not\simeq -1)$	$(x_1 \not\simeq 0)$	$(x_2\langle 2\rangle \not\simeq -1)$	$(0 \not\simeq -1)$
Concrete interval $[\![I_C]\!]_\Gamma$	$[0b1100; 0b1101[$	$[0b0000; 0b1100[$	$[0b10; 0b00[$	$[0b1; 0b0[$
Bit-width w_i	$w_1 = 4$		$w_2 = 2$	$w_3 = 1$
Forbidding values for	y		$y\langle 2\rangle$	$y\langle 1\rangle$

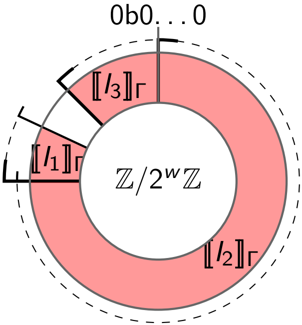


Not in this talk...

...but in (the final version of) the paper,
full interpolation procedure is described,
handling intervals of multiple bitwidths.

Not in this talk...

...but in (the final version of) the paper,
full interpolation procedure is described,
handling intervals of multiple bitwidths.
...but in the Yices implementation,
interpolation procedure handling intervals of multiple bitwidths;
we also have the optional part of an MCSAT theory, namely the propagation mechanism:



In case a single value is left uncovered, we produce term t and explanation $\mathcal{C}$ such that

$$C_1, \ldots, C_m, \mathcal{C} \models y \simeq t$$

which allows us to perform explainable propagations

Conclusion: experimentation

Still in progress, but 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 bit-blast shoots us in the foot for the rest of the run.

Conclusion: experimentation

Still in progress, but 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 bit-blast shoots us in the foot for the rest of the run.

Interpolation procedure is rather insensitive to big bitwidth.

Conclusion: experimentation

Still in progress, but 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 bit-blast shoots us in the foot for the rest of the run.

Interpolation procedure is rather insensitive to big bitwidth.

Some nice examples that we solve in less than a second (most are <0.2): the
`QF_BV/pspace/ndist.*.smt2` and the
`QF_BV/pspace/shift1add.*.smt2`
benchmarks fom the SMTLib library

Conclusion: where do we go from here?

We extend the fragment little by little:

- 0-extensions and sign-extensions are not implemented yet, but are easy to add;

Conclusion: where do we go from here?

We extend the fragment little by little:

- 0-extensions and sign-extensions are not implemented yet, but are easy to add;
- Handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable y are not necessarily in $\{-1, 0, 1\}$

Conclusion: where do we go from here?

We extend the fragment little by little:

- 0-extensions and sign-extensions are not implemented yet, but are easy to add;
- Handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable y are not necessarily in $\{-1, 0, 1\}$

  For this we must solve the question:
  if interval I forbids values for $c \cdot y$ (where c is still constant), what is the collection of intervals that are forbidden for y (taking care of overflows and divisibility of the bounds by c)?

Conclusion: where do we go from here?

We extend the fragment little by little:

- 0-extensions and sign-extensions are not implemented yet, but are easy to add;
- Handling the full fragment of linear bv-arithmetic, where coefficients of conflict variable y are not necessarily in $\{-1, 0, 1\}$

  For this we must solve the question:
  if interval I forbids values for $c \cdot y$ (where c is still constant), what is the collection of intervals that are forbidden for y (taking care of overflows and divisibility of the bounds by c)?
- Taking inspiration from other works on quantifier elimination in bitvector arithmetic [JC16].

Interpolation between two formulae

Here, specific form of interpolation, related to quantifier-elimination, serves MCSAT.
How can MCSAT, or our specific, model-driven form of interpolation can help solving the more standard form of interpolation between two formulae, remains to clarify, in connection with e.g., [Gri11].

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.

L. M. de Moura and D. Jovanovic.
A model-constructing satisfiability calculus.
In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Proc. of the 14th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, volume 7737 of *LNCS*, pages 1–12. Springer-Verlag, 2013.

S. Graham-Lengrand and D. Jovanović.
An MCSAT treatment of bit-vectors.
In M. Brain and L. Hadarean, editors, *15th Int. Work. on Satisfiability Modulo Theories (SMT 2017)*, 2017.

A. Griggio.
Effective word-level interpolation for software verification.
In P. Bjesse and A. Slobodová, editors, *International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, October 30 - November 02, 2011*, pages 28–36. FMCAD Inc., 2011

D. Jovanović, C. Barrett, and L. de Moura.
The design and implementation of the model constructing satisfiability calculus.
In *Proc. of the 13th Int. Conf. on Formal Methods In Computer-Aided Design (FMCAD'13)*. FMCAD Inc., 2013. Portland, Oregon

A. K. John and S. Chakraborty.
A layered algorithm for quantifier elimination from linear modular constraints.
Formal Methods in System Design, 49(3):272–323, 2016.

D. Jovanović and L. de Moura.
Solving non-linear arithmetic.
In B. Gramlich, D. Miller, and U. Sattler, editors, *Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR'12)*, volume 7364 of *LNCS*, pages 339–354. Springer-Verlag, 2012.

D. Jovanović.
Solving nonlinear integer arithmetic with MCSAT.
In A. Bouajjani and D. Monniaux, editors, *Proc. of the 18th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'17)*, volume 10145 of *LNCS*, pages 330–346. Springer-Verlag, 2017.

M. Janota and C. M. Wintersteiger.
On intervals and bounds in bit-vector arithmetic.
In T. King and R. Piskac, editors, *Proc. of the 14th Int. Work. on Satisfiability Modulo Theories (SMT'16)*, volume 1617 of

CEUR Workshop Proceedings, pages 81–84. CEUR-WS.org, 2016

A. Zeljic, C. M. Wintersteiger, and P. Rümmer.
Deciding bit-vector formulas with mcsat.
In N. Creignou and D. L. Berre, editors, *Proc. of the 19th Int. Conf. on Theory and Applications of Satisfiability Testing (RTA'06)*, volume 9710 of *LNCS*, pages 249–266. Springer-Verlag, 2016.